Zelflerende algoritmen voor het opstellen van biologische netwerken

Machine Learning for Biological Network Inference

Joeri Ruyssinck

Promotoren: prof. dr. ir. T. Dhaene, prof. dr. Y. Saeys
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

ISBN 978-94-6355-015-4
NUR 984, 915
Wettelijk depot: D/2017/10.500/50

UNIVERSITEIT
GENT

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Promotoren: prof. dr. ir. Tom Dhaene - Universiteit Gent
prof. dr. Yvan Saeys - Universiteit Gent

Leden examencommissie: em. prof. dr. ir. Daniël De Zutter (voorzitter)
– Universiteit Gent

prof. dr. ir. Pierre Geurts
– University of Liège

prof. dr. Celine Vens
– KU Leuven KULAK

prof. dr. ir. Sofie Van Hoecke
– Universiteit Gent

prof. dr. ir. Jan Fostier (secretaris)
– Universiteit Gent

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur

Vakgroep Informatietechnologie
iGent Toren, Technologiepark 15, B-9052 Gent, België
Tel.: +32 9 33 14900

Dankwoord

'Dit deel schrijf je typisch het laatst', wordt er steeds gezegd. Dit terwijl het een van de belangrijkste delen is, vooraan in het boek staat en vaak het eerste (en ook het enige) deel is dat jullie lezen. Begrijp dan ook dat dit contradictorisch stukje tekst perfect past als opener van een doctoraat: het diploma dat bewijst dat je zelfstandig onderzoek kan doen maar dat geen mens op zijn eentje kan behalen.

De stap naar een doctoraat kwam eerder onverwacht. Na enkele teleurstellende sollicitaties in mijn laatste masterjaar begon de twijfel wat toe te slaan. Is dit het nu? Ik had mij bijna laten strikken door de beloftes van de HR-dienst van een consultancy bedrijf in het Zaventemse. Mijn eerste dankwoord moet dan ook naar mijn vader gaan, die mij overtuigde dat er andere dingen zijn in het leven dan in eerste versnelling de ring van Brussel te verkennen. Veel van de ideeën in dit boek zijn tot stand gekomen in de kleine 15000 km fietsgependel die ik ondertussen heb afgewerkt en een goede vijf jaar later ben ik meer dan tevreden dat ik dat contract niet getekend heb en ben ik trots en dankbaar dat ik dit dankwoord kan schrijven.

Alles begon toen ik op een verdwaalde namiddag professor Laermans aansprak na de les. 'Professor, ik heb een vraag.' - 'Over dit laatste hoofdstuk?'-'Neen, ik overweeg te doctoreren, kan dat bij jullie?' Laat ons over dit voorval enkel onthouden dat ik Eric in al de jaren nadien nooit meer zo verbaasd heb gezien en ik je heel erg dankbaar ben dat ondanks mijn kort-door-de-bocht-vraag je me het e-mailadres van Tom heb bezorgd en ik zo enkele maanden later aan dit boek begon.

Martine en Marlies, bedankt omdat jullie op mijn eerste werkdag zo vroeg op het werk waren en mij vriendelijk ontvangen hebben. Die dagen was ik vaak nog rond of voor 8 uur op het werk, iets wat de nieuwere garde misschien moeilijk kan geloven. De eerste maanden op de 'techno-en-sumo' bureau waren steeds gevuld met plezier en bizarre avondactiviteiten, met als hoogtepunt het dragen van een grote pot tomatensaus rond middernacht van Elizabeth doorheen Ledeberg naar de Zuiderpoort. Bedankt voor het heerlijke eten, het plezier en de vele gesprekken in de gang: Fré, Ivo, Bram, Mathieu, Minh, Domenico, Marlies, Eric, Elizabeth, Simon, Sean, Krishnan, Selva.

Na een paar maanden werd er beslist om de 'bioinformatica-bureau' op te richten. Het was de start van ons kluizenaarsbestaan op de Zuiderpoort, waar de koffie een verdieping en een gebouw verder te vinden was. Bedankt hier ook voor de

leuke sfeer doorheen de jaren, de leuke maar eerder zeldzame conferentie samen en de vriendschap. Bedankt Prashant (Mr. Sunshine), Carolina, Ine, Jimmy, Dries, Yan, Jan, Mahdi, Pieter, Leen, Kathleen, Sergio, Giles, Assia, Dieter, Oil, Clemens. Yan, thanks for having me as your neighbor for all those years, I know it must have been hard. Jan, bedankt voor al je eerlijkheid. Lieven, bedankt voor de occasionele pint doorheen de jaren en de vele goede raad, vaak verpakt met de nodige dosis zelfspot. Ine, sorry voor al die keren dat ik uw bureau binnenwandelde en domme vragen stelde. Mahdi, my dear friend, you are a true 'zan zazil', never change. Pieter, bedankt voor zowel de serieuze als minder serieuze babbels die er vaak voor zorgden dat ik te laat thuis was. Dries, volgende week is het aan u hé! Ik gun het u van harte. Dieter, we zien elkaar te weinig en dat is echt een understatement, bedankt om bij de schaakclub te introduceren en de vele filosofische gesprekken de voorbije jaren. Leen, je verdient een meer dan extra bedankje voor je onmisbare steun doorheen de jaren en om mijn vele gezaag en flauwe humor te tolereren.

Ongeveer halverwege dit boek kwam ook het Dambi-research team aan het VIB tot stand. Alhoewel mijn contributies aan het labo met de jaren enkel afnamen waren de vrijdagnamiddagmeetings erg interessant en zijn de samenwerkingen met en via het VIB essentieel geweest voor dit werk. Bedankt Pieter, Isaac, Joris, Robrecht, Sofie, Sarah, Sophie, Wouter, Liesbet, Arne, Paco en Yvan.

Af en toe werd er in een enthousiaste bui ook nog eens aan sport gedaan na de uren. Mens sana in corpere sano. Aan de badmintonners: Arun, Mahdi, Leen, Giles, Annelies en Dries bedankt om mij te doen beseffen dat ik goed ben in badminton. Aan de badmintonners Sofie en Bram, bedankt om mij te doen beseffen dat ik er niets van kan. Aan het Lachgas-voetbalteam, bedankt voor al de keren dat we niet met 15-2 verloren zijn. Aan de lopers, bedankt om mij te overtuigen dat mijn knie ooit wel eens ging ophouden met protesteren en mij verder te pushen. Aan mijn tennisladies Lien, Liesbeth en Eugenie en trainer Guido, na een zware week was eens goed op een bal slaan vaak de ideale start om aan het weekend te beginnen.

Aan de occasionele quizteamleden wiens cultuurkennis perfect complementair is met mijn 'embarassing to know'-kennis. Bedankt Eric, Cedric en Dimitri voor ons gloriemoment op de EOS-quiz en bedankt ook Mario en Bart voor onze deelnames aan de VTK-Quiz. Verder werd er zo heel af en toe eens een IBCN of IBCN+ of IDLab quiz georganiseerd. Bedankt aan iedereen die doorheen de jaren heeft meegeholpen en bedankt voor de lunchmeetings met al de nodige humor. Met een speciale vermelding voor de core-leden doorheen de jaren Sofie, Marlies, Jelle, Leen, Giles en Eric. Op naar een geslaagde volgende editie volgende week.

Aan mijn collega's van het Reslab die de machine learning labo's en/of de NIPS-conferentie en de bijhorende vliegtuigvertragingen een pak aangenamer maakten, bedankt: Sander, Aaron, Philemon, Ira, Jeroen, Pieter-Jan, Michiel.

Laten we eerlijk zijn, zonder een sterk A-team en de adminstratieve staff die we hebben, zouden we hulpeloos verloren zijn. Het werk dat onze finances, het secretariaat en het A-team verzet om alles te laten draaien verdient een uitzonderlijke bedanking. Martine, bedankt in het bijzonder voor het occasionele gesprek over de (klein)kinderen of over zaken rond 'gezond verstand'.

'Af en toe' werd er ook nog eens aan projectwerk gedaan de laatste jaren. Dirk, niet alleen bedankt om mij dit werk te bezorgen maar ook voor onze samenwerking en jouw raad de laatste jaren. Ik heb zelden iemand zo kalm weten te zijn onder de druk van deadlines en projectproblemen, ik probeer het nog te leren. Aan de andere personen die veel en soms last-minute werk doen om samen met mij de projecten in een goede baan te leiden: bedankt Femke voor onze thesisstudenten, de 'LoS'-studies samen, het gedeelde leed tijdens de imec-overgang en je aanstekelijk enthousiasme over mijn dochter. Bedankt ook, andere-maar-even-belangrijke Femke voor je continue en recht-door-zee aanmoedigingen om mijn doctoraat af te werken. Bedankt ook Johan, Filip en Gilles voor onze samenwerking rond de data van het UZ. Bedankt Chris, Nasrin, Leen en Jelle voor onze NILM samenwerking. Bedankt o.a. TomVS voor al het werk rond slaap-apneu. Diego, voor het werk rond de actieve ventilatie. Roberto, '*grazie mille*', voor alle overige projecten, echt bedankt, zonder jouw inzet was dit doctoraat enkele maanden later af geraakt.

Aan mijn nieuwe SUMO-bureau vol enthousiaste 'jongeren' in ons sprankelend nieuw iGent gebouw. De sfeer en kruisbestuiving in het SUMO-lab piekt. Het is aangenaam om te kunnen lachen met onze groeipijnen. Ik hoop oprecht dat jullie allen binnen enkele jaren hier vooraan staan te prediken en ik hoop hierbij te kunnen helpen. Bedankt Diego, Nicolas, Ivo, Kyle, Leen, Arun, Yinghao, Joachim, Ivo en Roberto. Joachim, bedankt voor mijn meest efficiënte samenwerking in een papiersubmissie ooit en ik zal je missen als je bureau de komende maanden steeds leger zal worden. Ook al heb ik het gevoel dat we niet zo eenvoudig van je zullen afraken en dat is maar goed ook. Elias, Thijs, Thomas, Lucas, Matthias, Bert, Sander, Aza, Anna, Stijn en Andy jullie passen niet echt in een of andere bureau of voorgaande opsomming maar horen wel thuis in dit dankwoord voor jullie hulp of de occasionele babbel tijdens de lunch, net zoals de vele collegae die ik ongetwijfeld vergeten ben hier op te sommen.

Aan mijn burgievriendjes die even gek waren als mij om ook aan een doctoraat te beginnen en mij als halve-burgie hebben aanvaard de voorbije jaren. We spreken niet zoveel af, maar het is het altijd gezellig en het delen van de thesispijntjes hebben er zeker voor gezorgd dat dit boek uiteindelijk vorm kreeg. Dus ook bedankt mede-doctors: Karel, Maarten, Annelies, Jonathan, David en Sofie. Maarten, ik mis onze namiddagkoffie of in jouw geval choco meer dan je denkt en kijk van harte uit naar het moment wanneer je ons rijtje PhDs zal vervolledigen.

To my dearest Canadian friend, Pat Berger, thanks for all laughs we had throughout the years. Although an ocean separates us, you're the big brother I never had.

Aan mijn allerbeste vrienden uit het Zottegemse en omstreken: Kevin, Ann-Sofie, Frik, Niels, Saskia, Wouter, Koen en Sim. Na lang denken vraag ik mij nog steeds af waarom ik jullie zou moeten bedanken na alle kwalijke opmerkingen over mijn belastingvrije profiteursbestaan. Goede vrienden zijn echter zeldzaam en dit doctoraat heeft er vaak voor gezorgd dat ik niet eens een pint kon meepakken. Bij deze dus toch bedankt.

Aan de leden van mijn doctoraatsjury, bedankt om mijn werk zo in detail te willen doornemen, voor jullie opmerkingen in de leesverslagen en onze discussie tijdens de interne verdediging. Er kruipt heel veel tijd in en ik weet dat het voor zo'n zaken vaak de vrije tijd is die sneuvelt. Ik apprecieer het echt. Aan mijn promotoren Tom en Yvan ben ik uiteraard veel dank verschuldigd. Yvan, het was niet altijd eenvoudig de laatste jaren om mij te begeleiden aangezien ik maar zelden tijd vond om af te reizen naar het VIB of vorderingen te maken aan mijn doctoraat door ander projectwerk. Bedankt voor de vele keren dat je mij met het nodige geduld op gang hebt gezet in de beginjaren en omdat je ondanks het opzetten van je eigen groep toch steeds tijd vond voor mij. Het was steeds aangenaam om met je samen te werken. Tom, bedankt voor je vele vertrouwen in mij sinds het prille begin. Mijn onderwerp sloot minder aan bij de rest van de groep, dus het was alles behalve makkelijk voor jou om de koers samen uit te zetten maar toch is het tot een goed einde gekomen. Bedankt ook voor de zelfstandigheid die je me steeds gaf in de lesopdrachten, ik heb er steeds ten volle van genoten. Toen vorig jaar de start van Siggy iets moeilijker was dan verwacht ben ik overspoeld geweest door oprecht medeleven, steun en begrip van velen op het werk, niet het minst vanwege jou. Bedankt.

Aan mijn familie en schoonfamilie ben ik ook veel dank verschuldigd voor de steun de voorbije jaren. In het bijzonder aan mijn ouders ben ik te veel dank verschuldigd om hier te kunnen neerschrijven of om ooit te kunnen teruggeven. Vergeef het me dan ook dat ik het hier kort hou en onthoud van dit hele boek misschien enkel dat jullie mijn grootste inspiratie zijn en zullen blijven. Liefste mama, ik maak veel te weinig tijd voor jou en dat ondanks al jouw tijd die je mij hebt geschonken. Liefste papa, als er een iemand is wiens goed advies zelfs mijn koppigheid doet wankelen is het die van jou, weet dat ik het meer waardeer dan ik soms laat blijken.

Mijn allerliefste Do'tje, toen ik jou leerde kennen had ik geen idee dat in dit kleine, nogal bedeesde meisje zo'n sterke vrouw zou schuilen. In die enkele jaren zijn er niet enkel een paar doctoraten afgewerkt maar is er ook nog eens een huisje, tuintje en kindje bij gekomen. Ik weet niet waar je de energie en tijd soms vandaan haalt maar zonder jou te leren kennen zou ik nog niet de helft zijn van wie ik nu ben. Bedankt voor alles dat je doet, voor alles wie je bent en om mij na al die jaren

nog steeds niet beu te zijn. Ik hoop dat we na enkele momenten van tegenslag de voorbije jaren, die pagina's tezamen met onze boeken kunnen omslaan en opnieuw leren genieten van alles.

Als laatste is er nog een klein meisje met de allerbeste mama in de wereld die niet kan ontbreken in dit boek. Siggy, mijn lieve dochter, ik heb al meer van jou geleerd dan jij van mij ooit zal leren. Jouw dapperheid, vrolijkheid en eerlijkheid zijn wijsheden die niet in een boek kunnen neergeschreven worden. Papa ziet je graag.

Table of Contents

List of Figures

List of Tables

Samenvatting
– Summary in Dutch –

'Data is het nieuwe goud', is een recent en sterk geloof dat momenteel zowel in de industrie als in de academische wereld heerst. Traditioneel wordt het omzetten van data in bruikbare informatie of kennis als een strikt menselijke taak beschouwd. De opkomst van computers heeft er onmiskenbaar voor gezorgd dat grotere hoeveelheden en vooral ook complexere data kan verwerkt worden. Het is echter zo dat computers in dit proces nog steeds de rol opnemen van een stuk gereedschap in de handen van een intelligente gebruiker. Die dient zeer nauwlettend stap voor stap instructies te geven aan de machine om tot een bepaald eindresultaat te komen. Deze rollenverdeling beperkt ten zeerste de mogelijkheden om bepaalde problemen op te lossen in situaties waar het opstellen van een dergelijke stappenplan niet mogelijk is. Zo is het duidelijk dat vele taken die een zekere vorm van intelligentie vergen, zoals bijvoorbeeld autorijden, ons niet aangeleerd werden in de vorm van een set van instructies maar door het aanbieden van voorbeeldsituaties en oefening. Bij uitbreiding, indien we willen meer kennis uit data halen of deze gebruiken om complexere taken uit te voeren, dient er een zekere vorm van (artificiële) intelligentie toegevoegd te worden aan de machine of computer. Machinaal leren is een vakgebied dat onderzoekt hoe we machines of computers kunnen aanleren om bepaalde taken uit te voeren aan de hand van data en voorbeelden zonder dat we deze expliciet hoeven te programmeren. Recente doorbraken en succesverhalen in machinaal leren hebben voor een positieve terugkoppeling gezorgd waarbij meer data wordt verzameld en opgeslagen zonder een specifiek doel voor ogen te hebben. Deze data wordt pas nadien met behulp van algoritmes automatisch geanalyseerd op potentieel waardevolle inzichten. In deze scriptie gebruiken we machinaal leren om meer kennis te verwerven over de interactie tussen biologische entiteiten die aanwezig zijn in de cel.

Leven op aarde bestaat uit een immens complexe samenhang van verschillende sterk connecteerde systemen. Om verdere inzichten te verwerven in biologische processen is het vaak niet mogelijk om bepaalde bouwblokken geïsoleerd van hun omgeving te bestuderen maar zijn een meer globale modellering en aanpak nodig die zoveel mogelijk context in rekening brengt. Elke cel van een levend organisme functioneert op basis van de informatie die bevat zit in een genoom dat beschreven staat in DNA. Bepaalde regionen in het DNA, die we genen noemen, dienen als basis in een proces dat uiteindelijk leidt tot de productie van eiwitten die verantwoordelijk zijn voor uitvoeren van het gros van de taken in een organisme.

Aangezien elke cel hetzelfde DNA bevat zijn veranderingen in genexpressie de drijvende kracht die ervoor zorgt dat cellen zich differentiëren en kunnen reageren op externe factoren. Een van de belangrijkste systemen die cellen gebruiken om genexpressie aan te passen noemen we transcriptionele regulatie. In dit proces regelen specifieke eiwitten, genaamd transcriptiefactoren, de hoeveelheden afgeschreven DNA via aan-of juist afhechting op de promotoren van de juiste genen op het juiste moment. Binnen de biochemie werden bepaalde laboratoriumtechnieken ontwikkeld die de expressie van quasi alle actieve genen tegelijk kunnen meten op een bepaald moment in de tijd. Deze momentopnames zijn uitgegroeid tot een cruciale informatiebron binnen de systeemsbiologie en worden in de praktijk uitgevoerd door technologieën zoals microarrays of RNA-Seq. De overgrote meerderheid van dit soort genexpressiemetingen wordt gebruikt in vergelijkende studies met een specifiek doel voor ogen waarbij de relatieve veranderingen in expressie onderzocht worden in verschillende condities of op verschillende tijdstippen. De huidige transitie van hoe met data wordt omgegaan kan ook toegepast worden in het kader van genexpressiemetingen. Zo is het aannemelijk dat collecties van zulke momentopnames meer dan alleen het antwoord verbergen dan op de originele vraag waarvoor het experiment is opgesteld. Meer concreet zijn algoritmes beschreven die collecties van genexpressiemetingen als invoer nemen en trachten regulerende effecten tussen genen te ontdekken. Vaak wordt het resultaat van dit soort methodes in de vorm van netwerken gevisualiseerd. Dit soort type methodes moeten kunnen omgaan met het uitdagende kader dat het aantal metingen beperkt is ten opzichte van het aantal genen dat gemeten wordt. Deze scriptie zal technieken uit machinaal leren onderzoeken om dergelijke kennis in de vorm van netwerken uit genexpressiemetingen te extraheren.

In een eerste deel van deze scriptie stellen we een algemene structuur voor gebruik makende van technieken uit machinaal leren om netwerken van genregulatie op te stellen. In dit soort type netwerken heeft een bepaald gen A een uitgaande connectie naar een ander gen B, indien A een regulerend effect heeft op het afschrijven van gen B. Onze voorgestelde structuur is een veralgemening van het succesvolle GENIE3 algoritme dat voorstelt om het netwerkinferentieprobleem op te splitsen in verschillende regressiemodellen. In elk regressiemodel worden de expressiewaarden van ieder gen om beurt voorspeld aan de hand van de expressiewaarden van alle andere potentiële regulatoren. Deze regressieproblemen worden gemodelleerd met behulp van ensembles bestaande uit beslissingsbomen en voor elke predictor wordt ingeschat hoeveel deze bijdraagt aan het regressiemodel. Deze inschatting wordt nadien beschouwd als indicatie dat er een regulerend effect bestaat tussen dit gen en het gen dat werd gemodelleerd. We veralgemenen GENIE3 door een bemonsteringstechniek voor te stellen die elk algoritme dat een dergelijke inschatting van zijn voorspellende variabelen kan maken, transformeert naar een ensemblemethode. We tonen aan dat ensemblemethodes de sleutel tot succes zijn in het netwerkinferentieprobleem. Bijkomend stellen we ook voor om de voorspellingen van meerdere zulke algoritmen samen te nemen om tot een meer accurate voorspelling te bekomen.

In een tweede luik stellen we een nieuw algoritme, Netter, voor. Netter is

een naverwerkingsalgoritme dat voorspellingen van netwerkinferentiealgoritmen verder verfijnt door gebruik te maken van graphlets en verscheidene andere eigenschappen die netwerken kunnen beschrijven. Veelvoorkomende strategieën die netwerkinferentiealgoritmen hanteren zijn enerzijds het gebruik maken van paarsgewijze metrieken tussen genen en anderzijds het opsplitsen van het probleem in onafhankelijke regressieproblemen. Beide methoden laten na het algoritme de specifieke doelstelling mee te geven om een globaal netwerk op te stellen, waardoor de voorspelling vaak een biologisch onrealistische structuur aanneemt. Netter is een flexibel algoritme dat kan gebruikt worden tezamen met elke methode die een rangschikking van potentiële genregulatorische verbanden kan opstellen. Bovendien kan Netter zowel generieke eigenschappen gebruiken, als eenvoudig aangepast worden om specifieke domeinkennis te includeren. We verfijnen de voorspellingen van zes verschillende netwerkinferentiemethodes op basis van drie eenvoudige netwerkeigenschappen en tonen aan dat Netter in staat is om de voorspellingen te verbeteren op zowel kunstmatig gegenereerde data als op referentiedata. Tenslotte tonen we ook aan dat Netter robuust is ten opzichte van de parameterinstellingen, beter presteert dan andere naverwekingsalgoritmen en als bijkomend voordeel heeft dat het ook op voorspellingen kan toegepast worden die niet gebaseerd zijn op paarsgewijze metrieken.

Als derde contributie aan dit werk passen we ook onze ontwikkelde algoritmen toe in de praktijk. Concreet verwerken we een collectie van microarrayexperimenten die gegenereerd zijn de context van een grote internationale samenwerking die processen in immuuncellen in kaart brengt. Gebruik makend van deze data stellen we netwerken op voor verschillende celtypes. We bieden deze netwerken aan als hulpbron voor verdere analyses en hebben hiervoor een website ontwikkeld die toestaat de voorspelde interacties te visualiseren. Verder analyseren we twee specifieke delen van de opgestelde netwerken meer in detail. We tonen in een eerste deel aan dat onze opgestelde netwerken verschillende connecties tussen genen voorstellen die bekend zijn in de literatuur als betrokken bij de differentiatie en normale functie van Interleukin-17-producing helper T (TH17) cellen. In een tweede deel bespreken we genen die een rol spelen in de unfolded protein response (UPR), een uiterst geconserveerde cascade die geactiveerd wordt bij stresscondities in het endoplasmatisch reticulum. Recent is aangetoond dat deze cascade ook een belangrijke functie vervult in immuunreacties van de cel. We tonen aan dat verscheidene verbanden tussen genen in onze afgeleide netwerken ook beschreven staan in de wetenschappelijke literatuur. We bespreken de resultaten van additionele experimenten in het labo die zijn opgesteld om nieuwe interacties te valideren voorspeld door onze netwerken.

In de appendices van dit werk, bespreken we toepassingen van machinaal leren die gebruik maken van data afkomstig van patiënten opgenomen op de dienst intensieve zorgen van het UZ Gent. In een eerste studie stellen we een nieuwe methode voor die gebruikt maakt van Random Survival Forests voor om de bezettingsgraad van het aantal bedden in de dienst intensieve zorgen te voorspellen. In een tweede studie, bespreken we het potentieel van long short-term memory neurale netwerken om eerste signalen van sepsis te voorspellen.

Summary

There is a recent and strong belief in both industry and academia that data is the new gold. Traditionally, turning data into information or knowledge has been an exclusively human task. Although computers have substantially increased our capabilities to handle larger amounts and more complex data, they have always remained the tool in this process, firmly placed in the hands of the intelligent human. This paradigm severely limits the possibilities to extract information or solve problems in a setting where it is not clear how a computer should process the data. Many tasks that we consider to be results of intelligent behavior are not learned by following a clear set of instructions but by learning from examples. For example, learning how to drive a car in traffic cannot be learned by reading the instruction manual but requires practice to provide example situations to learn from. Similarly, if we wish to extract more knowledge from data or use this data to perform more complex tasks, artificial intelligence needs to become a core part of the machine or computer. Machine learning investigates how computers or machines can perform tasks by learning from data without the need to be explicitly programmed. Recent successes and advances in machine learning have caused a strong reinforcing loop in which more data is being gathered without a specific goal in mind and is subsequently 'mined for gold'. In this dissertation, we apply machine learning techniques in order to gain knowledge about relations between biological entities in the cell.

Life is a tremendously complex system consisting of a large number of interconnected actors. In order to further understand biological processes, it is often not possible to pick individual blocks and investigate these in isolation. Instead, the system should be considered, modeled and researched with as much context as possible. Inside each cell, DNA acts as a blueprint containing all necessary information to construct and maintain the organism. Certain regions of the DNA, called genes, can be expressed into proteins, which perform the vast number of functions in the body. As each cell in an organism contains the same DNA, changes in gene expression are the driving factor in cell differentiation and a key system to respond to external factors. One of the major mechanisms cells use to influence gene expression is transcriptional regulation. In this process, certain proteins called transcription factors work in a combinatorial fashion to tune the amount of produced RNA through various mechanisms.

In biochemistry, wet lab techniques have been developed which can measure genome-wide the expression of genes at a certain moment in time. These snapshots of gene activity have proven to be indispensable tools in systems biology.

A large fraction of these gene expression measurements, using techniques such as microarrays or RNA-Seq, have been performed in a context where relative changes in expression due to certain perturbations, conditions or time aspects are investigated with a specific hypothesis or goal in mind.

Similarly to the paradigm shift described earlier, we can also consider such collections of genome-wide snapshots as general data, hiding a potential wealth of knowledge unrelated to the original purpose. In particular, algorithms have been described that use collections of gene expression snapshots to deduce transcriptional or other regulating effects between genes and present these results in the form of networks. These algorithms have to work in a an extremely challenging setting, as the number of genes that are being measured by far exceeds the amount of data points that are available. This work will discuss the use of machine learning methods to infer knowledge in the form of networks from gene expression measurements.

In a first part of this dissertation, we propose a general framework using machine learning techniques to infer gene regulatory networks. In these networks, a gene A has an outgoing edge to a gene B, if gene A through its gene products causes a (direct) effect on the transcription rate of gene B. Our framework generalizes the successful method GENIE3 which decomposes the network inference task into separate regression problems. For each gene in the network the expression values of a particular target gene are predicted using all other genes as possible predictors. Next, using tree-based ensemble methods, an importance measure for each predictor gene is calculated with respect to the target gene and a high feature importance is considered as putative evidence of a regulatory link existing between both genes. We generalize GENIE3 by proposing a subsampling approach which allows any feature selection algorithm that produces a feature ranking to be cast into an ensemble feature importance algorithm. We demonstrate that the ensemble setting is key to the network inference task, as only ensemble variants achieve top performance. In addition, we explore the effect of using rankwise averaged predictions of multiple ensemble algorithms as opposed to only one. We name this approach NIMEFI (Network Inference using Multiple Ensemble Feature Importance algorithms) and show that this approach outperforms all individual methods in general, although on a specific network a single method can perform better.

In a second part of this thesis, we propose a post-processing algorithm for gene regulatory network predictions, 'Netter', which uses graphlets and several other graph-invariant properties to transform the network into a more accurate prediction. Common inference strategies of GRN inference algorithms include the calculation of local pairwise measures between genes or the transformation of the problem into independent regression subproblems to derive connections between genes. Using such schemes, the algorithm is unaware that the goal is to infer an actual network topology and the global network structure cannot influence the inference process. Netter is a flexible system which can be applied in unison with any method producing a ranking from omics data and can be tailored to specific prior knowledge by expert users or applied in general uses cases. We re-rank predictions of six different state-of-the-art algorithms using three simple network prop-

erties as optimization criteria and show that Netter can improve the predictions made on both artificially generated data as well as commonly used benchmark data. Furthermore, Netter compares favorably to other post-processing algorithms and is not restricted to correlation-like predictions. Lastly, we demonstrate that the performance increase is robust for a wide range of parameter settings.

Thirdly, we also apply our gene regulatory network inference algorithms on a practical use case. More specifically, we collected and processed a large compendium of microarrays gathered in the context of an large international immunological cell project named ImmGem. From this data, we inferred networks for eight different different celltypes and one network inferred which was derived from the entire gene expression dataset. We provide these networks to the community, including a web-interface to quickly browse the inferred connections. We zoom in on two parts of the networks and analyze them in detail. First, we show that our inferred networks unravel several known connections between genes in the context of the differentiation and inner working of Interleukin-17-producing helper T (TH17) cells. In a second use case, we discuss connections associated with the unfolded protein response, a highly conserved pathway activated by the accumulation of unfolded proteins in the endoplasmic reticulum which aims to restore normal function. Recently it has been shown that this signaling pathway also plays a vital role in immunological responses. We show that our networks are able to identify several known connections by comparing versus literature. In addition, we perform wet-lab experiments in which we activate the UPR in vitro to potentially validate novel unknown interactions between genes in this setting.

In the appendices of this work, we also briefly discuss machine learning applications using data from critically ill patients admitted to the intensive care unit (ICU) of the Ghent University Hospital. In a first study, we propose a novel method using Random Survival Forests to predict the occupancy of the ICU over time. In a second study, we discuss the potential of using long short-term memory neural networks for the early detection of sepsis.

"Computers are like Old Testament gods; lots of rules and no mercy."

— Joseph Campbell

1 Introduction

This first chapter serves to situate and motivate the conducted research. It will first introduce the key concepts and context from an historical, societal and non-technical point of view. Next, it summarizes the main challenges and contributions of this thesis and the further outline. Finally, an overview of the publications that were authored during this research period is listed.

1.1 Research context

This thesis is about *how* to let machines turn data into information and *why* that is important.

This might seem like an overly compressed summary of this book. However, bear with me for a couple of paragraphs before things become technical and stay this way. Let me first define the concept of information. Information is anything that answers one or more questions. Without the proper questions, just like an answer, information cannot exist. Now, what is data? Data is anything which could be interpreted or transformed to become information. If you are reading this sentence, you are transforming the sequence of characters on this page, the data, into information. At this very moment, this information is answering the question of what you will find in this book. Traditionally, information has always been considered as one of the most valuable assets in industry and life, yet also

the most intangible. Data however, as per definition, is useless if no information can be extracted from it. Consider the following example: the well-guarded secret recipe of Coca-Cola, empowering the belief that the original drink is superior to all competitors. This piece of information, even though it might not even exist, has been vital in promoting the drink and its billion dollar success. On the contrary, 'data' related to the piece of information, namely the resulting drink, is sold for cheap in every shop around the world. A way, or in this case, the lack of a way to go from the data towards the information, is creating a substantial economic value.

More related to this work, about twenty-five years ago the World Wide Web was created by Tim Berners-Lee and Robert Cailliau. Back then, it was an efficient information space. In a rapid fashion more websites started to appear, eventually reaching over a billion different websites in 2014. The World Web Web rapidly transformed from an information space to a data space. If you do not know which address to type or where to look, no information can be extracted from the data. Google started approximately 20 years ago from a small research project of two students and now became the conglomerate AlphaBet Inc. which reported a revenue of over 90 billion US dollar in 2016. A large part of this success was only possible due to the very foundations which started this all: a search index turning data into information and most importantly presented in a simple way. I hope these examples were able to convince you that information is vital, data is useless and as a result, extracting information from data means generating value.

1.1.1 Machine learning

Despite this economic potential, traditionally, turning data into information has always been considered a strictly human task. Since the invention of writing, data was generated in such a way that humans could transform it, often in a labor intensive fashion. In addition, children and adults have to be taught in schools for several decades in order to master this process.

With the advent of computers, humans were able to process more and different types of data. However the information extraction paradigm did not change. Humans are merely using the computers, giving explicit instructions what to do. All information flowing in and out has be tailored to the human brain. Imagine two powerful reasoning systems, the human brain and a computer, forced to bottleneck in communication by a system which does not come naturally to either of them. It is no surprise this comes at a huge cost: the existence of problems which can be solved by humans but cannot be solved by computers. Often we describe such problems as things that require 'intelligence'. In the last years it has however become more clear that these limitations are often byproducts of the linguistic bottleneck of having to program a computer and tell it how to solve the problem.

A recent example of a computer program which was created (partially) by circumventing this bottleneck is AlphaGo [1]. It specializes in playing the board game Go which was long considered to be safe from computer domination as the amount of possible options presented to players exceeds the number of visible atoms in the universe. While (early) chess computers could still win from humans by efficiently searching and evaluating every option by relying on rapid calculations, this is much less the case in a game of Go. Lee Sobol, 18-time world champion, was surprisingly defeated 4-1 in a match in 2016 by AlphaGo. The computer program generated the knowledge to play by 'learning' from a database of 30 million moves made in actual matches of Go between humans players. Next, it continued to 'learn' by playing additional matches versus variants of itself. Creating information (in this case: how to play and win at Go) by explicitly giving data in the form of examples is called *machine learning*. Machine learning evolved and is closely related to artificial intelligence because the machine or computer seemingly intelligently learns to perform the requested task without explicit instructions. Therefore, the term learning is often used as a substitute for constructing or modeling. A more traditional definition of machine learning would then be the field which explores algorithms that can learn from data without the need to be explicitly programmed.

Let us consider a much more simple problem which machine learning can solve: programming a computer to turn handwritten digits into the information they present: the digit [2]. Around 500 million traditional letters and packages are sent in the U.S.A. on an average day. In a first step, the modal addressee of such a letter is not identified by a bar code but by a traditional (handwritten) piece of data called the postal code. Therefore, if we wish to avoid having people reading millions of postal codes and sorting the letters in the right box at the postal office every day (which we really should), we need to program a computer how to read handwritten digits. It turns out this is not an easy task to do.

Have a look at the following picture (Figure 1.1) and ask yourself the question: how would you describe the digit 1 to an intelligent person who is not familiar with Arabic digits. A small line going diagonally from the middle to the top right corner, followed by a longer straight line going from top to bottom? It turns out that terms like 'small' and 'longer' are complicated concepts in the language which we speak to computers. In addition, digits might appear slightly rotated on the letter. Some people might not bother to put the first small line and still it would be considered a 1, which is not the case if you omit the second line. Furthermore, if you place the first line in a more horizontal way and the second slightly rotated, the digit suddenly becomes a 7. To avoid confusion some people like to add an extra small line crossing the big line in the center, but others do not. This arbitrary and complicated set of rules to interpret digits seems to contradict with the fact that we consider it to be a simple task. This contradiction is solved by realizing that in

reality digits are not taught this way, but instead we would show the person how every digit looks like and for each example we provide we tell the person: this is a 0, this is a 1, etc. As the person sees more and more examples, he will eventually know how an average digit looks like, which variations are tolerated and ironically, create a new variant which we refer to as the person's handwriting. Similarly, with machine learning we show the algorithm many pictures of handwritten digits and add a label representing the digit to each of them. After the learning phase, the computer program could identify new pictures or generate new examples of handwritten digits if requested.

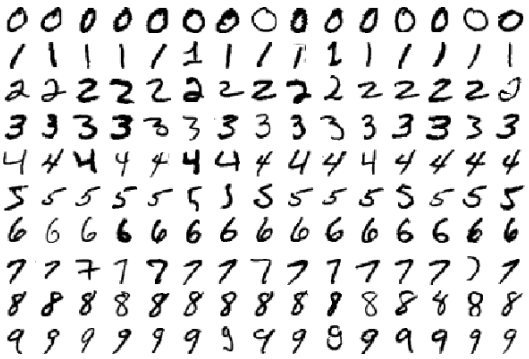

Figure 1.1: **Examples of handwritten digits in the MNIST dataset. [2]**

The true potential of machine learning techniques becomes clear when they are applied to problems for which no human solution is known or possible. In this setting, they can automatically mine insights from data. Indeed, the same concept of learning from provided data can equally be applied to problems for which the connections in the data are unknown. Three categories of machine learning techniques are usually defined. *Supervised learning* concerns algorithms which can automatically learn to map specific inputs to a desired output label. In case of a continuous output value, this is called regression modeling. In classification problems, the labels are categorical. In both settings, the data examples to learn from need to contain labels. *Unsupervised learning* are machine learning algorithms which can find structure and connections in the data in case no labels are available. A well known example are clustering techniques. *Reinforcement learning* algorithms aim to learn a computer program which can interact with the environment to perform a certain goal by providing rewards or by punishing the program while it explores solutions.

1.1.2 The Data Science age

Pioneering machine learning techniques already appeared in the early 50s but especially during the past five years concepts like machine learning and data science have seen a rapid gain in interest in both industry and in the academic world. To illustrate, the course machine learning at this faculty was only organized for the first time in 2010. The Neural Information Processing Systems (NIPS) conference, the largest machine learning conference in the world, hosted around 1200 people in the 2012 edition, but close to 6000 attended in 2016. What is causing this sudden and substantial increase in interest for machine learning and artificial intelligence?

A first and the most important reason is the evolution of the 'machine' in machine learning. Traditional algorithms were designed to run as a single core program on a traditional computer, but this is no longer the case. For example, high volume storage clusters are now available giving way to new machine learning frameworks and algorithms which specialize in extracting information from extremely large data sets. This gave way to the emergent field of 'Big Data' analysis [3]. Machine learning frameworks such as Mahout [4] specifically extend technologies such as the Hadoop ecosystem [5] to perform machine learning analysis on such data. Similarly, the use of Graphics Processing Units (GPUs) have revolutionized how (deep) neural network architectures can be constructed and trained. The field Deep Learning [6] is arguably the key factor in the current machine learning hype. By adding multiple hidden layers to the neural network in a well chosen way, the neural network can extract and learn discriminative features from the data to be used later in the classifier. In contrast, traditional techniques require a feature engineering step in which the domain expert creates this feature set which the classifier then learns to combine in order to make predictions. Deep Learning has crushed all records and competition in tasks such as image recognition during the past years and is considered to be one of the greatest breakthroughs in machine learning. GPUs specifically designed for scientific computing already exist and in 2016 Google announced the Tensor Processing Unit, a computational chip specifically designed for machine learning.

A second reason is the exponential increase of data created and stored in almost all industries and scientific fields. The existence of privacy ends with the purchase of a smartphone. Social media interactions and content, GPS locations, everything is monitored and stored. In manufacturing and large industrial undertakings, machines such as wind turbines are being equipped with cheap sensors that continuously log every vibration, temperature change and other operational parameters. The Internet of Things [7], in which everyday objects are being equipped with sensors and connected to the Internet to create a Smart Home, Smart City, etc. is also creating a growing amount of residential logged data. In medicine, digital patient records, lab results tests and bed side monitors are digitally stored in local databases or in global initiatives creating data lakes, often anonymized. In physics,

particle accelerators such as the famous Large Hadron Collidor built by CERN are creating vast amounts of data at each collision. In biomedicine and biochemistry, high throughput experiments try to capture characteristics at a cell and genome-wide level. In many of the settings described above, the actual decision to log and store data is taken before it is decided what and how the data should be processed into information. The world is convinced that data is the new gold.

This bring us to the third reason why data science is booming: perception, media coverage and in popular phrasing: the bandwagon effect. IBM's Watson [8] was one of the first artificial intelligences to generate a lot of attention in the past years by winning a special episode of the popular show 'Jeopardy!'. Furthermore, it later came with the statement that IBM would invest a billion dollars in the Watson project. Google acquired DeepMind, a British artificial intelligence company, only four years after being founded at a price of 500 million dollars. If major players in the field are giving such strong signals, the rest will follow. This recent boom comes with a lot of opportunities (the chance to write this book being one of my favorites) but also comes with major risks. The greatest risk being expertise dilution. Many companies are interested in the gold rush but often have few knowledge and experience about what is possible. Therefore, the goals are unrealistically set and the infrastructure and budgets come pre-allocated before any technical feedback is acquired. Aggravating the situation, the people executing those projects are often new and rushed to the field and may not be familiar with the very basics of machine learning. This situation is dangerous to the core field of machine learning, as many companies and projects leaders will undoubtedly become disappointed in this process, quickly turning the positive spiral in a negative one. This situation is quite comparable to what happened to the field of information theory in the sixties and the paper by Claude Shannon: the bandwagon [9], brings up very similar concerns to those that have been voiced by machine learning experts in the past years.

Do machine learning and data science deserve the current hype? Yes. The progress and achievements in the last years certainly warrant the current interest. However, we should be wary of the consequences and raise the level of criticism if we wish to continue the current success. The most important concept is to stick to the basics. Information can only be extracted if it is in your data to begin with. Setting goals upfront to extract a certain amount or specific information is pointless in machine learning. Furthermore, frameworks such as TensorFlow or even scikit-learn which automate and hide many of the complex tasks in machine learning come with the associated risk that buttons are being pushed without knowing the underlying meaning. More often than not, the performance of the built model in terms of accuracy or performance answers less questions than the knowledge you gained trying to build the model. Predictions are often useless, but being able to predict is priceless.

1.1.3 Biology primer

This work will focus on the principles outlined in the previous subsection and more specifically use machine learning to tackle challenges in the field of systems biology, bioinformatics and immunology. In the appendices attached to this work, we also discuss two cases of applying machine learning on healthcare cases. As they are not part of the core research presented in this work we will however not discuss them further in this introduction. The remainder of this subsection serves to briefly introduce the biological concepts and technologies related to this work.

Life is tremendously complex and diverse, yet all living things on Earth have a genome which is a blueprint that contains all necessary information to construct and maintain that specific organism. The genome is encoded in DNA and is present in each cell of the organism. DNA can be considered as a genetic code consisting of a series of four different bases. These four different bases can be abbreviated and represented as a string sequence constructed from the characters A (adenine), T (thymine), C (cytosine) and G (guanine). A gene is defined as a region in the DNA that is responsible for one or more tasks in the organism and as such can be considered as a basic hereditary unit. The central dogma of molecular biology [10] states *i.a.* that the process in which DNA is converted to functional products is called gene expression and occurs in two key stages: transcription and translation. Transcription is the process which creates copies of the DNA in form of messenger RNA (mRNA). This RNA sequence is then further processed and translated into a protein. Proteins are macromolecules which perform a vast number of functions within the organism. As each cell in an organism contains the same genome in the form of DNA, changes in gene expression (which genes are being transcribed and at which rate) are the driving factor in cell differentiation and the key system to respond to external factors.

1.1.3.1 Gene expression measurements

Systems biology is a scientific field which tries to gain understanding of complex biological systems by building computational models which can explain the behavior of such systems. The key concept in systems biology is that the whole is bigger than the sum of its parts and therefore that complex systems should not be approached in a reductionist way of thinking. Instead, processes should be considered with as much context as possible in multi-level models at a high resolution. In this setting, high throughput techniques which provide data and insights at a genome-wide scale are vital information sources. A prime example are gene expression measurements which provide a snapshot of the 'activity' of each gene in a cell at a certain moment in time. With the activity of a gene, we mean a qualitative measure of how many copies of a certain gene (mRNA) are being being transcribed and therefore are present in the cell. Two key technologies exist which can pro-

vide such data: microarrays and RNA-Seq. A microarray contains a large number of spots which each contain many copies of a certain DNA-sequence called probes. Probes are specifically chosen to be complementary with known parts of the mRNA of a gene. The microarray is spread with extracted mRNA combined with fluorescent tags and is allowed to bind with the probeset. After some time, the unbound remaining mRNA is washed away and each spot is scanned with a laser. The luminance of a certain spot is directly related to amount of mRNA with fluorescent tags that was bound to that specific probe and therefore to the number of copies of the corresponding gene which were present in the sample. RNA-Seq is a more recent technique which is replacing microarray technology as the standard in most labs around the world. RNA-Seq is based on next-generaton sequencing techniques and holds several advantages over microarrays. First, there is no need to design specific probes up front, which means RNA-Seq can be used to detect unknown sequences and be used in species with no reference genome. Second, this lack of the need of probes also eliminates any noise generated from incomplete or unwanted bindings between a certain mRNA. Lastly, RNA-Seq provides a more quantitative measure, the number of reads. Despite these advantages, microarrays technology is still being used side-to-side to RNA-Seq because it is relatively cheap and often suffices to answer the scientific or medical question that is being investigated. In this work, we will focus on analyzing data created by microarray technology. The main reasons for this are the existence of benchmark data and large repositories of microarrays data related to a specific use case we discuss. However, all methods described can also be applied to RNA-Seq data.

1.1.3.2 Network representations

In popular media, genes are often linked one-to-one to a certain disease or trait (for example, the gene responsible for diabetes or hair color). In reality however, the large majority of genes work in unison with a set of other genes to realize a certain function. These interactions come in the form of different types: protein-protein, transcriptional (protein-DNA), signaling (phosphorylation), etc. and can be structured in complex networks, cascades and pathways. Network representations serve as ideal and flexible abstractions of relations between biological entities. For the human user, networks have proven to be ideal representation methods to structure and visualize large amounts of data. From an algorithmic point of view, networks can be further analyzed and described using concepts from network theory such as connectivity, closeness, etc. Networks can also serve as a whole as input for further inference algorithms. For the above mentioned reasons, networks are chosen to be the end-point of the data to information process in this work.

With this, we have created the context and motivation of all words present in the title of this work and will now continue to discuss the challenges and contributions related to this dissertation.

1.1.4 Challenges

All information is contained in the DNA and flows in a forward and complex way until eventually certain phenotypes such a trait (e.g., hair colour) or a disease (e.g., cancer) can be observed. However, in reality we are most often concerned in solving the (partially) inverse problem: e.g. given a certain cancer, what is the cause, at which level does it occur and how can be prevent or treat? Following this inverse flow of information means diving in a giant haystack of interacting systems operating at different levels, continuously changing in time and acting in a non-deterministic way. Traditional wet-lab experiments such as qPCR or a western blot are vital techniques but can only be used in a setting in which one already knows what should be measured or investigated and cannot answer the question of where to look. In contrast, high-throughput techniques such as flow cytometry experiments or gene expression measurements offer information at a higher level but come with various downsides. One of the major drawbacks is that these type of technologies produce such vast amounts of data that they no longer can be interpreted by the human brain. The design of statistical methods and specific algorithms which can extract as much information as possible from this data is the key challenge that computational systems biology is solving and also the main challenge in this dissertation.

More specifically, one of the long standing challenges in computational systems biology is how gene regulatory interactions can be derived from snapshots of gene expression measurements in the form of networks. Many of such algorithms have been proposed which strike different balances between the level of detail they wish to infer and the size of the system that is under consideration. A classification can be made in two categories of algorithms. The first type aim to emulate a natural system with respect to its dynamical activity. Here, the components have biological interpretations and the conclusions are drawn from the generative nature of the model. The second category of methods approaches the problem model-free and aims to draw conclusions about biochemical interactions between genes or gene products without requiring plausible biological mechanisms. A typical approach of such methods is to calculate pair-wise measures between each pair of genes that represent co-variation. Several measures have been suggested including Pearson correlation, mutual information [11–15] or more recently (feature) importance scores derived from training machine learning models [16–18]. Several challenges exist, which explains why this domain has spawned such a plethora of different methods over time.

A first series of challenges are directly related to properties of the data. Microarray data is noisy, the number of samples is limited due to the cost to perform a single experiment and only offers insights at a gene transcriptional level although many other mechanisms (e.g. phosphorylation, miRNA, etc.) exist to achieve gene regulation. The number of potential regulatory interactions also by far exceeds the

available measurements. Therefore, if one wishes to tackle the problem using machine learning, one has to operate in an extreme imbalance between the number of features available and the number of data samples. Furthermore it is not clear how one can distinguish between regulatory effects that are direct or indirect (e.g. the expression of gene A influences the expression of gene C through a gene B) unless specific time-series or knockout-data is available. In practice, many popular methods propose heuristics to automatically lower the number of indirect interactions present in the network.

A second series of challenges is related directly to the development of the algorithms and the integration with other data sources. For example, often prior knowledge is available in the form of already known interactions or in more abstract forms such as known topological properties of the network. It is not clear how such versatile properties can directly be included in the prediction process without losing general applicability. Related, other experimental data can be available, e.g. ChIP-seq experiments that offer insights on DNA binding sites. Challenges there include how and at which stage of the prediction such information can be included to make predictions more accurate.

A third series of challenges revolves around the problem of how the performance of gene regulatory network inference methods can be evaluated in general and on a specific application. For the first question two options exist: the use of synthetic data or the use of real benchmark data in comparative studies. The latter suffers from the fact that no complete ground truth is available to compare. Therefore, the evaluation of the network can be biased towards areas of the network which are well studied and known. The former suffers from the fact that synthetic data is an abstraction of reality and that the method becomes tuned to the specific simulator which generated the data. Large scale evaluations in the form of blinded competitions have taken place which have produced key insights about the performance of methods and how better results can be obtained. However, they do not answer the first question of what the quality of prediction of a method will be if is it applied on a specific data set or how it should be evaluated. Comparison with literature can give insights on the quality of the prediction but is challenging to perform automatically and unbiased.

The most important challenge however remains how one can use such methods in practice to generate real scientific value. Gene regulatory network inference methods are per definition hypothesis generating tools and do not offer any value if they are not accompanied by a second step which can confirm these hypotheses. Most often the second step comes in the form of more traditional 'low-throughput' wet-lab experiments using knock-out organisms or specific inhibitors which can provide detailed evidence of interactions. There are several scenarios in which these two steps can come together. In first scenario, the algorithm developer also performs the wet-lab validation step and has the necessary infrastructure, exper-

tise, funds and knowledge to both perform the experiments and interpret the results. This is a very rare situation which does not occur often in practice. In a second scenario, the algorithm developer is able to introduce his method to the non-computational community, convince labs that it is a useful technique and most challenging make it so that the algorithm can be applied in a user-friendly way without requiring specific knowledge. In a third scenario, a specific lab has a mix of both computational developers and wet-lab scientists. In this setting, the challenge is find an appropriate use-case in which the technique can be of value and to find a common language to speak in order to set-up the experiment and interpret the results.

1.2 Thesis contributions and outline

The three chapters that follow this introduction present research in the context that has been presented and address one or more challenges in the field that were listed.

Chapter two presents a novel framework which uses machine learning techniques to infer gene regulatory networks from gene expression data. It addresses the first series of challenges related to how we can develop methods that can deal with specific properties of the data. To be more specific, we generalize a successful method, named GENIE3 [16], by proposing a subsampling approach which allows any feature selection algorithm that produces a feature ranking to be cast into an ensemble feature importance algorithm. We demonstrate that the ensemble setting is key to the network inference task. In addition, we explore the effect of using rankwise averaged predictions of multiple ensemble algorithms as opposed to only one. We name this approach NIMEFI (Network Inference using Multiple Ensemble Feature Importance algorithms) and show that this approach outperforms all individual methods in general.

Chapter three presents a post-processing algorithm for gene regulatory network predictions, Netter, which uses graphlets and several other graph-invariant properties to transform the network into a more accurate prediction. It addresses the second series of challenges related to how prior knowledge can be integrated in the prediction task. Netter is a flexible system which can be applied in unison with any inference method. We re-rank predictions of six different algorithms using three simple network properties as optimization criteria and show that Netter can improve the predictions made on both artificially generated data as well as commonly used benchmark data. Furthermore, we show that Netter compares favorably to other post-processing algorithms.

Chapter four addresses challenges presented in the last two categories. In this section we apply our gene regulatory network inference algorithms on a practical use case. More specifically, we collected and processed a large compendium of microarrays gathered in the context of a large international immunological cell

project named ImmGem [19]. We provide these networks to the community, including a web-interface to quickly browse the inferred connections. In one of two use cases we present, we discuss results of wet-lab experiments which were performed in the context of the inferred networks.

Chapter five concludes this thesis.

Finally, in the appendices of this work we present machine learning applications using data from critically ill patients admitted to the intensive care unit (ICU) of the Ghent University Hospital.

This dissertation is composed of a number of publications that were realized within the scope of this PhD. The remainder of this introduction provides an overview of the papers which were published during this research period.

1.3 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my PhD research.

1.3.1 A1 publications (listed in the Science Citation Index)

1. **Joeri Ruyssinck**, Huynh-Thu Vân Anh, Pierre Geurts, Tom Dhaene, Piet Demeester and Yvan Saeys. *NIMEFI: gene regulatory network inference using multiple ensemble feature importance algorithms.* Published in PLOS One, 9(3), 2014.

2. **Joeri Ruyssinck**, Piet Demeester, Tom Dhaene and Yvan Saeys. *Netter: reranking gene network inference predictions using structural network properties*. Published in BMC Bioinformatics, 1-8,2016.

3. **Joeri Ruyssinck***, Joachim van der Herten* , Rein Houthooft, Femke Ongenae , Ivo Couckuyt , Bram Gadeyne, Kirsten Colpaert , Johan Decruyenaere, Filip De Turck and Tom Dhaene *Random survival forests for predicting the bed occupancy in the intensive care unit*. Published in Computational and Mathematical Methods in Medicine, (*Contributed equally) 17(76),2016.

4. Leen De Baets*, **Joeri Ruyssinck***, Johan Decruyenaere, Filip De Turck, Femke Ongenae and Tom Dhaene. *Early detection of sepsis through the prediction of positive blood cultures using long short-term memory neural networks*. Submitted to Artificial Intelligence in Medicine, (*Contributed equally)

5. **Joeri Ruyssinck**, Lana Vandersarren, Jeroen Creytens , Karl Vergote, Sophie Janssens , Tom Dhaene and Yvan Saeys. *Large-scale network inference*

of the Immunological Genome Project gene expression data with applications in the unfolded protein response. Submitted to PLOS Computational Biology,

6. Rein Houthooft, **Joeri Ruyssinck**, Joachim van der Herten, Sean Stijven, Ivo Couckuyt, Bram Gadeyne, Femke Ongenae, Kirsten Colpaert, Johan Decruyenaere, Tom Dhaene and Filip De Turck. *Predictive modelling of survival and length of stay in critically ill patients using sequential organ failure scores*. Published in Artificial Intelligence in Medicine, 63(3). p.191-207, 2015

7. Leen De Baets, **Joeri Ruyssinck** , Chris Develder, Tom Dhaene and Dirk Deschrijver. *On the Bayesian optimization and robustness of event-detection methods in NILM*. Accepted in Energy and Buildings, 2017

8. Leen De Baets, **Joeri Ruyssinck**, Chris Develder, Tom Dhaene and Dirk Deschrijver. *Appliance classification using VI trajectories and convolutional neural networks*. Submitted to Electronics Letters,

9. Nasrin Sadeghianpourhamami, **Joeri Ruyssinck** , Dirk Deschrijver, Tom Dhaene and Chris Develder. *Comprehensive feature selection for appliance classification in NILM*. Submitted to Energy and Buildings,

1.3.2 Contributions to international conferences

1. **Joeri Ruyssinck**, Huynh-Thu Vân Anh, Pierre Geurts, Tom Dhaene, Piet Demeester and Yvan Saeys. *Inferring gene regulatory networks using ensembles of feature selection techniques* Program of the sixth international workshop on machine learning in systems biology (MLSB), 2012. (talk+poster)

2. **Joeri Ruyssinck**, Huynh-Thu Vân Anh, Pierre Geurts, Tom Dhaene, Piet Demeester and Yvan Saeys. *Inferring gene regulatory network topologies using ensembles of feature selection techniques* Proceedings of the 21st Belgian-Dutch conference on machine learning, 2012. (talk)

3. **Joeri Ruyssinck**, Tom Dhaene and Yvan Saeys. *A re-ranking algorithm for gene regulatory network predictions using graphlets and graph-invariant properties* Intelligent Systems for Molecular Biology, 21st Annual international conference, Abstracts, 2013. (poster)

4. **Joeri Ruyssinck**, Tom Dhaene, Femke Ongenae, Filip De Turck, Kirsten Colpaert and Johan Decruyenaere. *Predicting the Intensive Care Unit Bed Occupancy with Survival Random Forests* NIPS Workshop on Machine Learning for Clinical Data Analysis, 2014. (poster)

5. Robrecht Cannoodt, **Joeri Ruyssinck**, Katleen De Preter, Tom Dhaene and Yvan Saeys. *Network inference by integrating biclustering and feature selection* BeNeLux Bioinformatics Conference, 2013. (talk+poster)

6. Sofie Van Gassen, **Joeri Ruyssinck**,Yvan Saeys and Tom Dhaene. *Stable feature selection techniques for microarray data.*, BeNeLux Bioinformatics Conference, 2013, (talk)

7. Leen De Baets, **Joeri Ruyssinck**, Dirk Deschrijver and Tom Dhaene. *Event detection in NILM using Cepstrum smoothing*. 3rd International Workshop on Non-Intrusive Load Monitoring, 2014, (best poster award)

8. Leen De Baets, **Joeri Ruyssinck**, Dirk Deschrijver and Tom Dhaene. *Cepstrum analysis applied on event detection in NILM*. 25th Belgian-Dutch Conference on Machine Learning, 2016, (poster)

9. Thomas Peiffer, **Joeri Ruyssinck**, Johan Decruyenaere, Filip De Turck, Femke Ongenae and Tom Dhaene. *Early detection of positive blood cultures using recurrent neural networks on time series data*. 25th Belgian-Dutch Conference on Machine Learning, 2016, (poster)

10. Leen De Baets, **Joeri Ruyssinck**, Thomas Peiffer, Johan Decruyenaere, Filip De Turck, Femke Ongenae and Tom Dhaene. *Positive blood culture detection in time series data using a BiLSTM network*. NIPS Workshop on Machine Learning for Clinical Data Analysis, 2016. (poster)

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. *Mastering the game of Go with deep neural networks and tree search*. Nature, 529(7587):484–489, 2016.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[3] V. Marx. *Biology: The big challenges of big data*. Nature, 498(7453):255–260, 2013.

[4] Apache Software Foundation. *Apache Mahout:: Scalable machine-learning and data-mining library* [online]. Available from: http://mahout.apache.org [cited 2017-03-12].

[5] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. *The hadoop distributed file system*. In Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on, pages 1–10. IEEE, 2010.

[6] Y. LeCun, Y. Bengio, and G. Hinton. *Deep learning*. Nature, 521(7553):436–444, 2015.

[7] L. Atzori, A. Iera, and G. Morabito. *The internet of things: A survey*. Computer networks, 54(15):2787–2805, 2010.

[8] D. A. Ferrucci. *Introduction to this is watson*. IBM Journal of Research and Development, 56(3.4):1–1, 2012.

[9] C. E. Shannon. *The Bandwagon*. IEEE Transactions Information Theory, 2:3, 1956.

[10] F. Crick et al. *Central dogma of molecular biology*. Nature, 227(5258):561–563, 1970.

[11] A. J. Butte and I. S. Kohane. *Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements*. Pac Symp Biocomput, 5, 2000.

[12] R. de Matos Simoes and F. Emmert-Streib. *Bagging statistical network inference from large-scale gene expression data*. PLoS ONE, 7, 2012.

[13] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, and R. D. Favera. *ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context*. BMC Bioinforma, 7, 2006.

[14] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, and G. Cottarel. *Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles*. PLoS Biol, 5, 2007.

[15] P. E. Meyer, K. Kontos, F. Lafitte, and G. Bontempi. *Information-theoretic inference of large transcriptional regulatory networks*. EURASIP journal on bioinformatics and systems biology, 2007(1):1–9, 2007.

[16] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. *Inferring Regulatory Networks from Expression Data Using Tree-Based Methods*. PLoS ONE, 5, 2010.

[17] J. Ruyssinck, V. A. Huynh-Thu, P. Geurts, T. Dhaene, P. Demeester, and Y. Saeys. *Nimefi: Gene regulatory network inference using multiple ensemble feature importance algorithms*. PLoS ONE, 9, 2014.

[18] A. C. Haury, F. Mordelet, P. Vera-Licona, and J. P. Vert. *TIGRESS: Trustful Inference of Gene REgulation using Stability Selection*. BMC Syst Biol, 6, 2012.

[19] T. S. Heng, M. W. Painter, K. Elpek, V. Lukacs-Kornek, N. Mauermann, S. J. Turley, D. Koller, F. S. Kim, A. J. Wagers, N. Asinovski, et al. *The Immunological Genome Project: networks of gene expression in immune cells*. Nature immunology, 9(10):1091–1094, 2008.

"He uses statistics as a drunken man uses lamp posts; for support rather than illumination."

— Andrew Lang

2

NIMEFI: Gene Regulatory Network Inference using Multiple Ensemble Feature Importance Algorithms

In this chapter, we propose an algorithm to infer gene regulatory networks from a compendium of gene expression measurements. NIMEFI (Gene Regulatory Network Inference using Multiple Ensemble Feature Importance Algorithms) discovers connections between genes in the network on a large-scale by transforming the network inference task into independent regression problems. Two genes in the network are considered to be connected by an edge if the expression measures of one gene are informative to predict the expression values of the other gene. NIMEFI will form the basis of all further research described in this book.

⋆ ⋆ ⋆

Joeri Ruyssinck, Vân Anh Huynh-Thu, Pierre Geurts, Tom Dhaene, Piet Demeester and Yvan Saeys.

Published in PLoS One 9(3), March 2014.

Abstract

One of the long-standing open challenges in computational systems biology is the topology inference of gene regulatory networks from high-throughput omics data. Recently, two community-wide efforts, DREAM4 and DREAM5, have been established to benchmark network inference techniques using gene expression measurements. In these challenges the overall top performer was the GENIE3 algorithm. This method decomposes the network inference task into separate regression problems for each gene in the network in which the expression values of a particular target gene are predicted using all other genes as possible predictors. Next, using tree-based ensemble methods, an importance measure for each predictor gene is calculated with respect to the target gene and a high feature importance is considered as putative evidence of a regulatory link existing between both genes. The contribution of this work is twofold. First, we generalize the regression decomposition strategy of GENIE3 to other feature importance methods. We compare the performance of support vector regression, the elastic net, random forest regression, symbolic regression and their ensemble variants in this setting to the original GENIE3 algorithm. To create the ensemble variants, we propose a subsampling approach which allows us to cast any feature selection algorithm that produces a feature ranking into an ensemble feature importance algorithm. We demonstrate that the ensemble setting is key to the network inference task, as only ensemble variants achieve top performance. As second contribution, we explore the effect of using rankwise averaged predictions of multiple ensemble algorithms as opposed to only one. We name this approach NIMEFI (Network Inference using Multiple Ensemble Feature Importance algorithms) and show that this approach outperforms all individual methods in general, although on a specific network a single method can perform better. An implementation of NIMEFI has been made publicly available.

2.1 Introduction

Transcriptional regulation is a key mechanism for cells to accomplish changes in gene expression levels. As a consequence, deciphering the structure of the gene regulatory network (GRN) is crucial to gain insights in biological processes of interest or the pathology of a cell. The availability of large collections of gene expression compendia offer the potential to infer the network topology in high-throughput and on a large-scale. As a consequence, many computational tools to infer GRNs from expression data have been developed and are being used in practical use cases [1]. However, inferring the GRN from expression data is a severely underdetermined problem, as the number of possible interactions largely exceeds the number of available measurements. Coping with this underdetermination has turned out to be a very difficult problem and has led to the development of an over-

whelming number of algorithms which use different strategies to overcome this inherent difficulty. Not only do these algorithms differ in the success they have to elucidate the network, they strike a balance between complexity and scalability and their predictions can be highly complementary [2, 3].

Algorithms that focus on inferring the topology of large GRNs typically calculate pair-wise measures between genes. Early methods used Pearsons correlation coefficient [4], but failed to identify non-linear relationships between genes. To capture these more complex dependencies, information theoretic measures have been proposed. In particular many models infer networks using the mutual information between a pair of genes as a measure [5]. These methods generally suffer from predicting many false positive links due to indirect effects and consequently various refinements have been proposed. CLR [6] corrects the predictions based on the specific background distribution of all mutual information scores. The ARACNE algorithm [7] uses the Data Processing Inequality on every triplet of genes to filter out indirect effects. MRNET [8] builds on the maximum relevance, minimum redundancy concept [9] using an iterative feature selection scheme. Finally, C3NET [10] and its ensemble extension BC3NET [11] try to avoid inferring indirect effects by only predicting a link between two genes if the mutual information between genes is at least maximal for one of the genes with respect to all other. More recently, the ANOVerence [12] algorithm proposed the η_2 score as an alternative measure to evaluate dependencies between genes. The η_2 score is a non-parametric and non-linear correlation coefficient derived using ANOVA. Finally, the TIGRESS [13] method solves the network inference problem by using a feature selection technique (LARS) combined with stability selection.

Various comparative studies have been performed which evaluate network inference algorithms [3, 14–16]. Due to the large variety of algorithms, these studies typically focus on a small subset of techniques and aim to derive interesting properties. Large scale evaluations of techniques have been performed in the context of the DREAM (Dialogue for Reverse Engineering Assessments and Methods) challenges [17]. DREAM aims to stimulate research in the field and provide researchers with benchmark datasets to validate their work. Community-wide network inference challenges where participants were invited to run their algorithms on blinded datasets have been organized. These challenges are the most comprehensive assessments of GRN inference algorithms. In both the DREAM4 in silico 100 multifactorial challenge and the latest DREAM5 network inference challenge [18–20] the overall top performer was the GENIE3 algorithm [21]. This method approaches the network inference problem by decomposing it into a separate regression problem for each possible target gene. Next, using a tree-based ensemble method, an importance measure for each predictor is calculated and a high feature importance is used as an indication that a link is present between the predictor and the target gene in the GRN.

Motivated by the success of GENIE3 and other ensemble methods based on feature importance, such as TIGRESS, we wish to explore the potential of several other ensemble feature importance techniques in the regression decomposition setting. We present a general framework which casts any feature selection algorithm into an ensemble setting by taking random subsamples of varying size of both the experiments and the potential regulatory genes. Furthermore, given the known complementary of GRN inference methods [17], we explore if it is beneficial to combine the predictions of several ensemble feature importance scoring algorithms as opposed to using a single one. Summarizing, we name this approach NIMEFI (Network Inference using Multiple Ensembles of Feature Importance algorithms) and compare the performance of this new method to several recently proposed state-of-the-art techniques.

2.2 Materials and Methods

2.2.1 Problem statement, evaluation and data sources

In this chapter we focus on the inference of the directed topology of large gene regulatory networks using gene expression data. Self-regulating interactions are not taken into consideration. As input data, we assume a compendium containing several gene expression measurements obtained from one or more experiments. We make no further assumptions about whether the data was compiled using gene-knockouts, multifactorial pertubations, steady-state measurements or any other experimental settings, nor do we take any time-related information into account. Although the directionality of a regulatory link is hard to infer without extracting specific information in interventional or time-series data, we opt for a directed topology setting throughout this chapter as this was the setting of the DREAM challenges and it allows for a fair comparison to other algorithms. Furthermore, it has been shown that the GENIE3 algorithm is able to predict directionality using the same data handling strategy [21]. Lastly, we allow for an optional limitative list of known regulatory genes as input data, in which case no outgoing links from other genes are allowed as predictions.

As such, let us define a learning sample (LS) from which to infer the GRN as a matrix of S rows by G columns, in which each row can be interpreted as the expression values of all G genes in one of the S available samples.

$$LS = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,G} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ x_{S,1} & x_{S,2} & \cdots & x_{S,G} \end{pmatrix}, x_{s,g} : \text{the expression of gene } g \text{ in sample } s$$

The desired output of the algorithm is then defined as a (fully connected) directed graph in which each node corresponds to a single gene and an edge between a certain node i and another node j indicates that gene i regulates the expression of gene j. For each possible edge in the network a score is given which represents the confidence that this is a true regulating link.

A recurrent problem among GRN inference methods is selecting a suitable cut-off value to transform the obtained ranking into an actual network structure. Some methods (e.g. [10, 11]) try to automatically select a network instead of a ranking while others leave the decision to the end-user. In this work, we do not automatically select a threshold value and instead will focus on the ranking. We believe that it is beneficial to the end-user to explore the network at various threshold levels. Some parts of the network can have a weaker signal in the gene expression set, but still be relevant to the user. As we focus on evaluating the ranking as opposed to a network, we will adopt the widely used DREAM5 procedure to score network predictions. In this procedure, all edges are sorted by decreasing confidence score and only the top 100,000 predictions are retained. Next, only considering the ranks, both the Area Under the Receiver Operating Characteristic curve (AUROC) and Area Under the Precision-Recall curve (AUPR) are calculated with respect to a known gold standard. We will evaluate all algorithms using five gene expression sets provided by the DREAM4 in silico 100 multifactorial challenge (DREAM4) and two gene expression sets from the DREAM5 network inference challenge (DREAM5). Furthermore, we created several artificial gene expression datasets using both the SynTReN [22] (SYNTREN-100) and GeneNetWeaver (GNW-100,GNW-200) [18, 23] simulators. Table 2.1 provides an overview of the various datasets.

The DREAM4 dataset consists of five artificial networks, each of size 100 (100 genes described by 100 experiments). These datasets were created for the DREAM4 in silico 100 multifactorial challenge and aim to mimic samples from multifactorial perturbation data, which is defined as static steady-state expression profiles achieved by slightly perturbing all gene expression values at the same time. In addition, two more datasets were used from the DREAM5 network inference challenge. The first is an artificial dataset consisting of 1643 genes, including a known list of 195 potential regulatory genes. No genes, other than these included in the list are regulatory genes in the gold standard used for validation. The topology of the in-silico network is based on known GRNs of model organisms. The compendium consists of various experimental settings. The second DREAM5 dataset describes a large real expression compendium of *E. coli*. This dataset contains a known list of 334 potential regulatory genes, and consists of measurements of 4511 genes obtained in various experimental conditions. To further test our findings, we created two network topologies of 100 nodes using an underlying *E. coli* network, one using SynTReN and one using GNW. For both underlying net-

Name	# Networks	# Compendia	# Samples	# Genes	# Regulatory genes	Type
DREAM4	5	5	100	100	100	Artificial
DREAM5 artificial	1	1	805	1643	195	Artificial
DREAM5 *E. coli*	1	1	805	4511	334	Real
SynTRreN-100	1	20	100	100	100	Artificial
GNW-100	1	20	100	100	100	Artificial
GNW-200	1	15	200	200	200	Artificial

Table 2.1: Characteristics of the different datasets used for evaluation.

works, 20 artificial expression compendia of 100 samples each were created using default settings. Lastly, using GNW and the same settings, we created fifteen different network topologies consisting of 200 nodes, for each network a dataset of 200 samples was simulated, again using default settings.

2.2.2 Ensemble feature selection techniques

Feature selection is an important preprocessing step in many machine learning applications, where it is often used to find the smallest subset of features that maximally increases the performance of the model. Other benefits of applying feature selection include the ability to build simpler and faster models using only a subset of all features, as well as gaining a better understanding of the processes described by the data, by focusing on a selected subset of features [24, 25]. Three types of feature selection techniques can be distinguished. Filter methods operate directly on the dataset, and provide a feature weighting, ranking or subset as output. These methods have the advantage of being fast and independent of the model, but at the cost of inferior results. Wrapper methods perform a search in the space of feature subsets, guided by the outcome of the model. They often report better results than filter methods, but at the price of an increased computational cost. Finally, embedded methods use internal information of the model to perform feature selection (e.g. use of the weight vector in linear models). They often provide a good trade-off between performance and computational cost. Recently, the concept of ensemble feature selection (EFS) was introduced in various problems. Just like ensemble models for classification and regression, EFS performs feature selection by combining different feature selection algorithms, usually obtained by bootstrapping, and then aggregates their results as the final output. EFS often results in better performance and more stable feature rankings than a single feature selection technique [26–28].

2.2.3 Generalizing GENIE3

GENIE3 decomposes the network inference problem between g genes as g separate regression problems. Each regression problem aims to predict the expression values of a particular target gene using the other genes as input genes. Using feature selection in this regression context then amounts to finding the other genes that are most indicative in modeling the expression values of the target gene, thus providing evidence of important regulators of the target gene. GENIE3 provides a ranking of the regulators of the target gene by deriving a weight for each regulator based on an ensemble of tree-based regression models, such as random forests [29]. Conceptually however, any kind of feature selection technique could be used to provide this ranking. The general principle of such a feature selection

based approach to network inference is depicted in Figure 2.1 and can be summarized as follows:

1. For $t = 1$ to G
   (a) Build a regression model predicting the vector of expression values of the target gene t : $LS_t = (x_{1,t}, \ldots, x_{S,t})$ using the learning sample matrix without the target gene values: LS_{-t}. Each column representing a possible predictor.
   (b) Use a feature selection technique FS to compute a feature importance (FI) value for each predictor column (gene) g in LS_{-t} : $FI_{t,g}$.
2. Aggregate the G individual regression problems to get a global ranking of all possible regulatory links in the network.

The feature selection technique used here is thus supposed to be able to compute a feature importance value for each input gene (feature) of the regression problem associated to gene t. In principle, any feature selection technique that is able to deal with a continuous output and returns some kind of feature importance measure or score can be used in this framework.

Instead of using a single feature selection algorithm to compute the importance values we could also use an ensemble of M feature selection algorithms $EFS = FS_1, FS_2, \ldots, F_M$ to calculate the importance values. The GENIE3 approach in the random forest setting is just a special case of such a more general ensemble setting, combining the bagging procedure to generate different regression tree models with random sampling of the variables within each regression tree node.

In general, any appropriate feature selection technique in this regression context can be easily cast into the EFS setting by generating M subsamples of the original learning sample LS, e.g. using regular subsampling or subsampling with replacement (bootstrap sampling). More specifically, we used the following approach to transform any FS algorithm which can produce a ranking to create an EFS method that produces FI scores for a regression subproblem with target gene t (Figure 2.1) :

1. For $i = 1$ to I
   (a) Take a subsample without replacement of size x of the samples (rows) of LS_{-t} and the target vector LS_t; where x is a uniformly randomly generated integer number between X_{min} and X_{max}.
   (b) Further reduce the matrix and target vector in size by taking a subsample of size y without replacement of the possible genes (columns); where y is a uniformly randomly generated integer number between Y_{min} and Y_{max}.

(c) Rank the features in the reduced predictor matrix by their ability to predict the reduced target vector using a FS technique of choice.

(d) Assign a score of '1' to the top Z features of the ranking and a score of '0' to all other

2. Sum the scores across all iterations and use these sums as the final FI scores for each predictor in the regression subproblem.

In the above algorithm the parameters $Z, Y_{min}, Y_{max}, X_{min}, X_{max}$ are typically set to default values of 5, 20%, 80% of the number of predictors and 20%, 80% of the number of experiments respectively. I is set to allow convergence. We explore and discuss the effect of these parameters on the algorithm performance in the results section. However, the general idea is to create subsamples of varying sizes, effectively searching for both global connections between genes as well as effects which can only be seen in a small number of experiments. By also sampling the predictors in each iteration, we avoid the problem of dominant genes masking possible secondary interactions. For computational reasons we imposed a maximum value of 200 on the upper bounds.

2.2.4 Calculating feature importance values using different machine learning techniques

In our experiments, the following machine learning algorithms in the regression context were used to calculate feature importance values. Two linear methods: linear support vector regression (SVR) and regularized regression using the elastic net (EL) as well as their respective ensemble versions (E-SVR, E-EL). We also used symbolic regression (SR), an inherently ensemble, non-linear method. For each of these methods we briefly describe how the feature rankings were calculated. For comparison reasons we also added an ensemble random forest regression (E-RFR) variant, differing from GENIE3 only due to the use of the subsampling scheme described earlier instead of a single regression for each target gene. In all of our experiments we used GENIE3 in random forest mode with the number of trees (T) set to 1000 and the number of randomly selected genes at each node of the tree (K) set to $g - 1$.

Support vector regression: from the Lagrange multipliers α , the support vectors x_i and the training labels y_i one can calculate the weight vector w in case of a linear kernel as:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

The absolute value of the weight w_j of the weight vector w corresponding to a certain feature j can be used to assess feature importance, as it is clear that small

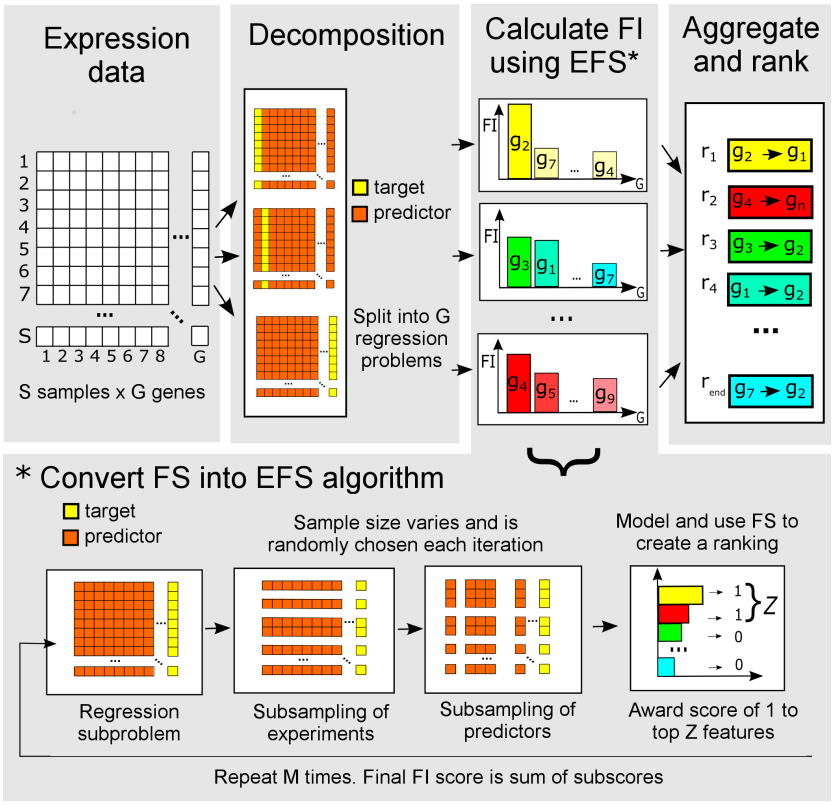

Figure 2.1: **Overview of the EFS approach to the network inference task.** *The problem is split into independent regression subproblems for each gene in the network. Next feature importance (FI) scores are calculated in each subproblem for all possible regulatory genes with respect to the target gene using an ensemble feature selection (EFS) method. These FI scores are then assigned as the weight of an edge in the network from the regulatory gene to the target gene. Finally, all weights are aggregated across the subproblems, creating a global confidence ranking of edges. We cast any feature selection (FS) method which can provide a ranking into an EFS method by taking random samples of varying size of both the experiments and the possible predictor genes and assigning a score of 1 to the top features in the ranking.*

values will have less impact on the predicted value [24]. We rank features in a single step as opposed to using the recursive feature elimination scheme of [30] for computational reasons. All experiments were conducted using the libsvm [31] package for R, using epsilon regression and a linear kernel function. The cost of constraint violation parameter C was set to the default value of 1, all other parameters were also set to their respective default values.

Elastic net: the absolute values of the coefficients of the predictors after a fit with a regularized linear model using the elastic net penalty function [32] are used as a measure for feature importance. The elastic net offers a compromise between ridge regression and the LASSO. The mixing coefficient, α $[0, 1]$, determines the tradeoff between the L1-norm and L2-norm regularization. If α is set to 1.0, it corresponds to the LASSO penalty, 0.0 corresponds to the ridge penalty. The λ parameter is defined as the regularization parameter and was determined by cross-validation. Experiments were conducted using the glmnet package for R [33] with α set to 0.3. We explored several other settings for the α parameter. Setting α to 1.0 had a negative effect on the performance, because less than Z features received a non-zero score in the regression subproblem. We did not notice any major performance differences between other settings of α.

Symbolic regression: this method aims to capture the input-output relation with an algebraic expression found through a genetic programming approach. In each iteration of the algorithm a large set of formulae, the population, is explored and regenerated using modifications such as crossover and mutation operations. We used pareto-aware symbolic regression [34] which utilizes multiple objectives to assess models. In our case, a Pareto-front is used which strikes a balance between the complexity of a formula and the goodness of fit. Next we determine a feature importance score for each score by a presence-based measure [35]. As irrelevant variables will cause extra complexity without lowering the error measure of the mode, these variables are discouraged to be included in the population. As such, the final population after a modeling run will likely only contain the most relevant variables. We use the percentage of the population in which a feature is present as the feature importance score. Other feature scoring approaches were also explored, but achieved similar or worse results. Experiments were conducted using the proprietary DataModeler software using four independent evolutions.

2.2.5 Comparisons with TIGRESS

We included the results of TIGRESS in all our performance comparisons. The performance scores of TIGRESS for the DREAM5 dataset were taken from the DREAM5 challenge results. All other performance metrics were obtained by running TIGRESS using the GenePattern platform (dream.broadinstitute.org) with default parameter settings and 1000 iterations.

2.2.6 Merging multiple feature importance algorithms

To explore the known complementarity of GRN inference methods, we create new predictions by combining the predictions of several algorithms. In particular, we focus on the combination of several FI scoring algorithms as opposed to only using a single algorithm, an approach which we named NIMEFI. In order to create combined predictions, we aggregate using Borda count, which in this setting is equal to averaging rankwise across predictions to obtain a newly predicted rank for each possible regulatory interaction. We do not impose a prior cut-off value to the individual prediction rankings of the methods before merging. Other aggregation strategies such as minimum rank or median rank were explored but were omitted from this chapter as the results were either similar or worse than averaging rankwise.

2.3 Results

2.3.1 Performance comparison on the DREAM4 size 100 in silico multifactorial dataset

Table 2.2 lists the performance of several algorithms on the DREAM4 multifactorial dataset. For comparison reasons, we include the scores of the TIGRESS algorithm. A first observation is that the ensemble setting seems to be key to the network inference problem. Both the single-step versions of the ELand SVR are unable to compete with the other algorithms. However, when casting these algorithms into the ensemble setting, their performance increases drastically reaching similar scores than those of GENIE3 and TIGRESS. The performance of Symbolic Regression is also similar to all the EFS-methods. Both Symbolic Regression and GENIE3 can be classified as inherently ensemble feature importance scoring algorithms. A second observation which can be made is that GENIE3 outperforms E-RFR, although in both algorithms random forest regression is used. GENIE3 calculates the feature importance based on the variance reduction due to splitting on a feature [36]. Our results indicate this measure outperforms the counting measure implemented in the subsampling scheme. We believe this is partially because the variance reduction measure allows for a more objective comparison between different regression problems, allowing for a better aggregation towards a global ranking across the different regression subproblems. A final observation which can be made is the seemingly overall better performance of the averaged predictions of multiple ensemble feature importance algorithms as opposed to predictions of a single one. All combined predictions show an increase in AUROC score, while retaining a similar score for AUPR. In the last column the standard deviation of the performance scores over five runs is shown. The standard deviation is close to 0 for all algorithms. Both EL and SVR are deterministic, the other algorithms

introduce randomness but output stable results due to their ensemble nature.

Table 2.2: Performance comparison of several algorithms on the DREAM4 in silico multifactorial dataset. EL= Elastic Net, E-EL = Ensemble Elastic Net, SVR= Support Vector Regression, E-SVR= Ensemble Support Vector Regression, SR= Symbolic Regression, E-RFR= Ensemble Random Forest Regression, G= GENIE3). '+' indicates rankwise averaging of several methods. ALL= GENIE3+E-SVR+E-EL. Ensemble variants, indicated with 'E-', were created using the subsampling scheme with default settings.

Method-Eval	Net1	Net2	Net3	Net4	Net5	Avg.
EL-AUROC	0.63	0.62	0.65	0.68	0.65	**0.64**
EL-AUPR	0.13	0.11	0.19	0.18	0.18	**0.15**
E-EL-AUROC	0.74	0.67	0.74	0.76	0.75	**0.73**
E-EL-AUPR	0.16	0.15	0.23	0.21	0.21	**0.19**
SVR-AUROC	0.58	0.60	0.62	0.61	0.58	**0.57**
SVR-AUPR	0.02	0.04	0.04	0.04	0.03	**0.03**
E-SVR-AUROC	0.74	0.78	0.80	0.80	0.81	**0.78**
E-SRV-AUPR	0.13	0.16	0.20	0.19	0.19	**0.17**
SR-AUROC	0.71	0.69	0.72	0.75	0.74	**0.72**
SR-AUPR	0.16	0.13	0.25	0.19	0.24	**0.19**
E-RFR-AUROC	0.73	0.70	0.74	0.77	0.77	**0.74**
E-RFR-AUPR	0.15	0.15	0.22	0.20	0.20	**0.18**
G-AUROC	0.74	0.69	0.76	0.80	0.77	**0.75**
G-AUPR	0.17	0.15	0.26	0.24	0.23	**0.20**
TIGRESS-AUROC	0.75	0.70	0.76	0.77	0.75	**0.76**
TIGRESS-AUPR	0.16	0.16	0.26	0.23	0.23	**0.20**
G+E-SVR-AUROC	0.76	0.75	0.79	0.82	0.81	**0.79**
G+E-SVR-AUPR	0.16	0.16	0.25	0.23	0.24	**0.20**
ALL-AUROC	0.76	0.73	0.78	0.81	0.80	**0.78**
ALL-AUPR	0.16	0.16	0.25	0.23	0.24	**0.20**

2.3.2 Performance evaluation on the DREAM5 dataset

Table 2.3 shows the performance on the DREAM5 datasets. We included the results of ANOverence, ranked second best overall in the DREAM5 challenge, with the best performance on the *E. coli* network. However, as opposed to GENIE3 and TIGRESS, ANOVerence does include meta-information of the microarray chips to guide the network inference process. GENIE3 performs well on the artificial dataset, outscoring all individual methods. Note that in the original DREAM5 challenge results, GENIE3 reached a lower overall score because a different parameter setting was used for the number of input variables that are randomly chose at each node. The ensemble versions of EL and SVR are able to perform slightly better than ANOverence but are outperformed by TIGRESS with regard to the

Table 2.3: Performance comparison of several algorithms on the DREAM5 dataset. E-EL = Ensemble Elastic Net, E-SVR= Ensemble Support Vector Regression,G=GENIE3, '+' indicates rankwise averaging of several methods

Method	Artificial		*E. coli*	
	$AUROC$	$AUPR$	AUROC	$AUPR$
E-SVM	0.79	0.24	0.61	0.12
E-EL	0.78	0.28	0.63	0.11
G	0.81	0.38	0.62	0.10
G+E-SVR	0.83	0.35	0.63	0.11
G+E-SVR+E-EL	0.82	0.32	0.63	0.11
TIGRESS	0.78	0.30	0.60	0.07
ANOVerence	0.78	0.25	0.67	0.12

AUPR score. The combinations GENIE3+E-SVR and GENIE3+E-SVR+E-EL are very competitive, reaching a higher AUROC score than GENIE3 but perform worse with regard to the AUPR score. Both methods however clearly outperform TIGRESS and ANOverence on this particular dataset. On the biological *E. coli* dataset the individual methods, E-SVM, E-EL and GENIE3, show a similar performance. The combination predictions have a slightly higher score overall, but the difference is minimal compared to the individual methods. ANOVerence reaches the overall best score, making effective use of the available meta-information of the microarray chips.

2.3.3 Performance evaluation on the SyNTreN and GeneNetWeaver datasets

The previous experiments were performed on benchmark data and the sample size was insufficient to determine if the NIMEFI approach outperformed the individual ensemble methods. In order to further explore if there is a significant difference in performance, we performed additional experiments on synthetic generated data of which the results can be seen in Figure 2.2. The boxplots show the AUROC and AUPR scores on three artificial datasets as described in the previous section. For these datasets we show the results of E-EL, E-SVR and GENIE3 and all possible combinations of these three methods. Again for comparison reasons, the results of TIGRESS are shown. We omitted the Symbolic Regression algorithm from the comparison as the computational complexity of this algorithm greatly exceeds all others, resulting in quickly deteriorating performance as the expression matrix size increases. The results for E-RFR were also omitted from the figure in favor of GENIE3 due to a consistent better performance of the latter method.

Again, as on the DREAM4 dataset, the ensemble versions of EL and SVR seem to be very competitive with GENIE3 without an overall winner between these three

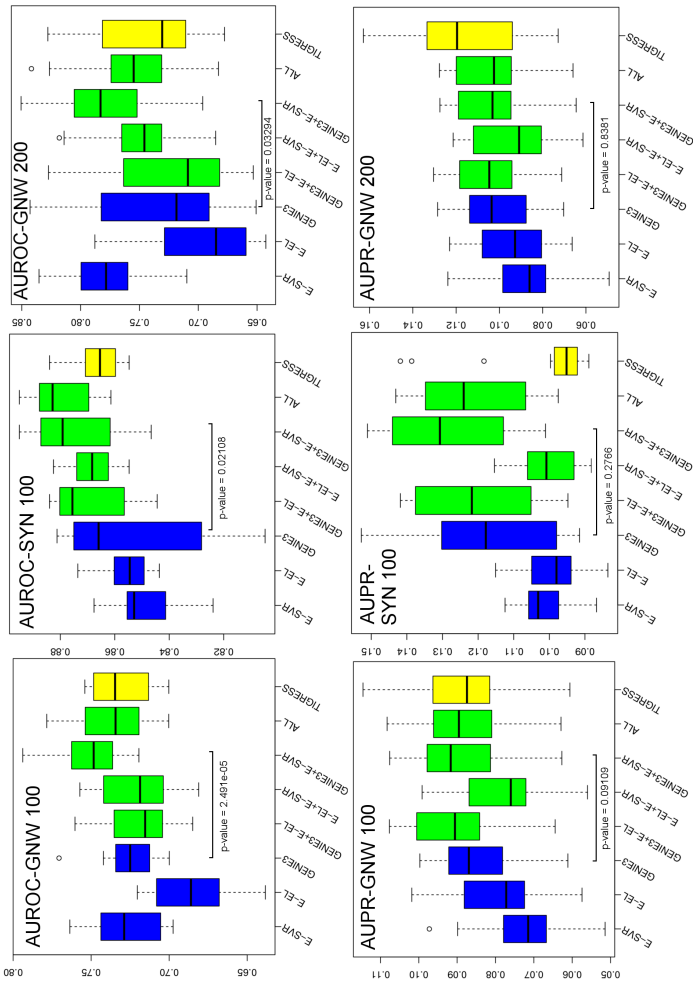

Figure 2.2: **Boxplots of AUROC and AUPR scores on the three artificially created datasets.** *Shown in blue is the performance of three individual algorithms: GENIE3 and the ensemble versions of support vector regression and the elastic net (E-SVR, E-EL). Indicated in green the results after rankwise merging the individual methods and in yellow the performance of TIGRESS.Indicated in the figure are the results of Mann-Withney U-tests between GENIE and GENIE3+E-SVR (sample size 20 for GNW-100 and SYN 100, sample size 15 for GNW-200) showing that the AUROC scores are significantly improved.*

algorithms surfacing over all datasets. Interestingly, is the performance increase by rankwise combining predictions as seen in GENIE3+E-SVR and GENIE3+E-SVR+E-EL (ALL). Especially the combined prediction of GENIE3+E-SVR seems most promising. Indeed comparing GENIE3 to GENIE3+E-SVR we notice a significant improvement in AUROC score in all three datasets (Mann-Withney U-test, GENIE and GENIE3+E-SVR, sample size 20 for GNW-100 and SYN 100, sample size 15 for GNW-200, p-values: $2.491e^{-5}$, 0.02108, 0.03294), while the AUPR values are not significantly different (Mann-Withney U-test, same settings, p-values: 0.0910, 0.2766, 0.8381).

Combined with the results of the previous section, this suggests that overall it is beneficial to use multiple ensembles feature selection techniques, as they provide better and more consistent performance among different datasets.

2.3.4 Influence of the parameter settings of the subsampling scheme

In the previous results, the parameters $Z, Y_{min}, Y_{max}, X_{min}, X_{max}$ were always set to their respective default values of 5, 20%, 80%, 20% and 80%. The parameter I was set to 10000 to allow for complete convergence. In this section, we explore various different settings for these parameters. We show average AUROC scores using the E-SVR variant on the DREAM4 in silico multifactorial dataset. The AUPR scores show similar behavior.

Table 2.4 shows the effect of only varying the Z parameter, which controls how many predictors receive a score of '1' in a single feature ranking of a subsample regression problem. The results indicate that for low values of Z, the performance is stable but consistently degrades towards higher values for Z. This result can intuitively be explained by the fact that for higher values of Z the algorithm loses discriminative power between the top ranked genes as all genes in the top Z of the feature ranking will receive the same score of 1. One could also try to determine Z in an automated way using the feature importance scores in a single subsample regression problem. We explored this option by using an algorithm based on a Grubbs test for outliers to determine the optimal number of predictors, however this lead to a significant decrease in performance across all datasets. We believe determining a cut-off value is a non-trivial problem in this context, as the distribution of the feature importance scores varies significantly across the regression subproblems. This is further complicated by the fact that some feature importance scores, e.g. the absolute value of the SVM feature weight vector, are hard to interpret outside of an ordinal context. Concluding, we advise a value of 5 for the Z parameter for all datasets, as it is in the stable region. Furthermore, from a biological perspective, it is unrealistic to e.g set $Z = 20$ as this would imply that a gene is directly regulated by at least 20 other genes present in the subsample of

predictors.

Table 2.4: Influence of the subsampling scheme parameter Z on the E-SVR AUROC score using the DREAM4 dataset. Default setting indicated in boldface. Difference in AUROC score compared to the default setting is shown.

Z rank threshold				
2	5	10	25	50
0.00	**0.79**	-0.02	-0.04	-0.08

Next, we investigate the effect of the parameters X_{min} and X_{max}, which control the size of the subsample of the experiments in the algorithm. Table 2.5 lists the performance for varying values. We notice that the performance suffers if only small subsamples ($\leq 50\%$) are included throughout the iterations. This is the case if X_{max} is set too low, especially in combination with a low X_{min}. If the X_{max} parameter is set to 80% or 100% we notice a stable region in which the X_{min} parameter has little effect on the performance. If both parameters are set to 100%, resulting in only subsampling the predictors in each iteration, a small performance drop can be seen. From a computational perspective, smaller subsamples are preferred as the regression subproblem is faster to solve. Taking both observations into account, we suggest a default value of 80% for X_{max} and a default value of 20% for X_{min}.

Table 2.5: Influence of the subsampling scheme parameters X_{min} and X_{max} on the E-SVR AUROC score using the DREAM4 dataset.Default setting indicated in boldface. Difference in AUROC score compared to the default setting is shown.

	X_{max}				
X_{min}	5	20	50	80	100
5	-0.12	-0.12	-0.07	-0.01	-0.01
20	/	-0.11	-0.05	**0.79**	-0.01
50	/	/	-0.01	0.00	-0.01
80	/	/	/	-0.01	-0.01
100	/	/	/	/	-0.02

Furthermore, the effect of the parameters Y_{min} and Y_{max} on the performance is shown in Table 2.6. These parameters control the size of the subsample of the possible predictors in each regression subproblem. The results indicate that the performance decreases with higher values for Y_{min} ($\geq 50\%$), especially in combination with higher Y_{max} values. We believe the algorithms benefits from smaller predictor subsamples as it allows for alternative features to be picked up as important predictors in the regression subproblem, which would otherwise be missed because another (dominant) gene was used in the prediction model. Again from a computational perspective, smaller values for Y_{min} and Y_{max} are preferred

as it decreases the complexity of the regression subproblem. As such, we also advice for default values of 20% for Y_{min} and a default value of 80% for Y_{max}.

Table 2.6: Influence of the subsampling scheme parameters Y_{min} and Y_{max} on the E-SVR AUROC score using the DREAM4 dataset. Default setting indicated in boldface. Difference in AUROC score compared to the default setting is shown.

	Y_{max}			
Y_{min}	20	50	80	100
5	-0.01	0.00	0.00	-0.02
20	0.00	0.00	**0.79**	-0.02
50	/	0.00	-0.02	-0.05
80	/	/	-0.07	-0.10
100	/	/	/	-0.13

Lastly, we explore the effect of the number of iterations on the stability of the algorithm. Figure 2.3 shows eight boxplots over ten runs for different number of total iterations. As desired, the performance variance between runs decreases as the number of subsamples is increased, reaching an almost completely stable result at values above 1500 iterations.

2.3.5 Additional comparisons between methods

We explored if there are other differences in the predictions of the investigated methods besides the performance. First, we analyzed if there were any biases in the node degree distribution of the inferred networks. We transformed the network predictions into an undirected setting by retaining the first edge in the ranking between each possible pair of nodes and removing the other. Next, networks were created from the undirected rankings by imposing several cut-off values. Figure 2.4 shows the node degree distribution of all algorithms on four networks selected from the different datasets. Here a cut-off value close to the number of links in the underlying gold standard network was chosen (450 for GNW-200, 150 for all others). The figure indicates that the different methods can have a dissimilar node degree distribution on a certain network, however we found no clear bias associated with an algorithm across all network predictions. We notice that although the performance of two algorithms can be similar, the individual predictions seem to vary. Figure 2.5 illustrates this behavior further. In this figure we selected two predictions (GENIE3 and E-SVR) which had a similar performance score on a network in the GNW-100 dataset. Both algorithms had an AUROC score of 0.70 and an AUPR score of 0.08. We plotted the 500 top edges of the GENIE3 ranking versus the position they received in the E-SVR ranking and vice versa. True positive links are shown in green. For both plots, several links appearing at the top of the ranking for one prediction, are ranked at the very bottom of the other. This

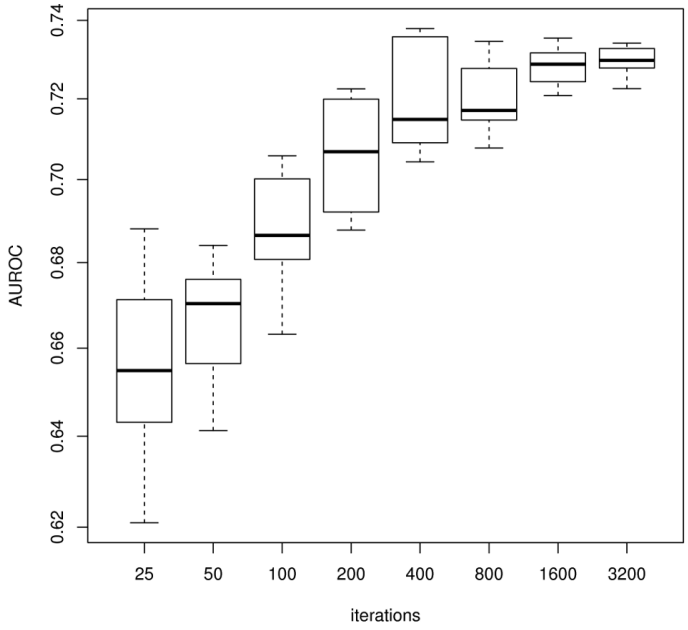

Figure 2.3: **Boxplots of AUROC scores over ten runs with respect to the number of iterations.** *The boxplots show the AUROC score over ten runs of the E-SVR algorithm on the first network of the DREAM4 dataset. The variance decreases as the number of subsamples is increased, reaching a stable result at about 1500 iterations.*

observation and the varying node degree distribution seem to hint that the predictions of the algorithms can show a different behavior, however it is non-trivial to quantify this difference.

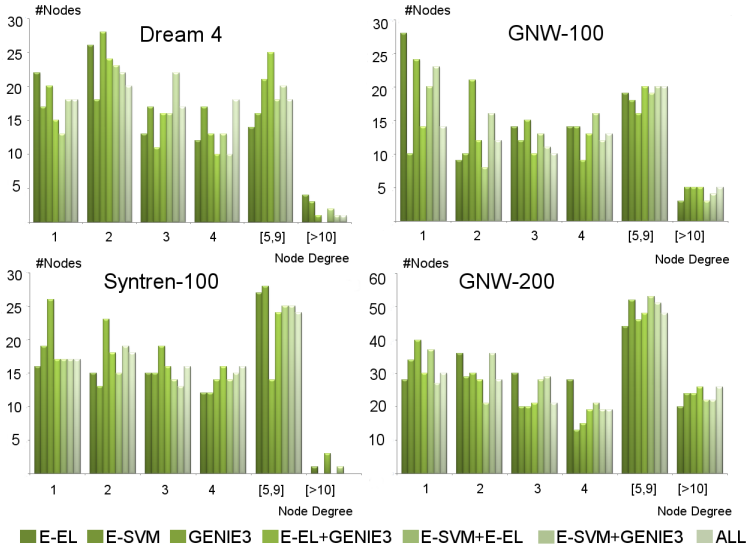

Figure 2.4: **Node degree distribution of four network predictions selected across the different datasets.** *Networks predictions were interpreted in an undirected setting. The networks were created from the rankings by imposing a cut-off value close to the number of true links in the corresponding gold network. Although the figure indicates that the node degree distribution can vary for the different algorithm predictions, there is no consistent pattern across the expression sets.*

Finally, we investigated the ability of the algorithms to predict the correct directionality of the link as in [21]. First, we removed all bi-directional links from the gold standard networks. Next, we counted the number of times a gold link $a \rightarrow b$ was ranked before the opposite link $b \rightarrow a$, proportional to the total number of gold standard links. We performed this analysis for all networks in the DREAM4, GNW-100, SYNTREN-100 and GNW-200 datasets. We omitted the two networks of the DREAM5 dataset as a list of regulatory genes was available. Figure 2.6 shows a boxplot of the directionality scores for all algorithms on the remaining 60 networks. A first observation is that the E-SVR and GENIE3 algorithm predict directionality better than the E-EL algorithm. The plot also indicates that combining multiple ensemble feature importance algorithms is beneficial to the directionality prediction in the case of GENIE3+E-SVR (Wilcoxon rank sum test with continuity correction: GENIE3 and GENIE3+E-SVR, sample size 60, p-value= $9.302e^{-5}$).

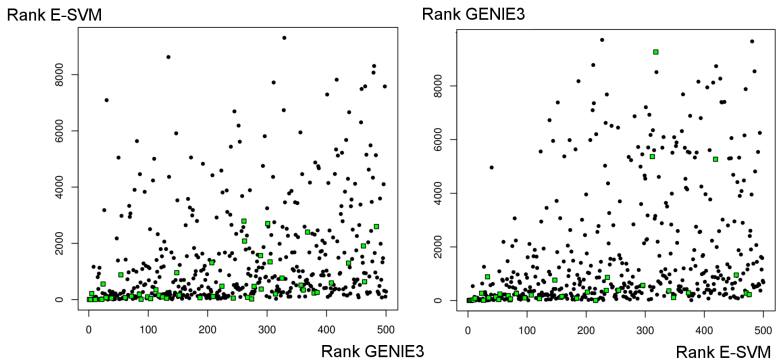

Figure 2.5: **Comparison of the given rank at the edge level of two algorithm predictions.** *In the figure on the left we plot the rank of the top* 500 *most confident links of the GENIE3 prediction versus the rank which these edges received in the E-SVM prediction. True positive links are indicated as green squares. Although the AUROC and AUPR scores of both methods are almost identical for this network, several top predicted edges by GENIE3, including true positives, appear much further down the ranking of E-SVM and vice versa.*

2.3.6 Computational aspects

The ensemble subsampling approach used in NIMEFI results in modeling a regression problem (of sample size S by G features), N times, and using a feature selection algorithm of choice in each iteration to obtain a feature ranking. If an embedded feature selection technique is used, as was the case with the elastic net and support vector regression techniques, the cost breaks down to solving the S by G regression problem N times, which is entirely dependent on the machine learning technique and implementation which is used. As such, if for example $S \ll G$, as is the case with gene expression compendia, using support vector regression in dual representation becomes computationally interesting. Furthermore, each iteration of the algorithm is independent and each target gene can also be solved in parallel. Due to these properties, running NIMEFI as a backfill job on a cluster becomes very interesting as the individual work packages can be tuned to almost every time slot.

In our work, we did not focus on optimizing the running time of the specific algorithms as we made use of general available libraries to compare a wide array of methods. However to give an indication of the relative running times, Table 2.7 gives an overview of the running times of some of the algorithms. These measurements were conducted using an Intel i3 CPU M350 clocked at 2.27GHz, 8.00 GB of RAM memory and a 64-bit operating system. For the theoretical computational complexity, we refer to the specific software packages as listed in the Materials and Methods section.

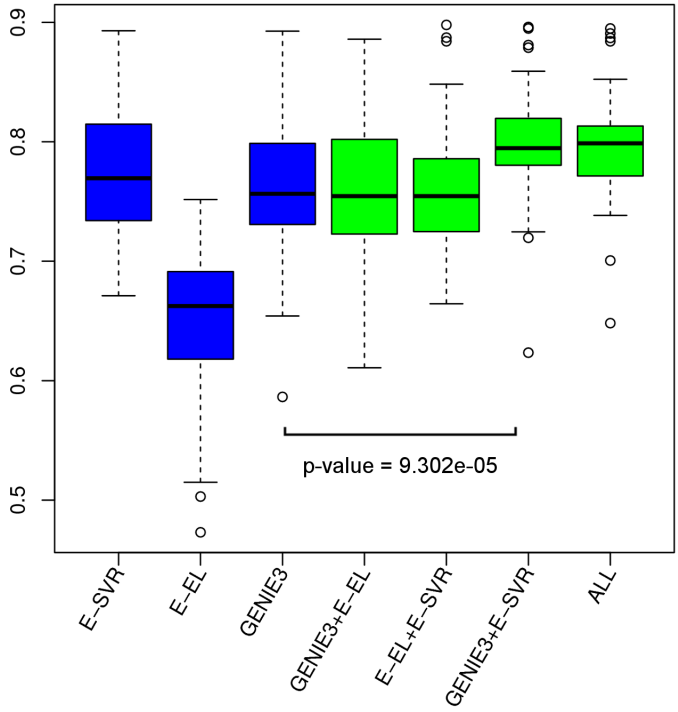

Figure 2.6: **Boxplots showing the ability to predict the correct directionality of a true positive link.** *For all predictions we counted the number of times a gold standard link $a \rightarrow b$ was ranked before the opposite link $b \rightarrow a$, proportional to the total number of links in the gold network. We performed this analysis for all networks in the DREAM4, GNW-100, SYNTREN-100 and GNW-200 datasets. The boxplots show the results for all algorithms. The GENIE3+E-SVR is significantly better at predicting the correct direction compared to GENIE3 (Wilcoxon rank sum test with continuity correction: GENIE3 and GENIE3+E-SVR, sample size 60, p-value=* $9.302e^{-5}$*).*

Table 2.7: Comparison of indicative running times of E-SVR, E-EL and GENIE3. Run time in seconds for a single complete iteration or subsample of the algorithm considering all genes.

Method	Running time DREAM4	Running time DREAM5
GENIE3	$4.77E^{+02}$	$4.01E^{+05}$
E-SVR	$3.10E^{+0}$	$4.92E^{+02}$
E-EL	$1.39E^{+2}$	$8.46E^{+02}$

2.4 Discussion

In this article we generalized the GENIE3 algorithm for GRN inference to other feature importance scoring algorithms in the same regression context. We presented a subsampling approach which allows any feature selection algorithm that can produce a ranking to be cast into an ensemble feature importance scoring algorithm. Using this scheme, we have analyzed the performance of several FS algorithms using the DREAM4 multifactorial and DREAM5 benchmarks, as well as several other artificially simulated datasets created using the SynTReN and GeneNetWeaver tools. We show that using this approach, several algorithms achieve equally good performance than GENIE3, demonstrating that an ensemble setting is key to achieve state-of-the-art-performance on the network inference task.

In the DREAM5 challenge an ensemble of different network inference methods outperformed the single methods, establishing the 'wisdom of the crowds-approach' to the gene regulatory network inference problem. Motivated by these conclusions, we explored the method of using a rankwise combination of several ensemble feature importance scoring algorithms to the network inference task as opposed to a single one, an approach which we named NIMEFI.

On the DREAM4 dataset, the performance of the combined predictions outscores the single predictions, resulting in an overall score better than TIGRESS and the original GENIE3 algorithm. Using the artificially created datasets these finding were confirmed. Although no clear winner could be found among the different EFS algorithms, their combined predictions achieved a significantly higher AUROC score on all three datasets.

The results on the DREAM5 dataset were inconclusive. On the artificial dataset GENIE3 outperforms all other single algorithms but the NIMEFI combinations achieve a slightly better AUROC score at the cost of a slightly lower AUPR. On the biological dataset, all three individual methods perform similar and the NIMEFI combinations achieve a minimal performance gain.

Comparing the different methods further, we have shown examples in which the performance of the individual ensemble methods can be very similar although the predicted networks are different with respect to the node degree distribution or the predicted individual edges. Moreover, we have also investigated the ability of the methods to predict the correct directionality of the link.

We also explored the impact of the parameters of the proposed subsampling approach on the performance. We suggested and motivated default values and have shown that within reasonable and intuitive ranges, the performance is stable with regard to these settings.

Concluding, our findings indicate that the use of NIMEFI as opposed to using a single ensemble feature importance can further increase performance on the network inference task, reaching state-of-the-art performance on the DREAM

datasets as well as on several artificial datasets. Combining the good performance with the computational attractive parallelization nature of NIMEFI, we believe our approach is an interesting alternative to other GRN methods. An implementation of our method is available for download at http://bioinformatics.intec.ugent.be/.

References

[1] P. B. Madhamshettiwar, S. R. Maetschke, M. J. Davis, A. Reverter, and M. A. Ragan. *Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets*. Genome Medicine, 4(5):41, 2012.

[2] T. Michoel, R. De Smet, A. Joshi, Y. Van de Peer, and K. Marchal. *Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks*. BMC Systems Biology, 3(1):49, 2009.

[3] R. De Smet and K. Marchal. *Advantages and limitations of current network inference methods*. Nat Rev Micro, 8(10):717–729, October 2010.

[4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. *Cluster analysis and display of genome-wide expression patterns*. Proceedings of the National Academy of Sciences, 95(25):14863–14868, December 1998.

[5] A. J. Butte and I. S. Kohane. *Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements*. In Pacific Symposium on Biocomputing, pages 418–429, 2000.

[6] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. *Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles*. PLoS Biol, 5(1):e8, January 2007.

[7] A. A. Margolin, K. Wang, W. K. Lim, M. Kustagi, I. Nemenman, and A. Califano. *Reverse engineering cellular networks*. Nature Protocols, 1(2):662–671, 2006.

[8] P. Meyer, K. Kontos, F. Lafitte, and G. Bontempi. *Information-theoretic inference of large transcriptional regulatory networks.* EURASIP journal on bioinformatics & systems biology, 2007.

[9] C. Ding and H. Peng. *Minimum Redundancy Feature Selection from Microarray Gene Expression Data*. In J Bioinform Comput Biol, pages 523–529, 2003.

[10] G. Altay and F. Streib. *Inferring the conservative causal core of gene regulatory networks*. BMC Systems Biology, 4(1):132, September 2010.

[11] R. de Matos Simoes and F. Emmert-Streib. *Bagging Statistical Network Inference from Large-Scale Gene Expression Data*. PLoS ONE, 7(3):e33624, 03 2012.

[12] R. Küffner, T. Petri, P. Tavakkolkhah, L. Windhager, and R. Zimmer. *Inferring gene regulatory networks by ANOVA*. Bioinformatics, 28(10):1376–1382, May 2012.

[13] A.-C. Haury, F. Mordelet, P. Vera-Licona, and J.-P. Vert. *TIGRESS: Trustful Inference of Gene REgulation using Stability Selection*. BMC Systems Biology, 6(1):145, 2012.

[14] N. Soranzo, G. Bianconi, and C. Altafini. *Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data*. Bioinformatics, 23(13):1640–1647, July 2007.

[15] H. Hache, H. Lehrach, and R. Herwig. *Reverse engineering of gene regulatory networks: a comparative study*. EURASIP J. Bioinformatics Syst. Biol., 2009:8:1–8:12, January 2009.

[16] V. Narendra, N. I. Lytkin, C. F. Aliferis, and A. Statnikov. *A comprehensive assessment of methods for de-novo reverse-engineering of genome-scale regulatory networks*. Genomics, 97(1):7–18, January 2011.

[17] D. Marbach, J. Costello, R. Kuffner, N. Vega, R. Prill, D. Camacho, K. Allison, M. Kellis, J. Collins, and G. Stolovitzky. *Wisdom of crowds for robust gene network inference*. Nat Meth, 9(8):796–804, August 2012.

[18] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. *Generating Realistic In Silico Gene Networks for Performance Assessment of Reverse Engineering Methods*. Journal of computational biology, 16(2):229–239, February 2009.

[19] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky. *Revealing strengths and weaknesses of methods for gene network inference*. Proceedings of the National Academy of Sciences, 107(14):6286–6291, April 2010.

[20] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky. *Towards a Rigorous Assessment of Systems Biology Models: The DREAM3 Challenges*. PLoS ONE, 5(2):e9202, February 2010.

[21] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. *Inferring Regulatory Networks from Expression Data Using Tree-Based Methods*. PLoS ONE, 5(9), 2010.

[22] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal. *SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms*. BMC Bioinformatics, 7(1):43, 2006.

[23] T. Schaffter, D. Marbach, and D. Floreano. *GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods*. Bioinformatics, 27:2263–2270, June 2011.

[24] I. Guyon and A. Elisseeff. *An introduction to variable and feature selection*. J. Mach. Learn. Res., 3:1157–1182, 2003.

[25] Y. Saeys, I. n. Inza, and P. Larrañaga. *A review of feature selection techniques in bioinformatics*. Bioinformatics, 23(19):2507–2517, October 2007.

[26] T. Abeel, T. Helleputte, Y. Van de Peer, P. Dupont, and Y. Saeys. *Robust biomarker identification for cancer diagnosis with ensemble feature selection methods*. Bioinformatics, 26(3):392–398, February 2010.

[27] F. R. Bach. *Bolasso: model consistent Lasso estimation through the bootstrap*. In Proceedings of the 25th international conference on Machine learning, ICML '08, pages 33–40, New York, NY, USA, 2008. ACM.

[28] Y. Saeys, T. Abeel, and Y. Van de Peer. *Robust Feature Selection Using Ensemble Feature Selection Techniques*. In Machine Learning and Knowledge Discovery in Databases SE - Lecture Notes in Computer Science, volume 5212, pages 313–325. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

[29] L. Breiman. *Random Forests*. In Machine Learning, pages 5–32, 2001.

[30] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. *Gene Selection for Cancer Classification using Support Vector Machines*. Machine Learning, 46(1-3):389–422, 2002.

[31] C.-C. Chang and C.-J. Lin. *LIBSVM: A library for support vector machines*. ACM Trans. Intell. Syst. Technol., 2(3):27:1—-27:27, May 2011.

[32] H. Zou and T. Hastie. *Regularization and variable selection via the Elastic Net*. Journal of the Royal Statistical Society, Series B, 67:301–320, 2005.

[33] J. Friedman, T. Hastie, and R. Tibshirani. *Regularization paths for generalized linear models via coordinate descent*. Journal of Statistical Software, 33(1):1–22, 2009.

[34] G. Smits and M. Kotanchek. *Pareto-Front Exploitation in Symbolic Regression*. In U.-M. OReilly, T. Yu, R. Riolo, and B. Worzel, editors, Genetic Programming Theory and Practice II SE - 17, volume 8 of *Genetic Programming*, pages 283–299. Springer US, 2005.

[35] K. Vladislavleva, K. Veeramachaneni, M. Burland, J. Parcon, and U.-M. O'Reilly. *Knowledge mining with genetic programming methods for variable selection in flavor design*. In Proceedings of the 12th annual conference on Genetic and evolutionary computation, GECCO '10, pages 941–948, New York, NY, USA, 2010. ACM.

[36] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

"There is a single light of science, and to brighten it anywhere is to brighten it everywhere."

— Isaac Asimov

3

Netter: re-ranking gene network inference predictions using structural network properties

In this next chapter, we propose a method named Netter to further refine predictions made by a network inference algorithm by using structural properties of gene regulatory networks. The large majority of proposed inference algorithms do not take into account that gene regulatory networks have several network characteristics which define them and are not random graphs. Netter can be used as a second step after applying any network inference of choice and proves to be a valuable addition to further boost the accuracy of such predictions.

⋆ ⋆ ⋆

Joeri Ruyssinck, Piet Demeester, Tom Dhaene and Yvan Saeys.

Published in BMC Bioinformatics 17.1:76, February 2016.

Abstract

Many algorithms have been developed to infer the topology of gene regulatory networks from gene expression data. These methods typically produce a ranking of links between genes with associated confidence scores, after which a certain threshold is chosen to produce the inferred topology. However, the structural properties of the predicted network do not resemble those typical for a gene regulatory network, as most algorithms only take into account connections found in the data and do not include known graph properties in their inference process. This lowers the prediction accuracy of these methods, limiting their usability in practice. We propose a post-processing algorithm which is applicable to any confidence ranking of regulatory interactions obtained from a network inference method which can use, *inter alia*, graphlets and several graph-invariant properties to re-rank the links into a more accurate prediction. To demonstrate the potential of our approach, we re-rank predictions of six different state-of-the-art algorithms using three simple network properties as optimization criteria and show that Netter can improve the predictions made on both artificially generated data as well as the DREAM4 and DREAM5 benchmarks. Additionally, the DREAM5 *E.coli.* community prediction inferred from real expression data is further improved. Furthermore, Netter compares favorably to other post-processing algorithms and is not restricted to correlation-like predictions. Lastly, we demonstrate that the performance increase is robust for a wide range of parameter settings. Netter is available at http://bioinformatics.intec.ugent.be. Concluding, we believe that Netter is an interesting second step in the network inference process to further increase the quality of prediction. It is a flexible system which can be applied in unison with any method producing a ranking from omics data and can be tailored to specific prior knowledge by expert users or applied in general uses cases.

3.1 Introduction

Network representations are widely used and vital in many areas of science and engineering. They serve both as an endpoint for users to structure, visualize and handle large amounts of data and as a starting point for various algorithms that use networks for automated hypothesis generation. In systems biology, one of the long-standing challenges is the reverse engineering of the cell's transcriptome in the form of gene regulatory networks (GRNs). This has proven to be a daunting task, as the number of genes in the network vastly exceeds the number of available measurements. As a result, many computational methods have been developed [1–3] which try to overcome this challenge using various strategies. These methods differ not only in their accuracy to infer the network but also strike a different balance between scalability and complexity [4, 5]. In a recent community-wide effort, a large blind assessment of unsupervised inference methods using

microarray gene expression data was conducted [6]. This study concluded that no single inference method performs best across all data sets but that in contrast, the integration of several techniques to form an ensemble 'community' prediction led to a robust and top performance. In a collaborative effort between the DREAM organizers, the GenePattern team [7] and individual contributors, a web service GP-DREAM was set up to run and combine current state-of-the-art methods. To date, five methods are offered: ANOVerence [8], CLR [9], GENIE3 [10], the Inferelator [11] and TIGRESS [12].

Common inference strategies of GRN inference algorithms include the calculation of local pairwise measures between genes or the transformation of the problem into independent regression subproblems to derive connections between genes. It is clear that using these schemes, the algorithm is unaware that the goal is to infer an actual network topology. Therefore, the global network structure cannot influence the inference process. For example, relevance networks [13] are created by calculating the mutual information between each possible gene interaction pair. A high mutual information score between a gene pair is then considered as putative evidence of a regulatory interaction. It is well known that this technique predicts a large number of false positive interactions due to indirect effects. Two widely-used methods, CLR [9] and ARACNE [14] acknowledge this weakness and implement strategies to mitigate this problem by incorporating a more global network context. ARACNE uses the Data Processing Inequality on every triplet of genes to filter out the weakest connection. CLR builds a background model for each pair of interacting genes and will transform the mutual information score to its likelihood within the network context. WGCNA [15] also incorporates a global network context in the network reconstruction step of the algorithm. Pairwise correlations are raised to the optimal power to maximally fit a scale-free topology property of the constructed network. In a more general context, Network Deconvolution [16] was proposed as a post-processing technique to infer direct effects from an observed correlation matrix containing both direct and indirect effects. Similarly, a post-processing method named Silencer [17] uses a matrix transformation to turn the correlation matrix into a highly discriminative 'silenced' matrix, which enhances only the terms associated with direct causal links. However, in general and as to date, almost none of the state-of-the-art algorithms make use of general or specific structural knowledge of gene regulatory networks to guide the inference progress. In contrast, such structural properties of GRN and biological networks in general have been studied extensively in literature [18, 19], introducing concepts such as modularity, hub-nodes and scale-free biological networks. The topology of a specific GRN is highly dependent on the experimental conditions and the type of cell [20] although general topological properties have been reported. It has been described that both prokaryotic and eukaryotic transcription networks exhibit an approximately scale-free out-degree distribution, while the in-

degree distribution follows a restricted exponential function [21]. Furthermore, the concept of relatively isolated sets of co-expressed genes under specific conditions, called modules, has been introduced, as discussed in [22]. Topological analyses of GRN have also revealed the existence of network motifs [23], recurrent subgraphs in the larger network which appear more frequent than would be expected in randomized networks. The existence of such network motifs and their frequency of occurrence inevitably has an impact on the global network structure. Finally, prior knowledge about the topology of the specific GRN of the cell at hand can be available, in the simplest scenario in the form of already known regulatory links extracted from literature. We believe that the current non-inclusion of such known structural properties in the inference process leads to predictions that do not achieve their full potential. Furthermore, they are often characterized by different graph-invariant measures than networks found in literature. Although it is hard to completely transform these predictions into more realistic networks, it is clear that the inclusion of structural knowledge is desirable and will be beneficial to the prediction accuracy. However, including such complex and diverse topological information directly in the inference process of existing algorithms is non-trivial and offers little room for modifiability.

Instead in this work, we propose and validate a post-processing approach that aims to be easily modifiable and extendable. The resulting algorithm, named Netter, uses as input any ranking of regulatory links sorted by decreasing confidence obtained by a network inference algorithm of choice. It then re-ranks the links based on graph-invariant properties, effectively penalizing regulatory links which are less likely to be true in the inferred network structure and boosting others. It is not the goal of this work to put forth structural properties of GRN, instead we wish to offer a flexible system in which the end user can decide which structural properties are to be included or emphasized. Expert users can easily design and include novel structural properties and consequently tune Netter to a specific use case. However, to validate our approach, we also introduce and motivate three simple structural properties and default settings which can be used in a more general setting in which specific knowledge of the GRN is unavailable. Using these proposed structural properties and settings, we demonstrate that Netter improves the predictions of six state-of-the-art inference methods on a wide array of synthetic and real gene expression datasets including the DREAM4 and DREAM5 benchmarks. Netter also slightly improves the DREAM5 community prediction of the *E.coli.* network inferred from real expression data. We compare and discuss the performance of Netter with other techniques that aim to incorporate the global network or post-process GRN predictions. Next, we further investigate and discuss the characteristics of improvement of Netter. Lastly, we show that the performance gain of Netter is robust with regard to its parameter settings and the exact definition of its structural properties.

3.2 Materials and Methods

Figure 3.1 shows an overview of the Netter algorithm. In the following subsections, we first formalize the problem statement and elaborate on the different steps of the algorithm. Next, we introduce three different structural properties which will be used to re-rank the input networks. Finally, we discuss the different network inference methods that will be used to create the input networks and briefly discuss the computational aspects of Netter. In the process of formalizing the problem, we will introduce a parameter for every design decision. In practice however, many of these parameters do not substantially influence the outcome of Netter and do not need tuning as we will further discuss in the Results and Discussion section.

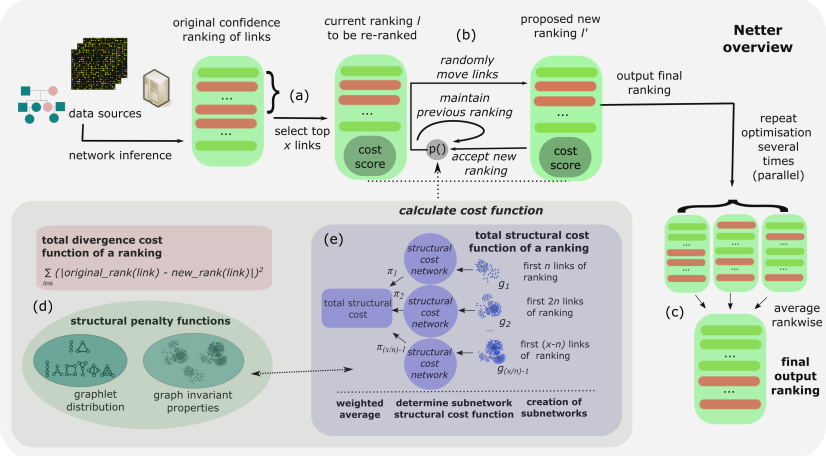

Figure 3.1: **Overview of the re-ranking approach of Netter.** *A ranking of regulatory links sorted by decreasing confidence is assumed. This prediction can be obtained by an inference method of choice using any data source. In a first step, the top x links of the ranking are extracted. Netter will assign these links a new position, whereas the other links maintain their original ranks. The extracted ranking is awarded a cost and using simulated annealing the cost function is minimized several times, obtaining re-ranked lists in the progress which are then averaged to obtain the final output ranking. The cost function strikes a balance between modifying the original ranking to have better structural properties while remaining true to the original ranking.*

3.2.1 Input, problem definition and output

Most GRN inference methods return an absolute ranking of all possible edges, sorted by decreasing confidence that this link is a true regulatory interaction. This ranking is then later transformed into a network representation by selecting a threshold determined by the algorithm or end-user. Netter uses as input such an absolute ranking of potential gene regulatory links. This ranking can be incomplete,

however no regulatory links will be added as Netter is strictly a re-ranking approach. No further assumptions are made about which algorithms or data sources were used. Although we focus here on unsupervised network inference methods which use microarray expression data to infer network topologies, Netter is generally applicable to any method producing a ranking from omics data. In practice, it only makes sense to re-rank the top of the ranking noting that networks consisting of 100 genes already produce a complete ranking of 9900 potential regulatory links (excluding self-regulatory interactions). Therefore, in the first step of the algorithm (Figure 3.1, a), the top x most confident links of the prediction are extracted, where x is a user-chosen value. The algorithm will work on these links only, assigning them a new rank, whereas the remaining links maintain their original ranks and cannot influence the decision process.

3.2.2 Formulation as an optimization problem

Using the extracted top x links, an optimization procedure is started which is performed several times and can be executed in parallel (Figure 3.1, b). Each optimization procedure outputs a new ranking, after which the final ranking of Netter is obtained by averaging rank-wise over all output rankings (Figure 3.1, c). Averaging over multiple output rankings is a crucial step in Netter. It guarantees robustness and performance gain as the total cost function which is optimized is non-convex with many local optima. We will further discuss this in the Results and Discussion section. A single optimization procedure tries to find a ranking $l \in L$, the set of all possible rankings, which minimizes a total cost function f assigning a positive value to any ranking in L.

$$\min_{l \in L} f(l), \text{with } f : L \rightarrow \mathbf{R}^{+}$$

This total cost function is defined as the weighted sum of two cost functions s and Δ with the same (co)domain as f:

$$f(l) = s(l) + \alpha.\Delta(l)$$

Here α is a global balance factor, s is a structural cost function giving a score to a ranking based on structural properties and Δ is a divergence function quantifying how much a ranking is different from the original ranking. Intuitively, f strikes a balance between modifying the original ranking to have better structural properties while remaining true to the original ranking (Figure 3.1, d).

3.2.3 Simulated annealing optimization

This optimization problem is then solved by following a simulated annealing approach to the problem. In a single step of the optimization process, we create a

new ranking l' by randomly moving γ links up or down the ranking by θ positions. γ is sampled uniform from $[1, \Gamma]$ in each step, while θ is sampled uniform for each link from $[-\Theta, +\Theta]$. Θ, Γ being user-set integer values. In practice, this way the optimization process will explore both minor as substantial changes to the ranking. The newly generated ranking l' is accepted as the new ranking if $f(l') < f(l)$ or with a probability of $e^{-(f(l')-f(l))/T}$ otherwise, with T being the current temperature of the annealing process, as proposed by [24]. We use a traditional exponential cooling schedule in which the temperature is repeatedly lowered by a constant factor μ after each iteration. To avoid manual tuning of the annealing parameters for each network, Netter will automatically adjust the parameters and restart if the acceptance ratio of bad mutations during a starting window (e.g. the first 10% iterations) is not within user-defined limits.

3.2.4 Assigning a structural cost function and a divergence cost function to a ranking

In Netter, the structural cost function s assigns a score to a ranking l based on topological properties. We adopt the following procedure (Figure 3.1, e) to transform a ranking into network representations of which structural properties can be calculated. In a first step, a subnetwork g_1, containing n links, is created by using the top n links of the ranking. Next, a subnetwork g_2 is created, containing the first $2n$ links of the ranking. This process is repeated until a subnetwork $g_{[(x/n)-1]}$ is created, containing all but the last n links of the ranking. Summarizing, subnetworks g_1, g_2, …, g_i, $g_{[(x/n)-1]}$ of increasing size are created from the ranking l, consisting of n, $2n$, …, $i.n$, x-n links respectively. We can then calculate a score for each of these subnetworks by using a structural property function s_{struct} which depends on some structural properties. s is then defined as the weighted sum of the structural scores of the individual subnetworks g_i created from the ranking l.

$$s(l) = \sum_i \pi_i . s_{struct}(g_i)$$

The coefficients π_i, each associated with a subnetwork, are set to decrease according to the network size. Smaller subnetworks, corresponding to the top of the ranking are in this way more influential in the total structural cost of the ranking. Intuitively, this way the optimization procedure will make the top of the ranking correspond more to structurally realistic networks by moving links to the top of the ranking which structurally improve the network and move down others which seem odd to be present. As the size of the search space of possible rankings allows for an almost infinite number of rankings which effectively minimize the structural cost function close to its lowest possible value, the divergence function Δ needs to be included in the total cost function f. This function thus acts as a regularization term and is defined as:

$$\Delta(l) = \sum_{link}(|original\_rank(link) - new\_rank(link)|^2)$$

3.2.4.1 Structural property functions

The structural property function s is defined as the weighted sum of individual structural penalty functions s_{struct} which each have a user defined weighting coefficient. The number of structural properties which one could associate with a typical GRN are plenty and are much subject to debate. Furthermore, some structural properties are highly dependent on the cell at hand or the experimental conditions. Therefore, Netter is designed to allow for the easy inclusion and exclusion of new or existing structural properties. Expert users or researchers which have prior knowledge can tune Netter to specific use cases. For example, a custom penalty could be defined which penalizes the non-inclusion of known interactions. It is not the main focus of this work to develop or suggest (complex) structural penalty functions. However, to validate our re-ranking approach, we introduce several general structural properties based on graph-invariant properties and graphlets. In this article, we restrict these functions to be a simple v-shaped mapping of a certain structural property of the network y to a cost value, although Netter can include any function that maps a network to a positive value. The v-shaped function is defined as follows:

$$s_{struct}(y) := \|ay + b\|$$

Here, the parameters a and b can be specific for each of the s_{struct}. In the results section we discuss how changes in the relative weighing coefficients and the exact shape of the individual structural penalty functions (by varying a and b) influence the performance of Netter.

3.2.4.2 Graphlet-based structural penalty

Graphlets have been introduced as small connected non-isomorphic induced subgraphs of a larger network [25]. They differ from the concept of network motifs by the fact that an induced subgraph needs to contain all the edges between its nodes which are present in the parent network. Since the original study, several other graphlet-based network properties have been derived and successfully applied in various applications [26]. If we focus on 4-node graphlets, it is clear that hub-like structures in the network will promote the occurrence of G4 (see Figure 3.2) graphlets. We postulate that the relative frequency of G4 graphlets as compared to all other 4-node graphlets could be used as an optimization criterion to create networks which are more modular. The need for increased modularity can be motivated by the fact that in the inferred networks, the network topology resembles

a full-mesh structure as opposed to a scale-free, modular topology that is generally associated with a GRN. To be precise, we created a graphlet-based penalty function which defines y as the relative percentage of G4 graphlets compared to all other 4-node graphlets. Next, a and b are set in the v-shaped cost function such that a lower cost corresponds to networks with a high relative percentage of G4 graphlets. Including this penalty does not eliminate the possibility of other valid biological structures to appear in the network (e.g. feed forward loops), as strong signals will always be retained due to the divergence cost penalty or other included penalties. This penalty will merely discourage the occurence of weak signals connected to stronger signals (links that appear at the top of the original ranking) that would result in less modular networks. In practice, penalties based on other graphlets can (and are encouraged) to be included in Netter to further refine the desired network shape. One can also include penalties based on subgraph counts in a directed network context (i.e. network motifs). However, for demonstration purposes, we will only include the G4-graphlet relative frequency as optimization criterion as we believe it is the most simple and intuitive criteria. In the Results section, we discuss the default mapping (a and b) and the stability of this penalty.

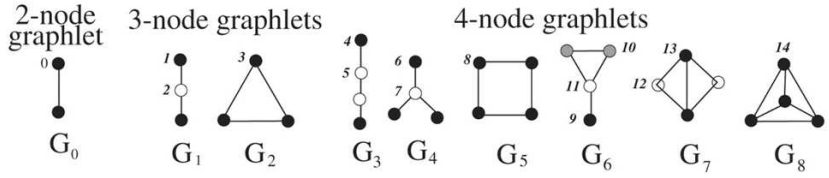

Figure 3.2: **All 3-node and 4-node connected network graphlets.** *Figure adapted from [26].*

3.2.4.3 Regulatory gene limiting penalty

In the case a list of known regulatory genes is not available, as in the DREAM4 benchmark, predictions tend to favor the presence of outgoing links originating from almost every node in the inferred network. This is due to indirect effects, if for example a gene A regulates both genes B and C in the same process, most algorithms will also predict a regulatory link between B and C. Furthermore, in the absence of interventional or time-series data, the direction of a regulatory link is hard to predict resulting in a large number of bi-directional links as both directed edges will usually be close to each other in the ranking. In reality, it is improbable that every gene in the network has an outgoing link in the network, as this would suggest that the gene has a regulatory function. Although the graphlet-based structural penalty partially addresses these problems, a simple regulatory gene limiting penalty was created which defines y as the number of nodes in the network with

at least one outgoing link relative to the total number of genes in the network. Parameters a and b were set such that a high cost was associated with networks containing a high percentual number of nodes that have outgoing links.

3.2.4.4 Anti-dominating penalty

In some cases after re-ranking, we noticed that a regulatory gene and its target genes would completely dominate the top of the prediction list, leaving no room for other modules. This behavior is unwanted, as one wants to discover different areas of the network. This penalty counters this problem by penalizing a percentual large number of links originating from the same gene in the network. The anti-dominating penalty defines y as the ratio between the maximum number of links originating from a same gene in the network and the total number of links in the network.

3.2.5 Computational aspects of Netter

The large search space of possible rankings results in the necessity of performing many steps to minimize the optimization function. Therefore, it is critical that a single step is performed as efficient as possible. Two computationally expensive processes can be distinguished in a single iteration. First, the new candidate ranking l' created from l, needs to be transformed into new subnetworks g_i. Second, structural penalties need to be calculated using the newly created subnetworks of which some, e.g. the graphlet count, can be computationally expensive. Executing both processes from scratch would result in an unacceptable runtime. However, because the new ranking l' is similar to the current ranking l an incremental approach to the problem can be used. Therefore, Netter uses an incremental update scheme to keep track of the subnetworks and can efficiently revert back in case the new ranking is rejected. All penalty functions, including the graphlet enumerator have been defined and developed to work in an incremental way and new structural penalties should also be implemented in this setting. Each optimization procedure in Netter is 'embarrassingly parallel'. Therefore, Netter will assign new optimization runs to each idle, available core. To give an estimate of the execution time of Netter: a typical set-up as described further including 100 independent optimization runs, took 5 single core hours on a Intel i3 CPU M350 clocked at 2.27 GHz, 8.00 GB of RAM and a 64-bit OS. However, the running time is highly dependent on the parameter settings and the list of included penalties. Furthermore, the number of independent runs (=100) is conservative and can be further lowered if computing power is an issue. We discuss this in more detail in the Results and Discussion subsection.

3.2.6 Selected network inference methods

In order to test Netter we performed a large number of experiments using a variety of network inference methods. We selected six network inference methods in total with varying complexity and performance. In addition, in case of the DREAM5 networks, the community prediction networks as created and published in [27] were added. Of the six selected network inference methods, three are based on mutual information scores: CLR [9], ARACNE [14] and BC3NET [28]. Three other methods use machine learning techniques to infer the network GENIE3 [10], NIMEFI [29] and TIGRESS [12].

3.2.7 Selected data sets and evaluation measures

Netter's performance was evaluated using the five expression datasets from the DREAM4 *in silico* 100 multifactorial [27, 30, 31] challenge and the two expression compendia from the DREAM5 network inference challenge [6]. Furthermore, to avoid overfitting specific structural properties of these benchmarks, we created an additional 25 networks of different dimensions and associated expression compendia using two different synthetic gene expression data generators SynTRen [32] and GeneNetWeaver [30, 33]. Table 3.1 provides an overview of the dimensions and properties of the datasets. Using all of these datasets, we inferred the network topology using the algorithms described in the next subsection. Next, we chose a cutoff value x and re-ranked the resulting prediction using Netter. As evaluation measure, we consider both the Area Under the Receiver Operating Characteristic curve (AUROC) and the Area under the Precision-Recall (AUPR) curve, only taking into account the true edges present in the first x predicted links of the original ranking. Gold edges which are not present in this original ranking are removed from the gold standard prior to calculating the scores. This allows for a fair comparison between the original ranking and the re-ranked list as Netter is strictly a re-ranking algorithm and cannot add any edges outside the selected x edges. Furthermore, it allows a more clear comparison between networks of different dimensions. As a result, the AUROC and AUPR scores in this article depend on the original predicted ranking and cannot be compared between different methods. For some of the additional tests, a reduced dataset of 15 networks was used instead of the full dataset to ease the computational demands. This networks in this dataset were randomly selected from the full dataset and contain only GENIE3 predictions. For each test, we will clearly indicate if the full or reduced dataset was used.

Table 3.1: Overview of the datasets used in the performance tests.

Name	Networks	Reg.genes	Genes	Samples	Type
DREAM4	5	100	100	100	Artif.
DREAM5 artif.	1	195	1643	805	Artif.
DREAM5 *E. coli.*	1	334	4511	805	Real
SynTRreN-100	5	100	100	100	Artif.
SynTRreN-150	5	150	150	150	Artif.
GNW-200	15	200	200	200	Artif.

3.3 Results

To interpret the performance results of Netter, it is important to note that from a theoretical point of view, a post-processing approach can never improve every network prediction it is applied on. If this would be the case, repeatedly applying this algorithm on the outcome of a previous re-ranking would eventually result in the perfect ranking. An analogy can be found in lossless compression, where one also tries to find general properties to obtain a good compression ratio for a large set of probable items sampled from the population. In the specific case of Netter, each consecutive re-ranking will result in less information of the original prediction being used to guide the re-ranking process and therefore should be avoided. Furthermore, for a specific network it is hard to explain why a loss in prediction accuracy occurred. A possible reason is that the initial prediction was of insufficient quality to guide to optimization process in the right direction. It is known that these network inference algorithms achieve low accuracy and that algorithms can produce different rankings even with those obtaining similar performance metric scores [6, 29]. Further on in this section, we will briefly discuss the performance gain of Netter with regard to the initial prediction accuracy. Also in the following subsections, we present the results of performance tests, compare Netter to other similar technique, discuss the effect of successive applications of Netter and comprehensively investigate the influence of the various parameters settings and choice of the structural cost function definitions.

3.3.1 Performance tests

We ran Netter on all networks and all method predictions using the following settings. The cutoff value x was set to the first 750 links or the number of non-zero links in the case less edges received a non-zero score. The mutation parameters Θ and Γ were set to 70 links and 50 positions respectively. The subnetwork size parameter n was set to 25 and the associated coefficients π_i were set to $0.5i$, for $i = [1 \ldots \text{number of subnetworks}]$. The annealing scheme allowed an acceptance ratio of bad mutations of approximately 10% after the first 3000 of 30000 iter-

ations. The optimization process was performed 100 times for each prediction before averaging and all penalty functions discussed in the previous section were included. The relative weighing parameter was set to 25 for the graphlet penalty, 2 for the gene regulatory penalty and 75 for the anti-dominating penalty, α was set to 10^{-5}. The influence of the individual penalty cost function shape, the relative weighing coefficients and other parameters on the performance is discussed in the next section. Each re-ranking experiment was repeated three times and, due to the ensemble approach of Netter, the rankings were almost identical.

Figure 3.3 shows the change in AUPR and AUROC compared to the original ranking after applying Netter on all datasets except DREAM5, each dot resembles a re-ranked network. The results show that Netter substantially increases the prediction accuracy in the majority of cases across all algorithms. For the AUROC scores, the boxplot bars remain well above the baseline of 0.0, with only a few re-rankings resulting in a decrease in performance. Looking at the AUPR, the increase in performance is more pronounced compared to the AUROC change, with some re-rankings achieving an increase in AUPR of more than 0.25, which in some cases nets to a relative increase of more than 100% compared to the original ranking. To give a more intuitive view on the accuracy gain, we take a closer look at a network (GNW-network 2) on which a substantial improvement was achieved. Figure 3.4 shows a network comparison view between the original GENIE3 ranking and the re-ranked list in which the first 75 links are plotted. The true positive links are shown as black solid lines, whereas grey curved lines indicate false positives. The resulting networks have 31 out of 75 of their predicted links in common. In the original, there were 36 true positive links, while the re-ranked prediction contains 69. Of the 36 true positives in the original prediction, 28 are still in the re-ranked network while 41 of the 44 new links entering the network are true positives. Further analysis shows that especially the top of the ranking is improved (Figure 3.5). Indeed, for this example the first false positive is encountered at position 50 for the re-ranked list and at position 1 for the original. The fact that the improvement occurs at the top of the ranking is a desirable feature in practice. Focusing on DREAM5, Table 3.2 shows an overview of the AUPR of GENIE3, NIMEFI, TIGRESS and the community network. We did not re-rank the predictions of the mutual information methods, as these methods were outperformed by the former in the DREAM5 challenge. The table shows that the original AUPR score on the artificial network is already quite high and Netter is unable to further improve the prediction. However, on the *E. coli.* network inferred using real expression data, Netter substantially improves the predictions of GENIE3 and NIMEFI while the TIGRESS performance decreases. Netter is also able to slightly improve the community network as produced by the DREAM5 challenge.

Table 3.2: AUPR before and after re-ranking predictions of the DREAM5 dataset.

Net.	GENIE3		NIMEFI		TIGRESS		Community	
	Orig.	New	Orig.	New	Orig.	New	Orig.	New
Artif.	0.94	0.96	0.81	0.82	0.92	0.90	0.91	0.88
E. coli.	0.15	0.21	0.18	0.21	0.20	0.16	0.13	0.15

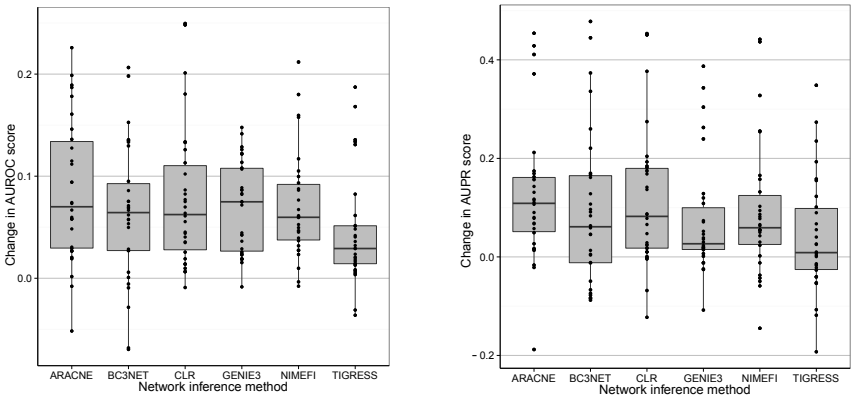

Figure 3.3: **Change in AUROC and AUPR scores after applying Netter.** *Change in AUROC and AUPR scores after applying Netter on all datasets except DREAM-5 which are shown in Table 3.2. The different bars represent the network inference algorithm used to create the initial network. Each dot on the figure is a different re-ranked network and is the result of a single Netter re-ranking procedure consisting of 100 averaged independent optimization runs.*

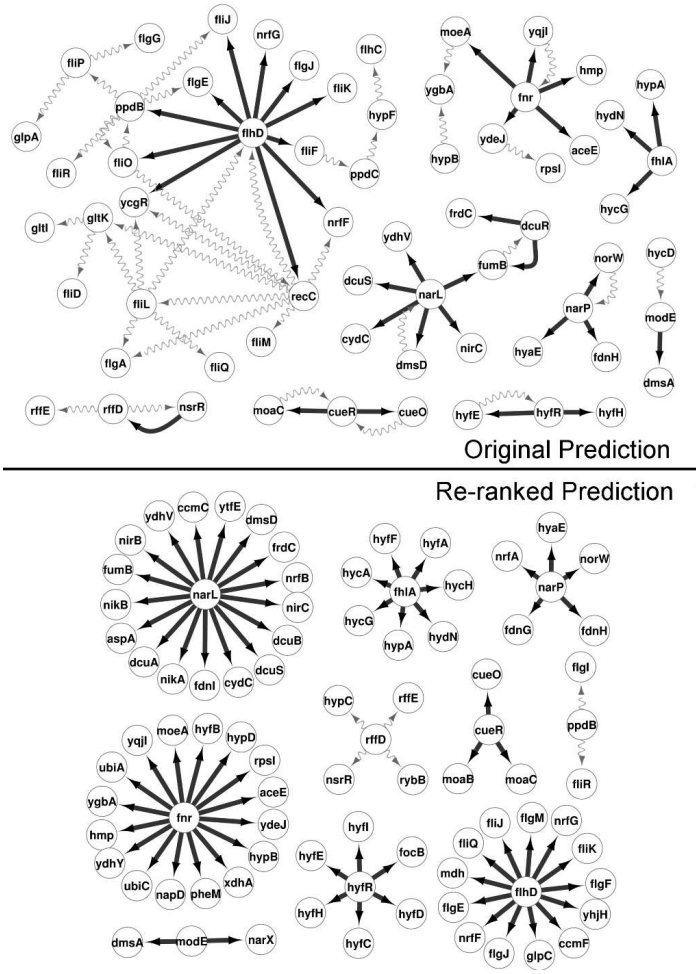

Figure 3.4: **Network comparison view of a GENIE3 prediction before and after the re-ranking procedure of Netter.** *The first 75 links of each ranking are plotted. True positive links are shown as black solid lines, whereas grey curved lines indicated false positives.*

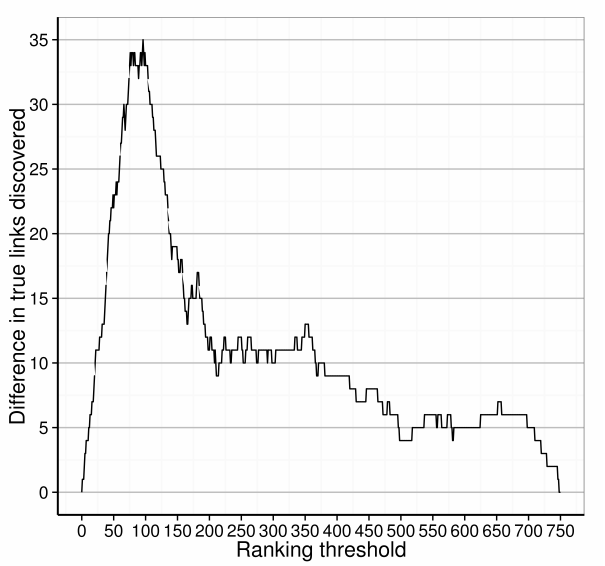

Figure 3.5: **The difference in the number of true links discovered at various thresholds for a re-ranking.** *At every possible threshold of the ranking, the number of true positive links discovered by the original ranking is subtracted from the number of true positive links discovered by the re-ranked network. The network is the same as the one plotted in Figure 3.4.*

3.3.2 Comparing Netter to similar techniques

In this subsection, we will compare Netter with other post-processing approaches for GRN inference predictions and other algorithms that incorporate global network information in their inference process. We are not aware of any other methods that use structural properties of the output network to guide the inference prediction on a large scale. However, as discussed in the introduction, both CLR and ARACNE can be considered as extensions of relevance networks which correct the mutual information (MI) scores using a more global network context. Network Deconvolution and the Silencer on the other hand are post-processing techniques that attempt to separate direct causal effects from indirect effects and have been applied for GRN inference. As mentioned in the introduction, WGCNA raises a pairwise correlation matrix to a certain power to maximally fit the scale-free topology measure. Although, the idea is similar to Netter, both methods cannot be compared directly. WCGNA only changes the edge weight values but does not change the ranking of edges. As baseline for our comparison, we infer networks by calculating MI scores for each pair of genes. Next, we also infer the networks using ARACNE and CLR. For each network, we post-process these three predictions using Netter, Network Deconvolution and the Silencer. This results in twelve different predictions for each network. We use the same full dataset as in the performance tests. Again we use the AUROC and AUPR scores as evaluation metrics, however we do not adopt the pre-processing procedure described in the 'Selected data sets and evaluation measures' subsection, as we are interested in comparing between methods as opposed to relative gains in this test.

Figure 3.6 shows the change in evaluation metric compared to the MI prediction. Each dot resembles a final network prediction. In total 11 boxplots are shown, two include ARACNE and CLR predictions without further post-processing. The remaining nine are post-processed networks of the mutual information, ARACNE and CLR predictions using Netter, the Silencer or Network Deconvolution. The figure shows that the ARACNE method has a higher AUPR score compared to the MI network but at the cost of a decreased AUROC score. This is caused by ARACNE setting a large number of interactions to 0, a more aggressive approach than most other algorithms. CLR both has higher AUROC and AUPR scores than the original MI prediction. These performance gains are to be expected, as both algorithms are widely adopted and have been successfully applied to various use cases. Among the post-processing algorithms, Netter is the clearly the most successful one. Applying Netter results in a substantial improvement for the AUPR score of the ARACNE and CLR predictions as also shown in the previous subsections and a small improvement in AUPR score for the MI network. The smaller gain for the MI network can be explained by the lower accuracy of the initial prediction, as we will further discuss in the following subsection. Netter does not seem to influence the AUROC score of the MI, ARACNE or CLR predictions.

This is because Netter is a conservative approach, only re-ranking the first x (*in casu* 750), allowing no new links to enter the prediction. Applying Network Deconvolution results in a decrease in AUROC and AUPR in all but a few cases for the MI prediction. It has no effect on the ARACNE predictions and lowers the prediction accuracy of CLR in general. The Silencer is able to correct the loss in AUROC score originating from ARACNE but does not have a positive effect in all other cases. The performance of the Silencer has been subject to controversy [34]. Concluding, we believe that Netter compares favorably to other post-processing approaches. Furthermore it has the advantage that it is not limited to correlation-like measures but can be applied to rankings or ranking ensembles of different algorithms.

3.3.3 Characteristics of improvement with regard to the initial prediction accuracy

Figure 3.7 shows the results of the performance tests in a different way. We grouped the 180 re-rankings into 6 equally sized bins using the accuracy of the initial prediction as binning criteria. The y-axis shows the relative gain in AUPR compared to the original prediction (e.g. an original AUPR score of 0.20 which is increased to 0.40 by applying Netter would be plotted at a y-value of 100%). For each bin, the means of the boxplot are well above the baseline of 0 and in less than 30% of the cases the performance is lowered. Furthermore, higher gains (up to 150%) are achieved as opposed to less accurate re-rankings (maximum at -50%). The potency of Netter to improve predictions is at its lowest for predictions which are the least accurate to start with. This makes sense, as Netter relies on the accuracy of the initial prediction to guide its re-ranking process in the right direction. We see a general trend that applying Netter becomes increasingly interesting up to a certain level if the initial accuracy of the prediction is higher. The majority of predictions with the highest initial accuracy have a lower mean improvement, although some of the most accurate initial predictions can still be substantially improved by applying Netter.

3.3.4 Successive applications of Netter

Netter can also be applied on the outcome of a previous Netter re-ranking. Figure 3.8 shows the evolution of the AUPR score of chaining Netter on the reduced test dataset of 15 GENIE3 predictions. A second re-ranking procedure has a mixed effect on the performance, with about as many networks improving in accuracy as predictions becoming less accurate. Further successive applications of Netter result in an accuracy loss in the general case although many networks continue to show an improvement compared to original ranking after 5 re-rankings. The obtained accuracy is comparable to running Netter with increasingly less stringent

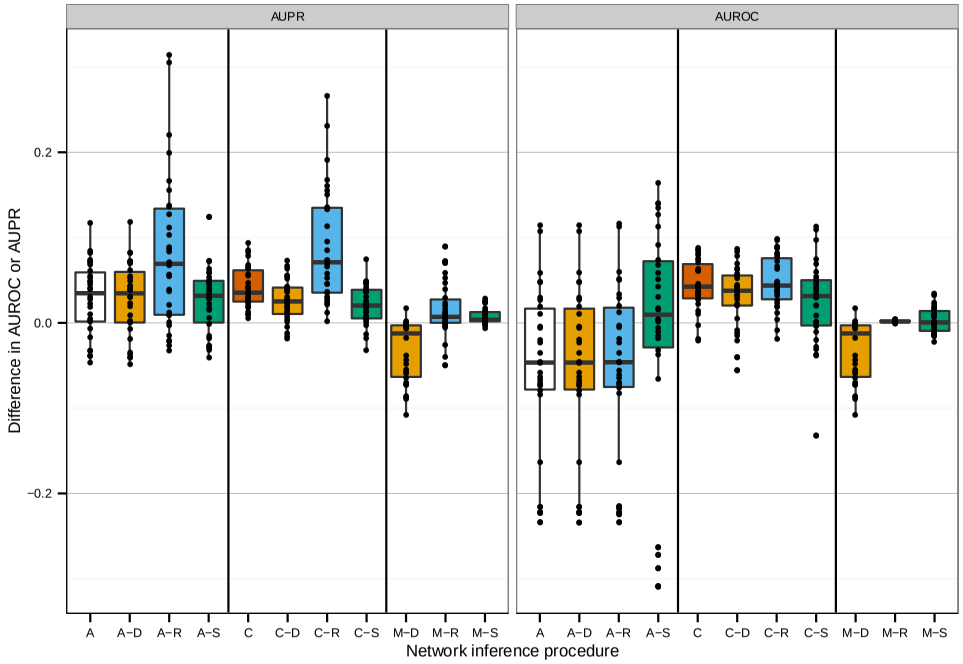

Figure 3.6: **Performance comparison of Netter to similar (post-processing) algorithms.** *(A= ARACNE, C= CLR, M= Mutual information, R= Netter re-ranking, D= Network Deconvolution, S= Silencer. '+' indicates post-processing.). The MI prediction is used as a baseline and the relative difference in AUPR and AUROC of the complete ranking of the other predictions is plotted. Each dot represents a single network prediction.*

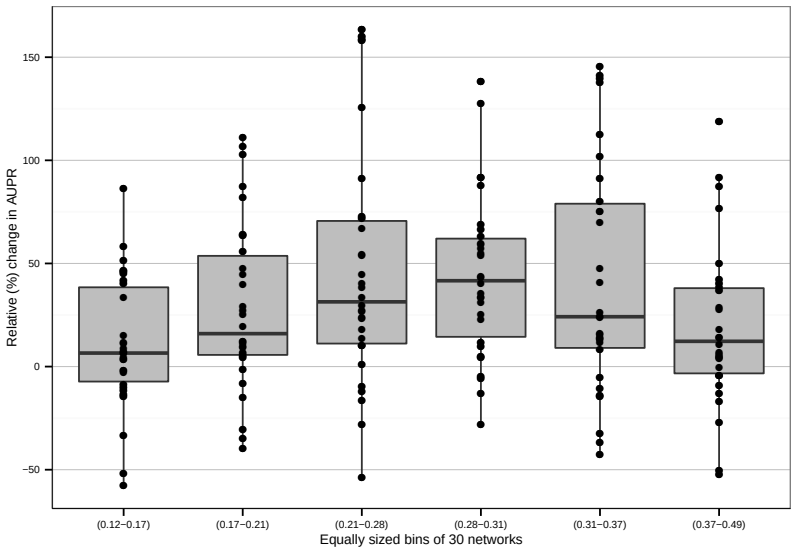

Figure 3.7: **Characteristics of improvement with regard to the initial prediction accuracy.** *Relative (%) change in AUPR of the full dataset is plotted, binned in equally sized groups of 30 networks. In general, Netter's potential to improve the prediction is higher when the initial prediction is more accurate.*

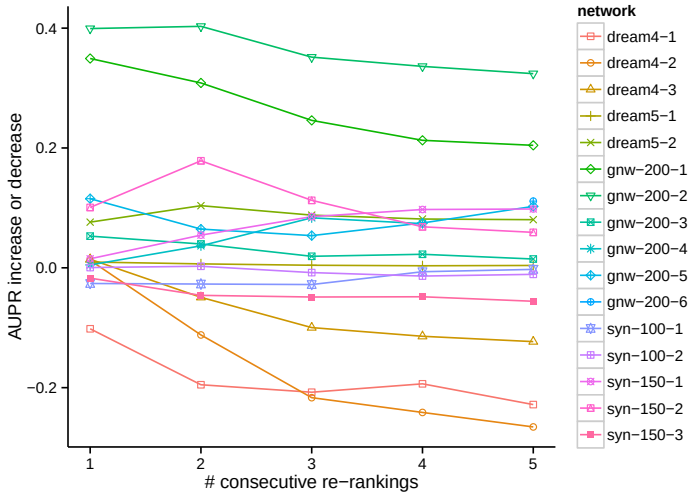

Figure 3.8: **Evolution of the performance during consecutive applications of Netter.** *Netter is consecutively applied using default setting and penalty functions on the reduced test dataset. The performance increase or decrease compared to the original prediction is plotted after each re-ranking.*

regularization penalty (divergence cost function) as the influence of the original ranking is decreased with every re-ranking.

3.3.5 Parameter and structure cost function stability analysis

The large number of parameters which can be set in Netter raises the questions of how one can tune these parameters and how influential these parameters are on the prediction accuracy. Furthermore, one needs to be sure that a small change in the definition of the structural functions does not lead to a large change in the re-ranking accuracy. To address the first question, Netter is equipped with a logger system which can track among others the prediction accuracy, the total cost function, the individual penalty functions and the accept/revert ratio of the simulated annealing process at desired intervals. To address the second question, first the performance tests used a large and diverse dataset: including benchmark data and networks of different dimensions, created by two different simulators to decrease the change of obtaining inflated figures by chance. Secondly, we have performed parameter sweeps by changing the value of one parameter and keeping the other constant. Thirdly, we substituted the default structural cost function mapping for each penalty with three times sixteen other simple structural cost functions with different slopes and intersects by varying a and b. Table 3.3 lists the default parameter settings and explores different values for the structural penalty functions, the balance factor α, the subnetwork size parameter n and associated coefficients π_i. The tables shows the average AUPR over all 15 networks. We discuss the parameter settings and the results of the stability tests in the following subsections.

Table 3.3: Stability tests of α, n, π_i and the relative weights of the structural penalties. The average AUPR score on a subset of 15 GENIE3 predictions is shown and compared to the score using default settings. Parameters not listed were set to default values. (Def. Default Settings, g4=graphlet, r=regulatory, a=anti-dominating)

Def.n=25,π_i=0.5i 0.41	(50,0.5i) 0.41	(75,0.5i) 0.41	(50,0.25i) 0.41	(50,3i) 0.42
Def.α=10^{-5} 0.41	α=10^{-2} 0.37	α=10^{-3} 0.40	α=10^{-4} 0.41	α=10^{-6} 0.42
Def.$g4$=2.0 0.41	$g4$=0.0 0.39	$g4$=1.0 0.41	$g4$=5.0 0.41	$g4$=10.0 0.41
Def.r=25.0 0.41	r=0.0 0.36	r=1.0 0.38	r=15.0 0.41	g=35.0 0.41
Def.a=25.0 0.41	a=0.0 0.40	a=50.0 0.41	a=75.0 0.41	a=100.0 0.41

3.3.5.1 Influence of the number of optimization runs on the convergence of Netter

Netter runs a number of independent optimization runs before averaging and producing the final output ranking. We have shown that using this ensemble method, the output of Netter is robust if the same settings are used. We further explore the stability of Netter with regard to a variable number of independent optimization runs. Figure 3.9 shows 10 runs of Netter using 10, 40, 70 and 100 independent runs before averaging on the *E.coli.* DREAM5 network. All other networks show similar behaviour. It shows that the mean performance gain increases if more optimization runs are performed. The variance between the final re-ranking also decreases with an increasing number of optimization runs. However, the mean performance difference between 10 runs and 100 runs is only 0.01, while the difference with the original ranking evalation is 0.08. Therefore, if computing power is a bottleneck and many networks need to be re-ranked, a reduced number of optimization runs can be used without a large loss in accuracy.

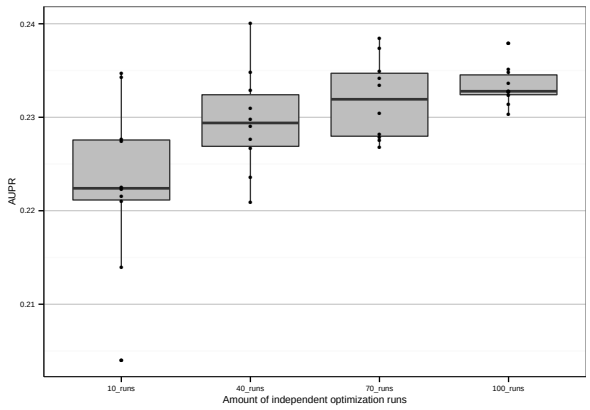

Figure 3.9: **Influence of the number of optimization runs on the convergence of Netter.** *Netter is run ten times with a varying number of independent optimization runs (10, 40, 70, 100). Each dot represents the AUPR of the re-ranked prediction.*

3.3.5.2 Influence of the subnetwork size n and coefficients π_i

When calculating the structural cost function, the ranking is divided into subnetworks of increasing size. The size is determined by the parameter n and the impact on the total structural cost function of a single subnetwork g_i is determined by the associated coefficient π_i. Increasing the subnetwork size will decrease the computation time, as there are fewer subnetworks of which the structural properties need to be tracked. On the other hand, a larger subnetwork size leads to less struc-

tural differentiation options for the different links, possibly resulting in a lower accuracy. Table 3.3 shows the results for varying n and π_i. The performance is stable with regard to the coefficient choice for π_i and the subnetwork size n over a wide range of values. Concluding, we recommend to set n to a small value (e.g. 25 or 1/30 of x) to ease the computational demands but to allow for maximum differentiation, however the choice of n and π_i is not crucial with regard to the performance.

3.3.5.3 Influence of varying the global balance factor α

Probably the most important parameter in the re-ranking algorithm is the parameter α which determines the trade-off between the divergence cost and structural cost of a ranking. If this parameter is set too high, the algorithm will not allow any changes to be made to the original ranking. Whereas if the parameter is set too low, the re-ranking process will not use the original ranking to guide the optimization process. We vary this parameter by setting the values 10^{-i}, with $i = 2 \dots 6$. The results are shown in Table 3.3. For high values of α, the network will only allow small changes to the network, resulting in accuracy which is between the accuracy of the original prediction and the maximum accuracy which can be achieved after re-ranking. Interestingly, the accuracy seems to be stable for the values $i = 4 \dots 6$. We believe this is due to the ensemble approach in which we average over several optimization processes.

3.3.5.4 Influence of varying the relative weight of a individual structure penalty function

The impact of the individual penalty functions on the total structural cost function can be adjusted by changing the associated weights of each penalty function. These weights are typically set by running the algorithm several times with some initial settings and by tracking the individual penalty scores using the logging system. The influence of these parameters is shown in Table 3.3. For all three penalty functions, a performance loss can be seen if the penalty influence is set to zero and as such is not included in the structural cost function. The weight of all three penalties is shown to be robust for a wide range of values, meaning that a small change in this weight does not result in a big effect on the outcome. As a rule of thumb, we suggest that the weights are set using the logger system to values such that all penalties which the user designed and included more or less equally contribute to the decrease in the overall penalty function. This way, the weights of the individual penalty functions seem to have little effect on the accuracy increase of the re-ranking process.

3.3.5.5 Influence of the individual structure cost penalty mappings

In order to test the robustness, we replaced the default v-shaped function ($f(y) = \|ay + b\|$) of each structural penalty in a 4 by 4 grid search. b was set such that the function had zero cost at different values for the structural property y and for each setting of b, four different slopes were selected by varying a. For the graphlet based and the gene limiting penalty, the decrease in average AUPR over the 15 networks was at most 0.02 and corresponded to the setting in which the penalty function was moved furthest from the original intersect. We therefore conclude that these penalties are stable over a wide range of possible mapping definitions. The anti-dominate penalty showed a slightly faster decrease in AUPR if the intersection with the x-axis was moved further to the right. In the extreme case the performance dropped to 0.38 from 0.41. The performance loss is slightly more pronounced because unlike the latter penalties the penalty cost associated with y-values left of the intersect have no meaning, as it does not make sense to discourage rankings which explore different regions of the network. Concluding, the exact shape of all three structural penalty functions is robust and only decreases slowly if the function is moved closer to the inverse function.

3.3.6 Further exploration of the impact of the structural penalty function definition

In addition to the tests in the previous subsection in which we varied the shape and the relative performance of the structural penalty functions, we believe it also important to investigate how Netter behaves in extreme settings. The goal is to both establish some baselines for the performance metrics and to help gain intuition about the presented performance and stability results. In a first test, we excluded all structural penalties and the divergence cost function and again re-ranked the reduced subset of 15 networks. The simulated annealing scheme was altered to accept every proposed ranking. This results in randomly shuffling the ranking for a set number of iterations before averaging the obtained rankings. Table 3.4 shows the AUPR results for 300, 3000 and the default value of 30000 iterations averaged over the standard value of 100 independent optimization runs. This experiment was repeated 10 times and the standard deviation between runs is shown between brackets. The table shows that the performance drops as the number of iterations increases. This is expected, as the initial prediction is more confident about the top of the ranking which would as a result contain more true positive links. Randomly shuffling the ranking would eventually lead to a uniform distribution of the true positive links, resulting in a worse AUPR score and an AUROC score of 0.5. Due to the ensemble nature of Netter, the standard deviation of the performance loss between the final obtained rankings remains small, although the obtained ranking diverges more than in the latter case.

Table 3.4: AUPR results of re-ranking without penalty functions for a set number of iterations. Average values over 10 runs are shown on the reduced test dataset. Standard deviation is listed between brackets.

Initial	Default re-rank.	300 iter.	3000 iter.	30000 iter.
0.34	0.41	0.34 ($\pm$0.01)	0.31 ($\pm$0.02)	0.21 ($\pm$0.02)

In a second test, we modify the structural penalties such that they attempt to optimize the inverse function. For the regulatory gene limiting penalty and the graphlet-based penalty this is achieved by changing the v-shaped function intercept to $1 - b$. The optimization process will then attempt to lower the number of G4 graphlets and increase the numbers of nodes with outgoing edges. We excluded the anti-dominating penalty from these experiments, as the inverse of this function is not well defined. Table 3.5 lists the average AUPR score over the subset of 15 networks.Even in the extreme case in which one uses two inverted functions which are clearly not typical for a gene regulatory network, the accuracy of the prediction remains higher than the randomly shuffled network. This is due to the divergence cost function which attempts to keep the new ranking as close as possible to the original. In case only one inverted function is used, the performance loss is less pronounced, suggesting that other structural properties can counter the effects of ill-chosen penalty functions to some extent. Overall we believe that the performance gain is promising if well-motivated structural properties are used and the performance gain is robust to the exact transformation of the structural property into a penalty function.

Table 3.5: AUPR results of re-ranking with the inverse of the default structural properties.

Initial	Default re-rank.	Inv. graphlet	Inv. regulatory	Both inv.
0.34	0.41	0.32	0.30	0.26

3.4 Discussion

In this work we presented Netter, a novel post-processing algorithm for gene regulatory network predictions. Our algorithm re-ranks a sorted list of predicted regulatory interactions using known structural properties of the network. The algorithm works by defining an optimization problem in which we minimize a weighted sum of desired structural properties and a regularization term penalizing divergence from the original prediction. This optimization problem is solved several times using simulated annealing, after which the obtained networks are aggregated using average rank to obtain the final output. We offer a flexible system in which desired structural properties can be developed and included. Expert users can tune

the system to include specific prior knowledge but we show that by using three suggested more general penalty functions we can obtain a large accuracy gain on benchmark and artificial data. Using these settings Netter outperforms other post-processing methods such as the Silencer and Network Deconvolution. Although our method is heavily parametrized, we have shown that the performance increase is robust for a wide range of values and structural cost penalty functions. Furthermore, especially the top of the ranking is improved by Netter, making our method appealing for practical use. Finally, we have shown that Netter can further improve the DREAM5 community prediction of the *E.coli.* network inferred from real expression data.

References

[1] V. Narendra, N. I. Lytkin, C. F. Aliferis, and A. Statnikov. *A comprehensive assessment of methods for de-novo reverse-engineering of genome-scale regulatory networks*. Genomics, 97(1):7–18, January 2011.

[2] N. Soranzo, G. Bianconi, and C. Altafini. *Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data*. Bioinformatics, 23(13):1640–1647, July 2007.

[3] H. Hache, H. Lehrach, and R. Herwig. *Reverse engineering of gene regulatory networks: a comparative study*. EURASIP J. Bioinformatics Syst. Biol., 2009:8:1–8:12, January 2009.

[4] R. De Smet and K. Marchal. *Advantages and limitations of current network inference methods*. Nat. Rev. Micro., 8(10):717–729, October 2010.

[5] T. Michoel, R. De Smet, A. Joshi, Y. Van de Peer, and K. Marchal. *Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks*. BMC Systems Biology, 3(1):49, 2009.

[6] D. Marbach, J. Costello, R. Kuffner, N. Vega, R. Prill, D. Camacho, K. Allison, The DREAM5 Consortium, M. Kellis, J. Collins, and G. Stolovitzky. *Wisdom of crowds for robust gene network inference*. Nat Meth, 9(8):796–804, August 2012.

[7] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, and J. P. Mesirov. *GenePattern 2.0*. Nat. Genet., 38(5):500–501, May 2006.

[8] R. Küffner, T. Petri, P. Tavakkolkhah, L. Windhager, and R. Zimmer. *Inferring gene regulatory networks by ANOVA*. Bioinformatics, 28(10):1376–1382, May 2012.

[9] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. *Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles*. PLoS Biol, 5(1):e8, January 2007.

[10] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. *Inferring Regulatory Networks from Expression Data Using Tree-Based Methods*. PLoS ONE, 5(9):e12776, 2010.

[11] R. Bonneau, D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, and V. Thorsson. *The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo.* Genome Biology, 7(5):R36, 2006.

[12] A.-C. Haury, F. Mordelet, P. Vera-Licona, and J.-P. Vert. *TIGRESS: Trustful Inference of Gene REgulation using Stability Selection*. BMC Syst. Biol., 6(1):145, 2012.

[13] A. Butte and I. Kohane. *Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements*. In Pacific Symposium on Biocomputing, pages 418–442, 2000.

[14] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano. *ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context*. BMC Bioinformatics, 7(Suppl 1):S7–S7, March 2006.

[15] P. Langfelder and S. Horvath. *WGCNA: an R package for weighted correlation network analysis.* BMC Bioinformatics, 9:559, 2008.

[16] S. Feizi, D. Marbach, M. Médard, and M. Kellis. *Network deconvolution as a general method to distinguish direct dependencies in networks*. Nat. Biotechnol., 31(8):726–733, July 2013.

[17] B. Barzel and A.-L. Barabási. *Network link prediction by global silencing of indirect correlations*. Nat. Biotechnol., 31(8):720–725, July 2013.

[18] A.-L. Barabási and Z. N. Oltvai. *Network biology: understanding the cell's functional organization*. Nat. Rev. Genet., 5(2):101–113, February 2004.

[19] M. G. Grigorov. *Global properties of biological networks*. Drug Discovery Today, 10(5):365–372, 2005.

[20] N. M. Luscombe, M. M. Babu, H. Yu, M. Snyder, S. A. Teichmann, and M. Gerstein. *Genomic analysis of regulatory network dynamics reveals large topological changes.* Nature, 431(7006):308–312, September 2004.

[21] R. Albert. *Scale-free networks in cell biology.* Journal of Cell Science, 118:4947–4957, November 2005.

[22] T. Schlitt and A. Brazma. *Current approaches to gene regulatory network modelling.* BMC Bioinformatics, 8(Suppl 6):S9, 2007.

[23] U. Alon. *Network motifs: theory and experimental approaches.* Nat. Rev. Genet., 8(6):450–461, 2007.

[24] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. *Optimization by Simulated Annealing*. Science, 220(4598):671–680, May 1983.

[25] N. Pržulj, D. G. Corneil, and I. Jurisica. *Modeling interactome: scale-free or geometric?* Bioinformatics, 20(18):3508–3515, December 2004.

[26] N. Pržulj. *Biological network comparison using graphlet degree distribution.* Bioinformatics, 23(2):e177–183, January 2007.

[27] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky. *Revealing strengths and weaknesses of methods for gene network inference*. Proceedings of the National Academy of Sciences, 107(14):6286–6291, April 2010.

[28] R. de Matos Simoes and F. Emmert-Streib. *Bagging statistical network inference from large-scale gene expression data.* PLoS ONE, 7(3):e33624, January 2012.

[29] J. Ruyssinck, V. A. Huynh-Thu, P. Geurts, T. Dhaene, P. Demeester, and Y. Saeys. *NIMEFI: Gene Regulatory Network Inference using Multiple Ensemble Feature Importance Algorithms*. PLoS ONE, 9(3):e92709, 03 2014.

[30] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. *Generating Realistic In Silico Gene Networks for Performance Assessment of Reverse Engineering Methods*. Journal of Computational Biology, 16(2):229–239, February 2009.

[31] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky. *Towards a Rigorous Assessment of Systems Biology Models: The DREAM3 Challenges*. PLoS ONE, 5(2):e9202, February 2010.

[32] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal. *SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms*. BMC Bioinformatics, 7(1):43, 2006.

[33] T. Schaffter, D. Marbach, and D. Floreano. *GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods*. Bioinformatics, 27(16):2263–2270, June 2011.

[34] P. Bastiaens, M. R. Birtwistle, N. Bluthgen, F. J. Bruggeman, K.-H. Cho, C. Cosentino, A. de la Fuente, J. B. Hoek, A. Kiyatkin, S. Klamt, W. Kolch, S. Legewie, P. Mendes, T. Naka, T. Santra, E. Sontag, H. V. Westerhoff, and B. N. Kholodenko. *Silence on the relevant literature and errors in implementation*. Nat. Biotech., 33(4):336–339, April 2015.

"They knew many things but had no idea why. And strangely this made them more, rather than less, certain that they were right."

— Neal Stephenson, Anathem

4

Large-scale network inference of the Immunological Genome Project gene expression data with applications in the unfolded protein response

In this final chapter, we apply gene regulatory network inference methods in a practical setting. We derive networks from a large collection of gene expression measurements and use these networks to find potential new genes involved in the unfolded protein response, a highly conserved pathway which has recently also been linked to immune responses. Wet-lab experiments were performed that were selected using the inferred networks to validate potential new interactions.

⋆ ⋆ ⋆

Joeri Ruyssinck, Lana Vandersarren, Jeroen Creytens , Karl Vergote, Sophie Janssens , Tom Dhaene and Yvan Saeys

Submitted to PLOS Computational Biology

Abstract

The Immunological Genome Project has provided the community with a large and publicly available collection of gene-expression measurements related to the mouse immune system. This large collection was generated using strict protocols and includes many cell types which makes it an unique resource and an ideal candidate to analyze using advanced computational algorithms. Large-scale network inference methods typically focus on extracting high-level biochemical interactions between genes without providing answers on plausible mechanisms and can therefore be applied on a genome-wide level. Recent approaches using techniques from the machine learning community have shown to drastically improve the quality of the predicted networks compared to earlier approaches which calculate pair-wise measures of co-variation. In this work, we hypothesize that the former approaches are currently undervalued and more importantly can be used to extract interactions from gene expression collections unrelated to the original context in which they were performed. We create nine different expression compendia by grouping microarrays based on cell type and infer cell-specific networks from these collections as well as a single network inferred from the entire data. We assess the quality of the networks by discussing two inferred subnetworks in detail. We show that our networks are able to extract non-trivial known connections in both processes related to the transcriptional control programs of interleukin-17-producing helper T-cells and processes related to the unfolded protein response, a highly conserved pathway activated by the accumulation of unfolded proteins in the endoplasmic reticulum. Lastly, based on the inferred network, we identify using qPCR experiments potential downstream targets of a key regulator in the latter pathway. Our results show that indeed multiple genes suggested by the network behave as a downstream target. The inferred networks are offered to the community as a resource for download and can be consulted in the form of an interactive web-interface.

4.1 Introduction

The Immunological Genome Project (ImmGen) [1] is a multicenter, collaborative initiative including immunologists and computational biologists that has built a comprehensive and publicly available database of gene-expression of the mouse immune system. The ultimate goal of the ImmGen project is to establish a complete road-map of gene expression and gene regulatory networks in immune cells. The large number of expression compendia that ImmGen has created offers unique opportunities to apply computational tools and generate novel hypothesis.

During the past decade, the decreasing cost and continuing improvement of high-throughput gene expression measurement techniques such as microarrays and RNA-seq have led to a seemingly unstoppable increase in the number of performed

gene expression experiments. In parallel, there has been a persistent pressure of the community to make available these raw measurements, accompanying the findings that are reported in literature. Further standardization efforts, in the form of guidelines like MIAMA [2] and the establishment of public gene expression repositories such as the Gene Expression Omnibus (GEO) [3] and ArrayExpress [4] have set the stage for massive gene expression recycling [5]. Gene expression recycling hypotheses that novel insights can be extracted by computational tools that analyze gene expression compendia which have originally been created in a different context. Already in 2012, it was reported that some repositories for gene data would hit a milestone of close to a million assays [6]. At the start of 2016, major repositories such as GEO and ArrayExpress were close to doubling that figure. Despite this evolution and clear potential, it remains challenging to aggregate and efficiently (re-)use this huge amount of available data. Added-value databases , e.g. [7], focus on extracting and processing primary data to allow users to efficiently answer research driven questions, usually by means of query based systems and efficient visualizations of the data. The latter frequently comes in the form of networks, as they are well suited to efficiently display large amounts of entities and interactions.

In the context of gene expression, the concept of gene regulatory networks (GRNs) has been well established. These networks abstract the complex process of gene expression regulation in networks of genes interconnected if there is a regulating effect between a certain genes expression products and the expression of another gene. Inference methods for GRN have been described that try to capture GRN of different sizes and levels of abstraction [8, 9]. Two approaches exist: the statistical and the mathematical modeling perspective [10]. Similarly, the terms model-free and model-based have been used in this context as well as methods that combine both [11]. The mathematical modeling perspective aims to emulate a natural system with respect to its dynamical activity. The components have biological interpretations and the conclusions are drawn from the generative nature of the model. Well known examples of these models include Probabilistic Boolean Networks [12] or models based on ordinary differential equations [13]. The statistical approach, in contrast, is model-free and aims to draw conclusions about biochemical interactions between genes or gene products without requiring plausible biological mechanisms. A typical approach of such methods is to calculate pair-wise measures between each pair of genes that represent co-variation. Several measures have been suggested including Pearson correlation, mutual information [14–18] or more recently (feature) importance scores derived from training machine learning models [19–21]. In contrast to mathematical modeling methods, these methods are more scalable and therefore applicable to derive large-scale networks. However, it is harder to include other biological knowledge, e.g. protein-DNA binding events or prior structure knowledge, although some methods have been described [22].

Applying statistical methods to infer large scale GRN can have several applications. A first application is to use the inferred network(s) as input in a differential network analysis [23] or as a starting point for gene prioritization and network driven approaches [24]. However, the most frequent goal of creating GRN is to create a map of suspected gene interactions which can be used to derive novel hypotheses and guide further (wet-lab) validation experiments.

A major challenge in inferring large-scale blueprints is the severe mismatch between the number of possible interactions and genes in the network and the number of available samples (assays). Gene expression recycling is often the only way to slightly alleviate this problem and create sufficiently large compendia. Although it is now possible to collect relevant data in public databases for a certain research goal in mind, the heterogeneity of the data and differing methodologies for its creation becomes the impeding factor to overcome. One of the exceptions is the gene expression compendia generated in the context of the ImmGen project because of the common creation methodology and substantial size. This makes it an unique candidate to conduct large scale network inference.

In related work, networks of the ImmGen data and other immunological data have been inferred. For example, one of the tools that is offered on the consortium website is the Gene constellation tool, which allows the user to query a gene of interest and explore co-expressed genes (in sub-sets) of the data. Operating in a module setting, Ontogenet [25] was developed and applied to the ImmGen data to construct 81 coarse-grained modules and 334 fine-grained modules which link candidate regulators to sets of co-expressed genes. Using other immunological gene expression data Basso pioneered mutual information network inference techniques to create immunological networks [26].

However, the use of more recent and powerful computational approaches to infer networks from specific large datasets has remained largely unexplored in literature despite the potential gain in knowledge and the clear gain in accuracy such methods offer compared to traditional approaches. Such methods have been proven to produce much more accurate networks as compared to techniques based on Pearson correlation and mutual information but come at the price of a severely increased computational cost [27]. Therefore, in this work, we have created compendia from the raw gene expression offered by Immgen and inferred networks for eight different compendia comprising of different cell types and one network inferred from the entire gene expression dataset. For this, we applied inference methods based on ensemble feature importance methods [20]. Applications of more accurate ensemble feature importance methods remain largely unexplored although they are offered side to side to more traditional established methods on e.g. the GenePattern analysis platform [28]. We show that applying such methods to produce networks can offer a unique resource, especially when applied to a large and high qualitative dataset such as the ImmGen expression data. We offer these

networks to the public as a general resource to browse potential biochemical interactions between genes hidden in the data. In addition, we provide a web-interface to quickly browse the networks we provide. To demonstrate the potential of our approach, we zoom in on two inferred subnetworks and compare our findings to relevant literature.

Lastly, we report on the results of wet-lab experiments which were driven by this analysis. Links were explored between mechanisms involved in the unfolded protein response (UPR), a cellular stress related to the endoplasmic reticulum (ER) which is also involved in inflammatory responses [29, 30]. More specific, it is known that the inositol-requiring enzyme 1 (IRE1, also known as ERN1) initiates an atypical splicing of the X-box binding protein 1 (XBP1), resulting in the creation of the transcriptional activator XBP1s which activates *i.a.* genes involved in ER-associated degradation (ERAD). To identify potential downstream targets of XBP1s suggested by our networks, we induce the UPR *in vivo* by addition of tunicamycine and subsequently block the substrate access to the active site of IRE1 by inhibition through 4μ8c. The effects of his procedure were measured on an RNA level by means of qPCR at two time steps and using two different concentrations of 4μ8c.

4.2 Results and Discussion

We discuss two validation cases in which we zoom in on the inferred networks and qualitatively evaluate the networks based on available literature. In a first case, we discuss the transcriptional control programs related to interleukin-17-producing helper T-cells (TH17). In a second case, we discuss parts of the network that are related to UPR, a collection of reactionary mechanisms activated by ER stress. Lastly, we present the results of additional qPCR experiments which were performed in order to further validate potential XBP1s downstream targets which were suggested by the network.

4.2.1 Networks of transcriptional control related to TH17-cells

T helper cells (Th) play a vital role in the adaptive immune system. Interleukin-17-producing helper T (TH17) cells are a subset of T helper cells characterized by their production of the pro-inflammatory cytokine IL17. Pathogenic subsets of Th17 cells have been associated with a multitude of autoimmune diseases [31] and clinical data shows that anti-IL-17 approaches are effective for treatment [32]. Significant efforts have been made to elucidate the transcriptional control program of Th17 cells. Typically, the uniqueness of Th cell lineages is based on signature sets of cytokines and transcriptional regulators. In the specific case of Th17 cells, RORγt (ROCR) serves as a master regulator and works in unison with several key

transcription factors, including the Signal transducer and activator of transcription 3 (STAT3), the Interferon Regulatory Factor 4 (IRF4) and the Basic Leucine Zipper ATF-Like Transcription Factor (BATF) to activate initial differentiation. The MAF BZIP Transcription Factor (MAF) is typically identified as a general repressor in this process.

Two studies in particular have focused on analyzing the transcriptional network at a high temporal resolution and provide regulatory modules and molecular network representations [33, 34]. In addition, several lists of genes were compiled consisting of known and novel regulators that are involved in Th17 transcriptional processes. Based on these articles, we compared a list of literature curated genes to relevant subnetworks present in our inferred networks.

We inspect specific inferred subnetworks for the three cell type compendia: CD4+ T cells, CD8+ T cells and all T cells at two accuracy thresholds (top 6000 links, top 15000 links). We created these subnetworks by extracting the genes ROCR, STAT3, IRF4, BATF and MAF in the network and all of their directly connected neighboring genes.

As the selected nodes are known to be transcriptional regulators in CD4+ T cells, we would expect that for the CD8+ extracted subnetwork fewer genes which are involved in the control program are present in the network and the master regulators are more loosely connected. This is indeed the case, in the subnetworks extracted from the top-15000 networks: both the inferred CD4+ T (135 nodes,304 edges) and the all T-cells (164 nodes,359 edges) network have much higher number of nodes and edges present than the CD8+ cell (23,77) network. In addition, in the CD8+ network, the regulators MAF and RORC are not present, indicating that the algorithm did not predict any links to other genes at this threshold level. Furthermore, both the CD4+ and all T-cell subnetwork consist of a single component, further indicating that the algorithm is predicting close relations between the selected genes.

Inspecting the networks at a more detailed level, several known interactors in the network show up. For example: the interleukin-1 receptor-associated kinase 4 (IRAK4), which plays a critical role Th17 cell polarization and the response to IL-23 [35]; the Inducible T-Cell Costimulator (ICOS) which is necessary for optimal expansion and function of human Th17 cells [36] and the REL proto-oncogene (REL) which is known to control RORC activity by binding to specific promoter sites [37]. In total, for the CD4+ network, 13 known interacting genes are discovered by the subnetwork and 21 in the all T-cell network of which 7 can be found in both networks. On the contrary, 7 genes are found in the CD8+ network (2 in common with all t-cell network, none with CD4+). Figures 4.1 and 4.2 visualize the T-cell network in which the known interacting genes are marked as green.

In addition, we show that the known interactions we are able to recover are

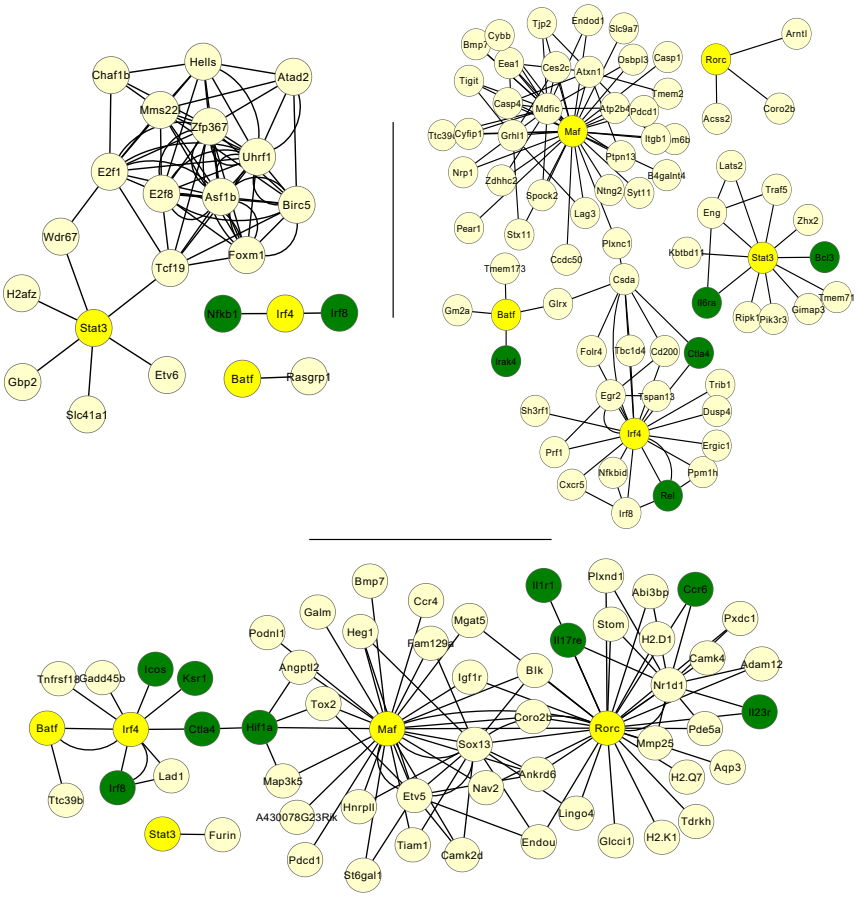

Figure 4.1: **Neighborhood of the T-cell networks at a 6000-link threshold around the genes ROCR, STAT3, IRF4, BATF and MAF.** *Known interacting genes in the process extracted from literature [33, 34] are marked as green. Top-left: CD8+, Top-right: CD4+, Bottom: All T-cell.*

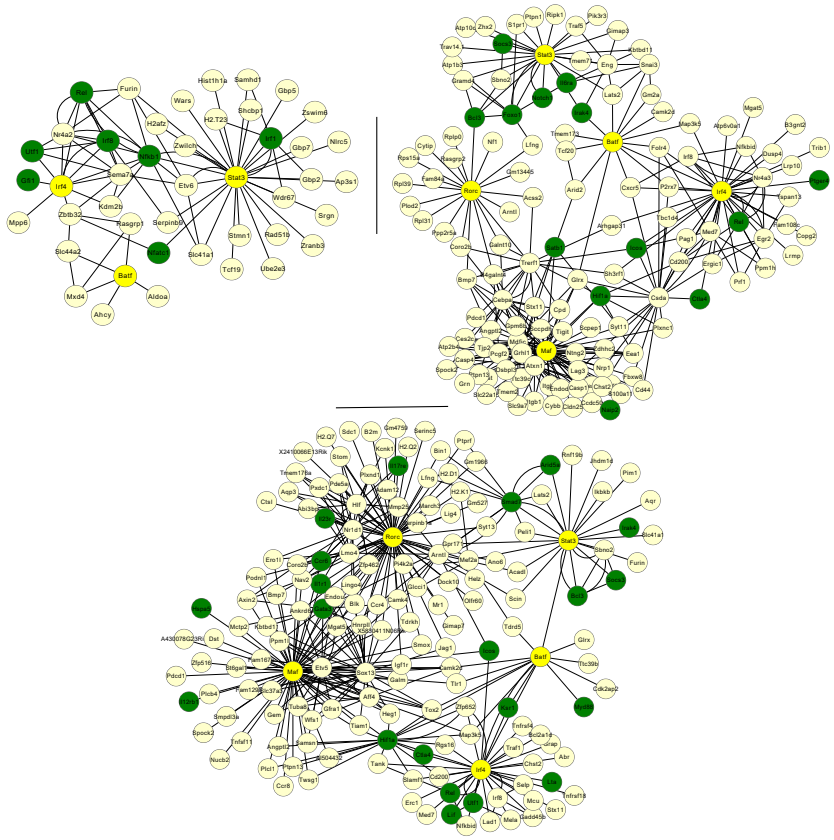

Figure 4.2: **Neighborhood of the T-cell networks at a 15000-link threshold around the genes ROCR, STAT3, IRF4, BATF and MAF.** *Known interacting genes in the process from literature [33, 34] are marked as green. Top-left: CD8+, Top-right: CD4+, Bottom: All T-cell.*

not trivially present in the data. For this, we consult the Gene Constellation tool provided in the ImmGen Data Browser. The Gene Constellation tool provides for a chosen gene the most closely correlated genes according to the tool, default the top 35 are shown. For all of the 31 recovered known interacting genes in the inferred networks, we consulted the Gene Constellation tool and marked if any of five key regulators: ROCR, STAT3, IRF4, BATF or MAF are present in the constellation. If so, this would indicate that the interacting gene would have also been found using a more simple correlation approach and are trivially present in the data. However, only six times does a master regulator appear in the returned list of a queried gene (STAT3 appears in the list of Bcl3, Naip2 and Irf8 ; BATF in the list of Ctla4; RORC in the list of Nfkb1). None of the queried genes list more than one of the master regulators. Furthermore also consulting the lists of the key regulators, none of the other master regulators appears in the returned lists.

4.2.2 Networks related to the unfolded protein response

The unfolded protein response (UPR) is a highly conserved pathway activated by the accumulation of unfolded proteins in the ER and aims to restore normal ER function. More recently, it has also been shown that the UPR signaling pathway plays a vital role in inflammatory responses [29, 30]. In mammals, the UPR is coordinated by three mechanisms. In a first sub-process, inositol-requiring enzyme 1 (IRE1, also known as ERN1) initiates an atypical splicing of XBP1, resulting in the creation of the transcriptional activator XBP1s. XBP1s downstream targets include *i.a.* genes involved in ER-associated degradation (ERAD). In a second pathway the PKR-like ER kinase (PERK) inhibits ribosome assembly, which causes translational inhibition and allows the cell to temporarily manage ER stress. Some mRNa escape the translation block, among them the Activating transcription factor 4 (ATF4) which in turn activates the expression factors involved in the amino acid metabolism, oxidative stress resistance and autophagy, in particular the C/EBP homologous protein (CHOP, also known as DDIT3) and the growth arrest and DNA damage-inducible protein (GADD34/PPP1R15A). In a third mechanism, the Activating transcription factor 6 (ATF6) is transferred to the Golgi compartment where after processing it produces a p50 fragment which migrates to the nucleus. There it activates the expression of several genes involved in ERAD and other ER related processes.

Figure 4.3 shows the subnetwork containing directly connected genes to XBP1 for the subnetwork inferred from the compendia using all cell types. We overlay this network with a list of 88 genes marked as being involved in the UPR pathway by Pathcards [38]. Eight of the 24 directly connected genes to XBP1 are associated to the UPR pathway, among them IRE1/Ern1 and Atf6. This clearly indicates that the inferred network is detecting relevant genes related to UPR pathway as

being connected to XBP1. Further inspection of the network also reveals several other genes not marked by Pathcards but which are known to be directly influenced by either XBP1, ATF4 or ATF6, e.g. the Cyclic AMP-responsive element-binding protein 3-like protein 2 (CREB3L2) [39]. Extracting the subnetwork centered around ATF4 also picks up several genes which are related to the UPR pathway. For example, the DnaJ homolog subfamily B member 9 (DNAJB8/ERDJ4) plays an important role in UPR induced apoptosis [40]. In addition, GADD34 which is activated by ATF4 and restores the translational inhibition initiated by the PERK/ATF4 pathways [41]. In contrast, none of the discovered genes are detected by the Gene Constellation tool.

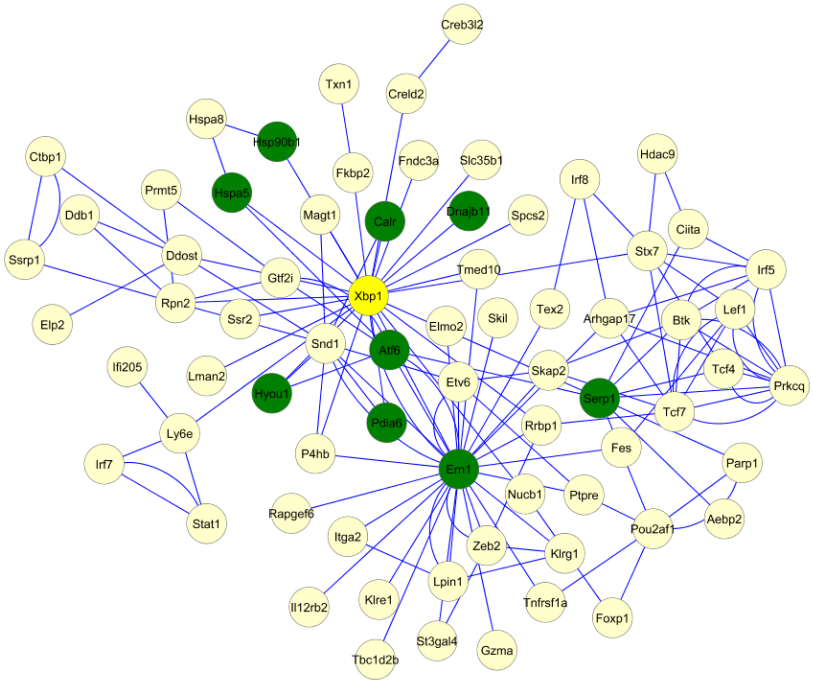

Figure 4.3: **Neighborhood of the network derived from all compendia around the gene XBP1** *Known genes involved in the UPR [38] are marked as green.*

4.2.3 mRNA expression changes in the inferred subnetworks by inducing UPR and inhibiting XBP1 or ATF4

The inferred subnetworks of XBP1 and ATF4 neighboring genes reveal several known key players in the UPR. To further investigate potential downstream targets

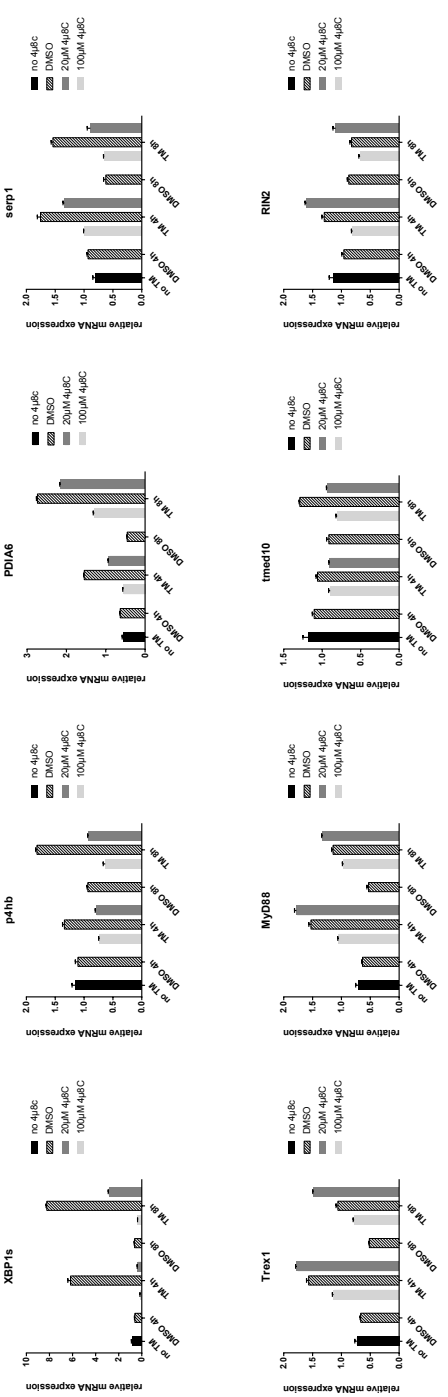

Figure 4.4: **Relative mRNA expression measurements of several selected neighboring genes in the XBP1 network which expression evolution matches that of potential downstream targets of XPB1s.** *The plots show the mRNA expression as determined by qPCR in conditions in which the UPR is induced and XBP1s is inhibited. Two concentrations of the inhibitor 4μ8c at two time points are shown and control conditions include non-induced cells and using only the solvant DMSO.*

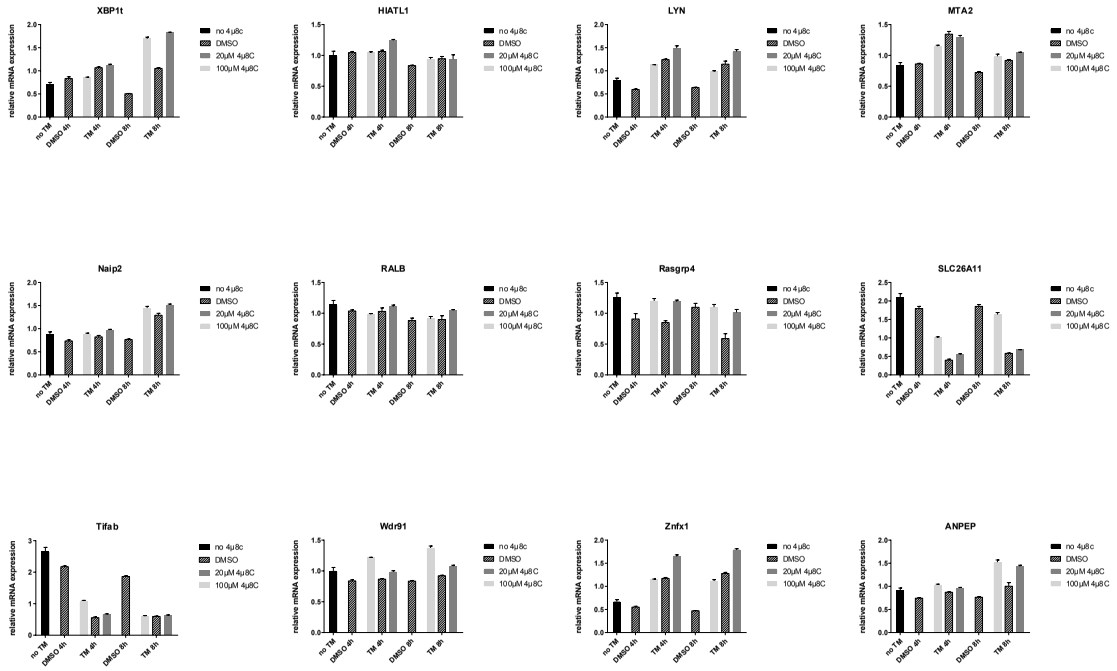

Figure 4.5: **Relative mRNA expression measurements of the remaining selected neighboring genes in the XBP1 network compared to Figure 4.4 which expression evolution does not match those of a downstream target of XBP1s.** *The plots show the mRNA expression as determined by qPCR in conditions in which the UPR is induced and XBP1s is inhibited. Two concentrations of the inhibitor 4μ8c at two time points are shown and control conditions include non-induced cells and using only the solvant DMSO.*

of XBP1s, we analyze the change in mRNA expression level of several selected neighboring genes in the network by means of qPCR after inducing the UPR and next inhibiting XBP1s production. We activate the UPR *in vitro* by administrating tunicamycin which is an N-linked glycosylation inhibitor known to induce ER stress. To validate the predicted links to XBP1, the substrate access to the active site of IRE1 was blocked by 4μ8c. qPCR was performed after 4 and 8 hours after addition of tunicamycin and relative mRNA levels were compared to the solvant, dimethyl sulfoxide (DMSO) with no UPR inducing agents and a sample with not DMSO or inhibitors. Several investigated genes indeed behave consistent with what would be expected from downstream targets of XBP1s. After inducing UPR their expression increases and after inhibiting XBP1s, the expression levels lower again. Figure 4.4 shows the expression profile of the genes that behave in a such matter, as control, XBP1s is also shown. There is a clear such effect for the genes P4HB, SERP1, PDIA6, Trex1, MyD88. For the genes Tmed10 and RIN2, the result is less convincing. All other genes which were included in the experiment are listed in Figure 4.5. Some clear negative examples show up such as Hiatl1 and Ralb. Interesting is SLC26A11 which shows a clear inverse response. Inducing the UPR clearly lowers the expression and after inhibiting XPB1s inverses the process which might indicate that it is target of the regulated IRE1-dependent decay of mRNA (RIDD) pathway.

4.3 Materials and Methods

4.3.1 Collection and inference procedure

The ImmGen raw gene expression data consisting of 679 assays was downloaded from GEO, series GSE1590: Immunological Genome Project data Phase 1. These 679 assays were further divided into eight additional compendia grouped by cell subtypes: dendritic cells (141 assays), b-cells (75), macrophages and monocytes (57), natural killer cells (35), $\gamma\delta$- T-cells (63), CD4+ T cells (52), CD8+ T cells (71), T-cells (255-including the previous three categories). Each of the 8 compendia and a complete compendium consisting of all 679 assays was pre-processed according to the same procedure. Normalization was performed using RMA [42] and in the case of a gene name mapping to more than one probeset, the probeset with the highest mean expression was chosen. Finally, inactive genes were filtered by only retaining the 12000 genes with the most variance across samples.

Next, for each compendia gene regulatory networks were inferred which we offer to community as a resource. We focused on methods with a strong bias towards accuracy at the cost of increase computational demands. The top performing methods in this category are (feature) scores derived from training machine learning models [27]. Of the potential candidates, we chose NIMEFI [20] a generaliza-

tion of the well-known GENIE3 algorithm [19]. Briefly, NIMEFI assigns scores to each possible edge in the complete network by calculating ensemble feature importance scores that are derived by training machine learning regression models on the gene expression data. As additional input for the algorithm, an extensive list of genes was compiled with suspected or validated (transcriptional) gene regulating activity. Only genes present on this list are considered by the algorithm as putative regulators.NIMEFI was run with default settings and using the SVM-variant to reduce computational demands.

4.3.2 Network visualization

Networks are ideally suited to structure and visualize large amounts of data. The network inference procedure results in a complete weighted graph. Each edge from gene A to B and its corresponding weight represents the confidence of the algorithm that gene A has a regulatory effect on gene B. In order to visualize the network, a threshold value is typically chosen and only edges with a weight exceeding this threshold are retained. Choosing such a threshold value is difficult and often is it more instructive to visualize the network using different thresholds. In our web-interface, the networks can easily be plotted at four levels of detail (1000,2000,4000 and 6000 edges). As a list, the top 20000 most confident links can be downloaded for each network. The web-interface is constructed using Cytoscape.js, a powerful Javascript library, sharing concepts and interoperable with the stand-alone application Cytoscape 3.0. It offers users a quick and easy way to browse the inferred networks. First, the user selects the cell type and the number of edges. Next, several options are available to navigate and analyze the network including searching for genes, creating new subnetworks, customizing visualization and perform a layout. Furthermore, each gene in the network can be clicked to consult further information in popular databases.

References

[1] T. S. Heng, M. W. Painter, K. Elpek, V. Lukacs-Kornek, N. Mauermann, S. J. Turley, D. Koller, F. S. Kim, A. J. Wagers, N. Asinovski, et al. *The Immunological Genome Project: networks of gene expression in immune cells*. Nature immunology, 9(10):1091–1094, 2008.

[2] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, et al. *Minimum information about a microarray experiment (MIAME) toward standards for microarray data*. Nature genetics, 29(4):365–371, 2001.

[3] R. Edgar, M. Domrachev, and A. E. Lash. *Gene Expression Omnibus: NCBI gene expression and hybridization array data repository*. Nucleic acids research, 30(1):207–210, 2002.

[4] A. Brazma, H. Parkinson, U. Sarkans, M. Shojatalab, J. Vilo, N. Abeygunawardena, E. Holloway, M. Kapushesky, P. Kemmeren, G. G. Lara, et al. *ArrayExpress a public repository for microarray gene expression data at the EBI*. Nucleic acids research, 31(1):68–71, 2003.

[5] J. Rung and A. Brazma. *Reuse of public genome-wide gene expression data*. Nature Reviews Genetics, 14(2):89–99, 2013.

[6] M. Baker et al. *Gene data to hit milestone*. Nature, 487(7407):282–283, 2012.

[7] R. Petryszak, M. Keays, Y. A. Tang, N. A. Fonseca, E. Barrera, T. Burdett, A. Füllgrabe, A. M.-P. Fuentes, S. Jupp, S. Koskinen, et al. *Expression Atlas update–an integrated database of gene and protein expression in humans, animals and plants*. Nucleic acids research, 44(D1):D746–D752, 2016.

[8] R. De Smet and K. Marchal. *Advantages and limitations of current network inference methods*. Nature Reviews Microbiology, 8(10):717–729, 2010.

[9] Y. Li, S. A. Pearl, and S. A. Jackson. *Gene Networks in Plant Biology: Approaches in Reconstruction and Analysis*. Trends in plant science, 20(10):664–675, 2015.

[10] F. Emmert-Streib, M. Dehmer, and B. Haibe-Kains. *Untangling statistical and biological models to understand network inference: the need for a genomics network ontology*. Frontiers in genetics, 5:1–6, 2014.

[11] V. A. Huynh-Thu and G. Sanguinetti. *Combining tree-based and dynamical systems for the inference of gene regulatory networks*. Bioinformatics, page btu863, 2015.

[12] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. *Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks*. Bioinformatics, 18(2):261–274, 2002.

[13] K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson. *Integrative analysis of cell cycle control in budding yeast*. Molecular biology of the cell, 15(8):3841–3862, 2004.

[14] A. J. Butte and I. S. Kohane. *Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements*. Pac Symp Biocomput, 5, 2000.

[15] R. de Matos Simoes and F. Emmert-Streib. *Bagging statistical network inference from large-scale gene expression data*. PLoS ONE, 7, 2012. Available from: http://dx.doi.org/10.1371/journal.pone.0033624, doi:10.1371/journal.pone.0033624.

[16] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, and R. D. Favera. *ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context*. BMC Bioinforma, 7, 2006. Available from: http://dx.doi.org/10.1186/1471-2105-7-S1-S7, doi:10.1186/1471-2105-7-S1-S7.

[17] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, and G. Cottarel. *Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles*. PLoS Biol, 5, 2007. Available from: http://dx.doi.org/10.1371/journal.pbio.0050008, doi:10.1371/journal.pbio.0050008.

[18] P. E. Meyer, K. Kontos, F. Lafitte, and G. Bontempi. *Information-theoretic inference of large transcriptional regulatory networks*. EURASIP journal on bioinformatics and systems biology, 2007(1):1–9, 2007.

[19] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. *Inferring Regulatory Networks from Expression Data Using Tree-Based Methods*. PLoS ONE, 5, 2010. Available from: http://dx.doi.org/10.1371/journal.pone.0012776, doi:10.1371/journal.pone.0012776.

[20] J. Ruyssinck, V. A. Huynh-Thu, P. Geurts, T. Dhaene, P. Demeester, and Y. Saeys. *Nimefi: Gene regulatory network inference using multiple ensemble feature importance algorithms*. PLoS ONE, 9, 2014. Available from: http://dx.doi.org/10.1371/journal.pone.0092709, doi:10.1371/journal.pone.0092709.

[21] A. C. Haury, F. Mordelet, P. Vera-Licona, and J. P. Vert. *TIGRESS: Trustful Inference of Gene REgulation using Stability Selection*. BMC Syst Biol, 6, 2012. Available from: http://dx.doi.org/10.1186/1752-0509-6-145, doi:10.1186/1752-0509-6-145.

[22] J. Ruyssinck, P. Demeester, T. Dhaene, and Y. Saeys. *Netter: re-ranking gene network inference predictions using structural network properties*. BMC bioinformatics, 17(1):1, 2016.

[23] A. de la Fuente. *From differential expression to differential networking: identification of dysfunctional regulatory networks in diseases*. Trends in genetics, 26(7):326–333, 2010.

[24] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, Q. Morris, et al. *GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function*. Genome Biol, 9(Suppl 1):S4, 2008.

[25] V. Jojic, T. Shay, K. Sylvia, O. Zuk, X. Sun, J. Kang, A. Regev, D. Koller, I. G. P. Consortium, et al. *Identification of transcriptional regulators in the mouse immune system*. Nature immunology, 14(6):633–643, 2013.

[26] K. Basso, A. A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera, and A. Califano. *Reverse engineering of regulatory networks in human B cells*. Nature genetics, 37(4):382–390, 2005.

[27] D. Marbach, J. Costello, R. Kuffner, N. Vega, R. Prill, and D. Camacho. *Wisdom of crowds for robust gene network inference*. Nat Meth, 9, 2012. Available from: http://dx.doi.org/10.1038/nmeth.2016, doi:10.1038/nmeth.2016.

[28] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, and J. P. Mesirov. *GenePattern 2.0*. Nat. Genet, 38, 2006. Available from: http://dx.doi.org/10.1038/ng0506-500, doi:10.1038/ng0506-500.

[29] S. Janssens, B. Pulendran, and B. N. Lambrecht. *Emerging functions of the unfolded protein response in immunity*. Nature immunology, 15(10):910–919, 2014.

[30] J. Grootjans, A. Kaser, R. J. Kaufman, and R. S. Blumberg. *The unfolded protein response in immunity and inflammation*. Nature Reviews Immunology, 2016.

[31] P. R. Burkett, G. M. zu Horste, and V. K. Kuchroo. *Pouring fuel on the fire: Th17 cells, the environment, and autoimmunity*. The Journal of clinical investigation, 125(6):2211–2219, 2015.

[32] S. L. Gaffen, R. Jain, A. V. Garg, and D. J. Cua. *IL-23-IL-17 immune axis: discovery, mechanistic understanding, and clinical testing*. Nature reviews. Immunology, 14(9):585, 2014.

[33] N. Yosef, A. K. Shalek, J. T. Gaublomme, H. Jin, Y. Lee, A. Awasthi, C. Wu, K. Karwacz, S. Xiao, M. Jorgolli, et al. *Dynamic regulatory network controlling TH17 cell differentiation*. Nature, 496(7446):461–468, 2013.

[34] M. Ciofani, A. Madar, C. Galan, M. Sellars, K. Mace, F. Pauli, A. Agarwal, W. Huang, C. N. Parkurst, M. Muratet, et al. *A validated regulatory network for Th17 cell specification*. Cell, 151(2):289–303, 2012.

[35] K. A. Staschke, S. Dong, J. Saha, J. Zhao, N. A. Brooks, D. L. Hepburn, J. Xia, M. F. Gulen, Z. Kang, C. Z. Altuntas, et al. *IRAK4 kinase activity is required for Th17 differentiation and Th17-mediated disease*. The Journal of Immunology, 183(1):568–577, 2009.

[36] C. M. Paulos, C. Carpenito, G. Plesa, M. M. Suhoski, A. Varela-Rohena, T. N. Golovina, R. G. Carroll, J. L. Riley, and C. H. June. *The inducible costimulator (ICOS) is critical for the development of human TH17 cells*. Science translational medicine, 2(55):55ra78–55ra78, 2010.

[37] Q. Ruan, V. Kameswaran, Y. Zhang, S. Zheng, J. Sun, J. Wang, J. DeVirgiliis, H.-C. Liou, A. A. Beg, and Y. H. Chen. *The Th17 immune response is controlled by the Rel–RORγ–RORγT transcriptional axis*. Journal of Experimental Medicine, pages jem–20110462, 2011.

[38] F. Belinky, N. Nativ, G. Stelzer, S. Zimmerman, T. I. Stein, M. Safran, and D. Lancet. *PathCards: multi-source consolidation of human biological pathways*. Database, 2015:bav006, 2015.

[39] Y.-H. Li, G. Tardif, D. Hum, M. Kapoor, H. Fahmi, J.-P. Pelletier, and J. Martel-Pelletier. *The unfolded protein response genes in human osteoarthritic chondrocytes: PERK emerges as a potential therapeutic target*. Arthritis research & therapy, 18(1):172, 2016.

[40] P. van Galen, A. Kreso, N. Mbong, D. G. Kent, T. Fitzmaurice, J. E. Chambers, S. Xie, E. Laurenti, K. Hermans, K. Eppert, et al. *The unfolded protein response governs integrity of the haematopoietic stem-cell pool during stress*. Nature, 510(7504):268–272, 2014.

[41] K. Ameri and A. L. Harris. *Activating transcription factor 4*. The international journal of biochemistry & cell biology, 40(1):14–21, 2008.

[42] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed. *Exploration, normalization, and summaries of high density oligonucleotide array probe level data*. Biostatistics, 4(2):249–264, 2003.

"I think we agree, the past is over."

— George W. Bush

5

Conclusions and future directions

This concluding chapter serves to link together the different contributions presented in this work and provides a critical analysis. Remaining challenges and future directions are respectively discussed and proposed.

This dissertation started by stating that there is an omnipresent belief that large collections of data exist in the world which are hiding a wealth of information. The introduction of this work provided the context which allowed for this state of mind and warned the reader of the potential risks. While there is undoubtedly truth to be found in this statement, it remains very hard to quantify how much this wealth of information is really worth. Several technological advances in the past century have led to technology bubbles and it remains an open question if history will not repeat itself. The rapid progress which is being made in machine learning has allowed for a publish-first, review-later policy in which manuscripts are published on pre-print servers before being peer-reviewed. While this is necessary to maintain the speed at which machine learning is evolving, it also makes it even harder to quantify success stories and less fruitful undertakings.

In our work, we embraced the former hypothesis and put forth that the large collections of gene expression measurements which are being collected and made publicly available in large databases are an example of such potential untouched goldmines. Chapter four served to validate this hypothesis and chapters two and three were technical advances that paved the way to allow this.

In chapter two of this dissertation we proposed a novel framework which takes

as input multiple snapshots of gene expression and extracts regulatory effects between genes in the form of networks. We have shown using benchmark data and additional artificially generated data that our methods achieves state-of-the-art performance and achieves much higher accuracy than traditional methods which are more frequently applied in practice. However, we are not the first to have presented methods in this area and several other novel methods have been developed which achieve accuracies which are in the same ballpark. Therefore, the question remains why less accurate techniques and even simple correlation networks remain more popular than these more recent advances. One of the answers is undoubtedly fear of change. These methods are being applied, they do their job and they are often much more simple to grasp than for example the techniques we present. There is no incentive to change as they are often a small part of the puzzle and not the main focus of the work that is being done. One of the main challenges that remain open for further work are studies which are able to convince the general audience that the gain in accuracy of more advanced methods is worth the hassle. While we have demonstrated this partially in chapter 5, this trade-off between complexity and accuracy is not new and is not going anywhere soon. Per definition, it is a subjective decision and several groups have voiced their opinions that the more simple techniques are adequate. In order to allow comparative studies which can make these types of choices a more objective matter, the bigger question is: how can we evaluate such methods? Objective metrics are currently being used but are hard to translate to what exactly they mean in practice. As all gene regulatory network inference algorithms are inaccurate, what does it exactly mean that one technique is less inaccurate than the other?

In chapter three of this work we proposed a novel post-processing strategy, Netter, which allows known network characteristics to be included in the prediction process. The algorithm was specifically designed to be very flexible and allow any kind of information related to the network structure to be included. We have shown that by using a conservative approach and very simple and widely accepted network properties of gene regulatory networks we can achieve much more accurate predictions on both benchmark data and many additional compendia of artificially generated data. This work has by far been received the most varied of critical responses in both peer-review as during discussions at international conferences. Similar to the discussion in the paragraph above, it was exactly the flexibility of the system which was considered to be a weakness instead of a strength of the algorithm. Indeed, flexibility in this setting means being able to press a lot of buttons and requires a certain grasp of what the algorithm does. The majority of people in the field agreed that the inclusion of network characteristics is a key direction to explore in order to further boost the accuracy of network inference methods. However, some were not convinced that the method works despite the empirical evidence. Other post-processing methods which have been suggested during the

past years, often published in top journals, take the responsibility away from the end-user and at most leave a single parameter to be set. Although we show in our work that we outperform all of these suggested methods, they remain more widely used. This brings us again to same open question which was posed in the previous paragraph: how can we convince the community that in some settings it is worth to put more effort in the computational approach and use more complex algorithms? From a more technical point of view, future work related to chapter three partially answers this question. A potential direction to expand this work would be to apply it in other settings not related to gene expression measurements to further prove its merit. Netter is not any way restricted to gene regulatory networks and could also be applied in other network settings, e.g. social network analysis. Another open direction would be the exploration and development of other network properties which can be used to further refine the predictions. A last suggestion would be to analyze what the effect is of including known regulatory interactions in the network inference process. Netter could easily be applied in such a setting by associating a high cost with moving known interactions out of the network.

In chapter four of this thesis, we applied gene regulatory network inference to mine for insights in a specific process associated to immune responses. We gather and process gene expression data provided by the Immunological Genome Project (ImmGen), a multi-center, collaborative initiative composed of immunologists and computational biologists. This data is an exceptionally large collection of microarray experiments which were all performed using the same strict protocol. This makes it an unique and extremely potent dataset to apply our methods on. In our analysis we show by comparing versus literature that the networks we infer using our methods produce much more accurate results than networks which have been inferred in this setting by the consortium using correlation-like metrics. In addition, our networks were used to set-up wet-lab experiments to validate potential new regulatory effects in a specific immune response pathway. Our results confirm several known actors in this process and reveal potential new players. Concluding, we believe our work is an example that there is indeed potential to reanalyze gene expression measurements outside of the context of which they were originally generated.

Unrelated to any specific chapter of this thesis further open challenges remain to analyze the performance and potential of the discussed methods on novel gene expression measurement techniques. These techniques are rapidly evolving and different platforms exist which might influence the specific characteristics of the data. Another open issue which this work did not address was how computational techniques can be made more 'user-friendly' to attract a wider audience.

"The popular stereotype of the researcher is that of a skeptic and a pessimist. Nothing could be further from the truth! Scientists must be optimists at heart, in order to block out the incessant chorus of those who say: "It cannot be done.""

— Anonymous

A

Random Survival Forests for Predicting the Bed Occupancy in the Intensive Care Unit

In this Appendix, we present an application of machine learning techniques on data collected from patients admitted at the Ghent University Hospital. Our goal is to predict the occupancy of the intensive care unit ward in order to avoid capacity problems and the need for reactive measures related to planned admissions. We propose a group-based approach as opposed to predicting the length of stay of individual patients. Our approach is based on a predictive model derived from Random Survival Forests.

⋆ ⋆ ⋆

Joeri Ruyssinck*, Joachim Van der Herten*, Rein Houthooft, Femke Ongenae, Ivo Couckuyt, Bram Gadeyne, Kirsten Colpaert, Johan Decruyenaere, Filip De Turck, Tom Dhaene

Published in Computational and Mathematical Methods in Medicine, Sep. 2016. (*contributed equally)

Abstract
Predicting the bed occupancy of an intensive care unit (ICU) is a daunting task. The uncertainty associated with the prognosis of critically ill patients and the random arrival of new patients can lead to capacity problems and the need for reactive measures. In this chapter, we work towards a predictive model based on Random Survival Forests which can assist physicians in estimating the bed occupancy. As input data, we make use of the Sequential Organ Failure Assessment (SOFA) score collected and calculated from 4098 patients at two ICU units of the Ghent University Hospital over a time period of four years. We compare the performance of our system with a baseline performance and a standard Random Forest regression approach. Our results indicate that Random Survival Forests can effectively be used to assist in the occupancy prediction problem. Furthermore, we show that a group based approach, such as Random Survival Forests, performs better compared to a setting in which the length of stay of a patient is individually assessed.

A.1 Introduction

In recent years there is an increasing trend to automatically monitor, collect, process and store clinical parameters of patients during their hospital visit. These automated systems have led to the creation of a vast array of heterogeneous and often incomplete data collections which are hypothesized to contain a wealth of hidden knowledge. However, these data compendia, often dubbed 'big data goldmines' in popular media, have mainly been left untouched due to the inherent difficulty they present to (manually) extract information. In this chapter, we present our work to create a system which can assist physicians to predict the bed occupancy of an intensive care unit (ICU) based on automatically monitored clinical parameters.

Predicting the number of free beds in an ICU is a difficult task as there is a substantial amount of uncertainty associated with the prognosis of critically ill patients. This prediction is further complicated by the fact that there is a constant arrival of new patients which unexpectedly require intensive care [1]. Nowadays, ICU physicians generally plan only a single day ahead based on clinical judgement whether or not a patient will leave the ICU. This can lead to situations in which capacity problems arise and planned surgeries have to be postponed. The development of an automated system which can assist physicians in these matters would clearly be beneficial to plan better and further ahead. This in turn could help to reduce the financial cost associated with an intensive care unit. The latter impact should not be underestimated, as it was reported that the cost of care in 2005 for critically ill patients accounts for about 0.66% of the gross domestic product in the United States [2].

In this work, we use machine learning techniques that are trained with recorded Sequential Organ Failure Assessment (SOFA) scores [3] in order to estimate bed

occupancy given the current population of ICU patients. The SOFA score is an established ICU scoring system which assesses the individual degree of organ failure in six organ systems on a daily basis. ICU scoring systems are often automatically gathered and presented at a regular interval to provide physicians a condensed overview of the status and evolution of a patient. Other frequently used scoring systems include i.a., APACHE II [4] and SAPS II [5]. A strong connection has been reported between the evolution of the SOFA score and the mortality of patients in ICU wards [6].

In related work, significant efforts have been made to create computational models which either predict the precise length of stay (regression models) or the risk of a prolonged stay (classification algorithms) of patients in the ICU using various clinical parameters. Kramer *et al.* [7] have developed methods to identify patients with an increased risk for a prolonged stay at the ICU based on data collected during the first five days of admission. Meyfroidt *et al.* [8] applied Gaussian processes to predict the length of stay of 960 patients undergoing cardiac surgery using data monitored during the first four hours of admission. Similarly, several studies have used Artificial Neural Networks to predict the length of stay of cardiac patients in different settings [9–11].

Houthooft *et al.* [12] applied machine learning techniques on SOFA score data to predict individual patient mortality, length of stay and prolonged stay in the ICU. They conclude that the individual length of stay of a patient is hard to predict and propose a split of patients in a two by two grid, based on the mortality risk of the patient and the probability of a prolonged stay. Verburg *et al.* [13] performed a large comparison of regression models to predict the length of stay of unplanned ICU admissions, they also conclude that it is difficult to predict using only patient characteristics at admission time.

In contrast to these studies, we will not focus on predicting the individual length of stay of a patient, but develop a system that predicts the evolution over time of the bed occupancy of an entire ICU given its current population. By opting to model and predict a group of patients instead of predicting at the individual level, we aim to reduce the variability and improve the total accuracy prediction of bed occupancy predictive models. For this, we link the bed occupancy problem to the domain of survival analysis and propose a predictive model using Random Survival Forests [14]. We compare our approach to a standard regression approach in which Random Forests [15] are used to predict the length of stay of each patient.

Note that in order to predict the exact bed occupancy at a time t, the arrival of new patients should also be forecasted. However, without actions by physicians (e.g., postpone surgeries), the arrival rate at the ICU is not affected by the number of occupied beds. An estimation of the evolution of available beds based on the current population of patients is required to anticipate problems, and act to control new admissions by either reserving additional capacity or moving patients to

other hospitals or other wards. These actions are triggered by the ability to detect capacity problems early given the number of planned surgeries, the average inter-arrival time of critically ill patients and the forecasted number of occupied beds by the current patients. The approach described in this article aims to deliver a better estimate of the latter aspect.

The remainder of this chapter is organised as follows: Section A.2 discusses the collection of the data, the scoring system (SOFA) and the applied methods from Machine Learning. In Section B.3 these methods are compared and the results are further analysed.

A.2 Materials and Methods

In this section we first discuss the properties of the dataset: how it was collected and how the data was further processed. Next, we proceed by introducing the algorithms we use in our work. Finally, we discuss the evaluation metric.

A.2.1 Data extraction

The data concerns all adult patients admitted between January 1, 2009 and September 17, 2013 at two ICU units of the Ghent University Hospital. In total 14,480 patient records were extracted including both monitored values, patient information as well as lab results. ICU stays of over 16 days were rare and did not occur often in our data, hence these patients were excluded as they are more the focus for models that attempt to predict prolonged stays. Furthermore, a large group of patients stays in the ICU for a very short time span. Such patients, with a length of stay of less than 3 days, were also omitted for two reasons: the first reason being that these patients are often recovering from uneventful surgeries and are in good condition, allowing the physician to make an accurate prediction for a short length of stay. The second reason being that the computational model we describe further will use data gathered from the previous three days to make a prediction, therefore the model would need to operate with insufficient data. In addition, ICU planning is mostly concerned with the remaining length of stay for patients with a longer stay. Note that our choice to predict based on 3 days of admission data is short in comparison to other studies (Kramer *et al.* [7], Houthooft *et al.* [12]). Finally, only the clinical parameters related to the SOFA scoring system were retained. In total, this lead to 4098 unique patient records.

A.2.2 Dataset processing: SOFA score calculations

The SOFA-score (Sequential Organ Failure score) [3] is one of several ICU scoring systems and tries to capture the status of a patient's organ function. The score is

in general used to determine the patient's status and evolution throughout his ICU stay and is calculated at a daily interval. The total SOFA-score is the sum of six SOFA subscores which are indicators of: the coagulation, renal, cardiovascular, respiratory, liver and central nervous system function. For each of the subsystems a score between 0 and 4 is awarded, with a high score being indicative for organ failure. SOFA scores were automatically calculated by an automated system at the hospital each day at 5AM using all available clinical parameters up to 24 hours upfront.

The coagulation subscore is calculated by measuring the minimum number of thrombocytes (number of platelets $\times 10^3/\mu l$) in the blood of the patient (α_{coag}) which is then mapped to a SOFA-subscore:

$$\texttt{SOFA}_{\text{coag}} = \begin{cases} 4 & : \alpha_{\text{coag}} \leq 20 \quad 3 \quad : \alpha_{\text{coag}} \leq 50 \\ 2 & : \alpha_{\text{coag}} \leq 100 \quad 1 \quad : \alpha_{\text{coag}} \leq 150 \\ 0 & : \alpha_{\text{coag}} > 150 \end{cases}.$$

The renal function SOFA score is based on two input values: the maximum amount of plasma creatine measured [mg/dl] (α_{ren}) and the sum of the urine volume [ml] (β_{ren}):

$$\texttt{SOFA}_{\text{renal}} = \begin{cases} 0 & : \alpha_{\text{ren}} < 1.2 \qquad\qquad 1 \quad : \alpha_{\text{ren}} < 1.9 \\ 2 & : \alpha_{\text{ren}} < 3.4 \qquad\qquad 3 \quad : \alpha_{\text{ren}} < 4.9 \vee \beta_{\text{ren}} < 500 \\ 4 & : \alpha_{\text{ren}} \geq 4.9 \vee \beta_{\text{ren}} < 200 \end{cases}.$$

The liver subscore is calculated by measuring the maximum bilirubine serum value [mg/dl] (α_{liver}) within the 24 h window:

$$\texttt{SOFA}_{\text{liver}} = \begin{cases} 4 & : \alpha_{\text{liver}} \geq 12 \quad 3 \quad : \alpha_{\text{liver}} \geq 6 \\ 2 & : \alpha_{\text{liver}} \geq 2 \quad 1 \quad : \alpha_{\text{liver}} \geq 1.2 \\ 0 & : \alpha_{\text{liver}} < 1.2 \end{cases}.$$

To calculate the nervous system subscore, the minimum value of the Glasgow coma score (α_{cns}) is used. In case the patient was sedated in the last 24 hours, the last known value of the Glasgow coma scale is used:

$$\texttt{SOFA}_{\text{cns}} = \begin{cases} 4 & : \alpha_{\text{cns}} \leq 6 \quad 3 \quad : \alpha_{\text{cns}} \leq 9 \\ 2 & : \alpha_{\text{cns}} \leq 12 \quad 1 \quad : \alpha_{\text{cns}} \leq 14 \\ 0 & : \alpha_{\text{cns}} > 14 \end{cases}.$$

The respiratory function SOFA score is calculated by measuring the minimum PaO_2/FiO_2-ratio (PF) [mm Hg] (α_{resp}) and by checking whether or not the patient was ventilated (V) in the last 24h period:

$$\texttt{SOFA}_{\text{resp}} = \begin{cases} 4 & : \alpha_{\text{resp}} \leq 100 \wedge V \quad 3 \quad : \alpha_{\text{resp}} \leq 200 \wedge V \\ 2 & : \alpha_{\text{resp}} \leq 300 \qquad\quad 1 \quad : \alpha_{\text{resp}} \leq 400 \\ 0 & : \alpha_{\text{resp}} > 400 \end{cases}.$$

To calculate the cardiovascular system function SOFA score, first the mean arterial pressure (map) (α_{cardio}) is calculated. Also, the maximum amount of administered doses of the following drugs is extracted in [μg/kg/min]: dopamine (dop), dobutamine (dobu), epinephrine (epi) and norepinephrine (nor):

$$\texttt{SOFA}_{\text{cardio}} = \begin{cases} 4 & : \text{dop} > 15 \vee \text{epi} > 0.1 \vee \text{nor} > 0.1 \\ 3 & : \text{dop} \in]5, 15] \vee \text{epi} \in]0, 0.1] \vee \text{nor} \in]0, 0.1] \\ 2 & : \text{dop} \in]0, 5] \vee \text{dobu} > 0 \\ 1 & : \text{map} < 70 \\ 0 & : \text{otherwise} \end{cases} .$$

A.2.3 Dataset definition

A dataset was created using the processed data in which the remaining length of stay of a patient at day d of his or her stay, measured in days, is defined as the target variable. Each patient in the processed data is split into one or more dataset entries. The predictor values of a single entry are:

- the SOFA-values of the patient during the previous 3 days (6×3 features),
- the number of days the patient has been admitted to the ICU, (1 features).

For each patient, dataset entries are created at the start of each day the patient is admitted and this starting at day $d = 4$, up to the morning before discharge. This approach can be regarded as a sliding window over the stay of each patient, creating an entry for each step of the window. The additional parameters summarizing the number of days the patient has already been admitted encodes important information on the status of the patient, together with the SOFA scores.

The final dataset contains 19 predictor variables and 11662 entries, generated from 4023 patients. Missing values were imputed as zero because this indicates that according to the doctor's judgement the organ was healthy and no monitoring was necessary.

A.2.4 Random Forests

The Random Forests (RF) is an ensemble learning technique used frequently for supervised learning problems (regression and classification). It consists of an ensemble of decision trees trained on a training set of labeled data points. After training a random forest, an unseen data point is predicted as the mode of the output values of the trees (classification) or the mean of the predictions of the trees (regression).

The algorithm constructs the forest as follows: training is initiated by drawing B bootstrap samples from the original data set: for each bootstrap the excluded data is referred to as Out-Of-Bag data (OOB). Next, a decision tree for each bootstrap sample is grown: the root node is split into two daughter nodes. These

daughter nodes are recursively split until no new daughter nodes can be formed because those would no longer hold more than a predefined minimum number of data points, or when the tree exceeds the maximum depth specified. These extreme nodes which can no longer be split are referred to as *terminal nodes*. The terminal nodes of a decision tree b are denoted by the set $T(b)$. To split a node, a random set of the candidate variables p is chosen, and an optimal split value c is calculated by optimizing an information metric. Popular metrics are the Gini impurity, the information gain, variance reduction or the residual sum of squares. The latter metric is used for the experiments in this work. A more in-depth overview is given in [16].

For regression, each terminal node h of a tree b has an associated value $F(h)$. In case h contains only a single training data point, the value corresponds to the value of the training point. If several training points are associated with the node, the average is taken. To predict an unseen data point x with tree b, the point is first dropped through the tree until it ends in a terminal node h. The prediction of a tree node can be written as:

$$H_{b,h}(x_i) = \begin{cases} F(h) & x_i \in h \\ 0 & \text{otherwise} \end{cases},$$

the predicted output of the RF ensemble of trees can then be written as

$$H_e(t, x_i) = \frac{1}{B} \sum_{b=1}^{B} \sum_{h \in T(b)} H_{b,h}(x_i).$$

Note that dropping x_i through a decision tree ends in a single terminal node due to it's binary nature.

A.2.5 Baseline approach

A second method to predict the Length of Stay (LOS) of a patient is also included in the results section. In this baseline method, the predicted LOS for a patient x is obtained by sampling a random patient from the data and selecting its LOS value. As the random numbers are drawn from the data distribution rather than a different distribution, this provides a competitive baseline which acts as a reference for the results obtained with RSF and RF.

A.2.6 Survival analysis and Random Survival Forests

Survival analysis concerns a class of statistical methods for analysing 'time to event'-data which occurs in a number of research fields such as medicine, biology and engineering. An event can for example be a failure of a component in engineering or in the case of treatment of ill patients, death. A typical feature of this

type of data is that it often contains censored or truncated data observations. For example, right censored data points are observations where all available information is that the event did not occur yet at a given time. Random Survival Forests (RSF) were introduced by [14] as a forest ensemble learner for the analysis of right-censored data. The length of stay of a patient at the ICU can be considered as an unusual instance of censoring in which entering the ICU is considered to be a 'birth' event, leaving the ICU a 'death' event and the death of an individual is considered to be right censoring.

The RSF-algorithm initiates similarly to RF, by drawing B bootstrap samples from the original data set. For each bootstrap sample a survival tree is grown: nodes are recursively split until no new daughter nodes can be formed because those would no longer hold more than $d_0 > 0$ unique death events. Good splits maximize the survival difference between the daughters nodes, in the experiments this was determined using the log-rank splitting rule [17]. The values in the terminal nodes are given by a Cumulative Hazard Function and are time dependent. For a terminal node h of a survival tree b at time t, this is given by the Nelson-Aalen estimator:

$$N_{b,h}(t) = \sum_{t_{l,h} \leq t} \frac{d_{l,h}}{I_{l,h}}$$

which sums the ratio of the number of deaths $d_{l,h}$ and the individuals at risk $I_{l,h}$ over all distinct time events $t_{l,h}$ prior or equal to t. All cases of h are assigned the same CHF. To compute the ensemble CHF of the survival forest with B trees for a given d-dimensional case x_i:

$$H_e^s(t, x_i) = \frac{1}{B} \sum_{b=1}^{B} \sum_{h \in T(b)} H_{b,h}^s(t, x_i) \tag{A.1}$$

with $H_{b,h}^s(t, x_i)$ equal to the Nelson-Aalen estimator if x_i ends in h when dropped through the survival tree:

$$H_{b,h}^s(t, x_i) = \begin{cases} N_{b,h}(t) & x_i \in h \\ 0 & \text{otherwise} \end{cases}$$

It can be observed that the main point of difference with the traditional RF are the values associated with each terminal node. These are given by the Nelson-Aalen estimator and depend on the time t.

A.2.7 Using Random Survival Forests to predict the ICU bed occupancy over time

In order to predict the bed occupancy of the ICU over time, we use the following approach: define the survival function $S(t)$ as the probability that a patient will still be admitted at the ICU:

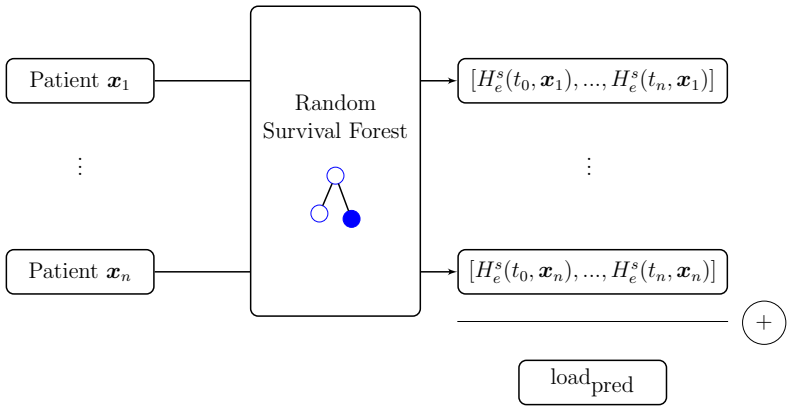

Figure A.1: **Schematic illustration of the computation of $\text{load}_{\text{pred}}$, given a set of patient data.**

$$S(t) = P(\{T < t\}). \tag{A.2}$$

Note that this survival function is reflected by the Nelson-Aalen estimator as used by RSF. We now approach the bed occupancy problem as follows:

1. Grow a Random Survival Forest using training data.
2. For a test set of patients currently admitted at the ICU, drop each patient down the forest.
3. Extract the individual predicted survival functions for each patient by evaluating the obtained CHF's (Equation A.1) over a time interval.
4. Sum these survival functions at each timepoint and consider the obtained vector, $\text{load}_{\text{pred}}$, as the expected number of patients at each time point.

Figure A.1 illustrates how $\text{load}_{\text{pred}}$ is obtained from a set of patient data, using a trained Random Survival Forest (steps 3 and 4).

A.2.8 Goal and Error definition

Ultimately, our goal is to develop and assess a computational model which can predict the occupancy of the ICU for the next several days. This model will use the predictor variables corresponding to the patients currently admitted to the ICU. As described in the previous section, these are the SOFA scores of the previous three days and the number of days the patient has been admitted. As output, the predicted occupancy of the ICU at each day is to be returned, as if no new patients would be admitted.

To assess the accuracy of such a method, standard 10-fold crossvalidation was used, including a modification to calculate a more intuitive measure error measure E for practical use. For each test fold, a set of 25 dataset entries is sampled without replacement, of which $\text{load}_{\text{pred}}$ is predicted using the trained forest. This set can be considered as a random occupation of the ICU with a group of patients with varying length of stay and days admitted. Next, the mean absolute error between $\text{load}_{\text{pred}}$ and the actual $\text{load}_{\text{real}}$ is calculated. As a final step, $\text{load}_{\text{pred}}$ and $\text{load}_{\text{real}}$ are truncated at day 10, as in practice the prediction accuracy of the models after this time period is less important. Note that the truncation occurs after calculating $\text{load}_{\text{pred}}$ and $\text{load}_{\text{real}}$, hence it does not influence the models to predict beyond day 10 if desired. This process is repeated 100 times and averaged to obtain E. Using this scheme E can be interpreted intuitively as the average error of the occupancy over time in an average sized ICU of 25 available beds in the first 10 days.

A.3 Results and Discussion

In this section we first evaluate our approach using the error measure described in the previous section. In addition, we further investigate which clinical parameters the algorithms indicate as important to make the prediction.

A.3.1 Performance evaluation

We compare our proposed RSF approach with both a standard Random Forest regression (RF) model trained to predict the individual length of stay of a single patient, and the baseline approach.

Figure A.2 plots the error measure E for all algorithms in a boxplot overview. Each individual dot in the boxplot represents a single result of E in one of the 10 folds. To support the observations in the remainder of this section, we verified their statistical significance by means of the Wilcoxon rank sum test with continuity correction (cut-off p-value ≤ 0.05).

A first observation is that the Random Forest regression approach performs worse than the baseline. We believe this is due to the fact the RF method is not able to accurately model and predict the individual length of stay of a patient with the given data. Instead, the model resorts to predicting the average length of a stay of a patient in many cases to minimize the prediction error for every individual, which leads to inaccurate predictions at the group level. By focussing on a group based approach that produces a day by day estimate by summing individual predictions this problem could be avoided. The RSF-method does this, as probabilities are predicted on the individual level for each point in time by the survival functions.

A second observation is that the RSF approach performs significantly better

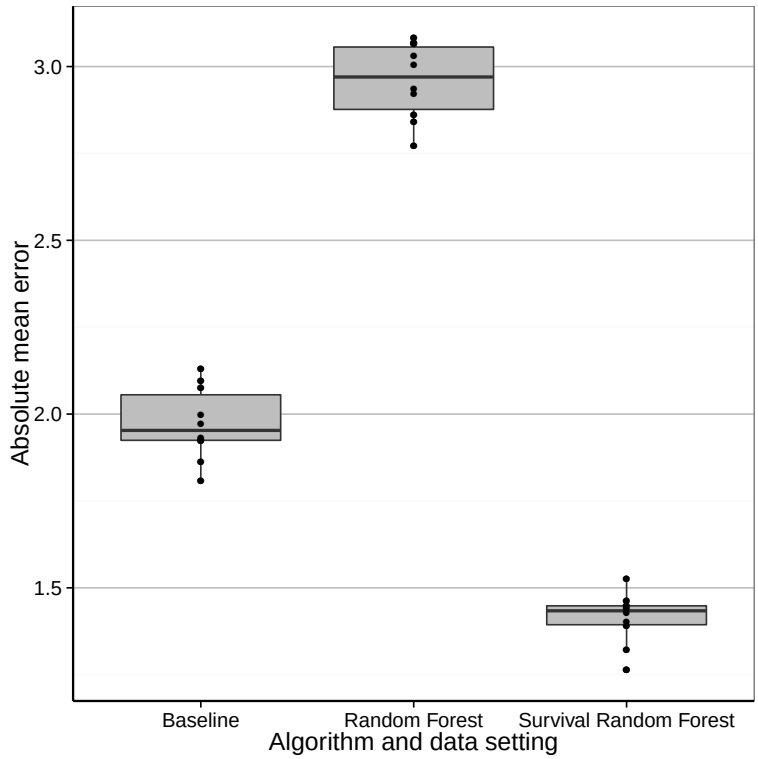

Figure A.2: **Boxplot for Error measure E for the three methods considered: Baseline (B), Random Forests (R), Random Survival Forests (S).**

than the baseline approach. This indicates that it is possible to construct machine learning models that use SOFA-scores to better predict the bed occupancy. In our opinion, this is a strong result as two random groups of 25 patients in the average case in the same ward should not lead to big differences in bed occupancy over time. Therefore any method that can significantly improve on the latter, is a useful tool in practice.

A.3.2 Contribution of the variables to the predictive power of the model

Both Random Forest and Survival Random Forests can provide further insight in which variables, i.c., SOFA-scores, are informative to make predictions. In the setting of both Regression Random Forests and Random Survival Forests, the Breiman-Cutler permutation variable importance measure [15] has been proposed. Briefly, this measure is obtained by comparing the normal OOB prediction error to the OOB prediction error when a certain feature x is randomly permuted for each tree and this further averaged over all trees. These measures were calculated for both algorithms and were further normalized to obtain relative scores as plotted in Figure A.3. A first observation which can be made is that the $\texttt{SOFA}_{\text{resp}}$ measure at day 3 is especially informative in both algorithms. Furthermore, both algorithms also indicate $\texttt{SOFA}_{\text{CNS}}$ at day three as a major second contributor. Both algorithms also indicate that the number of days the patient has already been admitted is a major predictor for the remaining length of stay. Further down the ranking, the relative importance is more spread across all features. This difference is more pronounced in RSF with some variables, e.g. $\texttt{SOFA}_{\text{coag}}$ not contributing to the predictive power of the model. Next, a clear trend is that the information available at day 3 is generally more informative than the scores at day 2 and 1. This is to be expected, as they provide the most up to date information on the condition and evolution of the patient.

A.4 Conclusion

Previous studies have concluded that modelling the individual length of stay of patients admitted in the ICU is a non-trivial task. In this work, we have explored and proposed a group based approach to the problem by constructing a novel data-driven algorithm to predict the overall bed occupancy of an ICU. Our approach uses Random Survival Forests to create individual survival functions which are then summed over time to obtain the overall bed occupancy prediction at each day. We have shown that this approach is to be preferred over a standard regression approach and performs significantly better than a baseline approach. Finally, we have also shown that the respiratory and nervous system SOFA scores together

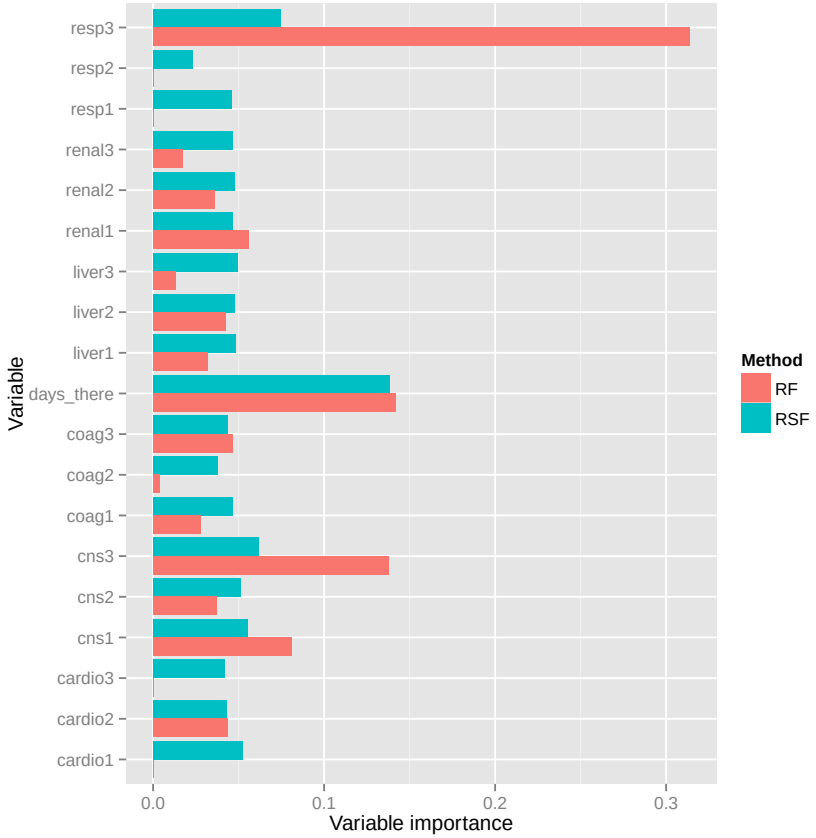

Figure A.3: **The relative variable importance measure for the Random Forest and Random Survival Forest.**

with the number of days the patient has already been admitted are the most important measures, and that the information of day 3 is most influential.

References

[1] T. D. Quinn, R. A. Gabriel, R. P. Dutton, and R. D. Urman. *Analysis of Unplanned Postoperative Admissions to the Intensive Care Unit*. Journal of intensive care medicine, 2015.

[2] N. A. Halpern and S. M. Pastores. *Critical care medicine in the United States 2000–2005: An analysis of bed numbers, occupancy rates, payer mix, and costs*. Critical care medicine, 38(1):65–71, 2010.

[3] J.-L. Vincent, R. Moreno, J. Takala, S. Willatts, A. De Mendonça, H. Bruining, C. Reinhart, P. Suter, and L. Thijs. *The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure*. Intensive care medicine, 22(7):707–710, 1996.

[4] W. A. Knaus, E. A. Draper, D. P. Wagner, and J. E. Zimmerman. *APACHE II: a severity of disease classification system.* Critical care medicine, 13(10):818–829, 1985.

[5] J.-R. Le Gall, S. Lemeshow, and F. Saulnier. *A new simplified acute physiology score (SAPS II) based on a European/North American multicenter study*. Jama, 270(24):2957–2963, 1993.

[6] J.-L. Vincent and R. Moreno. *Clinical review: scoring systems in the critically ill*. Critical care, 14(2):207, 2010.

[7] A. A. Kramer and J. E. Zimmerman. *A predictive model for the early identification of patients at risk for a prolonged intensive care unit length of stay*. BMC medical informatics and decision making, 10(1):1, 2010.

[8] G. Meyfroidt, F. Güiza, D. Cottem, W. De Becker, K. Van Loon, J.-M. Aerts, D. Berckmans, J. Ramon, M. Bruynooghe, and G. Van den Berghe. *Computerized prediction of intensive care unit discharge after cardiac surgery: development and validation of a Gaussian processes model*. BMC medical informatics and decision making, 11(1):1, 2011.

[9] R. J. LaFaro, P. Suryanarayana, K. P. Kubal, M. E. Inchiosa, V. M. Pothula, S. C. Yuan, D. Maerz, L. Mentes, S. M. Oleszkiewicz, A. Yusupov, R. Perline, and M. A. Inchiosa Jr. *Neural Network Prediction of ICU Length of Stay Following Cardiac Surgery Based on Pre-Incision Variables*. PLoS ONE, 10(12):1–19, 12 2016.

[10] P.-F. J. Tsai, P.-C. Chen, Y.-Y. Chen, H.-Y. Song, H.-M. Lin, F.-M. Lin, and Q.-P. Huang. *Length of Hospital Stay Prediction at the Admission Stage for Cardiology Patients Using Artificial Neural Network*. Journal of Healthcare Engineering, 2016, 2016.

[11] M. Rowan, T. Ryan, F. Hegarty, and N. O'Hare. *The use of artificial neural networks to stratify the length of stay of cardiac patients based on preoperative and initial postoperative factors*. Artificial Intelligence in Medicine, 40(3):211–221, 2007.

[12] R. Houthooft, J. Ruyssinck, J. van der Herten, S. Stijven, I. Couckuyt, B. Gadeyne, F. Ongenae, K. Colpaert, J. Decruyenaere, T. Dhaene, et al. *Predictive modelling of survival and length of stay in critically ill patients using sequential organ failure scores*. Artificial intelligence in medicine, 63(3):191–207, 2015.

[13] I. W. Verburg, N. F. de Keizer, E. de Jonge, and N. Peek. *Comparison of regression methods for modeling intensive care length of stay*. PloS one, 9(10):e109684, 2014.

[14] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. *Random survival forests*. The annals of applied statistics, pages 841–860, 2008.

[15] L. Breiman. *Random forests*. Machine learning, 45(1):5–32, 2001.

[16] L. Rokach and O. Maimon. *Top-down induction of decision trees classifiers - A survey*. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 35(4):476–487, 2005.

[17] M. LeBlanc and J. Crowley. *Survival trees by goodness of split*. Journal of the American Statistical Association, 88(422):457–467, 1993.

"We live only to discover beauty. All else is a form of waiting."

— Kahlil Gibran

B

Early detection of sepsis through the prediction of positive blood cultures using long short-term memory neural network

In this Appendix, we present a second application of machine learning techniques on data collected from patients admitted at the Ghent University Hospital. We explore the potential of a BiLSTM, a specific kind of temporal neural network, to predict whether a blood culture will return positive or negative. A blood culture test is indicative of the presence of bacteria or fungi in the blood. This is an abnormal condition which can potentially lead to sepsis, a severe immune response which is the leading cause of death in the intensive care unit of well-developed countries. Early detection of sepsis to allow targeted treatment is vital for the prognosis of patients.

⋆ ⋆ ⋆

Leen De Baets*, Joeri Ruyssinck*, Johan Decruyenaere, Filip De Turck, Femke Ongenae and Tom Dhaene

Submitted to Artificial Intelligence in Medicine. (*contributed equally)

Abstract
Blood cultures are often performed in the intensive care unit (ICU) to detect bloodstream infections and identify pathogen type, further guiding treatment. Early detection is essential, as a bloodstream infection can lead to sepsis, a severe immune response associated with increased risk of organ failure and death. The early clinical detection of a bloodstream infection is challenging but rapid targeted treatment, with in the first place antibiotics, substantially increases survival chances. As blood cultures require time to incubate, early clinical detection of sepsis using physiological signals combined with abnormal lab values is pivotal. In this work, we construct and explore the potential of a temporal computational model to predict the outcome of a blood culture test based on nine clinical parameters measured over time. We investigate whether our model can identify early signals which warrant further testing and compare our model to non-temporal models. We use a bidirectional long short-term memory neural network (Bi-LSTM), a type of recurrent neural network well suited for tasks where the time lag between a predictive event and outcome is unknown. Evaluation is performed using data from 2177 ICU admissions at the Ghent University Hospital. The Bi-LSTM network achieves, on average, an area under the receiver operating characteristic (ROC) curve of 0.95 and an area under the precision-recall (PR) curve of 0.75. In addition, our results show that predictions several hours upfront are possible with only a small decrease in accuracy. In this setting, it outperforms a non-temporal, traditional neural network. Our proposed computational model accurately predicts the outcome of blood culture tests using nine clinical parameters. Moreover, it can be used as an early warning system to detect patients at risk of bloodstream infection.

B.1 Introduction

The rapid development of computing technologies has already had a major impact on healthcare, especially in intensive care units (ICUs). This technological growth has been accompanied by an increase in the complexity of monitoring equipment, all the while generating large amounts of data that need rapid and accurate interpretation by medical staff [1]. Databases have become an essential part of ICUs for storing, integrating and sharing patient data. This data has a temporal nature, so time series often make an appearance, containing valuable and actionable medical information. However, it is difficult for physicians to continuously monitor these time-dependent parameters for subtle or sometimes even overt changes that suggest a relevant clinical deterioration due to a complication or new pathology because of the large amount of data and staff shortage. To resolve this, hospitals are looking more and more towards incorporating data analytics to process the data and provide meaningful insights. These insights support physicians in the swift assessment and treatment of a patient's medical condition.

Sepsis is a whole-body inflammatory response known as systemic inflammatory response syndrome (SIRS) due to infection. Septic shock is defined as the presence of sepsis combined with hemodynamic instability and vasopressor need despite fluid resuscitation. In the US, approximately 750,000 people are diagnosed with sepsis every year, with a mortality rate of 30% [2–4]. Despite medical advances such as antibiotics and vaccines, sepsis remains a primary cause of death among critically ill patients [5], and the number of related hospitalizations has increased three-fold in the last decade [6]. It is clear that early detection is crucial, as appropriate therapy aimed at the prevention of further organ failure can reduce mortality by 15% [7]. Sepsis due to a bloodstream infection can be accurately diagnosed by the presence of a positive blood culture for bacteria or fungi [8]. Growth of these organisms in the bloodstream can lead to septic shock and death [9]. Mortality can be significantly reduced with prompt antibiotic treatment. Symptoms indicative of a bloodstream infection are not specific and often subtle and therefore can cause delayed awareness. Twenty-five possible indicators of sepsis were put forth and listed in medical guidelines that should be monitored to enable early detection [10]. A lot of these indicators are based on physiological data and are time-dependent. For instance, subtle changes in some of these metrics of vital organ functions have been shown to be prognostic of sepsis, even within their normal range [2].

Patients admitted to the ICU suffer from diverse and severe conditions. Many of these afflictions result in severe inflammatory responses of the body which can occur without the presence of an infection. Therefore, it is challenging to detect clinical signals which can diagnose between patients which suffer from a grave illness with inflammatory responses without infection and those whose inflammatory response is partially triggered by an underlying infection.

This paper implements machine learning techniques to allow the accurate identification of patients with positive blood cultures, using physiological parameters continuously monitored within the ICU. Such a detection system could then alert physicians to early warning signs indicating positive blood cultures and possibly sepsis, allowing for a timely intervention. Such a computational model needs to be able to detect clinical signals in the monitored data which are specific to a (blood) infection and are not general indicators of a severe inflammatory response.

B.1.1 Related work

Several studies have used machine learning for the early detection of sepsis. Ho *et al*. [11] implement a predictive model using logistic regression on the publicly available MIMIC II Database [12]. Their model uses parameters such as age, gender, sequential organ failure assessment (SOFA) scores, and four physiological parameters measured within 6 hours of admission. An area under the curve (AUC)

of 0.823 was achieved. They also constructed a model to predict septic shock using additional parameters and other machine learning methods. Sixty minutes prior to shock onset, they obtained AUC scores between 0.847 and 0.882. Correlations between parameters measured at different timestamps were not taken into account.

A similar approach was taken by Mani *et al.* [13], predicting late onset neonatal sepsis using a data set of 299 infants. A blood culture was taken to further subdivide between negative and positive culture sepsis. Sixty temporal values, e.g., oxygen saturation, heart rate and blood count, were gathered at 6 hours intervals, ranging from 48 hours before to 12 hours after the first blood test. Those values were combined with 30 non-temporal variables, e.g., birth weight, Agpar scores, and demographics. As classifiers, support vector machines (SVM), Naive Bayes, averaged one dependence estimators (AODE), K-nearest neighbors, decision trees, regression trees, logistic regression, and random forests were explored. Overall, the SVM model performed best with a high sensitivity (up to 95%), however a lot of patients were classified as false positives, leading to a low specificity (47%).

Kim *et al.* [14] predict sepsis using the relationships between different time samples. To this end they create a feature vector containing, among others, the slope, mean, and standard deviation of the measured variables, i.e. heart rate, white blood cell count, temperature, and respiratory rate. In addition, they use non-temporal parameters such as age, height, and weight. A data set of 1213 sepsis patients and 26 non-sepsis postoperative patients was used to train a SVM, achieving an AUC score of 0.95, 0.92, 0.90, and 0.90 for respectively 0, 3, 6, and 12 hours in advance.

Instead of exclusively looking at measured physiological values, Lukaszewski *et al.* [15] trained an Artificial Neural Network (ANN) by extending these parameters with cytokine and chemokine blood values. On average, this model correctly predicts the outcome of the blood culture test in 83.09% of patient cases between 4 and 1 day(s) before clinical diagnosis with sepsis. However, this study was performed using a limited data set of only 92 patients, due the intrusive nature of requiring daily blood samples from the patients.

Ripoll *et al.* [16] propose computational models using different kernels, including a novel kernel called the Quotient Basis Kernel, to predict sepsis mortality. As input, they use the simplified acute physiology (SAP) and SOFA scores extracted from a test cohort of 400 patients. They report up to 80% accuracy, outperforming standard methods such as the basal SAPS scores and logistic regression.

Finally, Henry *et al.* [17] define a real-time early warning score for septic shock (TREW score). This score is calculated by first deriving features from the patient-specific measurement streams consisting of, a.o., respiratory rate, heart rate, blood pressure, and platelet count. These features are fed into a supervised learning algorithm whereby two alarms can be triggered. The first one is activated when the

score exceeds a fixed threshold value. The second is activated when the score remains above the threshold for a particular amount of time. TREW score identifies patients who will develop septic shock with an AUC of 0.82.

Previous research thus mainly uses clinical parameters to detect patients with sepsis, septic shock or positive blood cultures without directly taking into account time series data. They base themselves on either measurements at particular point in time or on features distilled from these time series. In this way, they do not exploit dependencies between the various parameters. Time series form a special and valuable kind of input data for machine learning algorithms. However, as most modeling techniques have been designed with a static data model in mind, most of them are not suited for adequately coping with the dynamic nature of time series. Therefore, we present research techniques able to model the correlations of parameter values across time. To this end, we explore and illustrate the potential of temporal machine learning models in the accurate detection of positive blood cultures.

B.2 Materials and Methods

In the first section, we describe the data collection and pre-processing. Next, we briefly introduce the used machine learning methods. Finally, we describe the evaluation metrics.

B.2.1 Data collection

Data was collected from 2177 patients admitted at four intensive care wards of the Ghent University Hospital, Belgium, between January 1, 2013 and October 1, 2015. All 2177 patients were at least 16 years old at the moment of admission. We limit the data to patients who had a blood culture test taken during their ICU stay as our goal is to construct a model which can distinguish between patients with a severe illness (and as such, often an inflammatory response) without infection and patients who suffer from a blood infection. Therefore, patients whose treating physician did not judge that it was necessary to grow a blood culture during their ICU stay (because there was no sign of inflammation) were not included in the cohort. We define two patient groups. One group consists of patients who had a first positive blood culture during their ICU stay and whose blood sample was taken at least 48 hours after their admission in the ICU. Additionally, no blood culture taken in another ward of the hospital was positive for this group up to four days prior to ICU admission. In total, this results in a group of 229 patients. The other group consists of patients who had one or more blood culture grown during their ICU stay that all returned negative, in total 1948 patients. For the former group of patients, 72 hours of data was extracted prior to the drawing of the blood

culture. For the latter, the first 72 hours after admission were extracted. In total, this results in 14 million values of 21 clinical parameters.

B.2.2 Data preprocessing and dataset construction

The extracted data was further preprocessed resulting in a dataset suited for the application of machine learning classification algorithms. First, we removed clinical parameters which were only monitored and present in a small fraction of patients. The remaining parameters were further grouped in 9 parameter features, listed in Table B.1. Next, we removed outliers from the data by defining bounds for each variable. We consider these relatively rare outliers (0.276% of the data) to be caused by human error or machine malfunctioning. The imposed limits are also listed in Table B.1.

After removal of the outliers, we further normalize the data:

$$n = \frac{x - \texttt{avg}}{3 * \texttt{std}} \tag{B.1}$$

where x is the value, $\texttt{avg}$ the average of all values and $\texttt{std}$ the standard deviation.

As each of these variables has its own monitoring frequency, this results in a different sequence length for each variable between patients. However, the methods used in this paper require equal sequence length across variables. This requirement is fulfilled by changing the sampling rate of the data. More specifically, the total sequence time and the sampling frequency are configured to be three days and one sample per hour respectively, resulting in a total of 72 points per variable across patients. If the sampling frequency of a variable is higher than one sample per hour, we calculate the minimal, maximal, and average value (depending on the variable, see Table B.1) in the sample window. If the sampling frequency is lower, we repeat the previous value. Data missing at the start of admission is imputed using mean value across all patients.

In summary, we obtain a time sequence of 72 data points across 2177 patients with each data point containing 9 clinical features. A patient's label is 1 with a positive blood culture and 0 otherwise.

B.2.3 Neural Networks

A neural network (NN) is a machine learning model loosely inspired by the working of the human brain. Its model consists of layer of connected neurons which are mathematical functions that map several inputs to an output value by taking the weighted sum of the input values and applying a (non-linear) activation function, such as a sigmoid function, on the result. In a traditional feed-forward NN, neurons are connected in such a way that no cycles exist in the network, resulting in a one-directional information flow from input to output. In this setting, the

Table B.1: Overview of included clinical parameters. Outlier bounds are listed for each clinical value, if applicable. All measured values outside these ranges are considered to be caused by human error and thus removed. If the sampling frequency of a variable is higher than one per hour, we subsample the data.

Variable	Outlier bounds	Sampling strategy
Temperature	$[29 - 43]$	max
Blood thrombocyte count		min
Blood leukocyte count		mean
C-Reactive protein count		max
Sepsis-related organ failure assessment		max
Heart rate	$[30 - 250]$	max
Respiratory Rate	$[0 - 100]$	max
Int. norm. ratio of prothrombine time		max
Mean systemic arterial pressure	$[30 - 170]$	max

NN consists of three types of layers. The input layer takes as input the features, while the output layer consists of neurons whose output represents the actual prediction. All other nodes form one or more hidden layers which transform the data non-linearly. The strength of the connections between nodes is represented using weight matrices determined during the training stage. As no cycles exist in the network, the final output of the NN is not determined by previous inputs, as such NN cannot explicitly exhibit temporal behavior. In the present work, the time series data is transformed into a traditional feature matrix consisting of 72 times 9 clinical features per patients before being used as input.

B.2.4 Long Short-Time Memory Neural Networks

A recurrent neural network (RNN) extends regular NNs by allowing for the presence of cycles within the network. These cycles enable the network to display a 'memory' by allowing the combination of present and past inputs in order to enhance prediction. RNNs using temporal data are traditionally trained by unfolding them into regular neural networks by introducing new layers for each time step. A recurring problem within such networks is the vanishing gradient problem caused by the amount of introduced layers, leading to the exponentially fading influence of older inputs. Ultimately making it challenging for the network to capture dependencies spanning over several periods of time.

Long Short-Time Memory (LSTM) neural networks [18] mitigate this problem by introducing the gating principle. These gates allow the network to conceptually implement small memory cells able to contain its hidden state for longer periods of time by blocking its inputs and/or outputs. In a standard LSTM, information only flows in a forward time direction. A bidirectional LSTM (Bi-LSTM) combines a

LSTM reading the input sequence from past to present (forward network) with a LSTM reading the input sequence from present to past (backward network). In this way the model captures reverse temporal dependencies present in the data.

The basic network we implemented for solving this problem has an input layer accepting the time sequence as a 72x9 matrix. This input is the passed on to a Bi-LSTM layer that has the tanh-function as activation function in order to introduce non-linearity. Other hyperparameters are tuned using a grid search and cross-validation.

B.2.5 Evaluation metrics

A binary classification allows for the following categorization of prediction:

- true positives (TP): examples correctly labeled as positive
- false positives (FP): negative examples incorrectly labeled as positive
- true negatives (TN): examples correctly labeled as negative
- false negatives (FN): positive examples incorrectly labeled as negative

To evaluate the accuracy of our predictions, we use the area under the receiver operator (auROC) curve, as well as the area under the precision-recall (auPR) curve. We define both curves using following definitions:

$$\text{True Positive Rate (TPR) or Recall} = \frac{TP}{(TP + FN)} \tag{B.2}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{(FP + TN)} \tag{B.3}$$

$$\text{Precision} = \frac{TP}{(TP + FN)} \tag{B.4}$$

The ROC curve is commonly used to report the performance of a binary classifier. It plots the TPR in function of the FPR, showing how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. The PR curve plots the precision in function of recall and is often used an an alternative to the TOC curve in situations with class imbalance. This applies to our current data, as the amount of positive blood cultures (229) account for only 10% of our data. For a further discussion of the relationships and differences between the two curves, we refer to the work of Davis and Goadrich [19].

B.3 Results and Discussion

In this section, we evaluate and discuss our models' performance based on their ability to accurately predict the outcome of a blood culture test. In the first section,

we discuss in detail our evaluation procedure and the ROC and PR curves resulting from the training of a Bi-LSTM network on the extracted data. Next, we compare its performance with a non-temporal NN. We conclude by discussing the ability of both networks to pick up early warning signals indicating the need to perform a blood culture test.

B.3.1 Performance evaluation

The dataset was split in a training (80% of the data) and testing (20%) set. The hyperparameters of the Bi-LSTM network (number of hidden layers, number of neurons, and learning rate) were determined performing a grid search using five-fold cross-validation on the training set. Early stopping based on the validation error was used as regularization. Across the five validation folds, we chose the hyperparameters resulting in the highest average auPR. This results in five networks trained with the same hyperparameters but trained on different folds of the training data. All of these are evaluated using the the test set. Table 2 shows the auROC and auPR of the five final networks on the test set and their average performance. Two settings are shown, one in which the networks were trained using the full 72 hours and one in which only the last 42 hours were used.

Table B.2: Performance evaluation of the Bi-LSTM networks trained on 48 and 72 hours of data. Using 5-fold cross validation we chose the hyperparameters with the highest average auPR. This results in five initiations of the same network which are then evaluated on the test set, for which the results are shown in the table. For both case where 48 and 72 hours of data is used, the auROC values are high and stable across the five models trained. The auPR scores show more variation, yet also achieve good performance.

Data	Eval. metric	Net 1	Net 2	Net 3	Net 4	Net 5	Avg.
48h	auROC	0.95	0.97	0.93	0.91	0.95	**0.94**
72h	auROC	0.98	0.96	0.95	0.97	0.96	**0.96**
48h	auPR	0.71	0.71	0.73	0.82	0.80	**0.75**
72h	auPR	0.85	0.74	0.70	0.78	0.73	**0.76**

The auROC scores indicate that the trained Bi-LSTM network is able to accurately predict the outcome of the blood culture test. The performance of the networks can vary on the same test set due to the inherent randomness within the training procedure (e.g. initialization of the weights) and the fact that different parts of the data were left out as validation set. For the auROC scores, this does not result in large deviations in performance. The auPR scores however show more variation but overall still performs well. Figure B.1 shows the shape of the ROC and PR curves. We also tested models using only 24 hours or less of data, but overall this led to a quick drop in auROC and auPR and increased variation between runs.

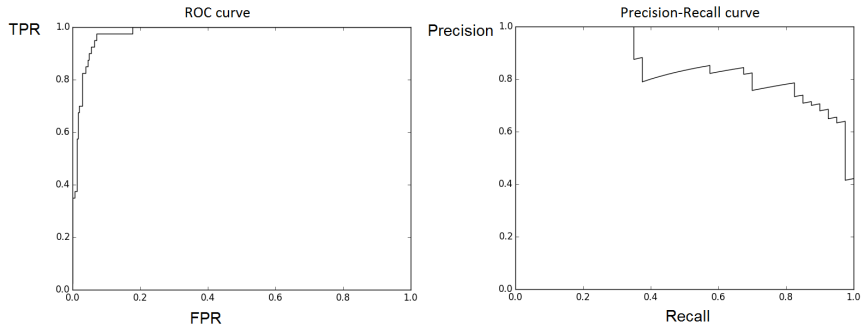

Figure B.1: The ROC and PR curve using a Bi-LSTM network that is trained used 72 hours of data.

In the previous section, we used the model to decide/predict simultaneously with the treating physician whether a blood culture test should be performed. In a practical setting, it would be valuable to accurately predict this need several hours in advance. In this setting, the model would be used as an early warning system based on clinical signs. To investigate this potential use, we constructed a Bi-LSTM as previously described, but omitted the last x hours of data, while varying x to be either 10 or 24 hours. Table B.3 lists the relative difference in auPR and auROC scores of the trained models compared to the baseline setting as reported in Table B.1. We used the same test and validation folds as previously described.

Table B.3: Performance change when the model is used as an early warning system. The baseline model performance from Table B.2 is compared one on one with the results from the networks that were trained by omitting the last x hours of data. The training folds and test set contain the same patients as in baseline setting. Results are shown for x equal to 10 and 24. The auROC scores remain largely the same when training the model while for some of the networks the auPR score drops substantially.

Data	Eval.	Net 1	Net 2	Net 3	Net 4	Net 5	Avg.
48h - 10h	auROC	0.02	−0.01	0.03	0.06	0.01	**0.02**
48h - 24h	auROC	0.02	−0.02	0.02	0.04	0.03	**0.02**
72h - 10h	auROC	−0.01	−0.01	0.03	0.00	−0.01	**0.00**
72h - 24h	auROC	−0.02	0.02	0.01	−0.01	0.00	**0.00**
48h - 10h	auPR	0.01	0.02	−0.02	−0.05	−0.09	**-0.03**
48h - 24h	auPR	0.05	−0.10	−0.13	−0.15	0.04	**-0.06**
72h - 10h	auPR	−0.32	0.03	−0.09	−0.21	−0.01	**-0.12**
72h - 24h	auPR	0.22	0.01	−0.04	−0.14	0.06	**-0.07**

Removing part of the data does not seem to significantly affect the auROC of the networks, while evaluations based on the auPR scores indicate a clear perfor-

mance drop. When the model would be applied as early warning system, its most important feature would be high recall (*in casu*, a patient at risk is detected by the model). Our last model is able to deliver this, but comes with the cost of an increased number of false positives compared to the full model that predicts at the moment of blood taking.

B.3.2 Comparison versus a non-temporal model

In this section, we briefly compare the performance of the Bi-LSTM networks (temporal model) with a traditional feedforward NN (non-temporal model). More precisely, we transform the time series input into a traditional feature matrix which is consequently used as input to train a NN. Table B.4 lists the resulting difference in accuracy between the non-temporal compared to the temporal model when using (48 - x) and (72 - x) hours of data, where x takes the values 0, 10, and 24. The table clearly illustrates no major shift in performance when the prediction is made at the moment the blood sample is taken. However, when the prediction is made earlier in time, the temporal model performs better. As medical staff decided not to perform a blood culture test at these moments, we can assume that in most cases the early signs of potential inflammation were more obscure. The temporal model outperformed the non-temporal one, seemingly benefiting from the ability to extract explicit temporal connections and evolutions from the data.

Table B.4: Comparison of a temporal model versus a non-temporal model. The baseline model performance from Table B.2 is compared to the performance of the neural network at three different time steps. If the prediction is made at the moment the blood test is taken, the performance is similar to that of a temporal model. In case the prediction is made at an earlier time, the temporal model (Bi-LSTM) performs better.

Data	Eval. metric	Net 1	Net 2	Net 3	Net 4	Net 5	Avg.
48h - 0h	auROC	-0.02	0.00	-0.01	0.04	-0.02	**0.00**
48h - 10h	auROC	-0.10	-0.04	-0.04	-0.04	-0.07	**-0.06**
48h - 24h	auROC	-0.08	-0.01	-0.05	-0.08	-0.06	**-0.06**
72h - 0h	auROC	-0.07	0.00	0.01	-0.01	0.01	**-0.01**
72h - 10h	auROC	-0.09	-0.05	-0.04	-0.05	-0.01	**-0.05**
72h - 24h	auROC	-0.09	-0.04	-0.02	-0.04	-0.05	**-0.05**
48h - 0h	auPR	0.04	0.06	-0.17	-0.11	-0.08	**-0.05**
48h - 10h	auPR	-0.21	-0.03	-0.04	-0.15	-0.15	**-0.12**
48h - 24h	auPR	-0.03	0.07	-0.01	-0.10	-0.17	**-0.05**
72h - 0h	auPR	-0.23	0.06	0.04	0.03	0.06	**-0.01**
72h - 10h	auPR	0.15	-0.11	0.06	0.14	-0.11	**0.03**
72h - 24h	auPR	0.00	-0.13	-0.03	0.02	-0.17	**-0.06**

B.4 Conclusion

In the ICU, it is challenging to distinguish between patients who suffer from severe inflammatory responses due to SIRS (such as trauma patients) or due to an underlying infection. Therefore, the ability to accurately detect early signs of a bloodstream infection may be life-saving. In this paper, we explore the use of machine learning methods to predict the outcome of a blood culture test using nine clinical parameters measured as a function of time. Our Bi-LSTM neural network uses these values in the form of time series at a frequency of one measurement per hour. We show that this model can accurately determine the outcome of a blood culture test at the moment the blood sample was taken. To further assess the practical value of our model, we also evaluate its prediction hours before the decision was taken to actually grow a blood culture. These predictions are slightly less accurate and especially suffer from a larger number of false positives. Even so, this model shows high potential as an early detection system for patients at risk of blood infection. Finally, we further highlight the importance of incorporating the time dimension within our model by comparing its performance with a traditional NN which cannot explicitly model these temporal aspects. Here, we conclude that although prediction accuracy is similar at the moment the blood sample is taken, the non-temporal model suffers a loss of accuracy when the prediction is made at an earlier moment in time. In summary, we demonstrate that data-driven temporal models which use commonly measured parameters show great promise to indicate the necessity to perform a blood culture test and can be helpful to predict the outcome.

References

[1] C. C. Yang and P. Veltri. *Intelligent healthcare informatics in big data era*. Artificial Intelligence in Medicine, 65(2):75–77, 2015.

[2] J. F. Dhainaut, A. F. Shorr, W. L. Macias, M. J. Kollef, M. Levi, K. Reinhart, and D. R. Nelson. *Dynamic evolution of coagulopathy in the first day of severe sepsis: relationship with mortality and organ failure*. Critical Care Medicine, 33(2):341–348, 2005.

[3] J. L. Vincent, J. C. Marshall, S. A. Namendys-Silva, B. Francois, I. Martin-Loeches, J. Lipman, K. Reinhart, M. Antonelli, P. Pickkers, H. Njimi, E. Jimenez, and Y. Sakr. *Assessment of the worldwide burden of critical illness: the Intensive Care Over Nations (ICON) audit*. The Lancet Respiratory Medicine, 2(5):380–386, 2014.

[4] B. Nguyen, E. P. Rivers, F. M. Abrahamian, G. J. Moran, E. Abraham, S. Trzeciak, D. T. Huang, T. Osborn, D. Stevens, and D. A. Talan. *Severe sepsis and septic shock: review of the literature and emergency department management guidelines*. Annals in Emergency Medicine, 48(1):54–e1, 2006.

[5] T. Lagu, M. B. Rothberg, M. Shieh, P. Pekow, J. S. Steingrub, and P. K. Lindenauer. *Hospitalizations, costs, and outcomes of severe sepsis in the United States 2003 to 2007*. Critical Care Medicine, 40(3):754–761, 2012.

[6] C. W. Seymour, T. D. Rea, J. M. Kahn, A. J. Walkey, D. M. Yealy, and D. C. Angus. *Severe sepsis in pre-hospital emergency care: analysis of incidence, care, and outcome*. American Journal of Respiratory and Critical Care Medicine, 186(12):1264–1271, 2012.

[7] E. Rivers, B. Nguyen, S. Havstad, J. Ressler, A. Muzzin, B. Knoblich, E. Peterson, and M. Tomlanovich. *Early goal-direct therapy in the treatment of severe sepsis and septic shock*. New England Journal of Medicine, 345(19):1368–1377, 2001.

[8] M. S. Rangel-Frausto, D. Pittet, M. Costigan, T. Hwang, C. S. Davis, and R. P. Wenzel. *The natural history of the systemic inflammatory response syndrome (SIRS): a prospective study*. Journal of the American Medical Informatics Association, 273(2):117–123, 1995.

[9] M. Morrell, V. J. Fraser, and M. H. Kollef. *Delaying the empiric treatment of candida bloodstream infection until positive blood culture results are obtained: a potential risk factor for hospital mortality*. Antimicrobial agents and chemotherapy, 49:3640–3645, 2005.

[10] M. M. Levy, M. P. Fink, J. C. Marshall, E. Abraham, D. Angus, D. Cook, J. Cohen, S. M. Opal, J. L. Vincent, and G. Ramsay. *2001 SCCM/ESICM/ACCP/ATS/SIS international sepsis definitions conference*. Intensive Care Medicine, 29(4):530–538, 2003.

[11] J. C. Ho, C. H. Lee, and J. Ghosh. *Imputation-enhanced prediction of septic shock in ICU patients.* In Proc. of the ACM SIGKDD Workshop on Health Informatics, 2012.

[12] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark. *Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): a public-access intensive care unit database*. Critical Care Medicine, 39(5):952, 2011.

[13] S. Mani, A. Ozdas, C. Aliferis, H. A. Varol, Q. Chen, R. Carnevale, Y. Chen, J. Romano-Keeler, H. Nian, and J. Weitkamp. *Medical decision support using machine learning for early detection of late-onset neonatal sepsis*. Journal of the American Medical Informatics Association, 21(2):326–336, 2014.

[14] J. Kim, J. M. Blum, and C. D. Scott. *Temporal features and kernel methods for predicting sepsis in postoperative patients*. 2010.

[15] R. A. Lukaszewski, A. M. Yates, M. C. Jackson, K. Swingler, J. M. Scherer, A. J. Simpson, P. Sadler, P. McQuillan, R. W. Titball, and T. J. G. B. and. *Presymptomatic prediction of sepsis in Intensive Care Unit patients. Clinical and Vaccine Immunology*. Clinical and Vaccine Immunology, 15(7):1089–1094, 2008.

[16] V. J. R. Ripoll, A. Vellido, E. Romero, and J. C. Ruiz-Rodríguez. *Sepsis mortality prediction with the Quotient Basis Kernel*. Artificial intelligence in medicine, 61(1):45–52, 2014.

[17] K. E. Henry, D. N. Hager, P. J. Pronovost, and S. Saria. *A targeted real-time early warning score (TREWScore) for septic shock*. Science Translational Medicine, 7(299):1–9, 2015.

[18] S. Hochreiter and J. Schmidhuber. *Long short-term memory*. Neural computation, 9(8):1735–1780, 1997.

[19] J. Davis and M. Goadrich. *The relationship between Precision-Recall and ROC curves*. In Proceedings of the 23rd international conference on Machine learning, pages 233–240. ACM, 2006.