A quick review of

Algorithms for massive datasets

Statistical Methods for Machine Learning

Professors

Dario Malchiodi

Nicolò Cesa-Bianchi

By

Ali Rafiei

University of Milan

August 2022

Task:

implement a classifier based on logistic regression to predict whether or not a flight will be canceled.

Let's solve it

The question is solving this problem with Logistic regression. Logistic regression is a supervised machine learning algorithm. Supervised means during working on our model to predict new data we have access to ground truth or actual value for each sample. Supervised learning is divided to two subcategories which are Classification and Regression. In Regression we work with continues numbers, for example we want to estimate the price of a house based on the number of its bedrooms, the price is a continues number so we need to solve it with Regression algorithms. But in classification our target are not continues numbers and we try to categories samples into different classes so also we call it categorization. For example in this report we try to find a flight will be cancelled or not, so our target is to find the labels which are "cancelled" or "done".

The problem should solve with Logistic regression and this algorithm has some similarity in some topics with Linear Regression. Thus first we introduce Linear Regression then Logistic Regression and solve our problem.

In this report we also try to find some solutions by using other machine learning algorithms, some of them are supervised like Artificial Neural Networks (ANN), Decision Tree, Support Vector Machines (SVM) and some of them are unsupervised like K Nearest Neighbors (KNN), Principle Components (PCA).

We start our journey in the world of machine learning algorithms by introducing Linear Regression.

1- Linear Regression

In Linear Regression we try to find the best function among a set of functions that we call it Hypothesis set. Inside the Hypothesis set there is a function which is really close to the Unknown Target Function. Unknown Target Function is the best function that supports all correct answers for each data. In the real world it is almost impossible to fine Target Unknown Function exactly because the concept of the Noise is always a part of our problems and it is not possible to model the noises but we try to find that function which is accurate as much as possible to Unknown Target Function.

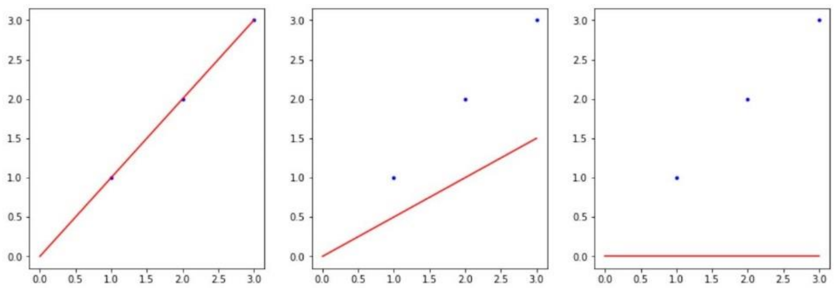
Hypothesis Set $\rightarrow$ $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

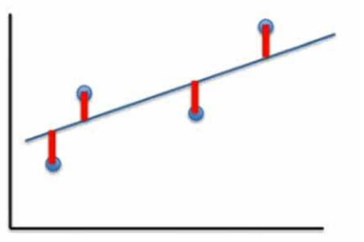
- $h_\theta(x) = \theta_0 + \theta_1 x_1$
- θ_0 : shift along the axis Y (bias)

- θ_1: weight (slope)
- x: training/test data
- $h_\theta(x)$: predictor (model)
- d: the number of features
- $task: finding\ \theta_0\ and\ \theta_1$

Example: We want to find the best line which can predict the target with high accuracy. In this sample we consume we could find these three lines and we want to know which one is the best. In next parts we will understand how we could find these lines.

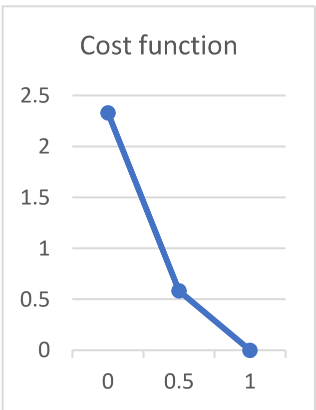
Hypothesis Set $\rightarrow h_\theta(x) = \theta_0 + \theta_1 x_1 \xRightarrow{\theta_0=0} h_\theta(x) = \theta_1 x_1$





Cost function $\rightarrow J(\theta_0, \theta_1) = \dfrac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

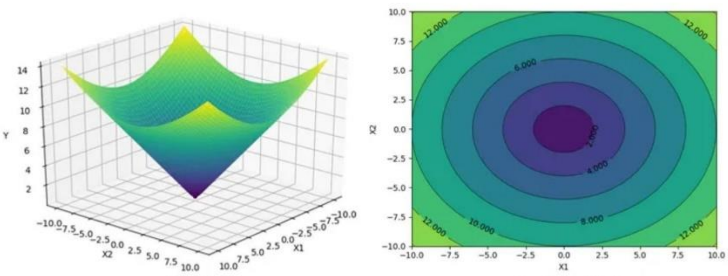
- $h_\theta(x^{(i)})$: $Predicted\ value$
- $y^{(i)}$: $Actual\ value$
- m: $Number\ of\ data$
- $J(\theta_0, \theta_1)$ is a convex function



- $\theta_1 = 1 \rightarrow h_\theta(x) = \theta_1 x_1 = x_1$
  - $J(\theta_1) = \frac{1}{2\times3}[(1-1)^2 + (2-2)^2 + (3-3)^2] = 0$
- $\theta_1 = 0.5 \rightarrow h_\theta(x) = \theta_1 x_1 = 0.5x_1$
  - $J(\theta_1) = \frac{1}{2\times3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] = 0.583$
- $\theta_1 = 0 \rightarrow h_\theta(x) = \theta_1 x_1 = h_\theta(x) = 0$
  - $J(\theta_1) = \frac{1}{2\times3}[(0-1)^2 + (0-2)^2 + (0-3)^2] = 2.33$

What we saw was related to problems with one variable. In two variable cases the cost function will has a 3d shape which X and Y axis are our two variables and the height would be the cost value. But sometimes to show the result a bit easier we use contour plots. As it is shown the minimum value has dark blue color in left 3d diagram and to show it simpler we Drew it like right 2d diagram with contours that minimum value for cost function is colored by dark blue.

Image source: https://www.adeveloperdiary.com/data-science/how-to-visualize-gradient-descent-using-contour-plot-in-python/



3D Plot Contour Plot

Finding parameters (partial derivative = 0):

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})^2$$

$\frac{\partial J}{\partial \theta_0} = 0 \rightarrow \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) \times 1 = 0$	$\theta_1 = \frac{m\sum xy - \sum x \sum y}{m\sum x^2 - (\sum x)^2}$
$\frac{\partial J}{\partial \theta_1} = 0 \rightarrow \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) \times x^{(i)} = 0$	$\theta_0 = \bar{y} - \theta_1 \bar{x}$

Example: finding the best θ_0 and θ_1 for given data

x	y	xy	x^2
0	12	0	0
1	19	19	1
2	29	58	4
3	37	111	9
4	45	180	16
$\sum x = 10$	$\sum y = 142$	$\sum xy = 368$	$\sum x^2 = 30$

$$\left.\begin{array}{l} \theta_1 = \frac{m\sum xy - \sum x \sum y}{m\sum x^2 - (\sum x)^2} = \frac{5 \times 368 - 10 \times 142}{5 \times 30 - 10^2} = 8.4 \\ \theta_0 = \bar{y} - \theta_1 \bar{x} = \frac{142}{5} - 8.4 \times \frac{10}{5} = 11.6 \end{array}\right\} \rightarrow h_\theta(x) = 11.6 + 8.4x$$

Gradient Descent:

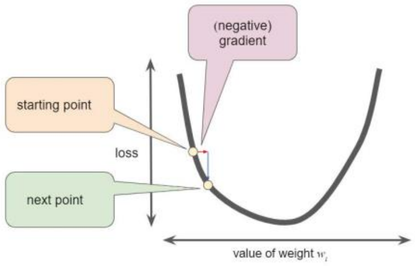
First, we review what is the gradient. Gradient is a vector, to find the vector we need to find partial derivative for each parameter of the function.

$$f(x, y, z) = 2x + 3y^2 - z^3$$

$$\nabla f = \frac{\partial f}{\partial x}\vec{\imath} + \frac{\partial f}{\partial y}\vec{\jmath} + \frac{\partial f}{\partial z}\vec{k} \quad \rightarrow \quad 2\vec{\imath} + 6y\vec{\jmath} + (-3z^2)\vec{k}$$

GD is an optimization algorithm to find the local minimum for a function. To find the minimum value we set a random value for parameters and in each step, we move toward minus of gradient to reach to the minimum value of the function. In our case the function is the cost function and the local minimum is the lowest value for cost.

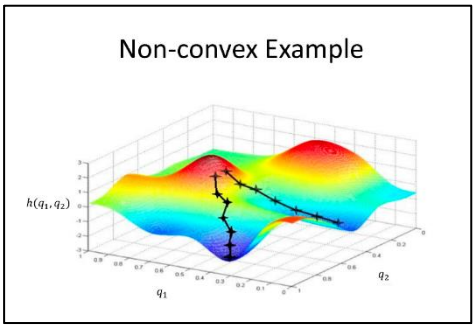
How to tune parameters? Repeating (updating) until convergence (for two variables function):



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

- θ_j: θ_0 and θ_1
- α: *learning rate*
- Movement of θ_j is toward negative of gradient.

Image source: https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent



One problem related to Gradient Descent is that it is possible we get stuck in a local minimum instead of finding global minimum so the starting point is important.

The cost function in Linear Regression is a convex function, Thus, in case of convergence it would be a global minimum and we are sure that we do not get stuck in local minimum.

Image source: https://shashank-ojha.github.io/ParallelGradientDescent/

Learning rate:

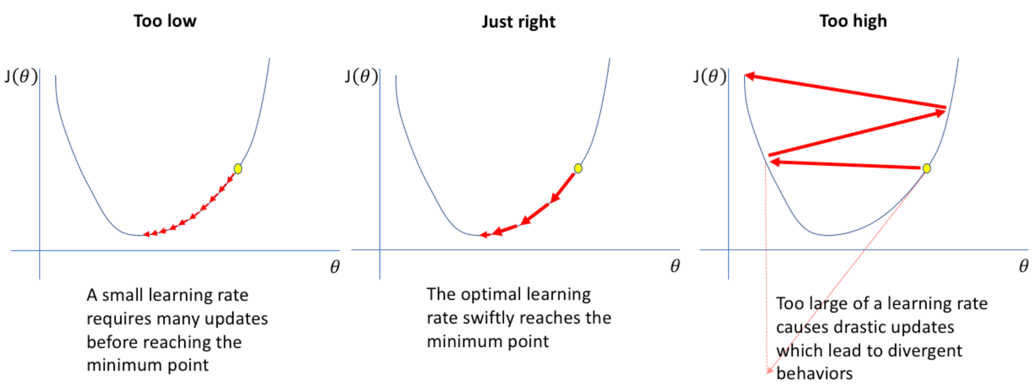


Image source: https://www.jeremyjordan.me/nn-learning-rate/

Using **Gradient Descent in Linear Regression** (with one variable):

Cost (convex) function of Linear Regression: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$

Gradient descent: $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j} J(\theta_0, \theta_1) \xrightarrow{J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}(\theta_0+\theta_1 x^{(i)}-y^{(i)})^2} \theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}\,(\frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2)$

$\frac{\partial J}{\partial\theta_0} = 0$	$\theta_0 = \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})$
$\frac{\partial J}{\partial\theta_1} = 0$	$\theta_1 = \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)}).\, x^{(i)}$

Normal equation:

We can solve Linear Regression problem also with Normal equation:

$X\theta = y$

$Normal\ Equation: X^T\, X\theta = X^T y \xrightarrow{X^T\, X\theta\, \times\, (X^T\, X\theta)^{-1} = I} \theta = (X^T\, X)^{-1} X^T y$

	Size	Bedrooms	Floors	Age	Price
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$\theta = (X^T\, X)^{-1} X^T y \rightarrow to\ solve\ Linear\ Regression\ with\ 4\ features$

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

$$\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2104 & 1416 & 1534 & 852 \\ 5 & 3 & 3 & 2 \\ 1 & 2 & 2 & 1 \\ 45 & 40 & 320 & 36 \end{bmatrix} \times \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}\right)^{-1} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2104 & 1416 & 1534 & 852 \\ 5 & 3 & 3 & 2 \\ 1 & 2 & 2 & 1 \\ 45 & 40 & 320 & 36 \end{bmatrix} \times \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

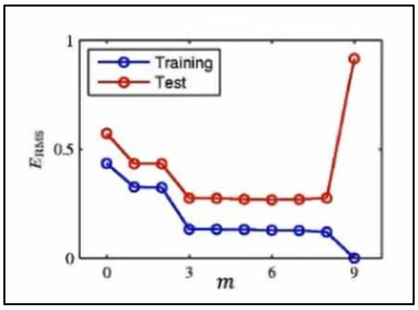
Gradient Descent	**Normal Equation**
Needs to define learning rate	Defining learning rate not needed
Needs much iterations	Iterations not needed
Good for large datasets	Not good for large datasets

Linear Regression Errors

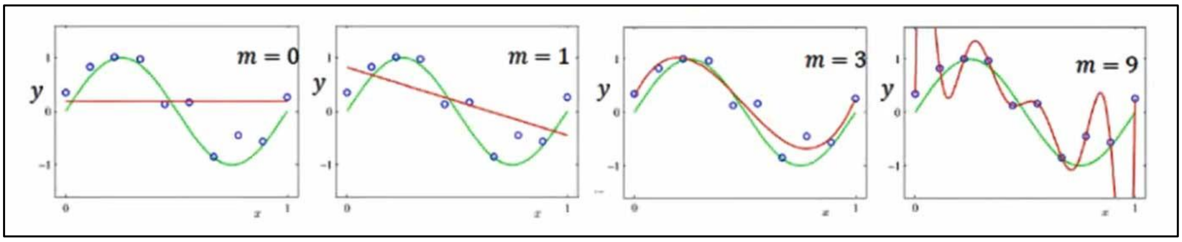
$Expected\ error\ =\ Structural\ Error\ +\ Approximation\ Error$

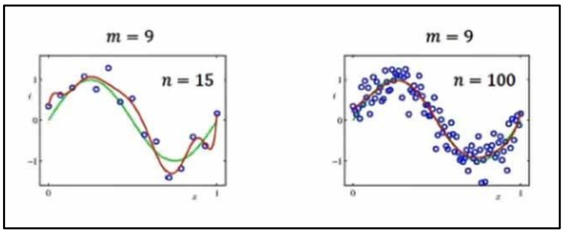
- $Structural\ Error: E_{x,y}[(y - w^{*T}x)^2] \rightarrow Type\ of\ model\ (linear\ , \ldots)$
- $Approximation\ Error: E_x[(w^{*T}x - \widehat{w}^T x)^2] \rightarrow Low\ training\ data$
  - $w^*: optimal\ Linear\ Regression\ paremeters\ (infinite\ training\ data)$
  - $\widehat{w}: found\ Linear\ Regression\ parameters\ based\ on\ limited\ data$

Overfitting



- Number of training data: 10
- Degree of equation: 9
  - 10 parameters
  - 10 equations 10 parameters
  - error = 0
- Error for new data → High





Training data: 15 | 100

Degree of equation: 9

Error for new data → Low

Avoid Overfitting

- Cross validation
- Regularization

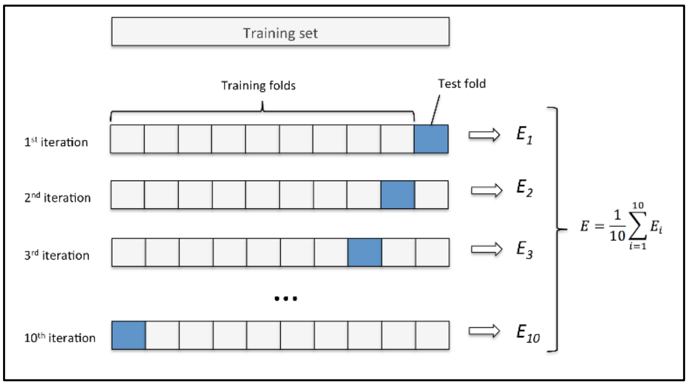
K-Fold Cross Validation:

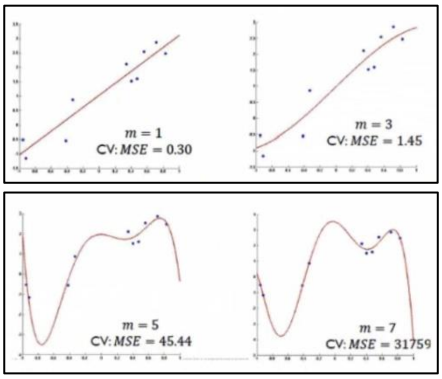
https://nbviewer.org/github/rasbt/python-machine-learning-book/blob/master/code/ch06/ch06.ipynb#K-fold-cross-validation

By using Cross Validation we separate our data to K folds. For example we have 500 data and K=10. It means I have 9 folds of 50 data for training and 1 fold of 50 data for test. In the next iteration I consider another 9 folds as training data and the other 1 fold for test. Due to having 10 folds we need to do it 10 times and it means all data are used one time for training and one time for test. Every time we find the error for that 1 fold which was considered as test and finally the error of the model is the mean of all 10 errors.

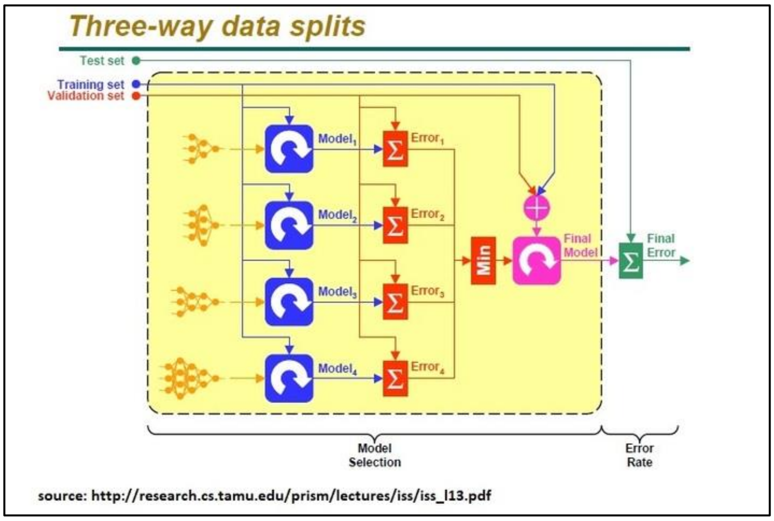
In this sample we worked on 4 models for prediction. In each model we did cross validation. When the line equation is degree 1 the error of cross validation is 0.30 (CV: MSE=0.30) for degree 3 is 1.45 for degree 5 is 45.44 for degree 7 is 31759 and we see in the simplest model we have the minimum error and as a result we decide to choose it as our best model.

Image source: https://nbviewer.org/github/rasbt/python-machine-learning-book/blob/master/code/ch06/ch06.ipynb#K-fold-cross-validation



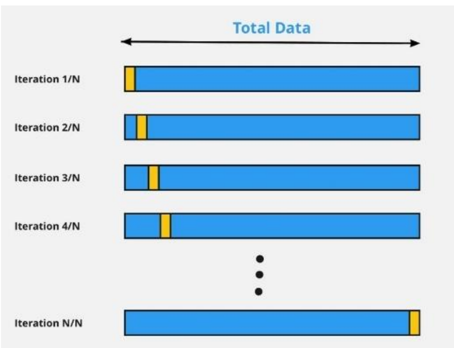


Splitting data in this figure shows that training set is used to train different models. By using validation set we can find which model has the minimum error. Finally by using test set we can find the error of the best model.



Leave One Out Cross Validation (LOOCV)

Image source: https://i1.wp.com/dataaspirant.com/wp-content/uploads/2020/12/7-LOOCV-Leave-One-Out-Cross-Validation.png?ssl=1



In this approach K equles to the number of our data. If we have 500 data every time we consider one data as test data and train the model by the other 499 data.

In this case we need to iterate it 500 times, each time find the error and the mean of 500 error would be the error of the model.

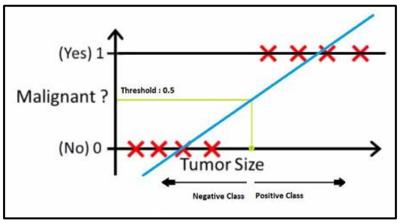
Finding best features for Linear Regression

- Ridge: ordinary least squares (OLS) + term($\alpha \sum \theta^2$)
- Lasso: ordinary least squares (OLS) + term($\alpha \sum |\theta|$)

By using Ridge and Lasso Regression we can find the best features which have more effect on prediction. As a result they can remove (Lasso Regression) or reduce the effect of non important features' coefficients (Ridge regression) in the line equation.

2- Logistic Regression

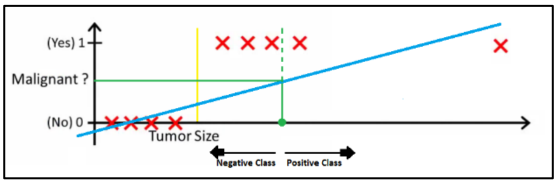
Logistic Regression is a supervised learning algorithm. Although the word of Regression is a part of this algorithm but is not used for solving Regression problems like what we saw in Linear Regression and it would be used for solving classification problems. In theory some basic parts are similar to Linear Regression and for this reason first we studied about Linear Regression to start understanding Logistic Regression better and easier.



Based on Linear Regression algorithm we can find a line to divide or samples to malignant and benign.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 \rightarrow \begin{cases} h_\theta(x) \geq 0.5 \rightarrow y = 1 \\ h_\theta(x) \leq 0.5 \rightarrow y = 0 \end{cases}$$

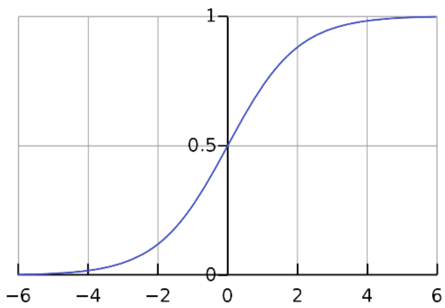
But by adding a new data to our training dataset the equation of the line will be changed and as a result we will have some wrong predictions so it means in this case Linear Regression and drawing a simple line is not a powerful way to reach to a strong classifier.



Linear Regression vs Logistic Regression

For simplicity we can consider the Logistic Regression as the Linear Regression which has a threshold. In Linear Regression the prediction function was $h_\theta(x) = \theta^T x$, and by applying a Logistic function (Sigmoid function) on it we will have the Logistic Regression like $h_\theta(x) = g(\theta^T x)$.c

Image source: https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Logistic-curve.svg



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

$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$, $0 < h_\theta(x) < 1$

$\begin{cases} if\ h_\theta(x) \geq 0.5 : y = 1 \\ if\ h_\theta(x) < 0.5 : y = 0 \end{cases} \rightarrow \begin{cases} if\ \theta^T x \geq 0.5 : y = 1 \\ if\ \theta^T x < 0.5 : y = 0 \end{cases}$

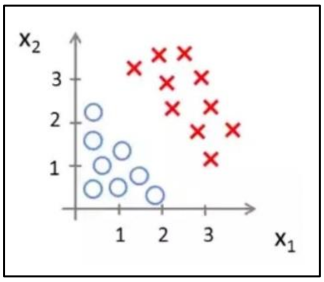
$g: logistic\ function \rightarrow sigmoid$

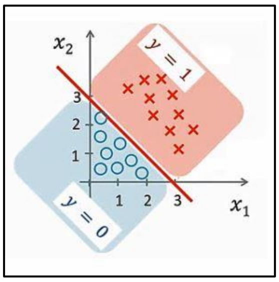
Now the task is finding the weights (θ). By minimizing this cost function (cross entropy) we will find them but first, we will see some examples to understand better the topic and we will back and study the cost function in more details.

$$Cost\ function \rightarrow\ J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]$$

Example: In this example we want to divide blue and red samples. We will see how we can find the weights in next part but for now we assume that we could find the weights and they are $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$. So now we need to find $\theta^T x$, and see it is greater than 0.5 or not to find the correct class.

Image source: https://viblo.asia/p/machine-learning-logistic-regression-bJzKm176K9N





$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

$\theta^T x = [-3 \quad 1 \quad 1]\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = -3 + x_1 + x_2$

$\begin{cases} -3 + x_1 + x_2 \geq 0 \rightarrow y = 1 \\ -3 + x_1 + x_2 < 0 \rightarrow y = 0 \end{cases}$

Cost function:

In Linear regression the cost function was $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$, but the function of Linear Regression was $h_\theta(x)$, but in Logistic Regression the function is $g(\theta^T x)$, and if we want to use this function in this cost function the result will have a function which is not convex. We know that to find global minimum we need a convex cost function so we will change the cost function.

$h_\theta(x)$ in linear regression is changed to $g(\theta^T x)$ and as a result $J(\theta)$ or cost function for it is not a convex function so I need to use another cost function to avoid getting stuck in local minimum.

$$Cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right) : y = 1 \\ -\log\left(1 - h_\theta(x)\right) : y = 0 \end{cases}$$

We can write above function in one single line easily like this:

$$Cost(h_\theta(x), y) = -ylog\big(h_\theta(x)\big) - (1-y)\log\big(1-h_\theta(x)\big)$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\big(x^{(i)}\big)\right) - \big(1-y^{(i)}\big)\log\left(1-h_\theta\big(x^{(i)}\big)\right)\right]$$

- $h_\theta(x){:}\ predicted\ target$
- $y{:}\ Actual\ target$

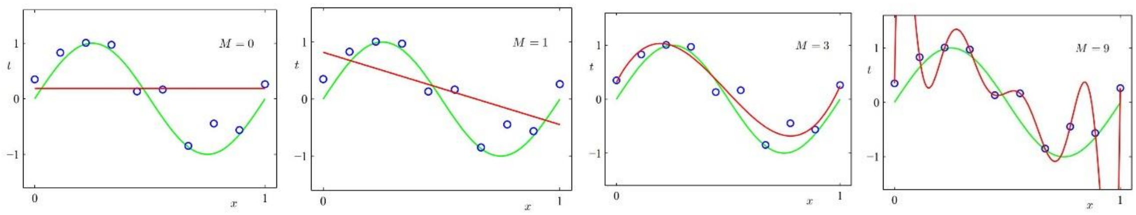
Now this new cost function is convex so we can use **gradient descent** to find weights (θ parameters) without getting stuck in local minimum.

Cost (convex) function of Logistic Regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\big(x^{(i)}\big)\right) - \big(1-y^{(i)}\big)\log\left(1-h_\theta\big(x^{(i)}\big)\right)\right]$

Gradient descent: $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \xrightarrow{J(\theta)\ fot\ Log\ Reg} \frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}).x_j^{(i)})$

Regression regularization

Image source: https://www.cs.cmu.edu/~atalwalk/teaching/winter17/cs260/lectures/lec09.pdf



	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

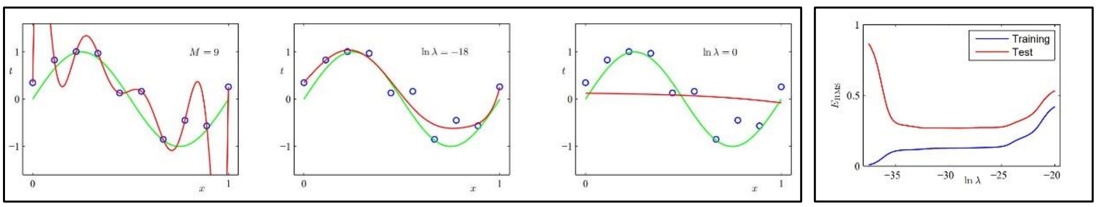
Overfitting problem is related to predict training data with high accuracy and predict test data with low accuracy.

One reason that leads to have overfitted model is increasing the parameters of model. As it is shown when the line equation is degree 1 we have only one weight with the value of 0.19 but by increasing the degree of equation the number and value of parameters are starting to increase.

To avoid having this problem we will use regression Regularization. The cost function was $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})^2$, and now we need to add $\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$, to it. The result would be:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Image source: https://haipeng-luo.net/courses/CSCI567/2021_fall/lec2.pdf



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
w_0	0.35	0.35	0.13
w_1	232.37	4.74	-0.05
w_2	-5321.83	-0.77	-0.06
w_3	48568.31	-31.97	-0.06
w_4	-231639.30	-3.89	-0.03
w_5	640042.26	55.28	-0.02
w_6	-1061800.52	41.32	-0.01
w_7	1042400.18	-45.95	-0.00
w_8	-557682.99	-91.53	0.00
w_9	125201.43	72.68	0.01

For degree 9 we will have:

$$column\ 1:\ \lambda = 0 \rightarrow \ln\lambda = -\infty \rightarrow \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 = 0$$

$$column\ 2:\ \lambda = e^{-18} \rightarrow \ln\lambda = -18$$

$$column\ 3:\ \lambda = 1 \rightarrow \ln\lambda = 0$$

Gradient descent in Regularized Linear Regression

Before we saw when we use gradient descent for Linear Regression the updating formulas were $\theta_0 = \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})$, and $\theta_1 = \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)}).x^{(i)}$, and now our cost function is $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$, so updating formulas for weight would be:

- $\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}).x^{(i)}$
- $\theta_{\mathrm{j}} := \theta_{\mathrm{j}} - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}).x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$

Normal Equation

Before we saw we can find weights by using $\theta = (X^T X)^{-1}X^T y$, but sometimes $X^T X$, is not invertible so now we can solve this problem and also regularized Linear Regression.

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

Logistic regression regularization:

Overfitting problem is related to predict training data with high accuracy and predict test data with low accuracy. One reason that leads to have overfitted model is increasing the parameters of model.

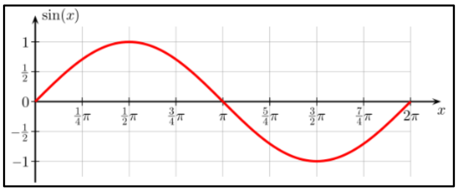
To avoid having this problem we will use Regularized Logistic Regression. The cost function was $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$, and $h_\theta(x)$, was $g(\theta^T x)$ which was $\frac{1}{1+e^{-\theta^T x}}$, now we need to add $\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$, to it. The result would be:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

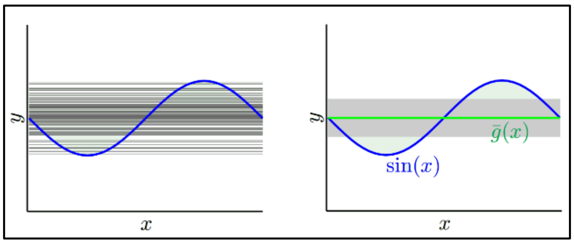
Bias & Variance

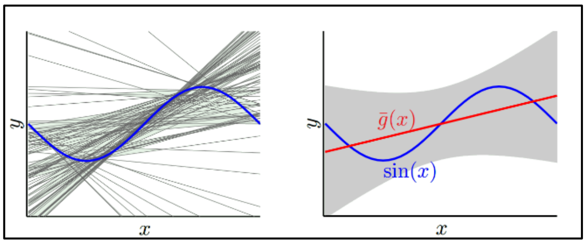
The amount of Error is a summation of **Bias**, **Variance** and **Noise.** To reducing the amount of Error we cannot solve the problem of existence of the noise because modeling the noise is impossible but we can reduce the amount of Bias and Variance.

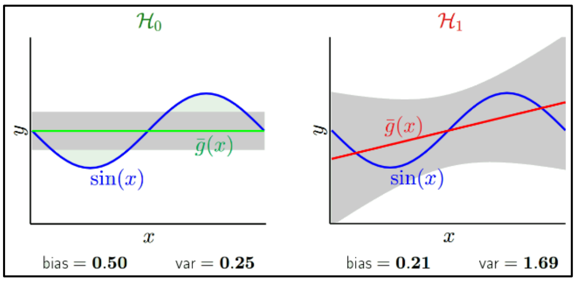
Image source: https://stats.stackexchange.com/questions/4284/intuitive-explanation-of-the-bias-variance-tradeoff



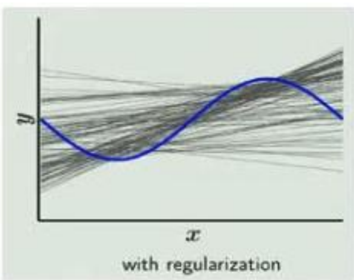
In this Sine function we can work with simple and complex Hypothesis set. The simple one is $H_0: f(x) = b$, and if K times we create training set as a result we will have K number of $f(x) = b$, which are shown by black lines and the mean of all found functions would be $\bar{f}$ which is shown by green line. We have the same strategy for the complex one which is $H_1: f(x) = ax + b$.

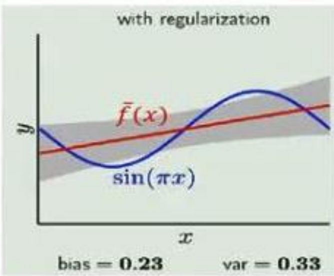






As we can see the amount of (Bias + Variance) in simplex Hypothesis set is (0.50 + 0.25) and in complex Hypothesis set is (0.21 + 1.69) so it means by working with simple one which is $H_0: f(x) = b$, we will have lower error.

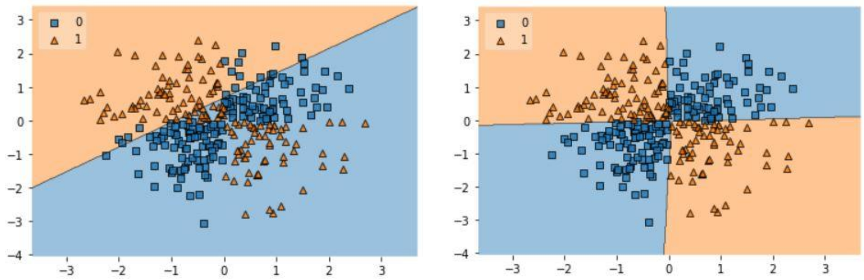


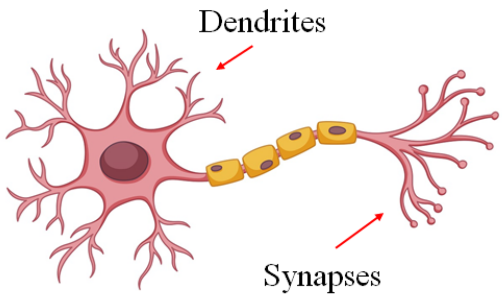


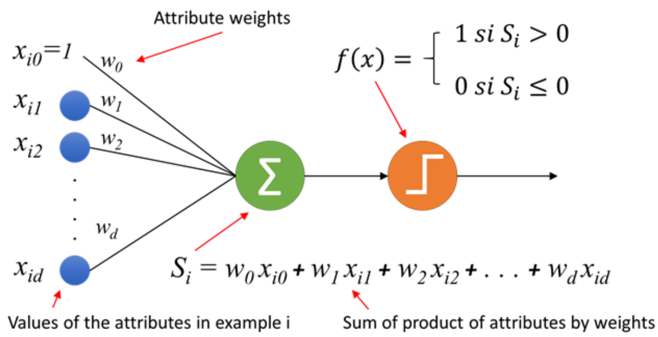
But when we use regularized term in complex hypothesis set which is $H_1: f(x) = ax + b$, we can the amout of (bias + variance) is (0.23 + 0.33) and it mean amont these three models, the last one is the best.

3- Artificial Neural Network

Logistic regression by increasing the number of parameters (features and weights) cannot work well.







NEURON **PERCEPTRON**

Image source: https://inteligenciafutura.mx/english-version-blog/blog-06-english-version

How to train a perceptron? → by using **delta rule**

- $w_i := w_i + \alpha . e . x_i$
- $w_i \rightarrow random\ numbers\ in\ first\ epoch$
- $x_i \rightarrow inputs$
- $\alpha \rightarrow learning\ rate$
- $e \rightarrow error$

Example: OR function

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

First epoch:

$x_1 = 0 \mid x_2 = 0 \mid w_1 = -0.2 \mid w_2 = 0.4$

- $y = f(0 \times -0.2 + 0 \times 0.4) = f(0) = 0$
- $e = 0 - 0 = 0$

$x_1 = 0 \mid x_2 = 1 \mid w_1 = -0.2 \mid w_2 = 0.4$

- $y = f(0 \times -0.2 + 1 \times 0.4) = f(0.4) = 1$
- $e = 0 - 0 = 0$

$x_1 = 1 \mid x_2 = 0 \mid w_1 = -0.2 \mid w_2 = 0.4$

- $y = f(1 \times -0.2 + 0 \times 0.4) = f(-0.2) = 0$
- $e = 1 - 0 = 1$
    - $w_i \coloneqq w_i + \alpha.e.x_i \mid \alpha = 0.2$
    - $w_1 \coloneqq w_1 + 0.2 \times 1 \times x_1 = -0.2 + 0.2 = 0$
    - $w_2 \coloneqq w_2 + 0.2 \times 1 \times x_2 = 0.4 + 0 = 0.4$

$x_1 = 1 \mid x_2 = 1 \mid w_1 = 0 \mid w_2 = 0.4$

- $y = f(1 \times 0 + 1 \times 0.4) = f(0.4) = 1$
- $e = 1 - 1 = 0$

We need to repeat epochs to reach an epoch without any errors.

$h_\theta(x) = f(0 + 0.2x_1 + 0.4x_2)$

Proof of delta (perceptron) rule:

Activation function → identity : $f(x) = x$

Output for 2 variables → $y = w_1x_1 + w_2x_2$

We try to minimize → $E = (t - y)^2$ → $-\nabla E$

$\nabla E = -2(t - y)\nabla y$

$E = (t - w_1x_1 - w_2x_2)^2$

$$\nabla E = \begin{pmatrix} \dfrac{\partial E}{\partial w_1} \\ \dfrac{\partial E}{\partial w_2} \end{pmatrix} = \begin{pmatrix} -2(t-y)x_1 \\ -2(t-y)x_2 \end{pmatrix}$$

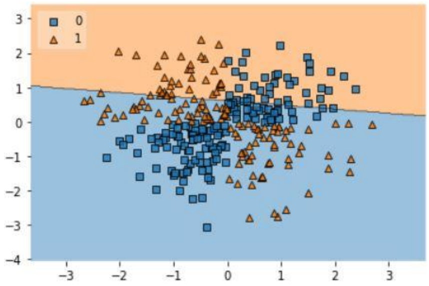
$w_i \coloneqq w_i - k\nabla E$

$w_1 \coloneqq w_1 - k(-2(t - y)x_1) = w_1 - k(-2.\mathrm{e}.x_1) = w_1 + 2k\mathrm{e}x_1 = w_1 + \alpha\mathrm{e}x_1$

$w_2 \coloneqq w_2 - k(-2(t - y)x_2) = w_2 - k(-2.\mathrm{e}.x_2) = w_2 + 2k\mathrm{e}x_2 = w_2 + \alpha\mathrm{e}x_2$

$w_i \coloneqq w_i + \alpha.e.x_i$

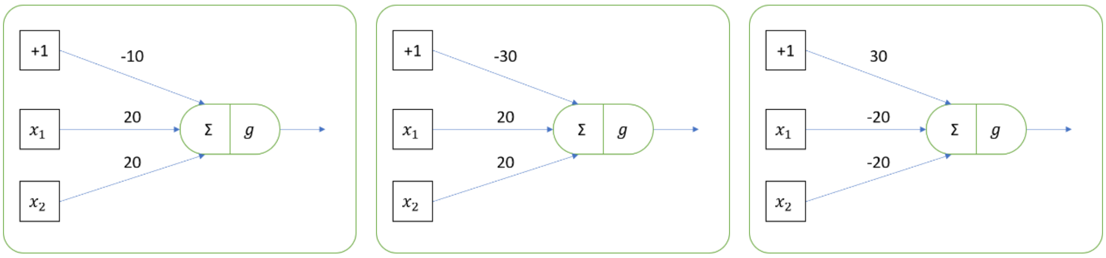
Multi Layer Perceptron



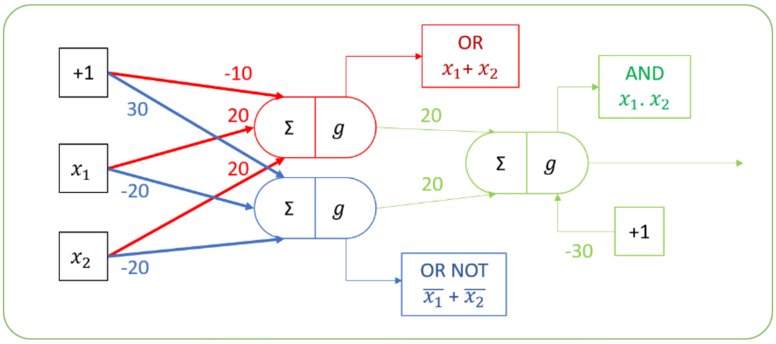
Single layer perceptron → can solve linear separable problems

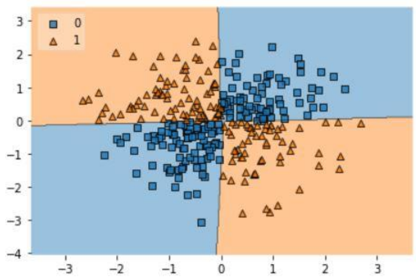
To solve non-linear separable problems → MLPs

$(x_1)\ OR\ (x_2)$ $\quad$ $(x_1)\ AND\ (x_2)$ $\quad$ $(not\ x_1)\ OR\ (not\ x_2)$



$(x_1)\ XOR\ (x_2)$

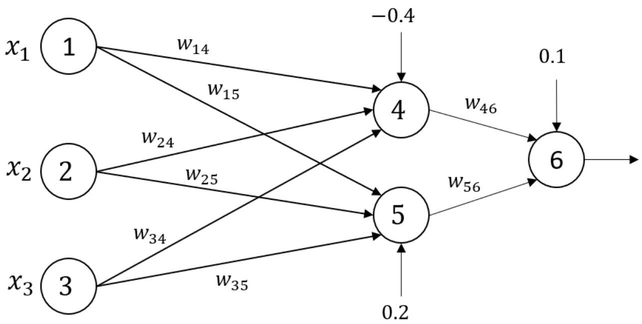




Backpropagation

A way to reduce gradient descent to reduce output error.

Example: from Data Mining: Concepts and Techniques, 3rd Edition. Jiawei Han



w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}
0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2

Input → (1, 0, 1)
Expected output → 1
First layer activation function → identity
Hidden layers activation function → sigmoid

Step1: Feed forward

Unit, j	*Net Input, I_j*	*Output, O_j*
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

Expected output → 1
Actual output → 0.474
We need to update weights → based on amount of errors
Finding errors → from last layer to first layer

Step2: Backpropagation

- Find error for output neuron: $e_y = (t - y)\acute{f}(y) \xRightarrow{activation:\, sigmoid} e_y = (t - y)y(1 - y)$
- Find error for hidden neuron: $e_j = (e_y . w_j)h_j\left(1 - h_j\right)$
  - $e_j \to current\ neuron\ (hidden\ neuron)$
  - $e_y \to error\ of\ output\ neuron$
  - $w_j \to the\ weight\ between\ hidden\ neuron\ and\ output\ neuron$
  - $h_j \to output\ of\ hidden\ neuron$

Unit, j	*Err_j*
6	$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$
5	$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

Step3: Finding new weights

$w_{ij} \coloneqq w_{ij} + \alpha . e_j . o_i$

- $\alpha = 0.9$
- $e_j \to error\ of\ output\ neuron$

- $o_i \rightarrow output\ of\ curren\ neuron$

Weight or Bias	New Value
w_{46}	$-0.3+(0.9)(0.1311)(0.332)=-0.261$
w_{56}	$-0.2+(0.9)(0.1311)(0.525)=-0.138$
w_{14}	$0.2+(0.9)(-0.0087)(1)=0.192$
w_{15}	$-0.3+(0.9)(-0.0065)(1)=-0.306$
w_{24}	$0.4+(0.9)(-0.0087)(0)=0.4$
w_{25}	$0.1+(0.9)(-0.0065)(0)=0.1$
w_{34}	$-0.5+(0.9)(-0.0087)(1)=-0.508$
w_{35}	$0.2+(0.9)(-0.0065)(1)=0.194$

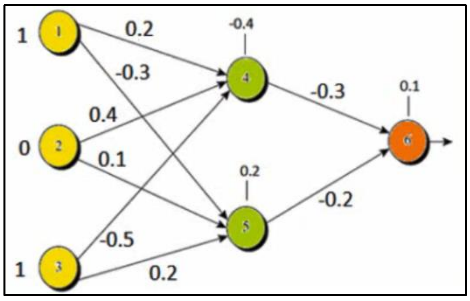
Step4: finding bias

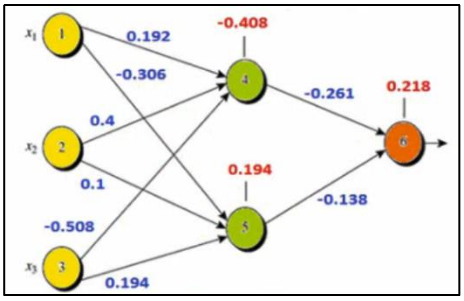
$b_i \coloneqq b_i + \alpha . e_i$

- $\alpha \rightarrow learning\ rate$
- $b_i \rightarrow bias\ of\ current\ neuron$
- $e_i \rightarrow error\ of\ current\ neuron$

θ_6	$0.1+(0.9)(0.1311)=0.218$
θ_5	$0.2+(0.9)(-0.0065)=0.194$
θ_4	$-0.4+(0.9)(-0.0087)=-0.408$

Result

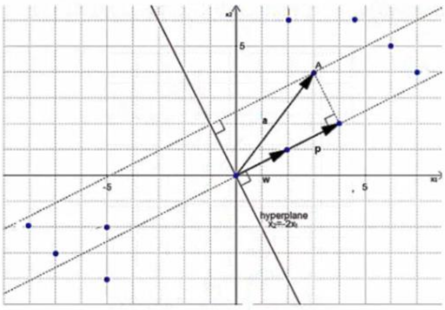




4- SVM

A line that has the longest distance from each two groups of data points.

The distance is the same from closest data from each groups.



Distance between point A and hyperplane:

$Line\ equation \rightarrow\ x_2 = -2x_1 \rightarrow x_2 + 2x_1 = 0$

$\text{Parameters} \rightarrow \text{w}\begin{vmatrix}2\\1\end{vmatrix} \quad \text{x}\begin{vmatrix}x_1\\x_2\end{vmatrix}$

$By\ using: w^T x = 0 \rightarrow [2 \quad 1]\begin{bmatrix}x_1\\x_2\end{bmatrix} = 0 \rightarrow 2x_1 + x_2 = 0$

$\text{Distance between A}\begin{vmatrix}3\\4\end{vmatrix}\ and\ hyperplane \rightarrow \dfrac{[2 \quad 1]\begin{bmatrix}3\\4\end{bmatrix}}{\|w\|} = \dfrac{[2 \quad 1]\begin{bmatrix}3\\4\end{bmatrix}}{\left\|\sqrt{2^2+1^2}\right\|} = 2\sqrt{5}$

What I need?

$Line\ equation \rightarrow w^T x + b = 0$

Margin value

$Line\ equation\ for\ class\ A \rightarrow w^T x + b = +1$

$Line\ equation\ for\ class\ B \rightarrow w^T x + b = -1$

$Separator\ line\ equation \rightarrow w^T x + b = 0$

$Distance\ between\ closest\ data\ point\ in\ class\ A\ and\ separator\ line$
$$\rightarrow \frac{|WX + b|}{\|W\|} \xrightarrow{w^T x + b = +1} \frac{1}{\|W\|}$$

$Margin \rightarrow 2\dfrac{1}{\|W\|} = \dfrac{2}{\|W\|}$

$The\ goal\ is\ minimizing \rightarrow \dfrac{2}{\|W\|}$

Support vectors:

The closest data points to the separator hyperplane.

SVM is stable algorithm, by changing data points (except support vectors) the hyperplane will not change.

$instead\ of\ minimizing\ \dfrac{2}{\|W\|}, we\ maximize\ \dfrac{1}{2}\|W\|^2\ because\ it\ is\ a\ convex\ function.$

$\text{Subject to}\begin{cases}WX_i + b \geq +1\\WX_i + b \leq -1\end{cases}\ or\ \ y_i(WX_i + b) \geq +1 \quad i \in [1, m]$

Solving optimization problem with Lagrange

$$\begin{cases} Maximize\ f(x,y) \\ subject\ to\ g(x,y) = 0 \end{cases}$$

$$\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda.\, g(x,y)$$

$$\nabla_{x,y,\lambda}\mathcal{L}(x,y,\lambda) = 0 \rightarrow to\ find\ x,y,\lambda$$

Example: $\begin{cases} Minimize\ f(x) = x^2 \\ subject\ to\ g(x){:}\ x = 1 \end{cases}$

$$\mathcal{L}(x,\lambda) = f(x) - \lambda.\, g(x)$$

$$\mathcal{L}(x,\lambda) = x^2 - \lambda(x-1) = x^2 - \lambda x + \lambda$$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x} = 0 \rightarrow 2x - \lambda = 0 \rightarrow \lambda = 2 \\ \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \rightarrow -x + 1 = 0 \rightarrow x = 1 \end{cases}$$

More than one constraint:

$$\mathcal{L}(x,\lambda) = f(x) - \sum_i \lambda_i g_i(x)$$

$$\nabla_{x,y,\lambda_1,\lambda_2,\dots}\mathcal{L}(x,y,\lambda_1,\lambda_2,\dots) = 0$$

Inequality constraints:

$$g(x) \geq 0 \rightarrow \ \lambda \geq 0$$

$$g(x) \leq 0 \rightarrow \ \lambda \leq 0$$

Example: $\begin{cases} Minimize\ f(x,y) = x^3 + y^2 \\ g(x,y) = x^2 - 1 \geq 0 \end{cases}$

$$\mathcal{L} = f - \lambda g$$

$$\mathcal{L} = x^3 + y^2 - \lambda(x^2 - 1)$$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x} = 0 \rightarrow 3x^2 - 2\lambda x = 0 \rightarrow \lambda = \pm\frac{3}{2} \rightarrow \ \lambda = +\frac{3}{2} \\ \frac{\partial \mathcal{L}}{\partial y} = 0 \rightarrow 2y = 0 \rightarrow y = 0 \\ \frac{\partial \mathcal{L}}{\partial y} = 0 \rightarrow 2y = 0 \rightarrow y = 0 \end{cases}$$

$$f = 1 + 0 = 1$$

Optimizing problem:

$$\begin{cases} Minimize \ \frac{1}{2}\|W\|^2 \\ y_i(WX_i + b) \geq +1 \quad i \in [1, m] \end{cases}$$

$$g_1 = y_1(WX_1 + b) - 1 \geq 0$$
$$g_2 = y_2(WX_2 + b) - 1 \leq 0$$

$$\mathcal{L} = f - \lambda_1 g_1 - \lambda_2 g_2$$

$$\mathcal{L}(x, b, \lambda) = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{m} \lambda_i [y_i(WX_i + b) - 1] \quad \lambda_i \geq 0$$

$$\mathcal{L}(x, b, \lambda) = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{m} \lambda_i y_i(WX_i + b) + \sum_{i=1}^{m} \lambda_i$$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w - \lambda_i y_i x_i = 0 \rightarrow w = \sum_{i=1}^{m} \lambda_i y_i x_i \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow -\sum_{i=1}^{m} \lambda_i y_i = 0 \rightarrow \sum_{i=1}^{m} \lambda_i y_i = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda_i} = 0 \end{cases}$$

Example: finding hyperplane separator

$$w^T x + b = 0 \rightarrow need\ to\ find\ w_1, w_2, b$$

$$\mathcal{L} = \frac{1}{2}\|W\|^2 - \lambda_1 g_1 - \lambda_2 g_2$$

$$\mathcal{L}(x, b, \lambda) = \frac{1}{2}\|W\|^2 - \lambda_1[y_1(WX_1 + b) - 1] - \lambda_2[y_2(WX_2 + b) - 1] \xRightarrow{y_1=1\ , y_2=-1}$$

$$\mathcal{L}(x, b, \lambda) = \tfrac{1}{2}\|W\|^2 - \lambda_1[(wx_1 + b) - 1] + \lambda_2[(wx_2 + b) + 1]$$

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w - \lambda_1 x_1 + \lambda_2 x_2 = 0$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow -\lambda_1 + \lambda_2 = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_1} = 0 \rightarrow wx_1 + b - 1 = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_2} = 0 \rightarrow wx_2 + b + 1 = 0$$

→

$$\lambda_1 = \lambda_2$$

$w - \lambda_1 x_1 + \lambda_1 x_2 = 0$

$wx_1 + b - 1 = wx_2 + b + 1 \rightarrow wx_1 - wx_2 = 2 \rightarrow w(x_1 - x_2) = 2$

$\xRightarrow{(1,1)-(2,2)} (w_1 - w_2).(x_1 - x_2) = 2 \rightarrow -w_1 - w_2 = 2 \rightarrow w_1 = -w_2 - 2$

$w - \lambda_1 x_1 + \lambda_1 x_2 = 0 \xRightarrow{(1,1),(2,2)} w - \lambda_1(1,1) + \lambda_1(2,2) = 0 \rightarrow$

$w - (\lambda_1, \lambda_1) + (2\lambda_1, 2\lambda_1) = 0 \rightarrow \ w + (\lambda_1, \lambda_1) = 0 \rightarrow$

$(w_1, w_2) - (\lambda_1, \lambda_1) = 0 \rightarrow (w_1 + \lambda_1, w_2 + \lambda_1) = 0 \rightarrow$

$w_1 + \lambda_1 = 0, w_2 + \lambda_1 \rightarrow$

$w_1 = w_2 = -1, \lambda_1 = \lambda_2 = 1$

Finding bias:

$wx_1 + b - 1 = 0$

$b = 1 - wx_1 = 1 - (w_1, w_2)(1,1) = 1 - (w_1 + w_2 = 1 - (-2) = 3$

Line equation → $-x_1 - x_2 + 3 = 0$

Dual lagrangian → to solve the problem easier

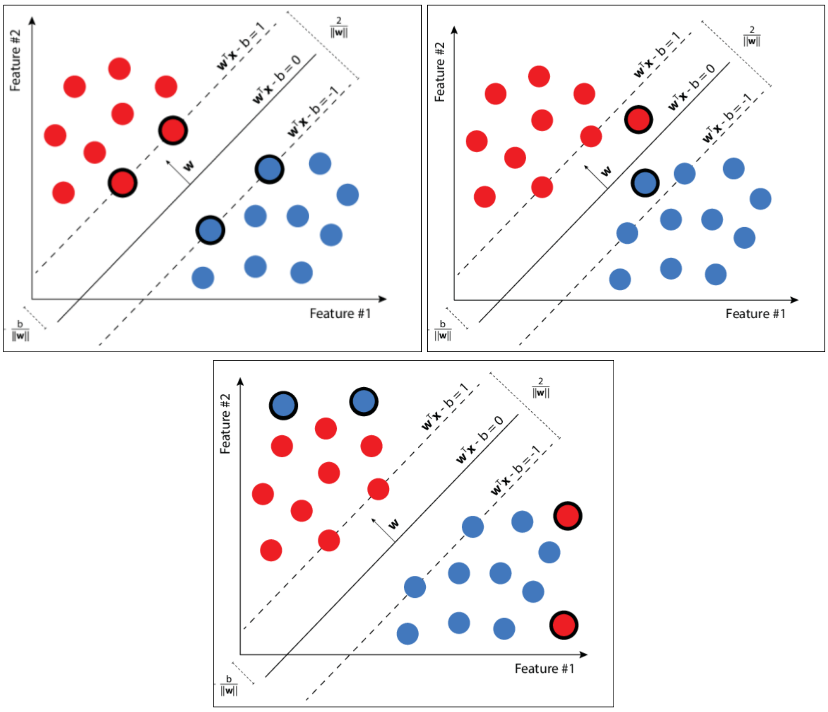
Before we had:

$$\mathcal{L}_{primal}(x, b, \lambda) = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{m} \lambda_i y_i (WX_i + b) + \sum_{i=1}^{m} \lambda_i$$

- $\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{m} \lambda_i y_i x_i$
- $\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \ \sum_{i=1}^{m} \lambda_i y_i = 0$

$$\mathcal{L}_{dual}(x, b, \lambda) \xRightarrow{w=\sum_{i=1}^{m} \lambda_i y_i x_i,\quad \sum_{i=1}^{m} \lambda_i y_i = 0} \frac{1}{2}\sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \left(X_i . X_j\right)$$

Soft margin SVM:



The goal is minimizing → $\frac{1}{2}\|W\|^2 + C\sum_{i=1}^{m} \varepsilon_i$

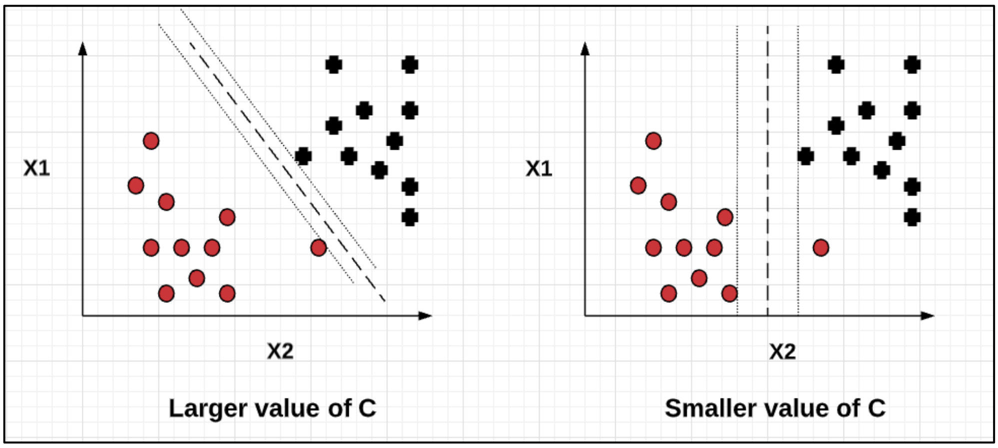
$$C: keep\ a\ balance\ between\ \frac{1}{2}\|W\|^2\ and\ \sum_{i=1}^{m} \varepsilon_i$$

$\varepsilon_i: error\ (distance\ to\ class'border)$

Constraint: $y_i(W.X + b) \geq 1 - \varepsilon_i \qquad and \qquad \varepsilon_i \geq 0$

- When C is low → error is not important + margin is important
- When C is high → error is important + margin is not important

Image source: https://vitalflux.com/svm-soft-margin-classifier-c-value-importance/



Optimization of soft margin SVM:

The goal is minimizing $\rightarrow \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{m} \varepsilon_i$

- $y_i(W.X + b) \geq 1 - \varepsilon_i \rightarrow y_i(W.X + b) - 1 + \varepsilon_i \geq 0 \rightarrow g_i$
- $\varepsilon_i \geq 0 \rightarrow \ \varepsilon_i \rightarrow h_i$

$$\mathcal{L} = f - \sum \lambda_i g_i - \sum \mu_i h_i$$

$$\mathcal{L}_{primal}(x, b, \lambda) = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{m} \varepsilon_i - \sum_{i=1}^{m} \lambda_i\{y_i(WX_i + b) - 1 + \varepsilon_i\} - \sum_{i=1}^{m} \mu_i \varepsilon_i$$

$$\frac{\partial \mathcal{L}_{primal}}{\partial w} = 0 \rightarrow w - \sum \lambda_i y_i x_i = 0 \rightarrow w = \sum_{i=1}^{m} \lambda_i y_i x_i$$

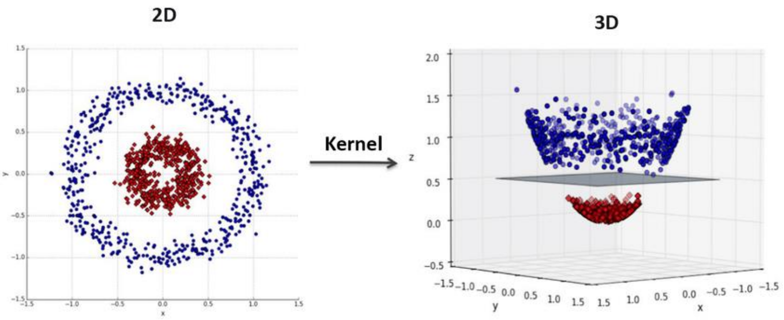
$$\frac{\partial \mathcal{L}_{primal}}{\partial b} = 0 \rightarrow \sum \lambda_i y_i = 0$$

$$\frac{\partial \mathcal{L}_{primal}}{\partial \varepsilon_i} = 0 \rightarrow C - \lambda_i - \mu_i = 0$$

Kernel:

To use linear separator for non-linear data points → we map data points to higher dimension space → Now data points will be separable linearly by a hyperplane

Image source: https://www.researchgate.net/figure/Non-linear-classifier-using-Kernel-trick-16_fig4_340610860



Kernel trick:

$$\mathcal{L}_{dual} = \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j (X_i . X_j)$$

Example: we have 4 data with 2 features:

- (d1*d2), (d1*d2), (d1*d3), (d1*d4)
- (d2*d2), (d2*d2), (d2*d3), (d2*d4)

- (d3*d2), (d3*d2), (d3*d3), (d3*d4)
- (d4*d2), (d4*d2), (d4*d3), (d4*d4)

By mapping these data to another space with 5 features we need to do load of computation

- $< \phi(X_i), \phi(X_j) >$

To solve it easier we use → Kernel trick

- $K(X_i, X_j) =< \phi(X_i), \phi(X_j) >$

Calculation without kernel trick:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \rightarrow \phi \rightarrow \begin{pmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{pmatrix} \qquad u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \overset{\phi}{\Rightarrow} \begin{vmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{vmatrix} \qquad v = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \overset{\phi}{\Rightarrow} \begin{vmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{vmatrix}$$

$$\phi(u).\phi(v) = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

Calculation with kernel trick:

$$K(x, y) = (x.y)^2 = (4 + 10 + 18)^2 = (32)^2 = 2^{10} = 1024$$

Advantages of SVM:

1. Simple train
2. Strong theory (not like NN's black box)
3. Good enough for simple and complex models
4. Good functionality when training data is low
5. Not getting stock in local minimum and find global minimum
6. Margin separator is clear
7. More useful in higher dimensions
8. When features are more than samples is also working good
9. Less probability for overfitting in comparison to Neural Networks
10. Optimal memory usage (working with support vectors)

Disadvantages of SVM:

1. Could be slow when support vectors are high
2. Difficult to find kernel function
3. Difficult to find parameter C
4. Not good with huge data (inner product of data)
5. With high number of noise we have overlapping

5- KNN

K = 1 → square class

K = 3 → triangle class

K = 7 → square class

Model-based learning techniques	Instance-based techniques
Use input data	Store input data
To learn a set of parameters	Until asked to predict a new input
To find a function	Search for similar data
Fast prediction	Sloe prediction

Voronoi diagram (decision boundaries):

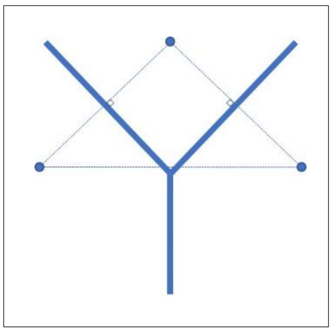
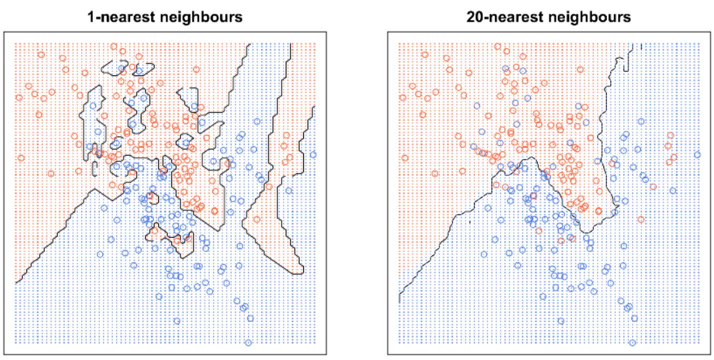


Image source: https://medium.com/analytics-vidhya/%E4%B8%80%E6%96%87%E6%90%9E%E6%87%82k%E8%BF%91%E9%82%BB%E7%AE%97%E6%B3%95-knn-a1c3571562c1

Small K → sensitive to noise

Large K → smoother



Age	Loan	Default	Distance
25	$40k	N	102000
35	$60k	N	82000
45	$80k	N	62000
20	$20k	N	122000
35	$120k	N	22000
52	$180k	N	124000
23	$95k	Y	47000
40	$62k	Y	80000
60	$100k	Y	42000
48	$220k	Y	78000
33	$150k	Y	8000

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771

Increase accuracy:

min-max normalization:

$$X_s = \frac{X - Min}{Max - Min}$$

$$Age \to (20, 60)$$

$$\frac{48-20}{60-20} = 0.7$$

$$Loan \to (18k, 22k)$$

$$\frac{142k - 18k}{220k - 18k} = 0.6138$$

New data:

- Age: 48
- Loan: 142000

Without normalization → $d = \sqrt{(48-33)^2 + (142k - 150k)^2} = 8000.01$

With normalization → $d = \sqrt{(0.625-0.7)^2 + (0.31 - 0.6138)^2} = 0.3160$

6- Dimension Reduction (PCA and t-SNE)

Review:

Diagonal matrix	Identity matrix	Orthogonal matrix
$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$AA^T = A^T A = I$

Matrix determinant → $|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$

Eigenvector and Eigenvalue → $A.v = \lambda.v \to \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} = 4 \times \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

Finding Eigenvalue → $A.v = \lambda.v \to A.v - \lambda.I.v = 0 \to (A - \lambda.I).v = 0 \to root\ of\ |A - \lambda.I|$

Example:

$$A-\lambda.I=\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix}-\lambda\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix}-\begin{bmatrix}\lambda & 0\\ 0 & \lambda\end{bmatrix}=\begin{bmatrix}-\lambda & 1\\ -2 & -3-\lambda\end{bmatrix}$$

$$|A-\lambda.I|=(-\lambda)(-3-\lambda)-(-2)(1)=\lambda^2+3\lambda+2-0\rightarrow\begin{cases}\lambda_1=-1\rightarrow v_1=?\\ \lambda_2=-2\rightarrow v_2=?\end{cases}$$

$$v_1=\begin{bmatrix}x\\ y\end{bmatrix}$$

$$A.v_1=\lambda_1.v_1\rightarrow\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=(-1)\begin{bmatrix}x\\ y\end{bmatrix}\rightarrow\begin{bmatrix}y\\ -2x-3y\end{bmatrix}=\begin{bmatrix}-x\\ -y\end{bmatrix}\rightarrow$$

$$\begin{bmatrix}y\\ -2x-3y\end{bmatrix}-\begin{bmatrix}-x\\ -y\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix}\rightarrow\begin{cases}y+x=0\rightarrow y=-x\\ -2x-3y+y=0\end{cases}\rightarrow v_1=\begin{bmatrix}x\\ -x\end{bmatrix}=x\begin{bmatrix}+1\\ -1\end{bmatrix}$$

$$v_2=\begin{bmatrix}m\\ n\end{bmatrix}$$

$$A.v_2=\lambda_2.v_2\rightarrow\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix}\begin{bmatrix}m\\ n\end{bmatrix}=(-2)\begin{bmatrix}m\\ n\end{bmatrix}\rightarrow\begin{bmatrix}n\\ -2m-3n\end{bmatrix}+2\begin{bmatrix}m\\ n\end{bmatrix}=0\rightarrow$$

$$\begin{bmatrix}n+2m\\ -2m-n\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix}\rightarrow\begin{cases}n+2m=0\rightarrow n=-2m\\ -2m-n=0\end{cases}\rightarrow v_2=\begin{bmatrix}m\\ -2m\end{bmatrix}=m\begin{bmatrix}+1\\ -2\end{bmatrix}$$

Covariance matrix →

$$X=\begin{bmatrix}1 & 2\\ 3 & 5\\ 5 & 8\end{bmatrix}\xRightarrow{zero-centered}\begin{cases}\overline{X_1}=(1+3+5)\div 3=3\\ \overline{X_2}=(2+5+8)\div 3=5\end{cases}\rightarrow\begin{bmatrix}-2 & -3\\ 0 & 0\\ 2 & 3\end{bmatrix}$$

$$\text{Covariance matrix}\rightarrow C=\frac{1}{2}X^TX=\frac{1}{2}\begin{bmatrix}-2 & 0 & 2\\ -3 & 0 & 3\end{bmatrix}\begin{bmatrix}-2 & -3\\ 0 & 0\\ 2 & 3\end{bmatrix}=\begin{bmatrix}4 & 6\\ 6 & 9\end{bmatrix}$$

Dimension Reduction → Unsupervised learning

- Data visualization
  - For easier imagination in 2d or 3d space
- Data compression
  - A high percentage of data variance should be maintained

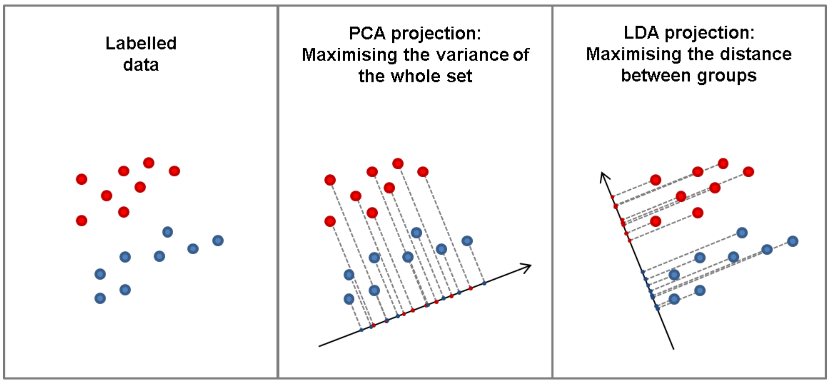
Feature selection vs feature extraction

Feature selection → $\begin{bmatrix}x_1\\ \vdots\\ x_d\end{bmatrix}\rightarrow\begin{bmatrix}x_{i1}\\ \vdots\\ x_{id'}\end{bmatrix}$

Feature extraction → $\begin{bmatrix}x_1\\ \vdots\\ x_d\end{bmatrix}\rightarrow f\left(\begin{bmatrix}x_1\\ \vdots\\ x_d\end{bmatrix}\right)\rightarrow\begin{bmatrix}y_1\\ \vdots\\ y_{d'}\end{bmatrix}$

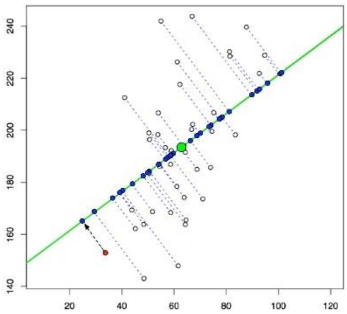
PCA vs LDA

Image source: https://vivekmuraleedharan73.medium.com/what-is-linear-discriminant-analysis-lda-7e33ff59020a



PCA → minimizing the distances of point projection on a line (PC1) → $d_1^2 + d_2^2 + \cdots + d_m^2$

Image source: https://programmathically.com/principal-components-analysis-explained-for-dummies/



Finding principle component 1 and PC2:

Step1: Centered

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \xRightarrow{zero-centered} \begin{cases} \overline{X_1} = (1+3+5) \div 3 = 3 \\ \overline{X_2} = (2+4+6) \div 3 = 4 \end{cases} \rightarrow \begin{bmatrix} -2 & -2 \\ 0 & 0 \\ 2 & 2 \end{bmatrix}$$

Step2: Covariance

$$\text{Covariance matrix} \rightarrow C = \frac{1}{3-1} X^T X = \frac{1}{2} \begin{bmatrix} -2 & 0 & 2 \\ -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} -2 & -2 \\ 0 & 0 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$$

Step3: Eigenvector and Eigenvalue

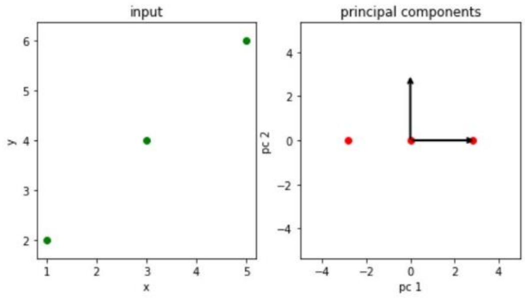
$$\begin{vmatrix} 4-\lambda & 4 \\ 4 & 4-\lambda \end{vmatrix} = 0 \rightarrow (4-\lambda)^2 = 16 \rightarrow \begin{cases} \lambda_1 = 8 \rightarrow v_1 = \begin{bmatrix} +1 \\ +1 \end{bmatrix} = \begin{bmatrix} \dfrac{1}{\sqrt{1^2+1^2}} \\ \dfrac{1}{\sqrt{1^2+1^2}} \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} \\ \lambda_2 = 0 \rightarrow v_2 = \begin{bmatrix} -1 \\ +1 \end{bmatrix} = \begin{bmatrix} \dfrac{-1}{\sqrt{-1^2+1^2}} \\ \dfrac{1}{\sqrt{-1^2+1^2}} \end{bmatrix} = \begin{bmatrix} -0.7 \\ 0.7 \end{bmatrix} \end{cases}$$

$$\lambda_1 > \lambda_2 \to \begin{cases} v_1 \to PC1 \\ v_2 \to PC2 \end{cases}$$

$$V = \begin{bmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \end{bmatrix} \to \begin{cases} PC1:\ y = mx \to y = \dfrac{0.7}{0.7}x \to y = x \\ PC2:\ y = mx \to y = \dfrac{-0.7}{0.7}x \to y = -x \end{cases}$$

$$p = X.V = \begin{bmatrix} -2 & -2 \\ 0 & 0 \\ 2 & 2 \end{bmatrix} \times \begin{bmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \end{bmatrix} = \begin{bmatrix} -2.8 & 0 \\ 0 & 0 \\ 2.8 & 0 \end{bmatrix}$$

We can remove vectors with low Eigenvalue



t-SNE:

7- Clustering

Assigning samples to clusters

- High intra-cluster similarity: cohesive within clusters
- Low inter-cluster similarity: distinctive between clusters

Applications:

- Social networks users
- Information retrieval (news categories)
- Marketing (customers order history)

Clustering algorithms:

- Partitioning
- Hierarchical
- Density

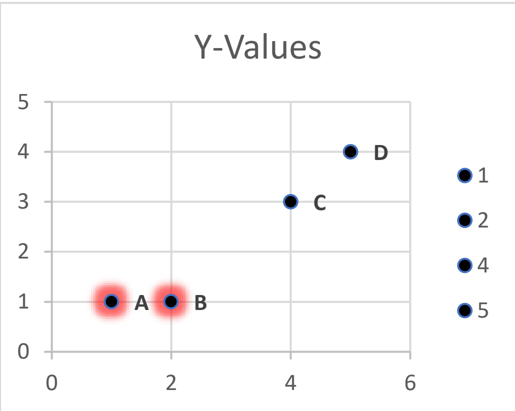
K-Means:

Data:

	X	Y
A	1	1
B	2	1
C	4	3
D	5	4

Number of clusters = 2

A and B are considered as center of clusters 1 and 2.



$C \rightarrow Center\ 1: \sqrt{(4-1)^2 + (3-1)^2} = 3.6$

$C \rightarrow Center\ 2: \sqrt{(4-2)^2 + (3-1)^2} = 2.8$

=====================

$D \rightarrow Center\ 1: \sqrt{(5-1)^2 + (4-1)^2} = 5$

$D \rightarrow Center\ 2: \sqrt{(5-2)^2 + (4-1)^2} = 4.2$

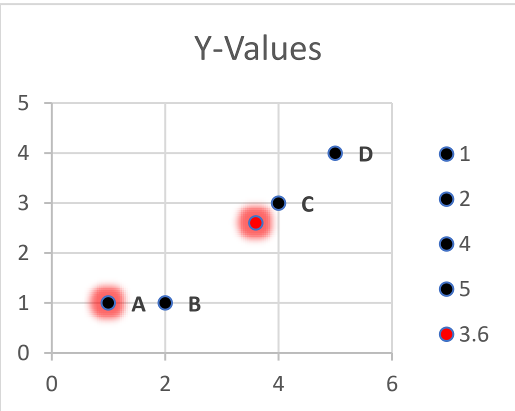
=====================

cluster 1 = {**A**}

cluster 2 = {**B**, C, D}

center of cluster 1 → $(1, 1)$

center of cluster 2 → $\left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = \left(\frac{11}{3}, \frac{8}{3}\right)$



$B \rightarrow Center\ 1: \sqrt{(2-1)^2 + (1-1)^2} = 1$

$B \rightarrow Center\ 2: \sqrt{(3.6-2)^2 + (2.6-1)^2} = 2.6$

=====================

$C \rightarrow Center\ 1: \sqrt{(4-1)^2 + (3-1)^2} = 5$

$C \rightarrow Center\ 2: \sqrt{(3.6-4)^2 + (2.6-3)^2} = 3.6$

=====================

$D \rightarrow Center\ 1: \sqrt{(5-1)^2 + (4-1)^2} = 5$

$D \rightarrow Center\ 2: \sqrt{(3.6-5)^2 + (2.6-4)^2} = 1.9$

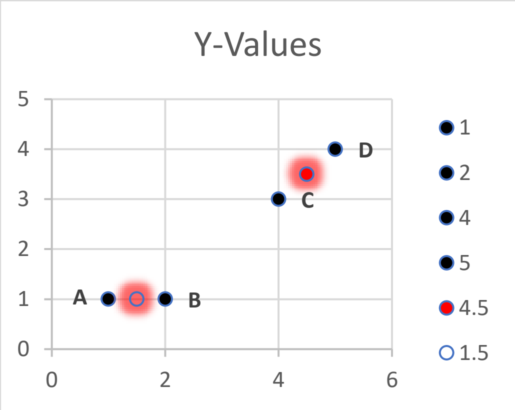
=====================

cluster 1 = {A, B}

cluster 2 = {C, D}

center of cluster 1 → $\left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1)$

center of cluster 2 → $\left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$



$A \rightarrow Center\ 1: \sqrt{(1.5-1)^2 + (1-1)^2} = 0.5$
$A \rightarrow Center\ 2: \sqrt{(4.5-1)^2 + (3.5-1)^2} = 4.3$
======================

$B \rightarrow Center\ 1: \sqrt{(2-1.5)^2 + (1-1)^2} = 0.5$
$B \rightarrow Center\ 2: \sqrt{(2-4.5)^2 + (1-3.5)^2} = 3.5$
======================

$C \rightarrow Center\ 1: \sqrt{(4-1.5)^2 + (3-1)^2} = 3.2$
$C \rightarrow Center\ 2: \sqrt{(4-4.5)^2 + (3-3.5)^2} = 0.7$
======================

$D \rightarrow Center\ 1: \sqrt{(5-1.5)^2 + (4-1)^2} = 4.6$
$D \rightarrow Center\ 2: \sqrt{(4.5-5)^2 + (3.5-4)^2} = 0.7$
======================

cluster 1 = {A, B}

cluster 2 = {C, D}

center of cluster 1 → $\left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1)$

center of cluster 2 → $\left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (\ 4.5\ ,\ \ 3.5\ )$

Lloyd's method:

- **Input**: a set of data point and an integer K
- **Output**: K centroids
- **Purpose**: choose centroids to minimize → $\sum_{i=1}^{N} \min d^2(x^{(i)}, c_j)$

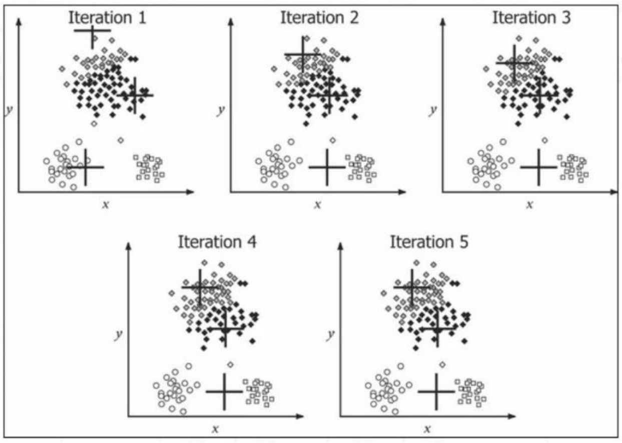
$$\sum_{i=1}^{N} \min d^2(x^{(i)}, c_j) \rightarrow \sum_{i=1}^{N} \min \left\| x^{(i)} - c_j \right\|^2$$

$$Cost\ function \rightarrow J(C) = \sum_{j=1}^{K} \sum_{i=1}^{N} \min \left\| x^{(i)} - c_j \right\|^2 \rightarrow \begin{cases} 1st\ sigma : for\ each\ cluster \\ 2nd\ sigma : for\ each\ data\ point \end{cases}$$

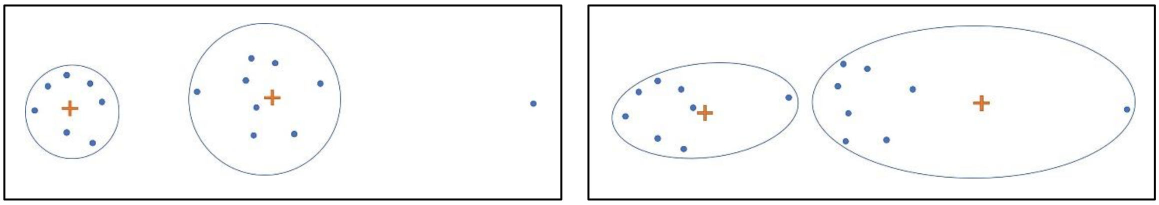
Clustering limitations:

1. Sensitivity to initial seeds (local minimum)

Image source: https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781784397180/6/ch06lvl1sec112/the-drawbacks-of-k-means



2. Sensitive to outliers



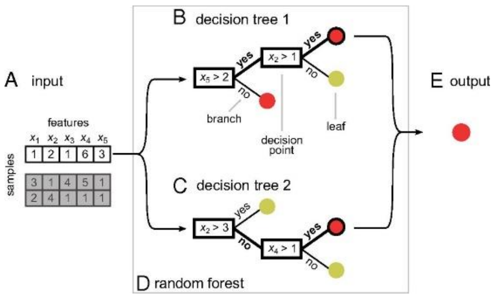
8- Ensemble learning

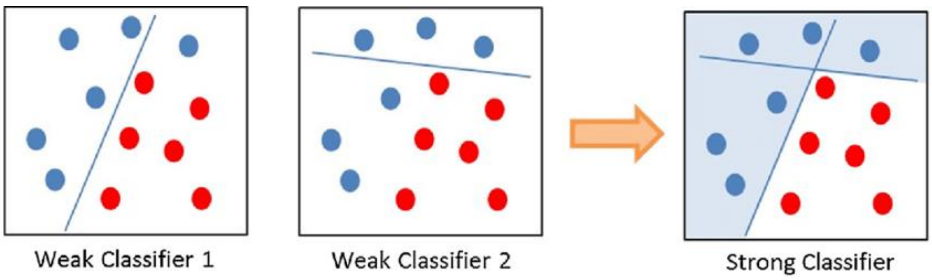
Combine weak learners to create strong learner

Weak learner → a learner with a little better than accidently classifier

Image source: https://www.pnas.org/doi/10.1073/pnas.1800256115

SVM





Bagging : Bootstrap aggregating

Bootstrap → for each bootstrap we will have a model

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Original Dataset

x_8	x_8	x_3	x_5	x_5	x_6	x_7	x_8	x_5	x_5	Bootstrap 1
x_{10}	x_2	x_7	x_5	x_5	x_6	x_6	x_5	x_9	x_2	Bootstrap 2

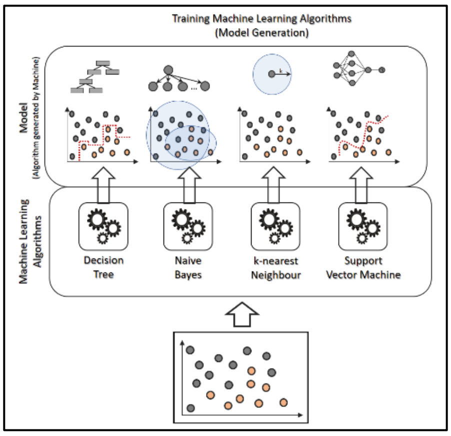
Repeatation: allowed

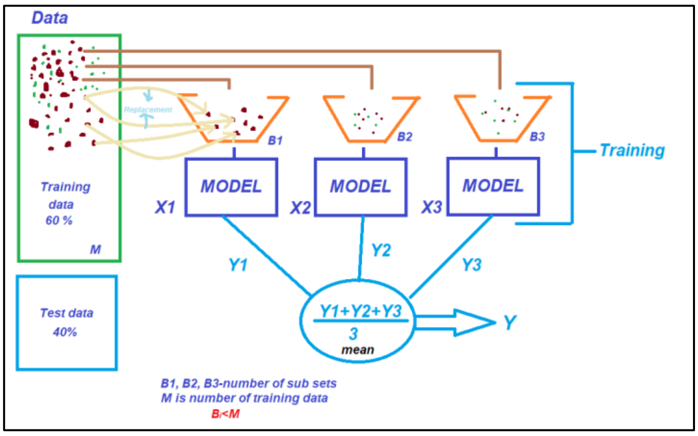
Size: as same as original dataset

Selection: randomly

Image source: https://data-science-blog.com/blog/2017/12/03/ensemble-learning/

Image source: https://medium.com/@nadir.tariverdiyev/machine-learning-algorithms-ensemble-methods-bagging-boosting-and-random-forests-7d3df7adfab8





Random forest:

Bagging on decision trees

Random forest → features will be selected randomly

Available for regression and classification

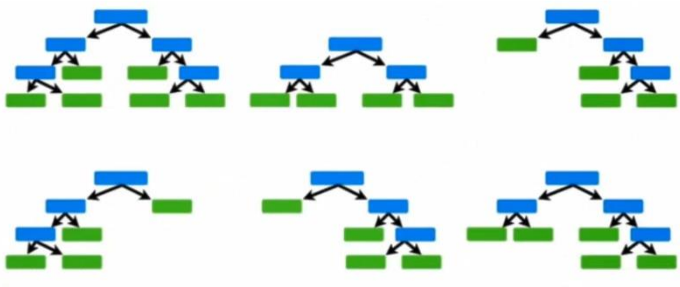
Result → majority voting

Bootstrap 1 → row2, row1, row4, row4

Out of bag → row3 : good sample for evaluation

F1	F2	F3	F4	class
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES

F1	F2	F3	F4	class
YES	YES	YES	180	YES
NO	NO	NO	125	NO
YES	NO	YES	167	YES
YES	NO	YES	167	YES



Number of features → 4

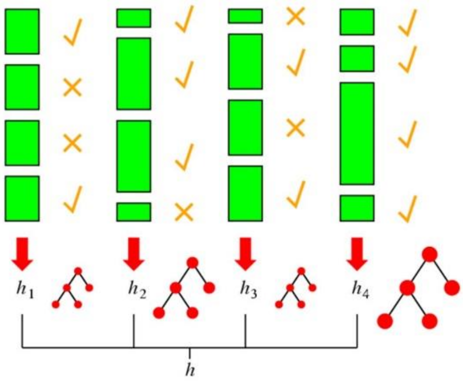
Feature for root → $\sqrt{4} = 2 \rightarrow randomly\ (f_2, f_3) \rightarrow\ f_2: better\ division$

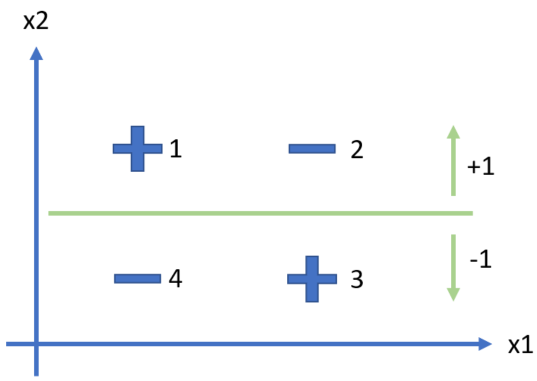
Next selection → $among\ (f_1, f_3, f_4) \rightarrow randomle\ \ (f_1, f_4)$

Test → $input\ (f_1, f_2, f_3, f_4) \rightarrow voting\ among\ 6\ decision\ tree\ models \rightarrow \begin{cases} 5\ models \rightarrow yes \\ 1\ model \rightarrow No \end{cases} \rightarrow$ $Result: yes$

Boosting:

Image source: https://courses.cs.washington.edu/courses/cse455/16wi/notes/15_FaceDetection.pdf





$H_m(x) = \alpha_1 h_1(x) + \cdots + \alpha_m h_m(x)$

$\hat{y} = sign(H_m(x))$

AdaBoost:

- Purpose: minimizing cost function
- Wrong predicted items will have higher weights
- Classifier could be complex but fortunately it is not overfitted

- Base classifier (weak learner) could be decision tree with limited depth, neural network, decision stump

Decision stump:

- A model consisting of a one-level decision tree
- Includes one internal node which immediately connected to the terminal node
- Prediction is based on the value of just a single input feature

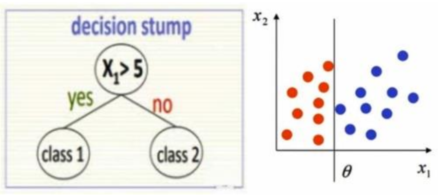
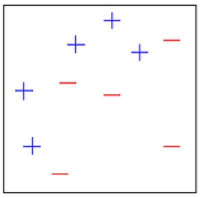
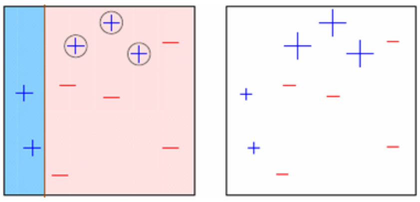


Image source: https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=lectures.boosting



Round 1:

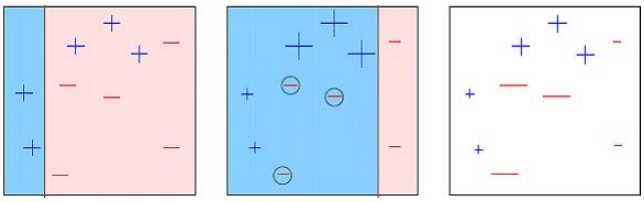
The weight for all data points → $\frac{1}{10}$



Round 2:

$H_1(x) = sign(3 - x_1)$

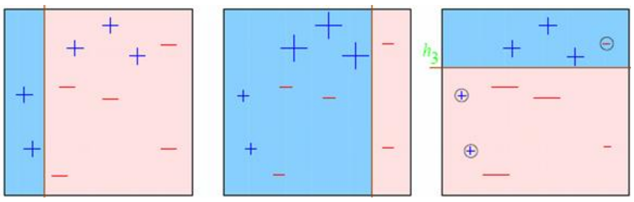
Increase the weights of wrong predicted items.



Round 3:

$H_2(x) = sign(7 - x_1)$

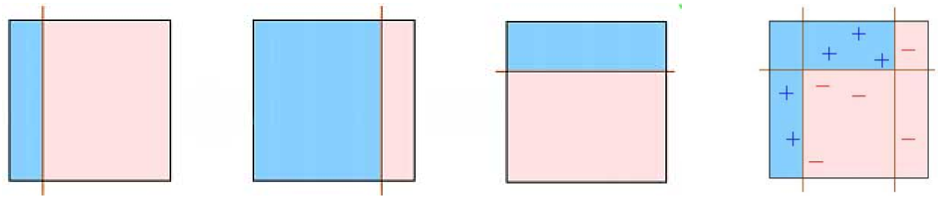
Increase the weights of wrong predicted items.



Round 3:

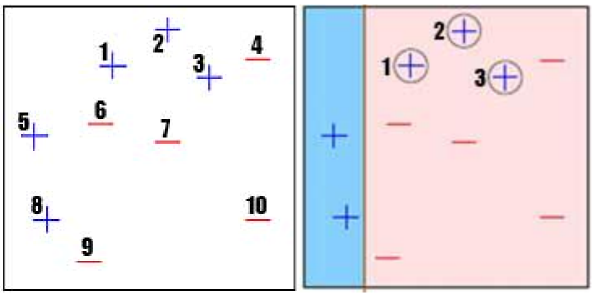
$H_3(x) = sign(x_2 - 4)$

Increase the weights of wrong predicted items.



How to find weights?

Round 1:



1	2	3	4	5	6	7	8	9	10
0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07
0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07

Summation of wrong predicted data → $J_1 = \frac{1}{10} + \frac{1}{10} + \frac{1}{10} = \frac{3}{10}$

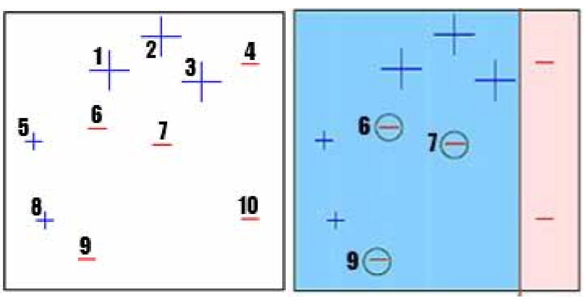
Error → $\varepsilon_1 = \frac{J_1}{weights\ summation} = \frac{\frac{3}{10}}{1} = \frac{3}{10}$

Coefficient of classifier → $\alpha_1 = \frac{1}{2} ln \frac{1-\varepsilon_1}{\varepsilon_1} = 0.42$

New wights → $\begin{cases} if\ classified\ correctly \\ if\ classified\ wrongly \end{cases} \rightarrow \begin{cases} weight \times e^{-\alpha_1} \\ weight \times e^{+\alpha_1} \end{cases} \rightarrow \begin{cases} 0.1 \times e^{-0.42} = 0.07 \\ 0.1 \times e^{+0.42} = 0.15 \end{cases}$

Normalization factor → $Z_1 = 3 \times (0.15) + 7 \times (0.07) = 0.94 \rightarrow \begin{cases} 0.15 \div 0.94 = 0.17 \\ 0.07 \div 0.94 = 0.07 \end{cases}$

Round 2:



1	2	3	4	5	6	7	8	9	10
0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07
0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04

0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05

Summation of wrong predicted data → $J_2 = \frac{7}{100} + \frac{7}{100} + \frac{7}{100} = \frac{21}{100}$

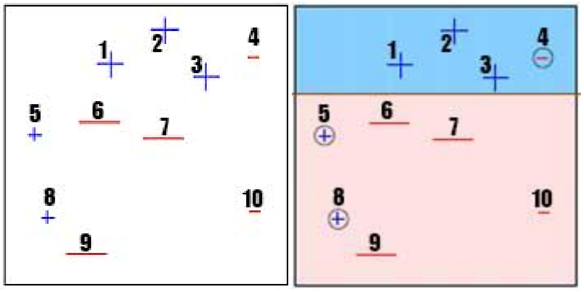
Error → $\varepsilon_1 = \frac{J_2}{weights\ summation} \cong \frac{21}{100}$

Coefficient of classifier → $\alpha_2 = \frac{1}{2} ln \frac{1-\varepsilon_2}{\varepsilon_2} \cong 0.65$

New wights → $\begin{cases} if\ classified\ correctly \rightarrow weight \times e^{-\alpha_1} \rightarrow \begin{cases} 0.17 \times e^{-0.65} \cong 0.09 \\ 0.07 \times e^{-0.65} \cong 0.04 \end{cases} \\ if\ classified\ wrongly\ \rightarrow weight \times e^{+\alpha_1} \rightarrow 0.07 \times e^{+0.65} \cong 0.14 \end{cases}$

Normalization factor → $Z_1 = 4 \times (0.04) + 3 \times (0.09) + 3 \times (0.14) = 0.82 \rightarrow \begin{cases} 0.04 \div 0.82 = 0.05 \\ 0.09 \div 0.82 = 0.11 \\ 0.14 \div 0.82 = 0.17 \end{cases}$

Round 3:



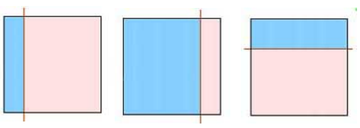
1	2	3	4	5	6	7	8	9	10
0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05
0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02

Summation of wrong predicted data → $J_3 = \frac{5}{100} + \frac{5}{100} + \frac{5}{100} = \frac{15}{100}$

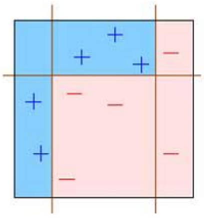
Error → $\varepsilon_1 = \frac{J_3}{weights\ summation} \cong \frac{14}{100}$

Coefficient of classifier → $\alpha_2 = \frac{1}{2} ln \frac{1-\varepsilon_3}{\varepsilon_3} \cong 0.92$

New wights → $\begin{cases} if\ classified\ correctly \rightarrow weight \times e^{-\alpha_1} \rightarrow \begin{cases} 0.05 \times e^{-0.92} \cong 0.02 \\ 0.11 \times e^{-0.92} \cong 0.04 \\ 0.17 \times e^{-0.92} \cong 0.07 \end{cases} \\ if\ classified\ wrongly\ \rightarrow weight \times e^{+\alpha_1} \rightarrow 0.05 \times e^{+0.92} \cong 0.11 \end{cases}$



$H_{final} = sgn(0.42(model1) + 0.65 + 0.92)$



9- Decision tree

Inner node → feature

Leaves → label

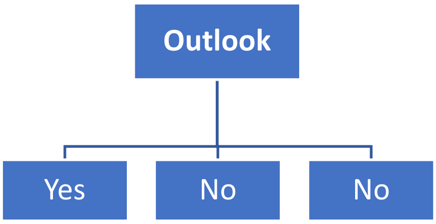
Induction and deduction

$$\text{Training data} \xrightarrow{induction} model\ (decision\ tree)$$

$$model \xrightarrow{deduction} prediction$$

Example:

Day	Wind	Temperature	Outlook	Humidity	Class
1	Weak	Hot	Sunny	High	Yes
2	Strong	Hot	Sunny	High	Yes
3	Weak	Hot	Rain	High	No
4	Weak	Mid	Overcast	High	Yes
5	Strong	Cold	Rain	Normal	No
6	Weak	Cold	Overcast	Normal	Yes
7	Strong	Cold	Rain	Normal	No
8	Weak	Mid	Sunny	Normal	Yes
9	Weak	Cold	Sunny	Normal	Yes
10	Strong	Mid	Overcast	Normal	Yes
11	Weak	Mid	Sunny	High	No
12	Strong	Mid	Rain	High	No
13	Weak	Hot	Overcast	Normal	Yes
14	Weak	Cold	Rain	High	Yes



Entropy:

Low entropy → high purity

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$\begin{cases} Class: Yes \to 9 \\ Class: No \to 5 \end{cases} \to p_{\oplus} = \frac{9}{15} \mid p_{\ominus} = \frac{4}{15} \to Entropy([9+, 4-]) = -\left(\frac{9}{15} log \frac{9}{15} + \frac{4}{15} log \frac{4}{15}\right)$$
$$= 0940$$

Gain:

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

Wind:

$$Wind \to \begin{cases} Weak \to 9 \to 7: Yes \mid 2: No \\ Strong \to 5 \to 2: Yes \mid 3: No \end{cases} \to \begin{cases} -\left(\frac{7}{9} log \frac{7}{9} + \frac{2}{9} log \frac{2}{9}\right) = 0.764 \\ -\left(\frac{2}{5} log \frac{2}{5} + \frac{3}{5} log \frac{3}{5}\right) = 0.970 \end{cases}$$

$$Gain(S, A) = Entropy(training\ data) - \left(\frac{9}{14} Entropy(weak) + \frac{5}{14} Entropy(strong)\right)$$

$$= 0.940 - (0.491 + 0.346) = 0.940 - 0.837 = 0.102$$

Temperature:

$$Temperature \to \begin{cases} Hot \to 4 \to 3: Yes \mid 1: No \\ Mid \to 5 \to 3: Yes \mid 2: No \\ Cold \to 5 \to 3: Yes \mid 2: No \end{cases} \to \begin{cases} -\left(\frac{3}{4} log \frac{3}{4} + \frac{1}{4} log \frac{1}{4}\right) = 0.811 \\ -\left(\frac{3}{5} log \frac{3}{5} + \frac{2}{5} log \frac{2}{5}\right) = 0.970 \\ -\left(\frac{3}{5} log \frac{3}{5} + \frac{2}{5} log \frac{2}{5}\right) = 0.970 \end{cases}$$

$$Gain(S, A) = Entropy(training\ data) - \left(\frac{4}{14} Entropy(H) + \frac{5}{14} Entropy(M) + \frac{5}{14} Entropy(C)\right)$$

$$= 0.940 - (0.231 + 0.346 + 0.346) = 0.940 - 0.932 = 0.008$$

Outlook:

$$Outlook \rightarrow \begin{cases} Sunny \rightarrow 5 \rightarrow 4{:}\, Yes \mid 1{:}\, No \\ Overcast \rightarrow 4 \rightarrow 4{:}\, Yes \mid 0{:}\, No \\ Rain \rightarrow 5 \rightarrow 1{:}\, Yes \mid 4{:}\, No \end{cases} \rightarrow \begin{cases} -\left(\frac{4}{5} log \frac{4}{5} + \frac{1}{5} log \frac{1}{5}\right) = 0.722 \\ -\left(\frac{4}{4} log \frac{4}{4} + \frac{0}{4} log \frac{0}{4}\right) = 0 \\ -\left(\frac{1}{5} log \frac{1}{5} + \frac{4}{5} log \frac{4}{5}\right) = 0.722 \end{cases}$$

$$Gain(S, A) = Entropy(training\ data) - \left(\frac{5}{14} Entropy(S) + \frac{4}{14} Entropy(O) + \frac{5}{14} Entropy(R)\right)$$

$$= 0.940 - (0.258 + 0 + 0.258) = 0.940 - 0.516 = 0.424$$

Outlook:

$$Outlook \rightarrow \begin{cases} Sunny \rightarrow 5 \rightarrow 4{:}\, Yes \mid 1{:}\, No \\ Overcast \rightarrow 4 \rightarrow 4{:}\, Yes \mid 0{:}\, No \\ Rain \rightarrow 5 \rightarrow 1{:}\, Yes \mid 4{:}\, No \end{cases} \rightarrow \begin{cases} -\left(\frac{4}{5} log \frac{4}{5} + \frac{1}{5} log \frac{1}{5}\right) = 0.722 \\ -\left(\frac{4}{4} log \frac{4}{4} + \frac{0}{4} log \frac{0}{4}\right) = 0 \\ -\left(\frac{1}{5} log \frac{1}{5} + \frac{4}{5} log \frac{4}{5}\right) = 0.722 \end{cases}$$

$$Gain(S, A) = Entropy(training\ data) - \left(\frac{5}{14} Entropy(S) + \frac{4}{14} Entropy(O) + \frac{5}{14} Entropy(R)\right)$$

$$= 0.940 - (0.258 + 0 + 0.258) = 0.940 - 0.516 = 0.424$$

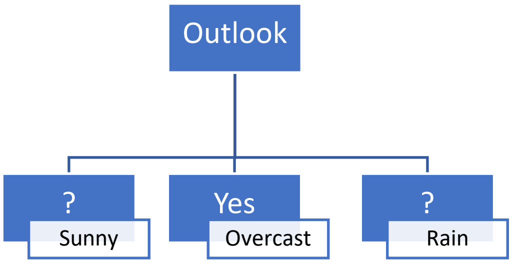
Humidity:

$$Hunidity \rightarrow \begin{cases} High \rightarrow 7 \rightarrow 5{:}\, Yes \mid 2{:}\, No \\ Normal \rightarrow 7 \rightarrow 4{:}\, Yes \mid 3{:}\, No \end{cases} \rightarrow \begin{cases} -\left(\frac{5}{7} log \frac{5}{7} + \frac{2}{7} log \frac{2}{7}\right) = 0.863 \\ -\left(\frac{4}{7} log \frac{4}{7} + \frac{3}{7} log \frac{3}{7}\right) = 0.985 \end{cases}$$

$$Gain(S, A) = Entropy(training\ data) - \left(\frac{7}{14} Entropy(High) + \frac{7}{14} Entropy(Normal)\right)$$

$$= 0.940 - (0.431 + 0.493) = 0.940 - 0.924 = 0.016$$

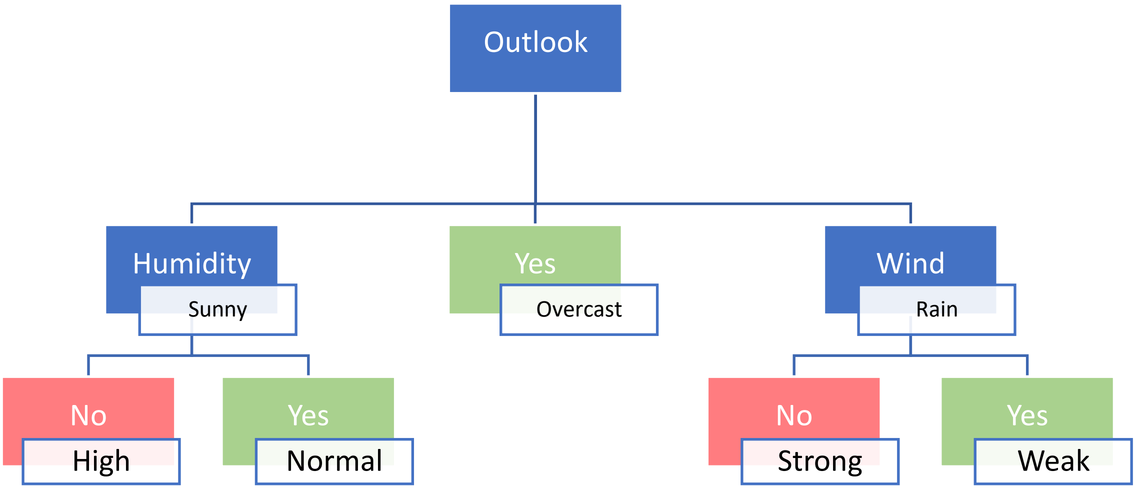
Wind	Temperature	Outlook	Humidity
0.102	0.008	**0.424**	0.016



$$Gain(S, Humidity) = 0.97 - \left(\frac{3}{5}\right) * 0 - \left(\frac{2}{5}\right) * 0$$
$$= \mathbf{0.97}$$

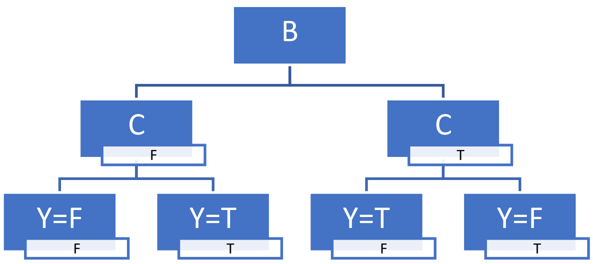
$$G(S, Temp) = 0.97 - \left(\frac{2}{5}\right) * 0 - \left(\frac{2}{5}\right) * 1 - \left(\frac{1}{5}\right) * 0$$
$$= 0.57$$

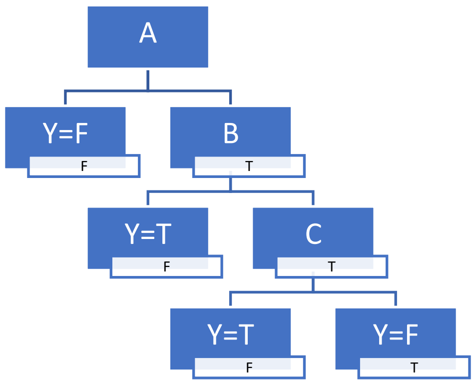
$$Gain(S, Wind) = 0.97 - \left(\frac{2}{5}\right) * 1 - \left(\frac{3}{5}\right) * .918 = 0.019$$



Greedy vs Optimal:

A	C	C	Y
F	F	F	F
T	F	T	T
T	T	F	T
T	T	T	F





Gini Coefficient:

Gini coefficient is fined for all features → The feature with the **lowest Gini coefficient** will be selected

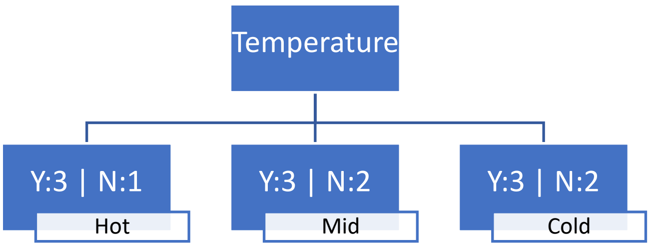
$$Gini(t) = 1 - \sum_{i=1}^{c} p_i^2$$

- T → node

$$Ginisplit(A) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

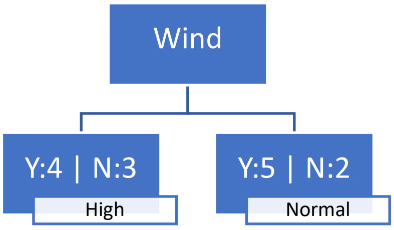
- Find Gini for all splits and choose the lowest one

Day	Wind	Temperature	Outlook	Humidity	Class
1	Weak	Hot	Sunny	High	Yes
2	Strong	Hot	Sunny	High	Yes
3	Weak	Hot	Rain	High	No
4	Weak	Mid	Overcast	High	Yes
5	Strong	Cold	Rain	Normal	No
6	Weak	Cold	Overcast	Normal	Yes
7	Strong	Cold	Rain	Normal	No
8	Weak	Mid	Sunny	Normal	Yes
9	Weak	Cold	Sunny	Normal	Yes
10	Strong	Mid	Overcast	Normal	Yes
11	Weak	Mid	Sunny	High	No
12	Strong	Mid	Rain	High	No
13	Weak	Hot	Overcast	Normal	Yes
14	Weak	Cold	Rain	High	Yes



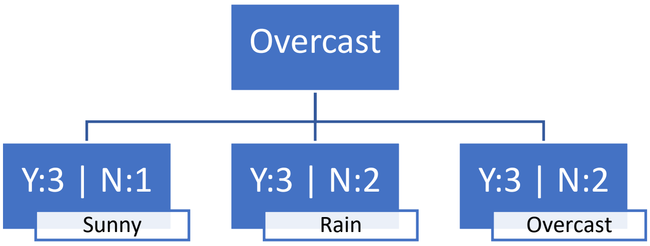
$$\begin{cases} Hot \rightarrow 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 1 - 0.5625 - 0.0625 = 0.375 \\ Mid \rightarrow 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 1 - 0.36 - 0.16 = 0.48 \\ Cold \rightarrow 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 1 - 0.36 - 0.16 = 0.48 \end{cases}$$

$$GiniSplit(Temperature) = \frac{4}{14} \times 0.375 + \frac{5}{14} \times 0.48 + \frac{5}{14} \times 0.48$$



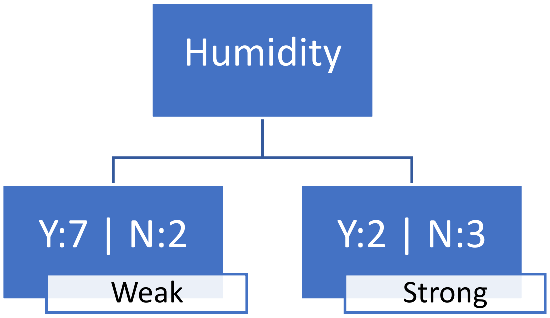
$$\begin{cases} Weak \rightarrow 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 1 - 0.326 - 0.183 = 0.491 \\ Strong \rightarrow 1 - \left(\frac{5}{7}\right)^2 - \left(\frac{2}{7}\right)^2 = 1 - 0.510 - 0.081 = 0.409 \end{cases}$$

$$GiniSplit(Temperature) = \frac{7}{14} \times 0.491 + \frac{7}{14} \times 0.409 = 0.245 + 0.204 = 0.449$$



$$\begin{cases} Sunny \rightarrow 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 1 - 0.409 - 0.001 = 0.59 \\ Rain \rightarrow 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 1 - 0.36 - 0.16 = 0.59 \\ Overcast \rightarrow 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2 = 1 - 1 - 0 = 0 \end{cases}$$

$$GiniSplit(Overcast) = \frac{5}{14} \times 0.59 + \frac{5}{14} \times 0.59 + \frac{4}{14} \times 0 = 0.21 + 0.21 + 0 = 0.42$$



$$\begin{cases} Weak \rightarrow 1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2 = 1 - 0.604 - 0.049 = 0.347 \\ Strong \rightarrow 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 1 - 0.16 - 0.36 = 0.48 \end{cases}$$

$$GiniSplit(Humidity) = \frac{9}{14} \times 0.347 + \frac{5}{14} \times 0.48 = 0.223 + 0.171 = 0.394$$

rule-based representation

$$if(Outlook = Sunny \wedge Hunidity = Normal) \xRightarrow{Then} PlayTennis = YES$$

10- Association rule mining (market-basket analysis)

Unsupervised learning

The effect of existance of an itemset on existance of another itemset

Fine a rule among data

$$Cheese \xrightarrow{then} Milk$$

$$\begin{cases} Support: 10\% \rightarrow the\ percentage\ of\ customers\ that\ they\ have\ bought\ both\ of\ them \\ confidence: 80\% \rightarrow the\ percentage\ of\ the\ cheese\ buyers\ want\ to\ buy\ also\ a\ milk \end{cases}$$

frequent item-sets (frequent patterns):

TID	items
1	{I1, I3, I4}
2	{I2, I3, I5}
3	{I1, I2, I3, I5}
4	{I2, I5}

$$I = \{i_1, i_2, i_3, i_4, i_5\} \rightarrow items\ (n = 5)$$

$D = \{t_1, t_2, t_3, t_4\} \rightarrow transactions\ (m = 4)$

$X \rightarrow Y$:

$$Support = \frac{frq(X,Y)}{N}$$

$frq(X,Y)$: $how\ many\ time\ X\ and\ Y\ happened\ together$

N: $number\ of\ all\ transactions$

$I1 \rightarrow I3$

$$Support = \frac{2}{4} = 50\%$$

$X \rightarrow Y$:

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

$frq(X,Y)$: $how\ many\ time\ X\ and\ Y\ happened\ together$

$frq(X)$: $how\ many\ time\ X\ happened$

$I1 \rightarrow I3$

$$Support = \frac{2}{2} = 100\% \rightarrow whoever\ bought\ I1\ they\ also\ bouhjt\ I3$$

Apriority algorithm:

1- generate frequent item-sets (iteratively) → support >= minsup

2- generate rules (all possible subsets of rules) → confidence >=minconf

Example:

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

minusp = 2

minconf = 70%

1- generate frequent item-sets (iteratively) → support >= minsup

K=1

Itemset	Support
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

K=2

Itemset	Support
{A, B}	4
{A, C}	4
{A, D}	1
{A, E}	2
{B, C}	4
{B, D}	2
{B, E}	2
{C, D}	0
{C, E}	1
{D, E}	0

K=3

Itemset	Support
{A, B, C}	2
{A, B, D}	1
{A, B, E}	2
{A, C, D}	0
{A, C, E}	1
{A, D, E}	0
{B, C, D}	0
{B, C, E}	1
{B, D, E}	0

2- generate rules (all possible subsets of rules) → confidence >=minconf

{A, B, E}

$A \rightarrow \{B, E\}$	$C = \frac{2}{6} < 70\%$
$B \rightarrow \{A, E\}$	$C = \frac{2}{7} < 70\%$
$E \rightarrow \{A, B\}$	$C = \frac{2}{2} > 70\%$
$\{A, B\} \rightarrow E$	$C = \frac{2}{4} < 70\%$
$\{A, E\} \rightarrow B$	$C = \frac{2}{2} > 70\%$
$\{B, E\} \rightarrow A$	$C = \frac{2}{2} > 70\%$

Lift:

Lift > 1 → positive correlation

Lift < 1 → negative correlation

Lift = 1 → independent

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

A → B

All transactions → 100

A: buying video game → 60

B: buying graphic card → 75

A and B → 40

Minsup → 30%

Minconf → 60%

$$Support = \frac{40}{100} = 40\%$$

$$Confidence = \frac{40}{60} = 66\%$$

$$Lift = \frac{\frac{40}{100}}{\frac{60}{100} \times \frac{75}{100}} < 1 \rightarrow negative\ correlation$$