Machine Learning (Honours)

Ke Yuan

Joao Almeida-Domingues[*]

University of Glasgow

September 30th, 2020 – December 15th, 2020

Contents

These lecture notes were collated by me from a mixture of sources , the two main sources being the lecture notes provided by the lecturer and the content presented in-lecture. All other referenced material (if used) can be found in the *Bibliography* and *References* sections.

The primary goal of these notes is to function as a succinct but comprehensive revision aid, hence if you came by them via a search engine , please note that they're not intended to be a reflection of the quality of the materials referenced or the content lectured.

Lastly, with regards to formatting, the pdf doc was typeset in LaTeX, using a modified version of Stefano Maggiolo's class

[*]2334590D@student.gla.ac.uk

1 Introduction

Recommender Systems , Information Retrieval , Supervised Learning , Unsupervised Learning , Regression , Classification , Clustering , Projection

Lecture 1
October 30th, 2020

ML starts with data, and the two key questions/problems are if we can find similarities across the different observations and if we can make predictions about them. Nowadays, ML can be though of as an ever-growing set of algorithms, which can be hard to understand and use and which may have to be tuned. Hence, it is important to understand them

AI move in the 70s evolves towards NN which can be seen as a function. During that time some studies shown that those functions did quite well at approximating certain kinds of behaviours, and so they tried to scale up to more complex tasks. They quickly realise that the real world was far too complex to be modelled in this manner, and so researchers focused on 2 key aspects : modelling the brain and predictions from data

1.1 *Course Topics*

- Learning from data
- Caveats
- Examining common algorithms
- How to implement algos and use popular libraries
- Essential maths and stats

1.2 *Prerequisites*

only the fundamentals, nothing too advanced

- Linear Algebra
- Probability and Random Variables
- Calculus
- Logs
- Python (scientific stack)

1.3 *Real World Applications*

Recommender Systems are one of the most popular applications in ML, specifically because it is hard to use a physical model to describe one's preferences, i.e you can't really write a traditional equation which describes preferences. However, given enough data, we can look for patterns

Biotech companies are others who make use of ML in order to diagnose patients and discover *biomarkers*. The patients can then be clustered into groups which improves the efficacy of clinical trials.

Within the SoCS the IR group use ML for such tasks as search, topic identification in noisy environments (e.g. news feeds), language models and video annotation. Within the HCI group ML is applied to improve speech and gesture recognition

1.4 *Supervised and Unsupervised Learning*

Supervised

1.1 definition. Regression a type of model that outputs continuous values

Regression, is essentially using a continuous function to learn from a set of examples. The algorithm learns by trying to fit the data to the regression line

1.2 example. predicting stock prices where x is time

1.3 definition. Classification model for distinguishing among two or more discrete classes

Classification, learning a rule that can separate objects of different types from another. In this case, the output variable y is no longer continuous, we want $f(x)$ to group its inputs and spit out a *discrete* choice (e.g [1,0]). So we can see it as providing a decision boundary between elements of different groups

1.4 example. Predicting skin cancer: we take images of moles filter them trough our $f(x)$ which will classify them as either malignant or benign

Unsupervised

1.5 definition. Clustering grouping together data points according to some criteria

We'll focus on the problem of clustering, i.e trying to find subgroups in the measures that we have. The key difference being that the data is not labelled

1.6 example. people with similar tastes, genes with similar functions

1.7 definition. Projection reduces the dimensionality of data (e.g PCA)

Projection, when we have data with a lot of dimensions which can be hard to visualise. We can use projection to reduce the number of related variables without losing the essential meaning/info.

1.8 example. Genetics (PCA), where the original data is a binary matrix where each column is a *snip* and each row represents a person

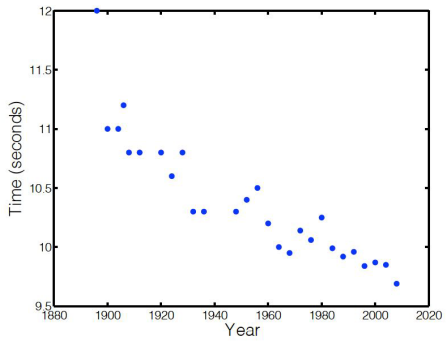
2 Supervised Learning - Linear Modelling

Lecture 2
September 30th, 20

Linear models
- Introduce the idea of building models.
- Talk about assumptions.
- Use a linear model.
- What constitutes a good model?
- Find the best linear model.
- Use it to predict the winning time in 2012.

In this lecture we'll use the dataset provided in section 1.1 of the textbook. Our goal will be to formalise this fitting and prediction process via a mathematical model which will allow us to scale this method to bigger problems

2.1 *Building Models*



Formalizing Intuitions

- Decided to draw a line through our data
- Chose a straight line
- Drew a good straight line
- Extended the line to 2012
- Read off the winning time for 2012

- Decided we need a model
- Chose a linear model
- Fitted a linear model
- Evaluated the model at 2012
- Used this as our prediction

Assumptions

When approaching the problem, we decided to assume the following:

1. there exists a meaningful relationship between year and winning time
2. the relationship is linear
3. the relationship will continue into the future

We are now going to investigate if we were right to do so

2.2 *Mathematically Defining the Model*

2.1 definition. Attributes/Features an input variable used in making predictions

2.2 example. Olympic year

2.3 definition. Targets/Labels in supervised learning, the answer portion of the example (e.g. spam/not spam)

2.4 example. Winning time

2.5 definition. Model the representation of what a machine learning system has learned from the training data, i.e some function which maps inputs to outputs

2.6 example. $f(x) = t$

2.7 definition. Training Data the subset of the dataset used to train a model

attribute-response pairs

2.8 definition. Linear Model a model that assigns one weight per feature to make predictions

$$t = f(x) = w_o + w_1 x = f(x; w_0, w_1)$$

Our linear model is essentially a line which can be *parametrized*. Usually a weight for bias is also included. In this example we only have our features represented by x and the feature's weights/parameters w which determine the properties of the line.

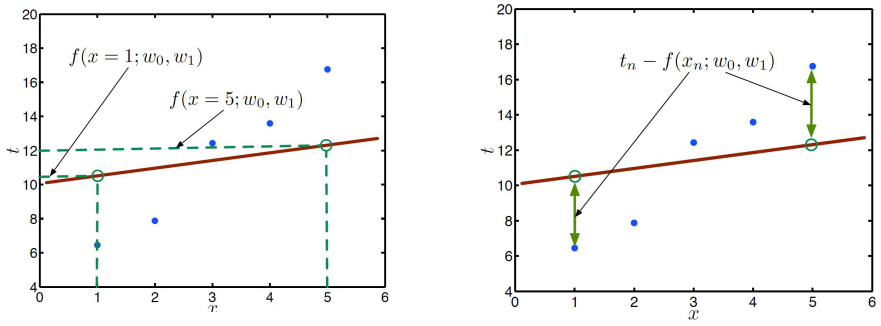
2.9 remark. note that this is simple high school geometry, you're tweaking the slope and y-intercept

Loss Function - Finding the right W

Now that we have data and a *family* of models we need to find the best (w_o, w_1) pair so that our model fits our data as best as possible. We are going to find the pair from the set of $(x_1, t_1), \ldots, (x_n, t_n)$ training examples

In order to do this we compare each pair given by our model against the expected value, and our goal here is to minimise the error. So we sum over all the pairs to find out the total cost of that particular model and we pick the parameters w which give us the smallest total cost

2.10 definition. Loss/Cost Function a function used for parameter estimation that maps an event like our (expected - estimated) function to a real number



2.11 definition. Squared Loss The average squared loss per example, which is calculated by diving the squared loss by the number of examples

$$\mathcal{L}_n = \left(t_n - f\left(x_n; w_0; w_1\right)\right)^2$$

2.12 remark. we need to square the mean to get the absolute values of each loss

By taking the mean, we can now have a loss function which tells us something about how the model did *on average*.

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - f\left(x_n; w_0, w_1\right)\right)^2$$

More generally for more complex models such as polynomial ones we have

$$t = w_0 + w_1 x + w_2 x^2 + w_3 x^2 + \ldots + w_K x^K = \sum_{k=0}^{K} w_k x^k$$

Vectorizing

In order to work with these models in a simplified manner we encapsulate the parameters and data into two vectors $\mathbf{w}, \mathbf{x}$ respectively. For our original example with two parameters now get

$$f\left(x_n; w_0, w_1\right) = \mathbf{w}'\mathbf{x}_n = w_0 + w_1 x_n$$

$$\mathcal{L}_n = \left(t_n - \mathbf{w}^\top \mathbf{x}_n\right)^2$$

We can vectorize our lost function further by encapsulating t also

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\top \mathbf{x}_1 \\ t_2 - \mathbf{w}^\top \mathbf{x}_2 \\ t_3 - \mathbf{w}^\top \mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\top \mathbf{x}_N \end{bmatrix} = \mathbf{t} - \mathbf{X}\mathbf{w}$$

We can now rewrite our mean loss in its final vector form

$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\top (\mathbf{t} - \mathbf{X}\mathbf{w})$$

Minimising the Loss

Now that we have a vector/matrix loss we can minimise it by taking the partial derivatives with respective to $\mathbf{w}$ and set it to 0

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{0} = \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\top (\mathbf{t} - \mathbf{X}\mathbf{w}) \right) = \frac{1}{N} \left(2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{t} \right)$$
$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{t}$$

Finally by premultiplying both sides we can solve for $\mathbf{w}$ giving us the optimum value of $\mathbf{w}$, i.e $\hat{\mathbf{w}}$

$$\widehat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{t}$$

2.13 remark. Useful identities for vector differentiation

$f(\mathbf{w})$	$\frac{\partial f}{\partial \mathbf{w}}$
$\mathbf{w}^\top \mathbf{x}$	$\mathbf{x}$
$\mathbf{x}^\top \mathbf{w}$	$\mathbf{x}$
$\mathbf{w}^\top \mathbf{w}$	$2\mathbf{w}$
$\mathbf{w}^\top \mathbf{C}\mathbf{w}$	$2\mathbf{C}\mathbf{w}$

More General Models

So far we've considered models which are linear in the parameter, and we've used many powers of x to get a polynomial function of any order with which we can better fit non-linear data. However, we are not restricted to polynomial functions, we can augment our data matrix X with any set of functions of x

$$\mathbf{X} = \begin{bmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_K(x_1) \\ h_1(x_2) & h_2(x_2) & \cdots & h_K(x_2) \\ \vdots & \vdots & \cdots & \vdots \\ h_1(x_N) & h_2(x_N) & \cdots & h_K(x_N) \end{bmatrix}$$

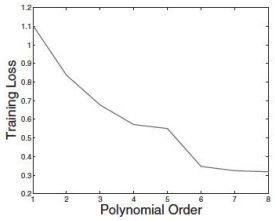
Take the set of functions to be

$$h_1(x) = 1$$
$$h_2(x) = x$$
$$h_3(x) = \sin\left(\frac{x-a}{b}\right)$$

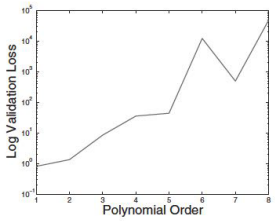
Our model can then be defined as

$$f(x;\mathbf{w}) = w_0 + w_1 x + w_2 \sin\left(\frac{x-a}{b}\right)$$

This model has 5 parameters w_0, w_1, w_2, a, b, we'll fix a, b because at this stage we do not know how to derive them analytically. We can derive the w's using the methods presented above, and we get an approximation better than that of the linear. Furthermore, the various model components are still clearly visible. We can see the constant term ($w0 = 36.610$), the downward linear trend ($w1 = -0.013$) and the non-linear sinusoidal term ($w3 = -0.133$) causing oscillations.



(a) Training loss for the Olympic men's 100 m data.



(b) Log validation loss for the Olympic men's 100 m data. When using the squared loss, this is also known as the squared predictive error and measures how close the predicted values are to the true values. Note that the log loss is plotted as the value increases so rapidly.

2.14 remark. see lecture notes for common basis functions

2.3 *Making Predictions*

Given a new vector of attributes x_{new}, and our original X matrix

$$\mathbf{X} = \begin{bmatrix} h_0(x_1) & h_1(x_1) & \dots & h_K(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ h_0(x_N) & h_1(x_N) & \dots & h_K(x_N) \end{bmatrix} \qquad \mathbf{x}_{\text{new}} = \begin{bmatrix} h_0(x_{\text{new}}) \\ \vdots \\ h_K(x_{\text{new}}) \end{bmatrix}$$

the prediction from the model, t_{new}, is computed as

$$t_{\text{new}} = \widehat{\mathbf{w}}^\top \mathbf{x}_{\text{new}}$$

We can then just compute the loss and compare our accurate different models are. However, care must be taken not to *overfit* our model to the data, i.e if our approximation too closely resembles the training data then predictions on unseen data will be poor because of noise

2.15 remark. There is a trade-off between generalisation (predictive ability) and over-fitting (decreasing the loss)

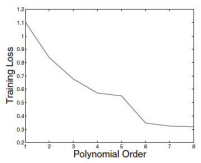
2.16 definition. Overfitting Creating a model that matches the training data so closely that the model fails to make correct predictions on new data

2.4 *Validation*

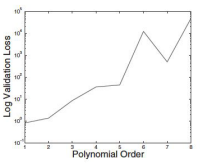
2.17 definition. Validation Set A subset of the dataset disjoint from the training set

In order to prevent over-fitting we can validate our model using a *validation set*. We can create a validation set by simply removing a few examples from our original dataset prior to training. Say we have N examples at the start, we can train our model using $N - C$ examples and validate our data using C. We can then choose the model which makes the best predictions on C

Note below how (as expected) the loss reduces as model complexity increases due to a better approximation to the data, but also how the loss rapidly increases when computed using the validation set. This indicates that even though the linear model performs worse on the testing data, it generalises better, hence it performs better on unseen data.



(a) Training loss for the Olympic men's 100 m data.



(b) Log validation loss for the Olympic men's 100 m data. When using the squared loss, this is also known as the squared predictive error and measures how close the predicted values are to the true values. Note that the log loss is plotted as the value increases so rapidly.

Another worry one should have is how to split the data. There is no one right answer, it will sometimes be clear from the model, often one can just split it randomly or go a step further and do it several times with different and average the results. The most comprehensive way is to perform *cross-validation*.

2.18 definition. Cross-validation testing the model against one or more non-overlapping data subsets withheld from the training set

When using *C-fold* cross-validation the training set is split into C *non-overlapping* folds of equal size, we then average the loss over all C folds.

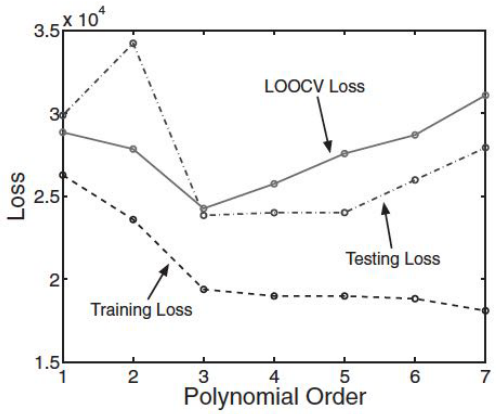
2.19 remark. If $C = N$ it's called *"Leave-one-out" CV*. Each example is held out in turn and used to test a model trained on the other $N - 1$

2.20 remark. CV is computationally expensive and the cost-benefit is too high for large datasets

Model Selection

If we were to perform LOOCV in the same dataset we would see that a 3rd order polynomial performs best. Using different methods can confuse the results. How can we know which one is right?

We can overcome this problem by generating a synthetic dataset. We then have some target pairs from a noisy third-order polynomial function and can use them to learn polynomial functions of increasing order. We can then use generate test targets using the original 3rd order polynomial to compute the true expected loss



2.5 *Regularisation*

References

Machine Learning Glossary Google Developers **google**

Machine Learning Glossary Google Developers. URL: https://developers.google.com/machine-learning/glossary.

Ng: Machine Learning **ng˙2012**

Andrew Ng. *Machine Learning*. 2012. URL: https://www.coursera.org/learn/machine-learning.

Rogers et al.: A first course in machine learning **rogers˙girolami˙2017**

Simon Rogers and Mark Girolami. *A first course in machine learning*. CRC Press, Taylor ; Francis Group, a Chapman ; Hall Book, 2017.