PLACEHOLDER, FINAL VERSION WILL BE UPLOADED ON MONDAY

M.S Ingstad & F.L Nilsen

(Dated: November 8, 2019)

This is a placeholder.

I. INTRODUCTION

Intro $\varpi\varrho\varsigma$

II. THEORY

Logistic regression

To preform a classification of a variable, based on a set of predictors, one can use logistic regression.......

This regression method is based on the sigmoid function, given by

$$S(x) = \frac{1}{1 - e^{-x}} = \frac{e^x}{1 + e^x}. \tag{1}$$

For classification purposes, this function have the useful property that it returns values between 0 and 1. So by using the sigmoid function we can get the likelihood for an event. Here we will look at the case with two classes, and p predictors, represented as a vector $\hat{x} \in \mathbb{R}^p$. We will also introduce $p+1$ parameters represented as $\hat{\beta} \in \mathbb{R}^{p+1}$ where we also have a bias β_0. From this we define the variable

$$t = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p. \tag{2}$$

From equation (1) and (2) we then define the probability of $y = 1$ given $\hat{x}$ and $\hat{\beta}$ as

$$p(y = 1|\hat{x}, \hat{\beta}) = S(t). \tag{3}$$

Since we only have two classes, the probability for $y = 0$ becomes $1 - p(y = 1|\hat{x}, \hat{\beta})$. Given a dataset $\mathcal{D} = \{\hat{x}_i, y_i\}$ with n data points the total likelihood for all outcomes will be given by the product of all probabilities or

$$p(\mathcal{D}|\beta) = \prod_{i=1}^{n} \left[p(y = 1|\hat{x_i}, \hat{\beta})\right]^{y_i} \left[1 - p(y = 1|\hat{x_i}, \hat{\beta})\right]^{1-y_i}, \tag{4}$$

where $p(y = 1|\hat{x_i}, \hat{\beta})$ is given by equation (3). We then define our cost function as the logarithm of this probability, multiplied with -1. By rewriting it we get a costfunction of

$$C(\hat{\beta}) = -\sum_{i=1}^{n} \left(y_i t_i - \log\left(1 + e^{t_i}\right)\right). \tag{5}$$

Here t_i is the value of t one gets from equation (2) by using $\hat{x}_i$. If we now have n data points, we can define the matrix $\mathbf{X} \in \mathbb{R}^{n\times p+1}$ (where the first column is ones). We then see that we get a vector $\hat{t} \in \mathbb{R}^n$ with all values for t from equation (2). In matrix form we find this vector as $\hat{t} = \mathbf{X}\hat{\beta}$. We see that the first term in the sum in the costfunction (equation (5)) becomes $\hat{y}^T\mathbf{X}\hat{\beta}$, so the derivative with regards to $\hat{\beta}$ becomes $\mathbf{X}^T\hat{y}$ (transposed to keed the dimensions). For the second term, we see that for a single data point, the derivative becomes $\hat{x}_i \frac{e^{t_i}}{1+e^{t_i}} = \hat{x_i}S(t_i)$. In matrix form the derivative of this term is then $\mathbf{X}^T S(\hat{t})$ So the final derivative the becomes

$$\frac{\partial C(\hat{\beta})}{\partial \hat{\beta}} = -\left(\mathbf{X}^T\hat{y} - \mathbf{X}^T S(\hat{t})\right) = -\mathbf{X}^T\left(\hat{y} - S(\hat{t})\right). \tag{6}$$

Feed Forward Neural Networks

A

Stochastic Gradient descent (SGD)

When doing regression or classification with different methods, one often end up with a cost function to minimize. If there is no expression for the minimum of this cost function, one have to use numerical methods to find it. One such method is stochastic gradient descent (SGD). In SGD one first divides the dataset one uses to evaluate the cost functions into several subsets, called minibatches. One then use a form of gradient descent, but the gradient is calculated over a single minibatch. So if we say we have a minibatch B_k and if a data point $\hat{x}_i$ is in B_k we say that $i \in B_k$, we get the following expression for the approximation of the gradient

$$\nabla_\beta C(\beta) = \sum_{i\in B_k} \nabla_\beta C\left(\hat{x}_i, \beta\right). \tag{7}$$

This is then used in an algorithm similar to gradient descent, where one finds the next values for $\hat{\beta}$ by the formula

$$\hat{\beta}^{(n+1)} = \hat{\beta}^{(n)} - \sum_{i\in B_k} \nabla_{\hat{\beta}} C\left(\hat{x}_i, \hat{\beta}\right), \tag{8}$$

for a minibatch B_k. If one have N minibatches, updating $\hat{\beta}$ N times, using a random minibatch each time is called an epoch. In SGD one will then typically continue until a predetermined number of epoch is reached, or until the value for $\hat{\beta}$ changes by less then a set amount (over a number of epochs).

III. METHOD

Method When analysing the results, we have several measures to get an idea of how good the model is. The most simple is to use the model to predict a number of outcomes, and then calculate the proportion of correct predictions. This is called the accuracy of the model. This measure however, have some problems when the dataset have a large proportion of one type of observation, as you can get a high accuracy by always predicting that class. Therefor we will also look at the confusion matrix. The elements in the confusion matrix for a binary classifier are the number of true negatives (TN), false positives (FP), false negatives (FN) and true positives (TP). From these four quantities we can then derive other quantities like the

When actually calculating these measures, we will use k-fold cross validation. Here the dataset is split into K subsets, and then one trains the model on $K-1$ subsets, while the last subset is used as the test data for the model. One then do this K times, with each set acting as the test data once. As the final estimate for the accuracy measure of the model one use the mean of all results. When using k-fold validation to find the values for TN, FP, FN and TP we will not compute the mean of each repetition, as each subset may not be of the same size. Instead we will divide the values by the number of data points in the test set so that we get normalized values.

IV. RESULTS

A. Regression

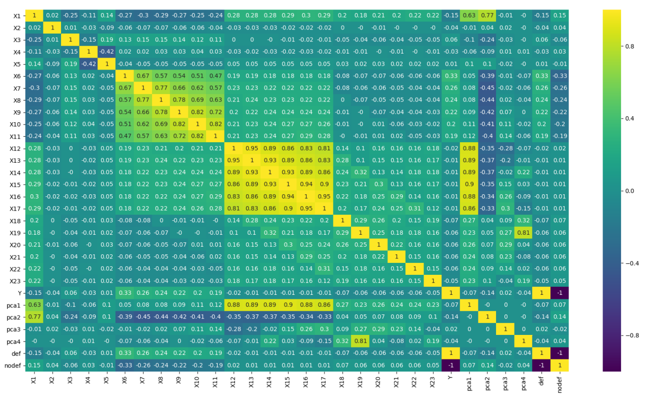
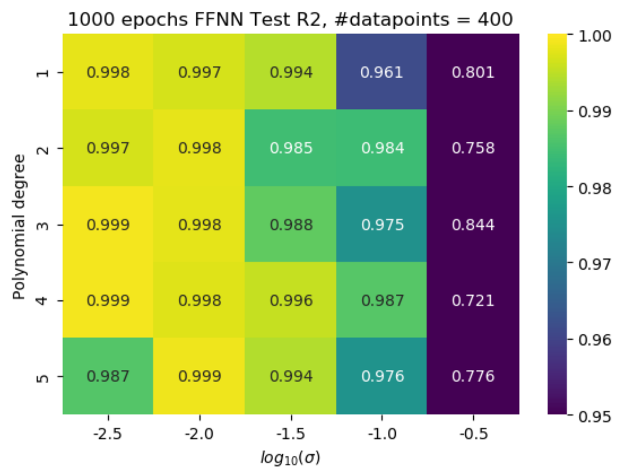
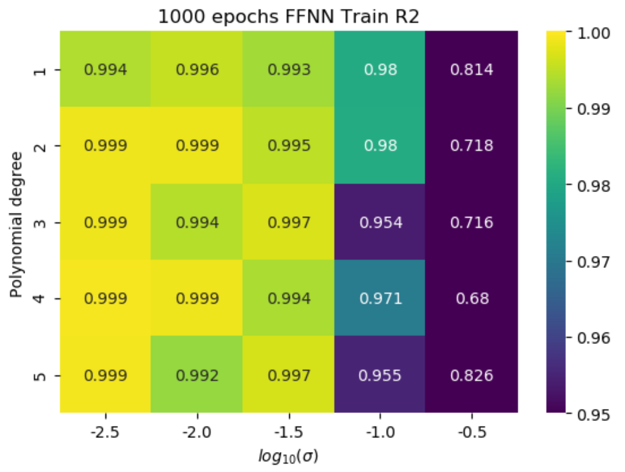


Figure 1. Placeholder correlation plot

Note that the results in Figure 2 are better than our results for 400 data points in project 1.



(a)Test data



(b)Training data

Figure 2. 400 data points, 60,30,15 hidden layers. eta 0.2, relu(0.005), 1000 epochs.

Table I. Search space for optimal parameter search. 400 data points of data used

order	η	hlayers	a	batches
1	0.05	[]	0.005	1
2	0.1	[30]	0.01	4
3	0.15	[30,15]	0.015	10
4	0.2	[60,30,15]	0.02	20
5		[32,16,8]		
6				
7				

B. Classification

When using k-fold cross validation on the entire dataset we got an accuracy of 0.772. We also got the values TN = 0.753, FP = 0.023, FN = 0.190 and TP = 0.033 (normalised values). From this we get that the ...

Table II. Results from the optimal parameter search, using the search space in Table I for different values of the noise $\hat{\sigma}$ with 400 data points. Finally the R^2 score evaluated on 300 epochs of training using the k-fold error with k=5 and k=4 is listed.

$\hat{\sigma}$	order	η	hlayers	a	batches	R^2
-2.5	6	0.2	[60,30,15]	0.01	20	0.995
-2.0	7	0.2	[32,16,8]	0.015	20	0.992
-1.5	5	0.2	[60,30,15]	0.02	20	0.987
-1.0	1	0.15	[60,30,15]	0.02	20	0.960
-0.5	3	0.2	[60,30,15]	0.02	1	0.833

V. DISCUSSION

Discussion

VI. CONCLUSION

Conclusion

VII. ADDITIONAL RESULTS

Can be found in the github repository https://github.com/Magnus-SI/FYS-STK3155