Introdução à Aprendizagem Automática (IAA)

SUSANA BRÁS

SUSANA.BRAS@UA.PT

IAA – L5

Naïve Bayes Classifier

- Bayes Therorem
- Conditional Probability
- Classifier rational
- Assumption

Decision Tree Classifier

- Classifier rational
- Definitions: leaf, node, root, split, homogenous, impurity
- Identification of the best split
- Measures of impurity (non-homogeneity)
- Information gain, information ration
- Tree pruning
- Overfitting
- Algorithm steps

Linear Support Vector Machine (SVM)

- Decision boundary
- Boundary margin
- Support vectors
- SVM goal
- Cost function
- C parameter

Data Matrix

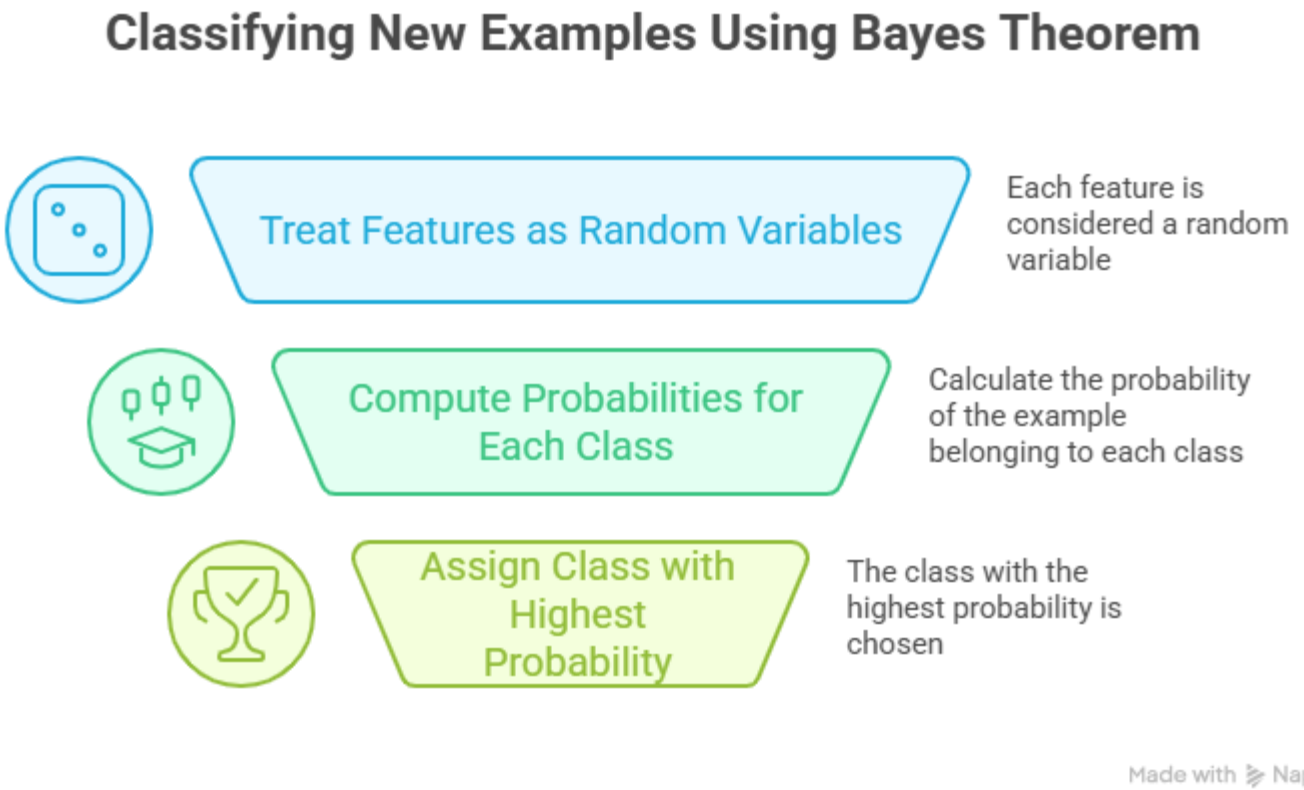
matrix X (mxn)	feature x_1	feature x_2	.....	feature x_n	Target Y
Example 1	$x_1^{(1)}$	$x_2^{(1)}$		$x_n^{(1)}$	$Y^{(1)}$
Example 2	$x_1^{(2)}$	$x_2^{(2)}$		$x_n^{(2)}$	$Y^{(2)}$
…					
Example i	$x_1^{(i)}$	$x_2^{(i)}$		$x_n^{(i)}$	$Y^{(i)}$
…					
…					
Example m	$x_1^{(m)}$	$x_2^{(m)}$		$x_n^{(m)}$	$Y^{(m)}$

x – input vector of features, attributes
y – output vector of labels, ground truth, target
m - number of training examples
n – number of features
$h_\theta(x)$ - model (hypothesis)
θ - vector of model parameters

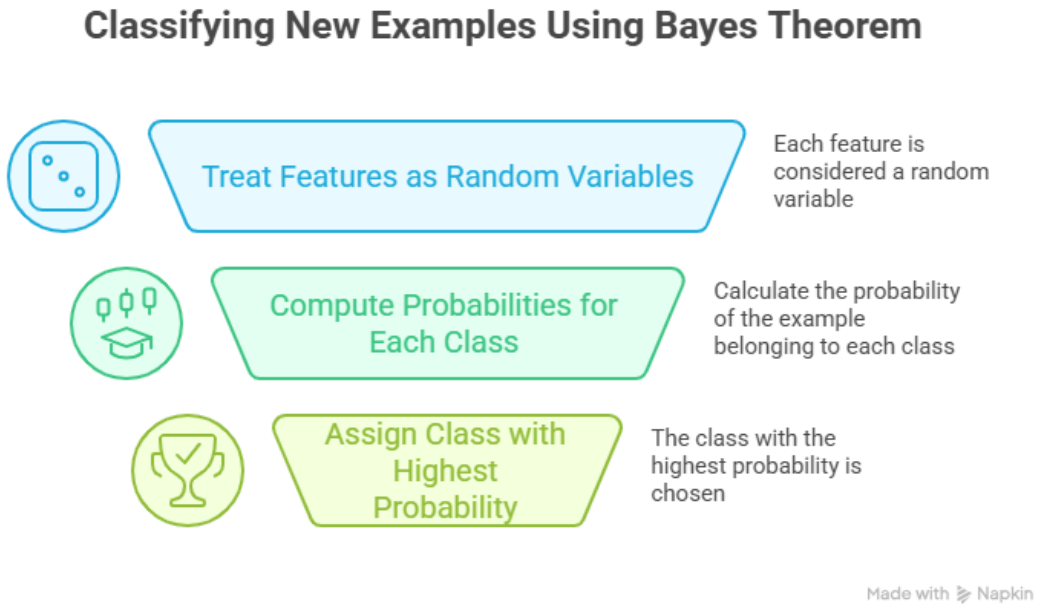
Training set: data matrix X (m rows, n columns)

Naïve Bayes Classifier

Bayesian Classifier



Bayesian Classifier



Use Bayes Theorem to compute the *a posteriori* probability

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

$$x_{new} \Rightarrow C_i : \quad P(C_i|x_{new}) > P(C_j|x_{new}), \quad \forall j \neq i$$

Nonparametric probability estimation -example

	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Prior probability of the class:

P(Class = No) = ?

P(Class = Yes) = ?

P(Status='Married'|Class No) = ?

P(Status='Married'|Class Yes) = ?

Nonparametric probability estimation -example

	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$P(C_i) = \frac{\text{N°of train examples that belong to } C_i}{\text{N°of all train examples}}$$

$$P(x=\text{'MaritalStatus'} \mid C_i) =$$

$$\frac{\text{\# of train examples that belong to Ci \& has this 'MaritalStatus'}}{\text{Number of all train examples that belong to Ci}}$$

P(Class = No) = 7/10
P(Class = Yes) = 3/10

Feature *Marital Status* has 3 string values :single, married, divorced

P(Status='Married'|Class No) = 4/7

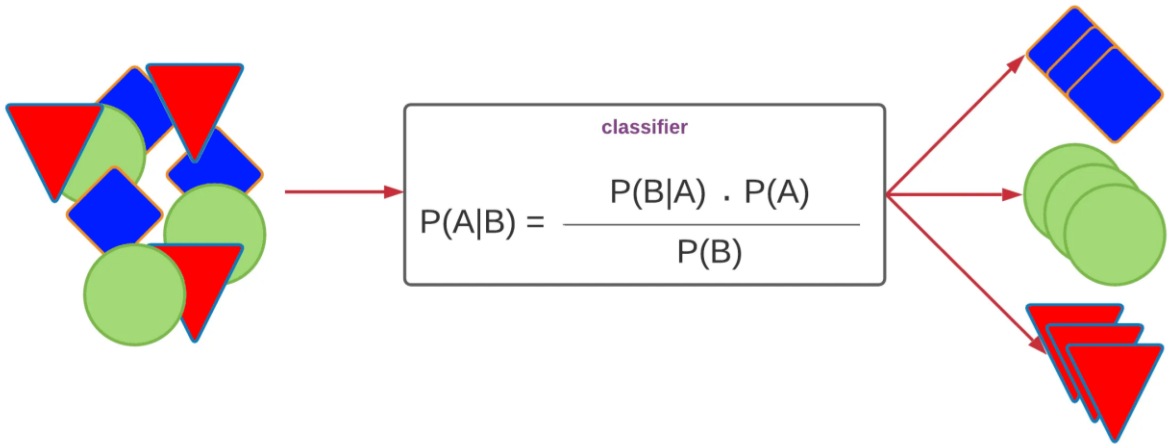
P(Status='Married'|Class Yes) = 0/3

The same counting is done for each string value of the feature, and for each feature.

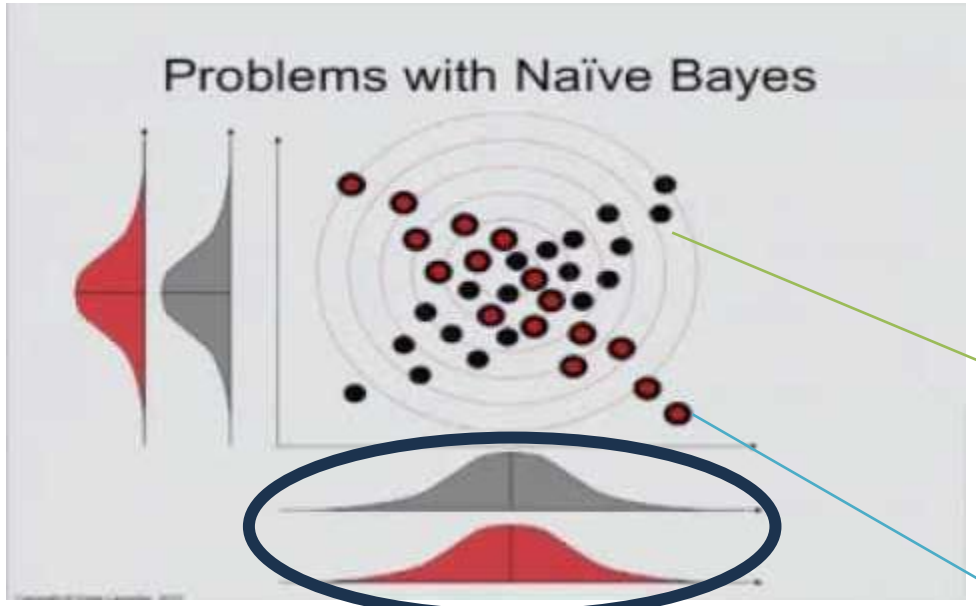
Naive Bayes (NB) Classifier -problems

Assumptions:

1. **Features are independent:** Each feature works on its own.
2. **Continuous features follow a normal distribution:** If a feature is a number (like height or age), it's assumed to follow Gaussian distribution.
3. **Discrete features follow a multinomial distribution:** If a feature is a category (like color or word count), it's modeled using a multinomial formula.
4. **All features are equally important:** Every feature has the same weight in deciding the outcome.
5. **No missing values:** The data must be complete (no blanks or empty entries).



NB will not be able to separate these classes



2 features

2 classes

This class has a positive correlation between the features (if x1 increases, x2 also increases)

This class has a negative correlation between the features (if x1 increases, x2 decreases)

Both classes present the same Gaussian Distribution

Naive Bayes (NB) Classifier

It is based on Bayes' theorem, which is a mathematical formula that describes the probability of an event occurring given the knowledge of other events.

How Naive Bayes works
The Naive Bayes classifier works by calculating the probability of each class given the input features. The class with the highest probability is then predicted as the output.

Pros

- It is easy and fast to predict a class of test data set.
- Naive Bayes classifier performs better compared to other models assuming independence.
- It performs well in case of categorical data as compared to numeric data.
- It requires less training data.

Cons

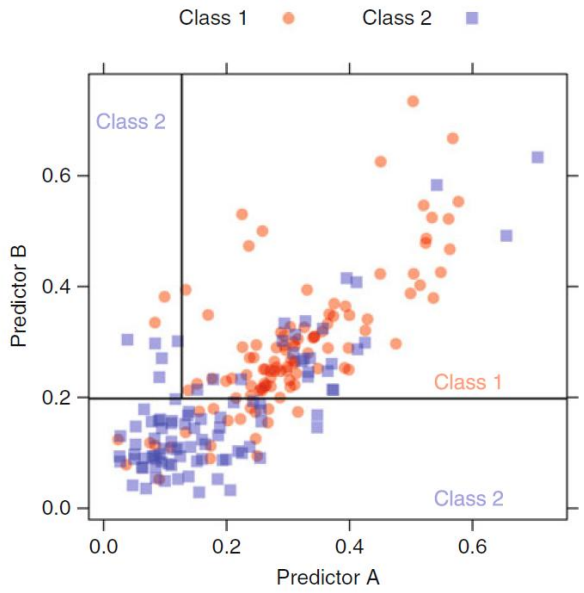
- If an instance in test data set has a category that was not present during training, then it will assign it "Zero" probability and won't be able to make prediction. This is known as **Zero frequency problem.**
- It is also known as a bad estimator.
- It solely relies on the independence predictor assumption.

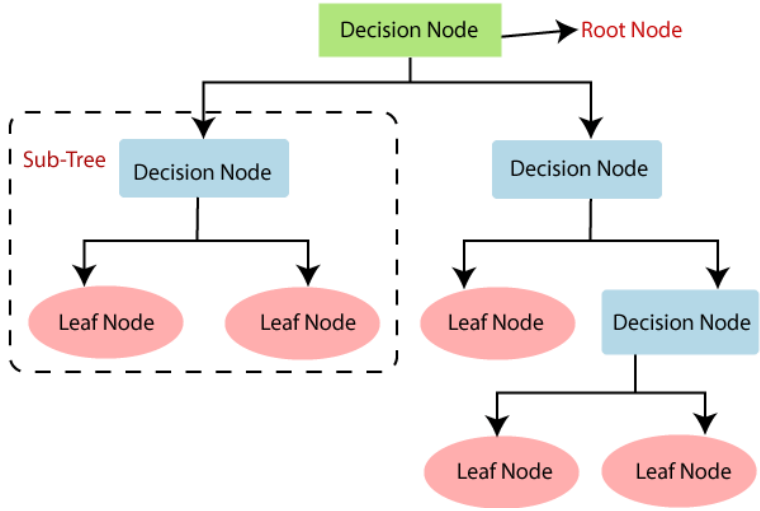
Decision Tree

Decision Tree

Tree-based models consist of one or more nested if-then statements for the predictors that partition the data.

```
if Predictor B >= 0.197 then
|   if Predictor A >= 0.13 then Class = 1
|   else Class = 2
else Class = 2
```

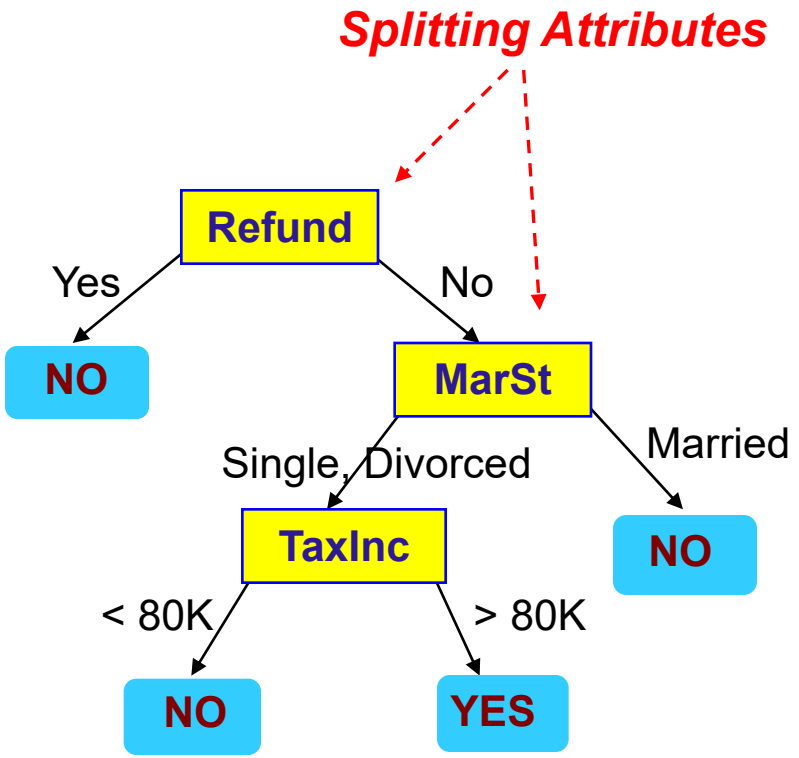




Classification by Decision Tree – model 1

	categorical	categorical	continuous	class
	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

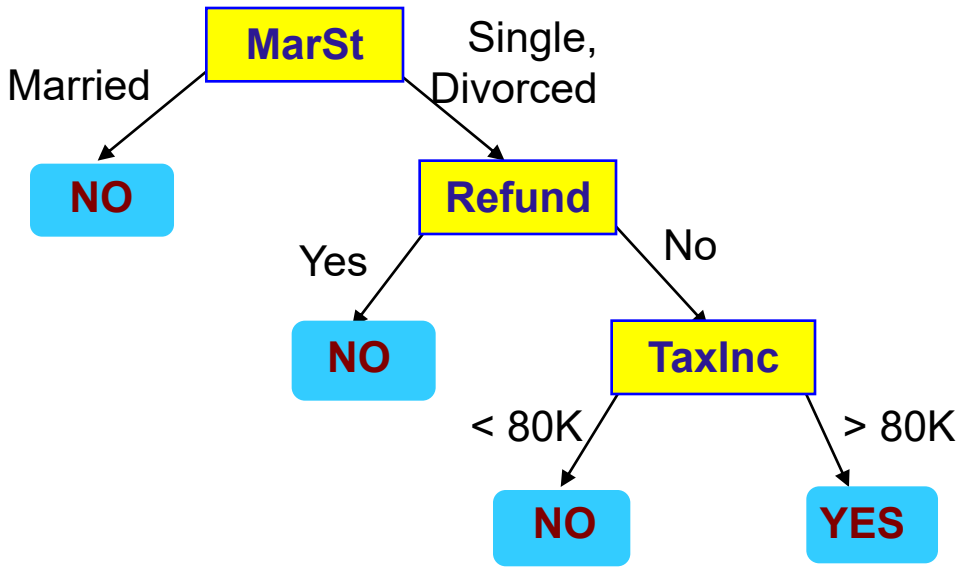


Model: Decision Tree

Classification by Decision Tree – model 2

		categorical	categorical	continuous	class
	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



There could be more than one tree that fits the same data!

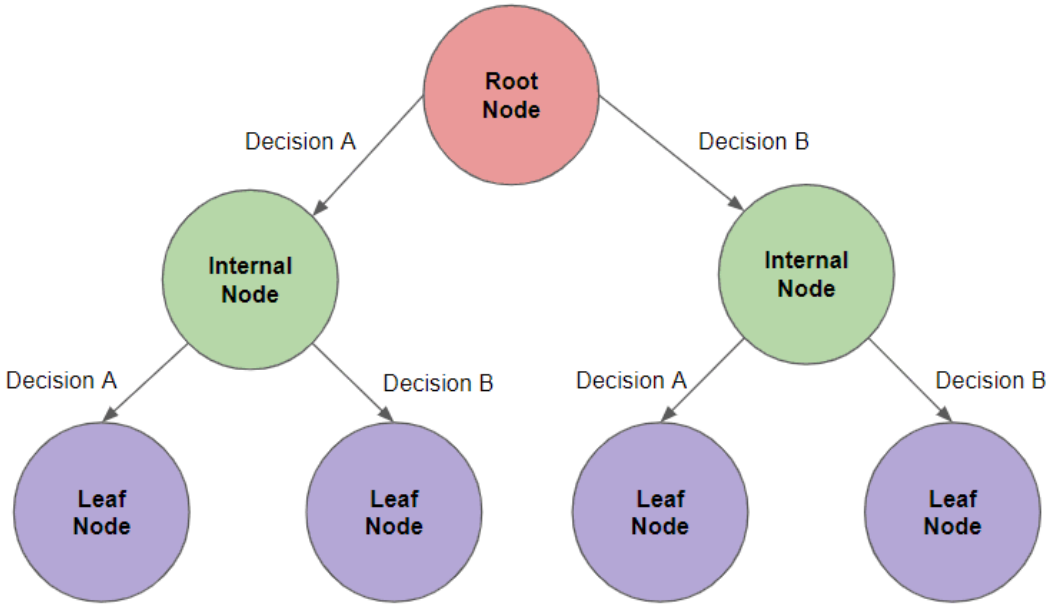
Decision Tree

Decision Tree has 2 basic type of nodes:

- ✓ **Leaf node** - a class label, determined by majority vote of training examples reaching that leaf.
- ✓ **Node** is a question on one feature. It branches out according to the answers.

  - **root node**: the first (top) node of the tree

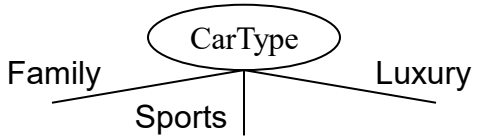
  - **child node**: the next node to a current node



Splitting

Categorical Features

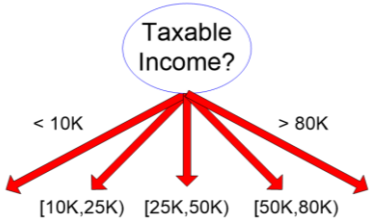
Multi-way split: Use as many partitions as distinct values.



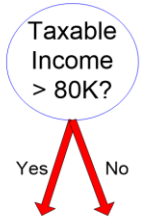
Binary split: Divides values into two subsets.

Categorical Features

Multi-way split: Use as many partitions as defined intervals.



Binary split: Divides values into two subsets.



How to determine the best split ?

Requirements:

- Obtaining the smallest tree
- Pure leaf nodes, i.e. all examples at a leaf having (almost) the same class (ex. C0 or C1).
- Select the feature that produces the “purest” (the most homogeneous) nodes
- Define a measure of node impurity (homogeneity).

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

C0: 9
C1: 1

Homogeneous,

Low degree of impurity

Measures of Node Impurity

Gini Index at a given node:

$$\mathrm{GINI}(\mathrm{node}) = 1 - \sum_{\mathrm{class_j}} \left[\mathrm{p}(\mathrm{Class_j}|\mathrm{node})\right]^2$$

Classification error at a given node:

$$\mathrm{Error}(\mathrm{node}) = 1 - \max_{\mathrm{Class_j}} \mathrm{p}(\mathrm{Class_j}|\mathrm{node})$$

Entropy (H) at a given node:

$$\mathrm{H}(\mathrm{node}) = -\sum_{\mathrm{class_j}} \mathrm{p}(\mathrm{Class_j}|\mathrm{node}) \log(\mathrm{p}(\mathrm{Class_j}|\mathrm{node}))$$

$\mathrm{p}(\mathrm{Class_j}|\mathrm{node})$: probability of $\mathrm{Class_j}$ at a given *node*

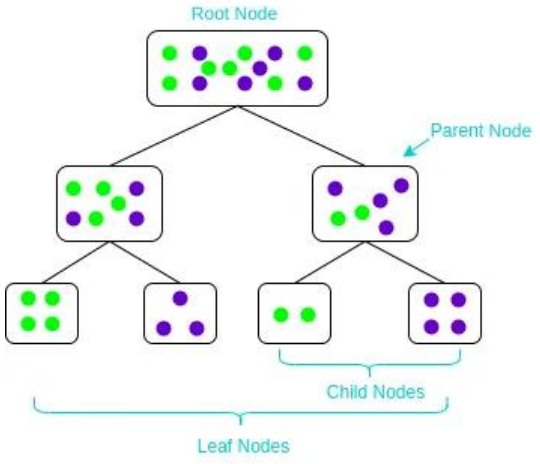
Building a Decision Tree

The Decision Tree classifier operates by recursively splitting the data based on the most informative features.

1. Start with the entire dataset at the root node.
2. Select the best feature to split the data (based on measures like Gini impurity).
3. Create child nodes for each possible value of the selected feature.
4. Repeat steps 2–3 for each child node until a stopping criterion is met (e.g., maximum depth reached, minimum samples per leaf, or pure leaf nodes).
5. Assign the majority class to each leaf node.

Classification Step

1. Start at the root node of the trained decision tree.
2. Evaluate the feature and split condition at the current node.
3. Repeat step 2 at each subsequent node until reaching a leaf node.
4. The class label of the leaf node becomes the prediction for the new instance.



Other Algorithms

- **ID3 (Iterative Dichotomiser 3)** algorithm. Use Information Gain to select the best attribute.
- **Modified ID3** – use Gain ratio.
- **C4.5 (**extension of ID3)- deals with numeric attributes, missing values, noisy data
- **CART (Classification And Regression Tree)** – similar approach
- **Random Forest** – ensemble of DT classifiers

Remark: There are many other feature selection criteria!

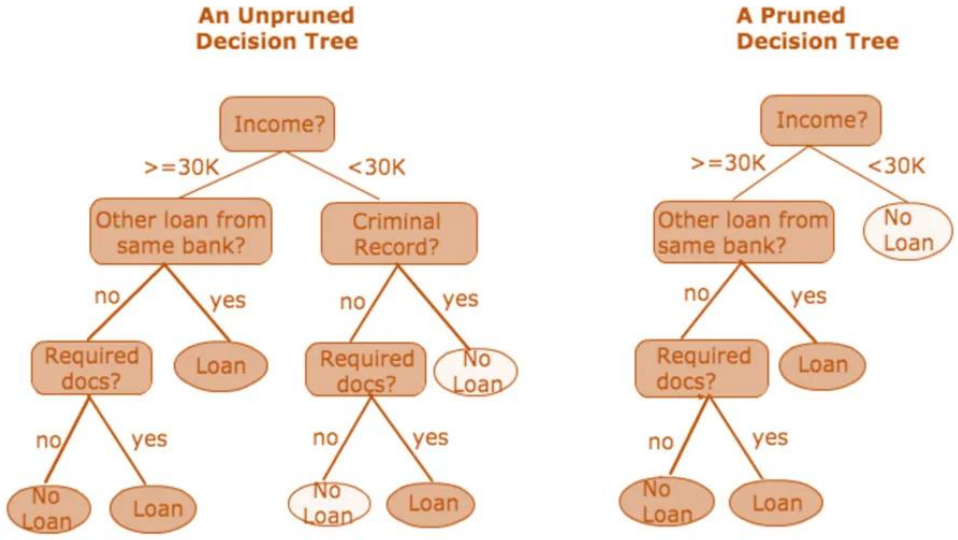
Gain ratio is a modification of the splitting criteria to deal better with overfitting. Gain ratio takes number and size of branches into account when choosing the most favorable feature to split at a node.

Gain ratio =Information_Gain / Split Entropy

Pruning

The overfitting prevention in decision tree, is usually based on pruning.

- **Post-pruning** - First, build full decision tree and then discard unreliable parts
- **Pre-pruning** – stop growing a branch when information gain does not change significantly
- **Post-pruning preferred in practice—pre-pruning can “stop too early”**



Decision Tree

Tree-based and rule-based models are popular modeling tools:

- They generate a set of conditions that are highly **interpretable** and are **easy to implement**.
- Because of the logic of their construction, they can effectively handle **many types of predictors** (sparse, skewed, continuous, categorical, etc.) without the need to pre-process them.
- These models can effectively **handle missing data** and implicitly conduct feature selection.

Models based on single trees or rules, however, do have particular weaknesses:

- **model instability** (i.e., slight changes in the data can drastically change the structure of the tree or rules and, hence, the interpretation).
- **less-than-optimal predictive performance**. This is due to the fact that these models define rectangular regions that contain more homogeneous outcome values. If the relationship between predictors and the response cannot be adequately defined by rectangular subspaces of the predictors, then tree-based or rule-based models will have larger prediction error than other kinds of models.
- to combat these problems, researchers developed **ensemble methods** that combine many trees (or rule-based models) into one model. Ensembles tend to have much better predictive performance than single trees

Decision Tree

Considerations

- Decision trees tend to **overfit** on data with a **large number of features**. Getting the right ratio of samples to number of features is important, since a tree with few samples in high-dimensional space is very likely to overfit.
- Consider performing **dimensionality reduction**.
- The **number of samples required to populate the tree doubles for each additional level** the tree grows to. It is important to **control the size** of the tree to prevent overfitting.
- **Balance your dataset** before training to prevent the tree from being biased toward the dominant classes. (NOTE: Ensemble trees, like random forest are immune to this problem)

Decision Tree

Key Parameters
Decision Trees have several important parameters that control their growth and complexity:
1 . **Max Depth**: This sets the maximum depth of the tree, which can be a valuable tool in preventing overfitting.
2. **Min Samples Split**: This parameter determines the minimum number of samples needed to split an internal node.
3. **Min Samples Leaf**: This specifies the minimum number of samples required at a leaf node.
4. **Criterion**: The function used to measure the quality of a split (usually "gini" for Gini impurity or "entropy" for information gain).

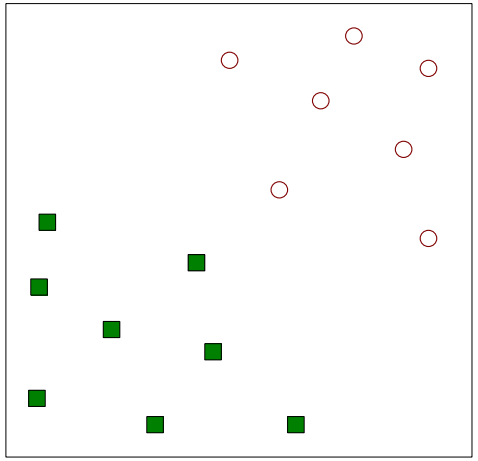
A decision tree is a non-parametric supervised learning algorithm that can be used for both classification and regression. It uses a tree-like structure to represent decisions and their potential outcomes. Decision trees are simple to understand and interpret and can be easily visualized. However, when a decision tree model becomes too complex, it does not generalize well from the training data and results in overfitting.

Pros:
1.Interpretability: Easy to understand and visualize the decision-making process.
2.No Feature Scaling: Can handle both numerical and categorical data without normalization.
3.Handles Non-linear Relationships: Can capture complex patterns in the data.
4.Feature Importance: Provides a clear indication of which features are most important for prediction.

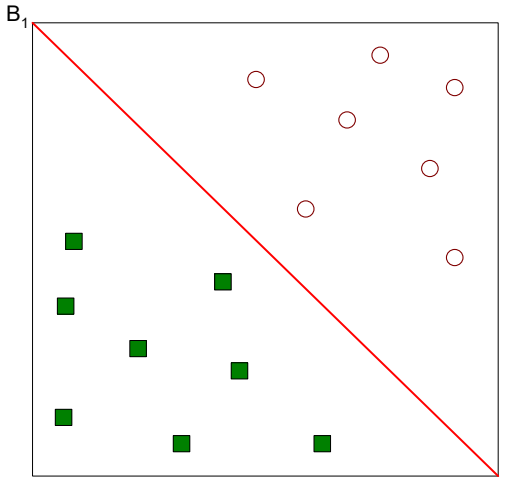
Cons:
1.Overfitting: Prone to creating overly complex trees that don't generalize well, especially with small datasets.
2.Instability: Small changes in the data can result in a completely different tree being generated.
3.Biased with Imbalanced Datasets: Can be biased towards dominant classes.
4.Inability to Extrapolate: Cannot make predictions beyond the range of the training data.

Linear Support Vector Machine (SVM)

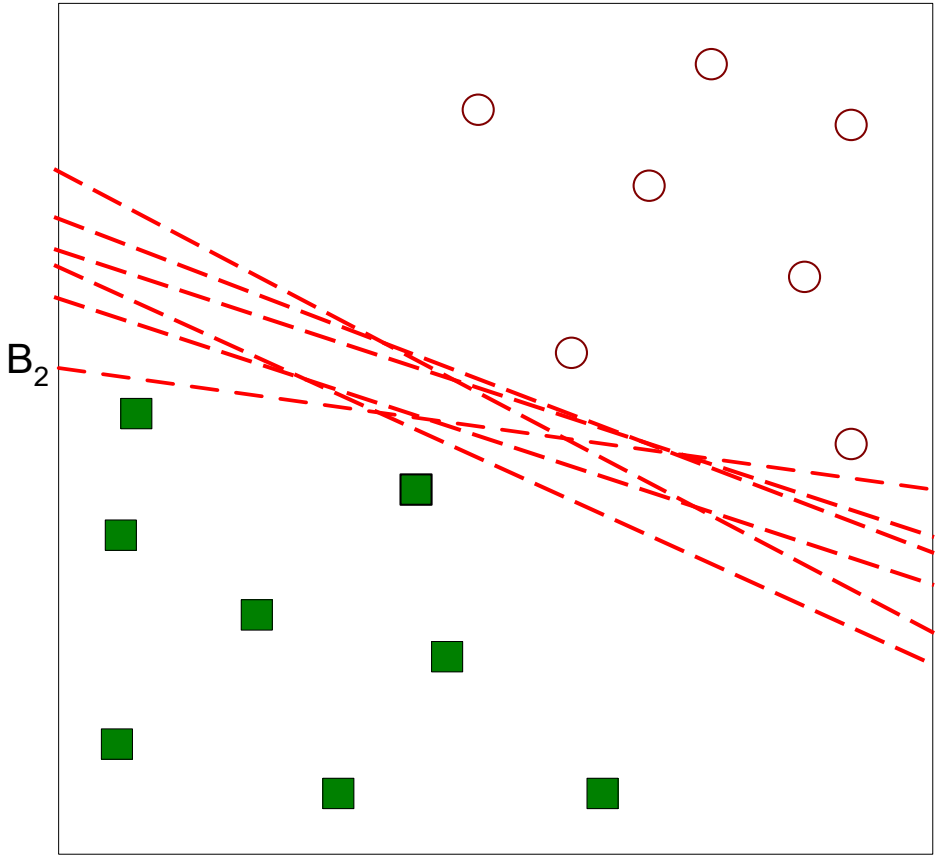
SVM – Linearly Separable Classes



Linearly Separable

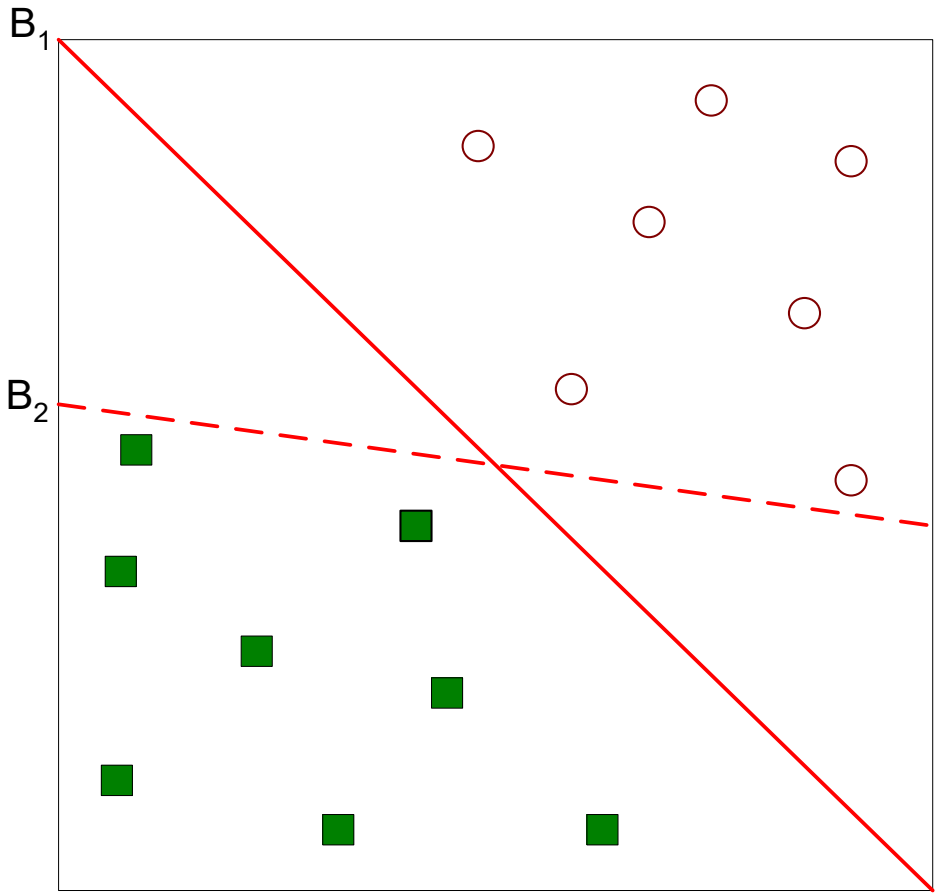


One Possible Solution



Many Possible Solutions

SVM – Linearly Separable Classes

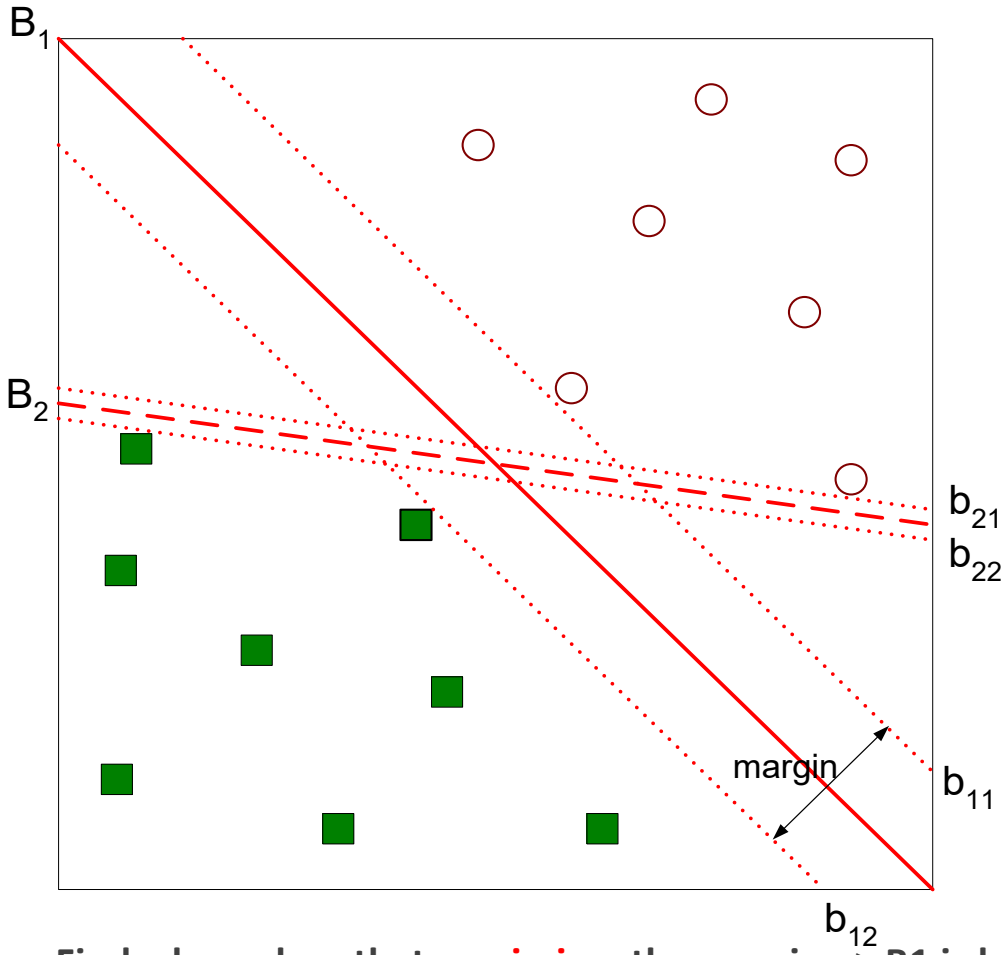


Which one is better? B1 or B2?

SVM – Linearly Separable Classes

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



Find a boundary that maximizes the margin => B1 is better than B2

Classification – SVM

Why?

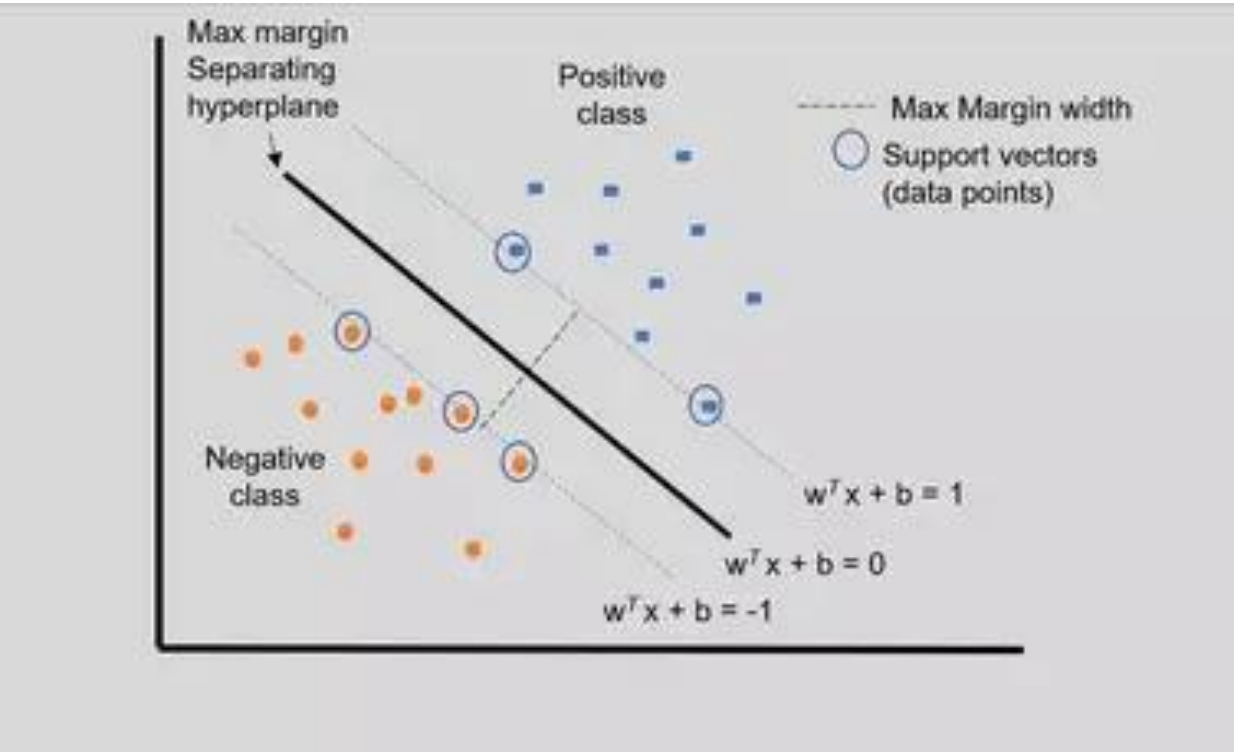
Only the closest points (support vectors) from each class are used to decide which is the optimum (the largest) margin between the classes.

The dimension of the hyperplane depends upon the number of features.

If the number of input features is 2, then the hyperplane is just a line.

If the number of input features is 3, then the hyperplane becomes a two-dimensional plane.

…



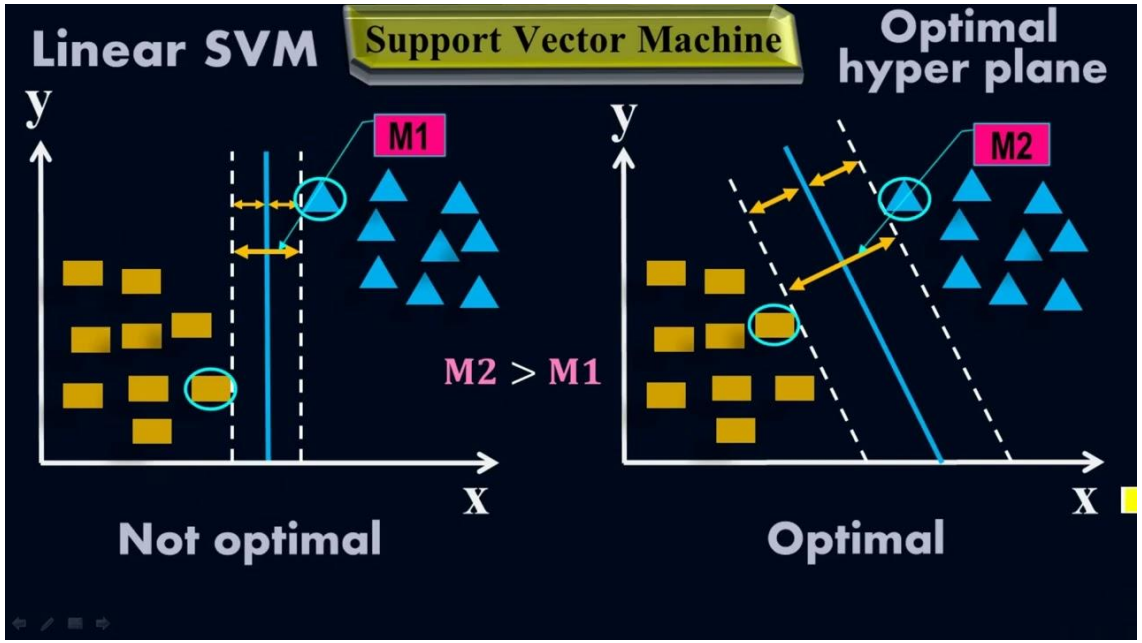
Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.

Using these support vectors, we maximize the margin of the classifier.

Deleting the support vectors will change the position of the hyperplane.

Classification – SVM

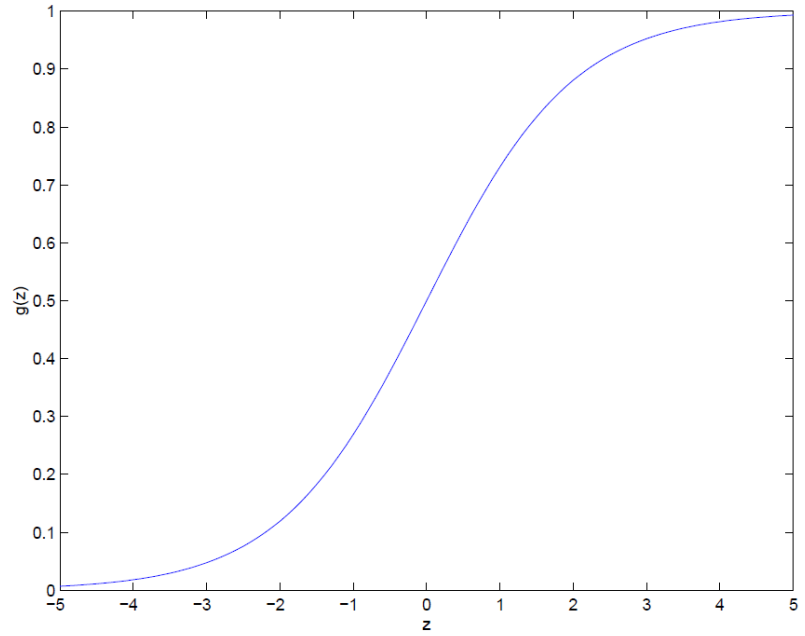
Why?



What is this function?

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

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots . \theta_n x_n$$

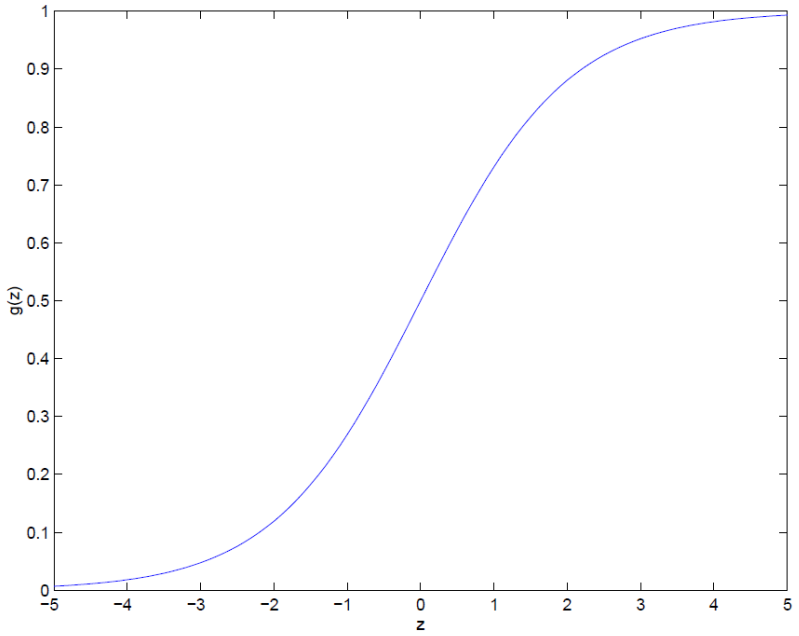


Logistic Regression (LogReg) -revised

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

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots . \theta_n x_n$$

$$\text{if } y = 1,\ h_\theta(x) \approx 1, \quad \theta^T x >> 0$$
$$\text{if } y = 0,\ h_\theta(x) \approx 0, \quad \theta^T x << 0$$



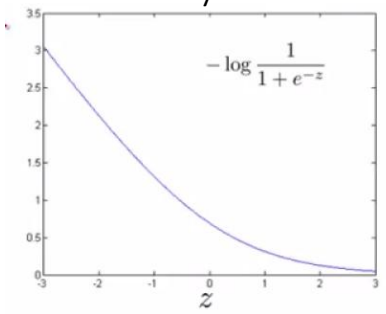
Logistic (sigmoid) function

SVM cost function

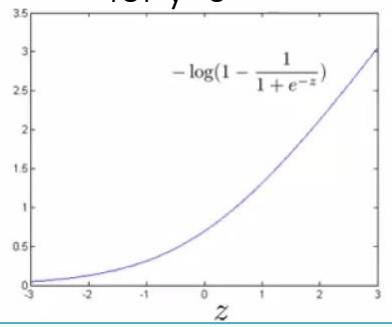
Regularized LogReg cost function:

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

for y=1



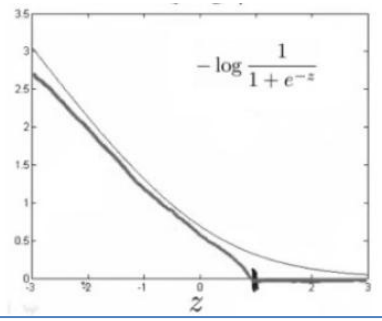
for y=0

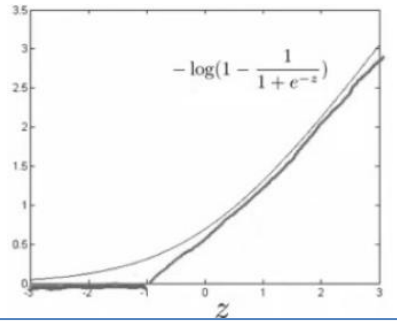


Regularized SVM cost function:

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

(Modification of LogReg cost function, *cost0* & *cost1* are assimptotic safety margins with computational advantages)





$$z = \theta^T x$$

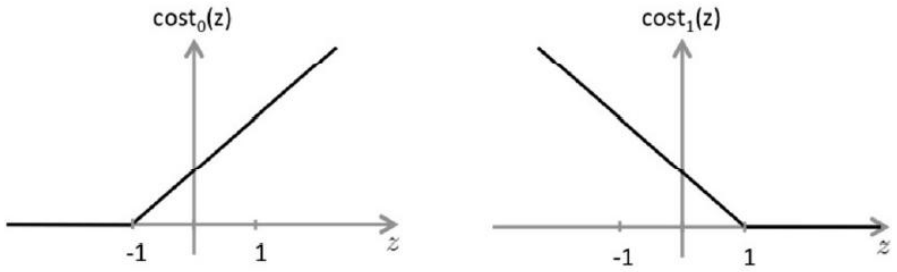
SVM cost function

Regularized LogReg cost function:

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

Regularized SVM cost function:

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

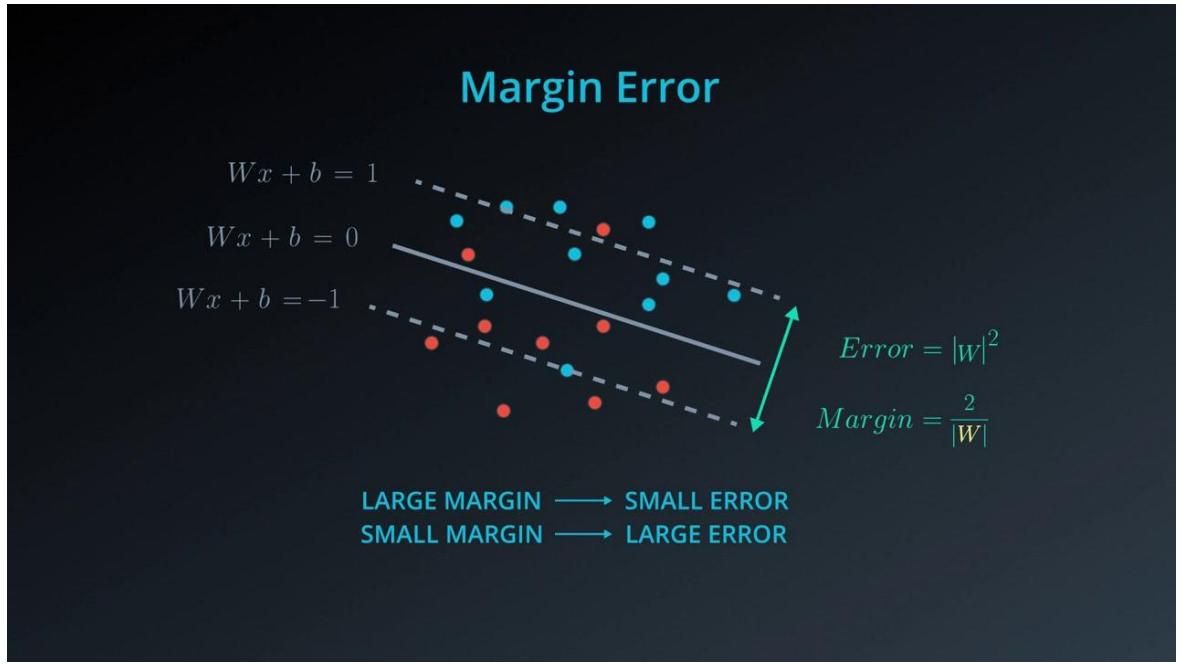


$$z = \theta^T x$$

Different way of parameterization: ***C* is equivalent to 1/λ.**

***C* >0** - parameter that controls the penalty for misclassified training examples.
Increase *C* - more importance to training data fitting.
Decrease C – more importance to generalization properties (combat overfitting).

SVM algorithm



GOAL: *Search for the largest margin that minimizes the classification error.*

C controls how wide is the "street".

Two quantities to optimize:

classification error (how many points are wrongly classified)

"margin error" (optimize the margin between the two classes)

Mathematical formulation

$\theta^T x => Wx + b$

$$\min_{W} \sum_{j=1}^{n} W_j^2 = |W|^2$$

$(L_2 \text{ norm})$ such that

$Wx^{(i)} + b \geq 1, \quad \text{if } y = 1$

$Wx^{(i)} + b \leq -1 \quad \text{if } y = 0$

Linear SVM - Summary

- Linear SVM finds the best straight-line boundary to separate data.
- It maximizes the margin (the gap between the boundary and the closest points).
- The support vectors help define this optimal boundary.

Advantages:

- Works Great with High-Dimensional Data
- Finds the Best Possible Decision Boundary
- Robust to Outliers (to Some Extent)

Limitations of Unsupervised Learning

- Only Works Well with Linearly Separable Data.
- Struggles with Large Datasets.
- Sensitive to Overlapping Data