GENERIC MASTER THESIS TITLE

by

Geir Tore Ulvik

THESIS

for the degree of

MASTER OF SCIENCE

Faculty of Mathematics and Natural Sciences
University of Oslo

November 2020

Abstract

This is the abstract text.

To someone

This is a dedication to my cats.

Acknowledgements

I acknowledge my acknowledgements.

Contents

Chapter 1

Introduction

Start your chapter by writing something smart. Then go get coffee.

Chapter 2

Theory

2.1 Supervised Learning

Prediction and classification. From standard regression methods for predicting a continuous variable, to classifying with logistic regression and neural networks.

2.1.1 Linear Regression

Ordinary Least Squares

Suppose you have a data set consisting of n data points. Each point is associated with a scalar target y_i, and a vector $\hat{x}$ containing values for p input features. Assuming the target variable y_i is linear in the inputs, it can be written as a linear function of the features, given by

$$y_i = \beta_0 x_{i,0} + \beta_1 x_{i,1} + ... + \beta_{p-1} x_{i,p-1} + \varepsilon_i, \tag{2.1}$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_{p-1})^T$ is a vector of length p containing unknown values, and ε are the errors in our estimate. This gives us a system of linear equation, which can be written in matrix form as

$$\hat{y} = \mathbf{X}\hat{\beta} + \hat{\varepsilon}, \tag{2.2}$$

where

$$\mathbf{X} = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & ... & x_{0,p-1} \\ x_{1,0} & x_{1,1} & x_{1,2} & ... & x_{1,p-1} \\ x_{2,0} & x_{2,1} & x_{2,2} & ... & x_{2,p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n-1,0} & x_{n-1,1} & x_{n-1,2} & ... & x_{n-1,p-1} \end{bmatrix} \tag{2.3}$$

In order to find the best possible coefficients β we want a suitable quantity to optimize - a **cost function**, $\mathcal{C}$. An example of such a function is the average squared error, defined as

$$\mathcal{C}(\beta) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - f(x_i))^2, \tag{2.4}$$

which we want to minimize. This can be rewritten in matrix notation

$$\mathcal{C}(\beta) = (\hat{y} - \mathbf{X}\beta)^T (\hat{y} - \mathbf{X}\beta). \tag{2.5}$$

.

To minimize the cost function $\mathcal{C}(\hat{\beta})$, we take the derivative with respect to the coefficients β

$$\frac{\partial \mathcal{C}(\hat{\beta})}{\partial \beta_j} = \frac{\partial}{\partial \beta_j} \frac{1}{n} \sum_{i=0}^{n-1} (y_i - f(x_i))^2 \tag{2.6}$$

$$= \frac{2}{n} \sum_{i=0}^{n-1} x_{ij}(y_i - f(x_i)) \tag{2.7}$$

This can be rewritten as

$$\mathbf{X}^T \hat{y} = \mathbf{X}^T \mathbf{X} \hat{\beta}. \tag{2.8}$$

We then assume that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible to get the solution ([1]).

Ridge Regression

OLS will not give a solution if the design matrix $\hat{X}$ is singular or near singular, as it depends on $\hat{X}^T\hat{X}$ being invertible. (cite?)

Ridge regression is known as a shrinkage method. It shrinks the regression coefficients $\hat{\beta}$ by imposing a penalty on their size (cite?). We are in other words adding a diagonal component to the matrix to invert. From OLS we therefore make the following change

$$\hat{X}^T\hat{X} \rightarrow \hat{X}^T\hat{X} + \lambda\hat{I} \tag{2.9}$$

where I is the identity matrix. (cite?). This gives

$$RSS(\lambda) = (\hat{y} - \hat{X}\hat{\beta})^T(\hat{y} - \hat{X}\hat{\beta}) - \lambda\hat{\beta}^T\hat{\beta} \tag{2.10}$$

and

$$\hat{\beta}^{ridge} = (\hat{X}^T\hat{X} + \lambda\hat{I})^{-1}\hat{X}^T\hat{y} \tag{2.11}$$

$\lambda > 0$ is a complexity parameter that controls the amount of shrinkage. The larger the value of λ the greater amount of shrinkage. [1]

This can also be written in a non-vector format which makes explicit the size constraint on the parameters. It also makes it easier to see the difference from Lasso regression:

$$\hat{\beta}^{ridge} = \text{argmin}_{beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \tag{2.12}$$

subject to

$$\sum_{j=1}^{p} \beta_j^2 \leq t \tag{2.13}$$

There is a one to one correspondence between λ and t. [1]

2.1.2 Classification

Logistic Regression

Differently to linear regression, classification problems are concerned with outcomes taking the form of discrete variables. For a specific physical problem, we'd like to identify its state, say whether it is an ordered of disordered system. (cite?) One of the most basic examples of a classifier algorithm is logistic regression.

2.1.3 Cost Functions

In order for a logistic regressor to improve it needs a way to track how it's performing. This is the purpose of a cost function. Essentially, the cost function says something about how wrong the model is in classifying the input. The objective in machine learning, and logistic regression, is then to minimize this error.

The cost function used in this project is called the **cross-entropy**, or the 'negative log likelihood', and takes the form

$$\mathcal{C}(\hat{\beta}) = -\sum_{i=1}^{n} \left(y_i(\beta_0 + \beta_1 x_i) - \log\left(1 + \exp\left(\beta_0 + \beta_1 x_i\right)\right)\right) \tag{2.14}$$

2.1.4 Gradient Descent

Minimizing the cost function is done using Gradient Descent. The jist of it is that in order to optimize the weights or coefficients, and biases to minimize the cost function, one can change their values to

$$\frac{\partial \mathcal{C}(\hat{\beta})}{\partial \hat{\beta}} = -\hat{X}^T\left(\hat{y} - \hat{p}\right) \tag{2.15}$$

Stochastic Gradient Decent

The stochastic gradient decent method address some of the shortcomings of the normal gradient decent method. The gradient decent method is for instance sensitive to the choise of learning rate (cite?).

The underlying idea of stochastic gradient decent comes form observing that the cost function we want to minimize, almost always can be written as a sum over n data points. (cite?). Which gives

$$C(\beta) = \sum_{i=1}^{n} c_i(\mathbf{x}_i\beta) \tag{2.16}$$

(cite?)

This means that we also can find the gradient as a sum over i gradients as follows:

$$\Delta_\beta C(\beta) = \sum_i^n \Delta_\beta c_i(\mathbf{x}_i\beta) \tag{2.17}$$

(cite?)

Randomness is included by only taking the gradient on a subset of data.

2.2 Neural Networks

2.2.1 Perceptron

Multilayer Perceptron

2.2.2 Backpropagation

2.3 Assessing the Performance of Models

If classification accuracy is not enough to gauge whether a model is performing well, or well in the desired way, alternative way to measure performance must be explored. For cases of imbalanced data there are a few widely used methods that reveal information about the model that the simple accuracy metric can't.

2.3.1 Imbalanced Data in Classification

The physical world is rarely balanced.

A common challenge in classification is imbalanced data, in which a large amount of the labeled data belongs to just one or a few of the classes. For binary classification, if 90% of the data belongs to one of the classes, then the classifier is likely to end up placing every single input in that class, as it will bring its accuracy to 90%. Techincally, this accuracy is correct, but it's not very useful since the decision isn't at all affected by the features of the input. Accuracy alone isn't a good enough measure of performance to reveal this.

2.3.2 Confusion Matrix

A confusion matrix is an n by n matrix containing correct classifications on the diagonal, and false positives and negatives in the off-diagonal elements. An example of such a matrix could be the following table: In the table above

	True Cat	True Dog	True Rabbit
Predicted Cat	**5**	2	0
Predicted Dog	3	**3**	2
Predicted Rabbit	0	1	**11**

Table 2.1: Confusion matrix for an example classification where the classes are Cat, Dog and Rabbit. Correct classifications in bold.

(2.1), the diagonal elements $i = j$ are the correct classifications, while the other elements correspond to cases where the model predicted class i but should've predicted class j. The confusion matrix thus gives information about false positives and false negatives, in addition to classification accuracy. This is very useful in cases where for example false positives can be readily ignored or filtere later, but false negatives may have severe consequences. An example of this could be detection of cancer, in which a false positive can be ruled out from further testing, while a false negative may lead to a patient being sent home when actually needing help. For a more in-depth look at confusion matrices see [2].

2.3.3 Receiver Operating Characteristic

The Receiver Operating Characteristic (ROC) is a widely used measure of a classifiers performance . The performance is measured as the effect of the true positive rate (TPR) and the false positive rate (FPR) as a function of thresholding the positive class. To evaluate the ROC curve for a model, traditionally the Area Under the Curve (AUC) is used, which ranges from 0 (an ideal "opposite" classifier) to 1.0 (an ideal classifier) with 0.5 indicating a random choice classifier, For a thorough explanation of ROC curves and the underlying concepts, see [2].

2.4 Nuclear Science

2.4.1 Shell Structure

- Comprehensive and predictive model of atomic nuclei
  - Evolving structure of atomic nuclei as a function of protons and neutrons from first principles
- Understanding the origin of the elements
  - Explosive nucleosynthesis

- Use of atomic nuclei to test fundamental symmetries
- Search for new applications of isotopes and solutions to societal problems

Nomenclature

A nucleus Y has Z protons and N neutrons with a mass of $A = Z + N$. This is written as ${}^{A}_{Z}Y_{N}$. For a given nucleus there may be several

- Isotopes - nuclei with the same number of protons, but varying number of neutrons
  - ^{66}Ni, ^{67}Ni, ^{68}Ni, ^{69}Ni, ^{70}Ni
- Isotones - nuclei with the same number of neutrons, but varying number of protons
  - ^{70}Zn, ^{69}Cu, ^{68}Ni, ^{67}Co, ^{66}Fe
- Isobars - nuclei with the same number of nucleons A
  - ^{68}Fe, ^{68}Co, ^{68}Ni, ^{68}Cu, ^{68}Zn

Chapter 3

Results

3.1 Classification

Our primary objective is separating experimental data into two possible categories. Supervised learning typically leverages large amounts of labeled data to learn representations of the classes present in the inputs. A challenge in Physics is that experiments generate enormous amounts of data, but it is not labeled. Hand-labeling data is time-consuming and cost-ineffective. A possible solution to this challenge is to train models on simulated data. The idea being that simulated and experimental data are similar enough that the learned patterns from simulated data are, to some degree, transferrable to experimental data. The simulated data contains two classes of events - a single electron and a double electron.

3.1.1 Simulated data

In table 3.1 the performance of each model trained is reported as the f1-score. The model architecture for each model is described in (ref appendix). ... including a state of the art pretrained network ([]VGG HERE) applied to the data using the approach described in **??**. To give a broader picture of the performance we also include the confusion matrix values in table 3.2. Similar f1-scores may still show differences in either of the confusion matrix values. 3.2.

3.1.2 Regression

- ? - Linear approach?
- ? - Dense network
- ? - CNN

3.1.3 Experimental data

Table 3.1: Mean F1-scores for classification of simulated data using multiple models. Error estimates are the standard deviation in results from k-fold cross-validation with $K = 5$ folds.

Logistic	Dense	Convolutional	Pretrained VGG16
0.734 $\pm 7.727 \times 10^{-3}$	0.907 $\pm 1.329 \times 10^{-2}$	0.959 $\pm 6.286 \times 10^{-3}$	0.894 $\pm 1.591 \times 10^{-2}$

Table 3.2: Mean confusion matrix values for classification of simulated data using multiple models. Error estimates are the standard deviation in results from k-fold cross-validation with $K = 5$ folds.

	TN	FP	FN	TP
Logistic	1.28×10^5 $\pm 1.546 \times 10^4$	6.16×10^4 $\pm 1.546 \times 10^4$	4.39×10^4 $\pm 1.118 \times 10^4$	1.46×10^5 $\pm 1.118 \times 10^4$
Dense	1.72×10^5 $\pm 1.125 \times 10^4$	1.85×10^4 $\pm 1.125 \times 10^4$	1.73×10^4 $\pm 5.205 \times 10^3$	1.73×10^5 $\pm 5.205 \times 10^3$
Convolutional	1.89×10^5 $\pm 7.287 \times 10^2$	1.39×10^3 $\pm 7.290 \times 10^2$	1.37×10^4 $\pm 2.692 \times 10^3$	1.76×10^5 $\pm 2.692 \times 10^3$
Pretrained VGG16	1.79×10^5 $\pm 9.079 \times 10^3$	1.15×10^4 $\pm 9.080 \times 10^3$	2.71×10^4 $\pm 9.409 \times 10^3$	1.63×10^5 $\pm 9.408 \times 10^3$

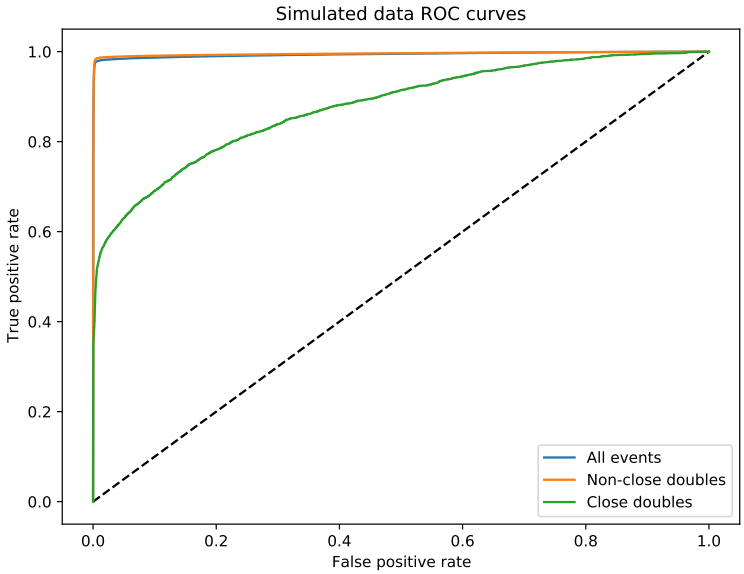

Figure 3.1: generic text

3.2 Regression

3.2.1 Position

3.2.2 Energy

3.2.3 Position

3.2.4 Energy

Chapter 4

Discussion

Chapter 5

Conclusion

Bibliography

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to Statistical Learning*, volume 7. 2000. ISBN 978-1-4614-7137-0. doi: 10.1007/978-1-4614-7138-7.

[2] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27 (8):861–874, 2006. ISSN 01678655. doi: 10.1016/j.patrec.2005.10.010.