Neural Network Approaches To Survival Analysis

Kalderstam, Jonas

2015

NEURAL NETWORK APPROACHES TO SURVIVAL ANALYSIS

Jonas Kalderstam

LUNDS UNIVERSITET

2015

Thesis for the degree of Doctor of Philosophy

Department of Astronomy and Theoretical Physics
Faculty of Science
Lund University

Thesis advisor: *Mattias Ohlsson*
Faculty opponent: *Azzam Taktak*

To be presented, with the permission of the Faculty of Science of Lund University, for public critiscism in Sal F at Fysicum, on the 29th of May 2015 at 13:15.

DOKUMENTDATABLAD enl SIS 61 41 21

Organization	Document name
LUND UNIVERSITY Department of Astronomy and Theoretical Physics Sölvegatan 14A SE–223 62 LUND Sweden	**DOCTORAL DISSERTATION** Date of issue April 2015 Sponsoring organization
Author(s) Jonas Kalderstam	

Title and subtitle

Neural Network Approaches to Survival Analysis

Abstract

Predicting the probable survival for a patient can be very challenging for many diseases. In many forms of cancer, the choice of treatment can be directly impacted by the estimated risk for the patient. This thesis explores different methods to predict the patient's survival chances using artificial neural networks (ANN).

ANN is a machine learning technique inspired by how neurons in the brain function. It is capable of learning to recognize patterns by looking at labeled examples, so-called supervised learning. Certain characteristics of medical data make it difficult to use ANN methods and the articles in this thesis investigates different methods of overcoming those difficulties.

One of the most prominent difficulties is the missing data known as censoring. Survival data usually originates from medical studies, which only are conducted during a limited time period for example during five years. During this time, some patients will leave the study for various reasons like death by unrelated causes. Some patients will also survive the study without experiencing cancer recurrence or death. These patients provide partial information about the survival characteristics of the disease but are challenging to include in statistical models.

Articles I–III, and V utilize a genetic algorithm to train ANN models to maximize (or minimize) non-differentiable functions, which are impossible to combine with traditional ANN training techniques which rely on gradient information. One of these functions is the concordance index, which compares survival predictions in a pair-wise fashion. This function is often used to compare prognostic models in survival analysis, and is maximized directly using the genetic algorithm approach. In contrast, Article V tries to produce the best grouping of the patients into low, intermediate, or high risk by maximizing, or minimizing the area under the survival curve.

Article IV does not use a genetic algorithm approach but instead takes the approach to modify the underlying data. Regular gradient methods are used to train ANNs on survival data where censored times are estimated in a maximum likelihood framework.

Key words:
Artificial Neural Networks, Machine Learning, Survival Analysis, Genetic Algorithms, Evolutionary Algorithms

Classification system and/or index terms (if any):

Supplementary bibliographical information:		Language English
ISSN and key title:		ISBN 978–91–7623–307–8
Recipient's notes	Number of pages 139	Price
	Security classification	

Distributor

Jonas Kalderstam, Department of Astronomy and Theoretical Physics
Sölvegatan 14A, SE–223 62 Lund, Sweden

NEURAL NETWORK APPROACHES TO SURVIVAL ANALYSIS

Jonas Kalderstam

LUNDS
UNIVERSITET

Printed in Sweden by Media-Tryck, Lund 2015

ISBN 978–91–7623–307–8 (print); ISBN 978–91–7623–308–5 (pdf)

SUMMARY IN SWEDISH

Denna avhandling behandlar artificiella neuron nätverk och deras applikation inom medicin. Den utgår ifrån att det är viktigt att kunna uppskatta en patients överlevnadschanser för att kunna erbjuda rätt behandling för olika former av cancer. Generellt kan man säga att ju värre prognos desto mer omfattande behandling behöver man sätta in. Vissa patienter kan botas med enbart kirurgi eller strålbehandling medan andra även kräver tilläggsbehandling så som cytostatika (cellgifter). Eftersom behandlingen kan vara påfrestande är det givetvis ett mål att inte överbehandla patienter. I vissa fall har man endast tillräckligt med resurser för att erbjuda en viss andel av patienterna den mer omfattande behandlingen. I båda fallen finns det ett stort behov av att tillförlitligt kunna uppskatta en patients prognos.

Det finns en uppsjö av olika faktorer som påverkar överlevnad och eventuell risk. Till exempel ökar många gånger koncentrationen av PSA (äggviteämne som produceras i prostatans körtelceller) i blodet vid prostatacancer och höga halter av östrogen och progesteron (två hormoner) kan ge ökad risk för bröstcancer. Att blodprov skulle uppvisa förhöjda nivåer av PSA eller östrogen är dock långt ifrån ett entydigt bevis på förekomsten av cancer. Bättre prediktion är möjlig om man även tar hänsyn till andra faktorer så som ålder eller genetik, men det blir snabbt ohanterligt att kombinera fler än ett fåtal faktorer, speciellt om man måste göra det med hänsyn till tusentals patienter.

I överlevnadsanalys försöker man lösa detta hjälp av statistiska modeller som kan kombinera ett teoretiskt sätt obegränsat antal faktorer. Ett sätt att skapa statistiska modeller är genom att använda sig av maskininlärning, även kallat artificiell intelligens i vissa sammanhang. Maskininlärning tillåter en dator att på egen hand lära sig att identifiera mönster och samband. Det är med hjälp av maskininlärning som en dator kan tyda dina röstkommandon, posten kan sortera dina vykort och du kan söka efter bilder hos Google. I avhandlingen ligger fokus på en speciell metod inom maskininlärning kallad artificiella

neuron nätverk (ANN) och på hur man kan träna dessa nätverk för applikationer inom överlevnadsanalys. Ett ANN är en förenklad modell av vår egen hjärna. Denna består av ett komplext nätverk av miljarder nervceller kallade neuroner. I jämförelse består ett ANN oftast av ett tiotal men ibland upp till flera tusen neuroner. Trots den högst begränsade kapaciteten jämfört med en mänsklig hjärna är ANN väldigt kapabla att lära sig att hitta mönster i data.

En annan maskininlärningsteknik som är inspirerad av naturen är genetiska algoritmer. En genetisk algoritm är en simulering av naturlig evolution där en population av modeller tillåts para sig och generera nya modeller som är korsningar av sina "föräldrar". Precis som i naturen förekommer det också slumpmässiga mutationer som introducerar förändringar i avkommans "gener". Genom att låta strukturen hos ANN representera generna kan datorn automatiskt utveckla egna modeller.

Konventionella träningsalgoritmer för ANN kräver ofta att den felfunktion (ett mått på hur mycket fel modellen gör vid prediktion av till exempel överlevnad) man försöker minimera kan deriveras, vilket för prognostiska tillämpningar ofta innebär en begränsning. Kombinationen av genetiska algoritmer och ANN gör det möjligt att bygga prognostiska modeller på ett mer direkt sätt än vad som annars hade varit möjligt. Detta eftersom en genetisk algoritm kan minimera vilken felfunktion som helst.

ACKNOWLEDGEMENTS

First I must express my sincerest gratitude to my supervisor Mattias Ohlsson and assistant-supervisor Patrik Edén. Mattias for always being available for questions or brain-storming and Patrik for always delivering a fresh perspective. And thank you both for giving me the freedom to make the research my own; not all would accept that a PhD-student of 2-months would declare that their great new idea really doesn't work and then completely change the direction of the project.

Thanks to Anders Irbäck for all the running adventures at St Hans and Skrylle, to Karl Fogelmark for making me appreciate the important things in life: Emacs and Free Software, to Michaela Reiter-Schad for all the discussions about finishing our dissertations, and to Bettina Greese for dragging our anti-social asses out to lunch once in while.

A special mention goes out to Jose Ribeiro who — when I explained my ideas of going back to university — advised that a young man such as myself must follow his heart.

Tobias Olsson, for tolerating my nonsense and for being a true friend all these years, どうもありがとうございます。

I thank my family, Åsa, Magnus, and *Mamma*, for always supporting me and making me believe that I can do anything.

Finally, the most deserving of my gratitude is probably Maria, without whom I might never have pursued a PhD. Thank you for being there, and for bursting my bubbles of delusion from time to time.

PUBLICATIONS

The thesis is based on the following publications:

I Jonas Kalderstam, Patrik Edén, Pär-Ola Bendahl, Carina Strand, Mårten Fernö, and Mattias Ohlsson. "Training artificial neural networks directly on the concordance index for censored data using genetic algorithms", *Artificial intelligence in medicine*, vol. 58, no. 2, pp. 125–132 (2013)

II Jonas Kalderstam, May Sadik, Lars Edenbrandt, and Mattias Ohlsson. "Analysis of regional Bone Scan Index measurements for the survival of patients with prostate cancer", *BMC medical imaging*, vol. 14, no. 1, pp. 24 (2014)

III Jonas Kalderstam, Patrik Edén, and Mattias Ohlsson. "Ensembles of genetically trained artificial neural networks for survival analysis", *Proc. 21st European Symposium on Artificial Neural Networks, Computational Intelligence And Machine Learning*, Bruges (Belgium, 2013)

IV Jonas Kalderstam, Patrik Edén, Johan Nilsson, and Mattias Ohlsson. "A regression model for survival data using neural networks", *LU TP 15-08* (submitted 2015).

V Jonas Kalderstam, Patrik Edén, and Mattias Ohlsson. "Identifying risk groups by optimizing on the area under the survival curve", *LU TP 15-09* (submitted 2015).

CONTENTS

Till Åsa,
som fick in mig på fysik

Begin with a function of arbitrary complexity. Feed it values, “sense data”. Then, take your result, square it, and feed it back into your original function, adding a new set of sense data. Continue to feed your results back into the original function ad infinitum. What do you have? The fundamental principle of human consciousness.

Academician Prokhor Zakharov, “The Feedback Principle”.

1

INTRODUCTION

The idea of an intelligent machine is far from new — it is ancient. Hephæstus, blacksmith to the gods, is for example said to have created various robotic servants for his workshop on Mount Olympus. Today, artificial intelligence and machine learning are everywhere: computers automatically identify people in photos, interpret our speech, recommend movies and books to us, and play chess. This thesis deals with the application of machine learning in medicine, specifically *clinical decision support* [1]. On hearing the terms *machine learning* and *clinical decision support*, one might imagine a patient walking into a doctor's office, inputting their symptoms and vital values on a terminal, dispensing a drop of their blood, allowing a computer to announce the most likely illness and suitable course of treatment — but this is not the point at all. The goal is to assist clinicians in determining the prognoses of patients, and to, in the long run, offer treatment specifically suited for each individual.

Clinical decision support could mean several things but the one aspect which is the focus of all the articles in this thesis is risk prediction. In many clinical applications, the choice of treatment depends on the prognosis [2]. In cancer, the treatments themselves can often be more physically straining than the actual disease symptoms, and it is naturally the case that doctors try to avoid subjecting patients to them if possible. The great uncertainty in prognosis however means that many are treated unnecessarily [3]. Developing better clinical decision support systems could decrease the uncertainty in prognoses, allowing treatment to be focused on the patients with the worst expected survival chances.

While there are many methods used in the machine learning field, this thesis focuses on one in particular which is inspired by the human brain: artificial neural networks. In Section 1.1 I explain how they work, followed by a discussion on how they are capable of learning on their own in Sections 1.2 and 1.3. Finally, in Section 1.4, I conclude by discussing some issues specific to survival data.

1.1 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks [4], also commonly referred to simply as neural networks, is an approach which is based on a very simple and abstract version of how the original biological neural networks in our brains function. Because it is inspired by the brain, we "know" a priori that artificial neural networks ought to be incredibly useful for machine learning and this is why it has enjoyed much publicity and speculation over the years. Even in popular culture one can find many references to neural networks: Lieutenant Commander Data from *Star-Trek: The Next Generation* frequently mentions that his android brain is based on a neural net, and Skynet from *The Terminator* is described as a neural network which becomes self-aware (and promptly decides to exterminate humanity).

A neural network is made up by a collection of neurons. Each neuron has a number of input and output connections to other neurons called synapses. Biological neurons are able to send signals to other neurons by "firing" electrical impulses via these connections. Multiple impulses from different neurons might amplify or dampen each other, depending on the synapses. With enough incoming signal strength, a neuron will "fire" and propagate the signal to other neurons. There are a lot more details to biological neurons such as ion-channels, sodium-levels, their relations to other nerve cells etc, but these facts are irrelevant when describing an artificial neuron. All that essentially matters is the propagation of the electrical signal. A simple abstract representation of a biological neuron called a *perceptron* [5] can be seen in figure 1.1.

In this figure, the electrical signals have been replaced by simple numbers. Bigger numbers represent stronger signals and just as an

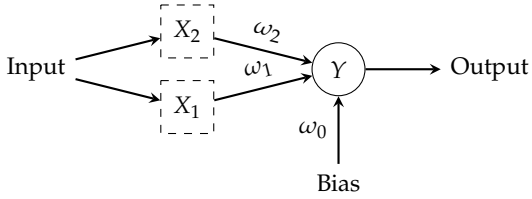


Figure 1.1: A linear perceptron with inputs, and a bias which always outputs 1. Connection strengths are determined by weights ω. The output of the perceptron is $Y = \omega_o + \omega_1 \cdot X_1 + \omega_2 \cdot X_2$.

electrical current has a polarity, the number can be either positive or negative. Input signals are sent from the nodes marked X along the connections (arrows) to the perceptron. The connections have different strengths ω called *weights* which modulate the signals; making them stronger/weaker and maybe changing their signs (polarity). At last, the signals reach the perceptron, which does its own modulation of the combined signal together with an already present current of strength ω_0 called the *bias*, and then re-transmits the final output signal. Mathematically, the process I have just described is quite straightforward:

$$Y(\vec{X}) = \omega_0 + \sum_{i=1}^{2} \omega_i \cdot X_i \tag{1.1}$$

All incoming signals, or input values, are multiplied with corresponding weights and then summed together with the bias weight. Even with this abysmally primitive model of a one-neuron brain,[1] we can solve some simple problems such as building *logic gates*.

As you may or may not know, a computer at its very core is comprised of a complex network of logic gates. These gates individually solve very simple problems but as a whole are capable of doing a great deal of computation. Each gate takes a number of inputs which are either 1 or 0. If we fix the number of inputs to two, figure 1.1 fits perfectly as a model of such a gate. For each unique input pattern, there is a corresponding target T, which also is either 1 or 0. Our per-

1 A human brain has 100 billion (10^{11}) neurons, each connected to 7000 other neurons [6], give or take.

X_1	X_2	T	Y
0	0	0	-10
0	1	0	-4
1	0	0	-4
1	1	1	2

Table 1.1: AND-gate targets in T, replicated by a perceptron in Y with weights $\vec{\omega} = \{-10, 6, 6\}$. $Y \geq 0.5$ is considered equivalent to $T = 1$, and $Y < 0.5$ equivalent to $T = 0$.

ceptron's output can take any value, so let us say that a value $Y \geq 0.5$ is considered equivalent to 1, and any value $Y < 0.5$ is equivalent to 0. As a first example, let us take the AND-gate which only outputs 1 if both of its inputs are 1, otherwise it outputs 0. One set of weights, making the perceptron replicate this gate, is: $\omega_0 = -10$, $\omega_1 = \omega_2 = 6$. All possible inputs, target values, and corresponding perceptron outputs can be seen in table 1.1. Constructing a weight vector for the OR-gate, which outputs 0 if all inputs are 0, or 1 if either input is 1, is left as an exercise for the reader.

One logic gate which a single perceptron cannot solve [7] is the exclusive-or (XOR). XOR outputs 1 *if only a single input* is 1, otherwise it outputs 0. To understand why a perceptron cannot solve the XOR-problem, we can express XOR with boolean algebra:

$$X_1 \oplus X_2 = (X_1 \vee X_2) \wedge \neg(X_1 \wedge X_2) \tag{1.2}$$

Which reads "X_1 exclusive-or X_2 equals (X_1 or X_2) and not (X_1 and X_2)". In other words, the XOR-gate is made up by one OR-gate, two AND-gates, and one NOT-gate (a NOT-gate merely turns 1 into 0, and 0 into 1).

To solve it we would need to connect several perceptrons (each implementing a specific gate) and construct a *multi-layer perceptron*

(neural network). But multiple perceptrons can always be reduced to just a *single* perceptron:

$$Y = \sum_j \tilde{\omega}_j \sum_i \hat{\omega}_{j,i} \cdot X_i = \sum_i X_i \left(\sum_j \tilde{\omega}_j \cdot \hat{\omega}_{j,i} \right) = \sum_i X_i \cdot \omega_i \tag{1.3}$$

So even though we have used the perceptron to create some logic gates, we are unable to combine them to construct a functional computer.[2]

To be able to construct a multi-layer perceptron, we'll have to tweak eq. 1.1 slightly by introducing an *activation function* (denoted by φ):

$$Y(\vec{X}) = \varphi \left(\omega_0 + \sum_{i=1}^{2} \omega_i \cdot X_i \right) \tag{1.4}$$

So far, the perceptrons have used a linear activation function, e.g. $\varphi(x) = x$. One non-linear activation function we can use instead is the *sigmoid* function (also called the logistic function), an illustration of which can be seen in figure 1.2:

$$\varphi(x) = \frac{1}{1 + \exp(-x)} \tag{1.5}$$

The sigmoid is merely a monotonic mapping from $(-\infty, \infty)$ to $(0, 1)$ (meaning that if $x_i < x_j$ then $\varphi(x_i) < \varphi(x_j)$), so the weights which made the perceptron function as an `AND`-gate still work.

Figure 1.3 shows a neural network implementing an `XOR`-gate as described by eq. 1.2 using sigmoid activation functions, and the corresponding truth table is presented in table 1.2. This solution cannot be reduced to a simple perceptron, courtesy of the sigmoid function. A neural network such as this is also called a *feed-forward* neural network because the neurons only connect from the inputs, towards the outputs; there are no connections going back.

2 You can actually construct any gate, and hence a computer, using only `NAND`-gates (`NOT-AND`) [8], which outputs 0 if both inputs are 1, and 1 otherwise.

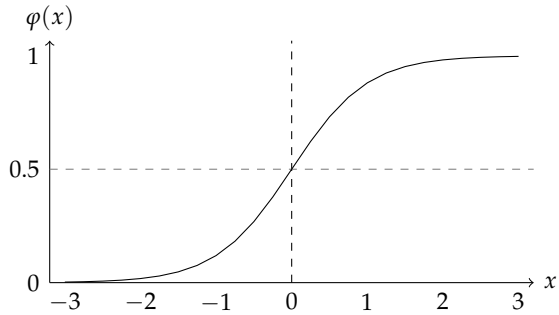


Figure 1.2: Illustration of the non-linear sigmoid activation function $\varphi(x) = \frac{1}{1+\exp(-x)}$.

X_1	X_2	T	Y
0	0	0	0.08
0	1	1	0.87
1	0	1	0.87
1	1	0	0.04

Table 1.2: Truth table for the XOR-solution in figure 1.3. $Y < 0.5$ is considered equivalent to $T = 0$, and $Y \geq 0.5$ equivalent to $T = 1$.

The neurons are organized in layers.[3] The first layer (to the left) is the input layer, and is made up by the data. The last layer (to the right) is the output layer. Any layers — and the neurons they contain — located between the input and the output are called *hidden*. The network in figure 1.3 has a single hidden layer with two hidden neurons, which are connected to the inputs and the bias. The single output neuron is then in turn connected to the hidden neurons and the bias (and sometimes also to the inputs). There can be many hidden layers, but this is theoretically not any better than a single hidden layer [9, 10].

The non-linearity of the sigmoid activation function mean we can now make an arbitrarily complex neural network. In theory, we could

3 Some authors make the restriction that layers can only be connected to immediately adjacent layers.

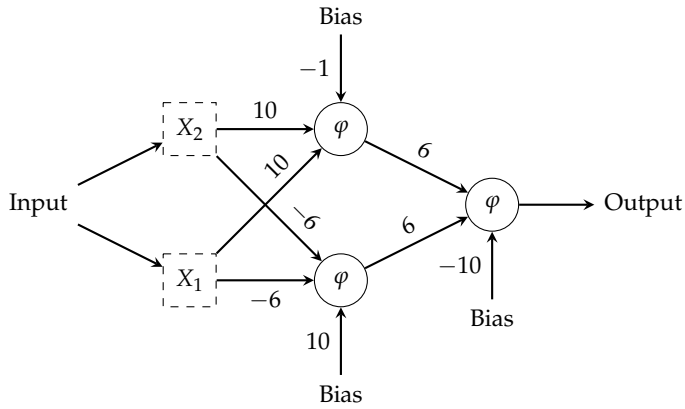


Figure 1.3: A neural network with a hidden layer containing two hidden neurons. φ is the sigmoid activation function, and the weights solve XOR exactly as the boolean expression in eq. 1.2. See also table 1.2.

even build a computer by combining many networks, each implementing a specific logic gate.

1.2 TRAINING A NEURAL NETWORK

Having a complex neural network is no good unless we can find appropriate weights for the connections. In the previous section, I demonstrated a few manually constructed solutions to some fairly simple problems. Manually constructing a solution to a more difficult problem would however be nigh on impossible. Take for example something which comes natural to most of us: recognizing a face in a picture. Facial recognition is something that humans excel at but I challenge you to explain how you do it. Now you might think to yourself something along the lines of "a face has eyes, a nose, and a mouth", to which I would ask *how do you recognize an eye, or a nose for that matter*? How do you go from a picture, a collection of pixels, to saying "that picture contains a face"? It is a question that most of us are fundamentally unable to answer.

So how is it that we are able to recognize so many things around us? No one ever tells children what a face is; they learn it on their

own. But, at some point someone does say to them "that is a chair". They do not explain how to visually identify an object as a chair, they merely point to one. A child might point to a table and say "chair" but then he would be corrected "no, that is a table". Each "yes" or "no" will alter the synapses in the child's brain — altering the weights of the neural network. This is an example of *supervised learning* and we can use the same technique to teach an artificial neural network to recognize things. Of course, one question is still *how* to alter the weights of the network.

First, I have to introduce the concept of an *error function*. If you look at table 1.2, T lists the correct answers, 1 (yes) or 0 (no), and Y is the output from the neural network. Obviously, a perfect network would output 0 and 1 instead of 0.08 and 0.87. So an output Y closer to the target T would have less error. Let us define a total error E as the sum of all (one for each row in the table) such differences:

$$E = \sum_{i=1}^{4} (T_i - Y_i)^2 \tag{1.6}$$

I also took the liberty of squaring each difference. Squaring the values mean that the minimum value is zero, which only a perfect solution can achieve. It also means that it does not matter if a network is outputting a Y too high, or too low; sign does not matter. Eq. 1.6 is commonly known as the *sum of squares* error function. It works well when we have access to exact target values because this means that we know a priori what the output of a perfect network would be and thus how far away our current network is from that solution.

Now, the clever bit enters the stage. We can calculate how we should change a weight to make the solution better by simply differentiating the error function with respect to a weight:

$$\Delta\omega_i = -\eta \frac{\delta E}{\delta \omega_i} \tag{1.7}$$

This is called *gradient descent* [11], and η is just a small constant called the *learning rate*. You can of course be increasingly clever, but all gradient techniques are based on the principle that you move in the decreasing direction of the derivative for as long as you can, and then

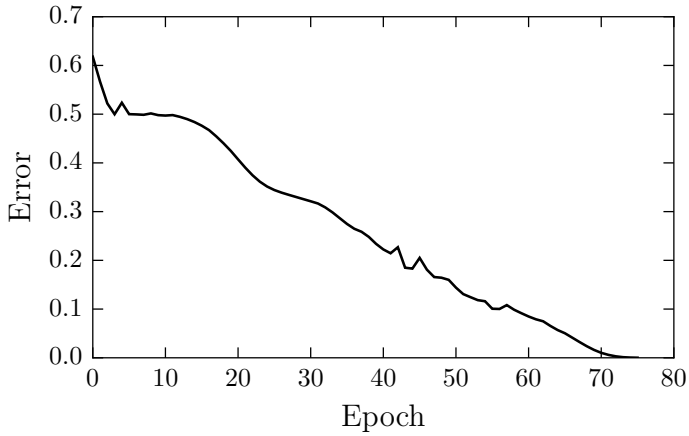


Figure 1.4: Using gradient descent to find a solution to the XOR-problem starting from a completely random set of weights. Less than 80 weight adjustments (called epochs) were required to reduce the error to basically zero. The final weights can be seen in fig. 1.5.

you have arrived at your solution. An example of gradient descent is shown in figure 1.4 where less than 80 weight adjustments were required to move an initially completely random network to a set of weights which solve XOR. The weights can be seen in figure 1.5.

Gradient descent works very well if you have proper targets to train on, but real data does not have simple ones and zeros as in the XOR-problem, instead it is *noisy* and contains random fluctuations. If you consider an actual XOR-gate, as in a piece of hardware made up by transistors, its output is a voltage. If you were to measure this output, you would note that it is not a constant 1V. Most measurements would be close to 1V but some measurements would deviate[4] to 0.9V, or even be 1.2V. What we would expect is a random normal distribution of values centered around the true value (1V). If we had enough measurements, we could simply average them all and the noise would be canceled out.

But with limited data, a neural network would eventually learn the noise specific to the training data, something that is called *overtraining*. Greater flexibility in a model, e.g. more hidden neurons,

4 Voltage differences are exaggerated for this example.

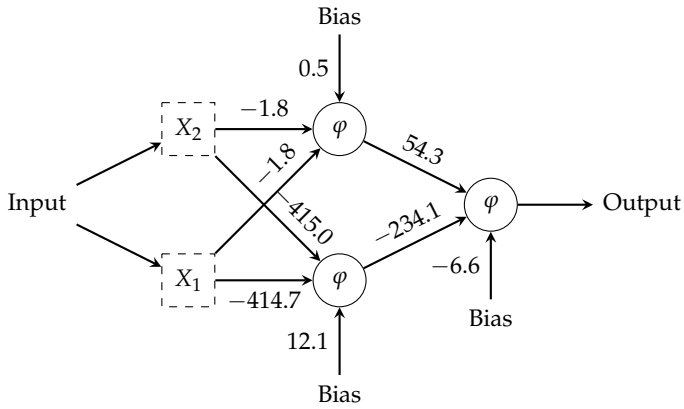


Figure 1.5: Weights found by doing gradient descent on XOR.

increases the possibility of over-training. However, it is possible to turn it to our advantage by combining many different models into an *ensemble* [12]. By averaging the model predictions, the ensemble will perform better than the average model. This effect is also commonly referred to as the *wisdom of the crowds* in other contexts [13]. The ensemble effect is visible in figure 1.6, where each individual neural network gets to train on the XOR-data in table 1.2 with gaussian random noise added to the targets; resulting in target vectors such as $\{-1.47, -0.89, -0.03, 1.9\}$. They are then tested on the true data where there is no noise ($\{0, 1, 1, 0\}$) to see if they learned the noise or the XOR-logic. To prove my point, any networks that *do* manage to learn the XOR-logic are discarded and only the rest are combined into an ensemble. *No* individual network is capable of correctly solving the problem but as you can see in the figure, as the ensemble grows the success rate increases. To make sure each member gets to train on different noise in a real setting, you could add random noise to the data as I have done here but usually you would randomly pick pieces of the data to train on for each member, so-called *bagging* [14].

Gradient methods are fast and efficient but some problems cannot be defined with differentiable error functions.

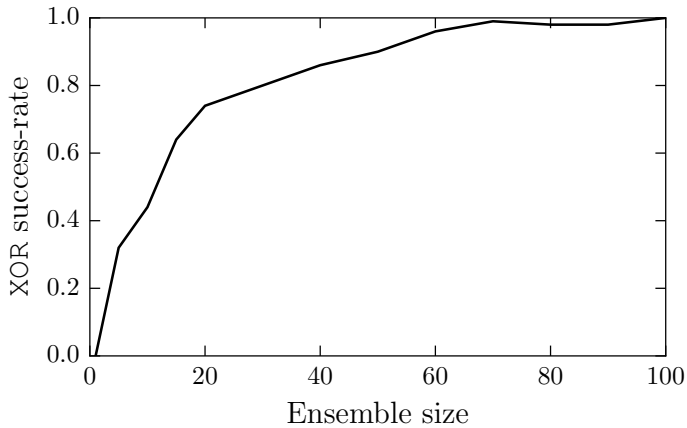
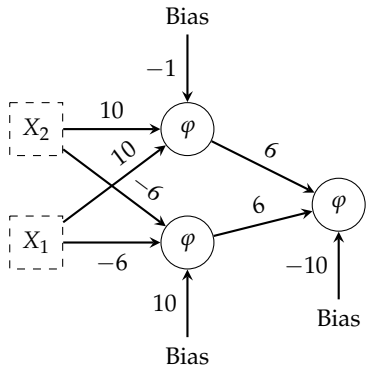


Figure 1.6: Making an ensemble from individual networks is very efficient. Here the size of the ensemble is shown along the x-axis, and the rate of success is shown along the y-axis. Each point is an average of 100 repetitions. Note that the training data is noisy and no individual network is capable of solving the non-noisy problem; while an ensemble of 40 networks has an 80% success-rate and 100 networks has a 99% success-rate.



Corresponding genome of network:

−1	10	10	10	−6	−6	−10	6	6

Figure 1.7: A neural network's weights can be written as a vector which can be used as the genome in a genetic algorithm.

1.3 TRAINING WITH GENETIC ALGORITHMS

Luckily, it is still possible to train a network even without gradient information. One way of doing this is using a *genetic algorithm* [15], sometimes also referred to as an *evolutionary algorithm*. Much like neural networks are abstract simple versions of human brains, a genetic algorithm encodes the gist of natural selection [16], where the strongest and most adapted individuals are more likely to reproduce and pass on their genes. The genes of a neural network are its weights [17] and they can be encoded into a *genome* by writing them as a vector in a predetermined order, as illustrated by figure 1.7.

Instead of training a single network as during gradient descent, a genetic algorithm generates a population of networks and each network's fitness is assessed using an error function, called a *fitness function* in this context. The networks having the highest fitness (lowest error) are more likely to be selected for *breeding*. Breeding two networks means producing offspring: a third (or more) network which is a mix between the two parent networks. Just as a child inherits half of its chromosomes from each parent [18], a neural network will in-

herit some weights from each parent network. Mixing two genomes is referred to as *crossover*. Even though the population may start out with completely random weights, networks with better fitness will evolve over time. The size of the population is usually kept constant, so to make room for the new generation every new-born network means the death of the least fit network in the population. A *generation* is said to have elapsed when a population has given birth to as many children as there are networks in the population. Over successive generations, it is highly probable that a dominant "bloodline" will emerge and take over the entire population. This will lead to in-breeding, which is why crossover alone is not enough for the genetic algorithm to be successful. Crossover is furthermore not enough if all individuals in the population lacks a certain gene, vital for the solution. As an example, take a population of humans who all have brown eyes. If no one in the population carries the gene for green eyes, then it does not matter how long you wait; no children with green eyes will ever be born. That is, unless a child is born with a *mutation*.

In nature, many mutations are either harmless or have a negative impact on the individual. Some however give rise to useful adaptations to the environment. Mutation is the reason some adult humans are able to digest milk [19], but it is also the cause of certain genetic disorders such as color blindness [20] and accelerated aging (Progeria) [21]. In the genetic algorithm, mutation promotes population diversity. Crossover will otherwise reduce the genetic diversity over time as in-breeding becomes more common. Also, as in the example with eye color, mutation is necessary in order to introduce otherwise missing genes. A simple way of introducing mutation into neural networks is to randomly change some weights in children after crossover. For example, for each new-born network, select one weight at random, and add a random number which is picked from the normal distribution:

$$\omega' = \omega + R \tag{1.8}$$

Most random numbers will be very close to zero, and so most mutations will only barely effect the networks. But, some can be just

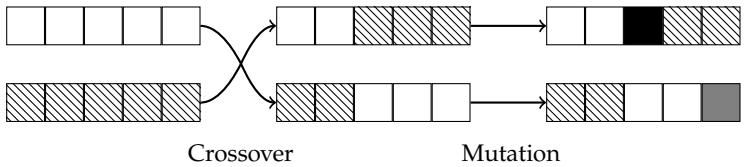


Figure 1.8: Generation of offspring in the genetic algorithm. First, two parents are selected at random (but individuals with better fitness are more likely to be selected). The genes of the two parents are colored differently in-order to trace their origin throughout the process. Second, crossover is performed on the parents. Here a pivot point is selected between the second and third genes. Each offspring gets one side of the pivot point from each parent. Third, the offspring are subjected to random mutations before being inserted into the population.

enough of a nudge "in the right direction", increasing the fitness of the offspring. Figure 1.8 illustrates the how two parents can give rise to two offspring. Here, a *pivot point* is randomly selected between the second and third genes. After crossover, one offspring has the first two genes of its mother and the last three genes from its father, while the second offspring sports the opposite set. Then, some random mutations are introduced: the first offspring gets its third gene mutated, and the second offspring gets its fifth gene mutated. The final result is two offspring which are similar, but not quite identical to their parents. Note that even if the two parents were genetically identical, mutation would ensure that the offspring were not.

A genetic algorithm is a flexible optimization method with few constraints. This freedom means that you could in theory use a genetic algorithm to figure out the best solution to many problems which are difficult to express mathematically. Two such examples are designing electronic circuits [22] and designing race cars [23]. A third example is ranking cancer patients according to survival chance [2].

1.4 SURVIVAL DATA

Previous sections have demonstrated how neural networks can be constructed and trained to solve problems. Specifically, I showed a

few designs and corresponding sets of weights that would answer questions such as "are both inputs active?" (AND), and "Is only one input active?" (XOR). The exact same approach can be used to answer a more clinical question like "is this patient, given these test results, at risk of dying from cancer?".

For the data sets used in this thesis, the endpoint of interest is typically death or cancer recurrence. If this were all, then training a neural network would be easy and not much different from training on the XOR-problem. What complicates matters is that a patient can exit the study for reasons other than the endpoint of interest. This patient is then said to be *censored*. To give a grim example, consider a patient which is part of the study for one year and then gets run over by a bus. The patient has now exited the study for reasons other than cancer-related death. The complication arises because now we have partial information and it is not obvious how to take that into account because there is no target variable to train on in this case. We know the patient was alive one year into the study, but have no idea on what would have happened next. A more cheerful example of censoring is when people simply survive the study, referred to as *tail-censored* as they make up the end of the survival time distribution. For many kinds of cancer, a substantial fraction of the patients will not die or have recurrence during the medical study's limited time frame. Ironically, improved treatment methods during the last fifty years have meant that more people are censored from studies, making future data analysis more difficult and studies more expensive.

When approached from a statistical perspective, censoring is considered to be uninformative [24], e.g. random and uncorrelated from the true survival time. A naive approach to dealing with this might be to simply discard all censored times and train your neural networks directly on the remaining non-censored times, but this would introduce a bias in the model. It is most apparent when you consider discarding a chunk of tail-censored patients from the data set. You would be removing the healthiest individuals from the data set: those that survived, and thus train only on the ones with the worst prognoses. If the goal is to identify the ones with the best prognoses to avoid over-treating them, this is clearly not a bright idea. A substantial part of the available data can also be censored. Figure 1.9

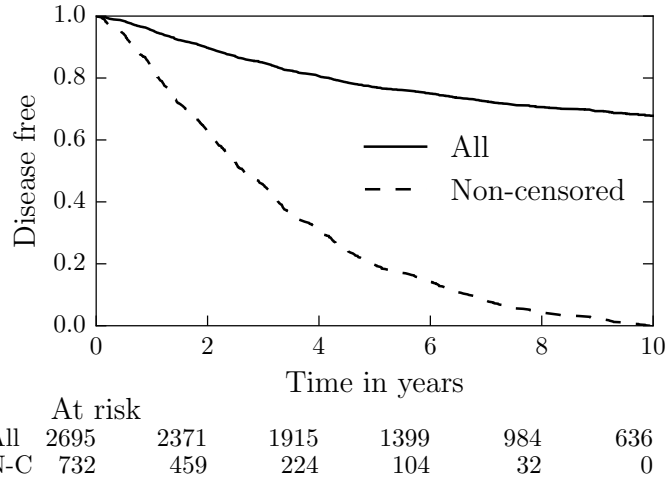


Figure 1.9: Kaplan-Meier plot of disease free survival in breast cancer [2]. The solid lines includes all patients in the data set (2695), while the dashed line only includes the non-censored individuals (732). All patients start at year 0 after the tumor(s) was removed surgically. Over time, patients are removed from the data due to cancer recurrence or are censored.

illustrates how the survival in a data set looks if the censored individuals are ignored. For the data sets used in this thesis, typically 50% or more patients are censored. One data set [25] even has more than 87% censoring. Discarding them would make smaller data sets all but disappear.

A further complication arises when you consider what kind of error function to use. The partial nature of censored survival times make something like the sum of squares unusable. This goes for all statistical models and not just ANN. One performance measure which is often used in medical statistics is Harrell's *concordance index* [26] (c-index). The c-index is a measure of how well a model is able to sort, or *rank*, the patients. It is a number which is calculated by comparing all predictions and survival times pair-wise:

$$\mathcal{C} = \frac{\sum_{i,j \in V} c(i,j)}{|V|} \tag{1.9}$$

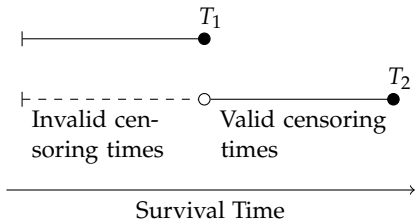


Figure 1.10: In dashed, the censoring range which would invalidate the pair from inclusion in calculation of the concordance index. Solid line indicates where censoring still makes it a valid pair. If none are censored, then any pair of times are valid.

You count the number of valid pairs (i, j in V) of predictions that are in concordance ($c(i, j)$) with the corresponding event times, and then divide by the number of valid pairs ($|V|$). Concordance means that if event times $T_i < T_j$, then the corresponding predictions $\tau_i < \tau_j$. A pair is valid if either none of T_i and T_j are censored, or if one of them is censored *after* the event time. This is also illustrated in figure 1.10.

The c-index is a number between 0 and 1 where 1 means the model's predictions results in a perfect sorting of the patients according to survival time. 0 means a perfect reversed sorting, so multiplying predictions by -1 results in perfect ranking. 0.5 is the expected result for a completely random ordering. It is not a differentiable function, which rules out using it as an error function in gradient descent but it can be used as the fitness function in a genetic algorithm. That means a neural network can be trained to rank patients according to survival time. The output of such a model is referred to as a *prognostic index* (or sometimes prognostic score). An accurate prognostic index could be used in a clinical setting to differentiate treatment, or to stratify groups according to risk in a phase-2 trial [27].

Hopefully, you should by now have some idea of how neural networks can be trained into prognostic models. The next chapter provides an overview of the articles, summarizing the goals and methodologies, and outlining the results.

REFERENCES

1. E. S. Berner, *Clinical decision support systems: theory and practice*. Springer Science & Business Media, 2007.
2. J. Kalderstam, P. Edén, P.-O. Bendahl, C. Strand, M. Fernö, and M. Ohlsson, "Training neural networks directly on the concordance index for censored data using genetic algorithms," *Artificial Intelligence in Medicine*, vol. 58, no. 2, pp. 125–132, 2013.
3. L. J. Esserman, I. M. Thompson, and B. Reid, "Overdiagnosis and overtreatment in cancer: an opportunity for improvement," *Jama*, vol. 310, no. 8, pp. 797–798, 2013.
4. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
5. F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
6. D. A. Drachman, "Do we have brain to spare?," *Neurology*, vol. 64, no. 12, pp. 2004–2005, 2005.
7. M. Minsky and S. Papert, "Perceptron: an introduction to computational geometry," *The MIT Press, Cambridge, expanded edition*, vol. 19, p. 88, 1969.
8. C. Peirce, M. Fisch, and C. Kloesel, *Writings of Charles S. Peirce: 1879-1884*. Writings of Charles S. Peirce, Indiana University Press, 1989.
9. G. Gybenko, "Approximation by superposition of sigmoidal functions," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.
10. K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.
11. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *NATURE*, vol. 323, p. 9, 1986.
12. P. S. A. Krogh, "Learning with ensembles: How over-fitting can be useful," in *Proceedings of the 1995 Conference*, vol. 8, p. 190, 1996.
13. C. Mackay, *Extraordinary Popular Delusions and the Madness of Crowds*. Barnes & Noble Publishing, 2004.

14. L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
15. D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.
16. C. Darwin, *On the Origin of the Species by Natural Selection*. Murray, 1859.
17. D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1* (N. S. Sridharan, ed.), IJCAI'89, (San Francisco, CA, USA), pp. 762–7, Morgan Kaufmann Publishers Inc., 1989.
18. W. S. Sutton, "The chromosomes in heredity," *The Biological Bulletin*, vol. 4, no. 5, pp. 231–250, 1903.
19. F.-Z. Chung, H. Tsujibo, U. Bhattacharyya, F. S. Sharief, and S. Li, "Genomic organization of human lactate dehydrogenase-a gene.," *Biochem. J*, vol. 231, pp. 537–541, 1985.
20. M. Albrecht, "Color blindness," *Nature methods*, vol. 7, no. 10, pp. 775–775, 2010.
21. J. K. Sinha, S. Ghosh, and M. Raghunath, "Progeria: A rare genetic premature ageing disorder," *The Indian journal of medical research*, vol. 139, no. 5, p. 667, 2014.
22. G. W. Greenwood and A. M. Tyrrell, *Introduction to evolvable hardware: a practical guide for designing self-adaptive systems*, vol. 5. John Wiley & Sons, 2006.
23. F. Castellani and G. Franceschini, "Use of genetic algorithms as an innovative tool for race car design," tech. rep., SAE Technical Paper, 2003.
24. J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*, vol. 360. John Wiley & Sons, 2011.
25. N. E. Breslow and N. Chatterjee, "Design and analysis of two-phase studies with binary outcome applied to wilms tumour prognosis," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 48, no. 4, pp. 457–468, 1999.
26. F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati, "Evaluating the yield of medical tests," *Jama*, vol. 247, no. 18, pp. 2543–2546, 1982.

27. M. R. Segal, "Regression trees for censored data," *Biometrics*, pp. 35–47, 1988.

2

OVERVIEW OF THE ARTICLES

In Articles I–III neural networks are trained to maximize the c-index using a genetic algorithm. Article V also features a genetic algorithm, but it optimizes the area under the survival curve instead in order to predict risk groups. A gradient method is used in Article IV to train on survival times which are estimated using a maximum likelihood approach.

2.1 TRAINING ARTIFICIAL NEURAL NETWORKS DIRECTLY ON THE CONCORDANCE INDEX FOR CENSORED DATA USING GENETIC ALGORITHMS

Censored data often means that prognostic models are evaluated using the concordance index (c-index) [1]. Using machine learning methods to generate prognostic models is then often challenging because the c-index is not differentiable, and machine learning usually uses some kind of gradient method [2] to train the models. A common workaround is to construct a function which is similar to the c-index, but also differentiable [3], which allows common gradient methods to be applied. In this article, we generate a prognostic model using neural networks by modifying the training method instead. By using a genetic algorithm [4, 5], we are able to use the c-index itself as the error function during training.

Each neural network outputs an arbitrary number which can be used to sort the patients according to survival. Because each network will have its own arbitrary scale, making an ensemble of many such networks is not possible by the usual method of averaging results. We

introduce the concept of a *normalized relative rank* to allow an ensemble [6] to be created. By comparing each network prediction to its own outputs on the training data, a rank relative to the training data can be calculated. By dividing by the size of the training data, a normalized rank between 0 and 1.0 is achieved which can be averaged in the ensemble result.

Our neural network model is compared to the well-known Cox proportional hazards model [7] on two data sets: one artificial, and one clinical. The artificial data set is constructed so as to be unsolvable by the Cox model in-order to demonstrate the neural network's advantage to detect non-linear correlations in the data and that our training method actually works in practice. The clinical data set is an amalgamation of 5 different medical studies [8–12]; all dealing with recurrence of breast cancer.

The results indicate that a linear model works best for the clinical data set. In terms of c-index, the two models are nearly identical and the difference is not statistically significant. This means that neural networks, in combination with the genetic training approach, are at least as good as the Cox model on linear data. But the ability to also exploit non-linear correlations, if they are there, makes the neural network approach ultimately more flexible.

My contributions

I did much of the theoretical work, performed all computational work, generated figures 3–8, and co-authored the manuscript.

2.2 ANALYSIS OF REGIONAL BONE SCAN INDEX MEASUREMENTS FOR THE SURVIVAL OF PATIENTS WITH PROSTATE CANCER

In prostate cancer, as in many other forms of cancer, an advanced state of the disease will have tumors spreading to the skeleton of the patient. These bone metastases have a severely negative impact on the survival [13, 14]. A common method for monitoring bone metastases is a bone scan. By scanning the skeleton, the tumor burden can be measured and calculated as a percentage of total skeletal mass, a so-

called *bone scan index* (BSI) [15]. Previous studies have shown that the BSI is correlated with survival in prostate cancer [16–18] but the question if tumor location plays a role is less known [19, 20]. This work tries to answer the question if it makes a difference where the tumors are located from the point of view of survival.

An automated method [21] for analyzing bone scan images was used to compute BSI values in twelve skeleton regions for 1013 patients diagnosed with prostate cancer. These input values were used to generate prognostic models with both the Cox model and the neural network approach we introduced in Article I. Both models performed similarly on the test set, as compared by the c-index, ruling out significant non-linear effects among the skeletal regions.

To ascertain whether any region is more strongly associated with survival than any other, a combination of forward and backward-elimination was used. While information about locality did not increase the c-index performance of models compared to models using only the total BSI value, we were able to conclude that information about three specific regions gives comparable information to that of all twelve regions or equivalently the total BSI.

My contributions

I was part of the theoretical work, performed all computational work, generated all figures, and co-authored the manuscript.

2.3 ENSEMBLES OF GENETICALLY TRAINED ARTIFICIAL NEURAL NETWORKS FOR SURVIVAL ANALYSIS

When developing the genetic training approach in Article I, enabling the c-index to be used directly as the error function when training neural networks, our approach was compared to the linear Cox model. In this paper, we extend the comparison to include another machine learning approach called *support vector machines* (SVM) on two other clinical data sets. The SVM approach, developed by Van Belle et al. [22, 23], uses a modified version of the c-index as its error function.

For this new comparison, the training methodology has also been refined. Optimal training parameters in the genetic algorithm, and the best number of hidden neurons in the networks, are found for each data set separately. In addition, the minimally optimal ensemble size is found to be 30 neural networks for both data sets.

Because we did not have access to the source code of Van Belle et al., we had to compare our results to what they reported in their paper. Slightly differing methodologies and different test set randomization makes the test results slightly numerically different. As a numerical guide, the Cox models can be compared directly between their and our results. When taking that difference into account, all three models (ANN, SVM, Cox) seem to perform very similar on the two data sets.

My contributions

I shared equally in the theoretical work, performed all computational work and co-authored the manuscript.

2.4 A REGRESSION MODEL FOR SURVIVAL DATA USING NEURAL NETWORKS

A prognostic index, as calculated in Articles I–III, can be used to sort patients into risk groups. A genetic algorithm allowed us to use the c-index, a ranking measure, as our error function during training. In this article a different approach is taken. Instead of changing the training algorithm, we construct a new error function which models the future survival of censored patients.

An estimation of the survival time can still be used to sort patients into risk groups but offers additional information compared to a prognostic index. While perhaps not useful directly in terms of treatment choice, it could assist a clinician in the communication with the patient. Furthermore, such point estimations can make it easier to ascertain which input variables are the most important contributors to the predicted survival [24]

This new error function is differentiable, which allows gradient methods to be used during training. The error function is a modi-

fication of the mean square error (MSE). The error to the true survival time of censored events is modeled using a maximum likelihood framework, thus the name *mean square likely error* (MSLE). The nature of the error function means that the neural networks will output a prediction of the actual survival time of the patients. A one-time pre-calculation of many factors is possible making the error function efficient ($\mathcal{O}(N)$) during actual training. The aim is to utilize information present in the data set about the future of censored events; information which is ignored by the c-index.

The new MSLE function is compared to a simpler and more naive approach which we call *mean square censored error* (MSCE), originally presented by Van Belle et al. [25]. In the MSCE, predictions for censored events can only have an error if the prediction is an underestimate of the censoring time, as no data exists beyond the censoring point.

The two error functions are compared on five distinct data sets [26–30] with differing characteristics in terms of number of patients, censoring, and number of input variables. MSLE had significantly better performance in terms of c-index on two data sets. In addition to comparing the c-index, we also compared the actual predictions using the mean square error. To be able to validate the predictions on censored events, we used only the non-censored events and randomly censored the data ourselves. This allowed the models to train on censored data, while still enabling predictions to be validated against the true survival times. In this case, MSLE gave rise to better predictions. We conclude that MSLE has advantages on large data sets with a high degree of censoring.

My contributions

I was part of the theoretical work, performed all computational work, generated all figures, and co-authored the manuscript.

2.5 IDENTIFYING RISK GROUPS BY OPTIMIZING ON THE AREA UNDER THE SURVIVAL CURVE

A prognostic index, or survival predictions, are often used to sort patients into risk groups. Risk grouping can be used to guide treatment choices, or to stratify clinical studies [31]. Most prognostic models, whether they are based on neural networks [3, 29], SVMs [22, 23], or Cox models [7], generate risk groupings almost as an after-thought. They are primarily concerned with optimizing something other than the optimal grouping. In many instances though, risk grouping *is* the primary use-case for prognostic models. In this article, we investigate the potential benefits of training neural networks specifically on generating the best possible risk grouping. We focus on generating groups corresponding to low, intermediate, and high risk.

Intuitively, it should be easier to find a good grouping such as that compared to ranking all patients individually according to risk. A ranking model might get trapped in local optima where any change would be worse than the current patient sort order. A grouping model however is not constrained by what the individual sort order might be and might thus be free to explore more aspects of the parameter space. The hypothesis is that this might allow more non-linear correlations to be exploited by the neural networks.

It is difficult to define what the best possible risk grouping is. It is clear that for the outer low and high-risk groups, the bigger the better: a low-risk group of 100 patients is clearly more trustworthy than a "group" of 1. For two groups of the same size, which is better depends on the survival of the patients. A good low-risk group would be expected to have a high median survival time, and a high end survival rate. Oppositely, a good high-risk group should have as low median survival time and end survival rate as possible. These properties are both captured in the survival curves (Kaplan-Meier estimator [32]). By once again using a genetic algorithm, and train neural networks on either maximizing, or minimizing, the *area under the survival curve* it is possible to implicitly optimize the risk-grouping directly. As a final step, we create ensembles which outputs a prediction: "low", "intermediate", or "high" risk.

This optimization procedure is compared with Cox proportional hazards [7] — which generates a prognostic index, and a decision tree method known as *recursive partitioning* (Rpart) [31] — which also generates a risk-grouping directly, on five clinical data sets with varying properties. Our interpretation of the results is that the method works and stands up well against Cox and Rpart, and that it manages to combine strengths from both models.

My contributions

I did much of the theoretical work, performed all computational work, generated all figures, and co-authored the manuscript.

REFERENCES

1. F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati, "Evaluating the yield of medical tests," *Jama*, vol. 247, no. 18, pp. 2543–2546, 1982.
2. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *NATURE*, vol. 323, p. 9, 1986.
3. E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini, "Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach," *Statistics in Medicine*, vol. 17, no. 10, pp. 1169–1186, 1998.
4. D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.
5. D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1* (N. S. Sridharan, ed.), IJCAI'89, (San Francisco, CA, USA), pp. 762–7, Morgan Kaufmann Publishers Inc., 1989.
6. P. S. A. Krogh, "Learning with ensembles: How over-fitting can be useful," in *Proceedings of the 1995 Conference*, vol. 8, p. 190, 1996.
7. D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 34, no. 2, pp. 187–

220, 1972.

8. L. Rydén, P.-E. Jönsson, G. Chebil, M. Dufmats, M. r. Fernö, K. Jirström, A.-C. Källström, G. Landberg, O. Stå l, S. Thorstenson, and B. Nordenskjöld, "Two years of adjuvant tamoxifen in premenopausal patients with breast cancer: a randomised, controlled trial with long-term follow-up.," *European journal of cancer*, vol. 41, pp. 256–64, Jan. 2005.
9. G. Chebil, P.-O. Bendahl, I. Idvall, and M. Fernö, "Comparison of immunohistochemical and biochemical assay of steroid receptors in primary breast cancer–clinical associations and reasons for discrepancies.," *Acta oncologica*, vol. 42, pp. 719–25, Jan. 2003.
10. A.-K. Falck, P.-O. Bendahl, C. Ingvar, P. Lindblom, K. Lövgren, K. Rennstam, M. Fernö, and L. Rydén, "Does Analysis of Disseminated Tumor Cells in Bone Marrow Give Additional Prognostic Information in Primary Breast Cancer? Analysis of Disseminated Tumor Cells in Bone Marrow — Report of a Prospective Study with 5 Years Follow-Up," *Cancer Research*, vol. 70, no. 24, pp. 105–6, 2010.
11. S. Hansen, D. A. Grabau, F. B. Sørensen, M. Bak, W. Vach, and C. Rose, "The prognostic value of angiogenesis by Chalkley counting in a confirmatory study design on 836 breast cancer patients.," *Clinical cancer research*, vol. 6, pp. 139–46, Jan. 2000.
12. Swedish Breast Cancer Cooperative, "Randomized trial of two versus five years of adjuvant tamoxifen for postmenopausal early stage breast cancer. Swedish Breast Cancer Cooperative Group.," *Journal of the National Cancer Institute*, vol. 88, no. 21, pp. 1543–9, 1996.
13. M. S. Soloway, S. W. Hardeman, D. Hickey, J. Raymond, B. Todd, S. Soloway, and M. Moinuddin, "Stratification of patients with metastatic prostate cancer based on extent of disease on initial bone scan.," *Cancer*, vol. 61, pp. 195–202, 1988.
14. M. Noguchi, H. Kikuchi, M. Ishibashi, and S. Noda, "Percentage of the positive area of bone metastasis is an independent predictor of disease death in advanced prostate cancer.," *British journal of cancer*, vol. 88, pp. 195–201, 2003.
15. Y. E. Erdi, J. L. Humm, M. Imbriaco, H. Yeung, and S. M. Larson, "Quantitative bone metastases analysis based on image segmenta-

tion," *J Nucl Med*, vol. 38, pp. 1401–1406, 1997.

16. R. Kaboteh, P. Gjertsson, H. k. Leek, M. Lomsky, M. Ohlsson, K. Sjöstrand, and L. Edenbrandt, "Progression of bone metastases in patients with prostate cancer - automated detection of new lesions and calculation of bone scan index.," *EJNMMI research*, vol. 3, no. 1, p. 64, 2013.
17. D. Ulmert, R. Kaboteh, J. J. Fox, C. Savage, M. J. Evans, H. Lilja, P.-A. Abrahamsson, T. Björk, A. Gerdtsson, A. Bjartell, P. Gjertsson, P. Höglund, M. Lomsky, M. Ohlsson, J. Richter, M. Sadik, M. J. Morris, H. I. Scher, K. Sjöstrand, A. Yu, M. Suurküla, L. Edenbrandt, and S. M. Larson, "A Novel Automated Platform for Quantifying the Extent of Skeletal Tumour Involvement in Prostate Cancer Patients Using the Bone Scan Index.," *European urology*, vol. 62, no. 1, pp. 78–84, 2012.
18. Y. Mitsui, H. Shiina, Y. Yamamoto, M. Haramoto, N. Arichi, H. Yasumoto, H. Kitagaki, and M. Igawa, "Prediction of survival benefit using an automated bone scan index in patients with castration-resistant prostate cancer.," *BJU international*, vol. 110, pp. E628–34, 2012.
19. J. Rigaud, R. Tiguert, L. Le Normand, G. Karam, P. Glemain, J.-M. Buzelin, and O. Bouchot, "Prognostic value of bone scan in patients with metastatic prostate cancer treated initially with androgen deprivation therapy.," *The Journal of urology*, vol. 168, pp. 1423–1426, 2002.
20. J. A. Hovsepian and D. P. Byar, "Quantitative radiology for staging and prognosis of patients with advanced prostatic carcinoma. Correlations with other pretreatment characteristics.," *Urology*, vol. 14, pp. 145–150, 1979.
21. M. Sadik, M. Suurküla, P. Höglund, A. Jarund, and L. Edenbrandt, "Quality of planar whole-body bone scan interpretations - a nationwide survey," *European Journal of Nuclear Medicine and Molecular Imaging*, vol. 35, pp. 1464–1472, 2008.
22. V. Van Belle, K. Pelckmans, J. A. K. Suykens, and S. Van Huffel, "Support Vector Machines For Survival Analysis," in *Proceedings of the third international conference on Computational Intelligence in Medicine and Healthcare (CIMED)* (E. Ifeachor and A. Anastasiou, eds.), pp. 1–8, 2007.

23. V. Van Belle, K. Pelckmans, S. Van Huffel, and J. Suykens, "Support vector methods for survival analysis: a comparison between ranking and regression approaches," *Artificial Intelligence in Medicine*, vol. 53, no. 2, pp. 107–118, 2011.
24. N. R. Cook, "Use and misuse of the receiver operating characteristic curve in risk prediction," *Circulation*, vol. 115, no. 7, pp. 928–935, 2007.
25. V. Van Belle, K. Pelckmans, J. A. Suykens, and S. Van Huffel, "Additive survival least-squares support vector machines," *Statistics in Medicine*, vol. 29, no. 2, pp. 296–308, 2010.
26. J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*, vol. 360. John Wiley & Sons, 2011.
27. C. L. Loprinzi, J. A. Laurie, H. S. Wieand, J. E. Krook, P. J. Novotny, J. W. Kugler, J. Bartel, M. Law, M. Bateman, and N. E. Klatt, "Prospective evaluation of prognostic variables from patient-completed questionnaires. north central cancer treatment group.," *Journal of Clinical Oncology*, vol. 12, no. 3, pp. 601–607, 1994.
28. T. M. Therneau, *Modeling survival data: extending the Cox model*. Springer, 2000.
29. J. Kalderstam, P. Edén, P.-O. Bendahl, C. Strand, M. Fernö, and M. Ohlsson, "Training neural networks directly on the concordance index for censored data using genetic algorithms," *Artificial Intelligence in Medicine*, vol. 58, no. 2, pp. 125–132, 2013.
30. J. Stehlik, L. B. Edwards, A. Y. Kucheryavaya, P. Aurora, J. D. Christie, R. Kirk, F. Dobbels, A. O. Rahmel, and M. I. Hertz, "The Registry of the International Society for Heart and Lung Transplantation: twenty-seventh official adult heart transplant report–2010.," *The Journal of heart and lung transplantation : the official publication of the International Society for Heart Transplantation*, vol. 29, no. 10, pp. 1089–103, 2010.
31. M. R. Segal, "Regression trees for censored data," *Biometrics*, pp. 35–47, 1988.
32. E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *Journal of the American statistical association*, vol. 53, no. 282, pp. 457–481, 1958.

INDEX