The influence of anomalies on binary prediction outcomes

A research on the influence of anomalies on neural networks and using Influence Functions to increase model performance with flawed data.

Oscar Schyns
375237
November 26, 2020

Abstract

In this paper I create a neural network to predict binary outcomes and add increasing amounts of anomalies/noise to the input data to check robustness. Model performance is evaluated as the Breakdown point of the network. Afterwards, Influence Functions are computed to evaluate the influence of anomalies, to create training set attacks and to filter anomalies. If the outcome is a binary variable, the loss of an observation is at most 1. This makes the model reasonably robust, which means that it needs many absurd anomalies to reach the Breakdown point. A no hidden layer neural network is modestly influenced by anomalies when they make up a small proportion of the data, but it rapidly decreases in performance when a certain threshold is reached. A network with a hidden layer can have a more continuous decrease in performance. However, this is not the case in the real-world data set which I used, where the anomalies have a different data generating process. Influence Functions can be used to create training set attacks to influence model parameters with limited observations, but using Influence Functions is not better than using the regular loss to filter anomalies. Removing suspected anomalies based on the loss sometimes increases performance, most notably the Brier Score, but it is probably not worth the extra training time in most situations. An observation with a high loss thus seems more likely to be an anomaly than an observation with a high influence on the loss. Influence or loss is not always bad for the model, as sometimes unexpected results teach people and model the patterns better.

Erasmus university Rotterdam
Erasmus school of Economics
Master Thesis [FEM61008]
Supervisor: Zhelonkin, M
Second assessor: Naghi, A.A.

Contents

1 Introduction

With the rise of computers and the internet, the amount of data in the world doubles every 2 years. Combined with stronger computers and larger markets for companies in a globalized world this creates a world where mathematical models become more important than ever. Mathematical models are successfully deployed in almost all fields, ranging from finance to environment, from agriculture to health care. Artificial Intelligence (AI) models such as logistic regression and neural networks can be used to predict binary outcomes, which predict whether an event is occurring or not. For instance, whether or not it is going to rain, the probability of a company going bankrupt, or even something as seemingly arbitrary as whether a supermarket customer decides to buy bacon. These models not only predict if the event is happening or not, but rather with what probability. A result of 0.45 means that the models predict that the event has a 45% chance of happening. Models make predictions based on input data, the patterns between the input data and the frequency of the event happening. However, the input data is not always correct. Large data sets often have anomalies, therefore the data cannot be assumed to be correct. Anomalies are for instance caused by accident, by the laziness of individuals, or by technical defects in sensors and computers. Anomalies in data sets can seriously reduce model performance.

Since mathematical models are used in almost all fields, decreased performance could have a negative impact in all branches of our society. Companies could make less profit and self driving cars could become more dangerous. Models analyzing climate change could become less accurate, making combating it harder, and insufficient performing models which track COVID-19 cases will cause more difficulty in containing the pandemic.

Whether you realise it or not, a well-functioning society and the quality of life of its citizens often depends on the performance of mathematical models. We live in a time where Artificial Intelligence (AI) makes more and more decisions for us; one of the great challenges of this era is to make sure that decisions made by AI are in the best interest of society, without unforeseen side effects. AI should not only perform well but it should also be better understandable, explainable and manageable to humans. Understanding the effect of anomalies on AI can improve our knowledge of how AI models work and how we can improve them.

In this paper I investigate the influence of anomalies on binary prediction outcomes, and try to reduce this influence in order for us to make better predictions, even with flawed data. The research focuses primarily on neural networks. First, I create a neural network with simulated data. Then I add an increasing amount of anomalies (noise) to the X values, and evaluate how the performance of the model declines. I also investigate where the Breakdown point of this neural network is, defined as the minimal amount of data needed for the model to have a worse prediction than a random coin toss model. Next I explain the performance decline with the calculated Influence Functions of the data. Finally, I use Influence Functions to detect anomalies and see if filtering suspected anomalies can improve model performance. To the best of knowledge, this is the first paper which uses Influence Functions to omit data from training a model with neural

networks and approximates the Breakdown point for a neural network.
The fact that Y is constrained between 0 and 1 makes the networks relatively robust. The loss is limited and thus also the influence on the parameters. A neural network without a hidden layer is not affected too much by the anomalies, until these reach a certain proportion where the parameters nearly immediately collapse. A model with a hidden layer can have a less instantaneous drop in performance and is overall more robust to outliers. This is not the case if real-world data is used, where the anomalies have a different structure. It requires a large proportion of anomalies to make the models reach the Breakdown point, especially the model with a hidden layer. Influence Functions can be used to create training set attacks, lowering model performance, while altering as few observations as possible. Observations with a high loss have a higher chance of being an anomaly than observations with a high influence on the loss, although this is not always the case. Omitting observations with the highest loss often increases the performance slightly in some situations but does cost extra training time, and requires knowing what proportion of the data is polluted which is most likely not the case in real time.

2 Literature review

Many papers have been written written about the influence of outliers/anomalies on neural networks, and about methods to train well performing neural networks on noisy data. The proportion of anomalies is also a topic often researched. Labelled data can hardly be assumed to be correct, as data often have flaws. Especially if data originate from crowd sourcing, according to Frénay and Verleysen (2013). The proportion of anomalies is between 1% and 10% in a typical raw data set Hampel et al. (2011).
Most papers on detecting anomalies and/or on reducing the influence of anomalies, fall in two categories. In the first category, anomaly detection is done based on the k-nearest-neighbor (k-NN) method. Suspected anomalies are then omitted or given less weight in the sample. In the second category, it is done by having a different loss function than the mean-squared-error (MSE) loss function. Squaring the loss increases the influence outliers have and can thus expand the influence of anomalies. Hautamaki et al. (2004) uses k-NN graphs to detect outliers. The algorithm that works best with real-world data marks observations as outliers if they do not belong to the k Nearest neighbors of at least T other observations. Rusiecki et al. (2014) also uses the k-NN algorithm to reduce the influence of anomalies. One method is dividing all errors by the anomaly score, which is the average distance between the observation and the k nearest neighbors. They use $k = 9$ and the Euclidean distance measure. This method is called the k-NN GAS method. El-Melegy and Essai (2007) tests other loss functions than the sum of squared residuals with neural networks in order to make them more robust. The authors found that with noisy data a less exponential error function performs better, since it is not influenced that much by a few outliers. Rusiecki (2013) first calculates all errors under a model with the regular MSE loss function and trim errors that are to far away from the median errors. This limits the maximum influence an observation can have on the loss. Rusiecki et al. (2014) evaluates different methods to train neural networks on noisy data. Instead of using the

MSE function, good alternatives include the Least Trimmed Absolute Value function as a loss function. With this function the h largest errors are trimmed such that the influence of outliers or anomalies is bounded. The value of h can be determined in multiple ways. the authors use the k-NN algorithm to select which instances are selected. They forecast all observations to be equal to the weighted average of the 9 nearest observations. If the forecast is to far off, usually with an error larger than 2-8 times the standard deviation of the data, they omit that observation. Neural networks have many saddle points, but local optima are of high quality, so not finding the global optima is not necessary Choromanska et al. (2015). Because a neural network has many parameters, the global optimum is also often an overfit on the data. Therefore many researchers use techniques to prevent overfitting, such as the early stopping algorithm. Hampel (1974) computes the Influence Curve (Influence Function) which essentially is the derivative of an estimator with respect to some distribution. This can calculate the effect of increasing an observation by a infinite small amount on an estimator, such as the mean of a distribution. With the Influence Function the gross-error-sensitivity can be computed, which is the supremum of the absolute value of the Influence Function. This measures the worst approximate influence, a fixed amount of contamination can have on the value of the estimator. If the gross-error-sensitivity is bounded the estimator is more robust. A lower bound on the Influence Function does lead to a larger minimum variance that can be achieved. Koh and Liang (2017) uses Influence Functions to understand predictions made by neural networks. They find that Influence Functions can also be used to approximate the effect of omitting a training point by linearly approximating. Even with the early stopping algorithm, where the coefficients are not the global minimum solution of the network, the Influence Function still is meaningful. One condition is that the Hessian matrix should have positive eigenvalues. If this is the case the loss can still be approximated with the formulas Koh and Liang (2017) uses in 4.2. They also show that Influence Functions are better at detecting mislabelled data than the loss of the possible anomaly. Croux et al. (2005) uses the second order Influence Function, which is the derivative of the Influence Function of the error rate in logistic discrimination. Outliers can have an unlimited influence on the error rate when estimating with Maximum Likelihood; thus they create a more robust weighted Maximum Likelihood estimator. If only a select number of variables from an observation is contaminated with anomalies, there should be a method to detect anomalies within the observation instead of classifying the entire observation as an anomaly. With many variables the observations with anomalies in any variable quickly increases to over 50%. Rousseeuw and Bossche (2018) creates a model that flags anomalies for every variable separately within an observation. The proposed DetectDeviatingCells algorithm works best if there are many dimensions and the correlations between variables are large. The method has a good relative performance to methods which label all variables within an observation the same: especially if there are many dimensions. A downside is the relatively high computing time. Raymaekers and Rousseeuw (2019) creates more robust estimators for correlation matrices, which can also be used for high dimensional data. These matrices are less influenced by a few outliers and are better estimators of the correlation matrix, if there are anomalies in the

data. A neural network where only the top layer is trained, is equivalent to training a logistic regression model and is often done, Donahue et al. (2014). This is equivalent to a neural network without hidden layers. This paper starts with a neural network without hidden layers since the parameters are easier to interpret.

3 Data

This paper is a simulation study. Therefore all data will be simulated. Since I investigate the influence of anomalies on binary prediction outcomes, all Y values in the data set should be binary. This means that the function that translates X values into Y values should be between 0 and 1. After having the probability of Y being 1, the true value can be simulated in order to make Y binary. I use the logistic function to compute probabilities from a set of X variables.

3.1 Logistic function

Perhaps the most used function to model binary prediction outcomes is the logistic function which is a S-shaped curve between $Y = 0$ and $Y = 1$. This makes it perfect to use in a prediction setting. Equation 1 shows a logistic function.

$$S(x) = \frac{1}{1 + exp(-(X^T\beta))}, \tag{1}$$

for the logistic function, I use 10 X variables without a constant term, sometimes called intercept. For each observation the variables are all normally distributed, with a mean of 0 and a standard deviation of 1. The weight vector equals:

$$\begin{bmatrix} 4 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \tag{2}$$

the logistic function has no constant term. If all X values equal 0, the result of the function equals 0.5. Since the average X value is 0, all weights are non negative and the function is point symmetric. The function does not need a constant term to be centered around 0.5 when the average X value is 0.

3.2 House price data

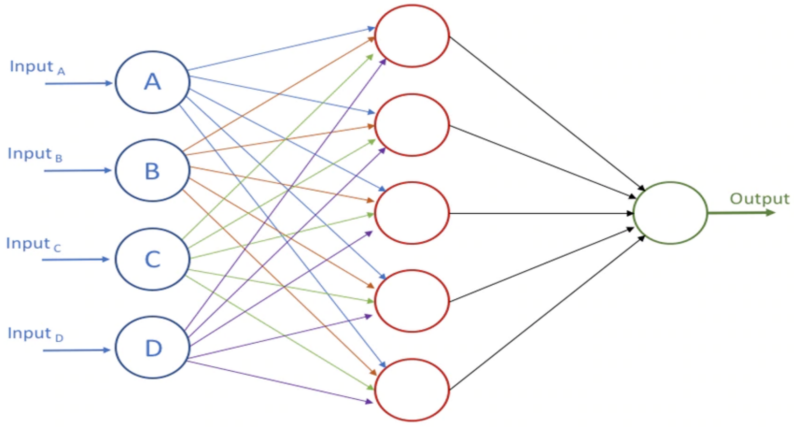
The methods in this paper are also used on real-world data. The data set used is a data set of 1461 observations with 10 explanatory variables describing a house. The Y variable is binary equaling 1 if the house is above the median house price in the same data set, and 0 otherwise. The explanatory variables can be the total square feet of the house or the number of bathrooms. In order to better compare the variables, all X variables are standardized with a mean of 0 and a standard deviation of 1.

4 Methodology

4.1 Neural Network

With these data I train neural networks to see how the performance of the model decreases as the number of anomalies increase. In order to prevent overfitting, I use the early stopping algorithm with 20% of the data in the validation set. The maximum number of training epochs equals 400. All activation functions in the neural networks are sigmoid or logistic, as seen in 1, but without a constant term in the input. The neural networks will start simple and get more complex to have a better understanding of the patterns emerging. The simplest network has no hidden layer, thereafter a neural network with a single hidden layers with 9 nodes is used and bias term in the hidden layer.

Figure 1: Neural Network with one hidden layer with five nodes in it



4.2 Anomalies

In order to study the influence of the anomalies/outliers, initially a network is trained on the unpolluted data set. Anomalies are only added to the in-sample data; if they are also present in out-of-sample data, it would be part of the data generating process and that is not the focus of this paper. After the network is trained on unpolluted data, anomalies are added to ϕ% of the observations. ϕ can be every integer between 0 and 50. The anomalies add a uniform random number between 0 and K to the first variable, with K as a positive integer. Afterwards more complex anomalies are tested, such as multiplying all X values by -1 for ϕ% of the observations, independent of K. With the logistic data generating process, multiplying all X by -1 corresponds to label changing. $S(x) = 1 - S(-1x)$, with equation (1). This cannot be assumed to be true in real-world data since the influence of X is not always point symmetric around 0. Label changing is an anomaly which can happen in the real-world if there is confusion about what $Y = 1$ corresponds to. If for a subset of the data $Y = 1$ means that event 1 does not occur, while for the rest of the data $Y = 1$ mean that event 1 does occur the labels are changed in the subset. The anomalies are generated separately for every iteration. If ϕ and K

increase, the location and size of the anomalies are not the same: this way the influence of the randomness can be seen. Could a data set with more noise but with different locations have a better performance? Simulating observations and adding between 0 and 50% noise to them will be considered one single simulation, as the unpolluted X and Y values do not change. In this paper I present both the results of single simulations and the average of multiple simulations. With the same house price data set it is also possible to do multiple simulations. By changing the distribution of the observations in the training, validation and test sets, different data sets can be constructed from the one original data set. For every simulation, the location and size of the anomalies will also be different. Every time ϕ is increased by 1 it is referred to as a single iteration.

4.3 Brier Score and the Breakdown point

The Brier Score is a score function that measures the accuracy of models which forecast binary outcomes.

$$BS = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{3}$$

equation (3) shows the formula of the Brier Score which equals the mean-squared-error. A perfect model would have a Brier Score of 0; a model without any information would just predict 0.5 always, leading to a Brier Score of 0.25. A model is thus broken down if the Brier Score is above 0.25, since it adds no information at that point. In this paper I increase the proportion of anomalies until the Breakdown point is reached to check model robustness. The Brier Score is always evaluated out-of-sample, since a good in-sample fit on corrupted data could have horrible consequences for the predicting power of the model out-of-sample. If a model is not broken down yet, but the parameters of the network are almost 0, I refer to it as a collapsed model.

4.4 Logarithmic Score

Another scoring rule widely used is the Logarithmic Scoring function. This scoring tool measures the performance of a forecasted probability. By taking the natural logarithm of the probability that the occurred event would happen. If an event that occurs had a 40% chance of happening, according to the model, the Logarithmic Score would be the natural logarithm of 0.4. If that event would not happen the score would be the logarithm of 0.6. The total score of a set of observations can simply be computed by adding the scores of all individual observations.

$$LS = \frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i), \tag{4}$$

the higher the Logarithmic Score is, the better. The Logarithmic Score is unbounded from below since the minimum score of an observation is the logarithm of 0, which equals minus infinity. The maximum score of an observation is the logarithm of 1, which equals

0. The total score can thus be any number smaller than or equal to 0. A perfect model would have a a Logarithmic Score of 0.

4.5 Influence Function

4.5.1 Upweighting a training point

For the Influence Function I use the formulas derived in Koh and Liang (2017). The Influence Function gives the slope of a parameter, when an observation is increased in weight by an infinitely small amount. The Influence Function can be calculated for the parameters or for the loss of the model. The Influence Function for the parameters is given by

$$I_{up,params}(z) \stackrel{def}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta})^T, \tag{5}$$

where $z_i = (x_i, y_i)$, $H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta^2 L(z_i,\hat{\theta})$ is the Hessian matrix and is assumed to be positive definite (PD), and $\nabla_\theta L(z,\hat{\theta})$ is the gradient matrix for all variables in all observations. Koh and Liang (2017) works with a gradient vector where z is a variable, but since all z values ought to be taken from the in-sample data, they are known and can be filled in to create a gradient matrix with n by k items, where k equals the number of parameters and n the number of observations. The Hessian matrix has dimensions k by k. With a neural network with no hidden layers and a sigmoid activation function these matrices can be derived by hand, and $k = 10$ with the data generating process used. Since the model has no bias term the parameter vector and an x observation vector have the same dimensions.

$$\nabla_\theta L(z,\hat{\theta}) = \frac{d}{d\theta}(Y-\hat{Y})^{\circ 2} = \frac{d}{d\theta}(Y-S)^{\circ 2} = -2(Y-S)\circ\frac{d}{d\theta}S = -2(Y-S)\circ S', \tag{6}$$

where $\frac{d}{d\theta}S$ is set to S' for simplicity and S is the n by 1 vector of the sigmoid function. The derivative of the sigmoid function with respect to the parameters equals $S\circ(1-S)\circ X$, where X has n rows and k columns. $\circ$ is the element wise product also known as the Hadamard product. The element wise product of a vector and a matrix is defined as long as the matrix and the vector share the same number of rows or columns or the vector is a scalar, and $S^{\circ n}$ equals the matrix element wise multiplied with itself n times. This is done for each parameter and each observation leading to a n by k matrix. The Hessian is calculated for one observation at a time and then takes the average value. For one observation the gradient matrix becomes a vector. A single iteration of the Hessian calculation can be derived as follows.

$$\begin{aligned}
\nabla^2_{\hat{\theta}} L(z_i, \hat{\theta}) &= \frac{d}{d\theta} - 2s_i(1-s_i)(y_i - s_i)x_i \\
&= -2\frac{d}{d\theta}\left\{y_i s_i + (-y_i - 1)s_i^2 + s_i^3\right\} x_i \\
&= -2\left\{y_i s_i' + 2(-y_i - 1)s_i s_i' + 3s_i^2 s_i'\right\} x_i \\
&= -2\left\{y_i + 2(-y_i - 1)s_i + 3s_i^2\right\} s_i' x_i \\
&= -2x_i^T\left\{y_i + 2(-y_i - 1)s_i + 3s_i^2\right\} s_i(1-s_i)x_i,
\end{aligned} \tag{7}$$

taking the average of all values of (7) gives the Hessian matrix.
$H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} -2x_i^T\left\{y_i + 2(-y_i - 1)s_i + 3s_i^2\right\} s_i(1-s_i)x_i$. Inverting this matrix and multiplying it by (6) and negative one gives (5), the influence of upweighing an observation on the coefficients. The result is a k by n matrix where the columns give the influence of that observation on the parameters on the rows.

$$I_{up,params}(z) = -\left\{\frac{1}{n}\sum_{n=1}^{n} -2x_i^T(y_i + 2(-y_i - 1)s_i + 3s_i^2)s_i(1-s_i)x_i\right\}^{-1} [-2(Y-S)\circ S\circ(1-S)\circ X]^T, \tag{8}$$

the Influence Function of the loss can also be calculated. With this function a linear approximation can be made of the influence of training without an observation without doing so.

$$\begin{aligned}
I_{up,loss}(z) &\stackrel{def}{=} \left.\frac{dL(z, \hat{\theta}_{\epsilon,z})}{d\epsilon}\right|_{\epsilon=0} \\
&= \nabla_\theta L(z, \hat{\theta}) \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} \\
&= -\nabla_\theta L(z, \hat{\theta}) H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})^T,
\end{aligned} \tag{9}$$

Using the derivations for the gradient and the Hessian matrix this becomes (10). The result is a n by n matrix where the columns represent the observations being increased in weight, and the rows represent the observations for which the increase in loss is being displayed. Since the Hessian is symmetric and is multiplied with a matrix and the transpose of the same matrix, the influence matrix is also symmetric. This means that the influence of an increase in the weight of observation 1 on the loss of observation 2, equals the influence of an increase in the weight of observation 2 on the loss of observation 1. To find the total influence on the loss of all observations, the column sum has to be taken from equation (10).

$$I_{up,loss}(z) = [2(Y-S)\circ S\circ(1-S)\circ X] \left(\frac{1}{n}\sum_{n=1}^{n} -2x_i^T(y_i + 2(-y_i-1)s_i + 3s_i^2)s_i(1-s_i)x_i\right)^{-1} [-2\circ(Y-S)\circ S\circ(1-S)\circ X]^T, \tag{10}$$

4.5.2 Perturbing a training input

Koh and Liang (2017) also derive Influence Functions, where instead of having a training data point increase in weight, the value is modified. The Influence Function calculates the influence when ϵ mass is changed from $z = (x, y)$ to $z = (x + \delta, y)$. The Influence Function on the parameters becomes

$$\begin{aligned} I_{pert,params}(z) &\stackrel{def}{=} \left.\frac{d\hat{\theta}_{\epsilon,z_\delta - z}}{d\epsilon}\right|_{\epsilon=0} \\ &= I_{up,params}(z_\delta) - I_{up,params}(z) \\ &= -H_{\hat{\theta}}^{-1}[\nabla_\theta L(z_\delta,\hat{\theta}) - \nabla_\theta L(z,\hat{\theta})]^T \\ &\approx -H_{\hat{\theta}}^{-1}[\nabla_x \nabla_\theta L(z,\hat{\theta})]^T\delta \qquad for\,||\delta|| \to 0, \end{aligned} \tag{11}$$

with $-H_{\hat{\theta}}^{-1}[\nabla_\theta L(z_\delta,\hat{\theta}) - \nabla_\theta L(z,\hat{\theta})]^T$ a three dimensional matrix of where each layer represent the influence for a given δ. In that case, $\nabla_\theta L(z_\delta,\hat{\theta})$ also becomes a three dimensional matrix where each slice has a different δ value and $\nabla_\theta L(z,\hat{\theta})$ also becomes a three dimensional matrix but each slice is the same since it does not depend on δ. The inverted Hessian should then be multiplied with each slice separately, and also the transpose should be done for each slice independently. The Influence Function on the loss when perturbing an input observation is derived in the appendix.

4.5.3 Out-of-sample Influence Functions

Koh and Liang (2017) show that the influence on the loss can also be easily calculated for data points not in the training set. The only difference is that the $\nabla_\theta L(z,\hat{\theta})$ term in the beginning of the product should be replaced by $\nabla_\theta L(z_{test},\hat{\theta})$. Where z_{test} is the vector of observations in the out-of-sample set. This gives the following equations.

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L(z_{test},\hat{\theta})H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta})^T, \tag{12}$$

$$I_{pert,loss}(z, z_{test}) = -\nabla_\theta L(z_{test},\hat{\theta})H_{\hat{\theta}}^{-1}\nabla_x\nabla_\theta L(z,\hat{\theta})^T, \tag{13}$$

where the columns represent which observation is being increased in weight or perturbed, and the rows display the influence on the loss of that observation.

4.5.4 Influence Functions more hidden layers

If the amount of hidden layers is increased, the Influence Functions become more complex. Since the coefficients of neural networks with hidden layers are difficult to interpret, I focus only on the influence of upweighting an observation on the loss, as seen in equation (9) . For the derivation of the gradient, the derivative of the prediction has to be taken with respect to θ, the parameters vector connecting the input to the first hidden layer. In the case of the neural network with a single hidden layer, the derivative of the prediction $\sigma(\omega' Z)$ with respect to θ, can be calculated with the chain rule. Z is the vector of values of the hidden nodes, and ω is the vector of parameters connecting the hidden nodes to the final prediction node.

$$\frac{d}{d\theta}\sigma(\omega' Z) = \sigma(\omega' Z)\left\{1 - \sigma(\omega' Z)\right\}\omega' \frac{d}{d\theta} Z, \tag{14}$$

where $\frac{d}{d\theta} Z$ is the derivative of a sigmoid function with respect to its coefficients, the same equation when there are no hidden layers. It is thus equal to $Z(1-Z)X$. And by taking the derivative of that expression the Hessian can be obtained (7).

5 Results

5.1 No hidden layer network

5.1.1 $K = 5$

The results correspond to data where a uniform number between 0 and 5 is added to the first variable. $K = 5$, $n = 100000$.

Figure 2: The parameters over different percentages of noise, 2 different simulations, n = 100000

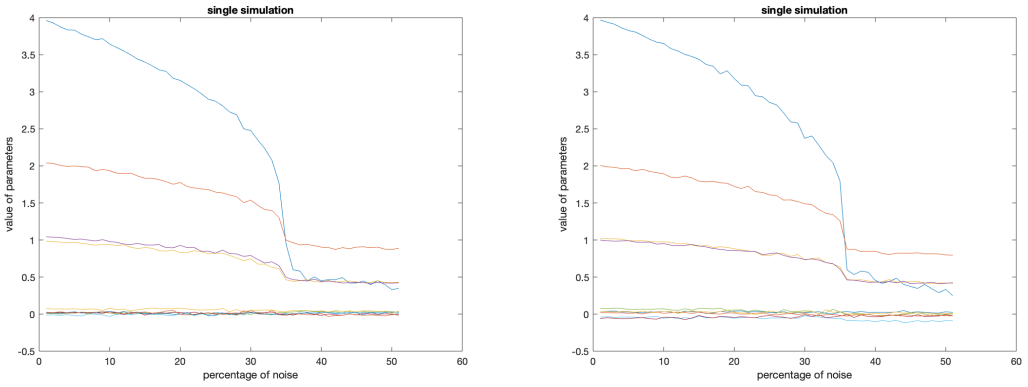
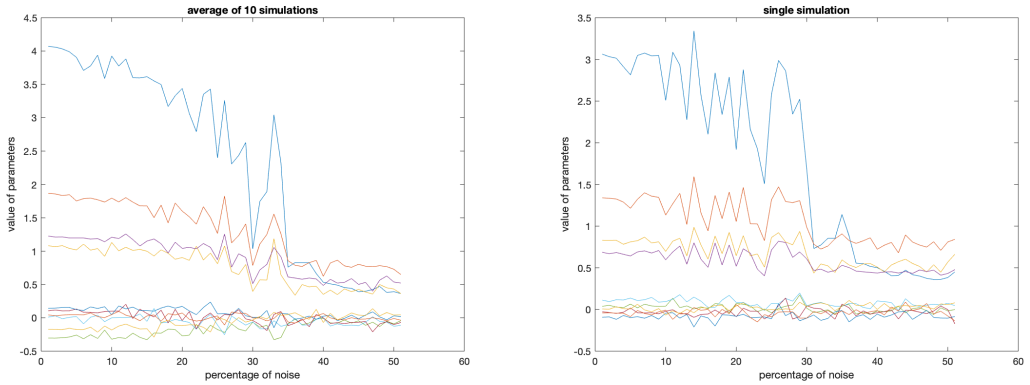


Figure 2 show a similar pattern for both simulations, indicating that with $n = 100000$ the stochastic element is reduced. Equation (2) shows that the parameters are similar to the true weights when there are no anomalies. The weight of the variable with the noise, variable 1, decreases to almost 0 as the percentage of noise increases in the data. This makes sense as the average value of the first variable increases but the effect does not. The effect per unit decreases in this scenario. All other significant parameters are decreasing also, to roughly 50% of the true value. All non significant variables do not really change and stay near the true value of 0. Remarkable is that the significant parameters decrease relatively slowly at first and then at 34/35% they drop steeply at once to a new value which remains relatively stable from thereon.

Figure 3: The parameters over different percentages of noise, n = 1000



The results become more noisy when there are fewer observations, as seen in 3. With fewer observations, the influence of the randomness of the outliers is more extensive, because of the law of large numbers. The proportion of anomalies when the model collapses is less stable, and the parameters sometimes increase. This can be explained by the fact that every iteration, the location and the value of the anomalies differ. When the model collapses depends thus on the location and on the value of the anomalies; with fewer observations, randomness has a larger influence. An interesting result is that all parameters are decreasing, which decreases the variance of forecasts.

Figure 4: The performance measures in-sample over different percentages of noise

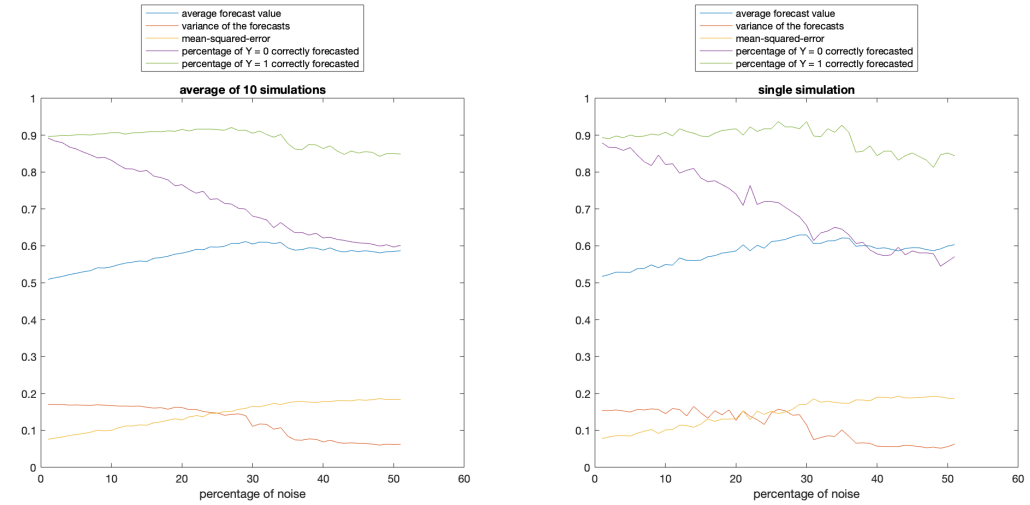
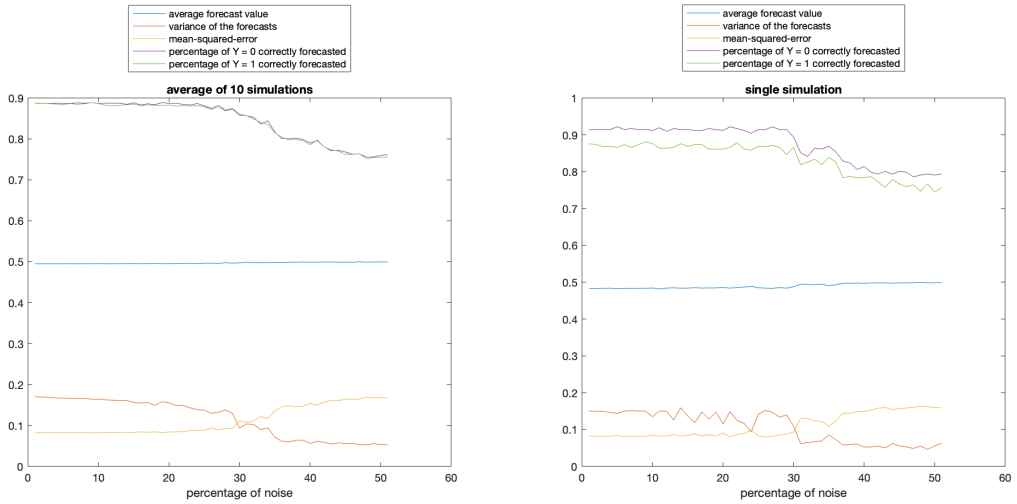


Figure 5: The performance measures out-of-sample over different percentages of noise



At first, the anomalies do not change the out-of-sample performance a lot. This changes when the proportion of anomalies is approximately 30%. As noise increases, so decreases the variance. This makes sense as the parameters are also decreasing when there is more noise, and lower parameters lead to more values being closer to 0.5. The difference between in-sample and out-of-sample is that in-sample the model has an upward bias, which increases until the model collapses and then drops slightly to a new level for the rest of the iterations. In-sample is better at labelling $Y = 1$ as $Y = 1$ since the average forecast value is larger than 0.5 while the actual average is almost exactly 0.5. Out-of-sample the accuracy is the same for both Y values.

Figure 6: The total accuracy over different percentages of noise

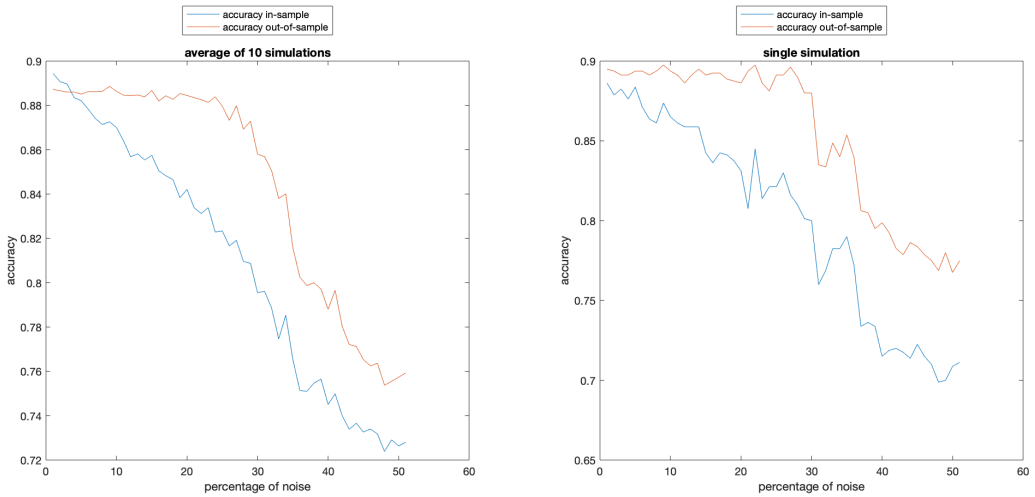


Figure 11 shows that in-sample the total accuracy drops almost linearly: the parameters barely change, but the influx of anomalies make it harder to predict well. The out-of-sample data is anomaly free: the performance remains stable until the parameters collapse which decreases the accuracy. What is odd about these results is that the out-of-sample accuracy does not seem to be affected by the noise at first. The change in parameters is not enough to make the model alter the forecast value, but figure 10 demonstrates that the mean-squared-error also stays stable until the parameters decrease rapidly. An explanation for the relative stability of the performance is that the slope of the sigmoid function is flat around the tails. Large parameters which decrease a bit can result in an comparable probability. Another remarkable result is that the accuracy out-of-sample is higher than in-sample for all levels of noise larger than 1% . This can be explained by the fact that the out-of-sample sets do not contain anomalies.

5.1.2 Dynamic K

In this section, K is set to different integers in order to investigate the influence of the K value on the model performance.

Figure 7: The parameters over different percentages of noise and for different K values

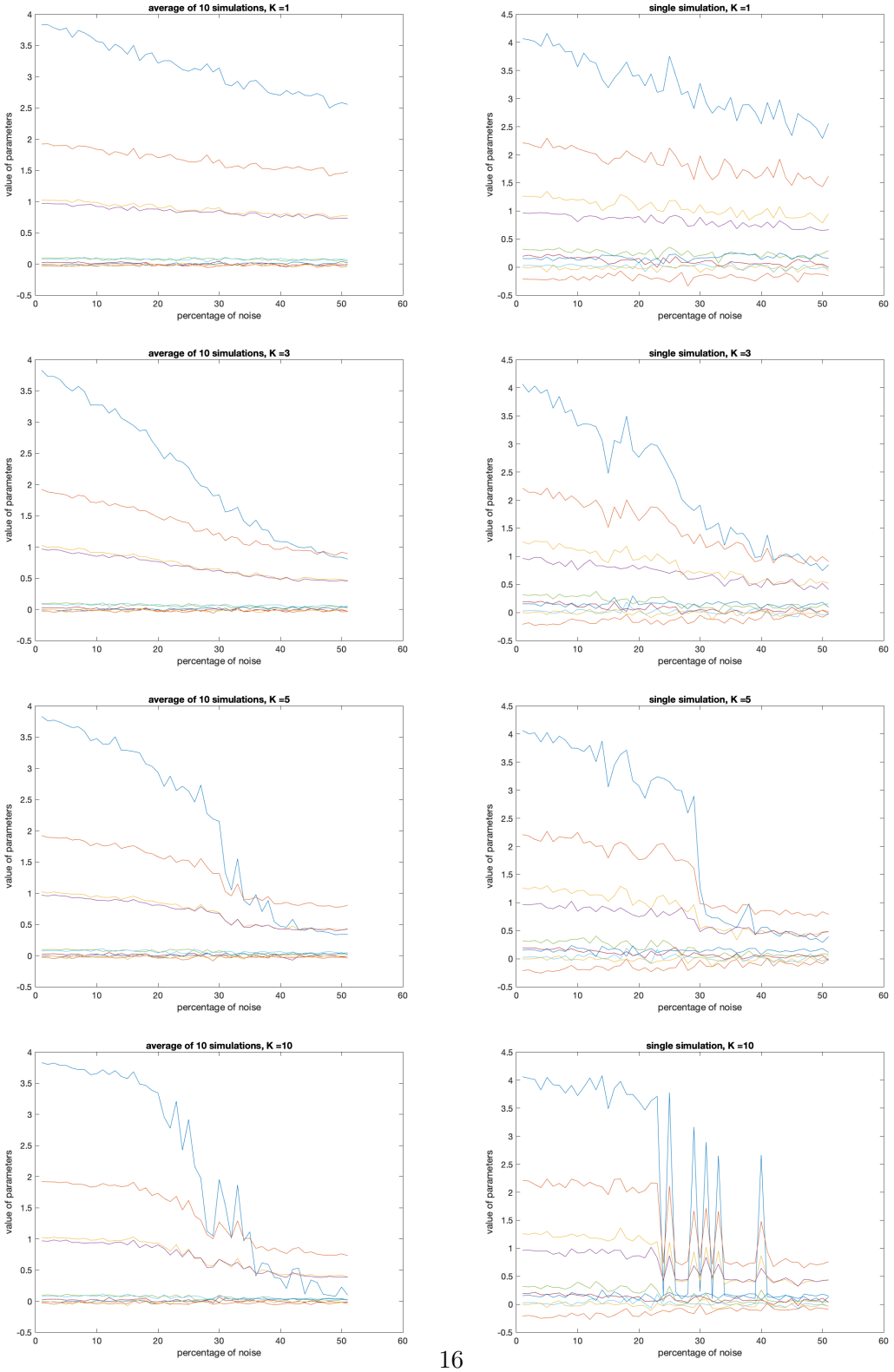
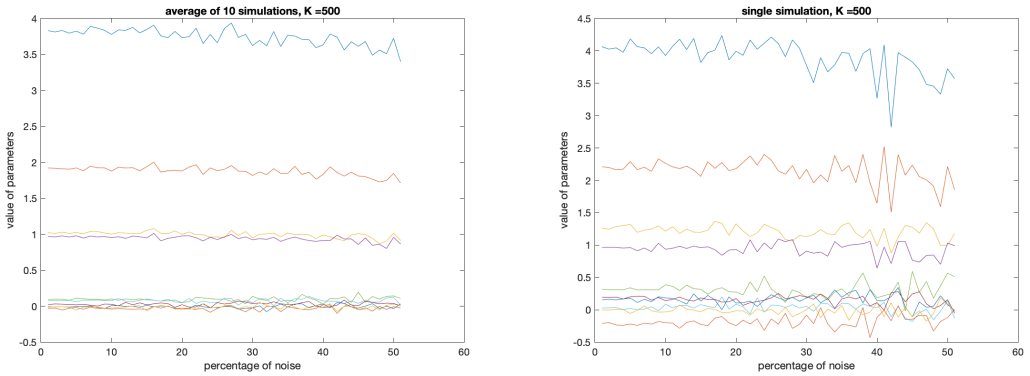


Figure 7 shows that the larger the K value, the sooner and more steeply the parameters of the model drop. If $K = 1$, the drop is almost linear, while if $K = 10$, the drop is more abrupt. The point where the parameters collapse also appears less stable when K is large. An explanation for this is that the larger the K value, the higher the variance of the anomalies is. These patterns do not hold when K becomes unreasonably large, as seen in the graphs below.

Figure 8: The value of the parameters for different percentages of noise, $K = 500$



When K is set to a very large value, the anomalies barely affect the model. In this case, the variable values are so extraordinary high, that trying to fit those observations well would mess up all other predictions. Usually, a colossal outlier combined with a quadratic loss function would have an enormous influence on the model, but as all predicted values are between 0 and 1, the loss is limited. The sigmoid activation function makes the model more robust. Even this simple no layer neural network appears to ignore absurd outliers. Even when 50% of the observations are anomalies, the model appears to ignore observations with huge anomalies added to them.

5.2 Single layer network

Most neural networks have at least one single layer. Some researchers however freeze all layers, except for the top one, in order to train that layer separately, Donahue et al. (2014). As the parameters of a neural network with a least one single layer are very hard to explain and comprehend, I focus mainly on the performance of the model and not on the parameter values. This neural network has the same data generating process as the previous network and has 9 nodes in the hidden layer. As before, $n = 1000$ and $K = 5$.

Figure 9: The performance measures in-sample over different percentages of noise

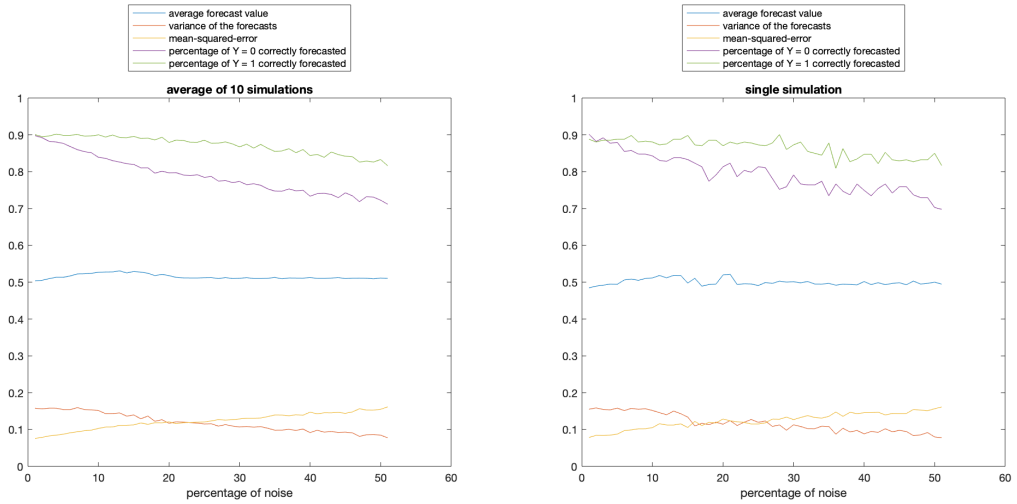
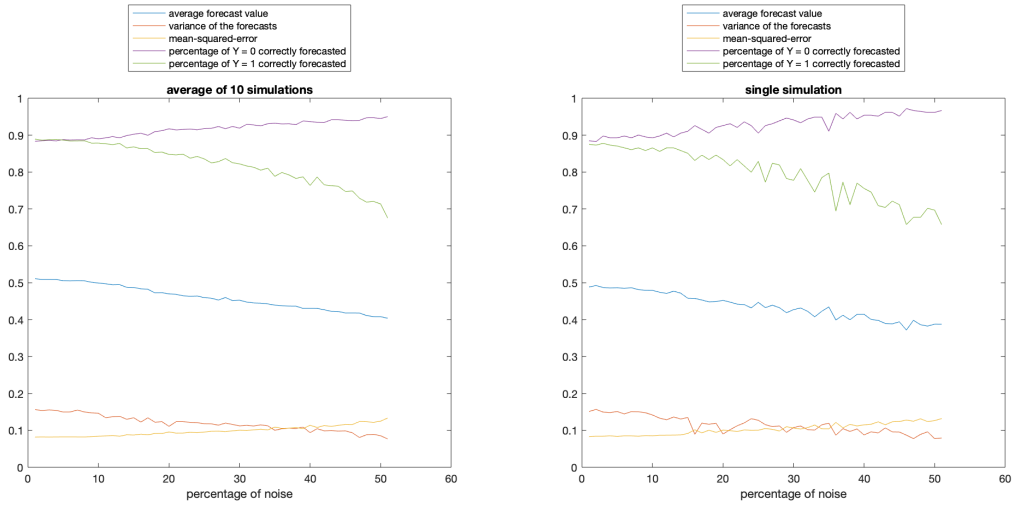


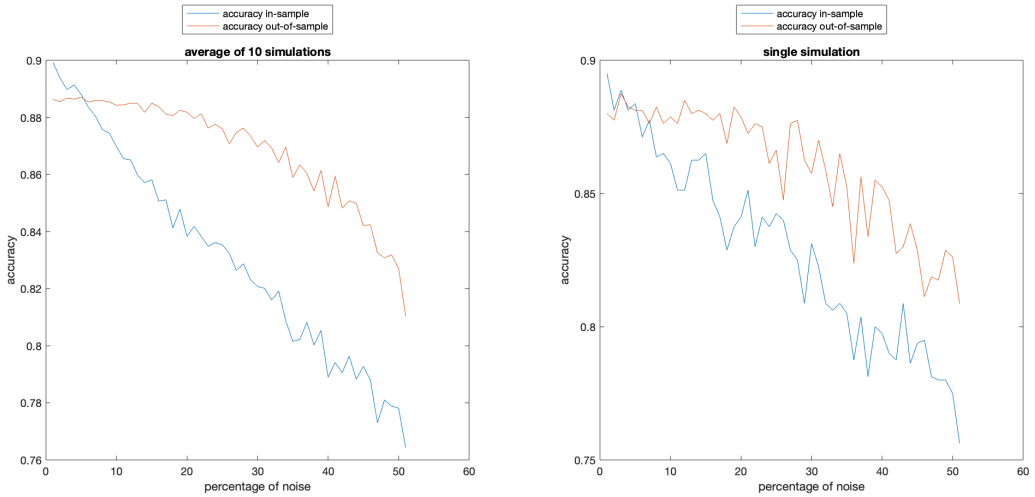
Figure 10: The performance measures out-of-sample over different percentages of noise



The early stopping algorithm prevents overfitting, leading to a worse performance in-sample, but a better performance out-of-sample. Similar to when there is no hidden layer, more noise leads to lower variance and a higher MSPE. In-sample the mean stays constant, since the in-sample mean of the true Y values does not change and is close to 0.5. This is not the case when there is no hidden layer, but a hidden layer with a bias term gives the model more flexibility. Out-of-sample the predicted mean decreases. The anomalies make X larger, which decreases the expected per unit influence X has on Y.

In-sample the model can use the bias term to keep the parameters lower and still have an average forecast value of 0.5. When out-of-sample the X values are lower, the model underestimates Y. In-sample the model labels more data $Y = 1$. This explains why a larger proportion of observations with Y values of 1 are correctly labelled. Out-of-sample it is the other way around. The out-of-sample X values are lower, so combined with the parameters based on the in-sample results, the model underestimates the proportion of Y values of 1.

Figure 11: The total accuracy over different percentages of noise



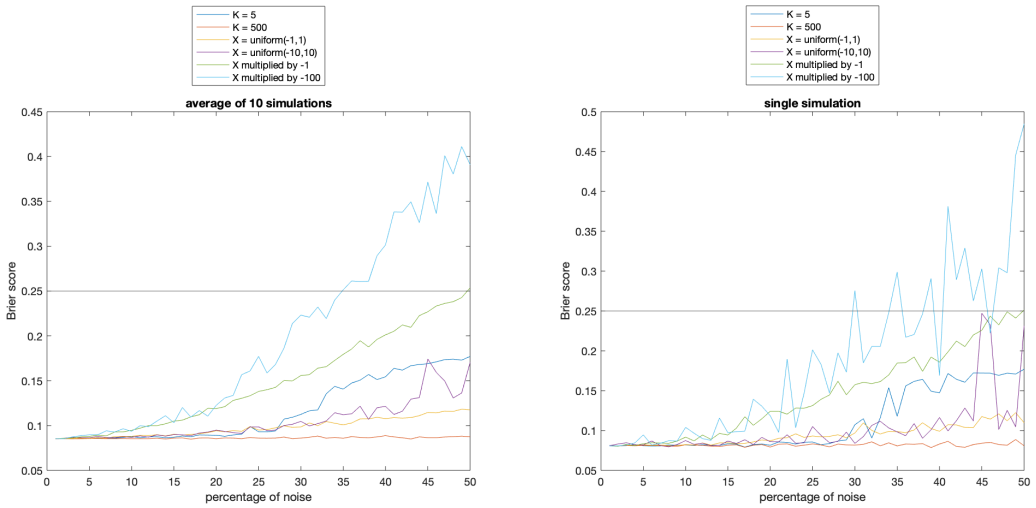
Similar to when there is no hidden layer, the out-of-sample forecasts perform better than in-sample forecasts, as there is no noise out-of-sample. The difference between the results with and without a hidden layer is that with a hidden layer, the decrease in performance is less abrupt, as a model with a hidden layer has more flexibility. When there are no anomalies, the out-of-sample forecasts perform better than in-sample. This is unusual, but it can result from the early stopping algorithm which prevents overfitting. When $/phi$ equals 0, the accuracy with a hidden layer is comparable to the accuracy without a hidden layer. This is a sign that the regularization method works in preventing overfitting.

5.3 Breakdown point

In this subsection n equals 1000. As stated before, a model is broken down if the Brier Score is 0.25 or larger. A Breakdown point shows how much corrupted data a model can handle before it performs worse than a model without information, that always gives 0.5 as a result. To reduce the amount of corrupted data needed to break a model down, it must first be determined what the most corrupting data is. Graph 8 shows that noise with a larger K value does not always have a larger influence on the model. The model

has the largest loss if the predicted value is a 0 or a 1, depending on the true value. With label changing this would be simple. Just change the label from a 0 to a 1, or the other way around, at the observations where the model would be the most certain of. When only considering X, multiplying all values with -1 makes the model assume that the effect is in reverse, that those values are negatively correlated with Y, instead of positively. I also change X in random numbers, to see how that would influence the Brier Score.

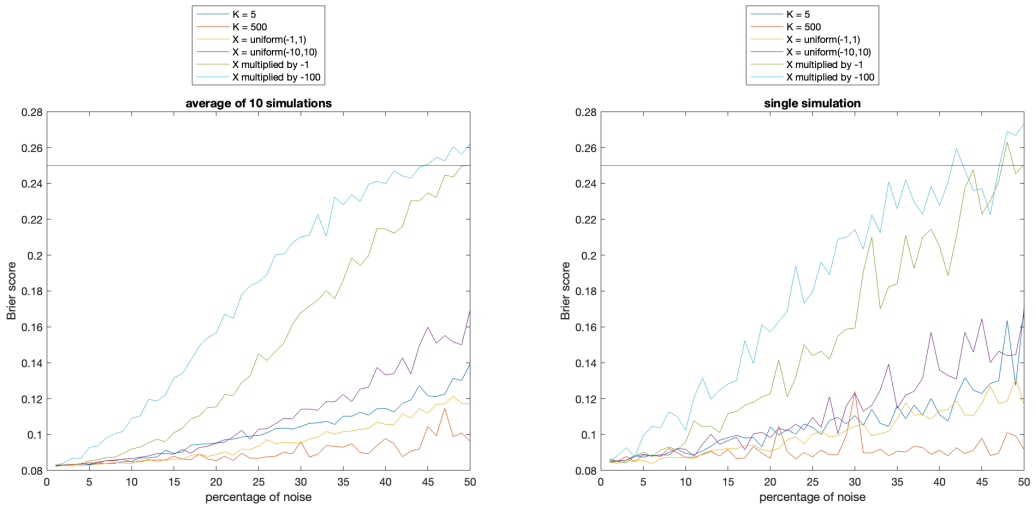
Figure 12: The out-of-sample Brier Score for different anomalies added to a no hidden layer network



The remarkable result is that most networks do not reach the Breakdown point. It looks like the anomalies often do not influence the parameters enough, and combined with the still large percentage of uncontaminated observations, the model still has some predicting power out-of-sample. The Brier Score also demonstrates that a K of 5 has a larger influence on the model than that of a significantly higher K value. The model seems to predict the non corrupted data reasonably well enough to outperform a random prediction. In previous results it was clear that a K value of 5 would decrease all significant parameters greatly, reducing the variance of the observations. As long as the parameters are positive, the model has some predicting power and will outperform a random model. The random uniform outliers do not seem to have a very large effect on the Brier Score. These anomalies can have a positive and a negative influence on the parameters, so having many of them can cancel each other partly out. Again all Y values are bounded between 0 and 1, so a single anomaly also has a bounded influence on the loss and thus the parameters. The only type of anomalies that really appear to break the model down fast, are the ones where X is multiplied by a negative scalar. This does not decrease the parameters to a smaller positive value, nor makes them more noisy, but rather changes the sign. With these methods, a relatively small portion of anomalies (around 35%) can have the model perform worse than a random model. In this situation

the uncontaminated data still outnumbers the anomalies 2 to 1. This combined with the fact that the loss of an observation is bounded might make you think that the model still has some predicting power. However, even a perfect model would have a Brier Score larger than 0, given the noise in the data generating process. Still, 35% of these extreme anomalies is substantial, especially as in the real-world the proportion of anomalies is often between 1 and 10%, Frénay and Verleysen (2013).

Figure 13: The out-of-sample Brier Score for different anomalies added to a single hidden layer network

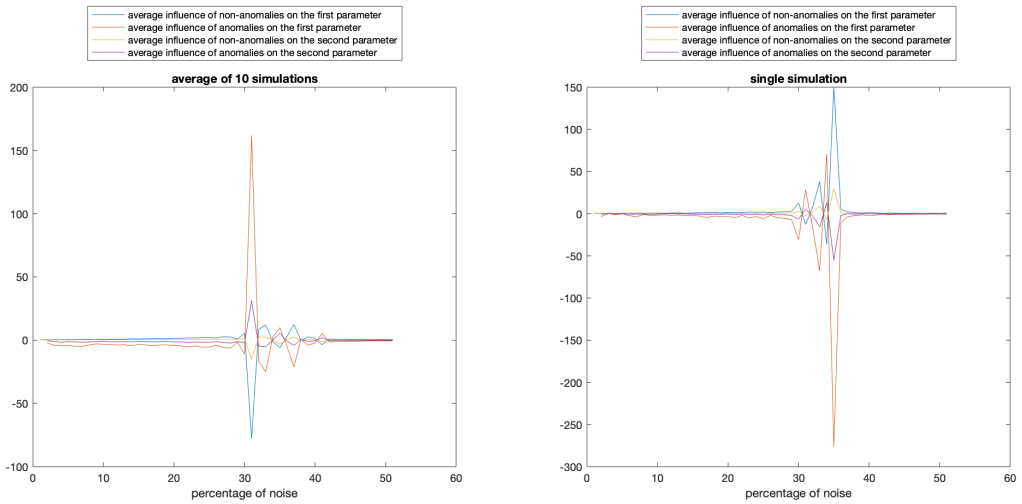


A model with a hidden layer with 9 nodes is not significantly more robust for most anomalies. The model without a hidden layer is already reasonably robust to anomalies when forecasting binary outcomes. The use of the early stopping algorithm also makes networks more robust. Only if the anomalies consist of multiplying all X values by -100, there is a significant improvement in performance when a hidden layer is added. A model with a hidden layer and 9 nodes has the capacity to recognize more complicated patterns. This way it can understand that X values might influence Y differently if they are far above or below 0. Still, anomalies consisting of multiplying all X values by -100 have the most destructive influence on the model, and are able to break it down when they make up almost 50% of the data. Again, in this case, anomalies with a K of 500 influence the model less than anomalies with a $K = 5$. Once more, this is a demonstration that a network which forecasts binary outcomes has a limited loss and can effectively ignore certain outliers. Most anomalies have a limited influence on the model when they make up a fairly small proportion of the data, which is true in most real-world cases.

5.4 Influence Function

In the following segment $K = 5$ and $n = 1000$. At first a model without a hidden layer is used and then a model with one hidden layer and 9 nodes in that layer is used. The same data generating process as always is used. With equation (5) the influence of increasing an observation on the parameters is calculated.

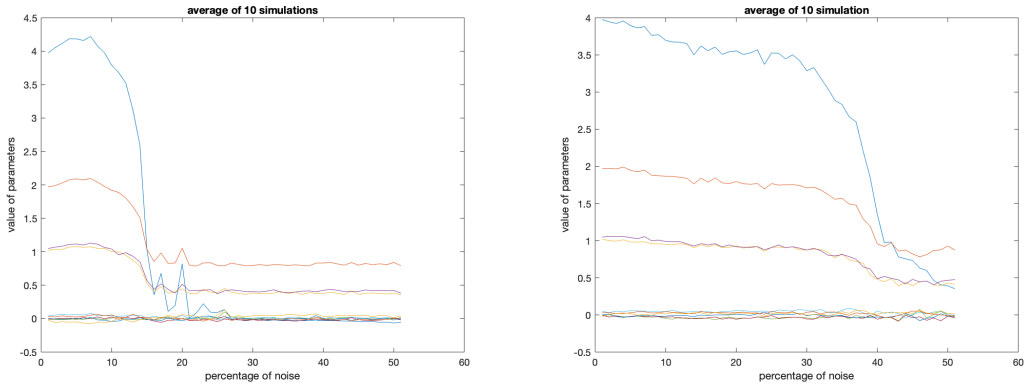
Figure 14: The average influence of all observations on the parameter of a variable



The influence of anomalies on the parameters is negative, since anomalies are observations with larger variable values; the influence of non anomalies is positive. The noise is added to the first variable so anomalies influence that parameter more than the second parameter. The second parameter is smaller than the first parameter so the absolute influence on the second parameter is closer to zero. After the model has collapsed, the influence is almost zero: after an amount of iterations, adding anomalies only barely changes the parameters and the model. Around the point where the parameters collapse, the influence of anomalies on the parameters can have a very large negative value. This is not exceptional, as around that point the weights often display a large negative drop. An exceptional result is that around the point where the parameters collapse, the average anomaly can have a large positive influence on the first parameter. This does not make sense intuitively, but it can be the result of the optimization methods the network uses, such as early stopping.

With equation (11) training set attacks can be created. Sometimes a model collapses with ϕ percent anomalies, but it recovers in the next iteration. This shows that some observations have a larger influence on the parameters than others, or that the size of the anomalies can make the model collapse earlier. By only adding the noise to the ϕ^{th} percent of observations with the most increasing or decreasing influence on the first parameter, the model can collapse much sooner. For the graph, formula (11) is used.

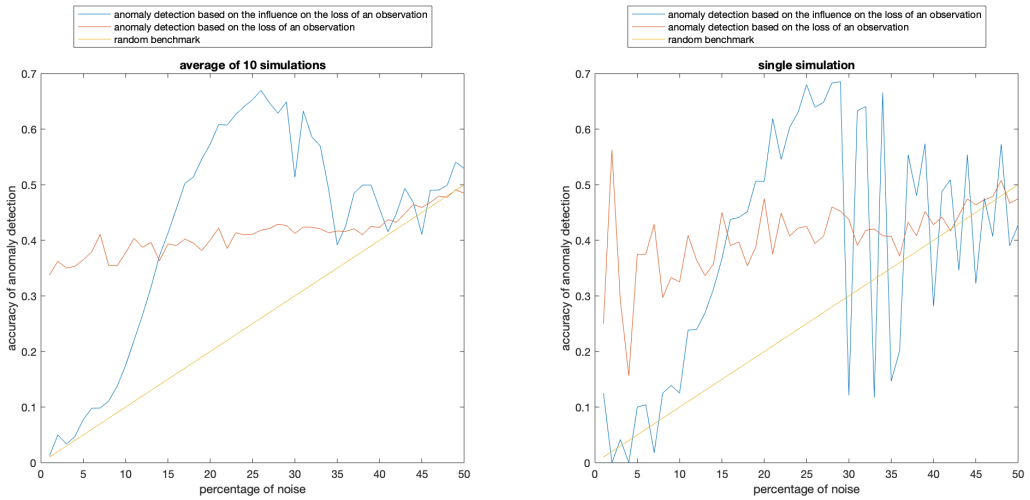
Figure 15: In the left graph, noise was added to the observations with the largest decreasing influence on the first parameter. The right graph shows the parameters where noise was added to the observations with the largest increasing influence on the first parameter.



In the first graph the model collapses at around 10%, in the second graph this happens after 40 iterations. Some observations influence the first weight a lot more. When the model collapses is dependent on which observations noise is added to, and on the amplitude of the noise.

The influence of increasing an observation on the loss can be used to detect anomalies. With (9) the ϕ percent of observations can be selected, which have the largest loss increase on all observations. Comparing this to the observations with the ϕ percent highest loss, we can see which method is best at detecting anomalies.

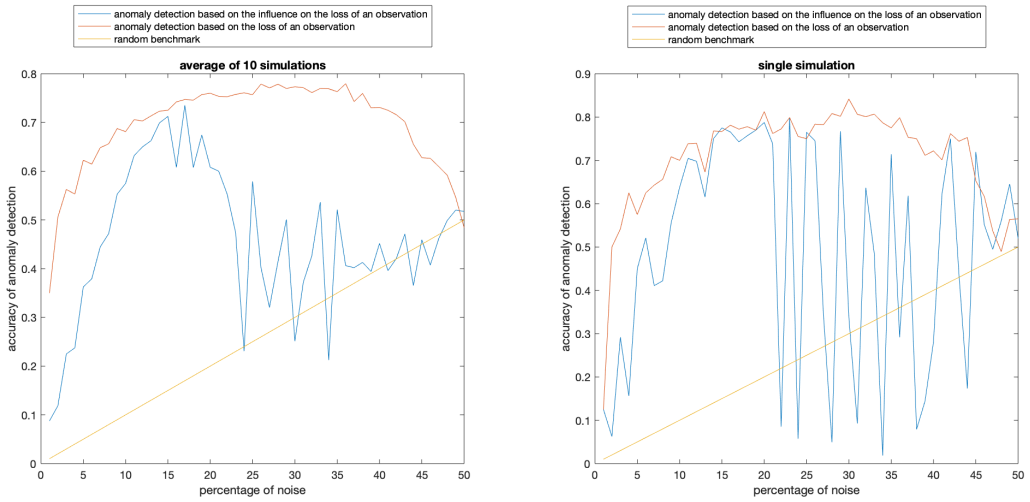
Figure 16: The accuracy of anomaly detection systems



The surprising result is that often observations with a high loss have a larger probability of being an anomaly, than observations which influence the total loss the most. And in most real-world scenarios the percentage of anomalies is between 1 and 10% Hampel et al. (2011). As the number of anomalies increases it is of course easier to detect an anomaly. Both models outperform the random benchmark. The best performance is seen when detecting is based on the Influence Function and the proportion of anomalies is roughly 25%, which is not very likely in the real-world. Even though both methods outperform the random benchmark, many non-anomalies are incorrectly flagged. Omitting these observations could remove valuable information. Both models therefore do not show great results; this is maybe due to the relatively small size of the noise that was added. In all observations noise was only added to the first variable, and this noise was not that large, which could explain the relatively poor performance.

When repeating the same experiment with more obvious anomalies (where all X values are multiplied by -1), the results look differently.

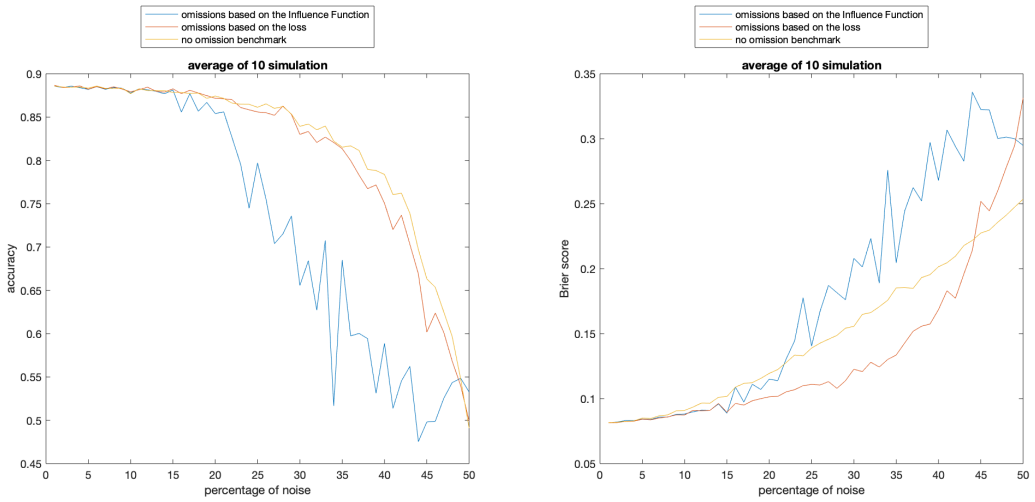
Figure 17: The accuracy of anomaly detection systems



When the anomalies are more absurd, the detection systems perform better. Once again, in this case the loss is usually a better indication whether an observation is an anomaly or not, than the influence that observation has on the loss. Both models beat the benchmark comfortably in most cases. The methods are able to detect well. Performance of detection based on the loss function does decrease when the proportion of anomalies is very high, but at that point the model probably breaks down and would not make sense anyway. If anomalies make up a majority of the observations they have the pattern the models tries to fit and the regular observations can be seen as anomalies/outliers. Detection based on the influence on the loss peaks earlier in performance as also can be seen in the plots in 21.

The Influence Function and the loss can be used to filter observations that might be anomalies. Removing the ϕ percent of observations with the highest loss, or removing those who influence the loss the most, could improve the performance of the model. A downside is that it would be expensive, since the model has to be trained twice, first to find the values of the loss and the influence on the loss. Then using those values to omit some observations, and then the model has to be trained again with fewer observations. Another downside is that some non-anomalies will be flagged as anomalies and thus be omitted from the data. The statistics to compare these methods, both with each other and with the standard model of not omitting variables, will be the Brier Score and the labeling accuracy, both out-of-sample.

Figure 18: The out-of-sample performance of the omission methods



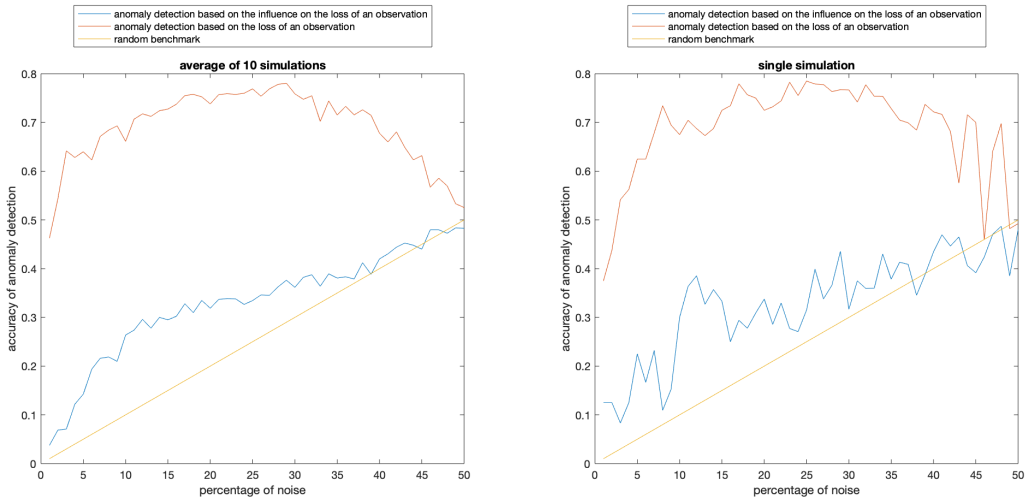
Using the Influence function to omit observations decreases the performance of the model. Omissions based on the loss can decrease the Brier Score under certain circumstances. The Logarithmic Score when omitting on the loss also improves when there are many anomalies, as seen in the appendix. Omitting based on the loss seems to be more effective when evaluating based on score functions that are more exponentially influenced by loss, such as the Brier Score or Logarithmic Score. The accuracy does barely changes when the loss is used to flag and omit potential anomalies. It must be noted however that when the proportion of anomalies is low, which is the case in real-world data, all models perform roughly the same. Overall it does not appear that the omission methods outperform the original model significantly, especially considering the fact that they cost more time. The upside of having less anomalies is not significantly greater than the downside of having less true observations. Influence Functions give a linear approximation of leaving an observation out of the training set. This is not the exactly the same as actually doing so. Another issue is that the Influence Functions change if

other observations are omitted. Having similar observations removed, could change an observation from begin grossly over-represented to being underrepresented. Loss is not always bad for a model. If outliers are part of the data generating process, having them in the training set can give the network a better sense of how the data is generated, especially since the potentially removed outliers can be rare.

5.4.1 Single hidden layer

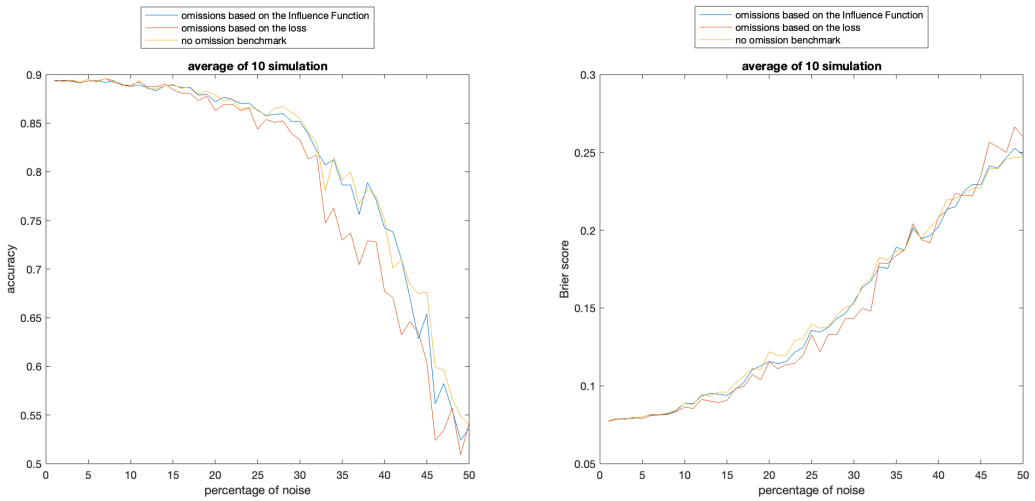
In this section the anomalies added consist of multiplying all X values by -1.

Figure 19: The accuracy of anomaly detection systems



With a hidden layer, the loss is again the more reliable way of detecting anomalies. The Influence Function performance is comparable to the random benchmark. Observations with a high influence on the loss have an almost equal percentage of being an anomaly than a random observation. When there are many anomalies, detection based on the loss decreases its performance, but this can be explained by that the model would breakdown with many anomalies.

Figure 20: The out-of-sample performance of the omission methods

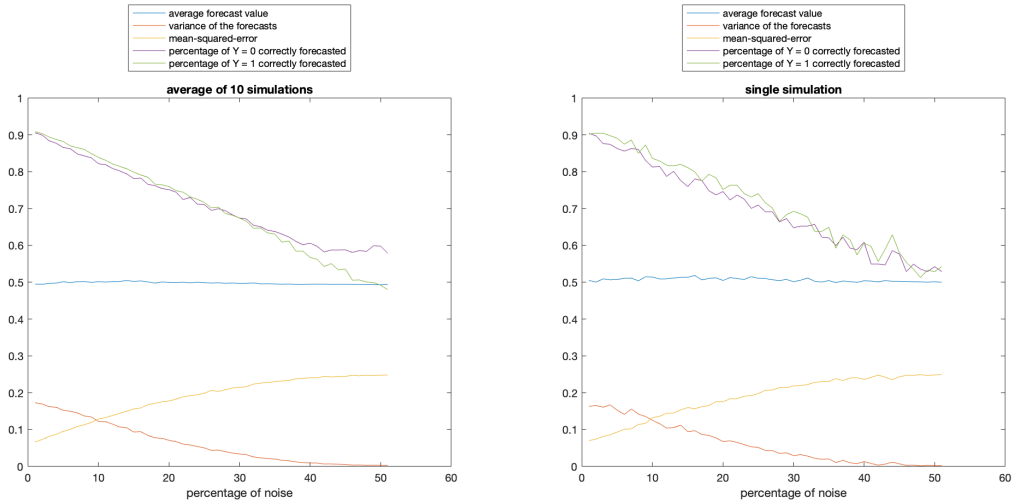


Omissions based on the Influence Function seem to perform comparably to the benchmark of not omitting any observations. Omissions based on the loss lead to lower accuracy, but in some cases also to a better Brier Score. The Logarithmic Score in the appendix tells the same story as the Brier Score does. An explanation for the poor performance of the methods is that influence on the loss, or in-sample loss, are not necessarily bad for the models or a result of anomalies. Edge cases are rare by definition, so removing them could have an extra hard impact on the model. Often a few edge cases teach the model complicated patterns so having fewer of them could have strong consequences.

5.5 Real-world data

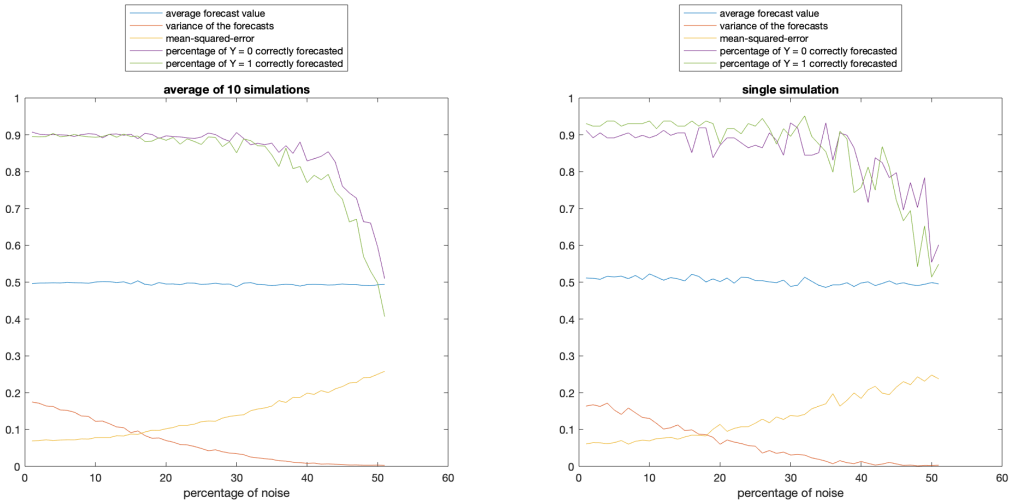
In this section the anomalies added consist of multiplying all X values by -1.

Figure 21: The performance measures in-sample over different percentages of noise



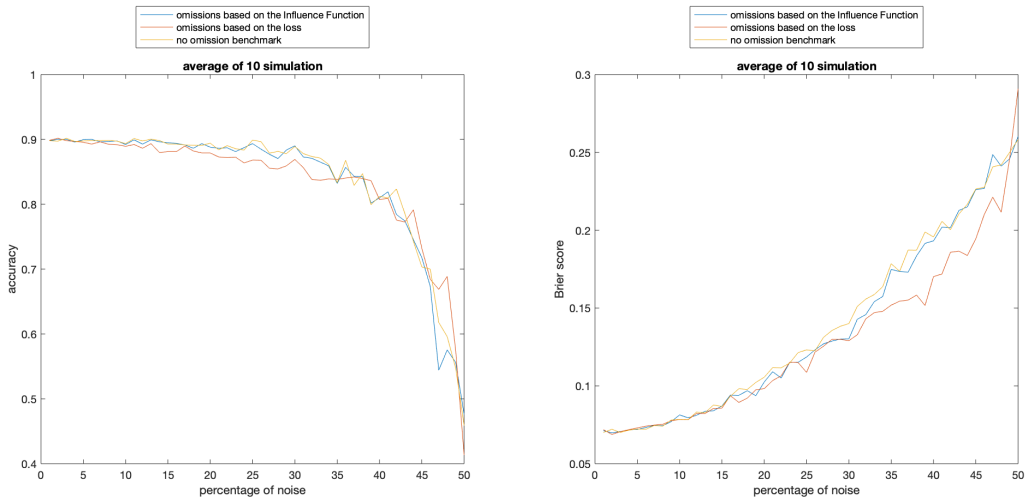
The similar pattern appears over here, more anomalies means less prediction variance. The less information a model has, the more it is inclined to predict all outcomes as 0.5. When the forecast variance declines, the mean-squared-error (or Brier Score) increases. Note that only where anomalies make up almost 50% of the data, the Brier Score reaches the threshold Breakdown value of 0.25. Even with many absurd anomalies the model still has some predicting power. The forecast accuracy drops linearly as the anomalies increase in frequency, which is a double edged sword, as it affects parameter values and makes the data the model tries to forecast more noisy.

Figure 22: The performance measures out-of-sample over different percentages of noise



In the out-of-sample data the model does not seem to have a bias. This is because multiplying observations with -1 increases the probabilities if the original probabilities are below 0.5; otherwise it decreases them, which has a cancelling out effect. At first the performance remains relatively stable, which is an indication that the model ignores the anomalies if there are not too many. This is possible since the loss is constrained when forecasting binary outcomes. When the proportion of anomalies is unreasonably large, the performance drops steeply, but the mean-squared-error, or Brier Score, stays below 0.25 for a long period. This indicates that the model still has some predicting power even in extreme situations.

Figure 23: The out-of-sample performance of the omission methods



Omitting variables based on these methods does not affect the over-all performance too much. The Logarithmic Score in the appendix also shows that omitting variables barely affects the performance in this scenario. Again the loss function over-performs when it comes to the Brier Score, but there are few indications that these methods are worth the extra training time. It most also be noted that omitting also has its risks and should therefore only be considered if the performance increase is significant. This does not appears to be the case. Omitting based on the loss does lead to a noticeable better Brier Score when the data has allot of noise but the accuracy does not outperform in that situation. And again, in real-world data the proportion of anomalies is often under 10%, and in those situation the omission methods do not outperform the benchmark. The graphs are not continuous in their decrease or increase, indicating that 10 simulations with 1000 observations may not be enough to draw strict conclusions.

6 Conclusion

Models without a hidden layer do not change dramatically if the proportion of anomalies is low. With $K = 5$, the parameters decrease slowly until the anomalies reach a certain threshold where the parameters collapse almost at once. When the parameters collapse, so does the performance of the model. With smaller parameters, the variance of predictions decreases and so does the accuracy. The first parameter where the anomalies are added decreases to 0, and the other significant parameters decrease to roughly 50% of their original value, with $K = 5$. The anomaly threshold where the model collapses depends on what observations anomalies are added to (some observations are more influential than others). it also depends on the size of the noise added to the observations. With the Influence Functions a training set attack can be done, where specific observations are altered to decrease model performance with as few anomalies as possible. This can lead to collapsing parameters with roughly half of the amount of anomalies compared to if anomalies are randomly distributed across observations. Anomalies have a decreasing influence on the parameters, exceptions occurring sometimes around the point where they collapse. This can be the result of the optimization techniques used, such as the early stopping algorithm.

If K equals to a lower value, the drop in height of the parameters is smaller and less abrupt. If the value of K is very high, the model ignores these observations and the Brier Score is better than with a lower K. A model needs many, or extreme, anomalies to have a Brier Score above 0.25, the famous Breakdown point. With all K values tried, the Brier Score stays under 0.25 for less then 50% anomalies. Even with more extreme anomalies, for instance multiplying all X values by -1, the network is still able to handle quite some of them before it breaks down. When Y is a double between 0 and 1, it limits the maximum loss a given observation can have and thus bounds the observation's maximum influence.

When having a hidden layer, the decrease of the out-of-sample performance is a bit less abrupt and more continuous. This is not the case in the real-world data where the anomalies have a different data generating process. A network with a hidden layer handles some types of anomalies better and needs more of them to reach the Breakdown point. The more complicated a neural network is, the more it can understand difficult patterns, including anomalies. In almost all situations, observations with a high loss have a higher probability of being an anomaly than observations with a high influence on the loss. But the Influence Function still outperforms the random benchmark when it comes to detecting anomalies. Removing a observation with the highest loss can result in a slightly higher Brier Score, but it does decrease the model accuracy. Omitting observations with a high influence on the loss results in a slightly worse or similar performance, according to the accuracy and Brier Score. Thus, in most scenarios, omitting would not be worth it. Apart from extra training time omitting also has its risks. For the omissions method I assume that the proportion of anomalies is known, which is likely not the case in real-world data. Furthermore it must be noted that an Influence Function gives the change in loss if a given observation will be increased in weight by an infinitely small number. It

does not give the change in loss when omitting it completely. If observations are omitted the Influence function of the observations still in the database also change. Therefore the more observations are omitted the less reliable the calculated Influence functions become. Another interesting point is that loss is not necessarily bad since it are weird outliers that teach models and humans how patterns work.

7 Discussion

In this paper I have researched the influence of anomalies on neural networks and used Influence Functions to filter them. Neural networks still are black boxes and the more layers they have, the more difficult it is for us to understand them. The current COVID-19 crisis shows us how difficult humans find it to comprehend exponential growth, let alone comprehending a neural network with hundreds of hidden layers. Many results in this thesis could be topics to investigate in other studies. For example, why the parameters of a neural network without a hidden layer drop so suddenly when the proportion anomalies reach a certain threshold. Another interesting topic for research would be the combination of previous studies which focused more on k-NN or other loss functions, with the Influence Functions. Maybe with different anomalies and other models, filtering would be worth the extra computing time and result in a significantly better Brier Score. This paper also shows the limits of linearly interpolating the Influence Function, to approximate the effect omitting variables has. Perhaps with the second order Influence Function used in Croux et al. (2005) a better approximation can be made. With even higher order Influence Functions a Taylor approximation can be made that better models the influence of omitting variables on the loss. Another idea is to omit less than ϕ percent of observations, maybe 50% or even a smaller percentage. In that scenario the removal of less non-anomalies may result in improving performance, but here the proportion of anomalies will also need to be known, or at least an educated guess is necessary. Perhaps a clever algorithm can be constructed that can determine the optimal fraction of observations being omitted without knowing the proportion of anomalies. If part of the data is known to be anomaly free (perhaps a result of better sensors or verification) it can be used as a verification set, to check with what percentage of observations being omitted the greatest performance is obtained. Also other AI models can be considered such as Random Forests and Support Vector Machines. Another conclusion of this thesis which needs more research is the finding that at the point where the parameters of the model collapse, the average anomaly can have a positive influence on the parameters. What is also hard to comprehend, is the finding that in some situations anomalies that do not make sense at all are not overwhelmingly represented in the set of observations with the most influence on the loss function.

References

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*,

pages 192–204, 2015.

Christophe Croux, Gentiane Haesbroeck, and Kristel Joossens. Logistic discrimination using robust estimators. *Available at SSRN 876912*, 2005.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

Moumen T El-Melegy and M Essai. From robust statistics to artificial intelligence: M-estimators for training feed-forward neural networks. In *Al-Azhar Engineering Ninth International Conference (AEIC)*, volume 2, pages 85–100, 2007.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE, 2004.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

Jakob Raymaekers and Peter J Rousseeuw. Fast robust correlation for high-dimensional data. *Technometrics*, pages 1–15, 2019.

Peter J Rousseeuw and Wannes Van Den Bossche. Detecting deviating data cells. *Technometrics*, 60(2):135–145, 2018.

Andrzej Rusiecki. Robust learning algorithm based on lta estimator. *Neurocomputing*, 120:624–632, 2013.

Andrzej Rusiecki, Mirosław Kordos, Tomasz Kamiński, and Krzysztof Greń. Training neural networks on noisy data. In *International Conference on Artificial Intelligence and Soft Computing*, pages 131–142. Springer, 2014.

A Formulas

The Influence Function on the loss when perturbing an observations equals.

$$\begin{aligned} I_{pert,loss}(z) &\stackrel{def}{=} -[\nabla_\theta L(z, \hat{\theta}_{z_\delta,-z})]^T\Big|_{\delta=0} \\ &= -\nabla_\theta L(z,\hat{\theta}) H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta L(z,\hat{\theta})^T, \end{aligned} \tag{15}$$

the only new element in these equations is $\nabla_x \nabla_\theta L(z, \hat{\theta})$, so we need to calculate that in order to solve these new equations. The old Influence Function on the loss had dimensions n by n so that the matrix denotes the influence of every observation on every other observations loss when that observation has a larger weight in the sample set. When perturbing a data point the matrix should also display the influence separately for all x values, so that we can see which x values should increase and which should decrease if we want to increase the loss. The new matrix should thus have 3 dimensions n by n by k. The gradient in (6) has dimensions n by k, which we take the derivative from with respect to k different x variables. $\nabla_x \nabla_\theta L(z, \hat{\theta})$ should therefore be a n by k by k matrix, also called a tensor. The transpose is not defined for a three dimensional matrix but in this case each slice (increment in the third dimension) should be transposed separately.

$$\begin{aligned} \nabla_{x_1} \nabla_\theta L(z,\hat{\theta}) &= \nabla_{x_1} - 2S \circ (1-S) \circ (Y-S) \circ X \\ &= -2\nabla_{x_1} \left\{ Y \circ S + (-Y-1) \circ S^{\circ 2} + S^{\circ 3} \right\} \circ X \\ &= -2 \left\{ Y \circ S + (-Y-1) \circ S^{\circ 2} + S^{\circ 3} \right\} \circ \nabla_{x_1} X \\ &+ \nabla_{x_1} \left[-2 \left\{ Y \circ S + (-Y-1) \circ S^{\circ 2} + S^{\circ 3} \right\} \right] \circ X \\ &= -2 \left\{ Y \circ S + (-Y-1) \circ S^{\circ 2} + S^{\circ 3} \right\} \circ J_1 - 2 \left\{ Y + 2(-Y-1) \circ S + 3S^{\circ 2} \right\} \circ X \hat{\theta}_1, \end{aligned} \tag{16}$$

only 2 elements depend on for which X is being differentiated. The matrix J_1 which is a n by k matrix with ones in the column 1 and zeros elsewhere, and the scalar θ_1. These elements are easily computed for other x values.

$$\nabla_x \nabla_\theta L(z,\hat{\theta}) = -2 \left\{ Y \circ S + (-Y-1) \circ S^{\circ 2} + S^{\circ 3} \right\} \circ J - 2 \left\{ Y + 2(-Y-1) \circ S + 3S^{\circ 2} \right\} \circ X \circ \hat{\theta}, \tag{17}$$

equation (17) creates a three dimensional matrix n by k by k. J is a three dimensional matrix with dimensions n by k by k with on slice number q ones in column number q and zeros elsewhere, and $\hat{\theta}$ a 1 by 1 by k scalar vector. With (15) the total Influence Function of the loss can be calculated. The matrix n by n by k displays by what amount the loss increases at a point if the variables of another point move from z to $z+1$ by an infinite small amount. For the influence when moving from z to $z+\delta$ the matrix should be multiplied with the scalar δ. By letting δ be positive when the loss is large and negative when the loss is far below zero the optimal local perturbations can be chosen

to attack the training set and reduce the model performance. If all observations increase with the same δ the column sum should be taken to observe the influence it has.

B Logarithmic Score plots

The following section displays Logarithmic plots when anomalies consist of multiplying all X values by -1.

Figure 24: Logarithmic Score, no hidden layer

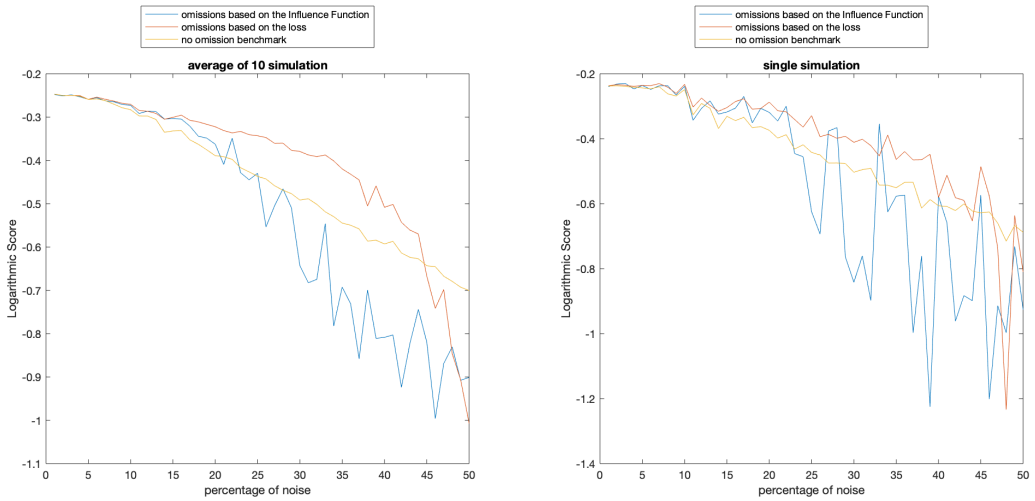


Figure 25: Logarithmic Score, single hidden layer

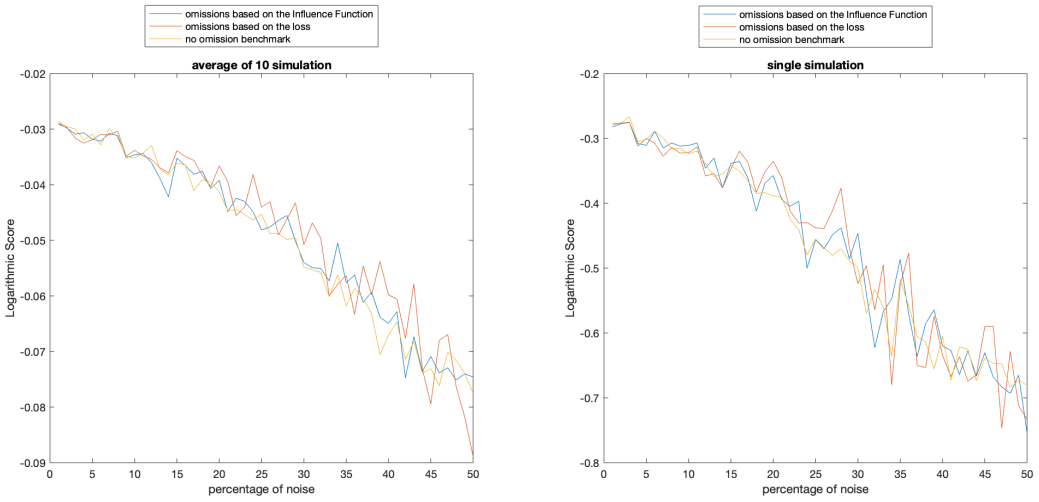


Figure 26: Logarithmic Score, house price data