SECTION 2

10. Feature Scaling

- **Normalization** :

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad [0; 1]$$

- **Standardization**

$$X' = \frac{X - \mu}{\sigma_{stdD}} \quad (\mu \leftarrow \text{avg}) \qquad [-3; 3]$$

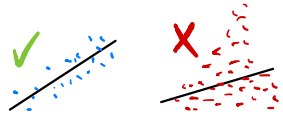
SECTION 3 : DATA PREPROCESSING

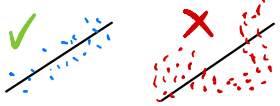
⚠ 1) Splitting the dataset into training set / test set
2) Feature scaling

] This order to not lose data

SECTION 7 : MULTIPLE LINEAR REGRESSION → No need to apply feature Scaling

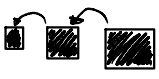
Assumption of Linear Regression

1. Linearity : linear relationship between Y and each X 
2. Homoscedasticy : equal variance 
3. Multivariate Normality : normality of error distribution 
4. Independence : of observations. Includes "no autocorrelation". We don't want to see any kind of pattern in our data.



5. Lack of Multicollinearity : predictors are not correlated with each other.

$X_1 \not\sim X_2$ ✓ $\qquad X_1 \sim X_2$ ✗

Building a model : Backward Elimination 

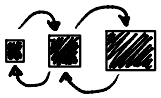
1. Select a significance level to stay in the model (e.g. SL = 0,05)
2. Fit the full model with all possible predictors (All-in)
3. Consider the predictor with the **highest P-Value**. If P > SL, go to STEP 4 otherwise go to FIN (model is ready)
4. Remove the predictor

5. Fit the model without this variable *

Building a model : Forward Selection

1. Select a significance level to enter the model (e.g. SL = 0,05)
2. Fit all simple regression model $y \sim x_n$, select the one with the lowest P-vale
3. Keep this variable and fit all possible models with one extra predictor added to the one (s) you already have
4. Consider the predictor with the lowest P-Value. If P < SL, go to step 3, otherwise go to FIN
   ↳ Keep the previous model

Building a model : Bidirectional Elimination



1. Select a significance level to enter and to stay in the model
   e.g. SL enter = 0.05 , SL stay = 0.05
2. Perform the next step of Forward Selection (new vars must have P<SLenter to enter)
3. Perform ALL the steps of Backward Elimination (old vars must have P< SLstay to stay)
4. No new vars can enter and no old vars can exit.

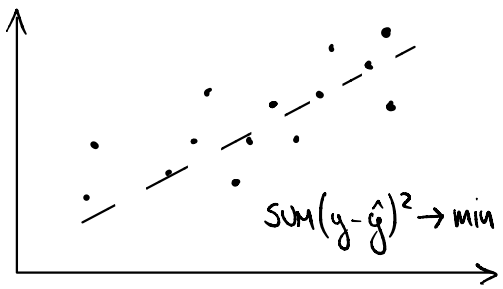
FIN : model is ready

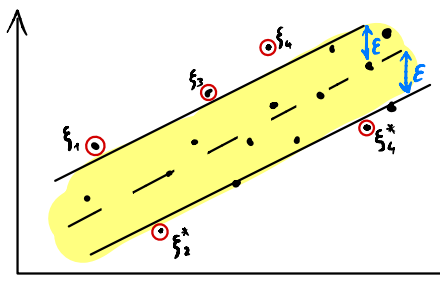
Building a model : All Possible Models (expensive)

1. Select a random of goodness of fit (e.g. Akaike criterion)
2. Construct all possible Regr Models : $2^n - 1$ total comb
3. Select the one with the best criterion

FIN : Model is ready

SECTION 9 : SUPPORT VECTOR REGRESSION (SVR)



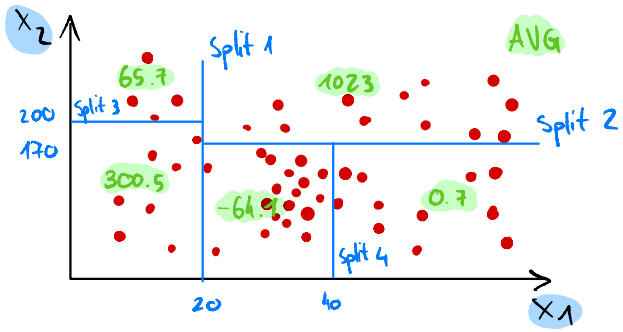


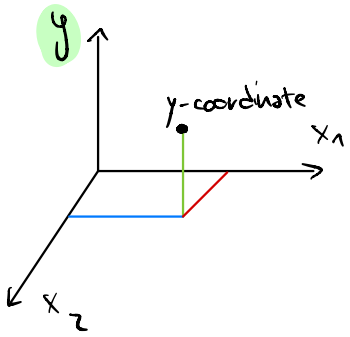
ε : Insensitive Tube
Slack Vars ξ_i and ξ_i^*

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \to \min$$

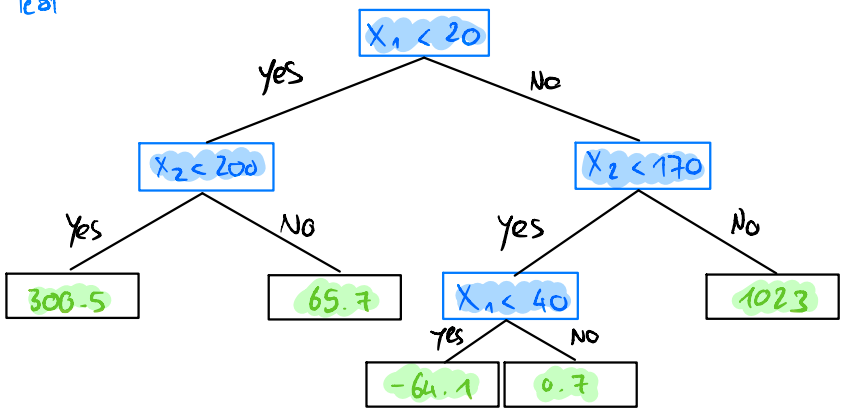
Each point is a vector

SECTION 10 : DECISION TREE REGRESSION





Every square is a leaf



SECTION 11 : RANDOM FOREST REGRESSION

STEP 1 : Pick at random k data points from the Training set

↓

STEP 2 : Build the Decision Tree associated to these k data points

↓

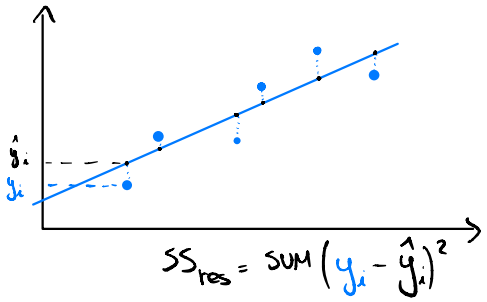
STEP 3 : Choose the number Ntree of trees you want to build and repeat steps 1 and 2

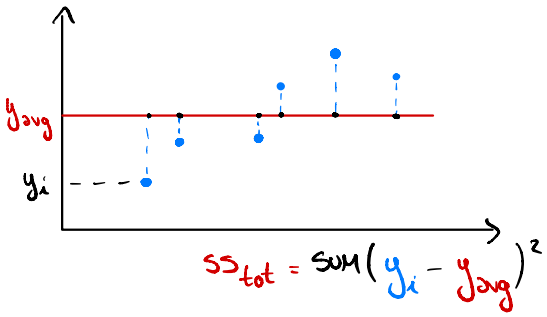
↓

STEP 4 : For a new data point, make each one of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the avg across all of the predicted Y values

SECTION 12 : EVALUATING REGRESSION MODELS PERFORMANCE

▸ R - squared





$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \qquad [0, 1]$$

▸ Adjusted R-Squared

$$Adj\ R^2 = 1 - (1 - R^2) \cdot \frac{n-1}{n-k-1}$$

k : # of indep. vars
n : sample size

SECTION 17 : K-NN

STEP 1 : Choose the number k of neighbors

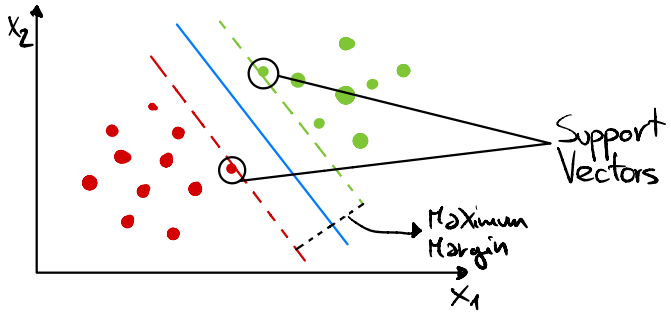
STEP 2 : Take the k nearest neighbors of the new data point, according to the Euclidian distance

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

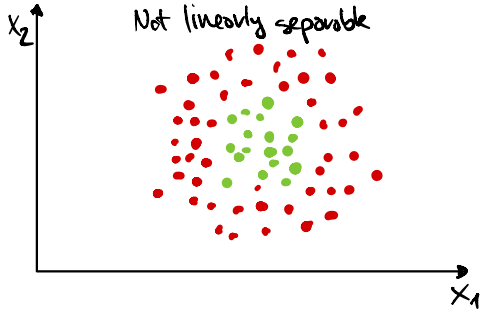
STEP 3 : Among these k neighbors, count the # of data points in each category

STEP 4 : Assign the new data point to the category where you counted the most neighbors

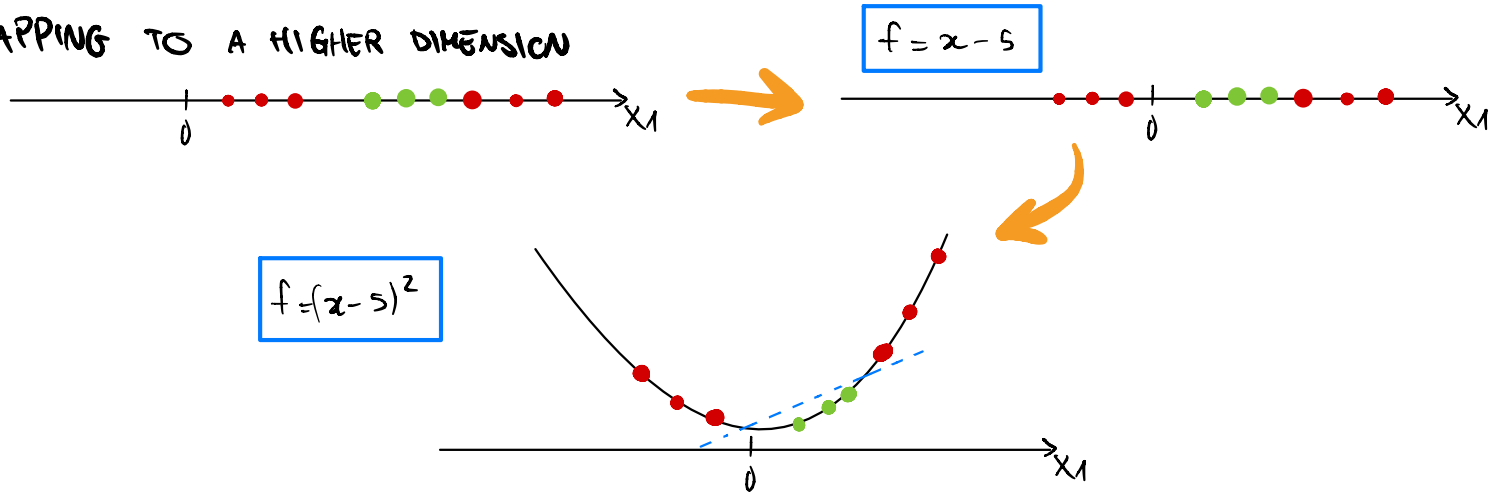
SECTION 18 : SVM



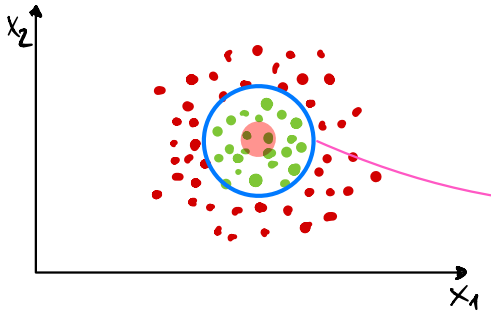
SECTION 19 : KERNEL SVM



▸ MAPPING TO A HIGHER DIMENSION

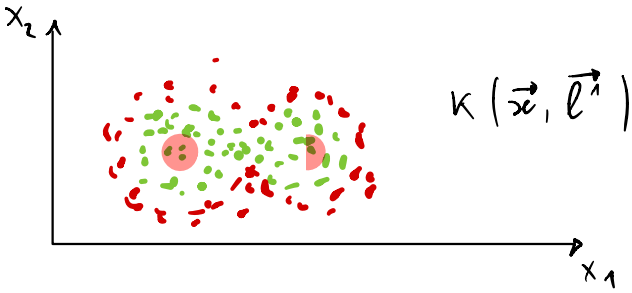


THE KERNEL TRICK



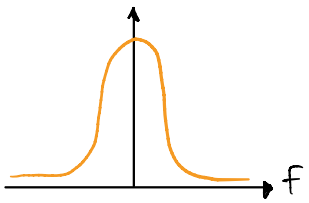
$$K(\vec{x}, \vec{l}^{\,i}) = e^{-\frac{\|\vec{x}-\vec{l}^{\,i}\|^2}{2\sigma^2}}$$

increase/decrease the decision boundary



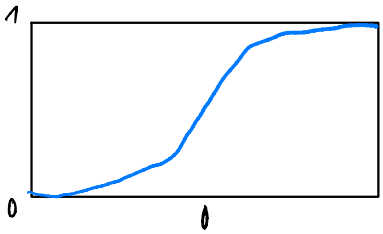
$$K(\vec{x}, \vec{l}^{\,1}) + K(\vec{x}, \vec{l}^{\,2})$$

TYPES OF KERNEL FUNCTIONS



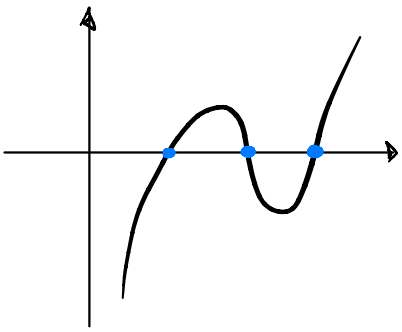
GAUSSIAN RBF KERNEL

$$K(\vec{x}, \vec{l}^{\,i}) = e^{-\frac{\|\vec{x}-\vec{l}^{\,i}\|^2}{2\sigma^2}}$$



SIGMOID KERNEL

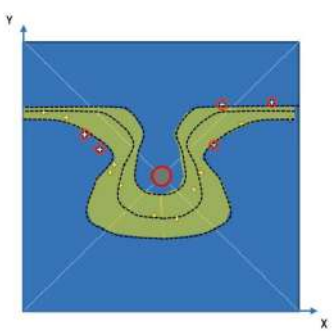
$$K(X,Y) = \tanh(\gamma \cdot X^T Y + r)$$

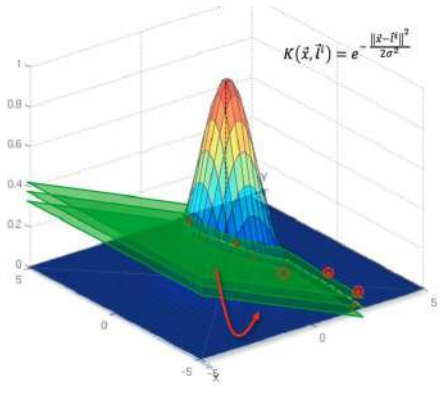


POLYNOMIAL KERNEL

$$K(X,Y) = (\gamma \cdot X^T Y + r)^d, \quad \gamma > 0$$

NON-LINEAR SVM

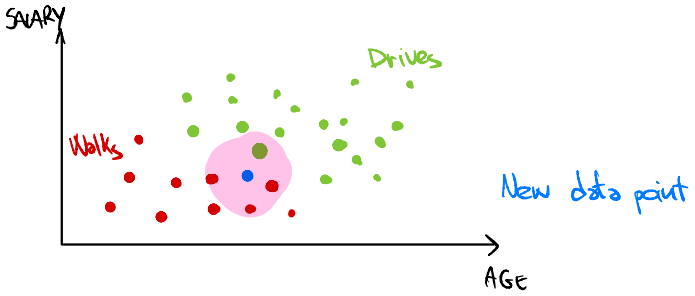




SECTION 20 : NAIVE BAYES

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

▸ NAIVE BAYES CLASSIFIER



$$= \frac{\#\text{ of Similar Obs Among those who walks}}{\text{Tot \# of walks}} \left(\frac{3}{10}\right)$$

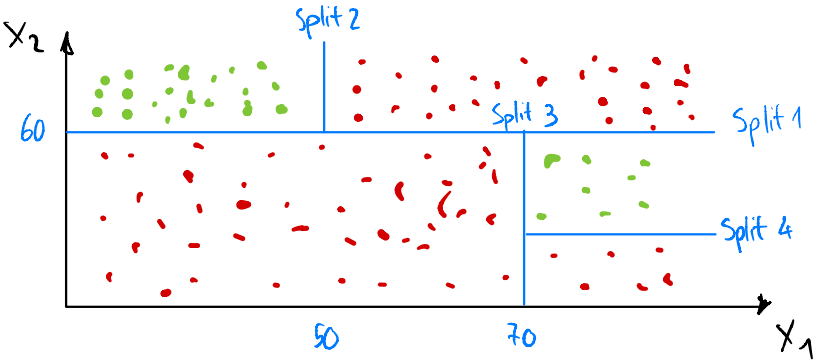
#3 Likelihood

#4 → $P(\text{Walks}|X) = \frac{P(X|\text{Walks}) \cdot P(\text{Walks})}{P(X)}$ ← #1 Prior prob

#2 Marginal Likelihood

$$= \frac{\#\text{ of Similar Observation}}{\text{Total Observation}} \left(\frac{4}{30}\right)$$

SECTION 21 : DECISION TREE



SECTION 22 : RANDOM FOREST CLASSIFICATION

STEP 1 : Pick at random k data points from the Training set

↓

STEP 2 : Build the Decision Tree associated to these k data points

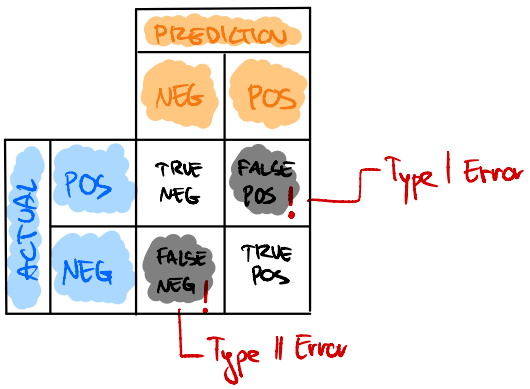
↓

STEP 3 : Choose the number Ntree of trees you want to build and repeat steps 1 and 2

↓

STEP 4 : For a new data point, make each one of your Ntree trees predict the category to which the data point belongs, and assign the new data point to the category that wins the majority vote.

SECTION 23 : CONFUSION MATRIX & ACCURACY RATIOS



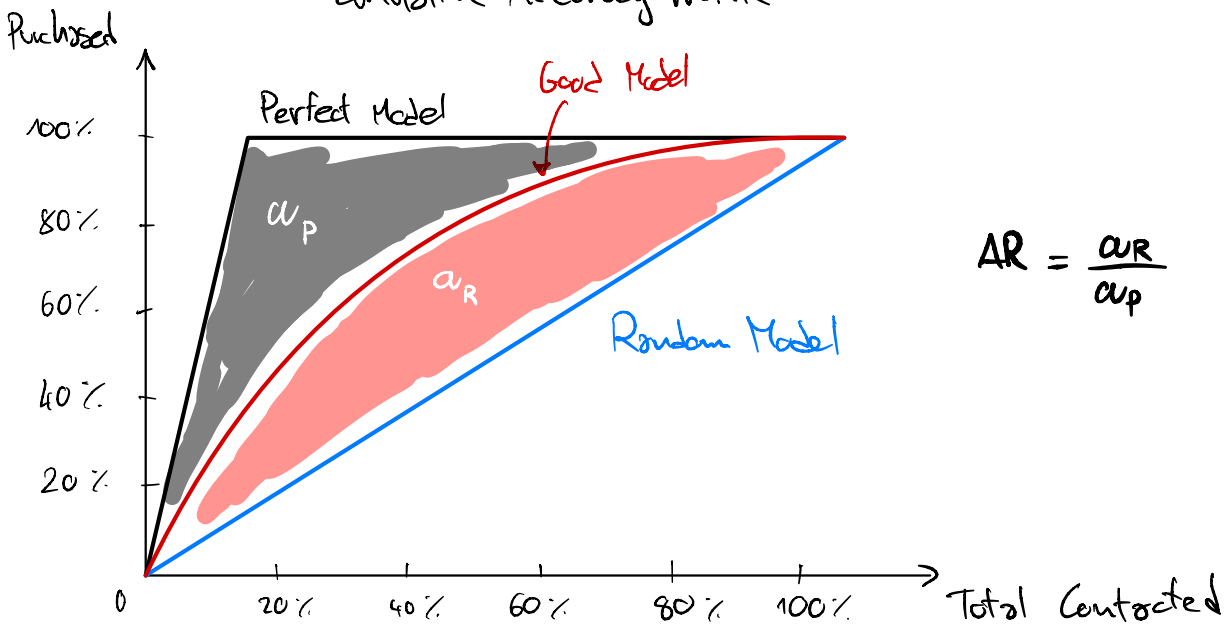
Accuracy Rate & Error Rate

$$AR = \frac{\text{Correct}}{\text{Total}} = \frac{TN+TP}{\text{Total}}$$

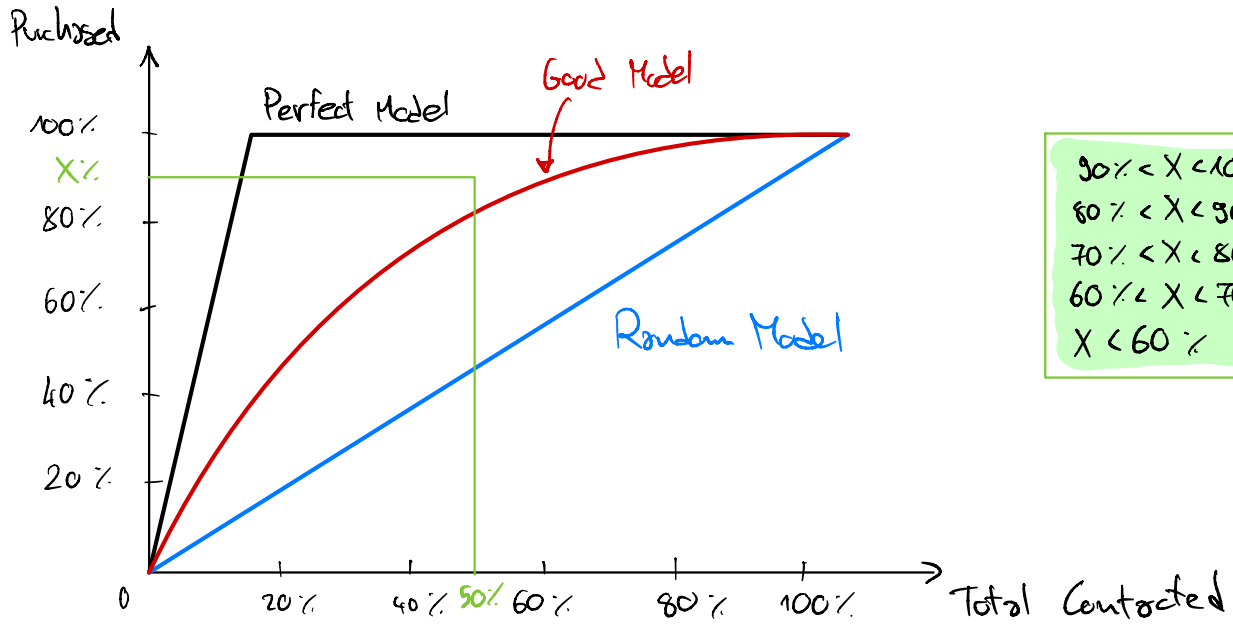
$$ER = \frac{\text{Incorrect}}{\text{Total}} = \frac{FP+FN}{\text{Total}}$$

SECTION 24 : CAP CURVE ANALYSIS

↑ Cumulative Accuracy Profile



$$AR = \frac{a_R}{a_P}$$



$90\% < X < 100\%$	Too Good
$80\% < X < 90\%$	Very Good
$70\% < X < 80\%$	Good
$60\% < X < 70\%$	Poor
$X < 60\%$	Rubbish

SECTION 27 : HIERARCHICAL CLUSTERING

▸ AGGLOMERATIVE HC

STEP 1 : Make each data point a single-point cluster → that forms N clusters

↓

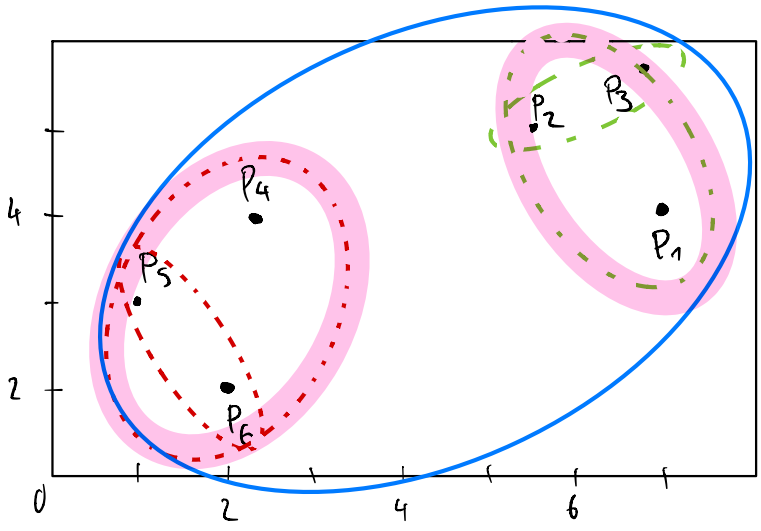
STEP 2 : Take the 2 closest data points and make them one clusters → that forms N-1 clusters

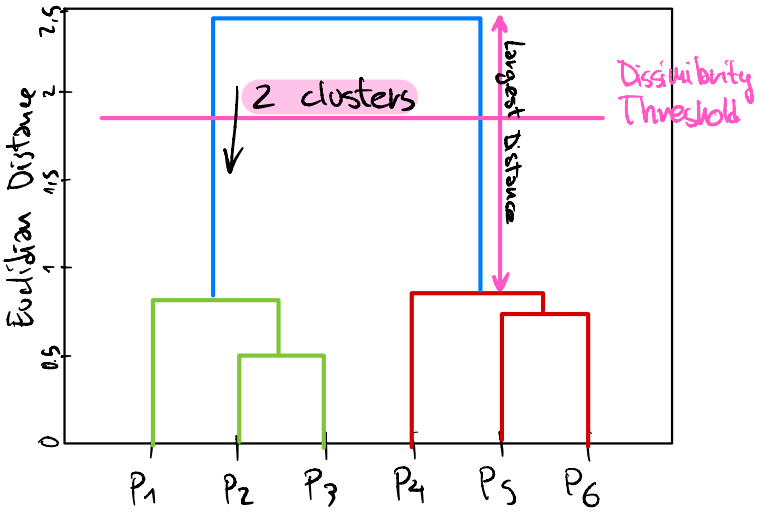
↓

STEP 3 : Take the 2 closest clusters and make them one cluster → that forms N-2 clusters

↓

STEP 4 : Repeat STEP 3 until there is only one cluster





SECTION 29 : APRIORI

Algorithm

STEP 1: Set a minimum support and confidence

↓

STEP 2: Take all the subsets in transactions having higher support than minimum support

↓

STEP 3: Take all the rules of these subsets having higher confidence than minimum confidence

↓

STEP 4: Sort the rules by decreasing lift

SECTION 30 : ECLAT

Algorithm

STEP 1: Set a minimum support

↓

STEP 2: Take all the subsets in transactions having higher support than minimum support

↓

STEP 3: Sort these subsets by decreasing support

SECTION 32 : UPPER CONFIDENCE BOUND (UCB)

The Multi-Armed Bandit Problem

You've got 5 or more or any number of slot machine and you can bet your money in any one of them and you need to **find out how to bet to maximize your returns**. Behind every machine there is a certain **distribution** and that's because you don't know which of these **distribution is optimal**. You need to **combine exploration** of these machine with their **exploitation** in order to **find out which one is the best.**

The modern application of this problem is advertising:

- We have d arms. For example, arms are ads that we display to user each time they connect to a web page.
- Each time a user connects to this web page, that makes a round.
- At each round n, we choose one ad to display to the user.
- At each round n, ad i gives reward $r_i(n) \in \{0, 1\}$: $r_i(n) = 1$ if the user clicked on the ad i, 0 if the user didn't.
- Our goal is to maximize the total reward we get over many rounds.

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i(n)$ - the number of times the ad i was selected up to round n,
- $R_i(n)$ - the sum of rewards of the ad i up to round n.

Step 2. From these two numbers we compute:

- the average reward of ad i up to round n

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

- the confidence interval $[\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)]$ at round n with

$$\Delta_i(n) = \sqrt{\frac{3}{2}\frac{\log(n)}{N_i(n)}}$$

Step 3. We select the ad i that has the maximum UCB $\bar{r}_i(n) + \Delta_i(n)$.

SECTION 33 : THOMPSON SAMPLING

To resolve the Multi-Armed Bandit Problem

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i^1(n)$ - the number of times the ad i got reward 1 up to round n,
- $N_i^0(n)$ - the number of times the ad i got reward 0 up to round n.

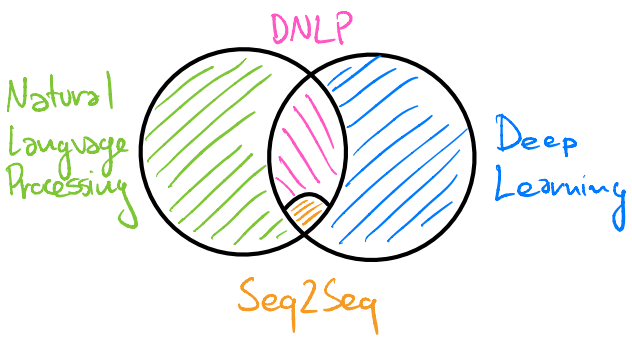
Step 2. For each ad i, we take a random draw from the distribution below:

$$\theta_i(n) = \beta(N_i^1(n) + 1, N_i^0(n) + 1)$$

Step 3. We select the ad that has the highest $\theta_i(n)$.

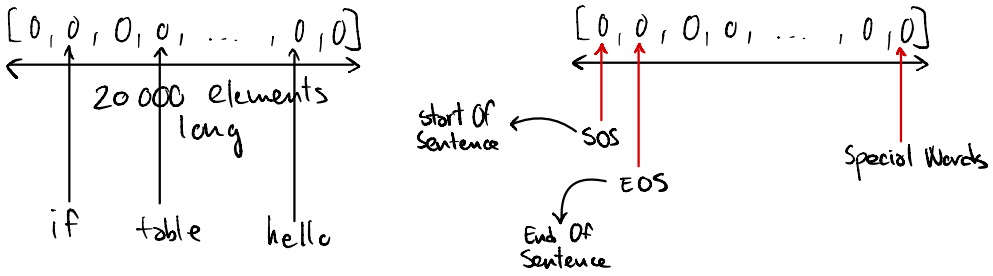
SECTION 34: NATURAL LANGUAGE PROCESSING

- Types of Natural Language Processing



Bag of Words

- Yes / No



Hello Kirill, Checking if you are back to Oz. Let me know if you are around ... Cheers, V

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, ... , 3]

20,000 elements long

Hello Kirill, Checking if you are back to Oz. Let me know if you are around ... Cheers, V

Goal (answer to the question)

[1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, ... , 3] → Yes / No ?

20,000 elements long

Training Data:

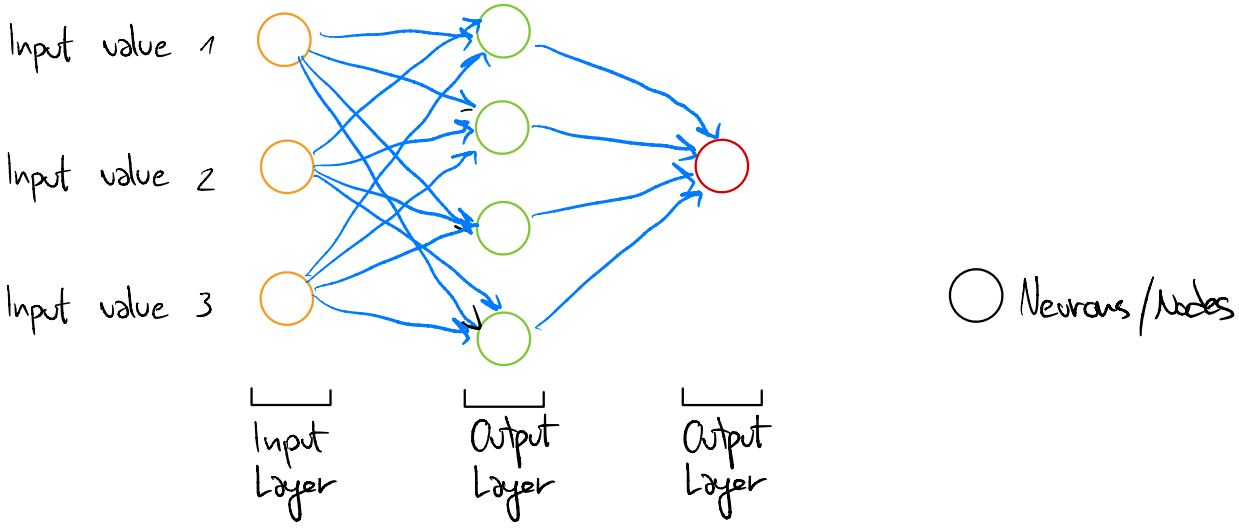
Hey mate, have you read about Hinton's capsule networks?	→	No
Did you like that recipe I sent you last week?	→	Yes
Hi Kirill, are you coming to dinner tonight?	→	Yes
Dear Kirill, would you like to service your car with us again?	→	No
Are you coming to Australia in December?	→	Yes
...	→	...

↓

Training Data:

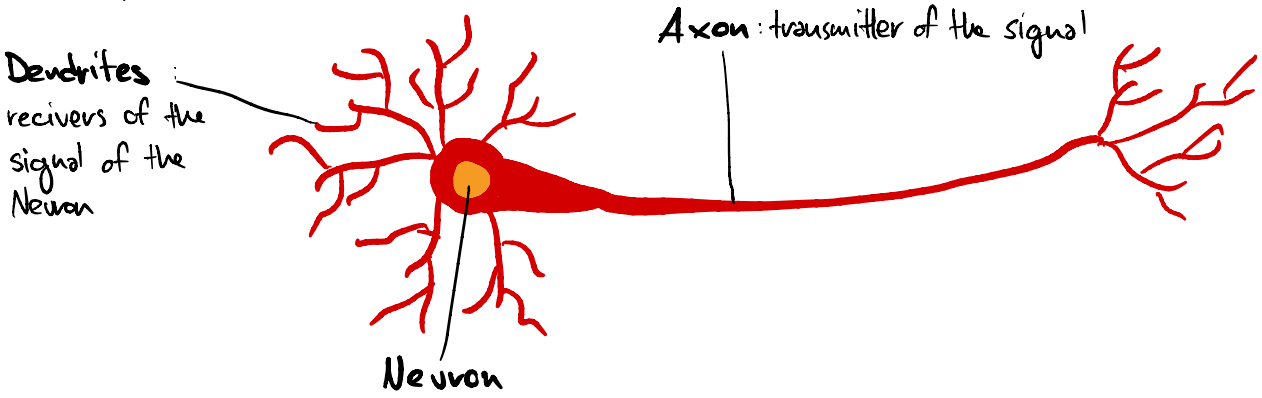
[1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, ... , 2]	→	No
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, ... , 0]	→	Yes
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, ... , 1]	→	Yes
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, ... , 1]	→	No
[1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, ... , 1]	→	Yes
...	→	...

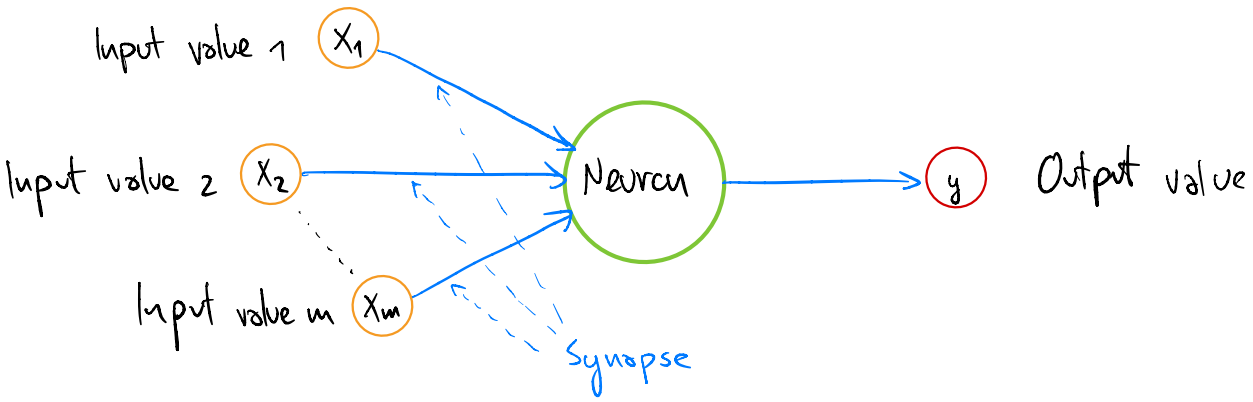
SECTION 35 : DEEP LEARNING

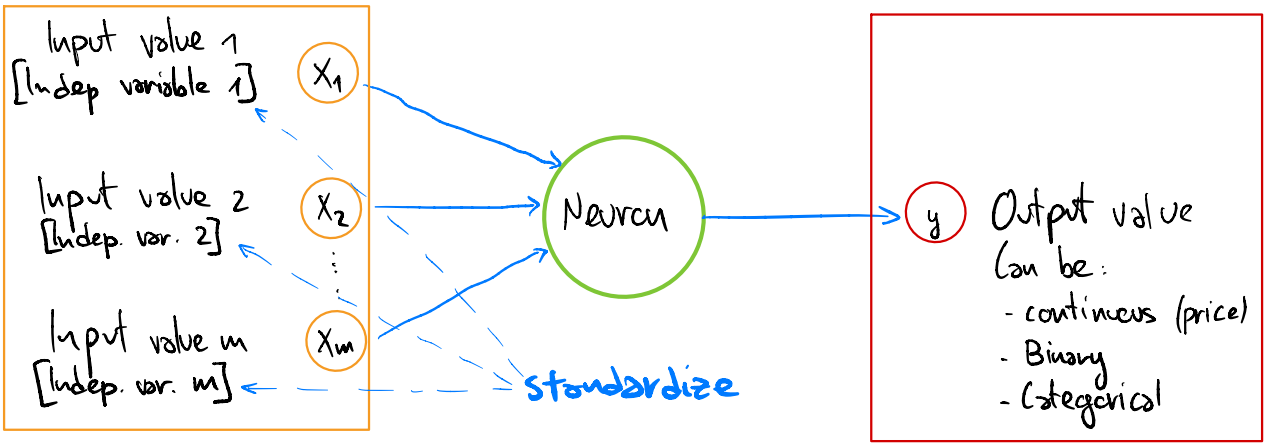


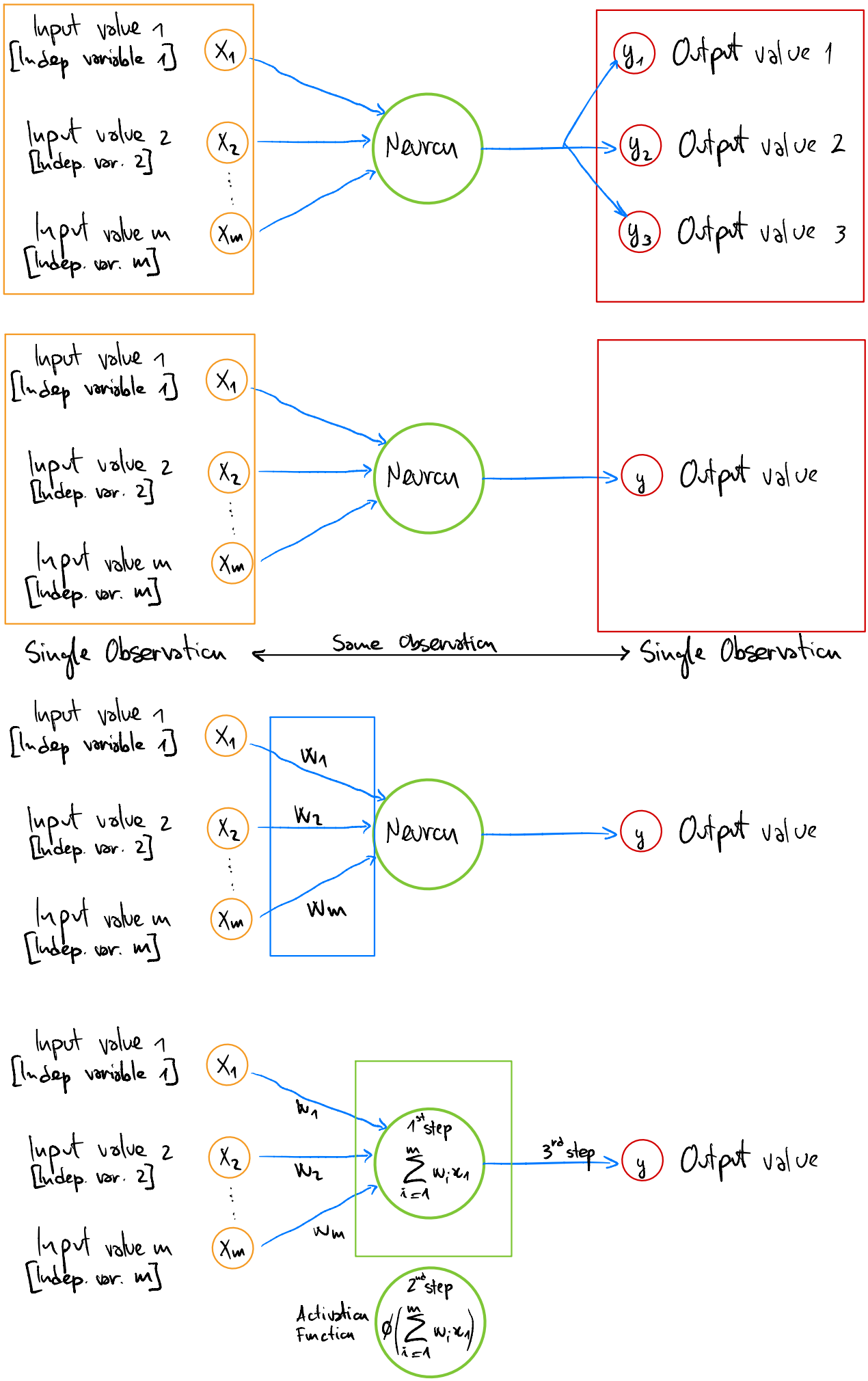
SECTION 36 : ARTIFICIAL NEURAL NETWORK

▸ THE NEURON

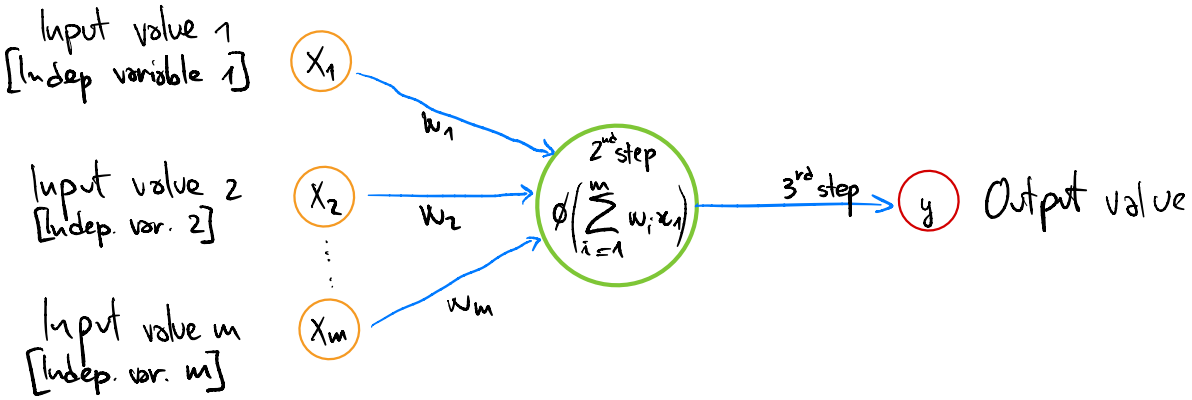


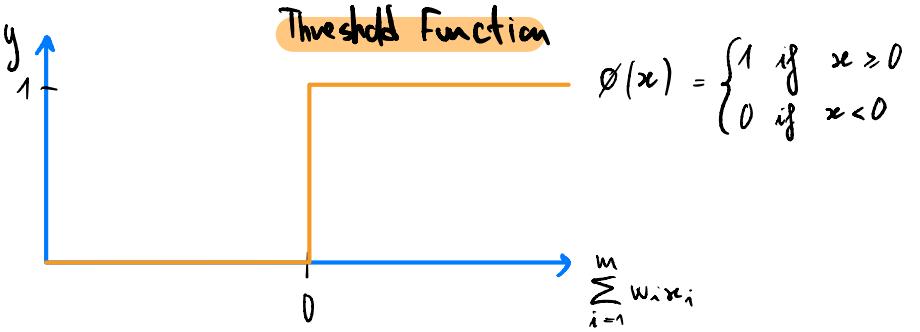


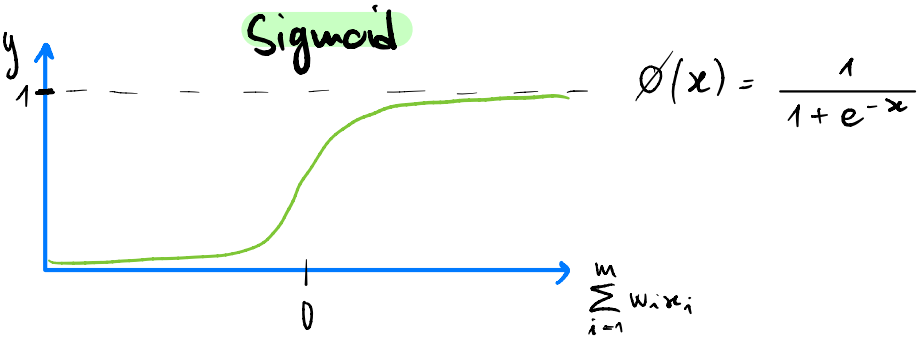


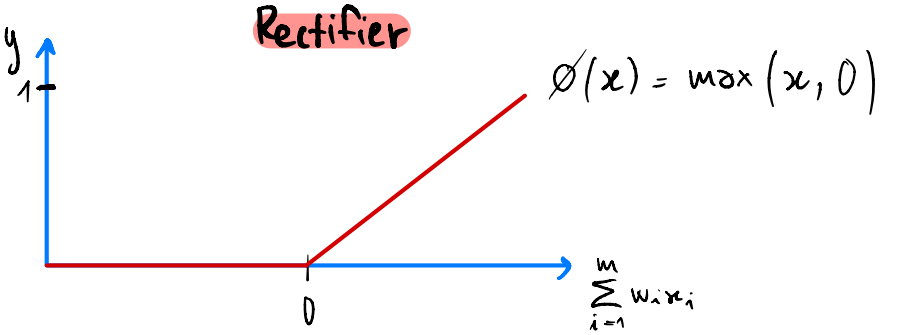

Input value 1
[Indep variable 1]
Input value 2
[Indep. var. 2]
Input value m
[Indep. var. m]
x_1
x_2
x_m
Neuron
y_1 Output value 1
y_2 Output value 2
y_3 Output value 3
Input value 1
[Indep variable 1]
Input value 2
[Indep. var. 2]
Input value m
[Indep. var. m]
x_1
x_2
x_m
Neuron
y Output value
Single Observation
Same observation
Single Observation
Input value 1
[Indep variable 1]
Input value 2
[Indep. var. 2]
Input value m
[Indep. var. m]
x_1
x_2
x_m
w_1
w_2
w_m
Neuron
y Output value
Input value 1
[Indep variable 1]
Input value 2
[Indep. var. 2]
Input value m
[Indep. var. m]
x_1
x_2
x_m
w_1
w_2
w_m
1st step
$\sum_{i=1}^{m} w_i x_1$
3rd step
y Output value
2nd step
Activation Function
$\phi\left(\sum_{i=1}^{m} w_i x_1\right)$

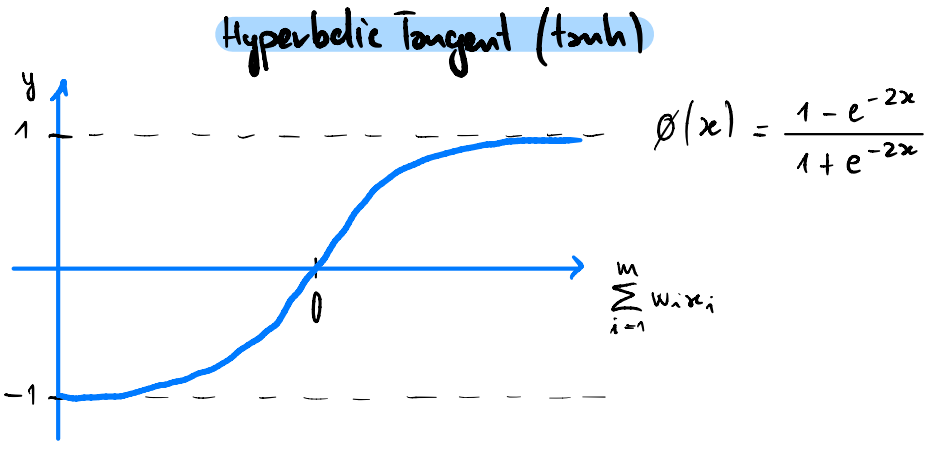
THE ACTIVATION FUNCTION



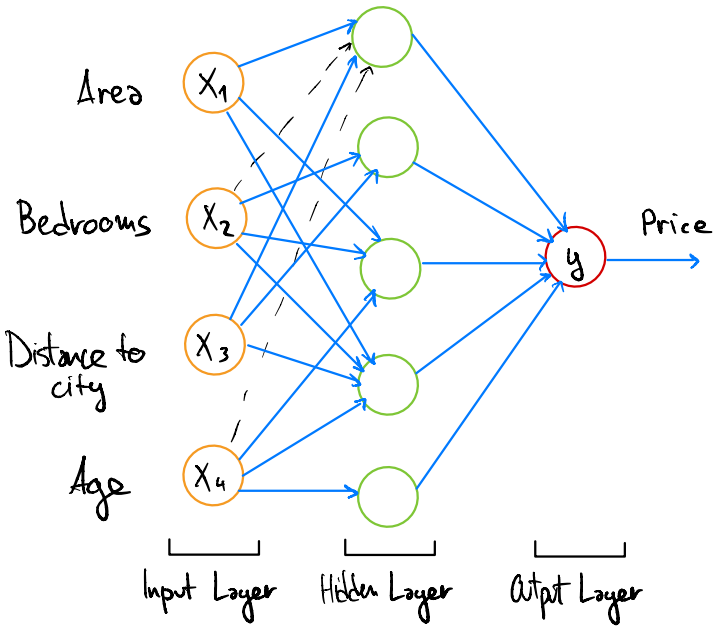








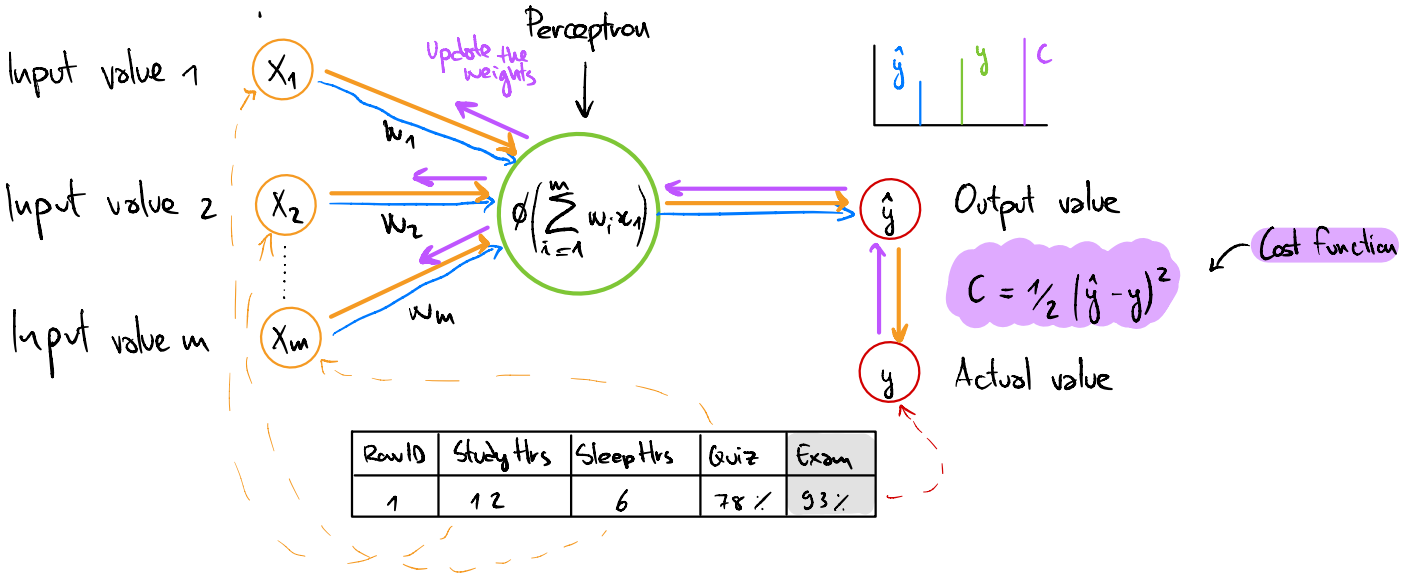
HOW DO NEURAL NETWORKS WORK?



$$Price = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

Some weights will have a 0 value

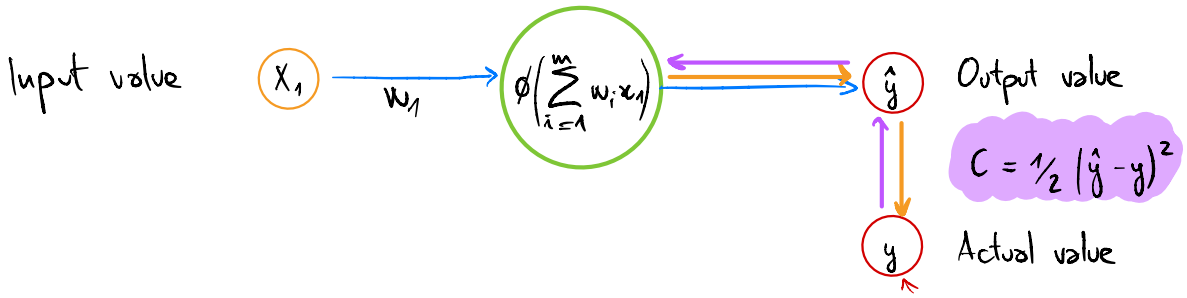
HOW DO NNs LEARN?



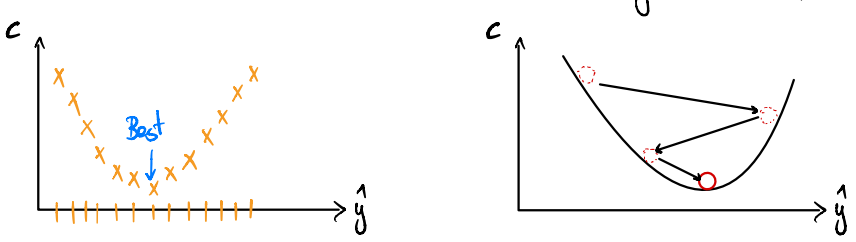
RowID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

(BATCH) GRADIENT DESCENT

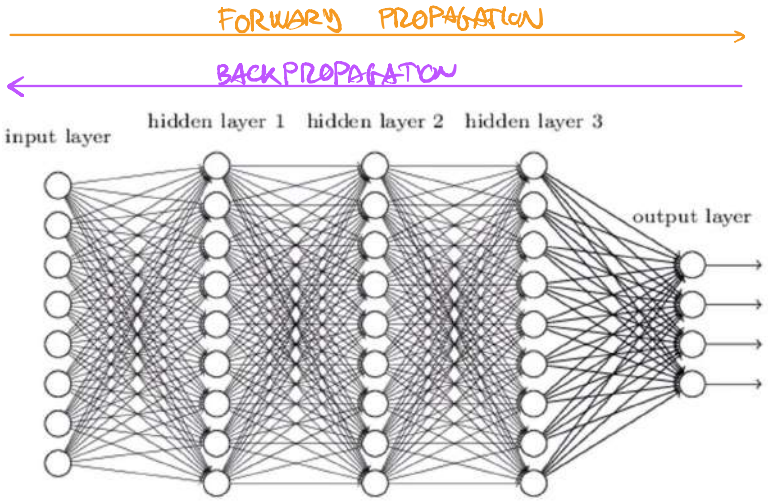
How the weights are adjusted.



- How we can minimize the cost function?



BACKPROPAGATION



Information is entered into the input layer and then is propagated forward to get y/output and then we compare those to the actual values that we have in our training set, and then we calculate the errors.
Then the errors are back propagated through the network in the opposite direction → allows to train the network by adjusting the weight. You are able to adjust all the weights at the same time, so you know which part of the error each of your weights in the NN is responsible for.

- TRAINING THE ANN WITH STOCHASTIC GRADIENT DESCENT

STEP 1: Randomly initialize the weights to small numbers close to 0 (but not 0)

STEP 2: Input the first observation of your dataset in the input layer, each feature in 1 input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted results y.

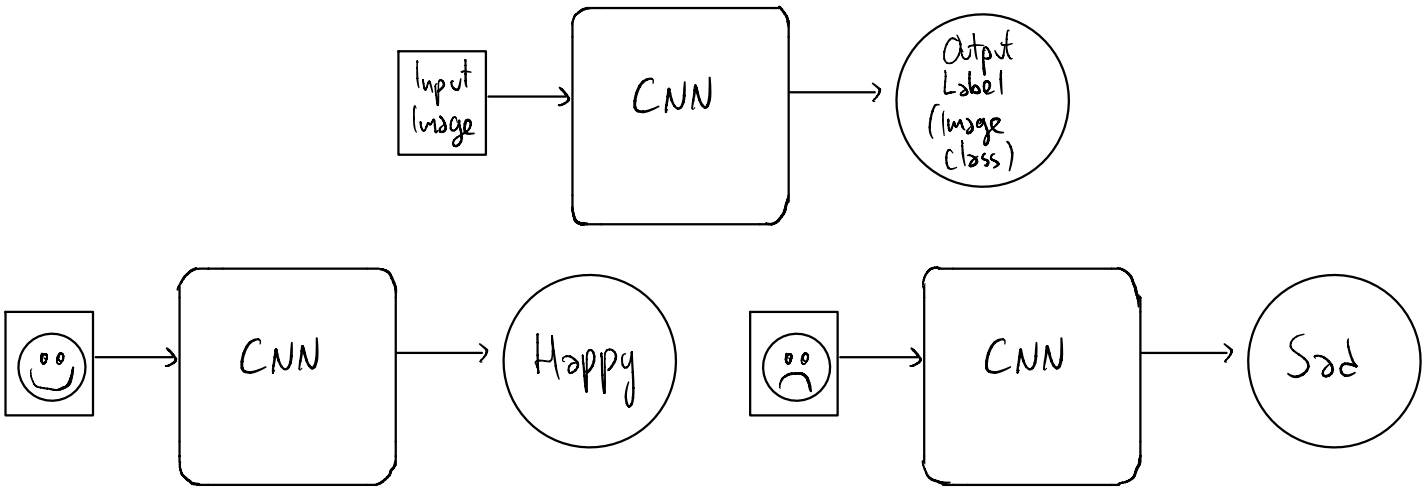
STEP 4: Compare the predicted results to the actual results. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat steps 1 to 5 and update the weights after each observation (Reinforcement Learning). Or: repeat steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs

SECTION 37 : CONVOLUTIONAL NEURAL NETWORKS

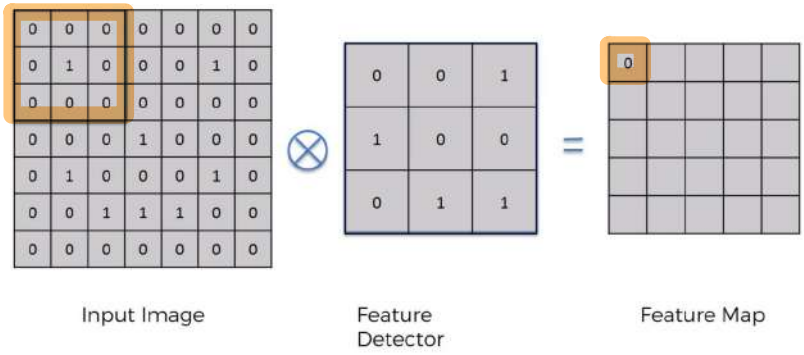


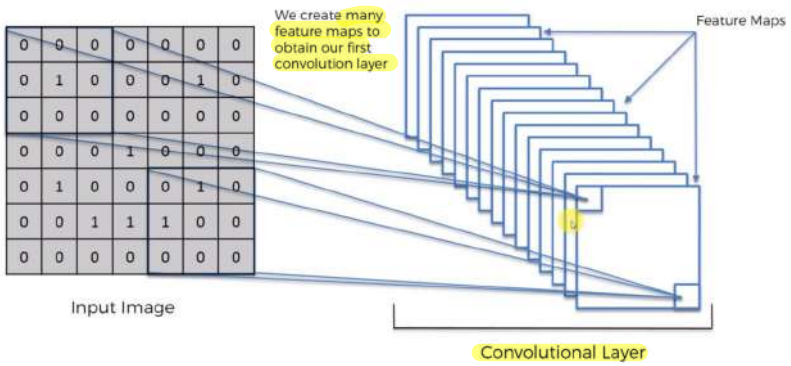
STEP 1 : Convolution
↓
STEP 2 : Max Pooling
↓
STEP 3 : Flattening
↓
STEP 4 : Full Connection

▸ STEP 1 : CONVOLUTION

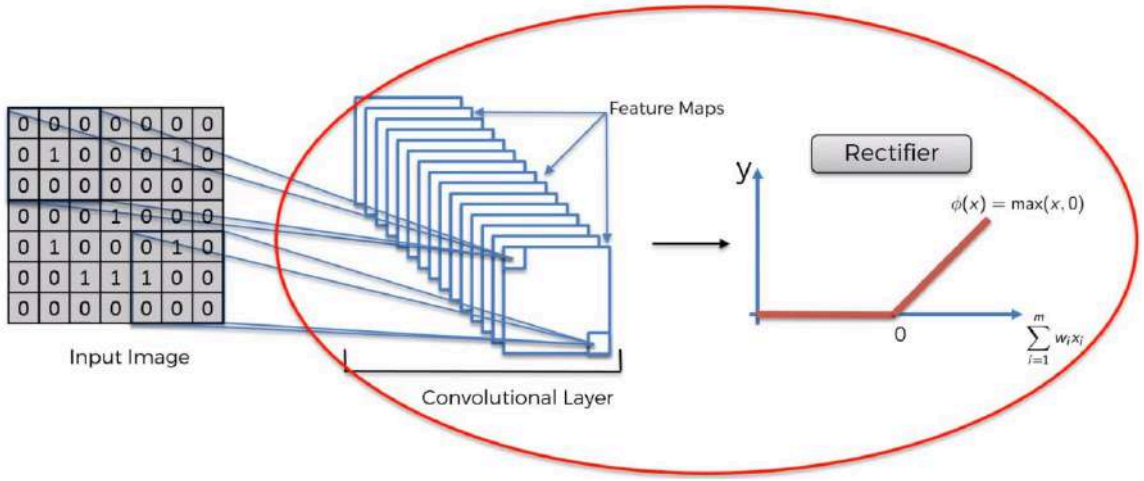
To find features in the images using the **feature detector**, put them into a **feature map**, and by having them in a feature map, it still preserves the **spatial relationship** between pixels.

$$(f \cdot g)(t) \overset{\text{def}}{=} \int_{-\infty}^{+\infty} f(\tau)\, g(t-\tau)\, d\tau$$

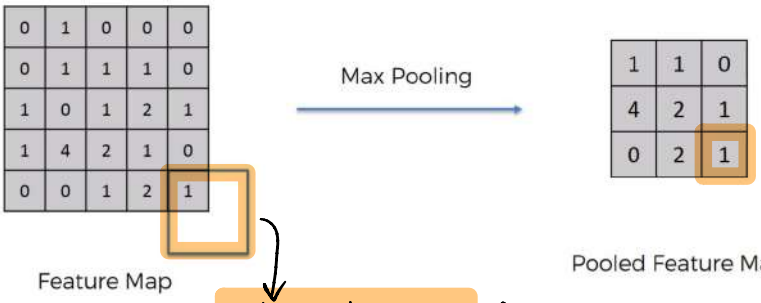




▸ STEP 1b : ReLU LAYER

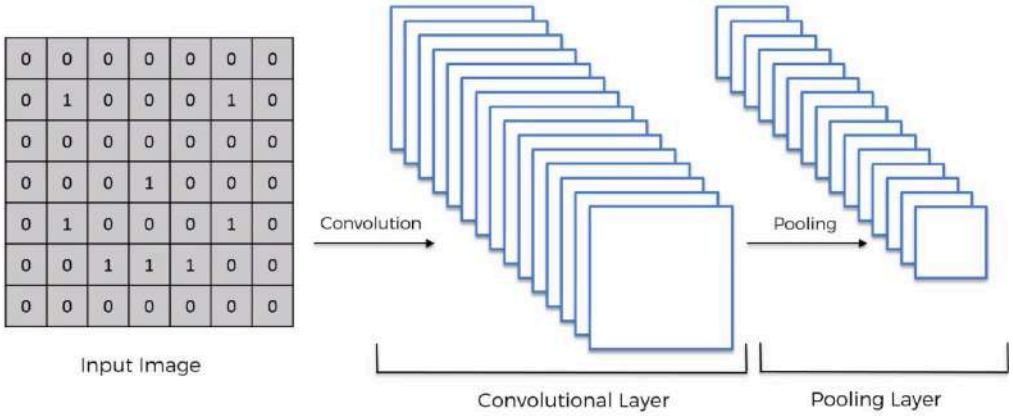


▸ STEP 2 : MAX POOLING

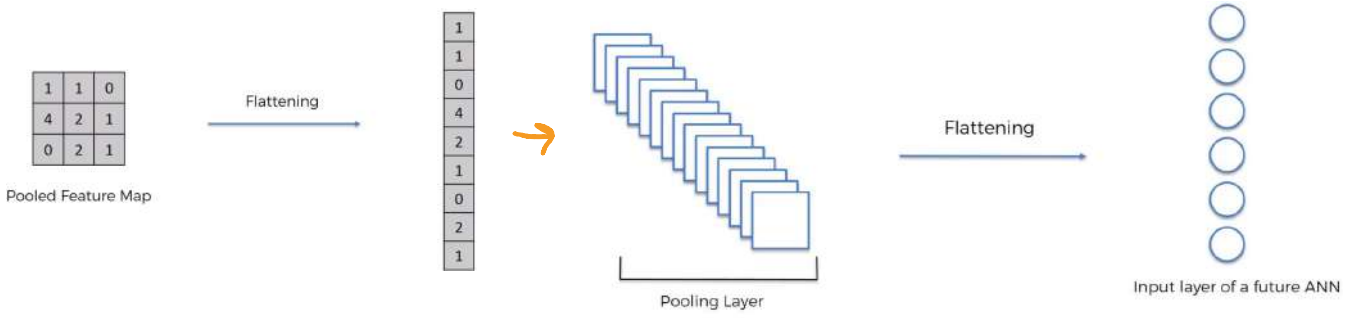


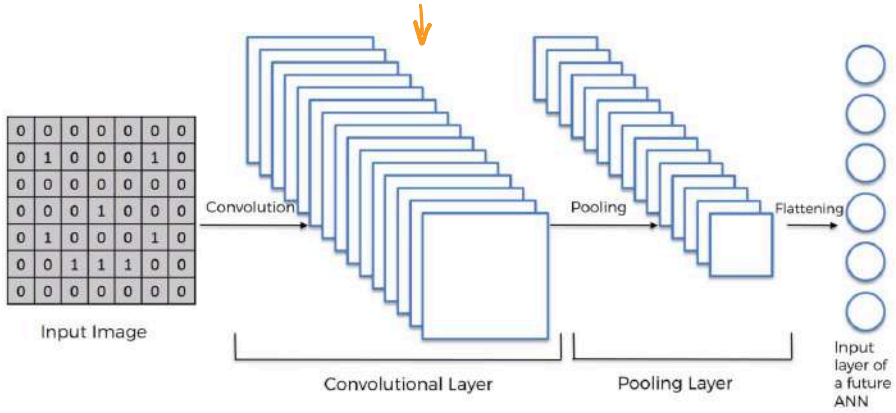
Select the max of the grid of the feature map

KEY POINT:
- we are able to preserve the features and moreover account for their spatial or textural or other kind of distorsions.
- we are reducing the size (by the 75 %)
- we're introducing spatial invariance
- we're reducing the number of parameters
- we're preventing overfitting by reducing informations



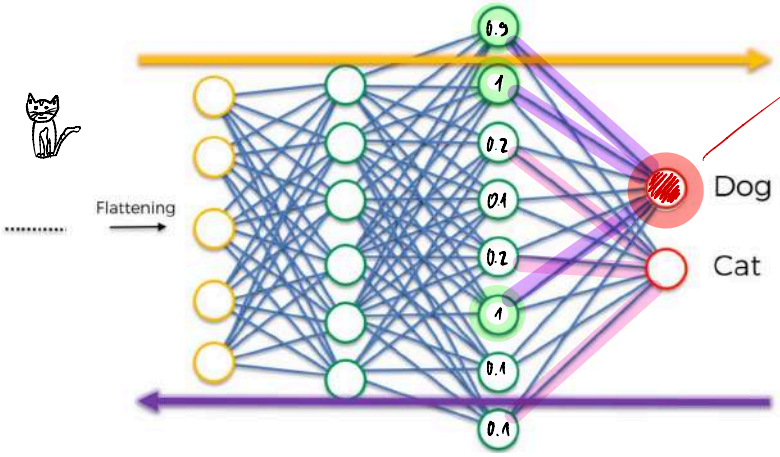
▸ STEP 3 : FLATTENING





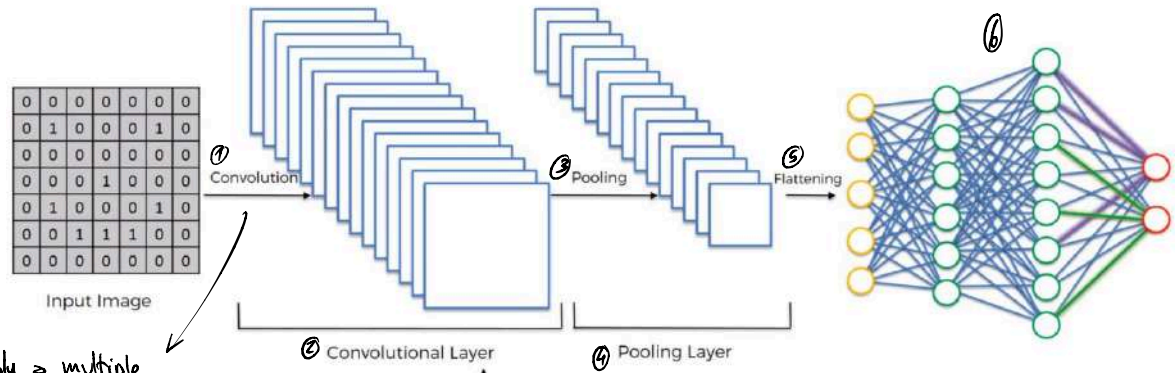
STEP 4 : FULL CONNECTION

In this step we're adding a whole ANN to our Convolution NN.

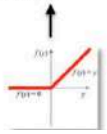


We need to understand **what weights to assign** to all of these synapsis that connect to the dog, so that we know which of the previous neurons are actually important for the dog.

SUMMARY

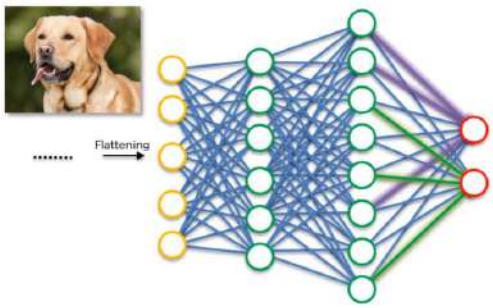


We apply a multiple different feature detectors to create these feature maps



We apply the ReLu to remove any linearity or increase non-linearity in our images

▸ SOFTMAX & CROSS-ENTROPY



$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$ ← Softmax Function

We apply the Softmax function that would bring the z values to be between 0 and 1 and it would make them add up to 1. The Softmax Function is a generalization of the logistic function.

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$

NN1 NN2

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

1/3 = 0.33 | 1/3 = 0.33

Mean Squared Error

0.25 | 0.71

Cross-Entropy

0.38 | 1.06

SECTION 39 : PRINCIPAL COMPONENT ANALYSIS (PCA)

Is used for :

- Noise filtering
- Feature Extraction
- Gene data analysis
- Visualization
- Stock Market Predictions

Goal :

- Identify **pattern** in data
- Detect the **correlation** between variables. If there is a **strong** corr and it's found, then you could **reduce the dimensionality**. You find the directions of maximum variance in high dimensional data, and then you project it into a **smaller dimensional subspace**

> Reduce the dimensions of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d)

- Standardize the data.
- Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition.
- Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace ($k \leq d$)/.
- Construct the projection matrix $\mathbf{W}$ from the selected k eigenvectors.
- Transform the original dataset $\mathbf{X}$ via $\mathbf{W}$ to obtain a k-dimensional feature subspace $\mathbf{Y}$

https://plot.ly/ipython-notebooks/principal-component-analysis/

SECTION 40 : LINEAR DISCRIMINANT ANALYSIS (LDA)

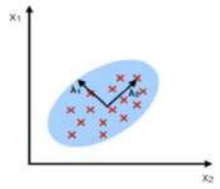
- Used as a dimensionality reduction technique
- Used in the pre-processing step for pattern classification
- Goal: to project a dataset onto a lower-dim space

LDA differs from PCA bc in addition to finding the component axises with LDA we are interested in the **axes that maximize the separation between multiple classes**.

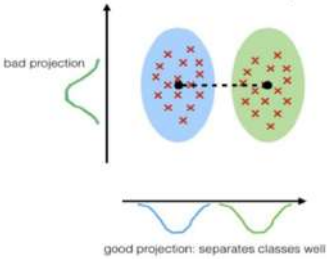
The **goal** of LDA is to project a **feature space** (a dataset n-dimensional samples)

onto a small subspace K (where $K \leq n-1$) while **maintaining the class-discriminatory information**. Is a **supervised** bc of the relation to the dep var.

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation

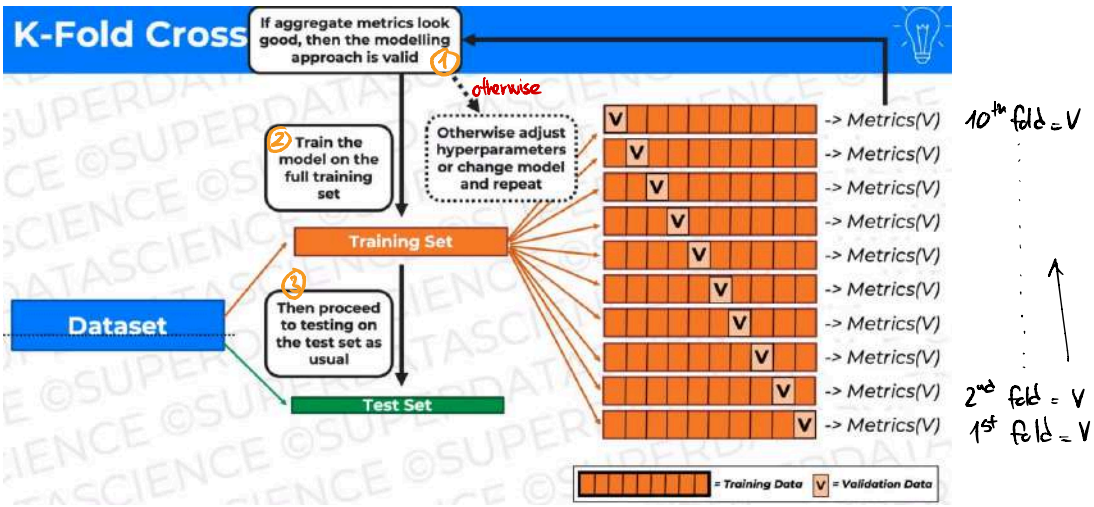


STEP:

1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$) and corresponding eigenvalues ($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix $\boldsymbol{W}$ (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (where $\boldsymbol{X}$ is a $n \times d$-dimensional matrix representing the n samples, and $\boldsymbol{y}$ are the transformed $n \times k$-dimensional samples in the new subspace).

SECTION 43: MODEL SELECTION

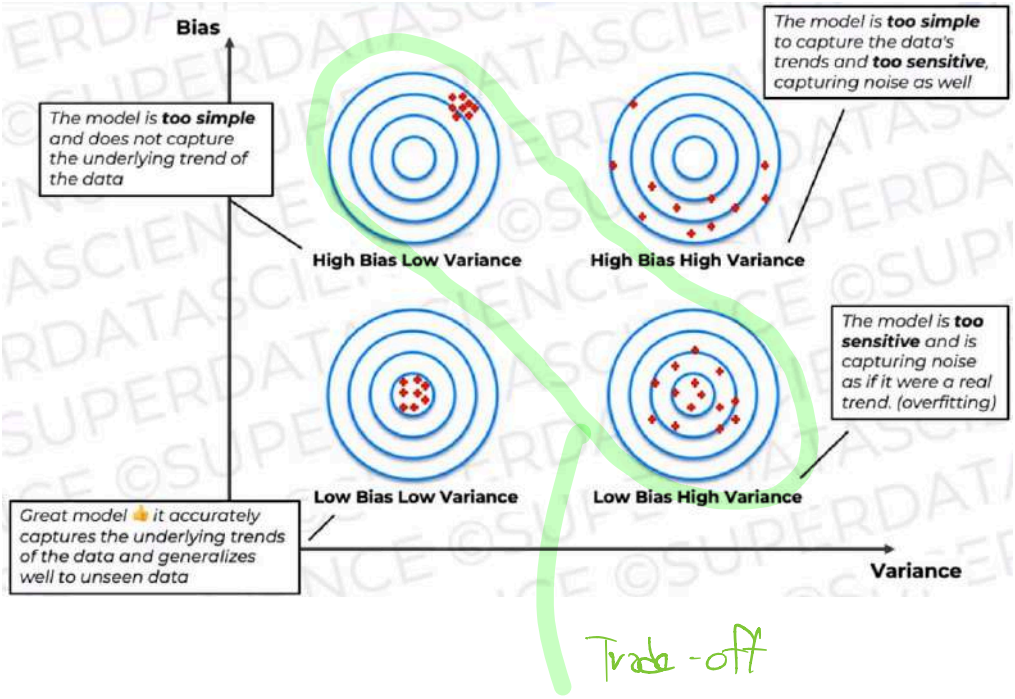
▸ k-Fold Cross-Validation



▸ Bias-Variance

Bias: systematic error that occurs in the ML model itself due to incorrect assumption in the ML process.
→ Bias as the error between avg model prediction and the ground truth.

Variance: how much the model can adjust depending on the given data set. Refers to the changes in the model when using different proportions of the training dataset.


Bias
The model is too simple and does not capture the underlying trend of the data
High Bias Low Variance
The model is too simple to capture the data's trends and too sensitive, capturing noise as well
High Bias High Variance
The model is too sensitive and is capturing noise as if it were a real trend. (overfitting)
Low Bias Low Variance
Low Bias High Variance
Great model 👍 it accurately captures the underlying trends of the data and generalizes well to unseen data
Variance

Trade-off