NOTES FOR THE INTRODUCTION TO MACHINE LEARNING

1. Some Definitions

1.1. **Variables.** Consider an example. Let X be a list of various branches of a company. We know how many employees are working at the particular branch, for how many years is the branch operating and the town/city where is the branch based. We would like to predict what is the annual profit of the branches.

In this example the number of employees, the number of the years and the location are input variables. The profit is an output variable. The input variables are usually denoted as X, output variables are usually denoted as Y. Input variables are also called predictors, independent variables or features. Output variables are sometimes called response or dependent variables.

The number of measurements (a size of the data) is usually denoted as n. In our example n will be the number of the branches of the company. The number of features is usually denoted as p. In our example $p = 3$ (the number of employees, the years of operation and the location of the particular branch).

Therefore, in general x is an $n \times p$ matrix x_{np}

1.2. **Expected value. Variance.** Let X be a random variable.

1.2.1. *Expected value for a finite case.* If we have outcomes $x_1, x_2..x_n$ with probabilities $p_1, p_2, .., p_n$ then the expected value will be

$$\mathbb{E}[X] = x_1p_1 + x_2p_2 + ... + x_np_n$$

If the outcomes are equally likely, then $p_1 = p_2 = .. = p_n = \frac{1}{n}$ and we get a simple average.

1.2.2. *Continuous Case.* Let X be a random variable with probability distribution $f(x)$. The probability distribution can be interpreted as a probability that the random variable has a particular value x. Or rather, if we have two values x_1 and x_2 the probability function tells us how much more(less) probable is the value x_1 comparing with x_2

The expected value for x is

$$\mathbb{E}[X] = \int xf(x)dx$$

1.2.3. *Variance.* For a given variable the variance is an expectation of the squared deviation of a random variable from its mean.

Therefore, if the mean of X is μ, then the variance is

$$Var[X] = \mathbb{E}[(X - \mu)^2]$$

1.2.4. *Discrete random variable.* If we have outcomes $x_1, x_2..x_i$ with probabilities $p_1, p_2, .., p_i$ then the expected value will be

$$Var[X] = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

where the mean is

$$\mu = \frac{1}{N}(x_1 + x_2 + .. + x_N)$$

1.2.5. *Continuous Case.* If X is a random variable with probability distribution $f(x)$, and the mean value μ, then the expected value will be

$$Var[X] = \mu^2 = \int (x - \mu)^2 f(x)dx$$

1.3. **Standard Deviation.** Standard Deviation is a measure of amount of deviation of the values. The smaller is the Standard Deviation, the closer the values are to the mean.

1.3.1. *Discrete random variable.* The Standard Deviation is

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

1.3.2. *Continuous Case.* If X is a random variable with probability distribution $f(x)$, and the mean value μ, then the Standard Deviation is

$$\sigma = \sqrt{\int (x - \mu)^2 f(x)dx}$$

2. Building the model

2.1. **Train-Test Split.** We usually randomly split available data into a train set and a test set. The train set is bigger, usually 80% of the available data. We train our model on the train set i.e., we try to find a model which fits the train data.

After that we use the obtained model and do predictions using X_{test} and compare the corresponding values available from the test set Y_{test}. Predicted values for the response are usually denoted as $\hat{Y}$. We try to get as smaller difference between Y_{test} and $\hat{Y}$.

If the difference is too big, that means that we have not built a good enough model. However we do not want to overfit the data: it can happen that the model fits the data too well, does not take into account the statistical noise etc. In this case it will not perform well when we change the data-points.

Therefore, we have to take into account the Bias-Variance Trade off

3. Bias-Variance Trade-off

3.1. **A Short Definition.**

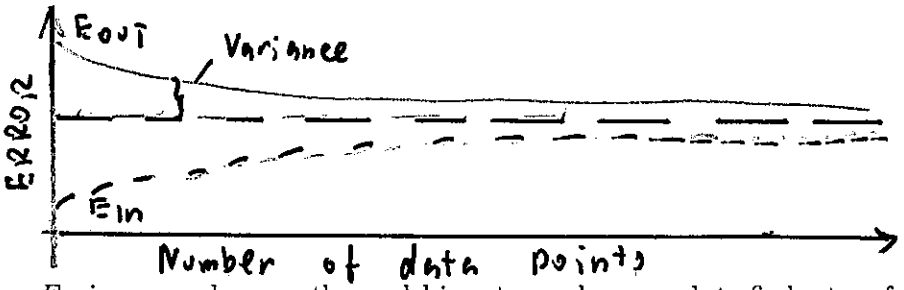
3.1.1. *Bias.* In order to fit the data we build some model (a function) $\hat{f}(x)$. The simple model has higher bias, it is less flexible. For example, the straight line has high bias. More flexible models approach train data point more closely and they have lower bias. Therefore bias refers to the error that we are introducing by approximating a complex data by a simpler model.

3.1.2. *Variance.* This is the measure of how much the function $\hat{f}(x)$ will change if we use another sample (training set) to estimate it

The bigger is the variance the smaller is the bias and vice versa.

3.2. **In more detail.**

- E_{in} - in sample error (training error)
- E_{out} - out of sample error



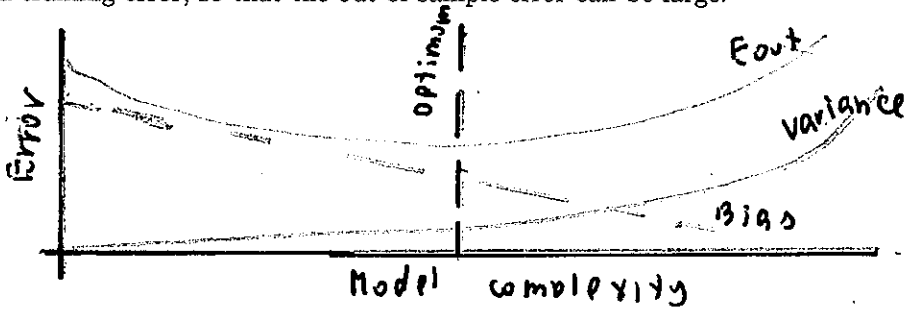
- E_{in} increases, because the model is not complex enough to find a true function.
- E_{out} will decrease

If we had an infinite data, we could beat the noise. The bias is the best our model can do to beat the noise. More complex model has less bias.

It is better to minimise E_{out}, rather than to minimise the bias, because we do not have an infinite data.

Variance measures how the outcome of the model will change if we change the sample. More it changes, bigger is the variance. In other words, variance measures errors we introduce by finite sampling.

Difference between E_{in} and E_{out} measures the difference between fitting and predicting. If the difference is big, we call the model "overfit". In general, one does not aim to simply reduce a training error, so that the out of sample error can be large.



- Consider a data set $D = (X, y)$ consisting of N independent pairs of independent X and dependent y variables. Assume that the data is generated by a model

$$y = f(x) + \epsilon$$

here ϵ is a noise.

- Let θ be a set of the parameters, which are present in the model.
- Assume that we have a statistical procedure to determine the function f. It can be done by minimising the cost function (the error). for example minimising

$$C(y_i, f(X, \theta)) = \sum_i (y_i - f(x_i; \theta))^2$$

- From this procedure we determine the parameters $\hat{\theta}_D$, they are naturally functions of the dataset D. This is because we would have obtained a different error for the different data set. In this way we form a predictor $f(x; \hat{\theta})$ which gives a prediction for a new data point.
- Let us denote an expectation value of all datasets as $\mathbb{E}_D$. The expectation value over the noise is denoted as $\mathbb{E}_\epsilon$.

We have

- The Bias

$$(Bias)^2 = \sum_i (f(x_i) - \mathbb{E}_D(f(x, \hat{\theta}_D)))^2$$

measures the deviation of the expectation value of the estimator $\mathbb{E}_D(f(x, \hat{\theta}_D))$ from its true value $f(x_i)$

- The variance:

$$Var = \sum_i \mathbb{E}_D \left[(f(x, \hat{\theta}_D) - \mathbb{E}_D[f(x, \hat{\theta}_D)])^2 \right]$$

tells us how much the estimator $f(x, \hat{\theta}_D)$ fluctuates due to the final size effects.

- Out of sample error is a sum of three terms

$$E_{out} = (Bias)^2 + (Var) + Noise$$

- The noise is

$$\sum_i \sigma_\epsilon^2 = \mathbb{E}_\epsilon[(y_i - f(x_i))^2]$$

More complex is the model, it has a lower bias but higher variance. When increasing the bias the variance decreases and vice versa.

4. Gradient Descent

4.1. **Newton method.** We would like to minimize the cost function $E(\theta)$, where θ are the parameters.

In the gradient descent (GD) method we initialise the parameters at some value θ_0 and then iteratively update the parameters according to

$$v_k = \eta_k \nabla_\theta E(\theta_k) \tag{1}$$

$$\theta_{k+1} = \theta_k - v_k \tag{2}$$

Here η_t is a learning rate. It indicates how big a step we should take in the direction of the gradient. If we take the learning rate to be too small, the algorithm will take too long to converge. On the other hand, too big learning rate can lead to the situation when we skip over the local minimum.

A possible approach to this problem is a Newton method. We shall consider one dimensional case for simplicity.

Consider the Taylor expansion

$$E(\theta_k + v_k) = E(\theta_k) + E'(\theta_k)\, v_k + \frac{1}{2} E''(\theta_k)(v_k)^2 + \ldots$$

The next iterate θ_{k+1} is defined by minimizing the quadratic approximation in v_k. To this end we take the derivative with respect to v, and since at the minimum the derivative is zero we get

$$0 = E'(\theta_k) + E''(\theta_k) v_k, \quad therefore \quad v_k = -\frac{E'(\theta_k)}{E''(\theta_k)}$$

Then we update the parameter θ_k as (2).

In the multidimensional case (that means when we have many parameters, and this is what happens in general) v becomes a vector v^i and we get a matrix of second derivatives $\nabla_{\theta_i} \nabla_{\theta_j} E(\theta_c)$, called Hessian. Therefore, we update the values of v_t according to

$$v_{t,i} = -(\nabla_{\theta_i} \nabla_{\theta_j} E(\theta_t))^{-1} \nabla_{\theta_j} E(\theta_t)$$

However the Newton method is difficult to implement in practice. Indeed when we have many parameters, the Hessian is difficult to compute, and to invert.

Let us define the optimal choice of the learning rate.. For the given value of θ the optimal choice of the learning rate $\eta_{opt.}$ is the value of η which allows to reach the minimum of $E(\theta)$ in a single step. To find it we expand $E(\theta)$ around the current value up to the second order in v, as we have done above

Then we differentiate with respect to v and put $\theta_{min} = \theta + v$. This way we get

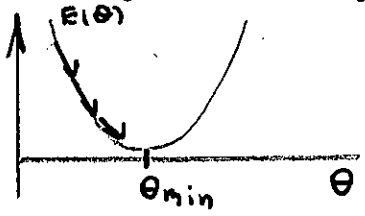
$$\theta_{min} = \theta - \frac{E'(\theta)}{E''(\theta)}$$

and

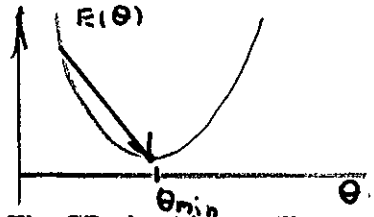
$$\eta_{opt} = \frac{1}{E''(\theta)}$$

There are four possibilities

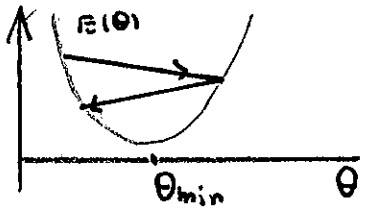
- $\eta < \eta_{opt.}$. The GD algorithm takes multiple steps to reach the minimum.



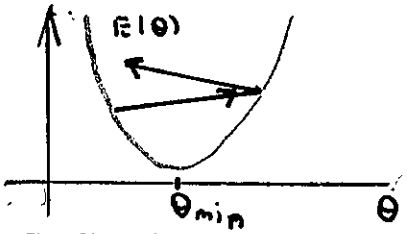
- $\eta = \eta_{opt.}$ The GD algorithm reaches the minimum in a single step.



- $\eta < 2\eta_{opt.}$. The GD algorithm oscillates and eventually converges to the minimum.



- $\eta > 2\eta_{opt.}$. The GD algorithm diverges.



4.2. **Stochastic Gradient Descent.** In this approach the data set is divided into subsets called mini batches. If the data set consists of n points and a mini batch consists of M points,

then we have $k = \frac{n}{M}$ mini batches B_k. At each gradient descent step we approximate gradient using a single mini batch B_k. We then cycle over k mini batches, one at the time and update the parameters θ at every step k. The full iteration over all k mini batches is called an epoch.

5. Bayes Rule

The conditional probability is denoted as $P(A|B)$. It is a probability of happening of the event A, under the condition that the event B happened.The probability that the event A will happen is denoted as $P(A)$.

The Bayes rule reads

$$P(A|B)P(B) = P(B|A)P(A)$$

- The likehood function

$$P(X|\theta)$$

  is a probability of observing the data set Xgiven the parameters θ.
- Prior distribution is a knowledge we have about parameters before collecting the data $P(\theta)$
- Posterior distribution $P(\theta|X)$ is a knowledge of parameters **after** measuring the data

Using the Bayes rule

$$P(\theta|X) = \frac{P(\theta|X)P(\theta)}{\int d\theta' P(X|\theta')P(\theta')}$$

we compute the posterior distribution in terms of the likehood function and the prior distribution.

The denominator $\int d\theta' P(X|\theta')P(\theta') = P(X)$.

The likehood function $P(X|\theta)$ is determined by the model and by the noise.

We would like to maximise $P(X|\theta)$. It is called MLE (Maximum Likehood Estimation). In MLE we choose the parameters in such a way that they maximize the likehood of the observed data.

If no information about θ before we look at the data is available, we select $P(\theta)$ so that it reflects our ignorance about the data. This kind of $P(\theta)$ is called an informative prior.

6. Linear regression

6.1. **Definition.** A simple linear regression assumes a linear dependence between independent and dependent variables

$$Y = \beta_0 + \beta_1 X$$

where β_0 and β_1 are coefficients.

Assume we have data points (x_1, y_1), (x,y_2),...(x_n, y_n). This is a sample we obtained from our measurement. We would like to find the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$, which will help us to make predictions by using the Linear Regression.

Let us define the Residual Sum of Squares (RSS) as

$$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + .., + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

The coefficients β_0 and β_1 are determined by minimizing RSS, i. e., solving the equations

$$\frac{\partial}{\partial \beta_0} RSS = \frac{\partial}{\partial \beta_1} RSS = 0$$

The solution is

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

are average values for x_i and y_i i.e., the sample means.

6.2. **Assessment of the accuracy.** Usually a sample mean is a good estimate of a measurement outcome. Let us have a single outcome $\hat{\mu}$. We would like to know how good it is as en estimate of our variable. the answer is provided by a Standard Error (SE)

$$SE(\hat{\mu}) = \sqrt{Var(\hat{\mu})} = \frac{\sigma^2}{n}$$

where σ is a standard deviation for y_i. Apparently, more measurements we have, the bigger is n and the smaller is the Standard Error.

In the same way, if we would like to know how close are the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ to the actual values β_0 and β_1, we compute the Standard Error for them

$$SE(\beta_0) = \sigma^2 \left(\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right)$$

$$SE(\beta_1) = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

these equations are correct if errors associated with each measurement are uncorrelated with the common variance. But often they provide a good estimate anyway. The common variance $\sigma^2 \sim Var\, \epsilon$ can be determined from data, using the Residual Standard Error

$$RSE(\hat{\mu}) = \sqrt{\frac{RSE}{n-2}} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$$

It shows how average response will deviate form the true regression line. It is a measure of a lack of fit.

6.3. **Hypothesis test.** Standard error can be used also to provide hypotheisis tests.

- Null hypothesis $\mathcal{H}_0$: there is no relationship between X and Y
- Alternative hypothesis $\mathcal{H}_a$: there is a relationship between X and Y

In the case of a simple Linear Regression

- $\mathcal{H}_0$ means that $\beta_1 = 0$
- $\mathcal{H}_a$ means that $\beta_1 \neq 0$

To test the Null Hypothesis we need to determine, if β_1 is sufficiently far from zero. In other words if $SE(\hat{\beta}_1)$ then $\hat{\beta}_1$ must be large to reject the null hypothesis.

In practice we compute t statistic, given by

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{\sigma^2} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

This is a probability of observing any value equal to $|t|$ or larger assuming that $\beta_1 = 0$ and is called p value. Large p value means that any correlation between independent and dependent variables is due to the chance.

In general a small p value means that there is a strong evidence in favour of an alternative hypothesis. In our case an alternative hypothesis is that Y depends on X.

6.3.1. R^2 *statistics.* if we would like to asses the accuracy of the model the RSS can be less informative. It gives us a number but sometimes it is difficult to say a particular number is big or not. (Big/small comparing to what?)

A better assessment can be performed using R^2. Let us define R^2 as

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where Total Sum of Squares (TSS) is

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

is an amount of variability **before** the regression is performed. RSS is an amount of variability left unexplained. Therefore, $(TSS - RSS)$ is an amount of variability that is explained by the regression. And R^2 is a portion of variablity that is explained by the model. Closer it is to 1, the better is the model.

7. Multiple Linear Regression

In the Multiple Linear Regression we have several estimators

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_P + \epsilon$$

similarly to how it happened in the simple linear regression, in the multiple linear regression the parameters $\beta_0, \beta_1, ..\beta_p$ are estimated by minimizing of the RSS

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - ... - \hat{\beta}_p x_{ip})^2$$

7.1. **Improved R^2.** In the Multiple Linear Regression each time we add a new variable R^2 either increases or stays the same. If the new variable is completely irrelevant for the response then R^2 stays the same. But if the new variable is not relevant for the response and yet the algorithm will find some some accidental correlation between this variable and the response, then R^2 will increase.

In order to prevent R^2 increasing by simply adding of a new variable, us introduce

$$F^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

Here again, n is a sample size, and p is a number of regressors.

On the other hand F^2 has a penalising factor $n - p - 1$. Therefore as p increases, F^2 decreases. And there is a competition between increasing of F^2 because of adding an a new variable on one side, and the penalising factor on the other side.

7.2. **Categorical Variables.** Categorical variables are the ones which do not have a numerical value. For example a geographical or a proper name, a job title etc. They can be included into regression models by using of so called dummy variables.

For example let y be an amount of profit that a particular brunch of a company makes per year. Let us suppose the company has offices in Tokyo and Osaka. We introduce two dummy variables D_1 and D_2, one for Tokyo and one for Osaka. If the branch is based in Tokyo, then $D_1 = 1$ and $D_2 = 0$. If a company is based in Osaka, then $D_1 = 0$ and $D_2 = 1$. Then one can write a multiple linear regression model as

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \gamma_1 D_1 + \gamma_2 D_2$$

where $x_1, .., x_p$ are the other (non-categorical) predictors. However, since

$$D_1 + D_2 = 1$$

one has to exclude one categorical variable from the regression, since the knowledge of D_1 determines D_2 and therefore both of them can not be considered as independent variables. This is called the dummy variable trap. The a general rule is: if one has m categories, one has to include $m - 1$ dummy variables.

7.3. **Feature Scaling.** Sometimes different features have different numerical scales. For example, X has two features: the number of employees in the particular branch of the company, and number of the years the branch operates. The first number can be thousand(s), the second one can be from 1 to 10.

In order the algorithm to perform better we need to bring them to a similar scale. There are two ways to do so

- Min - max scaling: We subtract minimal value of the feature and divide by $(min - max)$. In this way the features will get values for 0 to 1.
- Standardization: First we subtract the mean value, and then divide by standard deviation.

7.4. **Ridge Regression.** Ridge regression minimises

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

Here $\lambda \geq 0$ is a tuning parameter. It tries to make the coefficients β_j as small as possible and apparently controls the relative impact of the two terms in the expression above. When λ increases the flexibility of the model decreases.

7.5. **Lasso Regression.** LASSO regression is similar to the Ridge Regression. In this approach one tries to find the parameters β_j that minimize the expression

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

8. Logistic regression

Let us consider an example. A bank would like to predict a probability of a customer will default on their credit, depending on the customer's annual income. In this example an independent variable X is the annual income. The dependent variable is a binary outcome - Yes (the customer defaults) or No (the customer will not default.) One encode Y encoded as 1/0 (1 for "No" and 0 for "Yes"). One can model this problem as a Linear Regression

$$Y = \beta_0 + \beta_1 X$$

and say that if Y is higher then a chosen threshold then the answer will be "Yes", otherwise it will be "No".

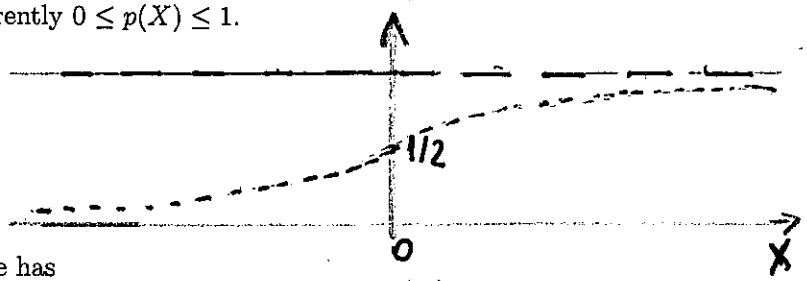
An application of Linear Regression for this problem has apparent drawbacks. For some values of X the Linear Regression Model can predict:

- Negative probability
- Probability which is more than 1

In order to avoid these problems, let us consider a probability $p(X)$ which depends on X as

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} = \frac{1}{1+e^{-\beta_0-\beta_1 X}}$$

Apparently $0 \leq p(X) \leq 1$.



$$g(x) = \frac{1}{1+e^{-x}}$$

One has

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

or

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

The left hand side of the equation is called odds. It can take values from zero to infinity.

Increase of X by one unit multiplies odds by e^{β_1}. Unlike the Linear regression, the bigger is X the faster is the increment of the odds (a derivative of the odds at each point X depends on the value of X).

For binary classification models we have two classes: $y = 0$ and $y = 1$. We can assign the outcome as

$$y = \begin{cases} 0 & \text{if } p < 0.5 \\ 1 & \text{if } p \geq 0.5 \end{cases}$$

Obviously, the choice of the threshold depends on the nature of the problem at hand. For medical problems is usually much higher then 0.5.

8.1. **Cost Function.** Cost function for a single training instance

$$c(\beta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1-\hat{p}) & \text{if } y = 0 \end{cases}$$

The model estimates high probabilities for positive instances ($y = 1$) and low probabilities for negative instances ($y = 0$).

Indeed, let us take $y = 1$. Then for $p = 1$ we have $c = 0$ and the cost function is small. On the other hand for $p = 0$ we have $c = \infty$ and the cost function is very high.

If we take $y = o$. Then for $p = 0$ we have $c = \infty$ and the cost function is high. On the other hand for $p = 0$ we have $c = 0$ and the cost function is small. Therefore it is a good cost function.

The cost function for whole training set is an average over all training instances

$$C(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\log\hat{p}^{(i)} + (1-y^{(i)})(1-\log\hat{p}^{(i)})\right)$$

This expression is called cross- entropy.

8.2. **Multiple Logistic Regression.** If we have several predictors and a binary outcome, then the simple logistic regression generalises to

$$p(x) = \frac{1}{1 + e^{-\beta_0 - \beta_1 X_1 + ... - \beta_p X_p}}$$

and

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

9. Softmax Regression

Softmax Regression is a generalization of the Linear regression. It is applied when the outcome contains multiple classes, i.e., it is not just a binary outcome

One computes a Softmax score for class k

$$s_k(x) = \beta_0^{(k)} + x_1\beta_1^{(k)} + ... + x_p\beta_p^{(k)}$$

Notice that the coefficients $\beta_0^{(k)}, \beta_1^{(k)} ..., \beta_p^{(k)}$ are in general different for different classes.

Then one finds the relevant probability

$$\hat{p}_k = \frac{e^{s_k(x)}}{\sum_{j=1}^{K} e^{s_j(x)}}$$

Here K is a number of classes The outcome for the given instance will be the class with maximal $\hat{p}_k$.

10. Bayes Classification

Let us consider a classification problem. Suppose we have training observations $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$

To quantify the accuracy of our estimate we introduce the Error rate

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}_i)$$

This is a training error rate and is a proportion of mistakes The function I is defined as

- $I(y_i \neq \hat{y}_i) = 1$
- $I(y_i = \hat{y}_i) = 0$

The test error rate for test observations (x_0, y_0) is given by $Ave(I(y_0 \neq \hat{y}_0))$ where $\hat{y}_0$ is the predicted class label which we obtain when we apply our classifier to the test instance x_0.

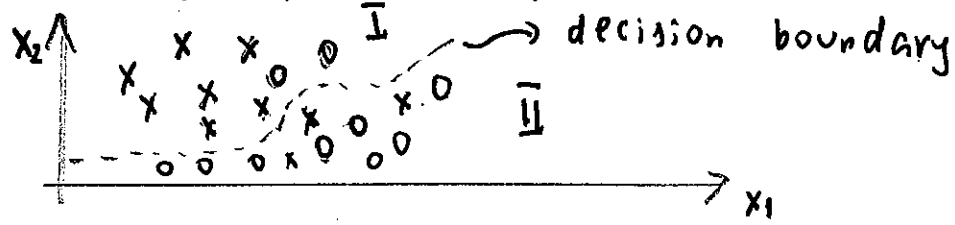
One tries to make $Ave(I(y_i \neq \hat{y}_i))$ as small as possible. It is minimized by if the outcome for x_0 is assigned the class $Y = j$ for which the conditional probability

$$Pr(Y = j | X = x_0)$$

is the largest. This is called the Bayes classifier.

Consider the Bayes Classifier when the outcome consists of only two classes. If $Pr(Y = 1|X = x_0) > 0.5$, then the outcome is assigned to the class one, otherwise assigned to the class two.

Here is an example of two predictors X_1 and X_2. The observations belong to two classes and are denoted as by $\times$ and 0. In the area I the probability for an observation being $\times$ is more than 0.5. In the area II the probability for an observation being $\times$ is less than 0.5. The two areas are divided by the Bayes decision boundary.



The Bayes decision boundary is defined by $Pr(Y = j|X = x_0) = 0.5$ The Bayes classifier has the smallest possible test error rate. At $X = x_0$ the Error rate is

$$1 - max_j Pr(Y = j|X = x_0)$$

In general, the overall error rate for the Bayes Classifier is

$$1 - \mathbb{E}\left(max_j Pr(Y = j|X = x_0)\right)$$

where the average is over all possible values of X.

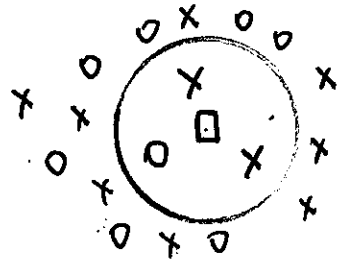
11. K Nearest Neighbours

Given a possible integer K a test observation x_0 the K Nearest Neighbours (KNN) classifier identifies K points in the training data around x_0. Let us denote them as $\mathbb{N}_0$. Then it estimates the conditional probability

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathbb{N}_0} I(y_i = j)$$

as a fraction of points in $\mathbb{N}_0$ whose response values are equal to j

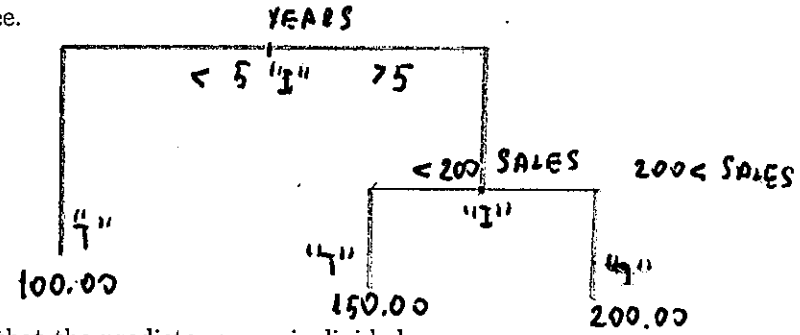
11.1. **Example,** $K = 3$. Let us consider an example.



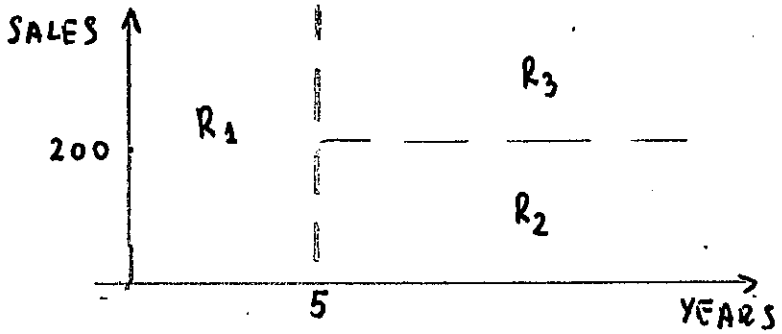
Apparently among three neighbours of the test observation $\square$, there are two that belong to the class $\times$ and one that belongs to the class $\circ$. Therefore, $\square$ is assigned to $\times$ class.

12. Decision Trees

12.1. **Regression.** Let us consider an example: We would like to estimate a salary of a salesperson according to the years of their experience, and sales they made. Their salaries are determined as follows: If they have less than five years of experience, they belong to the category A and their salary will be 100.000,00 dollars per year. If they have more then five years of experience, they belong to the category B. This category is further divided as follows: If they made more than 200 sales per year, their salary is 200.000,00 per year. If they made less than 200 sales per year, their salary is 150.000,00 per year. This model can be described by a decision tree.



One can see, that the predictor space is divided as



- Here "T" denotes Terminal Nodes
- Here "I" denotes Internal Nodes
- Segments, that connect nodes are called branches

Apparently, some features are more important than others. In our example the years of experience are more important, than the sales made- if the experience is less then 5 years, the salary is determined, no matter how many sales are made.

A general approach:

- One divides a predictor space into non overlapping regions $R_1, R_2, .., R_j$.
- To every observation that falls into a region R_j one assigns the same prediction, which is a mean response for the training observation in R_j.

We divide the feature space into boxes $R_1, R_2, .., R_j$, so that RSS is minimal

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Each split produces two new branches (a binary split).

We perform the best split at each particular step $\{X|X_j < s\}$ and $\{X|X_j > s\}$, so that we get the best reduction in RSS. We consider all possible predictors X_p and all possible s and choose the relevant feature X_j and the value of s so that RSS is minimal.

For any j and s we define a pair of half planes

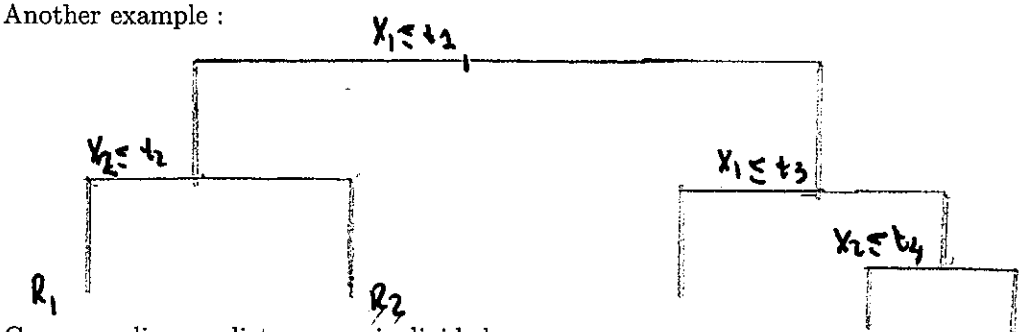
$$R_1(j,s) = \{X|X_j < s\}, \quad R_2(j,s) = \{X|X_j < s\}$$

and seek for the values of j and s so that minimize the expression

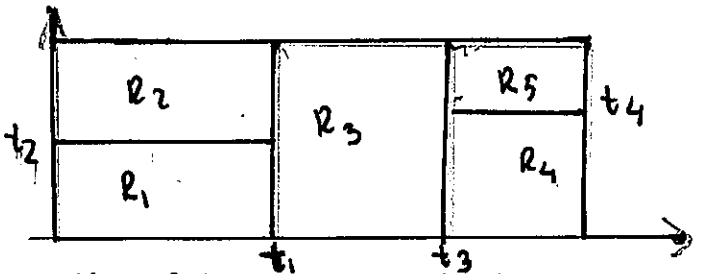
$$\sum_{i:x_i \in R_1(j,s)} (y - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y - \hat{y}_{R_2})^2$$

$\hat{y}_{R_1}$ is a mean response for the training observations in $R_1(j,s)$ and $\hat{y}_{R_2}$ is a mean response for the training observations in $R_2(j,s)$. Therefore, instead of splitting the entire predictor space, we split one of the two previously identified regions.

Another example :



Corresponding predictor space is divided as



In order to avoid over fitting, one can proceed as follows. Start with building a large tree T_0 and then consider a sequence of trees (sub trees), indexed by a parameter α. For each

value of α we have a sub- tree $T \subset T_0$, such that

$$\sum_{m=1,i}^{|T|} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ is a number of terminal nodes in the tree T. As α grows we have less terminal nodes.

12.2. **Classification.** The classification trees are very similar to regression trees. As usual in classification problems we predict a qualitative response. Therefore, an observation which belongs to a particular box is assigned to the most common class for the training response in this box.

To measure a classification error one usually uses either the Gini index or cross entropy. Suppose we have K classes (outcomes). We divided our predictor space into the regions

- The Gini index
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$
where $\hat{p}_{mk}$ is a proportion of training observations in the m region that are from k class.
- Cross entropy
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \ln \hat{p}_{mk}$$

Since $0 \leq \hat{p}_{mk} \leq 1$, one has $0 \leq D$. Apparently D is close to zero if $\hat{p}_{mk}$ is close to zero or close to one. This means that the mth region (node) is "pure".

13. Random Forests

At Random Forrest algorithm one considers as subset of m elements and not all predictors, before performing a split. each time the split is performing by using only one out of these m predictors.

The reason behind this as follows: Suppose among the predictors, there is one strong predictor and a few moderately strong ones. Then each time the split will be performed by using the one strong predictor as a top split. Then all possible trees will look the same.

14. Support Vector Machines

14.1. **A hyperplane.** On a two dimensional plane with coordinates X_1 and X_2 one can define a line using the following equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

Let us generalise this to p dimensions. In a p- dimensional setting a similar equation

$$\beta_0 + \beta_1 X_1 + ... + \beta_p X_p = 0$$

defines a $p-1$ dimensional hyperplane.

If a point with coordinates $(X_1, X_2, ..X_p)$ satisfies

$$\beta_0 + \beta_1 X_1 + ...\beta_p X_p > 0,$$

then the point lies on one side of the hyperplane.

If a point with coordinates $(X_1, X_2, ..X_p)$ satisfies

$$\beta_0 + \beta_1 X_1 + ...\beta_p X_p < 0,$$

then the point lies on the other side of the hyperplane.

14.2. **Classification.** The data X_{ij} are represented is $n \times p$ data matrix, with n training observations and p features

$$x_1 = \begin{pmatrix} x_{11} \\ ... \\ x_{1p} \end{pmatrix}, \quad ..., \quad x_n = \begin{pmatrix} x_{n1} \\ ... \\ x_{np} \end{pmatrix}$$

Suppose that the observations fall into two classes, i.e., $y_1, y_2, .., y_n \in \{-1, 1\}$

The test observation is

$$x^* = (x_1^*, ..., x_p^*)$$

Suppose we can construct a hyperplane, that separates the classes perfectly

$$\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} > 0, \quad if \quad y_i = 11$$

$$\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} > 0, \quad if \quad y_i = -1$$

This means that the separation has a property

$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) > 0$$

for all $i = 1, .., n$.

14.3. **A classifier.** A test observation is assigned a class, according on which side of the hyperplane it is located. The magnitude of

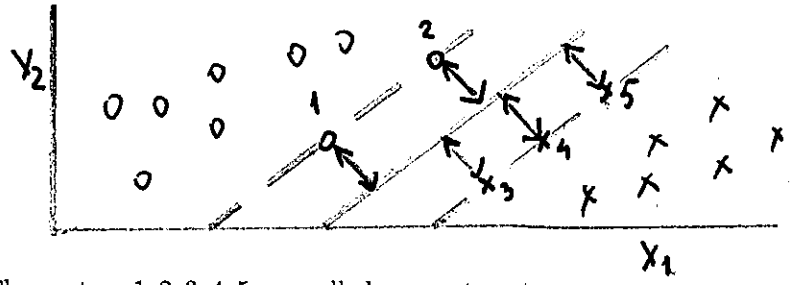
$$f(x^*) = \beta_0 + \beta_1 x_1^* + ... + \beta_p x_p^*$$

defines how far form the hyperplane the observation is located. The bigger is the magnitude, more confident we are, that its class is correctly identified.

If data points are perfectly separated there are infinitely many such dividing hyperplanes.

14.4. **Maximal Margin Hyperplane.** Let us compute a perpendicular distance of each training instance to the hyperplane. The smallest such distance is called **margin**. Maximal Margin Hyperplane is a hyperplane for which the margin is **maximal**.

We can classify a test observation according to on which side of the Maximal Margin Hyperplane it is. This is called **Maximal Margin Classifier**.



The vectors $1, 2, 3, 4, 5$ are called support vectors.

14.5. **How to construct.** We consider an optimization problem:

- We have $x_1, ..., x_n \in \mathbb{R}^p$ training observations and the class labels are $y_1, ..., y_n \in \{-1; 1\}$
- We maximize the margin M by choosing the parameters $\beta_0, \beta_1, ...\beta_p$
- Subject to constraints

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

- and

$$y_i(\beta_0 + \beta_1 x_{i1} + .. + \beta_p x_{ip}) \geq M$$

for all $i = 1, ..., n$

It can be shown, that the perpendicular distance form i th observation to the hyperplane is given by

$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip})$$

14.6. **Support Vector Classifier.** Support Vector Classifier is a generalization of the Maximal Margin Classifier to the Case of non-separable data. We consider an optimization problem:

- We have $x_1, ..., x_n \in \mathbb{R}^p$ training observations and the class labels are $y_1, ..., y_n \in \{-1; 1\}$
- We maximize the margin M by choosing the parameters $\beta_0, \beta_1, ...\beta_p$ and $\epsilon_1, ..., \epsilon_i$
- Subject to constraints

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

- and

$$y_i(\beta_0 + \beta_1 x_{i1} + .. + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

for all $i = 1, ..., n$

Here

- $C \geq 0$ is a training parameter
- M is a width of the margin
- The variables $\epsilon_1, ..., \epsilon_i$ are subject to

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad and \quad \epsilon_i \geq 0$$

that allow an individual observation to be on a wrong side of the dividing hyperplane

We classify the test variable according to the sign of

$$f(x^*) = \beta_0 + \beta_1 x_1^* + ... + \beta_p x_p^*$$

- The parameter C determines a number and severity of violation of the margin
- If $C = 0$, there is no budget for violation.
- There are no more than C violations allowed
- Small C means low bias and high variance

14.7. **Non-linear decision boundaries.** For non-linear decision boundaries the situation is very similar to the linear ones, but now we have

- For all $i = 1, ..., n$

$$y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \geq M(1 - \epsilon_i) \right)$$

- Subject to

$$\sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1, \quad \sum_{i=1}^{n} \epsilon_i \leq C, \quad and \quad \epsilon_i \geq 0$$

14.8. **Kernel.** The solution of support vector classifier depends only on inner products. Inner product is defined as

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

Linear support vector classifier is represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

We have $\binom{n}{2}$ inner products.

Each time when the inner product appears we replace it with

$$K(x_i, x_{i'})$$

which is a function of the inner product. This function is called a Kernel

- For support vector classifier

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

- One of the most popular Kernels is the radial Kernel

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$

  where γ is a parameter.

The kernel has a local behaviour: only the nearby points contribute to the decision on to which class the observation belongs.

15. K-means clustering

15.1. **Supervised vs. Unsupervised Learning.** Suppose we have a data from supermarket: customers who bye a tea of a bran A usually bye a cake of a brand $\mathcal{A}$ and a mineral water of brand $\overline{A}$. Customers who bye a tea of a brand B usually bye a cake of a brand $\mathcal{B}$ and mineral water of brand $\overline{B}$. This is an example of unsupervised learning. Unsupervised learning is often a part of exploratory data analysis.

- Supervised learning: For each predictor x_i we have a response y_i
- For unsupervised learning there is no response y_i

15.2. **Clustering.** The main purpose of clustering is to figure out on the basis of $x_1, ..., x_n$ whether the observations fall into relatively distinct groups.

In K means clustering we partition the observations into pre - specified number of clusters Let $C_1, ..., C_K$ be sets containing indices of each cluster. We have

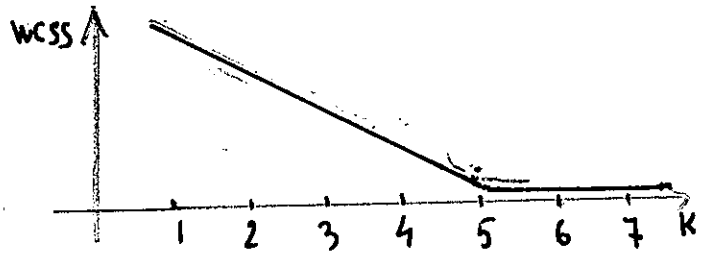
- $C_1 \cup C_2 \cup, ..., \cup C_K = 1, ..., n$ i.e., each observation belongs to one of the clusters
- $C_K \cap C_{K'} = \emptyset$ i.e., the clusters are not overlapping.

We would like to partition the data in such a way that Within Cluster Variation $W(C_k)$

$$W(C_K) = \frac{1}{|C_K|} \sum_{i,i' \in c_K} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

is minimal. Here C_k is a number of observations in the cluster.

15.3. **Elbow method.** We would like to determine the best number of clusters. One possibility is to plot within the cluster variation or within the cluster sum of squares (WCSS- the same as the within the cluster variation, without dividing by C_k) versus the number of clusters. More clusters we have the less WCSS is. Apparently, the minimal value is zero, when each data point becomes its own cluster and that is not what we need. We use the elbow method. For example, suppose the plot looks like:



In this example the best number of clusters is $K = 5$, because at this value of K the plots looks like an elbow.

16. Agglomerative Hierarchical Clustering

16.1. **The algorithm.** In hierarchical clustering we determine the clusters using so called dendograms.

The algorithm goes as follows

- Make each point as an individual cluster
- Take two closest data points and make them one cluster
- Take two closest clusters and make them one cluster
- Repeat the third step, until entire data will form one cluster

The question "How we define the distance between clusters" can be answered in different ways.

16.2. **Linkage.** The distance between the clusters can be defined in a different way. First, we understand a "distance" as and Eucledean distance. For example, the distance between two points with coordinates (x_1, y_1) and (x_2, y_2) is

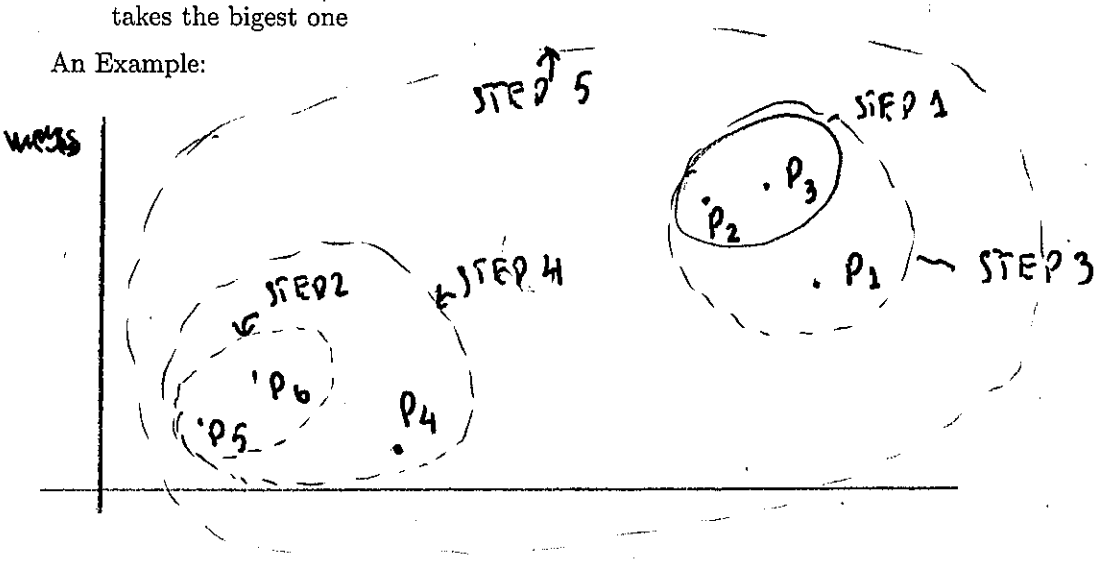
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The distance between clusters can be a distance between:

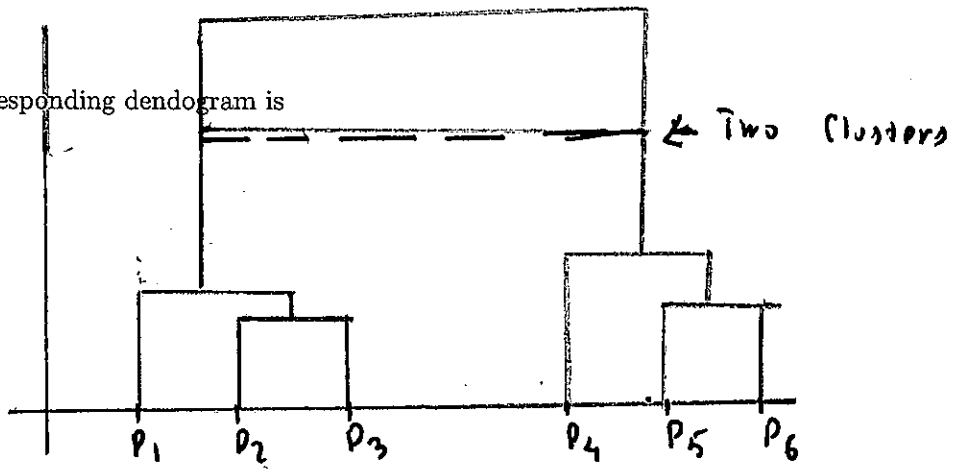
- Centroid: A distance between centroids (which is analogous to the center of mass of the system in physics)
- Average: One computes all pairwise distances in the cluster A and cluster B and then takes average
- Average: One computes all pairwise distances in the cluster A and cluster B and then takes the smallest one

- Average: One computes all pairwise distances in the cluster A and cluster B and then takes the bigest one

An Example:



Corresponding dendogram is



Vertical lines represent distance between clusters.

16.3. **Number of clusters.** One way to obtain the best number of clusters is as follows:

- We extend all horizontal lines
- Choose the tallest vertical line. The number how many times it is crossed by any horizontal line is the best number of clusters

The number of clusters is two in the example above.

16.4. **Issues with clustering.** .

With K-means:

- How many clusters to choose

With the Hierarchical Clustering:

- What is the dissimilarity measure?
- What is the linkage?
- Where to cut dendograms?

With both

- Should we standardize observables or features in some way?

17. Principal Component Analysis

Usually the data contains many features. One can say, that the variables live in a p-dimensional space. But not all of these features are equally interesting: the output does not vary equally along all features. Therefore, we choose the components (called principal components) along which the variance is the biggest, i.e., we perform the dimensional reduction.

The first principal component is defined as a normalized linear combination:

$$Z_1 = \phi_{11}x_1 + \phi_{12}x_2 + ... + \phi_{1p}x_p, \quad with \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

In order to find a principal component, we solve the optimization problem. Let us assume that each component of X has a mean zero We write all possible linear combinations

$$Z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}X_{ip}, \quad with \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

and then find such ϕ_{j1}, so that the variance

$$\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2$$

is maximal. The coefficients ϕ_{j1} tells us what "portion" of the feature x_{ij} contributes to the principal component, to the highest variance.

The second principal component

$$Z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + ... + \phi_{p2}X_{ip},$$

is a linear combination of the x_{ip}, that has the highest variance out of the ones that is left. Also this linear combination is not correlated with the first principal component.

Therefore, we have two vectors: The first principal component $Z_1 = (\phi_{11}, \phi_{21}, .., \phi_{p1})$ and $Z_2 = (\phi_{12}, \phi_{22}, .., \phi_{p2})$. These two vectors turn out to be orthogonal to each other.