COMP9417 Notes

Alperen Onur

May 2025

1 Regression

1.1 Supervised & Unsupervised Learning

The two types of machine learning algorithms fall under supervised or unsupervised learning. Supervised learning categorises algorithms with given labels and unsupervised algorithms categorise algorithms with no given labels.

1.2 Linear Regression

Regression predicts numerical values whereas **Classification** predicts discrete values. **Linear Regression** is a particular type of regression which models the relationship between an input variable and an output variable using a straight line,

$$\hat{y} = bx + c$$

where, $\hat{y}$ are our predicted labels based on our input features x. The goal in our case is to estimate the slope b and intercept c from the data. We make the following assumptions for linear regression,

- Linearity: The relationship between x and the mean of y is linear.
- Homoscedesticity: The variance of residual is the same for any value of x.
- Independence: Observations are independent of each other.
- Normality of residuals: For any fixed value of x, y is normally distributed.

1.3 Linear Regression Formulation

In linear models, the outcome is a linear combination of attributes,

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = h(x)$$

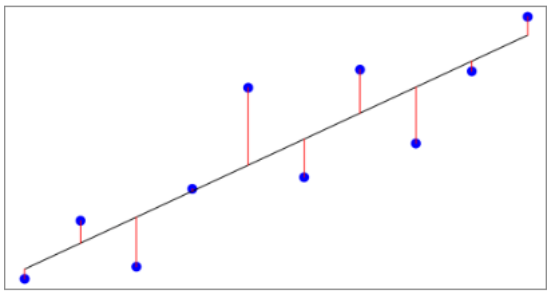
where θ_i is the weight from the observed training data for attribute i. The predicted value of the first training instance x_i is,

$$\hat{y}_1 = \theta_0 + \theta_1 x_{11} + \theta_2 x_{12} + \cdots + \theta_n x_{1n} = \sum_{i=0}^{n} \theta_i x_{1i} = x_1^T \theta = h(x_1).$$

Typically, x_0 is set to 1 and we define $x = [x_0, x_1, ..., x_n]^T$.

1.4 Minimising Error

We can fit an infinite amount of lines to a dataset depending on what we define as the best fit criteria. The most popular estimation model is "Least Square", also known as "Ordinary Least Squares" regression. This model attempts the minimise the difference between the predicted and actual values; minimising the error.



The goal is the minimise the error over all input samples. The total error is defined as the sum of squared errors and searching for $n + 1$ parameters to minimise that. The sum of squared error for m samples is denoted as,

$$J(\theta) = \sum_{j=1}^{m} (y_j - \sum_{i=0}^{n} \theta_i x_{ji})^2 = \sum_{j=1}^{m} (y_j - x_j^T \theta)^2 = (\mathbb{y} - X\theta)^T (\mathbb{y} - X\theta)$$

where $J(\theta)$ is the loss function and $X, \mathbb{y}$ are

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ & & \vdots & & \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \qquad \mathbb{y} = \begin{bmatrix} y_1 \\ y_2 \\ \\ y_m \end{bmatrix}$$

The loss function can also be generalised as averaging the squared error and minimising it which results in the same θ,

$$J(\theta) = \frac{1}{m}\sum_{j=1}^{m}(y_j - x_j^T)^2.$$

This is typically referred to as the **mean squared error (MSE)**.

1.5 Least Squares Regression

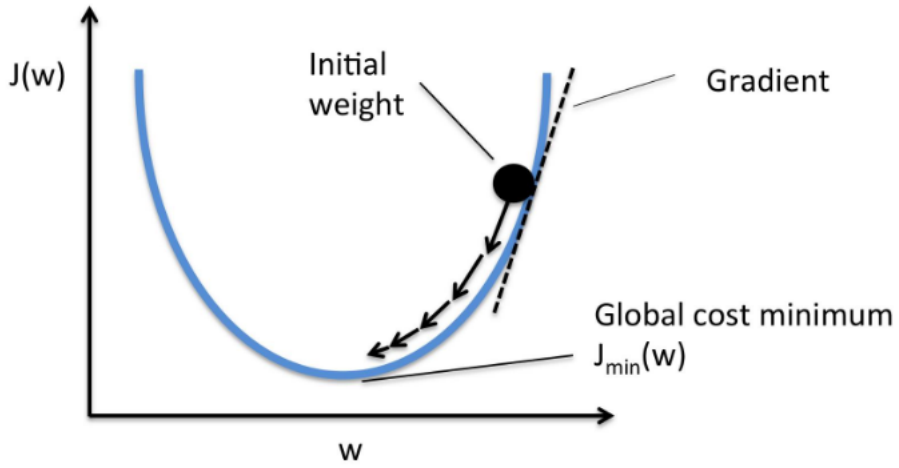
We need a method to estimate the parameters of the lost function as to minimise its overall cost. Less error means more accurate predictions. Computing $J(\theta)$ for different values of θ results in a convex function which has a single global minima. An algorithm to converge to this minima is **Gradient Decent**.

1.6 Gradient Decent

Gradient decent starts with some initial θ, and repeatedly performs an update,

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha\frac{\partial}{\partial\theta_i}J(\theta_i^{(t)})$$

where α is the learning rate. In each iteration of the algorithm, it takes a "step" the size of α in the direction with the steepest increase in $J(\theta)$. To implement this algorithm, we'll need the partial derivative of the MSE.



For one sample of m samples, the cost function is,

$$J(\theta) = (y_j - h_\theta(x_j))^2 = (y_j - \sum_{i=1}^{n} x_{ji}\theta_i)^2 \tag{1}$$

$$h_\theta(x_j) = \sum_{i=1}^{n} x_{ji}\theta_i = x_j^T\theta \tag{2}$$

Now taking the derivative we get,

$$\frac{\partial}{\partial\theta_i}J(\theta) = -2(y_j - h_\theta(x_j))x_{ji}. \tag{3}$$

So for a single training sample, the update rule is,

$$\theta_i^{(t+1)} = \theta_i^{(t)} + 2\alpha(y_j - h_\theta(x_j))x_{ji}. \tag{4}$$

This update rule for squared distance is called **Least Mean Squares (LMS)**. For multiple samples, there are a couple methods used for updating the LMS rule: **Batch Gradient Decent** and **Stochastic Gradient Decent**.

1.7 Batch Gradient Decent

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \alpha\frac{2}{m}\sum_{j=1}^{m}(y_j - h_{\theta_i^{(t)}}(x_j))x_{ji}.$$

For every i, replace the gradient with the sum of gradient for all samples until convergence (when θ is stabilised).

1.8 Stochastic Gradient Decent

For $j = 1$ to m:
[1] $\theta_i^{(t+1)} := \theta_i^{(t)} + 2\alpha(y_j - h_\theta(x_j))x_{ji}$ (for every i)
Repeat until algorithm converges. θ gets updated at each sample separately. This is much less costly than batch gradient decent but it may also never converge to a minimum.

1.9 Minimising Square Error with Normal Equations

We can also find the minimum of $J(\theta)$ by explicitly taking its derivatives and setting them to zero. This is the closed form solution,

$$\frac{\partial}{\partial\theta}J(\theta) = 0 \tag{5}$$

$$J(\theta) = (\mathbb{y} - X\theta)^T(\mathbb{y} - X\theta) \tag{6}$$

$$\frac{\partial}{\partial\theta}J(\theta) = -2X^T(\mathbb{y} - X\theta) = 0 \tag{7}$$

$$X^T(\mathbb{y} - X\theta) = 0 \tag{8}$$

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

1.10 Minimising Square Error with Probabilistic Interpretation

We can write the relationship between an input variable x and output variable y as,

$$y_j = x_j^T\theta + \epsilon_j$$

where ϵ_j is an error term (random noise). If we assume all ϵ_j are independent and identically distributed according to the Gaussian distribution $\epsilon_j \sim N(0, \sigma^2)$ then,

$$p(\epsilon_j) = \tfrac{1}{\sqrt{2\pi\sigma^2}}\exp(-\tfrac{\epsilon_j^2}{2\sigma^2}).$$

This implies that, $P(\epsilon_j) = P(y_j|x_j\,;\theta) = \tfrac{1}{\sqrt{2\pi\sigma^2}}\exp(-\tfrac{(y_j - x_j^T\theta)^2}{2\sigma^2})$. So we want to estimate θ such that we maximise the probability of output y given input x over all m training samples. Mathematically this is,

$$\mathcal{L}(\theta) = P(\mathbb{y}|X\,;\theta)$$

where $\mathcal{L}(\theta)$ is the likelihood function. Since we assumed independence over ϵ_j, then each of our training samples are independent from each other. Then it follows that,

$$\mathcal{L}(\theta) = \prod_{j=1}^{m} P(y_j|x_j\,;\theta) \tag{10}$$

$$= \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}}\exp - \frac{(y_j - x_j^T\theta)^2}{2\sigma^2}. \tag{11}$$

This is called the **maximum likelihood**. If we want to find a θ that maximises $\mathcal{L}(\theta)$, we can also maximise any strict increasing function of $\mathcal{L}(\theta)$. If we choose to maximise the **log likelihood** $\ell(\theta)$,

$$\ell(\theta) = \log \mathcal{L}(\theta) = \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(y_j - x_j^T\theta)^2}{2\sigma^2} \tag{12}$$

$$= m \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \frac{1}{\sigma^2} \frac{1}{2} \sum_{j=1}^{m} (y_j - x_j^T\theta)^2. \tag{13}$$

So maximising $\ell(\theta)$ is equal to minimising $\sum_{j=1}^{m}(y_j - x_j^T\theta)^2$ which is the MSE.

2 Statistical Techniques for Data Analysis

2.1 Sampling

Sampling is a way to draw conclusions about a population without having to measure the entire population. For groups that are fairly homogeneous, we do not need to collect alot of data. For populations with alot of irregularities, we need to either take measurements from the entire group or find a some other way to get a good idea of the groups trends without having to do so. Sampling gives us a way to do this!

We want from our sampling method to have as little bias that we can account for and if the chance of obtaining an unrepresentative sample is high then we can choose to not draw conclusions. The chance of an unrepresentative sample decreases with the size of the sample.

2.2 Estimation

Estimation refers to the process by which one makes inferences about a population based on information obtained from a sample. A "good" estimate means that the estimator is correct on average. If the expected value of the estimator equals the true population parameter then it is said to be an unbiased estimator. The following are a few examples of different estimators,

- Sample mean: $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
- Sample median: The middle value when the sample data is ordered. Less affected by outliers; biased.
- Sample Variance $s^2 = \frac{1}{N-1} \sum_i (x_i - m)^2$
- Sample Standard Deviation: $s = \sqrt{s^2}$ Indicates the average distance of each sample from the mean; biased.

- Sample Range: Provides a simple measure of data spread by taking the difference between the maximum and minimum values; biased.

2.3 Covariance

Covariance is the measure of the relationship between two random variables,

$$cov(x, y) = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{N-1} = \frac{(\sum_i x_i y_i) - N\overline{xy}}{N-1}$$

2.4 Correlation

Correlation is a measure to show how strongly a pair of random variables are. The Pearson correlation between x and y is,

$$r = \frac{cov(x,y)}{\sqrt{var(x)}\sqrt{var(y)}}.$$

This only captures linear relationships between two variables. The Pearson correlation can range between -1 and 1. A value close to 1 shows high values of x are associated with high values of y and low values of x are associated with low values of y. Generally this means that the scatter is low. A value near 0 indicated there is no association; large scatter. A value close to -1 suggests a strong inverse association between x and y. Correlation is a quick way of checking whether there is some linear association between two variables x and y where the sign tells you the direction of that association.

3 Univariate Linear Regression

In order to find the parameters we take the partial derivatives, set them to 0 and solve for θ_0. This will lead to,

$$\theta_1 = \frac{cov(h, w)}{var(h)} \tag{14}$$

$$\theta_0 = \overline{w} - \theta_1 \overline{h}. \tag{15}$$

3.1 Linear Regression Intuitions

Adding a constant to all x values affects only the intercept but not the regression coefficient. We can then zero-centre the x values by subtracting the mean of x in which case the intercept is equal to the mean of y. Similarly we can subtract the mean of y

from all y values to achieve a zero intercept, without changing the regression problem in an essential way. Another important point is that the sum of the residuals of the MSE makes linear regression susceptible to outliers far from the regression line.

3.2 Multiple Regression

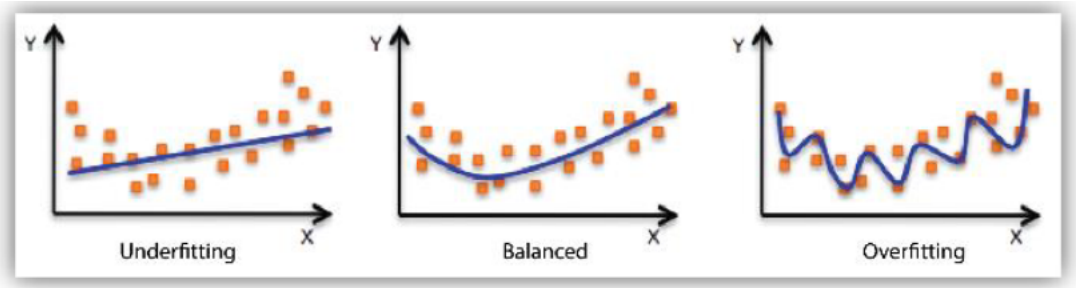
Often we need need to model the relationship of y to several other variables. Similar to univariate regression we can model this as,

$$\hat{w} = \theta_0 + \theta_1 h + \theta_2 b$$

and minimize the sum of the square residuals like so,

$$\sum_{j=1}^{m} (w_j - (\theta_0 + \theta_1 h_j + \theta_2 b_j))^2.$$

These types of regression models tend to produce curves rather than lines. So we can predict an output by adding different amounts of sample terms with each term increasing the overall degree of the models polynomial factor.



3.3 Regularisation

To control for overfitting or underfitting we introduce the concept of **regularisation**. Regularisation applies addition constraints the the weight vectors of a model. The regularised form for linear regression could be,

$$J(\theta) = \sum_{j=1}^{m} (y_j - h_\theta(x_j))^2 + \lambda \sum_{i=1}^{n} \theta_i^2.$$

The multiple least square regression problem is an optimisation problem and can be written as,

$$\theta^* = \arg\min \theta \ (y - X\theta)^T (y - X\theta)$$

with the regularised version of this being,

$$\theta^* = \arg\min \theta \ (y - X\theta)^T(y - X\theta) + \lambda||\theta||^2$$

where $||\theta||^2 = \sum_i \theta_i^2$ is the square norm of the vector θ; the dot product $\theta^T\theta$.

3.4 Ridge Regression

The regularised problem has a closed-form solution,

$$\theta = (X^T X + \lambda I)^{-1} X^T y$$

where I denotes the identity matrix. This adds λ amount of regularisation to the diagonal of $X^T X$. This is known as Ride Regression.

3.5 LASSO Regression

LASSO regression replaces the ride regression term $\sum_i \theta^2$ with the sum of absolute weights $\sum_i |\theta|$. LASSO regression favours more sparse solutions.

4 Train, Validation & Test Data

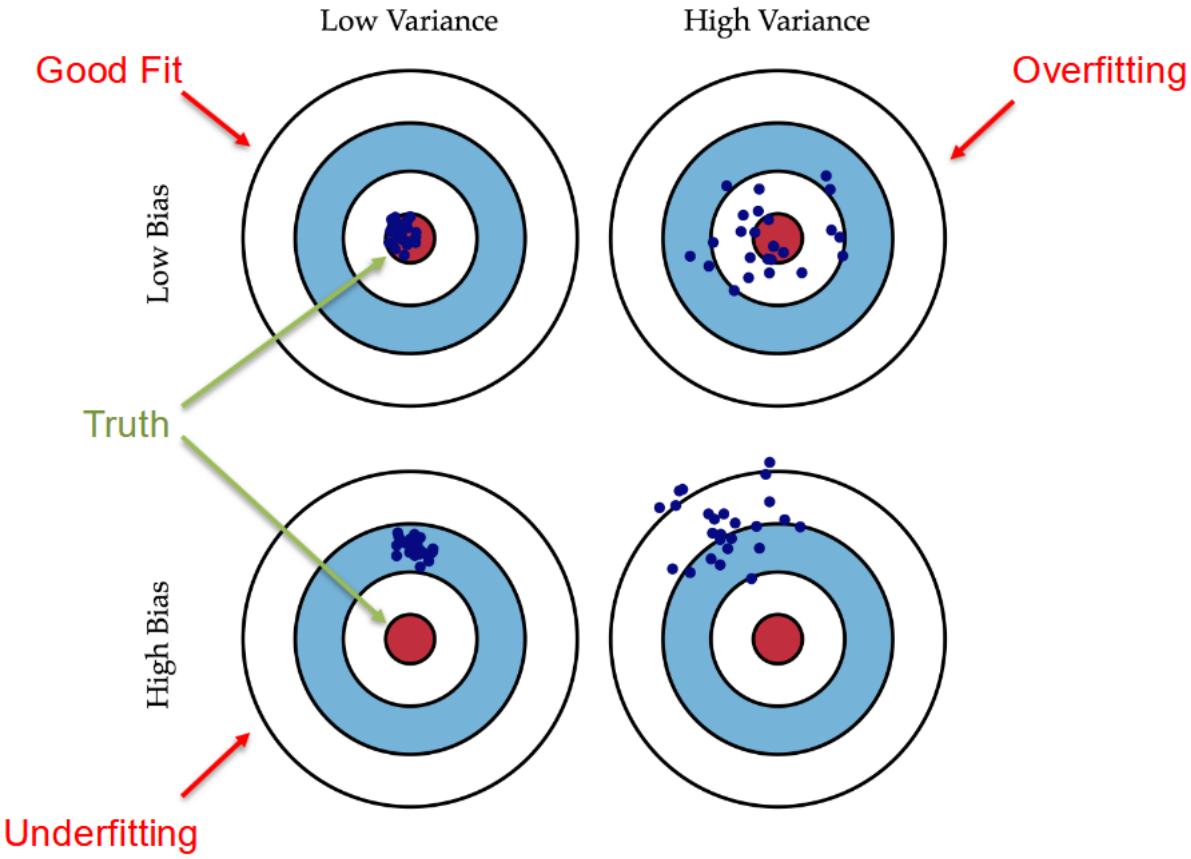
Train data is the data we use to learn our model and its parameters. Validation data is the unseen data by our model used to provide an unbiased evaluation of a model fit on the training dataset while tuning the models hyper-parameters. The test data is the data we use to test the model and shows how well our model generalises.

4.1 Model Selection

There are 3 ways to reduce complexity of a model,

- Subset-selection: search over a subset lattice such that each subset results in a new model and select one of those models.
- Shrinkage: Use regularisation to set coefficient of models to 0.
- Dimensionality reduction: Project points to a lower dimensional space.

4.2 Bias-Variance Tradeoff



Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Models with high bias pay very little attention to the training data and oversimplifies the model. **Variance** is the variability of the model for a given data point or a value which tells us the spread of data. Models with high variance pays a lot of attention to the training data but does not generalise on the data which has not been seen before. Underfitting means high bias and low variance. Overfitting means our model has captured a lot of noise along the underlying pattern in the data.

4.3 Bias Variance Decomposition

Whenw e assume $y = f + \epsilon$ and we estimate f with $\hat{f}$, then the expectation of error is,

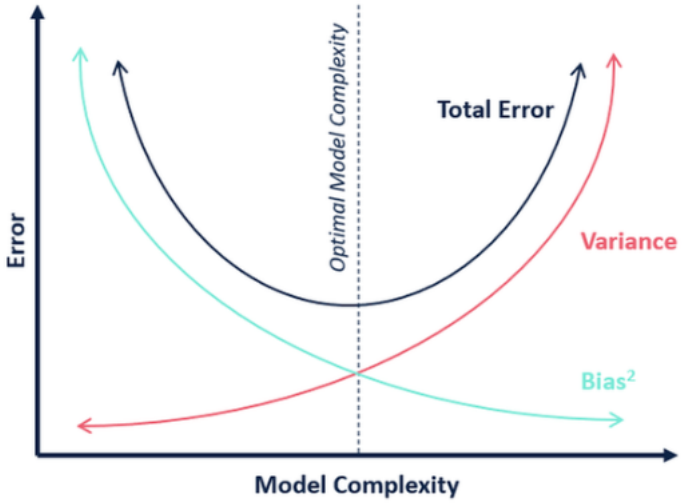
$$E[(y - f)^2] = (f - E[\hat{f}])^2 + \text{Var}(\hat{f}) + \text{Var}(\epsilon)$$

and so the MSE can be written as,

$$\text{MSE} = \text{Bias}^2 + \text{Variance} + \text{irreducible error}.$$

Irreducible error is the inherent uncertainty associated with a natural variability in a system. It can not be reduced since it is due to unknown factors. Reducible error is error that can and should be minimised further by adjustments to the model.

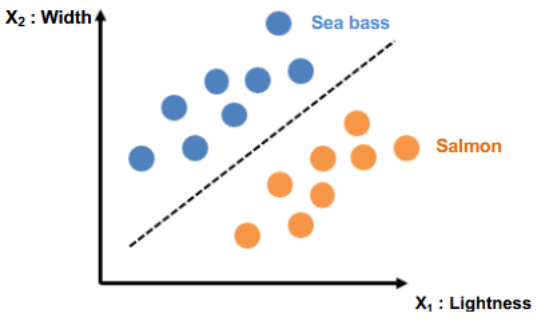
If our model is simple and has few parameters, then it may have high bias and low variance. On the other hand, if our model has a large number of parameters then it's going to have high variance and low bias. So we need to find a good balance without overfitting or underfitting to the data.



5 Classification

Classification is the concept of learning a classifier which is a function mapping from an input data point to one of a set of discrete outputs.

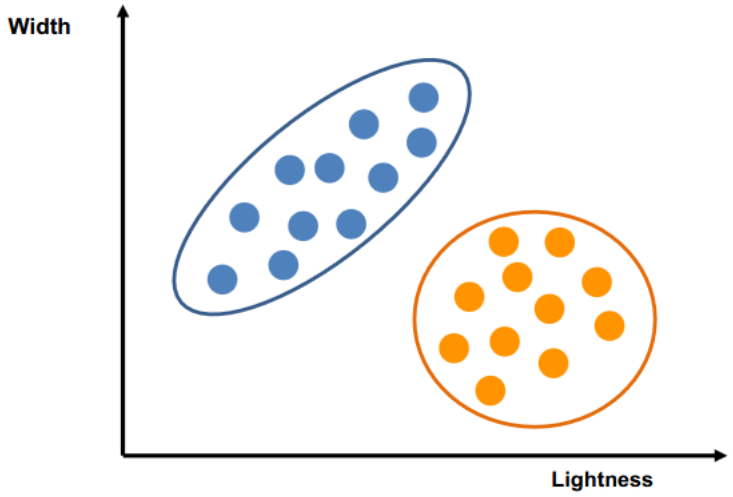
5.1 Linear Classifier



If we can find a line two separate two classes then this is called a linear classifier.

5.2 Generative algorithms

If we can instead of finding a discriminative line, build some models for each of the classes and make classifications predictions based on the test example to see which model it is most similar too. Generative algorithms learn $p(x|y)$ and $p(y)$ which is the class prior whereas discriminative algorithms only learn $p(y|x)$.



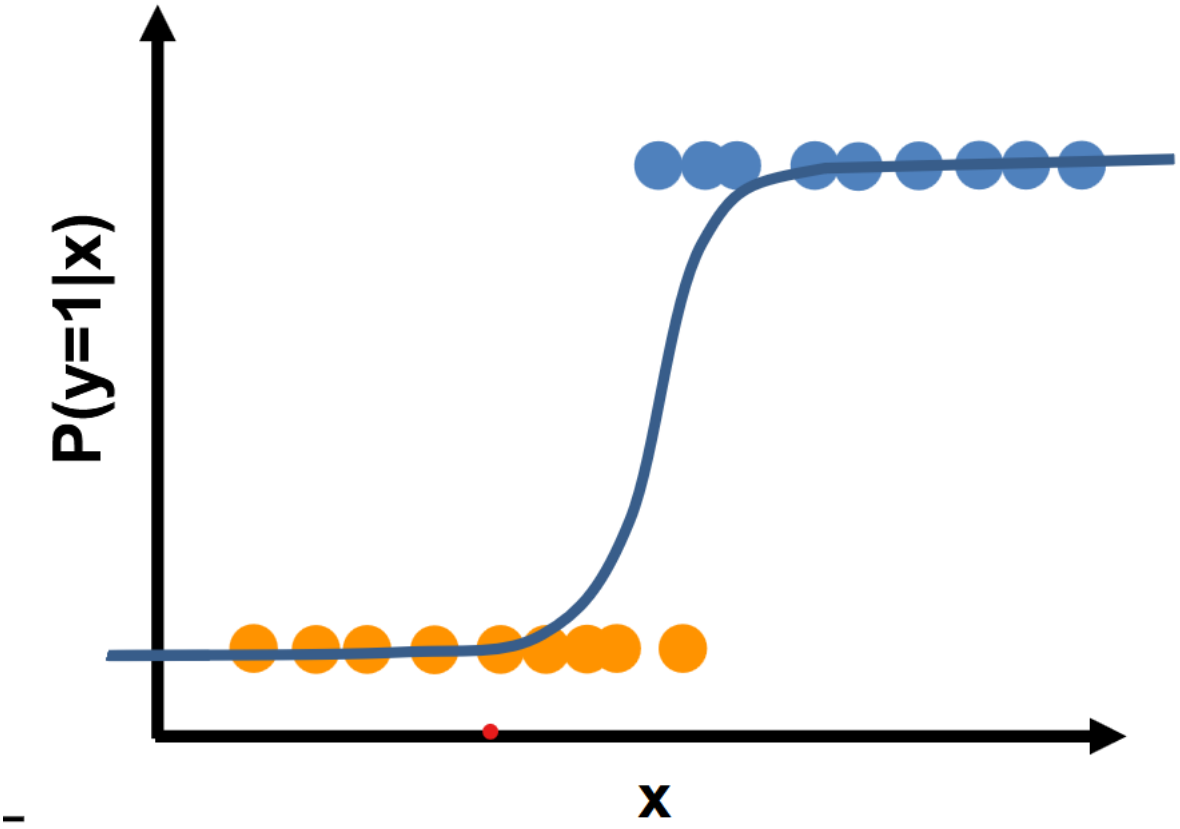
To predict the output sample x, we need to estimate $p(y|x)$,

$$p(y=0|x) = \frac{p(x|y=0)p(y=0)}{p(x)} \tag{16}$$

$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)} \tag{17}$$

5.3 Logistic Regression

In a binary classification, we can transform the y values into probability values in rage $[0, 1]$. We can model this using a sigmoid curve,



$$p(y=1|x) = \frac{1}{1+\exp(-f(x))} \tag{18}$$

$$\Rightarrow \log\left(\frac{P(y=1|x)}{1-P(y=1|x)}\right) \tag{19}$$

Now $f(x)$ can have a value between $-\infty$ and $+\infty$ so we can estimate $f(x)$ with a line,

$$\hat{f}(x) = x^T\beta \Rightarrow \log(\tfrac{P(y=1|x)}{1-P(y=1|x)}) = x^T\beta$$

We can generalise this to a linear solution with,

$$\hat{P}(y=1|x) = \tfrac{1}{1+\exp(-x^T\beta)}.$$

If $P(y=1|x) \geq 0.5$ then predict as class 1 and otherwise predict as class 0. We define the cost function as,

$$\text{cost}(h_\beta(x), y) = \begin{cases} -\log(h_\beta(x)) \text{ if } y = 1 \\ -\log(1-h_\beta(x)) \text{ if } y = 0 \end{cases}$$

where $\hat{P}(y=1|x) = h_\beta(x)$. This piecewise function can be combined into a single cost function as,

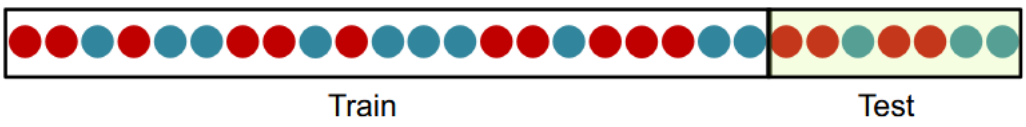
$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log(h_\beta(x^{(i)})) + (1+y^{(i)})\log(1-h)\beta(x^{(i)}))\right).$$

To find the parameters that minimise $J(\beta)$ the Gradient Decent algorithm can be used.

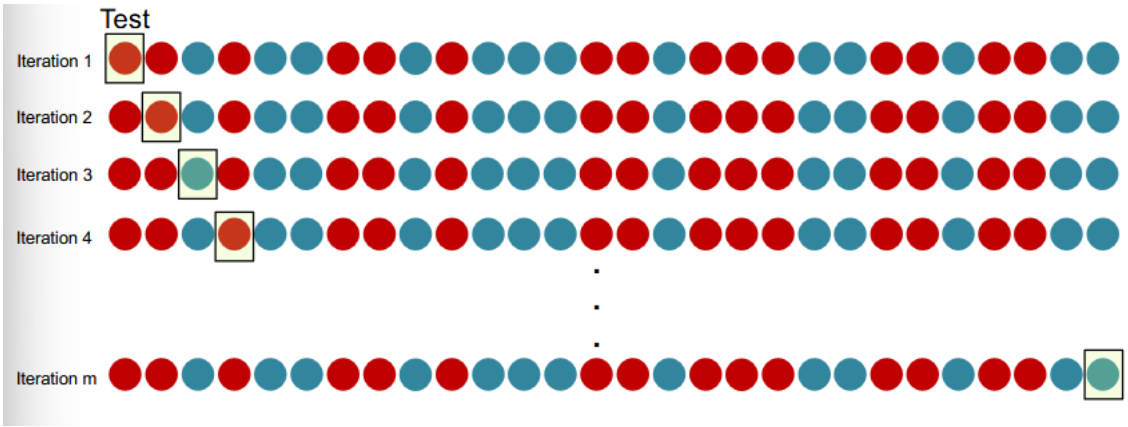
6 Generalisation

We want to generalise a sample set of data in the training set for our models. To examine how "general" our models are we can use **Cross-Validation** to asses the results of our model on an independent dataset.

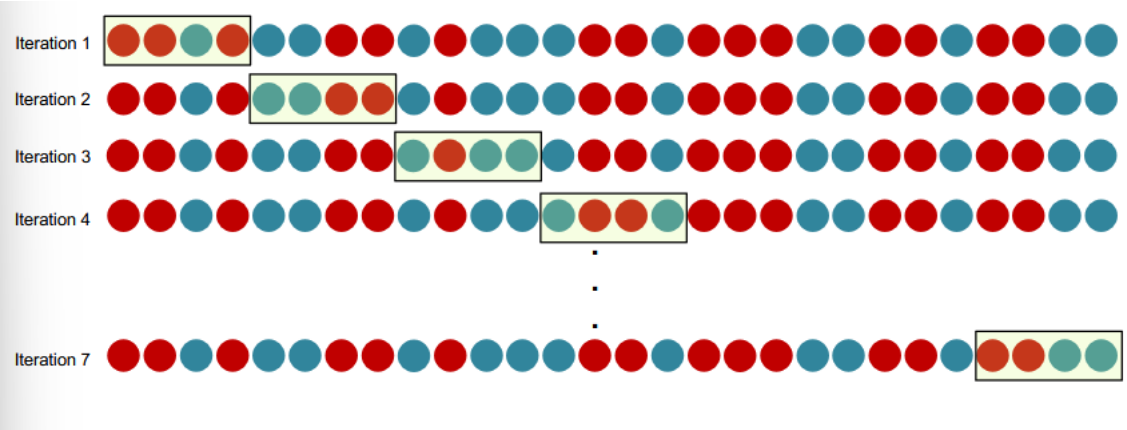
6.1 Holdout Method



6.2 Leave One Out Cross Validation

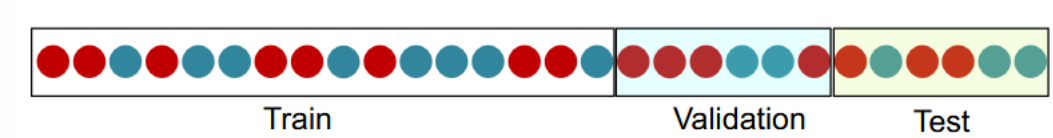


6.3 K-Fold Cross Validation



6.4 Validation set

If we want to estimate certain parameters during learning we'll instead use a validation set; separate from training and test data.



Estimating parameters is also called **hyperparameter tuning**.

6.5 Evaluation of Error

If we have a binary classification task then we have two classes, $y \in \{0, 1\}$, where we call the class $y = 1$ the positive class and $y = 0$ the negative class. Some evaluation metrics for this classification are:

- True Positive: predicted class of 1 is actually 1.
- True Negative: predicted class of 0 is actually 0.
- False Positive: predicted class of 1 is actually 0.
- False Negative: predicted class of 0 is actually 1.

6.6 Classification Accuracy

Classification accuracy on a sample of labelled pairs $(x, c(x))$ given a learned classification model that predicts, for each instance x, a class $\hat{c}(x)$,

$$\text{acc} = \frac{1}{|\text{Test}|} \sum_{x \in \text{Test}} I[\hat{c}(x) = c(x)]$$

where Test is a test set and $I[]$ is the indicator function which is 1 iff its argument evaluates to true and 0 otherwise. The classification error is $1-\text{acc}$.

6.7 Precision

The number of relevant objects classified correctly divided by the total number of relevant objects classified.

$$\text{Precision} = \frac{TP}{TP=FP}$$

6.8 Recall

The number of relevant objects classified correctly divided by total number of relevant objects classified correctly divided by the total number of correct objects.

$$\text{Recall} = \frac{TP}{TP+FN}$$

6.9 F1 Score

A measure of accuracy which is the harmonic mean of precision and recall defined as,

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

6.10 Missing values

Sometimes values will be missing in our data, in this case we have a few techniques to "fix-up" the data:

- Deleting samples with missing values

- Replacing the missing value with some statistics from the data (mean, median, ...)
- Assigning a unique category
- Predicting the missing values
- Using algorithms which support missing values

7 Nearest Neighbour

Nearest Neighbour is a classification algorithm that predicts whatever is the output value of the nearest data point to some query. To find the nearest data point, we need to define some metric of distance. There are many distance metrics including Minkowski distance, Euclidean distance, Manhattan distance, Chebyshev distance and more.

Exemplars are centroids that refer to a point to summarise or represent a large group of data. Sometimes **exemplars** are used in distance based learning because they can be selected based on their proximity to other points with the aim of minimising the distance between the exemplar point and each point in its cluster. Exemplars can be the arithmetic mean or geometric mean of the data.

7.1 Nearest Centroid Classifier

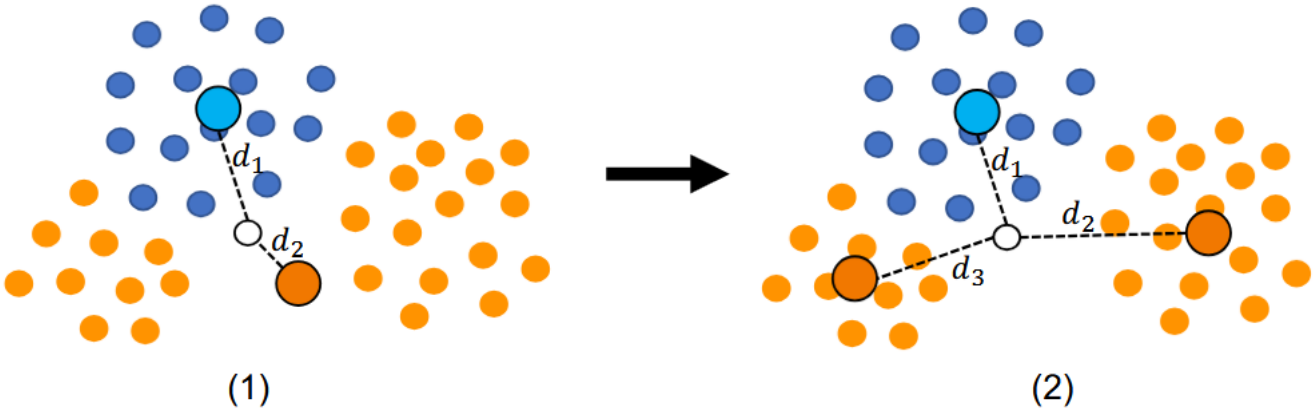
This is a classifier based on minimum distance principle where the exemplars are just the centroid means. The training sample pairs $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$ where x_i is the feature vector for the sample i and y_i is the class label define class centroids as,

$$\mu_k = \frac{1}{|C_k|} \sum_{j \in C_k} x_j.$$

The test set is a new unknown object with feature vector x is classified as class i if it much closer to the mean of class k than any other class mean vector.

If a class has more than one mode,

- If there is only one centroid per class, then it'll perform poorly
- If different modes can be defined, we can define one centroid per each mode which helps the classifier



7.2 Nearest Neighbour Classification

Refers to training instances that are searched for the instance that most closely resembles new or query instances. Store all training examples $< x_j, f(x_j) >$. Nearest neighbour for a given query instance x_q, first locate nearest training examples x_n, then estimate $\hat{f}(x_q) \leftarrow f(x_n)$. **$k$-Nearest Neighbour** instead will take a vote among its k nearest neighbours or the mean of f values of k nearest neighbours,

$$\hat{f}(x_q) \leftarrow \frac{\sum_{j=1}^{k} f(x_j)}{k}$$

7.3 K Nearest Neighbours (KNN) Algorithm

During training, for each training example $< x_j, f(x_j) >$, add the example to the list of training examples. During classification, given a query instance x_q to be classified, let $x_1, \cdots, x_k$ be the k instances from the training examples that are nearest to x_q be the distance function. We then return,

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{j=1}^{k} \delta(v, f(x_j))$$

where $\delta(a, b) = 1$ if $a = b$ and 0 otherwise. The distance function defines what is learned. Instance x_j can be described as a feature vector $x_j = (x_{j1}, \cdots, x_{jd})^T$ where x_{jr} denotes the value of the rth attribute/feature of x_j. The most commonly used distance function is Euclidean distance where the distance between x_i and x_j is defined to be,

$$\text{Dis}(x_{i,j}) = \sqrt{\sum_{r=1}^{d} (x_{ir} - x_{jr})^2}$$

7.4 Min-Max Normalisation

$$x'_{jr} = \frac{x_{jr} - \min(x_j r)}{\max(x_{jr}) - \min(x_{jr})}$$

where x_{jr} is the actual value of the attribute/feature r and x'_{jr} is the normalised value.

7.5 Z-score

$$x'_{jr} = \frac{x_{jr} - \mu_r}{\sigma_r}$$

7.6 More Nearest Neighbour

1NN perfectly separates training data so has low bias and high variance. By increasing k, we increase the bias and decrease the variance. If $k = m$ predictions will just be based on the majority class in the data set and this may overfit to the data. If we want to weigh nearer neighbours more heavily, we can use a distance function to construct a weight w_i and replace the classification algorithm by,

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_j))$$

where,

$$w_i = \frac{1}{\text{Dis}(x_q, x_i)^2}$$

8 Classification

Inductive Bias refers ti a set if assumptions that an ML model makes to generalise from limited training data. Different from bias. A strong inductive bias leads to high bias but lower variance and a weaker inductive bias reduces bias but increases variance. Frameworks in machine learning look for ways to represent the inductive bias.

8.1 Bayesian Machine Learning

Imagine a scenario where we need to classify different fish as either Salmon or Sea bass where C_i is the class. Probability based on past experience is called a **prior**. Say the prior probabilities were $P(c_{Salmon}) = 0.3$ and $P(c_{Seabass}) = 0.7$. Then only based on the

prior, sea bass would always be picked.

Say we have more information about the fish now: the length. Then we can update our predictions based on the length of the fish. These are called class conditionals.

$P(x\|c_i)$	Salmon	Sea bass
length > 100 cm	0.5	0.3
50 cm < length < 100 cm	0.4	0.5
length < 50 cm	0.1	0.2

Now we have $P(c_i)$ and $P(x|c_i)$ but what we want is $P(c_i|x)$. We calculate $P(c_i|x)$ using **Bayes Theorem**.

8.2 Bayes Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

where,

- $P(h)$ is the prior probability of hypothesis h,
- $P(D)$ is the prior probability of training data D,
- $P(h|D)$ is the probability of h given D and,
- $P(D|h)$ is the probability of D given h.

8.3 Decision Rule from Posteriors

If the output belongs to a set of k classes, $y \in \{C_1, C_2, \cdots, C_k\}$ where $1 \leq i \leq k$. Then in the Bayesian framework,

$$P(y = C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

where

- $P(y = C_i|x)$ is the **posterior probability**,
- $P(x|C_i)$ is the class conditional,

- $P(C_i)$ is the prior and,
- $P(x)$ is the marginal ($P(c) = \sum_i p(x|C_i)p(C_i)$).

The maximum posteriori hypothesis h_{MAP} gives the most most probable hypothesis on the training data,

$$h_{MAP} = \arg\max_{h \in H} P(h|D) \tag{20}$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \tag{21}$$

$$= \arg\max_{h \in H} P(D|h)P(h) \tag{22}$$

If $P(h_i) = P(h_j)$ then we can further simplify the **Maximum Likelihood Hypothesis**,

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i)$$

Coming back to the example about fishes and their lengths,

$P(x\|c_i)$	Salmon	Sea bass
length > 100 cm	0.5	0.3
50 cm < length < 100 cm	0.4	0.5
length < 50 cm	0.1	0.2

$$P(c = salmon|x = 70cm) \propto P(70cm|salmon) * P(salmon) = 0.12 \tag{23}$$

$$P(c = seabass|x = 70cm) \propto P(70cm|seabass) * P(seabass) = 0.35 \tag{24}$$

8.4 Posterior Probability

For a hypothesis h, the posterior probability is,

$$P(h|D) = \frac{P(D|h)P(h)}{\sum_{h_i \in H} P(D|h_i)P(h_i)}$$

where the denominator ensures all posterior probabilities sum to 1.

8.5 Predictions with Posterior Probabilities

If there are two competing hypothesis h_1 and h_2 with posterior probabilities $P(h_1|D)$ and $P(h_2|D)$ respectively, then the potential decision is made based on the higher posterior probability. An alternative is the make a decision based on the loss that occurs when decision h is made using a loss function $L(h)$. If the cost of misclassification is not the same for different classes, then we need to minimise the expected loss,

$$E[L(\alpha_i)] = R(a_i|x) = \sum_{h \in H} \lambda(a_i|h) P(h|x)$$

where, the loss associated to action a_i is $\lambda(a_i|h)$.

8.6 Learning A Real Valued Function

To compute $P(D|h)$ and $P(h)$ we can assume a parametric model and estimate the parameters using the data. Consider any real-valued function f and training samples $< x_i, y_i >$ where y_i is noisy training value. Then we have,

$$y_i = f(x_i) + \epsilon_i$$

where ϵ_i is random noise drawn independently for each x_I according to some Normal(Gaussian) distribution with variance σ^2 and mean zero. Then the maximum likelihood hypothesis, h_{ML} is the one that minimises the sum of squared errors,

$$h_{ML} = \arg\max_{h \in H} P(D|h) = \arg\max_{h \in H} \prod_{i=1}^{m} P(y_i|h) \tag{25}$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} (\frac{y_i - h(x_i)}{\sigma})^2 \right) \tag{26}$$

taking log to give a simpler expression,

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{1}{2} (\frac{y_i - h(x)}{\sigma})^2 \tag{27}$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} -\frac{1}{2} (\frac{y_i - h(x_i)}{\sigma})^2 \tag{28}$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} -(y_i - h(x_i))^2 \tag{29}$$

$$= \arg\min_{h \in H} \sum_{i=1}^{m} (y_i - h(x_i))^2 \tag{30}$$

8.7 Bayes Optimal Classifier

The Bayes optimal classifier is a probabilistic model that makes the most probable prediction for a new sample based on the training data. In MAP framework, we seek the most probable hypothesis among a space of hypothesis. In Bayesian optimal classification, the most probable classification of the new instance is obtained by combine the predictions of all hypotheses, weighted by their posterior probabilities,

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i) P(h_i|D).$$

The optimal classification oof the new instance is the value v_j, for which $P(v_j|D)$ is maximum.

8.8 Bayes Error

Define the probability of error for classifying some instance x by, $P(error|x) = P(class_1|x)$ if we predict $class_2$ and $P(error|x) = P(class_2|x)$ if we predict $class_1$. This gives,

$$\sum P(error) = \sum_x P(error|x)P(x).$$

If $\hat{P}(class_1|x) > \hat{P}(class_2|x)$ predict $class_1$ and otherwise predict $class_2$.

8.9 Naive Bayes Classifier

Assume target function $f : X \rightarrow V$, where each instance x described attributes $< x_1, x_2, \cdots, x_n >$. The most probable value of $f(x)$ is,

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j|x_1, x_2, \cdots, x_n) \tag{31}$$

$$= \arg\max_{v_j \in V} \frac{P(x_1, x_2, \cdots, x_n|v_j)P(v_j)}{P(x_1, x_2, \cdots, x_n)} \tag{32}$$

$$= \arg\max_{v_j \in V} P(x_1, x_2, \cdots, x_n|v_j)P(v_j) \tag{33}$$

In Naive Bayes we assume $P(x_1, x_2, \cdots, x_n|v_j) = \prod_i P(x_i|v_j)$ meaning attributes are statistically independent. Now we get,

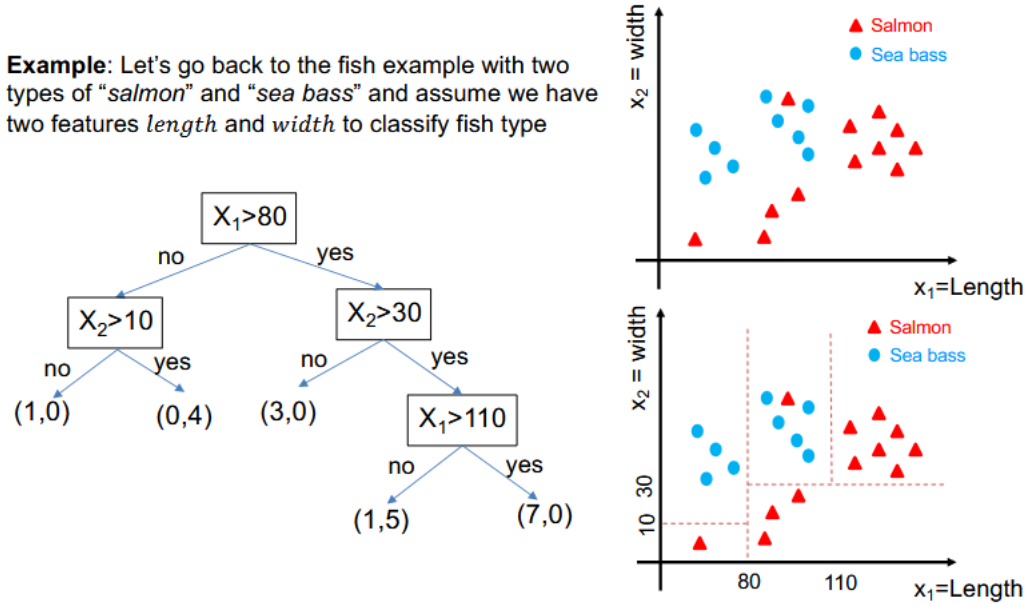
$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(x_i|v_j)$$

8.10 Naive Bayes Algorithm

For each target value v_j,
[1] $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$
[2] For each attribute value x_i:
[3] $\hat{P}(x_i|v_j) \leftarrow$ estimate $P(x_i|v_j)$ Classify the new instance,

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(x_i|v_j)$$

9 Decision Trees



Decision Trees work in a divide and conquer fashion where the data is split into subsets based on a criteria. If the data at the leaves are homogeneous/pure in terms of their classified value, then stop splitting. This type of splitting means that each leaf node assigns a classification. The difficult part of decision tress is at each node, which attribute/feature do we chose to split by?

9.1 Top-Down Induction of Decision Trees

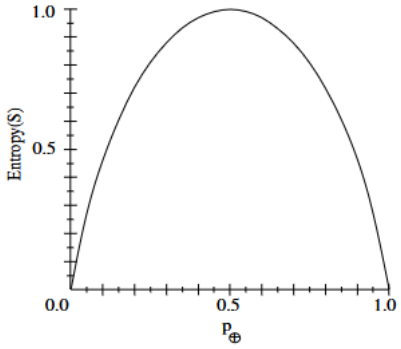
- $A \leftarrow$ the "best" decision attribute for next node to split examples
- Assign A as decision attribute for node
- For each value of A, create new child node
- Split training examples to child nodes
- if training examples perfectly classified, then stop. Else iterate over new child nodes.

9.2 Deciding on Splitting Attribute

We generally want to split by the attribute which results in the most "purity" in the child node. **Entropy** is used to measure the purity of a subset.

$$Entropy(S) = H(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

where S is the subset of training examples and $p_\oplus, p_\ominus$ are the portion of positive and negative examples in S.



9.3 Entropy

The optimal number of bits to encode a symbol with probability p is $-\log_2 p$. Suppose X can have one of $v_1, v_2, \cdots, v_k$,

$$P(X = v_1) = p_1, P(X = v_2) = p_2, \cdots, P(X = v_k) = p_k$$

Then on average, per symbol, the smallest number of bits to transmit a stream of symbols drawn from S's distribution is,

$$H(X) = -p \log_2 p_1 - p_2 \log_2 p_2 - \cdots - p_k \log_2 p_k \tag{34}$$

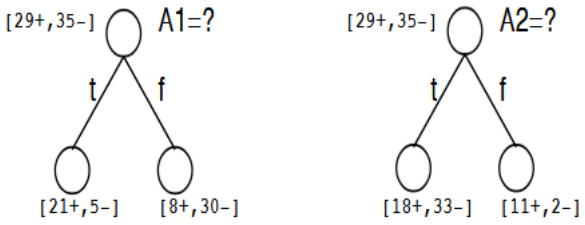
$$= -\sum_{j=1}^{k} p_j \log_2 p_j \tag{35}$$

where $H(X)$ is the entropy of X. So the expected number of bits to encode $\oplus$ or $\ominus$ of a randomly drawn member of a set S is,

$$p_\oplus(-\log_2 p_\oplus) + p_\ominus(-\log_2 p_\ominus)$$
$$Entropy(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus.$$

This is the Entropy that we want to minimise.

High entropy/impure set means X is very uniform and boring. Low entropy/pure set means X is not uniform and interesting.



$$H(S) = -\frac{29}{64}\log_2\frac{29}{64} - \frac{35}{64}\log_2\frac{35}{64} = 0.9936$$

9.4 Information Gain

The information gain $Gain(S, A)$ is the expected reduction in entropy due to sorting on A,

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where v is the possible values of attribute A, S is the set of examples we want to split and S_v is the subset of examples where $X_A = v$. We want to find the attribution which maximises the gain. For the previous example, the gain from splitting on different attributes is given by,

$$\begin{aligned} Gain(S, A1) &= Entropy(S) - (\frac{|S_t|}{|S|} Entropy(S_t) + \frac{|S_f|}{|S|} Entropy(S_f)) \\ &= 0.9936 - ((\frac{26}{64}(-\frac{21}{26}\log_2\left(\frac{21}{26}\right) - \frac{5}{26}\log_2(\frac{5}{26}))) + \\ &\quad (\frac{38}{64}(-\frac{8}{38}\log_2\left(\frac{8}{38}\right) - \frac{30}{38}\log_2(\frac{30}{38})))) \\ &= 0.9936 - (0.2869 + 0.4408) \\ &= 0.2658 \end{aligned}$$

$$\begin{aligned} Gain(S, A2) &= 0.9936 - (0.7464 + 0.0828) \\ &= 0.1643 \end{aligned}$$

So we chose $A1$ since it gives the larger expected reduction in entropy.

9.5 Gain Ratio

An issue with information gain is that it is biased towards attributes with large number of values/categories. The gain ration looks to fix this,

$$SplitEntropy(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

where,

- A: candidate attribute
- v: possible values of A
- S: Set of examples $\{X\}$ at the node
- S_v: subset where $X_A = v$

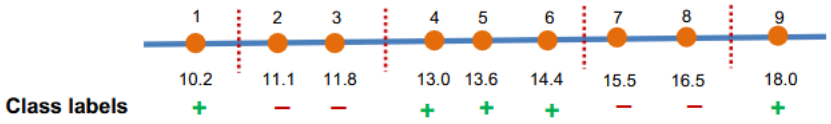
$$GainRatio(S, A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

9.6 Avoiding Overfitting

We can apply a method called **Pruning**. Pre-pruning is to stop growing when data split is not statistically significant. Post-pruning is to grow full tree, then remove sub trees which are overfitting (based on validation set).

9.7 Continuous Valued Attributes

Regular Decision Trees split on discrete attributes. To handle continuous attributes, we split the attribute on each boundary where the class changes. This can lead to $n - 1$ splits in the worst case if we don't know the class labels.



9.8 Attributes with Cost

Sometimes we need to evaluate information gain relative to cost,

$$\frac{Gain^2(S,A)}{Cosst(A)}$$

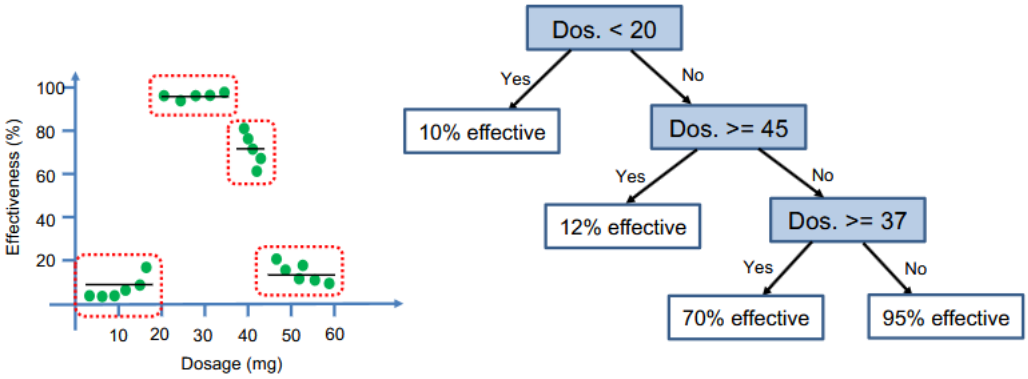
though this approach false positives a different cost to false negatives. We can define a loss function, $\lambda(\alpha_i|predicted\ class)$, and expected loss as,

$$R(\alpha_i|x) = \sum_{h \in H} \lambda(\alpha_i|predicted\ class)\ P(predicted\ class|x)$$

however if we use this loss function to split the tree then minimising R does not necessarily lead to the best long-term growth of the tree. So instead, we can grow the tree using information gain to its full depth and apply minimisation when post pruning.

10 Regression

Regression Trees are Decision Tress where each leaf performs a "mini Linear Regression".



Each leaf corresponds to average drug effectiveness in a different cluster of examples, the tree does a better job than *Linear Regression*

For splitting, the mean squared error is used for setting a threshholds,

$$MSE(Y_i) = \frac{1}{m} \sum_{j=1}^{m} (y_j - \overline{y})^2$$

where the MSE is equal to the variance of the examples in that subset. The weighted average of variance is,

$$weighted\ average\ gvariance = \sum_i^l \frac{|Y_i|}{|Y|} MSE(Y_i)$$

and we pick a threshold which minimises the weighted average of MSE/variance. That is, for each attribute, find the best split using the minimum weighted average variance and pick the minimum weighted average variance of those values.

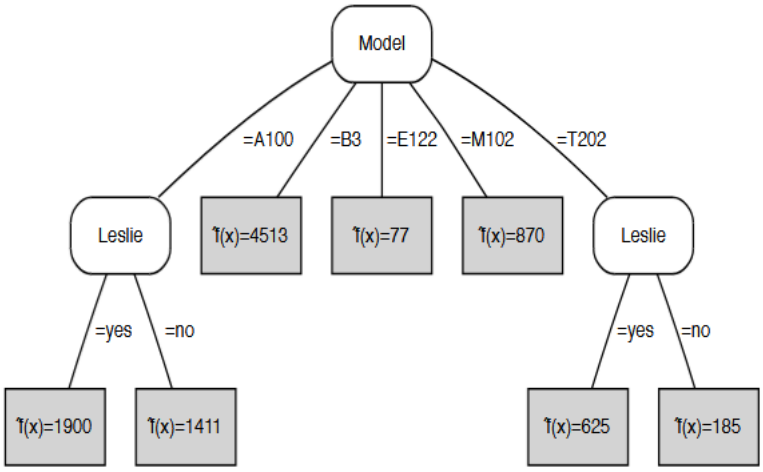
$$Model = [A100, B3, E112, M102, T202]$$
$$[1051, 1770, 1900][4513][77][870][99, 270, 625]$$

$$Condition = [excellent, good, fair]$$
$$[1770, 4513][270, 870, 1051, 1900][77, 99, 625]$$

$$Leslie = [yes, no]$$
$$[625,870,1900][77,99,270,1051,1770,4513]$$

- The means of the first split are 1574, 4513, 77, 870 and 331, and the weighted average of mean squared errors is 6.25×10^4.
- The means of the second split are 3142, 1023 and 267, with weighted average of mean squared errors 5.9×10^5;
- for the third split the means are 1132 and 1297, with weighted average of mean squared errors 1.72×10^6.

We therefore branch on $Model$ at the top level. This gives us three single-instance leaves, as well as three $A100$s and three $T202$s.

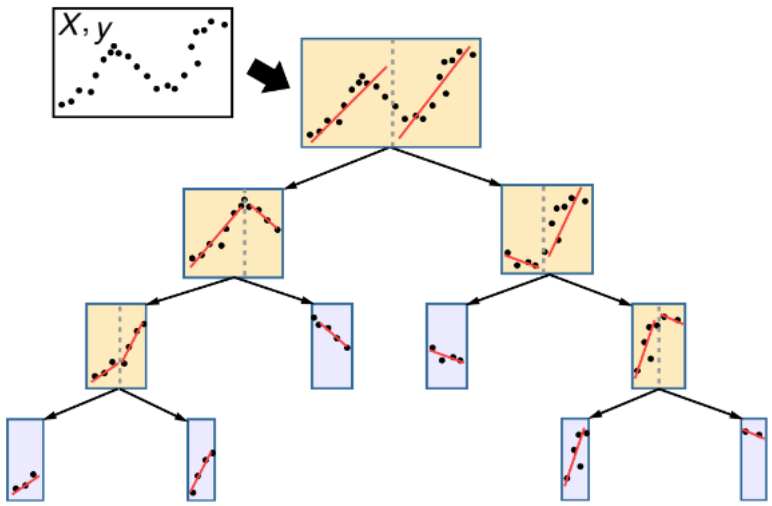


10.1 Overfitting in Regression Trees

Similar to Decision Trees, if we continue splitting all nodes that have a mean squared error bigger than zero, we will get a tree which fits the training data perfectly but will not generalise will to the test data. The simplest way to avoid this, similar to Decision Trees, is to only split examples when there is more than a minimum number, or using a maximum depth, or using a maximum number of leaves, etc.

10.2 Model Trees

Model Trees are similar to Regression Trees but with linear regression functions (or any model of your choice at each node) at each node. Linear Regression is applied to instances that reach a node after the full tree is built.



Model Trees use the splitting criteria of standard deviation reduction,

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} s \times sd(T_i)$$

where $T_1, T_2, \cdots$ are the sets from splits of data at node. Terminate splitting when standard deviation becomes smaller than certain fraction of sd for full training set or too few instances remain.