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 Linear Regression

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 equations, which can be written in matrix form as

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

where

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

The unknown values $\boldsymbol{\beta}$ are commonly referred to as *weights*, denoted $\boldsymbol{w}$, in machine learning literature. We will adopt this notation going forward. To find the best possible weights $\boldsymbol{w}$ we want a suitable quantity to optimize - a **cost function**, $\mathcal{C}$ (also referred to as an **objective function**). An example of such a function is the squared error - or the Euclidian vector norm, defined as

$$L_2(\boldsymbol{x}) = ||\boldsymbol{x}||_2 = \left(\sum x_i^2\right)^{\frac{1}{2}}. \tag{2.4}$$

From this we define the cost function

$$\mathcal{C} = ||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2. \tag{2.5}$$

In machine learning, it is most common to cast the optimization as a minimization problem ("minimize the cost"). To find the minimum of the cost function as defined above, we need a differentiation. To simplify that process, we rewrite the cost function on matrix form

$$\begin{aligned} \mathcal{C} &= ||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2, \\ \mathcal{C} &= (\hat{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{w})^T(\hat{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{w}). \end{aligned}$$

To minimize we take the derivative with respect to the weights $\boldsymbol{w}$, and find the minima by setting the derivative equal to zero

$$\nabla_{\boldsymbol{w}} \mathcal{C} = \nabla_{\boldsymbol{w}} (\hat{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{w})^T (\hat{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{w}), \tag{2.6}$$

$$= -2\boldsymbol{X}^T \hat{\boldsymbol{y}} + 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w}, \tag{2.7}$$

$$\boldsymbol{0} = -2\boldsymbol{X}^T \hat{\boldsymbol{y}} + 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w}, \tag{2.8}$$

$$\boldsymbol{X}^T \hat{\boldsymbol{y}} = \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w}, \tag{2.9}$$

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \hat{\boldsymbol{y}}. \tag{2.10}$$

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

2.2 Over- and underfitting

In machine learning, when fitting a model to a data set the goal is nearly always to predict values or classify samples from regions of data the model has not seen. This is not a simple task, especially taking into consideration that data is rarely, if ever, noiseless. When extrapolating to unseen regions we must take steps to ensure the model complexity is appropriate - we want it to fit the signal, not the noise. First off - what do the terms "overfit" and "underfit" mean? An overfit model will typically perform well during the fitting procedure, but when presented with data outside the fitted region its performance decreases considerably. An underfit model lacks the expressive power to capture essential signal variations in the data.

In statistics, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". (cite oxford dictionary). Mehta et. al [2] demonstrates this concept very well through polynomial regression.

A step in this process is splitting the available data in two - training data and test data. We fit, or 'train' the model on the training data, and then assess the performance of the model on the test data. This allows us to combat the problem of model complexity. If the model is too simple, it may fail to express the global trends in the data - it is underfitting. More complex models may be able to capture the global trends, but run the risk of also capturing spurious correlations in the data, e.g noise - the model is overfitting.

2.3 Regularization

With the computing resources available today, increasing model complexity to deal with underfitting is usually a simple task. However, this computational freedom has led to overfitting being the common challenge to overcome.

2.4 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.

For a logistic regressor to improve it needs a way to track its performance. 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

$$\mathcal{C}(\boldsymbol{w}) = -\sum_{i=1}^{n} \left(y_i(w_0 + w_1 x_i) - \log\left(1 + \exp\left(w_0 + w_1 x_i\right)\right)\right) \tag{2.11}$$

2.5 Gradient Descent

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

$$\frac{\partial \mathcal{C}(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\boldsymbol{X}^T\left(\hat{y} - \hat{p}\right) \tag{2.12}$$

2.5.1 Stochastic Gradient Descent

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

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

$$C(\beta) = \sum_{i=1}^{n} c_i(\mathbf{x}_i\beta) \tag{2.13}$$

(cite?)

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

$$\Delta_\beta C(\beta) = \sum_{i}^{n} \Delta_\beta c_i(\mathbf{x}_i\beta) \tag{2.14}$$

(cite?)

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

2.6 Neural Networks

2.6.1 Perceptron

Multilayer Perceptron

2.6.2 Backpropagation

2.7 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.

2.7.1 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%. Technically, 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.

2.7.2 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 (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

	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.

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 [3].

2.7.3 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 [3].

2.8 Nuclear Science

2.8.1 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 ${}^{\mathrm{A}}_{\mathrm{Z}}\mathrm{Y}_{\mathrm{N}}$. 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

Method

3.1 Pretrained network

Chapter 4

Results

4.1 Classification

Our primary objective is to separate 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.

4.1.1 Simulated data

In table 4.1 the performance of each model trained is reported using the f1-score. The model architecture for each model is described in (ref appendix). As a benchmark, we are including a state of the art pretrained network ([]VGG HERE) applied to the data using the approach described in 3.1. To give a broader picture of the performance we also include the calculated confusion matrix values in table 4.2. Even if f1-scores are similar, there may still be differences in either of the confusion matrix values.

4.1.2 Experimental data

Table 4.1: 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 4.2: 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 $\pm 9.408 \times 10^3$

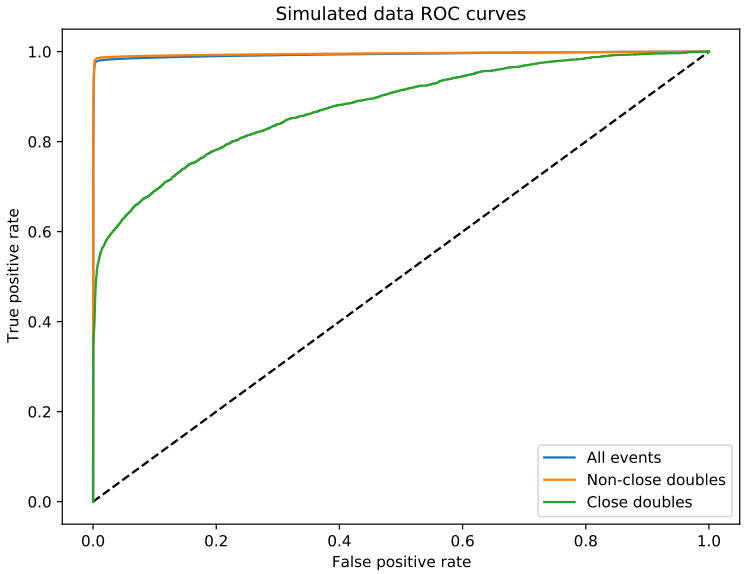

Figure 4.1: generic text

4.2 Regression

On data already classified, we attempt to predict the energy of the events and the position of origin for the electrons. Because there is a travel distance between the ejection site and the scintillator array, the positions aren't necessarily the locations of the highest-intensity pixels in the detector images.

4.2.1 Simulated data

Position - Single electron

Energy - Single electron

Position - Double electron

Energy - Double electron

Table 4.3: Mean R2-scores for regresson of positions of origin, on single events in simulated data, using multiple models. Error estimates are the standard deviation in results from k-fold cross-validation with $K = 5$ folds.

Linear	Dense	Convolutional	Pretrained VGG16
0.801 $\pm 3.664 \times 10^{-3}$	0.99 $\pm 8.185 \times 10^{-4}$	0.997 $\pm 2.401 \times 10^{-4}$	0.891 $\pm 6.864 \times 10^{-3}$

Table 4.4: Mean R2-scores for regresson of energy values, on single events in simulated data, using multiple models. Error estimates are the standard deviation in results from k-fold cross-validation with $K = 5$ folds.

Linear	Dense	Convolutional	Pretrained VGG16
0.926 $\pm 3.691 \times 10^{-2}$	0.931 $\pm 3.364 \times 10^{-2}$	0.936 $\pm 3.282 \times 10^{-2}$	0.935 $\pm 1.945 \times 10^{-2}$

Table 4.5: Mean R2-scores for regresson of positions of origin, on double events in simulated data, using multiple models. Error estimates are the standard deviation in results from k-fold cross-validation with $K = 5$ folds.

Linear	Dense	Convolutional	Pretrained VGG16
0.364 $\pm 5.796 \times 10^{-3}$	0.471 $\pm 1.809 \times 10^{-3}$	0.473 $\pm 2.394 \times 10^{-3}$	0.357 $\pm 1.079 \times 10^{-2}$

Table 4.6: Mean R2-scores for regresson of energy values, on double events in simulated data, using multiple models. Error estimates are the standard deviation in results from k-fold cross-validation with $K = 5$ folds.

Linear	Dense	Convolutional	Pretrained VGG16
0.415 $\pm 6.501 \times 10^{-2}$	0.416 $\pm 6.632 \times 10^{-2}$	0.428 $\pm 5.027 \times 10^{-2}$	0.404 $\pm 5.308 \times 10^{-2}$

Chapter 5

Discussion

Chapter 6

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] P. Mehta, M. Bukov, C. H. Wang, et al. A high-bias, low-variance introduction to Machine Learning for physicists. *Physics Reports*, 810:1–124, 2019. ISSN 03701573. doi: 10.1016/j.physrep.2019.03.001.

[3] 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.