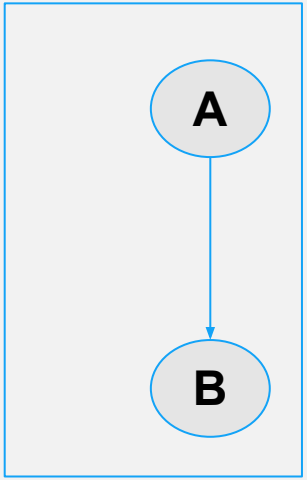
ML Algorithms for Classification Problems

Naive Bayes Classifier

Bayes Network

Bayes Network



Not observable P(A)

Observable P(B/A)

P(B/¬A)

Diagnostic reasoning: P(A/B)

P(A/¬B)

Bayes Rule

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}$$

P(A): prior

P(A/B): posterior

Bayes Rule

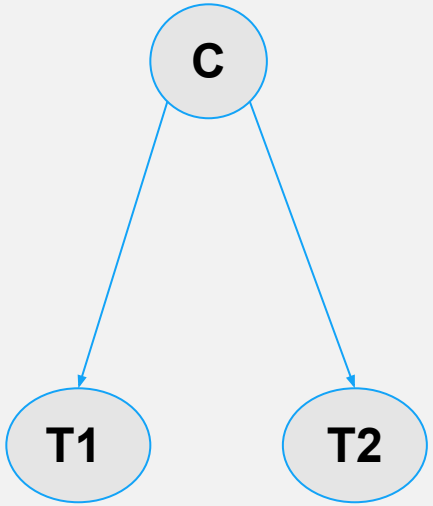
$$P(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$$

We could also infer:

$$P(\neg A/B) = \frac{P(B/\neg A)\,P(\neg A)}{P(B)}$$

$$P(A/B) + P(\neg A/B) = 1$$

Test Cancer example



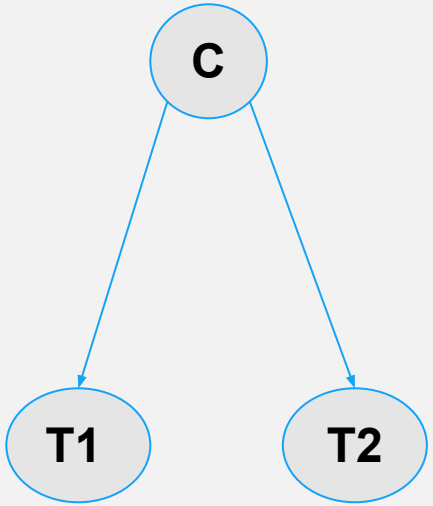
P(C)=0.01

P(+/C)=0.9

P(-/¬C)=0.8

P(C/ T1=+ , T2=+) = P(C/++) = ?

Test Cancer example



P(C)=0.01

P(+/C)=0.9

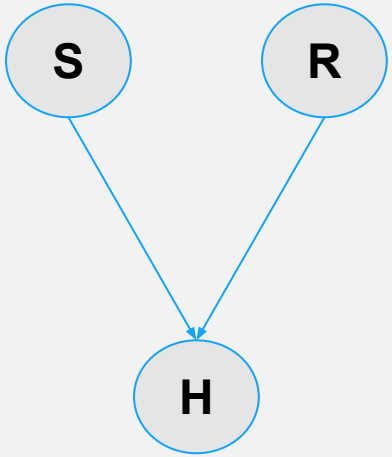
P(-/¬C)=0.8

P(C/++) = P(++/C)P(C) / P(++)

= P(+/C)P(+/C)P(C) / [P(++/C)P(C)+P(++/¬C) P(¬C)]

= 0.1698

Confounding cause



$P(S) = 0.7$

$P(R) = 0.01$

$P(H/S,R) = 1$

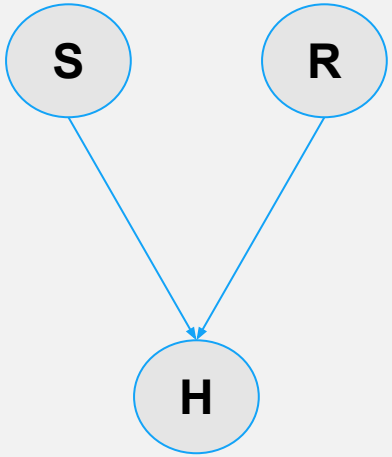
$P(H/\neg S,R) = 0.9$

$P(H/S,\neg R) = 0.7$

$P(H/\neg S,\neg R) = 0.1$

$P(R/S) = ?$

Being Happy Example



P(S) = 0.7

P(R) = 0.01

P(H/S,R) = 1

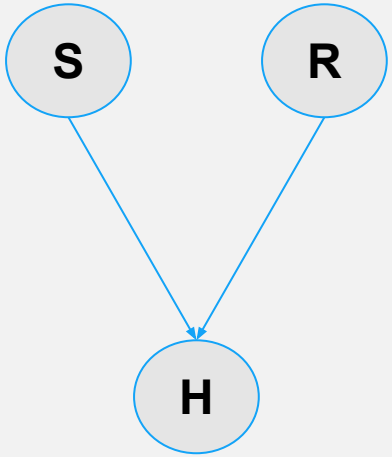
P(H/¬S,R) = 0.9

P(H/S,¬R) = 0.7

P(H/¬S,¬R) = 0.1

P(R/S) = 0.01 (R and S are independent!)

Being Happy Example



P(S) = 0.7

P(R) = 0.01

P(H/S,R) = 1

P(H/¬S,R) = 0.9

P(H/S,¬R) = 0.7

P(H/¬S,¬R) = 0.1

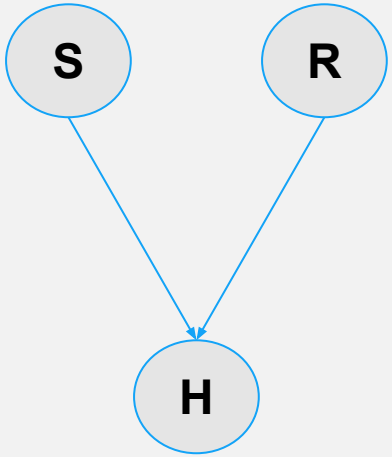
P(R/H,S)

= P(H/S,R)P(R/S) / P(H/S) *

= P(H/S,R)P(R) / [P(H/R,S)P(R)+P(H/¬R,S)P(¬R)]

= 0.0142

Being Happy Example



P(S) = 0.7

P(R) = 0.01

P(R/H) = ?

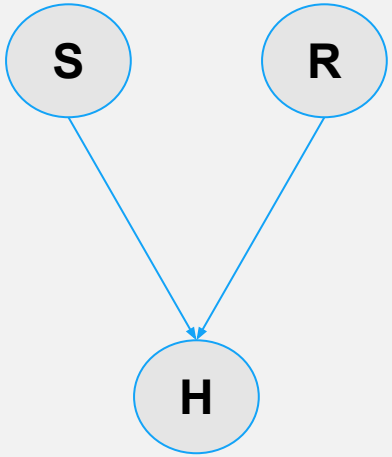
P(H/S,R) = 1

P(H/¬S,R) = 0.9

P(H/S,¬R) = 0.7

P(H/¬S,¬R) = 0.1

Being Happy Example



P(S) = 0.7

P(R) = 0.01

P(R/H) = 0.97

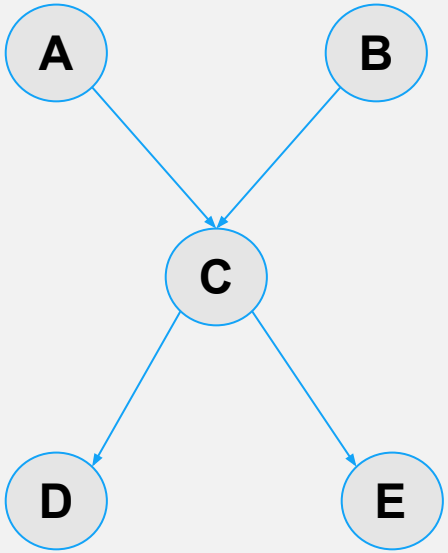
P(H/S,R) = 1

P(H/¬S,R) = 0.9

P(H/S,¬R) = 0.7

P(H/¬S,¬R) = 0.1

General Bayes networks



Any joint distribution requires 2^5 -1 =31 Probability value.

- Bayes networks define probability distributions over a graph of random variables.
- A bayes networks is defined by the probabilities:

P(A), P(B), P(C/A,B), P(D/C), P(E/C)

1 1 4 2 2

P(A,B,C,D,E) = product of 5 prob above (only requires 10 probability values)

Application of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.

- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.

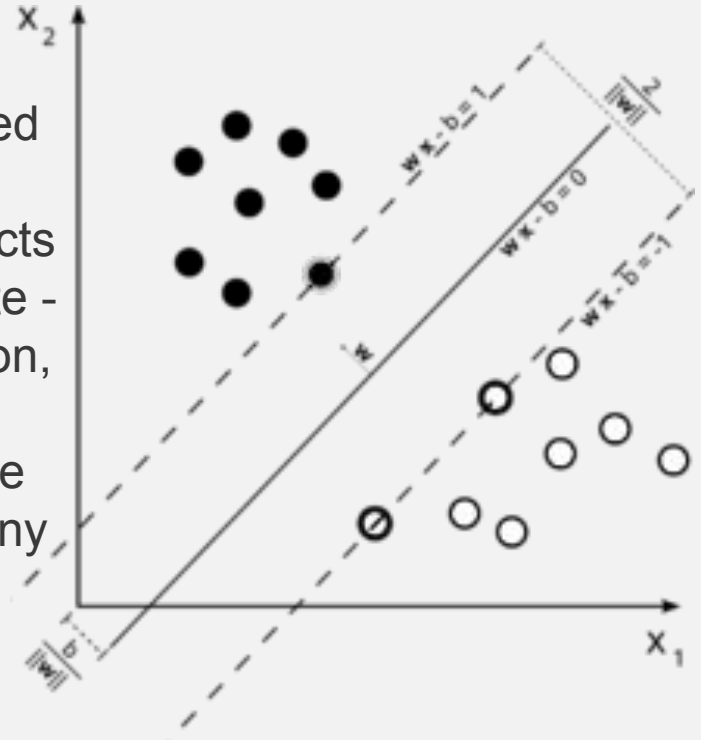
- **Text classification/ Spam Filtering/ Sentiment Analysis.**

- **Recommendation System.**

Support Vector Machines

Formal definition

SVMs are a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis. A SVM constructs a hyperplane or set of hyperplanes in a high - or infinite - dimensional space, which can be used for classification, regression or any other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class.



Linear SVMs

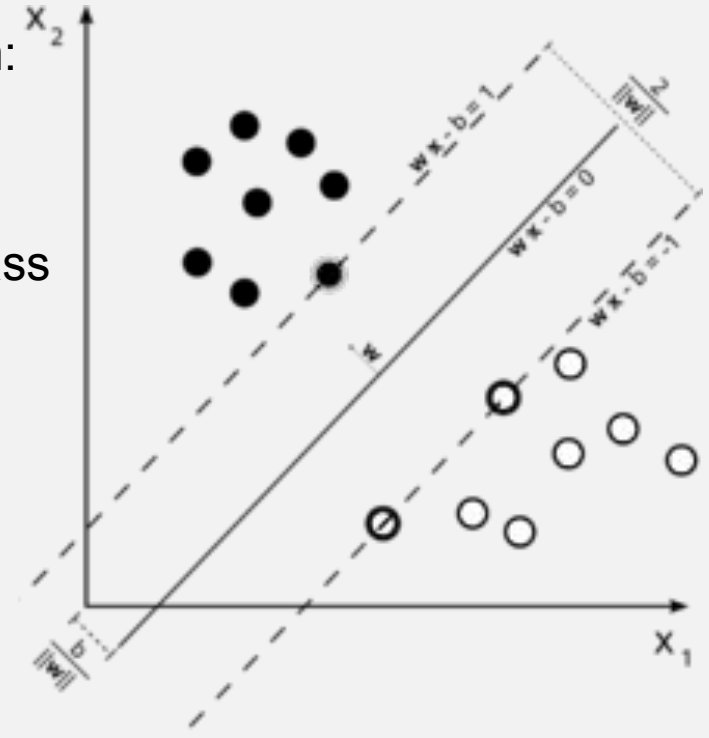
We are given a training dataset of n points of the form:

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$$

where the yi are either 1 or −1, each indicating the class to which the point xi belongs.

$$\vec{w} \cdot \vec{x} - b = 0$$

Any hyperplane can be written as the set of points xi satisfying:



Hard margin

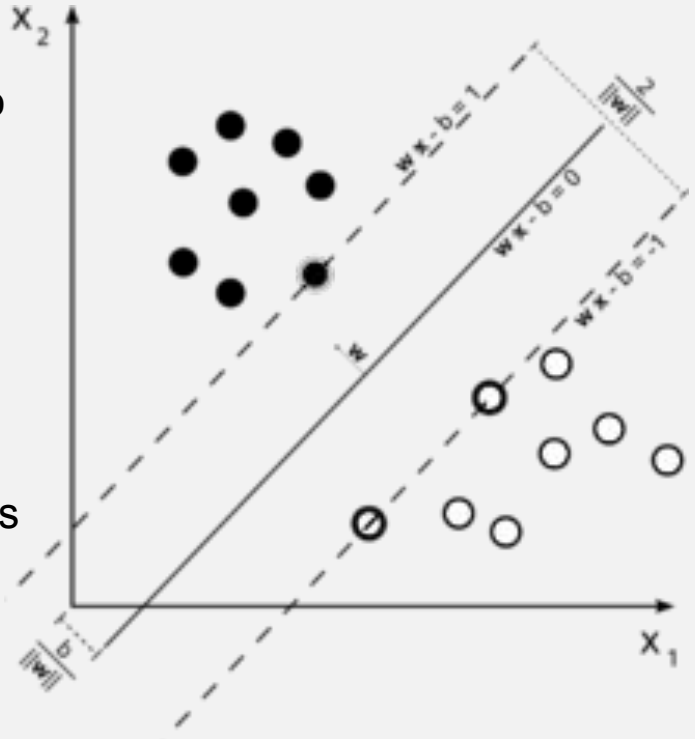
If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them.

With a normalized or standardized dataset, these hyperplanes can be described by the following equations:

$\vec{w} \cdot \vec{x} - b = 1$ (anything on or above this boundary is of one class, with label 1)

$\vec{w} \cdot \vec{x} - b = -1$ (anything on or below this boundary is of the other class, with label -1)



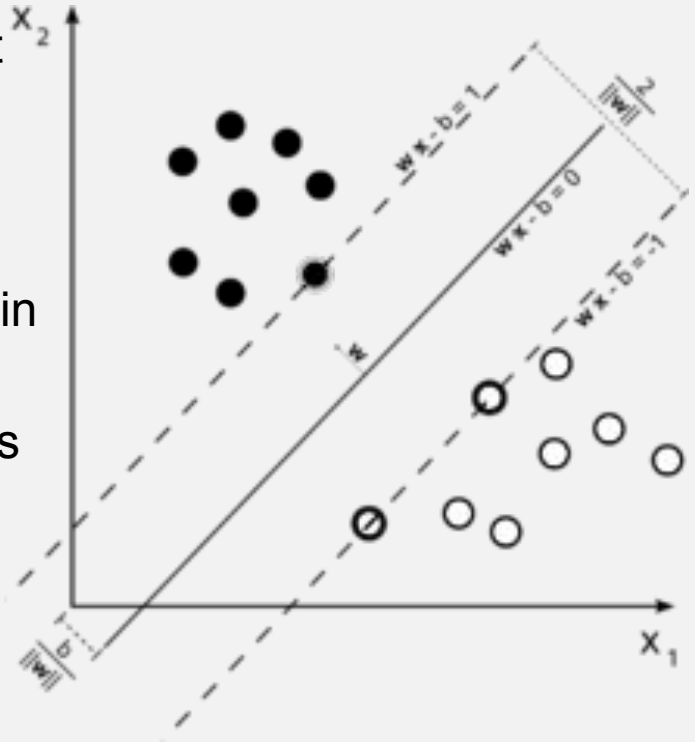
Soft margin

- To extend SVM to cases in which the data are not linearly separable, we introduce the hinge loss function: $\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right)$

This function is zero if the constraint in (1) is satisfied, in other words, if xi lies on the correct side of the margin. For data on the wrong side of the margin, the function's value is proportional to the distance from the margin.

We then wish to minimize:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right)\right] + \lambda\|\vec{w}\|^2$$



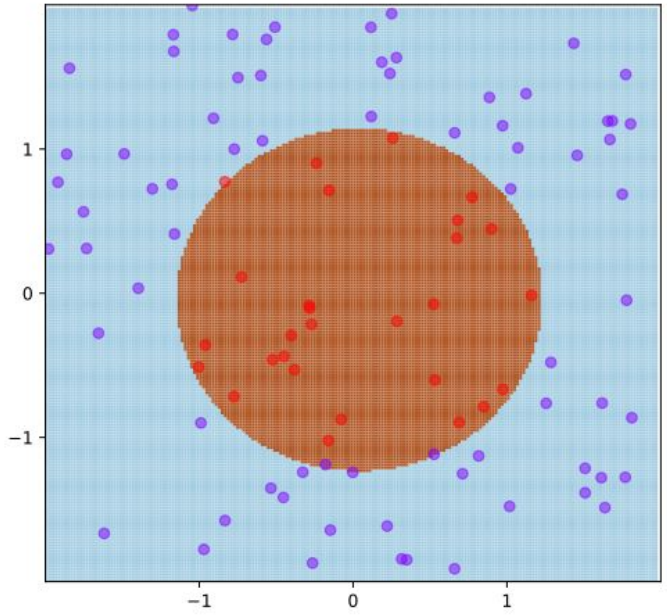
Non linear classification

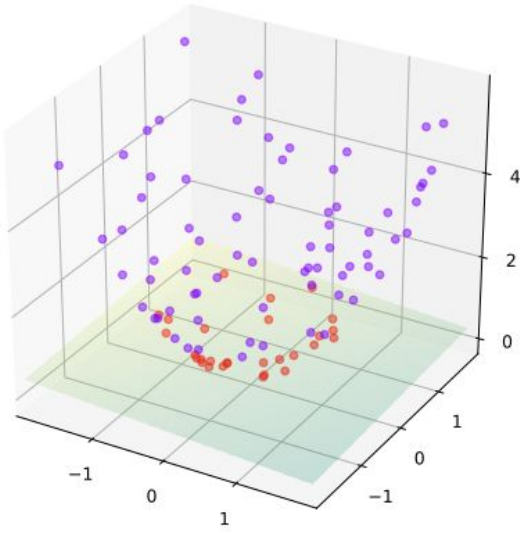
When data is non linearly separable to create nonlinear classifiers by applying the kernel trick to build maximum-margin hyperplanes.

The kernel trick avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary. For all x and x' in the input space X, certain functions k can be expressed as an inner product in another space V.

- Polynomial (homogeneous): $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$
- Polynomial (inhomogeneous): $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$
- Gaussian radial basis function: $k(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$, for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/2\sigma^2$
- Hyperbolic tangent: $k(\vec{x_i}, \vec{x_j}) = \tanh(\kappa \vec{x_i} \cdot \vec{x_j} + c)$, for some (not every) $\kappa > 0$ and $c < 0$

Non linear classification





How to determine the best Kernel

Here, we want to keep an eye on our objective function: minimizing the hinge-loss. We would setup a hyperparameter search (grid search, for example) and compare different kernels to each other. Based on the loss function (or an accuracy metric), we could determine which kernel is "appropriate" for the given task.

K- Nearest Neighbor

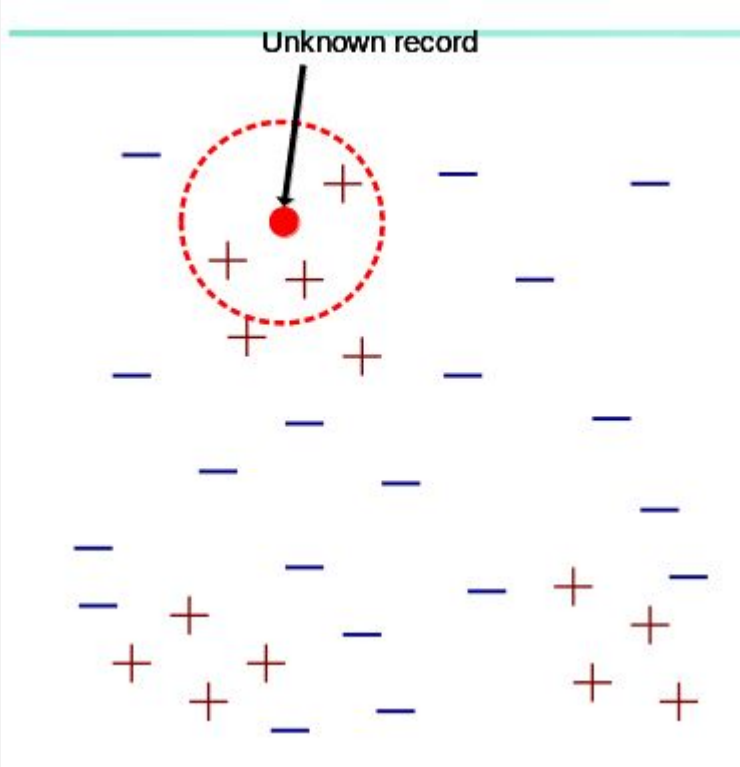
Nearest-Neighbor Classifiers

Requires three things

- The set of stored records
- Distance Metric to compute distance between records
- The value of k, the number of nearest neighbors to retrieve

To classify an unknown record:

- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



Nearest-Neighbor Classifiers

Compute distance between two points:

- Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

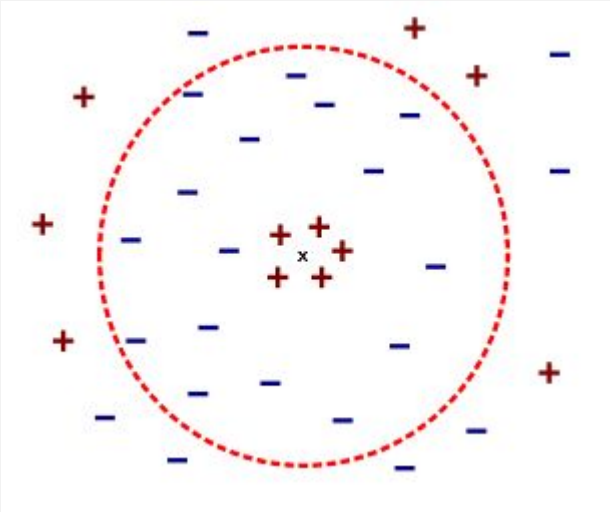
Determine the class from nearest neighbor list

- take the majority vote of class labels among the k-nearest neighbors
- Weigh the vote according to distance
  - weight factor, $w = 1/d^2$

Nearest-Neighbor Classifiers

Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes



Evaluation of Classification algorithms

Confusion matrix

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Confusion matrix

- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.
- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Confusion matrix

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

- **Accuracy:** Overall, how often is the classifier correct?
  (TP+TN)/total = (100+50)/165 = 0.91

- **Sensitivity/Recall:** When it's actually yes, how often does it predict yes?
  TP/actual yes = 100/105 = 0.95

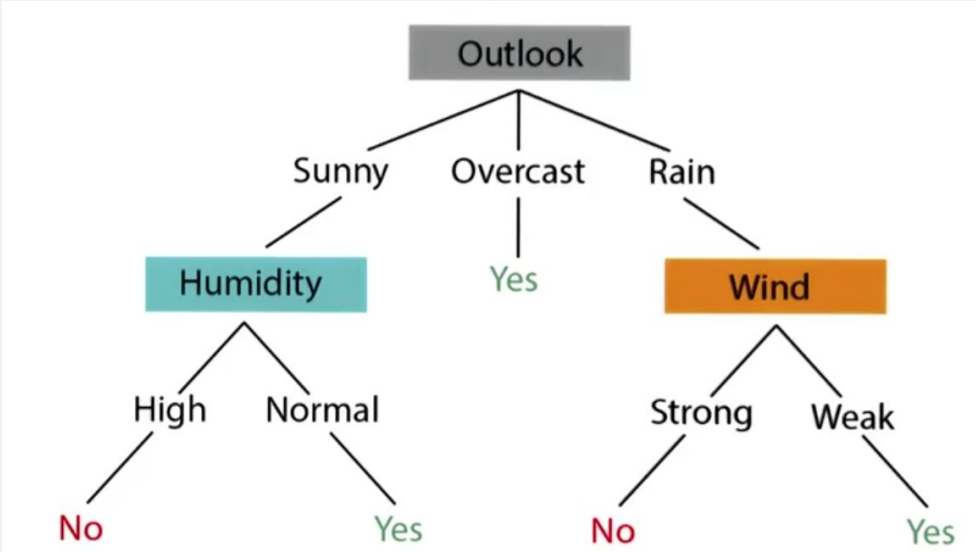
- **Precision:** When it predicts yes, how often is it correct?
  TP/predicted yes = 100/110 = 0.91

- **Prevalence:** How often does the yes condition actually occur in our sample?
  actual yes/total = 105/165 = 0.64

Decision Trees

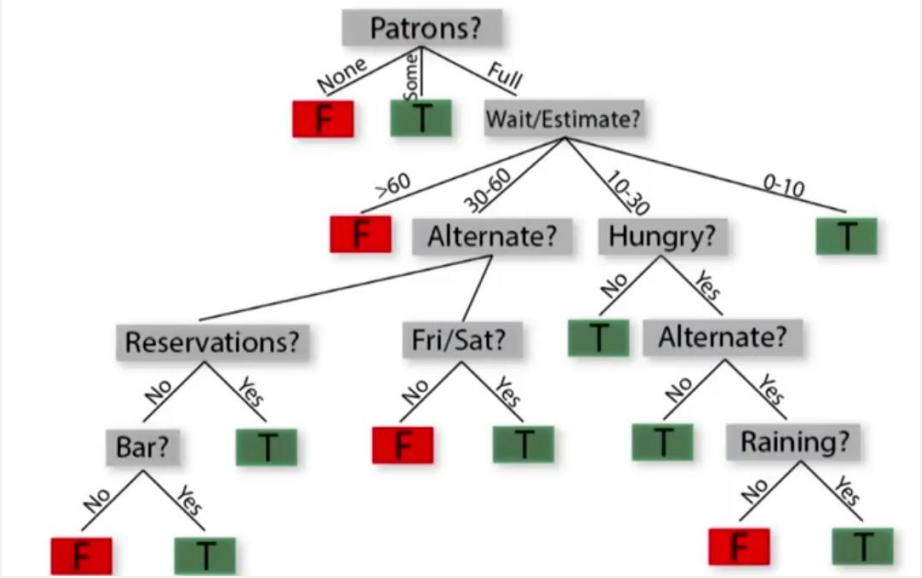
Play Tennis example

Humidity	Wind	Outlook	Play?
High	Strong	Sunny	Y
Normal	Strong	Overcast	
High	Weak	Rainy	



Eating at a restaurant example

Example	Input Attributes										Goal
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est.	Will Wait
x1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	y1=Yes
x2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	y2=No
x3	No	Yes	No	No	Some	$	No	No	Burger	1-10	y3=Yes
x4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10-30	y4=Yes
x5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	y5=No
x6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	1-10	y6=Yes
x7	No	Yes	No	No	None	$	Yes	No	Burger	1-10	y7=No
x8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	1-10	y8=Yes
x9	No	yes	Yes	No	Full	$	Yes	No	Burger	.60	y9=No
x10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	y10=No
x11	No	No	No	No	None	$	No	No	Thai	1-10	y11=No
x12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	y12=Yes



Entropy and Information Gain

Entropy is a measure of uncertainty and unpredictability in a random variable.

Gains are calculated for each attribute A.

An attribute A with d distinct values divides the training set E into subset E1,...,Ed.

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$$

$$-\left(\frac{6}{12} log_2 \frac{6}{12} + \frac{6}{12} log_2 \frac{6}{12}\right) = 1$$

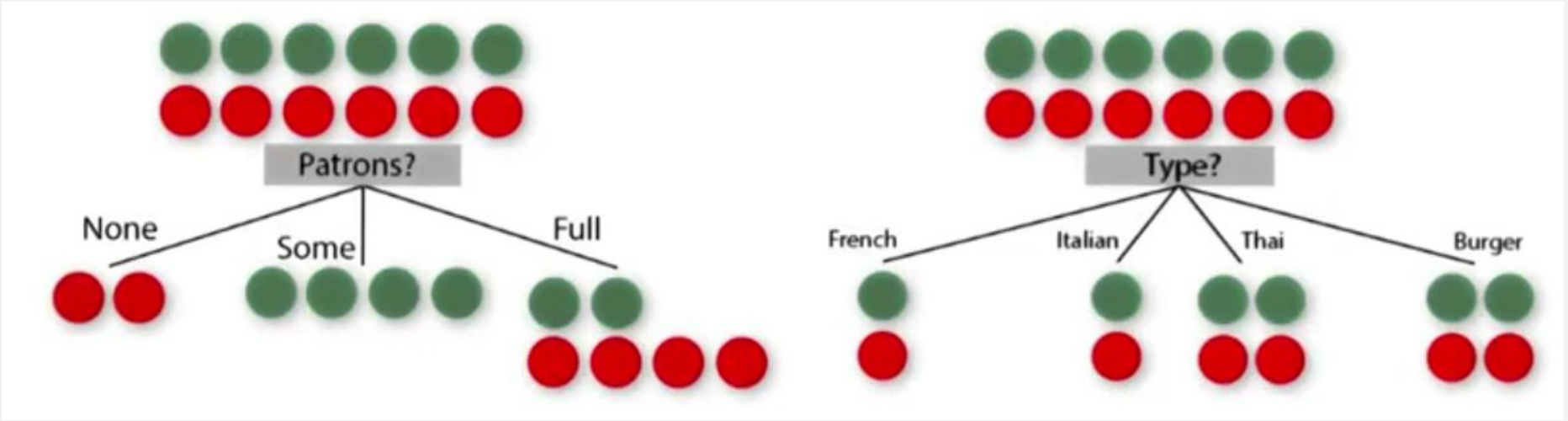
$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Gain(A) = B\left(\frac{p}{p + n}\right) - Remainder(A)$$

Entropy and Information Gain

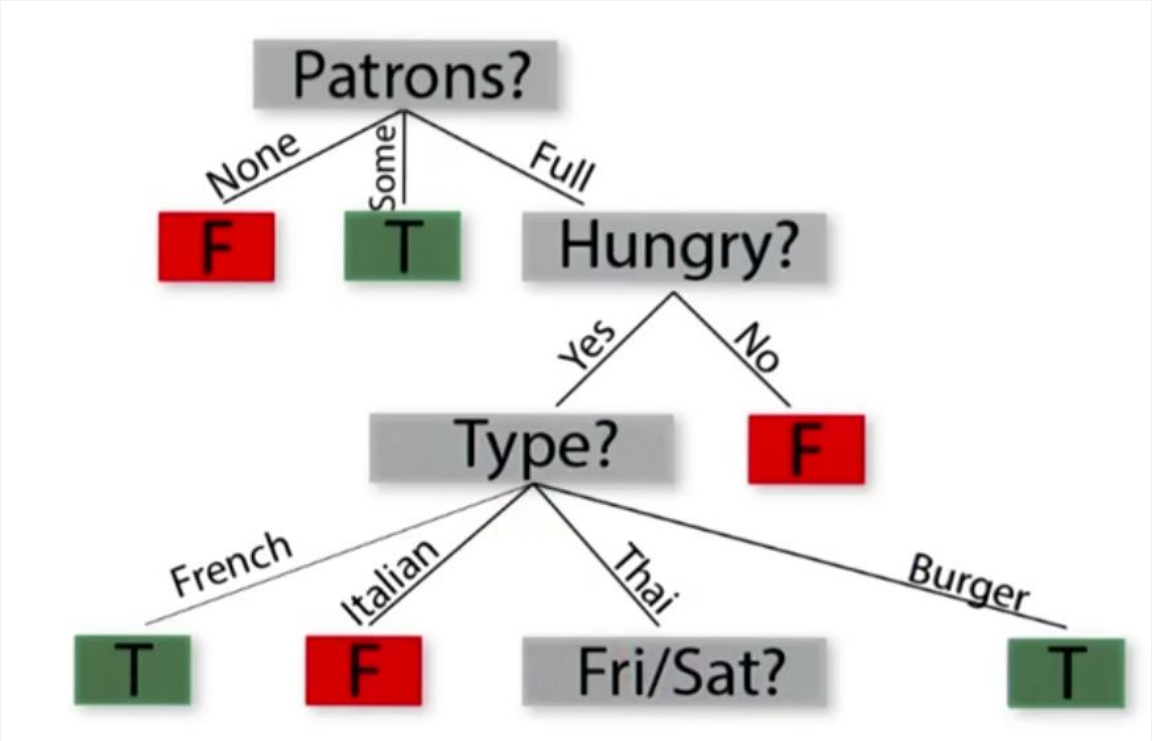
$$Gain(Patrons) = 1 - \left[\frac{2}{12}B(\frac{0}{2}) + \frac{4}{12}B(\frac{4}{4}) + \frac{6}{12}B(\frac{2}{6})\right] \approx 0.541 \text{ bits},$$

$$Gain(Type) = 1 - \left[\frac{2}{12}B(\frac{1}{2}) + \frac{2}{12}B(\frac{1}{2}) + \frac{4}{12}B(\frac{2}{4}) + \frac{4}{12}B(\frac{2}{4})\right] = 0 \text{ bits}$$



Entropy and Information Gain

The features with the highest information gain should be at the top of the decision tree:

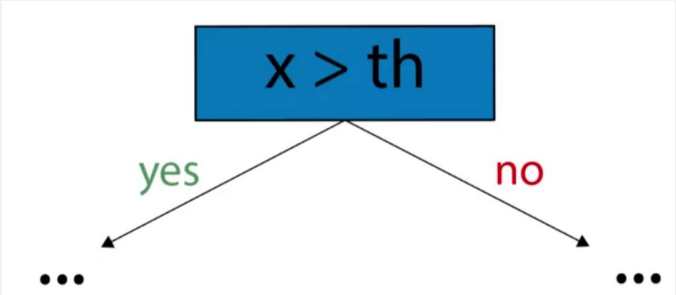


DTs with continuous values

For continuous variables, we should divide their range into two intervals by defining a threshold.

It is advised that the values taken by the variables are ordered, several potential thresholds are calculated as the average of each consecutive values of the attribute.

We should choose the value of the threshold that maximizes the gain.



Random Forests

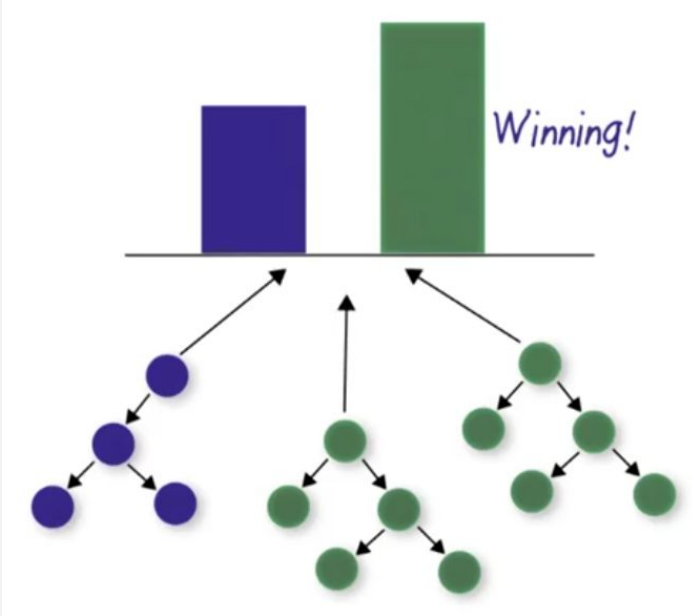
Random Forests

Random Forests consist in ensembling different decision trees and have them vote on the answer.

Bagging Bootstrap Aggregation:

- Input: Data set of size n times from data.
- Sample m times from attributes.
- Learn Tree on sampled data and attributes.

→ Does better than one decision tree since the random sampling help avoiding overfitting.

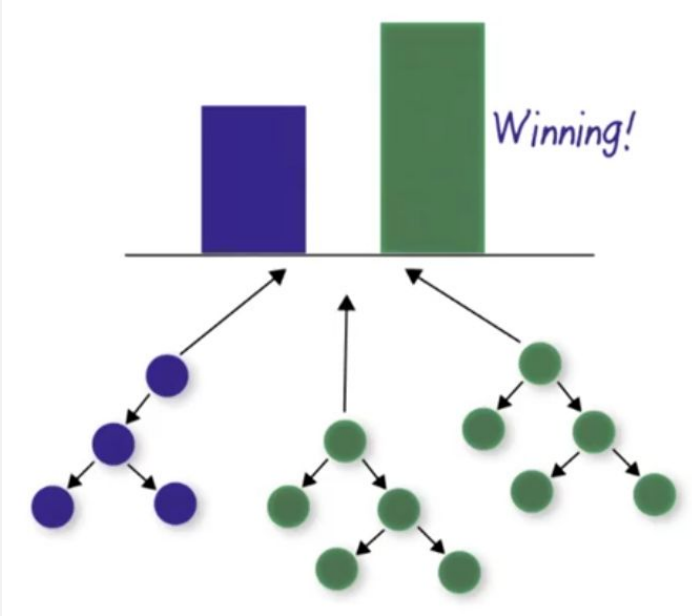


Random Forests

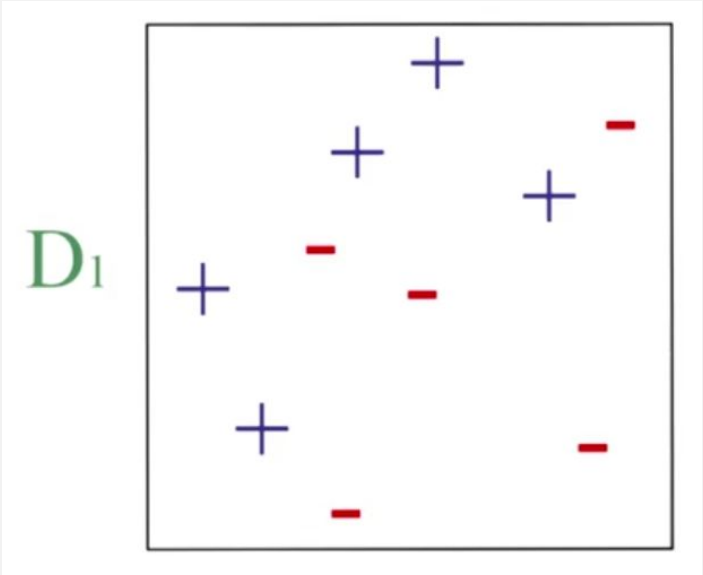
Decision Trees are very helpful in feature selection problems as they identify which features are more important

Most decision tree tools allow you to specify the number of decision tree leaves it is allowed to use.

By setting it to a small number, we get trees that are easy to examine. And by doing a lot of trees, or by doing a random forest we can see how stable these features are.



Boosting

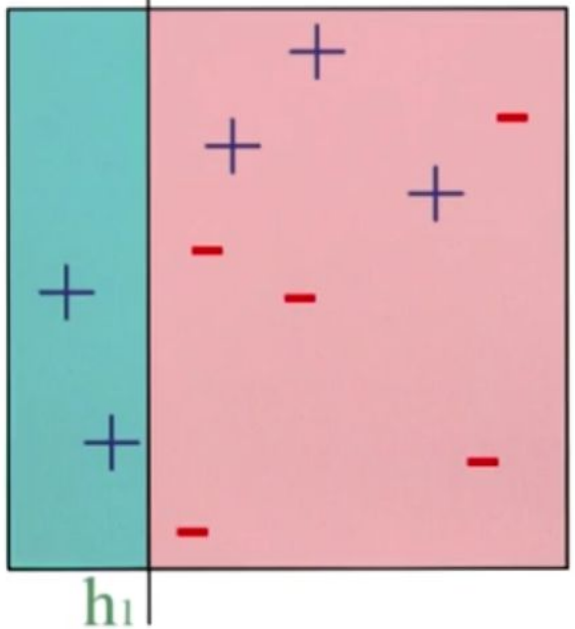


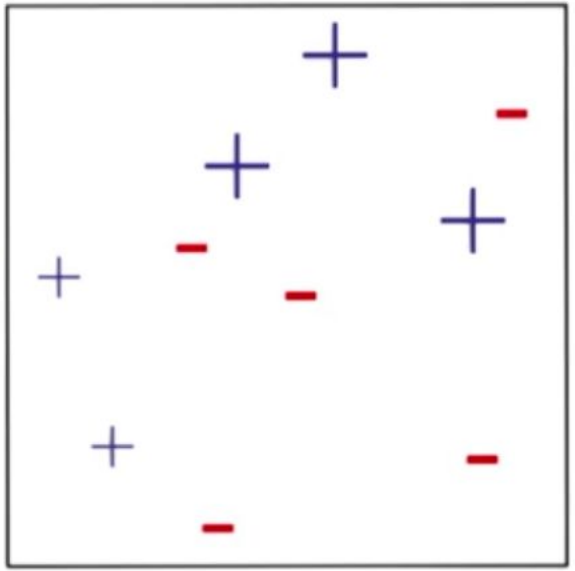
Boosting

$\epsilon_1 = 0.30$

$\alpha_1 = 0.42$

$$\alpha_t = \frac{1}{2} ln(\frac{1 - \epsilon_t}{\epsilon_t}) > 0$$





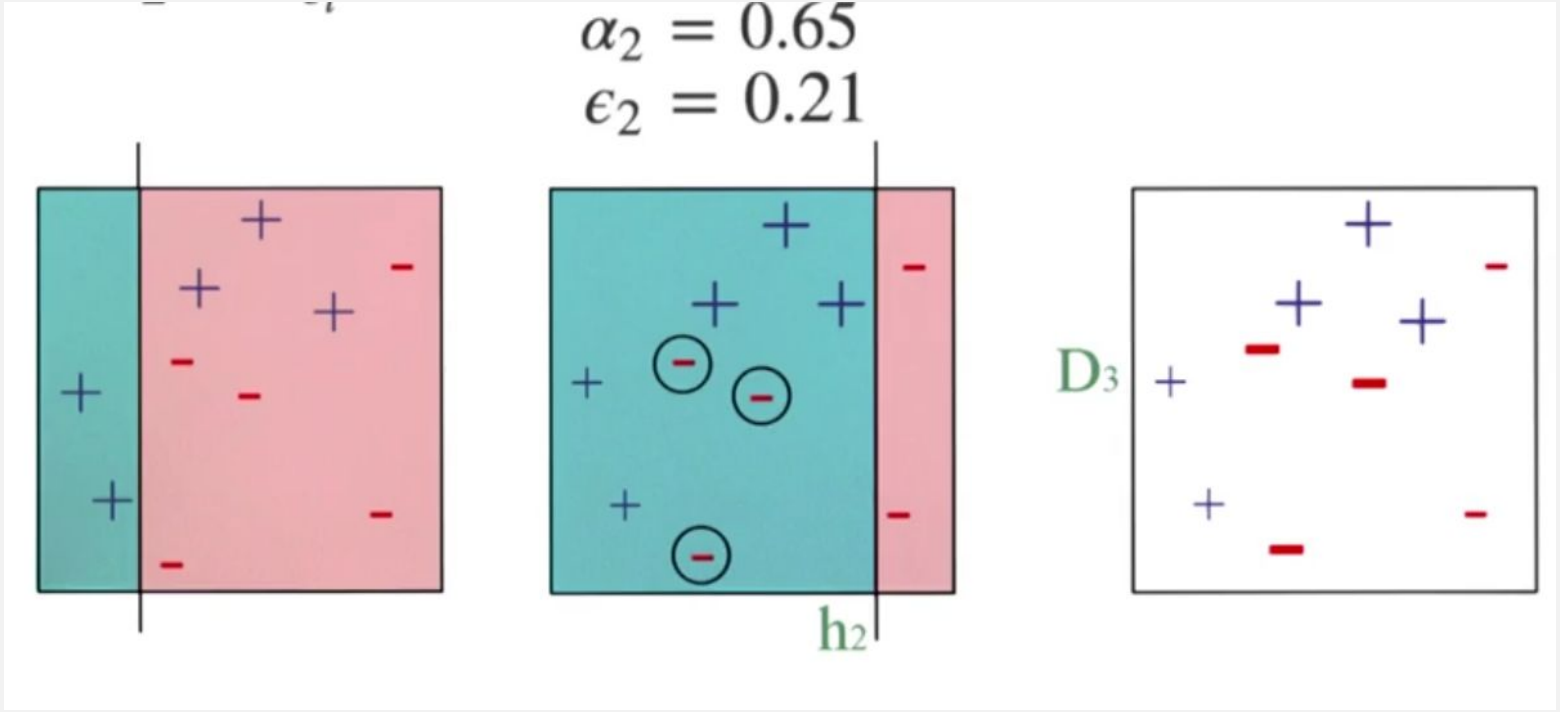
Boosting

The alpha calculated value has two roles:

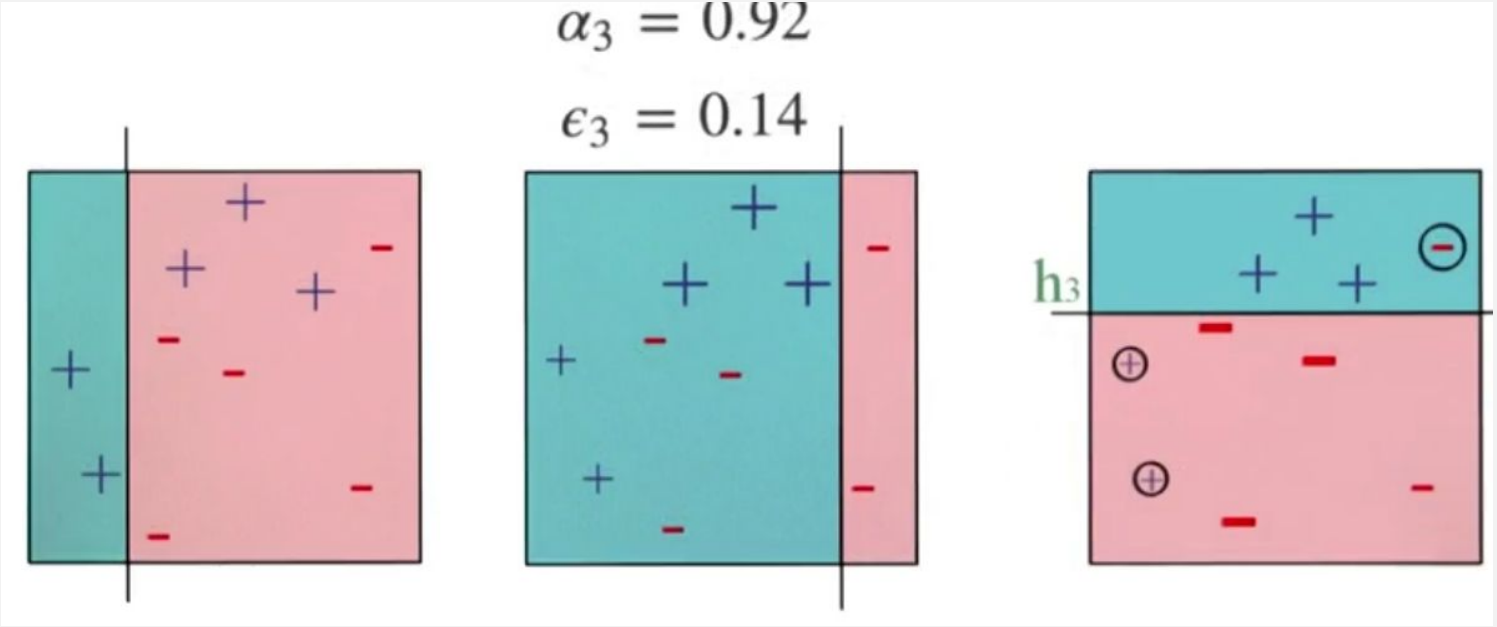
1. It will give us later the weight of the vote of the weak classifier

2. It gives us weights by which we increase the training examples we got wrong (multiply by exp(αt)), and decrease the importance of examples we got right (multiply by exp(-αt)) for the next iteration.

Boosting



Boosting



Stopping condition

We keep modeling the error during boosting and we stop when it converges.