University of
St Andrews

CS5014 Machine Learning

Lecture Notes

May 4, 2018

Lecturer:
David Harris-Birtill
Kasim Terzić

Submitted By:
140011146

1 Introduction

In machine learning, we wish to *learn* the mapping between input and output without having to program it explicitly.

> Maching learning is the field of study that gives computers the ability to learn without being explicitly programmed. - Arther Samuel.
>
> A computer program is said to learn from exerience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. - Tom Mitchell.

In general, most machine learning boils down to a simple prediction equation.

$$\hat{Y} = f(X, \theta) \tag{1}$$

$$Y = f(X, \theta) + \varepsilon \tag{2}$$

where f is a mathematical model for predicting Y from X. X is a list of input features that are used for prediction. They could be independent variables or some kind of predictor. $\hat{Y}$ is the output of the predicting model which are the predicted values. θ are the model parameters that the model tries to learn and optimise for. Finally, ε is the error, which represents the difference between predicted values and desired values.

1.1 Prediction error

In order to measure the difference between the predicted values and actual values, the error for each data point must be measured or calculated in some way. To do this a **loss function** (cost function) is used to evaluate the quality of the model. The goal is to then try to minimise the loss function by altering the θ parameters to allow the model to make better predictions.

An example of a common loss function is the squared error loss.

$$L(\theta) = (Y - f(X, \theta))^2 \tag{3}$$

The loss function serves as a way to measure the quality of prediction for the set of parameters.

1.2 Machine learning process

1. Gather data
2. Prepare data (data cleaning and preprocessing)
3. Pick an algorithm (or a few different ones)
4. Pick a set of parameters
5. Evaluate performance of model
6. If the performance is not good, re-evaluate by altering parameters
7. Apply model to new/unseen data

This is generally implemented as an optimisation process, where the loss function is minimised on each iteration by changing the θ parameters. This strategy picks the optimal set of parameters. An example of such an optmisation process is gradient descent.

1.3 Supervised machine learning

The idea of supervised machine learning it to be able to predict an outcome based on given input data with tagged correct outputs. This could involve predicting a value or classifying an output class type. This requires a **labelled dataset** so it is known what the output is. There are two main types of supervised learning:

- Regression - The data predicted for regression is usually continuous and if it is decrete, it has to be rounded to the closest discrete value. Regression predicts the *value* of the data.
- Classification - The probability it is of some class. Classification predicts whether the data belongs to a certain class.

Classification uses a **decision boundary**, which everything on one side of the boundary is classified as one class and another on the other side of the boundary. It is possible to have hyper-dimensional decision boundaries for when there are more than 3 features which cannot be easily visualised.

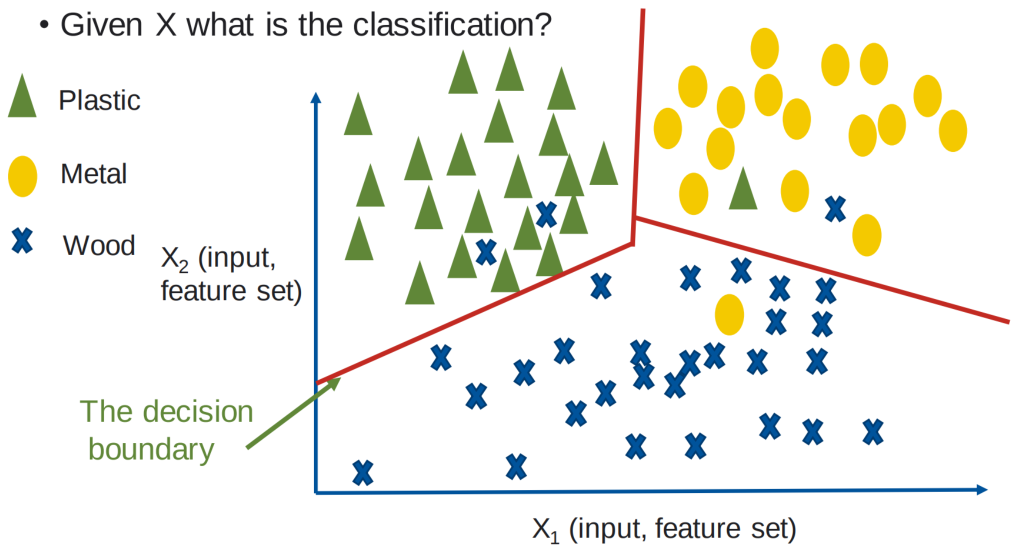

• Given X what is the classification?
Plastic
Metal
Wood
X_2 (input, feature set)
The decision boundary
X_1 (input, feature set)

Figure 1: Classification of 3 classes using 2 input features. Having more than 3 input features (3D plot) is difficult to visualise.

One major issue with many machine learning algorithms: **overfitting**. Often the models trained become optimised too well towards the training data, leading to very low errors, but an inability to generalise to unseen data.

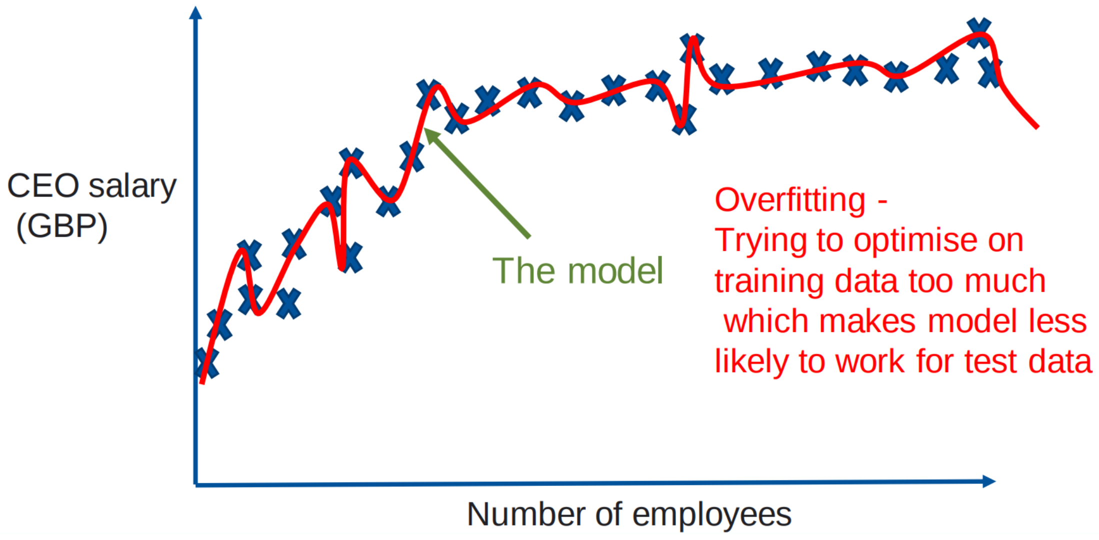


Figure 2: Example of overfitting in linear regression.

1.4 Unsupervised machine learning

In unsupervised learning, the dataset is **unlabelled**, and the output for predicted values is unknown. To learn, clusters are automatically created to separate the data into distinct groups. This is useful for discovering information where we don't know what we are looking for or to discover things we don't know about.

2 Linear regression

Linear regression models are simple but can provide a good and adequate descriptions of how the inputs affect the output. As linear regression only deals with linear models, it follows from the linear line equation $y = mx + c$ where the gradient m and intercept c are the θ parameters that the model tries to learn. This can be rewritten as:

$$f(X, \theta) = \theta_0 + \theta_1 X_1 \tag{4}$$

where θ_0 represents the c intercept and θ_1 represents the m gradient.

Furthermore, a loss function has to be defined to evaluate the quality of the fit without manual inspection. The goal of training is to reduce the error of the loss function. This is done by trying lots of different θ values and computing the error each time.

2.1 Gradient descent

Gradient descent is a method for parameter optimisation which automatically gets to the best model by minimising the lost function and automatically altering the θ parameters.

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} L(\theta_0, \theta_1) \tag{5}$$

where j = 0 and j = 1 update is done simultaneously. α here is the learning rate, which acts as a modifier to how much should be updated on each iteration. $\frac{\delta}{\delta\theta_j} L(\theta_0, \theta_1)$ is the derivative. The derivative of the loss function is used to see if we should increase or decrease the value of θ.

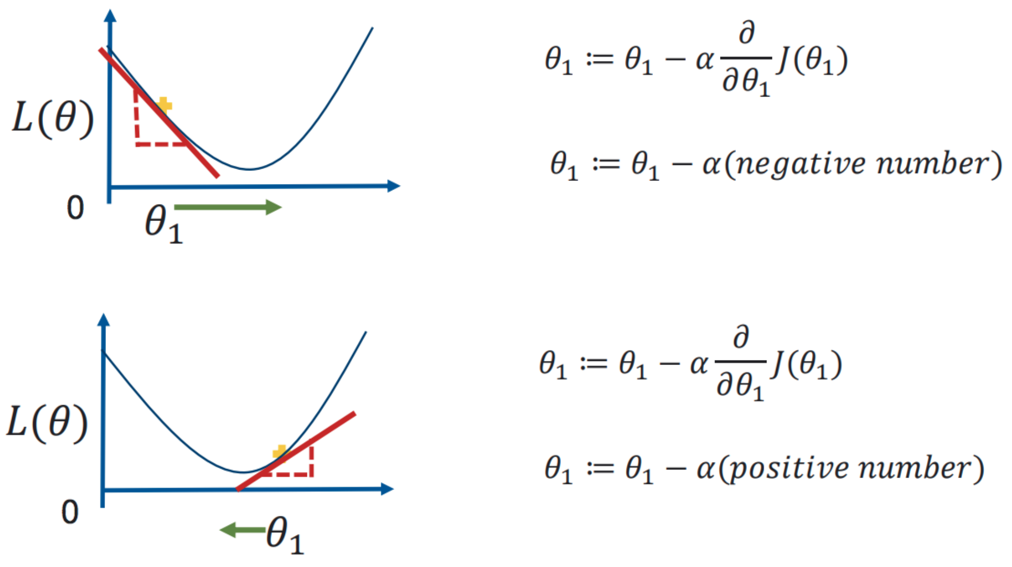

$L(\theta)$
0
θ_1
$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$
$\theta_1 := \theta_1 - \alpha(negative\ number)$
$L(\theta)$
0
θ_1
$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$
$\theta_1 := \theta_1 - \alpha(positive\ number)$

Figure 3: Gradient descent updating based on positive or negative gradient. For example, if the gradient is negative, we want to increase the θ values and recalculate the gradient.

There is an issue with gradient descent where it can get stuck in an local minimumm rather than the global minimum. There are a few ways of dealing with this:

- Try different initial values to start at different locations in the search space
- Add momentum to roll over local minima

A good way to find out what is happening with each training iteration is to plot the cost function at each iteration, then it can be seen if the cost is being minimised. The reduction of the cost function is controlled by the **learning rate** α. If α is too small, the gradient descent will be slow, requiring a large number of training iterations to minimise the cost and converge at the minimum. However, if the learning rate is too high, then it can overshoot the minimum and never converge. This effect can be seen by plotting the cost function and seeing if the graph converges to 0 or diverges.

2.2 Multivariate linear regression

Linear models can have many features and by extending to n features with n θ parameters. This extends the linear equation to be:

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...\theta_n x_n \tag{6}$$

The input features X and parameters θ can be more easily represented with matrix form:

$$X = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \tag{7}$$

Using matrix operations, all θ parameters can be updated simultaneously on every step rather than separately.

2.3 Feature scaling

When there are many input features, they can come from different scales. For example, one feature may have the range 0-1 while another has values ranging from 100-1000. Because of these different scales, the larger values may dominate the model even if both features are equally important. To deal with this issues, all features can be scaled to the same range through normalisation.

$$x_i = \frac{x_i - \mu_i}{\text{range}(x_i)} \tag{8}$$

One method of feature scaling is mean normalisation, where each input feature x_i is replaced with $x_i - \mu_i$ to make features with a mean of approximately zero.

2.4 Normal equation

$$\theta = (X^T X)^{-1} X^T y \tag{9}$$

where X^T is the transpose of X and $(X^T X)^{-1}$ is the inverse of $X^T X$. Feature scaling is not needed for the normal equation because each θ parameter is proportional to the input feature it is calculated from. X and y here are matrices of the inputs and outputs. With the normal equation, the θ parameters can be calculated analytically rather than require the many training iterations. It is sometimes preferable to use the normal equation over optimisation like gradient descent as there is not need for computation iterations and choice of learning rate. However, the normal equation is slow to calculate if the number of input features is very large, which gradient descent deals with well.

Normal equation

Pros:
- No need for iteration
- Don't need to choose α

Cons:
- Slow if number of input features is very large

Gradient descent

Cons:
- Needs many iterations
- Need to choose learning rate α

Pros:
- Works well when the number of features is large

3 Logistic regression

Logistic regression is a machine learning model to classify input data into output classes. A typical example of using logistic regression is to classify cancer or not cancer. We cannot use linear regression for classification tasks as it does not fit the data well and therefore is not well suited. In classification, a **sigmoid function** is used as it gives values between 0 and 1, like a switch between binary values. Coincidentally, sigmoid functions are also used as activation functions in neural networks.

The sigmoid function is defined as follows:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{10}$$

where $z = \theta^T x$. With classification, the output is typically a probability that of a certain class $0 \leq h(x) \leq 1$. The function is therefore:

$$h(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{11}$$

where $h(x)$ is the estimated probability that $y = 1$ on input x. In an example of multiple input features, we get get multiple θ values. For example, given blood and urine test values, does the patient have cancer:

$$h(x) = g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = \frac{1}{1 + e^{-\theta^T x}} \tag{12}$$

where x_1 is the blood test values and x_2 is the urine test values. $h(x)$ is the probability that the patient has cancer.

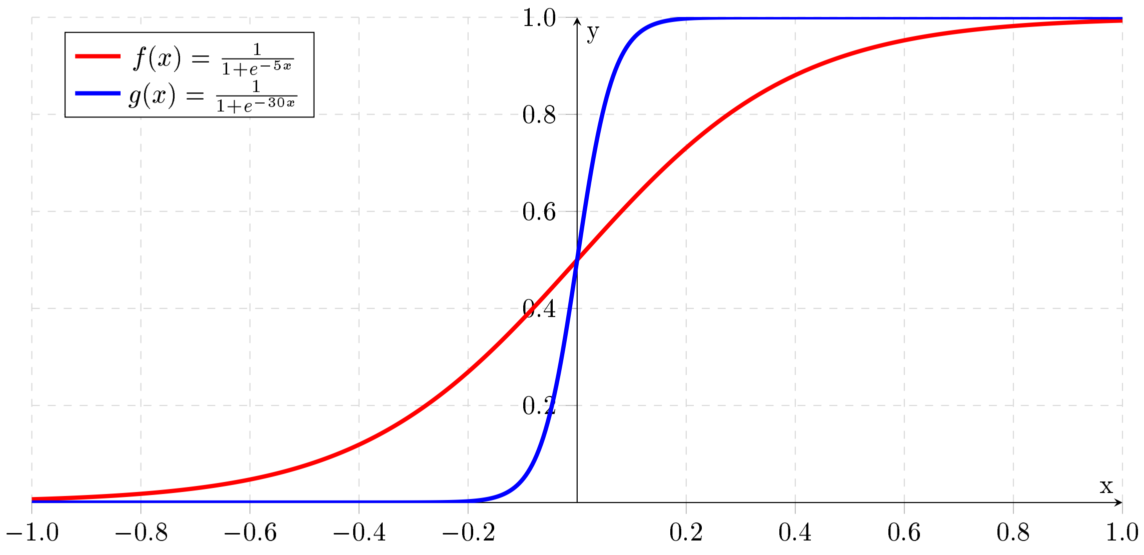


Figure 4: Example of a sigmoid function. The parameters θ^T determine the steepness.

For some classification problems, a linear logistic regression model is not enough. This happens when a linear decision boundary is not enough to distinguish between the different classes. In these cases, a non-linear function is needed.

3.1 Cost function

In logistic regression, the same squared error cost used in linear regression cannot be used because it doesn't make much sense. A cost function such as the squared error measures the *distance* between the predict value and actual value, however, this distance is not easily identified for a classification problem, where the actual value is which class the data belongs in.

$$Cost(h(x), y) = \begin{cases} -log(h(x)) & \text{if } y = 1 \\ -log(1 - h(x)) & \text{if } y = 0 \end{cases} \tag{13}$$

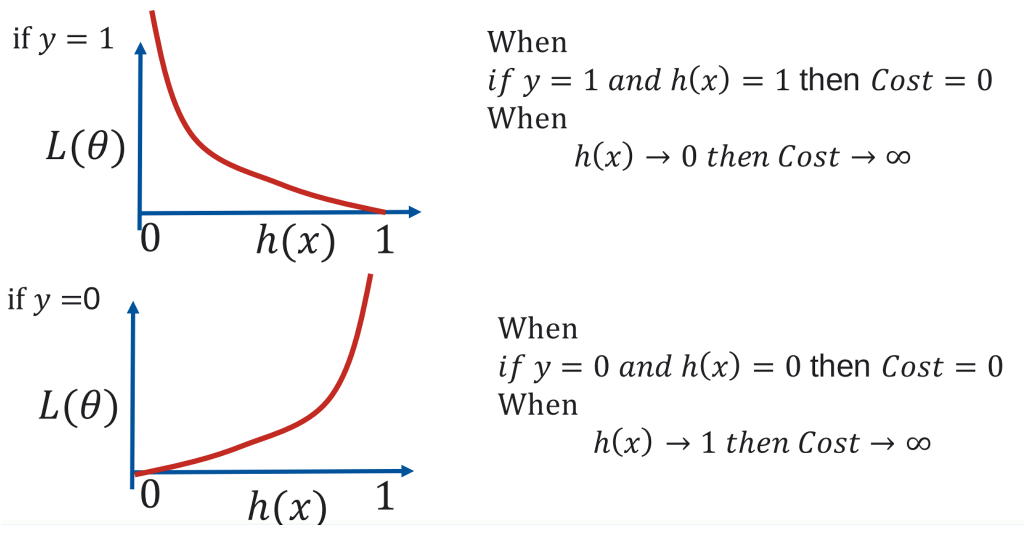


Figure 5: Cost function for logistic regression.

Log functions are used as the cost function so that the cost for a prediction of $h(x) \approx 1$ tends to 0 and a prediction of $h(x) \approx 0$ tends to ∞. This this cost function, logistic regression can also use gradient descent to tweak θ parameters.

3.2 Multiclass classification

There are often cases in classification where it is not simply a binary classification such as cancer or not cancer. This is multiclass classification where there are multiple classes, such as classifying the colour based on wavelength. Here, simply applying the previous techniques and cost functions is not enough as it will only give the probability of one certain class. To deal with multiclass problems, the typical solution is to use a **one-vs-all** method where each class is treated separately as a binary classification task. There are three main steps for doing this:

- Choose one class apply binary classification to all data points
- Repeat for all other classes
- Generate probability for each class based on the binary classification and pick the classification with the highest probability

4 Regularisation

When using too many features, it is often the case that overfitting happens where the model fits well to the training data, but does not generalise to unseen data. A typical way to deal with overfitting is to reduce the number of features, either manually or automatically. This is done because some input features may not be very relevant to the general problem, but still affect the prediction.

Regularisation is another method to reduce the effect of some input features without removing them completely. It works by keeping all input features, but using multipliers to reduce the scale of certain θ parameters. This allows more relevant features to have more impact while still keeping a lower effect of less relevant features. The parameters are penalised using a large multiplier to cause them to be minimised because of the increased constant. For example $... + 1000\theta_3 + 1000\theta_4$.

This changes the cost function by adding a regularisation parameter λ.

$$\text{Cost} = \frac{1}{2m}\left[\sum_{i=0}^{m}(h(x^i) - y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j\right] \tag{14}$$

The regularisation parameter λ determines how much to suppress the θ parameters. If λ is too large, then we may underfit as we have minimised the θ parameters too much. If λ is too small, then we may continue to overfit as it does not do enough to reduce the effect of θ parameters.

The regularisation parameter can also be added to the normal equation.

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right)^{-1} X^T y \tag{15}$$

In general, regularisation can be thought of as adding an extra component to the equation of minimising the cost function.

$$J(\theta) = L(\theta) + R(\theta) \tag{16}$$

where $J(\theta$ is the cost, $L(\theta)$ is the loss and $R(\theta)$ is the regularisation. Further, regularisation has a computational advantage over feature selection, as additional training and searching of models is not needed.

4.1 Ridge regression

The $\lambda\sum_{j=1}^{n}\theta_j$ term for regularisation can be calculated using the l_2 norm, usually denoted as $||\theta||_2^2$. The l_2 norm is the euclidean norm, which is a sum of squares of all elements, defined as:

$$||\theta||_2 = \sqrt{\sum_{j=1}^{n}\theta_j^2} \tag{17}$$

It measures the distance of θ from zero. As λ increases, the l_2 norm will *always* decrease. The reason regularisation has an advantage of a simple least squares regression is based on the **bias-variance trade-off**. As λ increases, the flexibility of the model decreases, leading to decreased variance, but increased bias. For some values of λ up to a limit, the variance will decrease rapidly with only a small increase in bias, which reduces the overall mean squared error of the model.

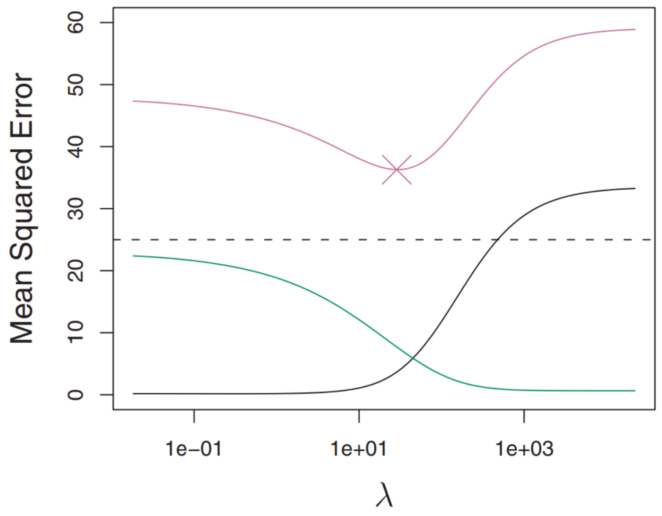


Figure 6: Plot showing why ridge regression leads to lower MSE for some values of λ. Squared bias (black), variance (green) and test MSE (purple) for the ridge regression predictions on a simulated dataset as a function of λ. Taken from *An Introduction to Statistical Learning*.

4.2 Lasso regression

The disadvantage of ridge regression is that while it will shrink all coefficients towards zero, it will not set them to be exactly zero unless $\lambda = \infty$. In other words, for datasets with a large number of features, increasing *lambda* will reduce their effect but not exclude any of them completely.

Lasso regression works the same way as ridge regression, except instead of using the l_2 norm, the l_1 norm is used. Rather than being a sum of squares, the l_1 norm is simply a sum of all elements.

$$||\theta||_1 = \sum_{j=1}^{n} |\theta_j| \tag{18}$$

Lasso regression also shrinks the coefficient estimates towards zero. However, the l_1 penalty is able to force some coefficient estimates to be equal to zero when the *lambda* value is large enough. In this sense, lasso regression performs variable selection like selecting the best features. It can be said that lasso regression yields *sparse* models because the models may only involve a smaller subset of features.

5 Evaluation

The goal of supervised machine learning is to train the models on known data in order to predict new data. To be able to evaluate how well the model works on new data, any given data has to be split into separate training and test data sets. The testing data is used as a final step after training models to evaluate how well they perform. Testing data should *never* be used to train, as otherwise this would bias the model heavily towards the testing data, rather than towards general unseen data. Further, testing data should not be used for any step of the machine learning process except for checking the performance of trained models at the very end.

5.1 Source of errors

- Not enough data which leads to overfitting to available data and inability to generalise

- Fine tuning parameters on test data which still causes overfitting because the model is designed to work well only on available data
- Training or making decisions based on testing data could lead to very low error rates on available data, but the heavy bias is undesirable

Most often, the source of errors in machine learning comes from overfitting in some way, either from looking and training towards the test data, or from not enough data or too many input features.

5.2 Classification metrics

Classification differs from regression in that all errors are treated the same. It is not as easy as using functions such as the root mean squared error or R^2 coefficient of determination. Classification are either correct or incorrect, which makes errors simpler as there is no need to calculate how far the predicted value is.

- Correct classification = true positives + true negatives
- Incorrect classification = false positives + false negatives

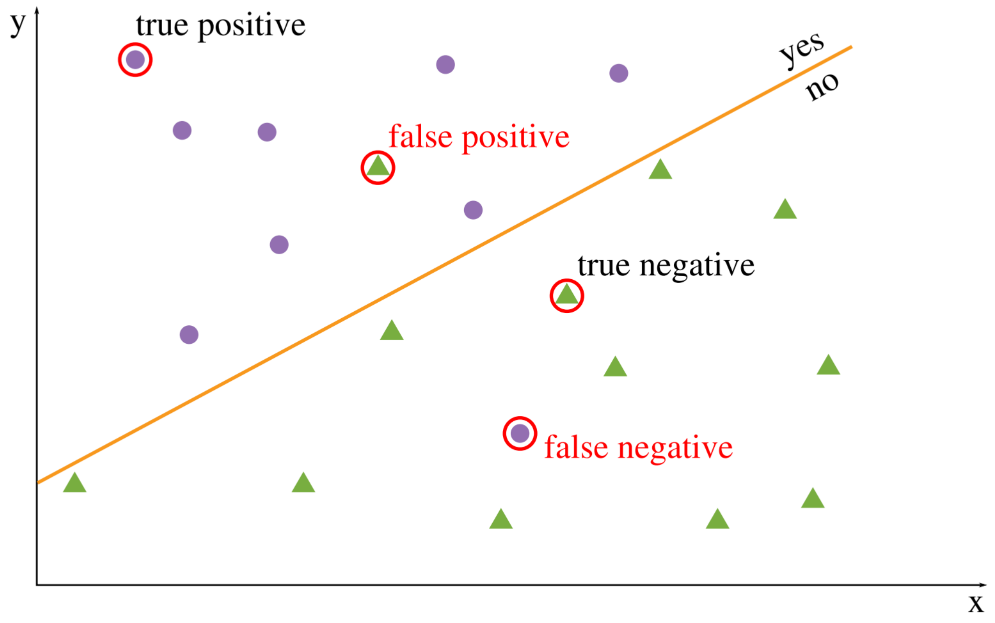


Figure 7: Examples of the different types of correct/incorrect classification

With different combinations of true/false positives/negatives, different metrics can be calculated for classification problems. For example, the accuracy (or classification rate) is defined as:

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{total positives} + \text{total negatives}} \tag{19}$$

There are many further metrics that can be used for classification, most notably are **precision** and **recall**.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{20}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{21}$$

The precision is the fraction of correct classifications among all classified data points and the recall is the fraction of correct classifications among all relevant data points. In other words, the precision tells us how much of what was classified is correctly classified, while the recall tells us how much of the positive classifications were found out of the total positive data points. The two scores represent a trade-off as to how correct the classifications are against how many relevant data points were found. This trade-off depends on the actual classification problem. For example, in the case of cancer identification, it may be better to bias towards false positives (higher recall, lower precision) to be on the safe side.

The trade-off between precision and recall can often be visualised by varying a threshold on the classification and plotting a precision-recall curve.

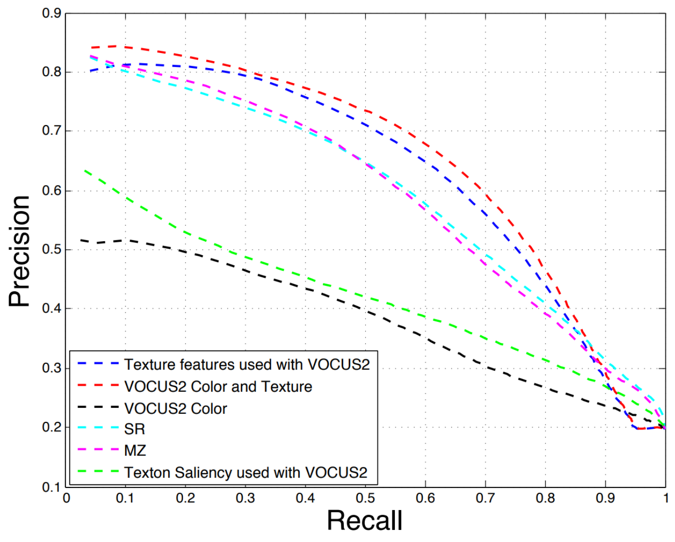
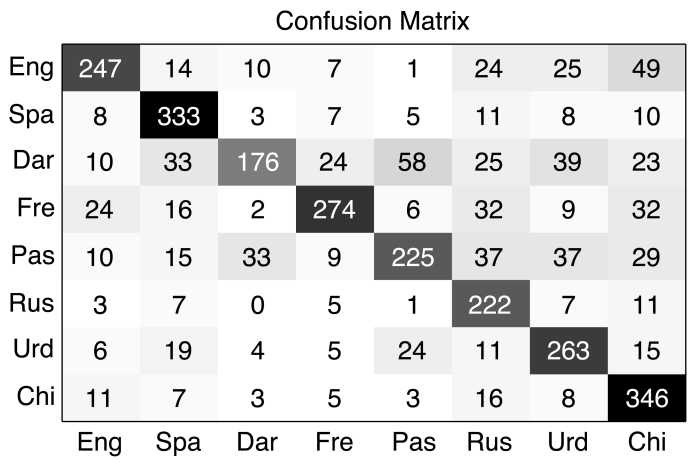


Figure 8: Precision-recall curves.

It is often difficult to compare precision-recall curves as different curves could represent different trade-offs. It is more useful to have a single number to compare, so the area under the curve is used as an alternative. The larger the area, the better the performance. This is because a good model that achieves both high precision and recall will have a larger area under the curve. This area is the **average precision** for one binary classification task averaged over all possible parameter values. With multiple classes, a precision-recall curve would be needed for each class in the one-vs-all method. To better visualise this for an average over all classes, a **confusion matrix** can be used.



Confusion Matrix

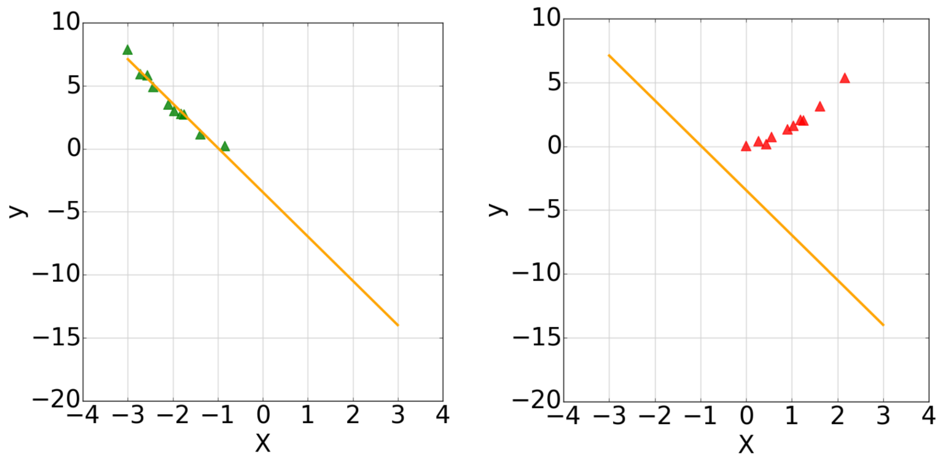
	Eng	Spa	Dar	Fre	Pas	Rus	Urd	Chi
Eng	247	14	10	7	1	24	25	49
Spa	8	333	3	7	5	11	8	10
Dar	10	33	176	24	58	25	39	23
Fre	24	16	2	274	6	32	9	32
Pas	10	15	33	9	225	37	37	29
Rus	3	7	0	5	1	222	7	11
Urd	6	19	4	5	24	11	263	15
Chi	11	7	3	5	3	16	8	346

Figure 9: Example of a confusion matrix.

The diagonal represents correct detections for that class. The rows and columns represent false positives and false negatives, as they are predicted the wrong class. Confusion matrices are useful to see which classes are better predicted compared to other classes

5.3 Validation

Because testing data is sometimes completely hidden, the given training data may need to be split into a validation set, which is used to evaluate the quality of the model. There is an issue during the splitting phase of training and validation data where the two sets are not good representation samples from the dataset. For example, if the dataset is split in half exactly, many important features may be missed.



(a) Prediction on training data (b) Prediction on validation data

Figure 10: Extreme example of an issue that can occur in splitting training/validation data.

The split training/validation datasets have to be representative samples from the domain of the entire dataset. However, this can be difficult to do without looking at the dataset, which compromises the experiment. A method to deal with this is called **stratified sampling**. In classification, this would ensure a representative number of classes are in each dataset. For regression problems, certain important features need to be used as the feature for stratification, for example gender proportions in data about people.

Another issue with splitting the data into a validation set is that it reduces the amount of data available for training. Further, randomly splitting the dataset, even with stratified sampling could lead to getting "easier" or "harder" datasets to work with. Another method to evaluate the performance of trained models is **k-fold cross validation**.

5.3.1 k-fold cross validation

The idea behind k-fold cross validation is to split the dataset into k equally sized subsets and run training k times, leaving out one of the subsets for testing. Each k samples is used for validation exactly once. Larger values of k means more splitting up of the data, which gives more reliable results, but requires more computation.

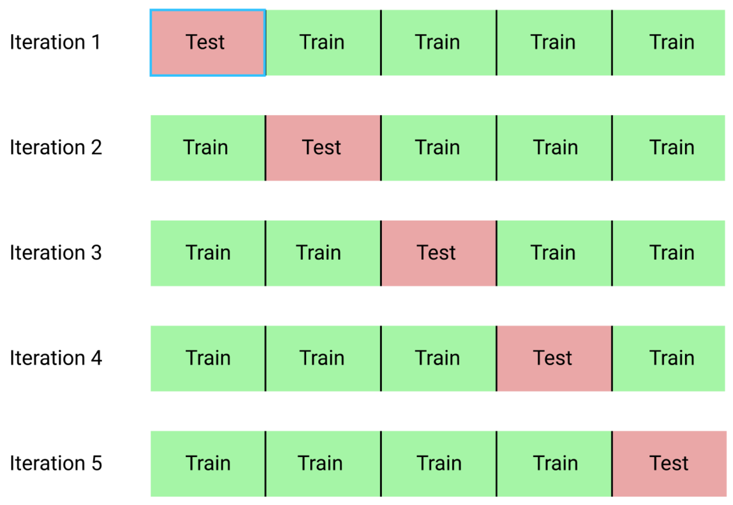


Figure 11: Example of 5-fold cross validation.

6 Data preparation

Often times the raw data received at the start of the machine learning process requires cleaning or preprocessing. This can be due to errors during data gathering, for example negative readings that aren't possible, or the way the data is presented, such as categorical values like colour (red, blue, green) or time. Data such as this has to be processed to be able to be passed to any machine learning algorithm, as we need numerical inputs in some cases.

Example data preparation process:

- Clean data
- Convert data types (categorical with dummy variables)
- Find mean and standard deviation of features
- Normalise data
- Select a subset of features
- Split into training and test/validation set
- Train algorithm
- Evaluate on test set

6.1 Categorical inputs

Make	Horsepower	Color	Accidents
Aubi TT	220	Red	13
Citreom C4	160	Blue	1
Fauxhall Corso	98	Red	2
Citreom C4	124	Black	0
Aubi TT	196	Red	7

Table 1: Example data with categorical inputs (colour)

Some algorithms such as decision trees and random forests can deal with categorical data, but for models such as linear regression, numerical data is required. There are two main methods of transforming categorical inputs into numeric inputs: **enumeration** and **dummy variables**.

6.1.1 Enumeration

To encode categories by enumeration, each class of the category can be assigned a number, for example assigning red = 1, blue = 2 etc. This is a very simple way to transform categories to numbers but creates a new issue. The distance between the numbers subtly implies a similarity between the values. For example if red = 1, blue = 2 and black = 3, we are implying that blue is similar to both red and black while red and black are not as similar to each other. Sometimes this can be desirable if encoded correctly to give more detailed information, such as if the colour numbers were ordered in a way that represented the difference.

Make	Horsepower	Color	Accidents
1	220	1	13
2	160	2	1
3	98	1	2
2	124	3	0
1	196	1	7

Table 2: Transformed categorical make and colour data using enumeration

6.1.2 Dummy variables

Instead of enumerating all the categorical values, a vector of ones and zeroes is used where each value corresponds to each possible category and the 1 or 0 represent true and false.

TT	C4	Corso	Horsepower	Red	Blue	Black	Accidents
1	0	0	220	1	0	0	13
0	1	0	160	0	1	0	1
0	0	1	98	1	0	0	2
0	1	0	124	0	0	1	0
1	0	0	196	1	0	0	7

Table 3: Transformed categorical make and colour data using enumeration

6.2 Input cleaning

When data is collected, there are often cases where mistakes or other errors appear and have to be cleaned beforehand. For example, all columns having the same data type and all rows having the same number of columns. Further errors can be dealt with in two ways, either try to fix them, for example inserting an average value to replace a NaN value, or remove the row entirely. The latter case may not be desirable in cases with little data. Other examples include spelling errors, negative values, faulty sensors and infinity values.

6.3 Feature selection

It was mentioned before that a reduced feature set can help reduce overfitting in the case of regularisation. In some cases, some features should be discarded entirely, especially if they are not relevant. There are a few good reasons for selecting only relevant features to use:

- It can improve accuracy with some models and reduce overfitting
- It can improve interpret-ability, as it is easier to understand models based on fewer parameters
- It can make infeasible problems feasible by significantly reducing the computational requirements

Feature selection can be done via **univariate tests** where the correlation between each features and the output is calculated and the good well correlated features are selected. Further, correlation between feature pairs can be calculated, as features that are well correlated to each other may be redundant.

There are also automatic algorithms for feature selection to select the k best features which give the best performance. Most notable, forward and backward stepwise selection.

6.3.1 Stepwise selection

In stepwise selection, features are either incrementally added or removed until the best number of features is found. In forward stepwise selection, each feature is added incrementally, trained on a new model and evaluated on validation or testing data. The best model is kept and another feature is added. This is repeated until the models no longer improve in performance and the best feature subset is selection.

Backwards stepwise selection works the other way by starting with all input features and removing features which affect the model the least.

6.4 Data augmentation

Data augmentation is a good technique both to increase size of the dataset and help models better generalise to new data. This is especially good for complex and modern classifiers, which may only work in very specific cases without data augmentation and need very large amounts of data. Augmentation can be done by transforming existing data in some way. For example shift, rotate, scale, distort, flip, change contrast in images. Domain knowledge is useful here to know how to augment the data.

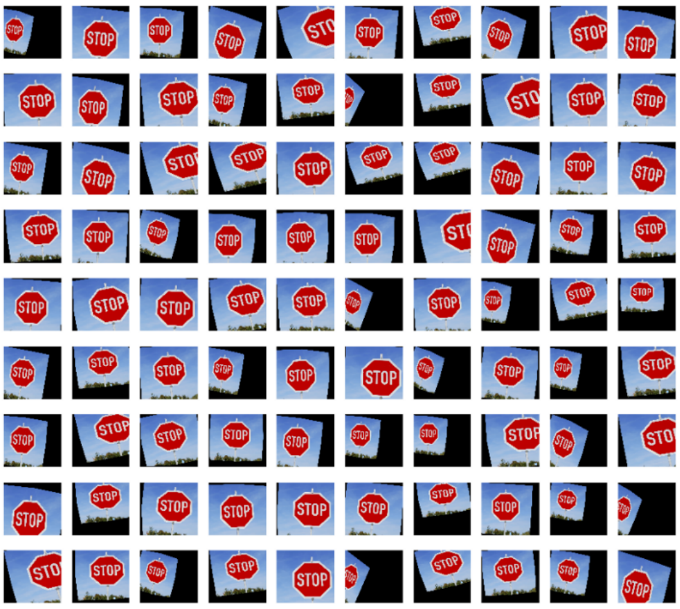


Figure 12: Example of data augmentation on images of a stop sign.

7 Basis expansion

7.1 Polynomial regression

To extend linear regression to settings where the function is non-linear, basis functions are used to construct a larger space of functions. A basis function (or number of basis functions) are defined on the input X. For example

$$h_1(X) = 1, h_2(X) = X, h_3(X) = X^2 \tag{22}$$

forms three basis functions which is used to transform the input. This transformation replaces the standard linear model

$$f(X) = \theta_0 + \theta_1 X \tag{23}$$

with new inputs

$$[h_1(X), h_2(X), h_3(X)]^T \rightarrow [1, X, X^2]^T \tag{24}$$

to get a polynomial function

$$\begin{align} f(X) &= \theta_0 h_1(X) + \theta_1 h_2(X) + \theta_2 h_3(X) \tag{25} \\ &= \theta_0 + \theta_1 X + \theta_2 X^2 \tag{26} \end{align}$$

Now the same linear regression can be run on the new input as the output is a linear combination of functions, which a linear regression model can still be applied on. The functions may be non-linear, but the input space has been transformed through the non-linear mapping. Of course, this can naturally be extended to m polynomials, but generally using more than 3 or 4 polynomials leads to high chances of overfitting.

7.2 Piecewise linear regression

Basis expansions are in fact more general than polynomial expansion. Basis functions can also be defined to create a piecewise linear regression model.

7.3 Regression splines

8 Bayesian classification

In Bayesian terms, input variables are called **evidence**. The Bayes rule allows the probability and evidence to be expressed into simpler distributions:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{27}$$

where

- $P(C_i|X)$ is the **posterior probability**. In other words, if we know X happened, how likely is C?
- $P(X|C_i)$ is the **likelihood**. If we know that the right answer is C, how likely is observation X?
- $P(C_i)$ is the **prior**, which is how often C_i is the correct answer in total. This can often be measured or estimated.
- $P(X)$ is the **probability of evidence**, which is how often this particular observation is obtained.