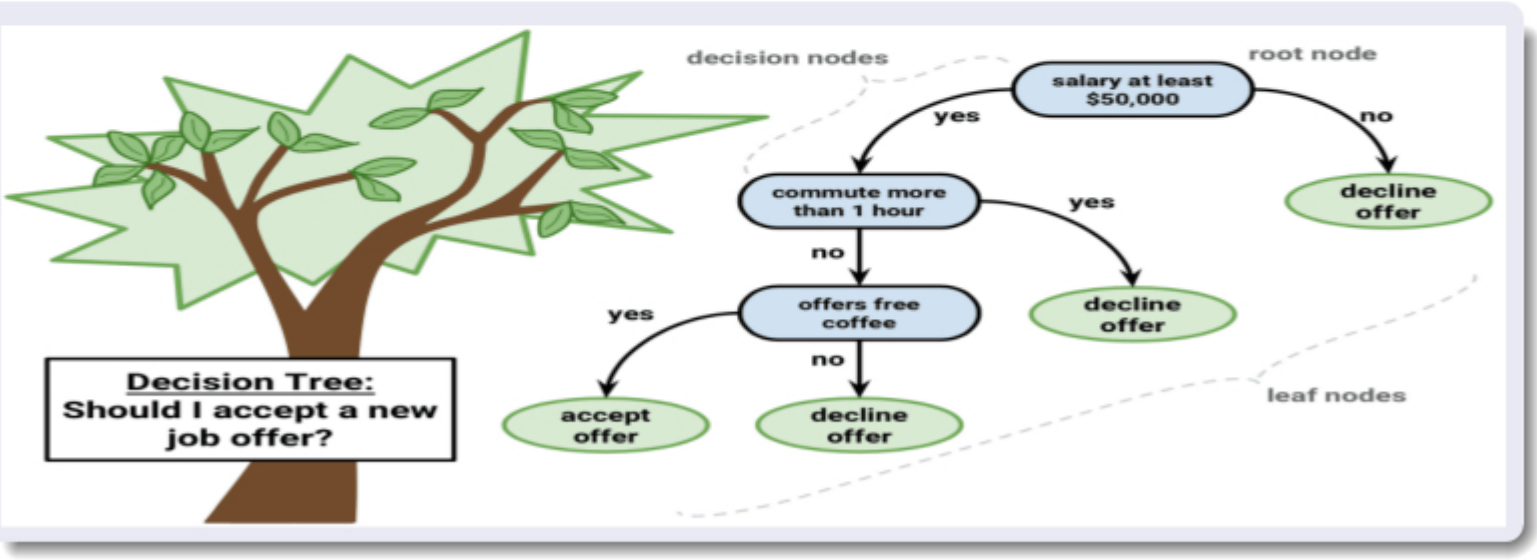
Decision Trees

September 27, 2022

Introduction

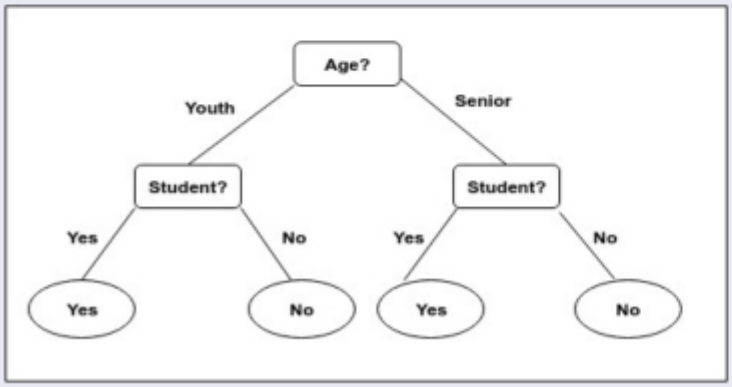
- A decision tree is a flowchart like tree structure
  - each internal node (non-leaf node) denotes a test on an attribute,
  - each branch represents an outcome of the test, and
  - each leaf node (or terminal node) holds a class label

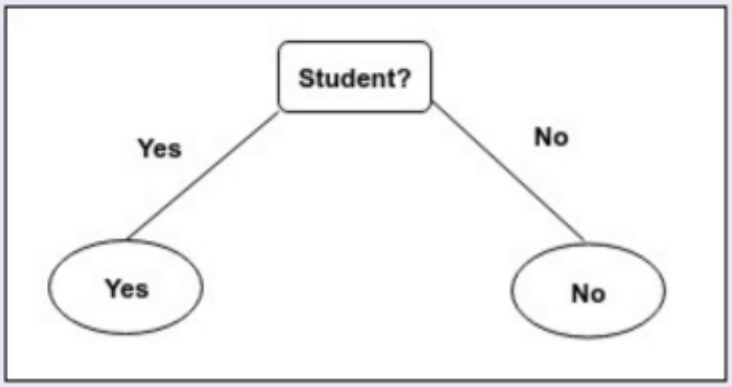


AllElectronics Customer Database

RID	Age	Student	Class:Buy?
1	youth	no	no
2	youth	no	no
3	senior	no	no
4	senior	yes	yes
5	senior	yes	yes
6	youth	no	no
7	youth	yes	yes
8	senior	yes	yes
9	youth	yes	yes
10	senior	no	no

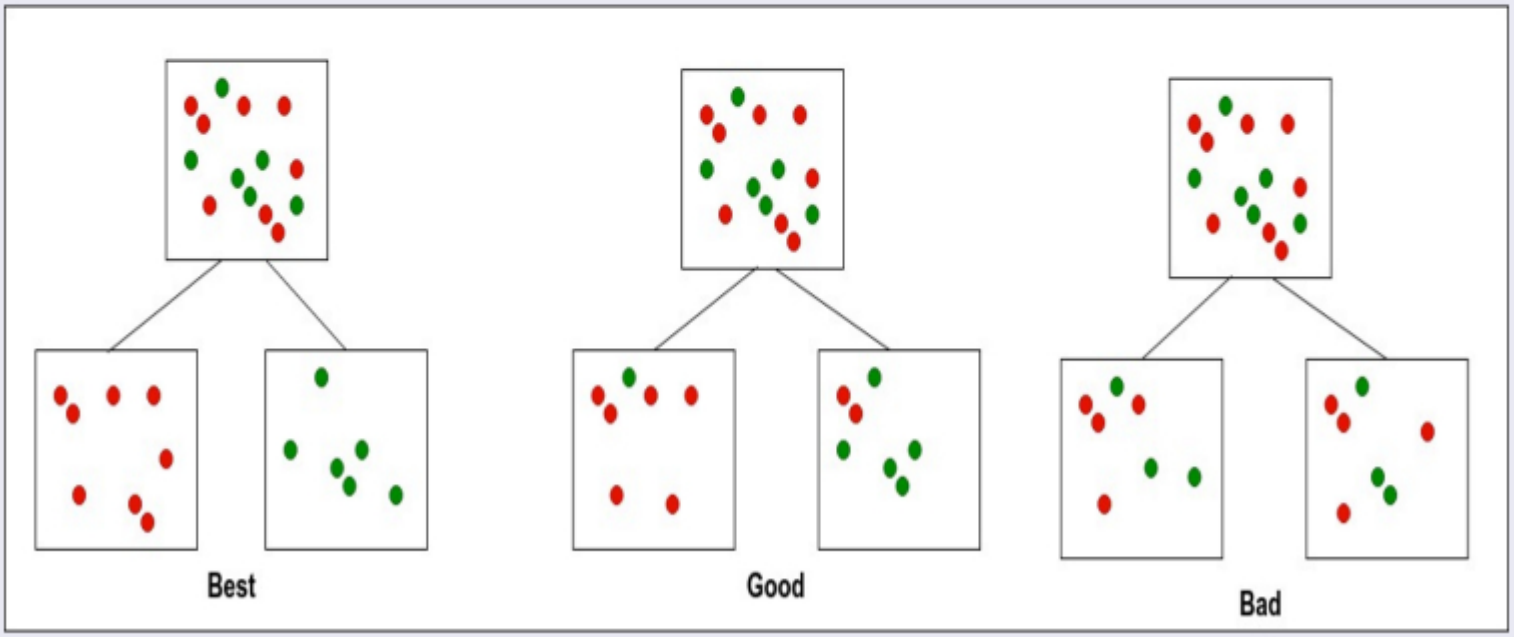
Two Decision Trees for the Same Data





Attribute Selection

- Selecting the best attribute for use at each stage of the decision tree induction is crucial.
- The attribute should be chosen that it separates the data points into classes which are as pure as possible.



Purity of a Partition - Entropy

- The purity of a data partition can be approximated by using Shannon's Entropy (commonly called Information Content in the context of data mining).
- It is a measure of homogeneity (or heterogeneity) of a data partition.
- A partition D which contains all items from the same class will have $Info(D) = 0$.
- A partition D with all items belonging to different classes will have the maximum information content.

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(p_i)$$

- Let the dataset D be partitioned on the attribute A, then the information of the new partition is given by

$$Info_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} x\ Info(D_j)$$

and the gain in information

$$Gain(A) = Info(D) - Info_A(D)$$

- The attribute selected at each stage is the one which has the greatest information gain. Decision Tree algorithms which use Information Gain as the attribute selection criterion are called ID3 Algorithms.

AllElectronics Customer Database

RID	Age	Income	Student	Credit Rating	Class:Buy?
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle aged	medium	no	excellent	yes
13	middle aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Info(D) = -\frac{5}{14} log_2\left(\frac{5}{14}\right) - \frac{9}{14} log_2\left(\frac{9}{14}\right) = 0.94$$

AllElectronics Customer Database

Feature under Consideration: Age

RID	Age	Income	Student	Credit Rating	Class:Buy?
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle aged	medium	no	excellent	yes
13	middle aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Info_{age} = \frac{5}{14}\left(-\frac{2}{5} * log_2\left(\frac{2}{5}\right) - \frac{3}{5} * log_2\left(\frac{3}{5}\right)\right) + \frac{4}{14}\left(-\frac{4}{4} * log_2\left(\frac{4}{4}\right)\right) + \frac{5}{14}\left(-\frac{3}{5} * log_2\left(\frac{3}{5}\right) - \frac{2}{5} * log_2\left(\frac{2}{5}\right)\right) = .6935$$

AllElectronics Customer Database

Feature under Consideration: Income

RID	Age	Income	Student	Credit Rating	Class:Buy?
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle aged	medium	no	excellent	yes
13	middle aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Info_{income} = \frac{4}{14}\left(-\frac{1}{2} * log_2\left(\frac{1}{2}\right) - \frac{1}{2} * log_2\left(\frac{1}{2}\right)\right) + \frac{6}{14}\left(-\frac{2}{6} * log_2\left(\frac{2}{6}\right) - \frac{4}{6} * log_2\left(\frac{4}{6}\right)\right) + \frac{4}{14}\left(-\frac{1}{4} * log_2\left(\frac{1}{4}\right) - \frac{3}{4} * log_2\left(\frac{3}{4}\right)\right) = .907$$

AllElectronics Customer Database

Feature under Consideration: Student

RID	Age	Income	Student	Credit Rating	Class:Buy?
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle aged	medium	no	excellent	yes
13	middle aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Info_{student} = \frac{1}{2}\left(-\frac{1}{7} * log_2\left(\frac{1}{7}\right) - \frac{6}{7} * log_2\left(\frac{6}{7}\right)\right) + \frac{1}{2}\left(-\frac{4}{7} * log_2\left(\frac{4}{7}\right) - \frac{3}{7} * log_2\left(\frac{3}{7}\right)\right) = .7875$$

AllElectronics Customer Database

Feature under Consideration: Credit Rating

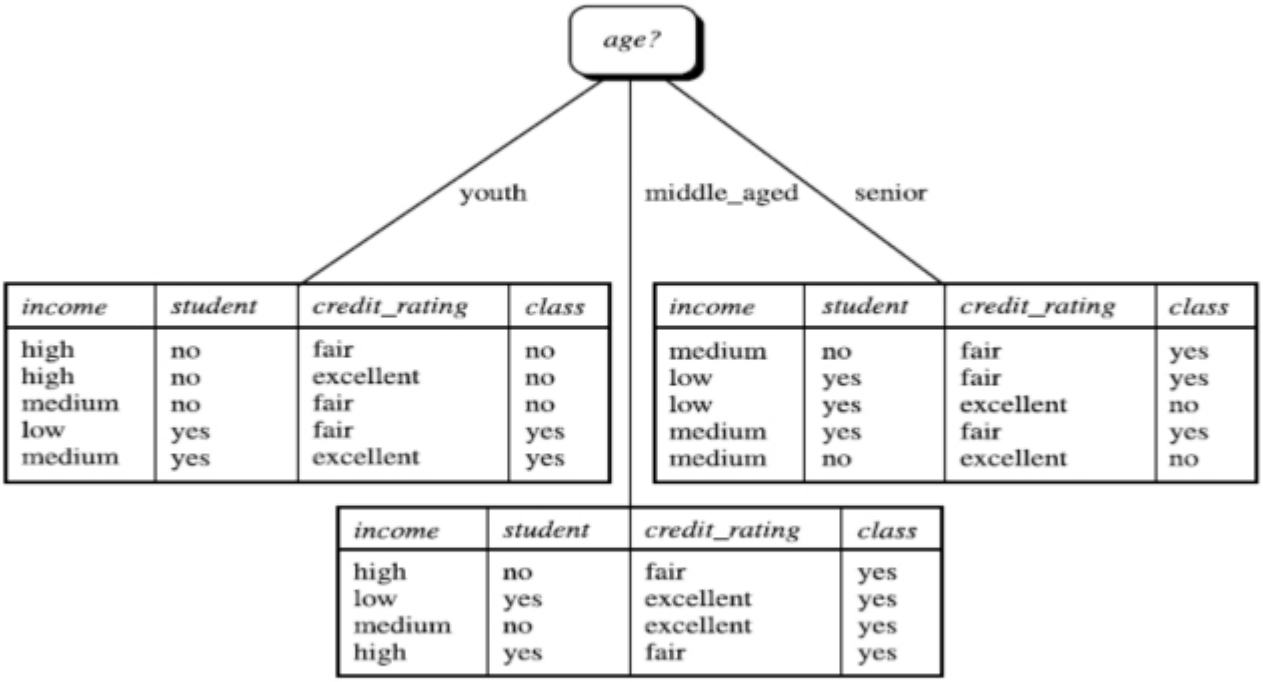
RID	Age	Income	Student	Credit Rating	Class:Buy?
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle aged	medium	no	excellent	yes
13	middle aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Info_{age} = \frac{8}{14}\left(-\frac{2}{8} * log_2\left(\frac{2}{8}\right) - \frac{6}{8} * log_2\left(\frac{6}{8}\right)\right) + \frac{6}{14}\left(-\frac{3}{6} * log_2\left(\frac{3}{6}\right) - \frac{3}{6} * log_2\left(\frac{3}{6}\right)\right) = .892$$

Computing the Information Gain

- $Info(D) = 0.94$
- $Gain_{age} = 0.94 - 0.6935 = 0.2465$
- $Gain_{income} = 0.033$
- $Gain_{student} = 0.1525$
- $Gain_{credit} = 0.048$
- **Hence, we choose Age as the splitting criterion**

Splitting Criterion: Age

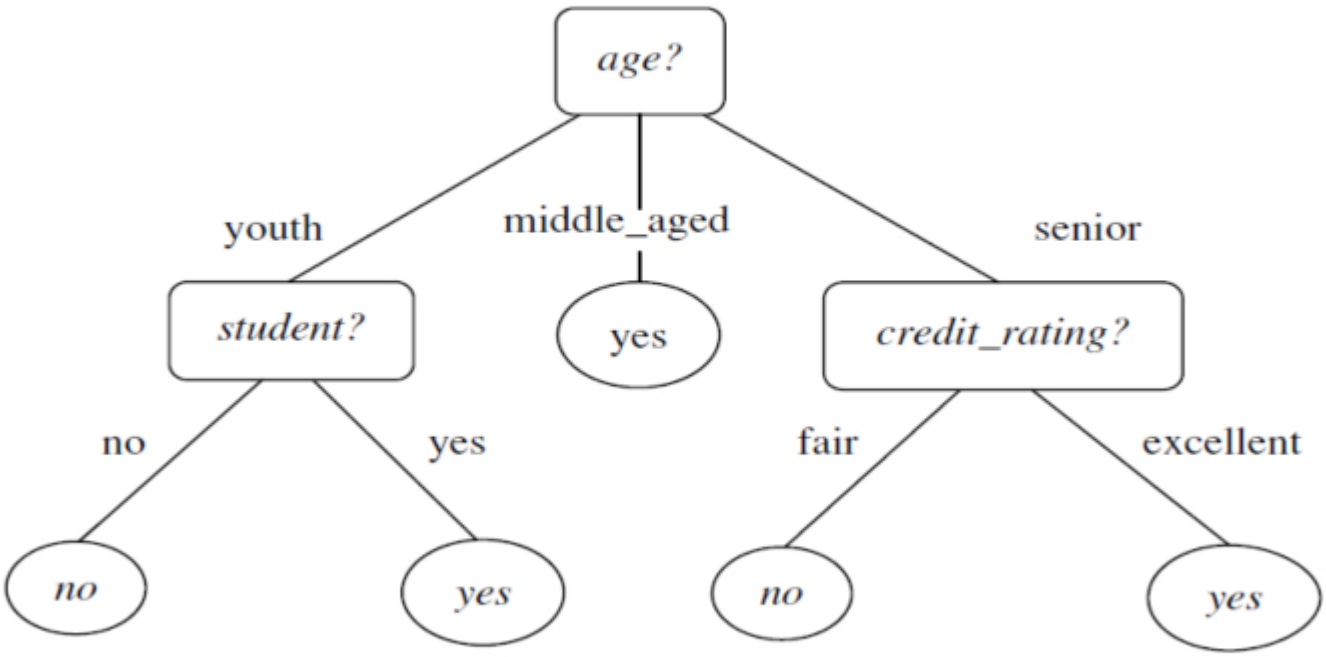


income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Final Decision Tree for AllElectronics



Drawbacks of the ID3 Algorithm

- Cannot handle numeric attributes or missing values.
- Prone to overfitting.
- Information Gain is biased towards criteria which have a large number of values.
- A modification of the ID3 algorithm called C4.5 uses a normalized value of information gain, called Gain Ratio to overcome this bias.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} x\ log_2\left(\frac{|D_j|}{|D|}\right)$$

and

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The CART Algorithm

- Classification And Regression Trees (CART) is a widely used decision tree algorithm.
- Exclusively produces binary trees.
- Uses a measure called Gini Index as the measure of purity.

$$Gini(D) = 1 - \sum_{i=1}^{|C|} p_i^2$$

$$Gini_{split}(D) = \frac{n_1}{n}\left(1 - \sum_{i=1}^{|C|} p_i^2\right) + \frac{n_2}{n}\left(1 - \sum_{i=1}^{|C|} p_i^2\right)$$

A Simple Example of CART Decision Tree Induction

Age	Salary	CLass
30	65	G
23	15	B
40	75	G
55	40	B
55	100	G
45	60	G

$$Gini(D) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.44$$

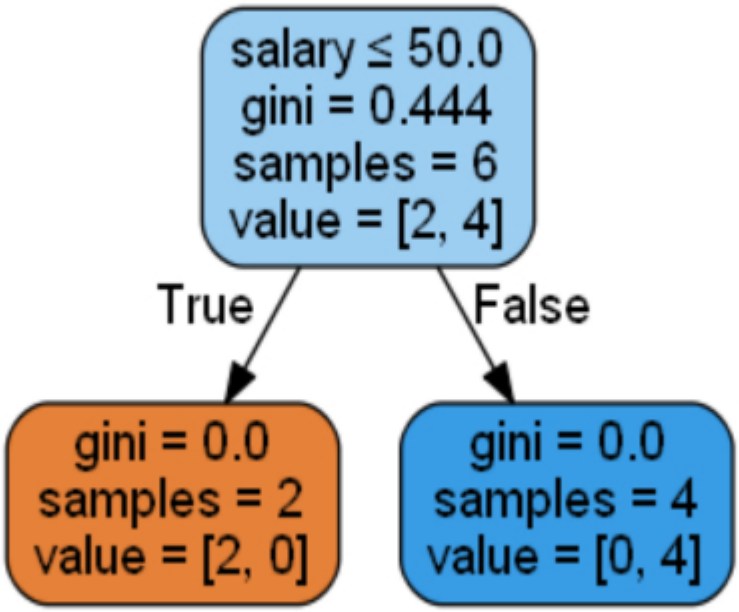
$$Gini_{age<23}(D) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.44$$

$$Gini_{age<26.5}(D) = \frac{1}{6}\left(\frac{1}{1}\right)^2 - \left(\frac{1}{1}\right)^2 + \frac{5}{6}\left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 0.267$$

$$Gini_{age<35}(D) = \frac{2}{6}\left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 + \frac{4}{6}\left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.4167$$

$$Gini_{age<42.5}(D) = 0.44 \quad Gini_{Salary<50} = 0$$
$$Gini_{age<50}(D) = 0.4167 \quad Gini_{Salary<62.5} = 0.22$$
$$Gini_{age<55}(D) = 0.44 \quad Gini_{Salary<70} = 0.33$$
$$Gini_{salary<15}(D) = 0.44 \quad Gini_{Salary<87.5} = 0.4$$
$$Gini_{salary<27.5}(D) = 0.267 \quad Gini_{Salary<100} = 0.44$$

A Simple Exaple of CART Decision Tree Induction

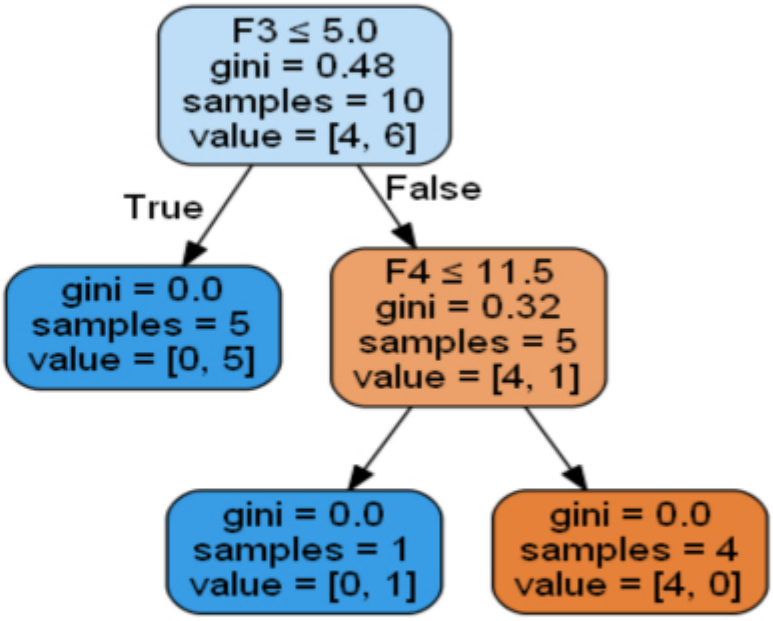


Overfitting in Decision Trees

- Overfitting refers to a phenomenon where the classification model performs exceptionally well on the training data but performs poorly on testing data.
- This happens because the classifier attempts to accommodate all the training samples into its model, even outliers or noisy samples, which makes the model complex and leads to poor generalization on a training set.
- **Pruning** can help keep the complexity of the tree in check, and reduce overfitting.
  - Pre-pruning : Pruning while the tree is being constructed by specifying criteria such as maximum depth, number of nodes etc.
  - Post-pruning : Pruning after the entire tree is constructed.

A Simple Example of Overfitting

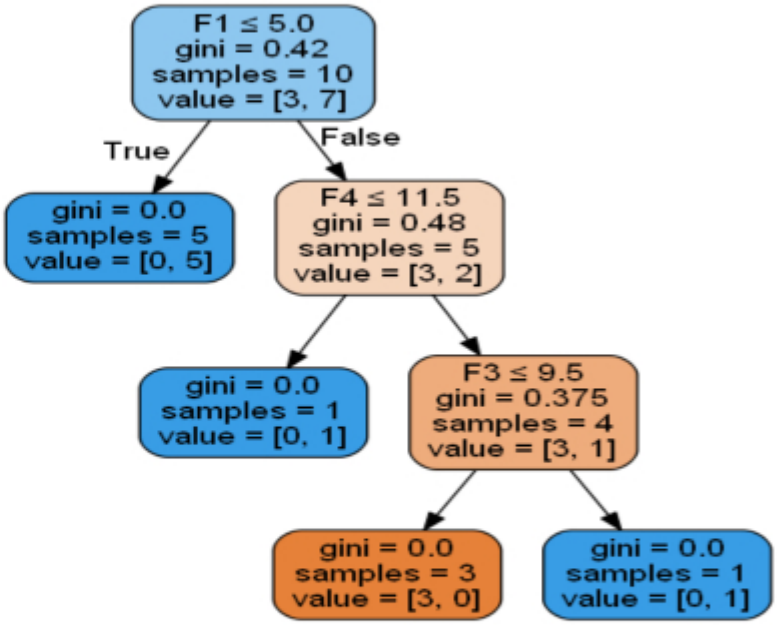
F1	F2	F3	F4	F5	T
7	18	7	11	22	B
1	5	1	18	36	B
0	15	0	2	4	B
7	5	7	12	24	A
1	15	1	12	24	A
3	20	3	6	12	B
0	5	0	18	36	B
7	10	7	12	24	A
10	8	10	20	40	A
9	20	9	17	34	A
4	14	4	9	18	B
1	19	1	9	18	B
10	19	10	18	36	A
10	14	10	7	14	B
9.75	9	9.75	16	32	A



Feature	Precision	Recall	F1	Support
A	1.00	1.00	1.00	2
B	1.00	1.00	1.00	3

A Simple Example of Overfitting

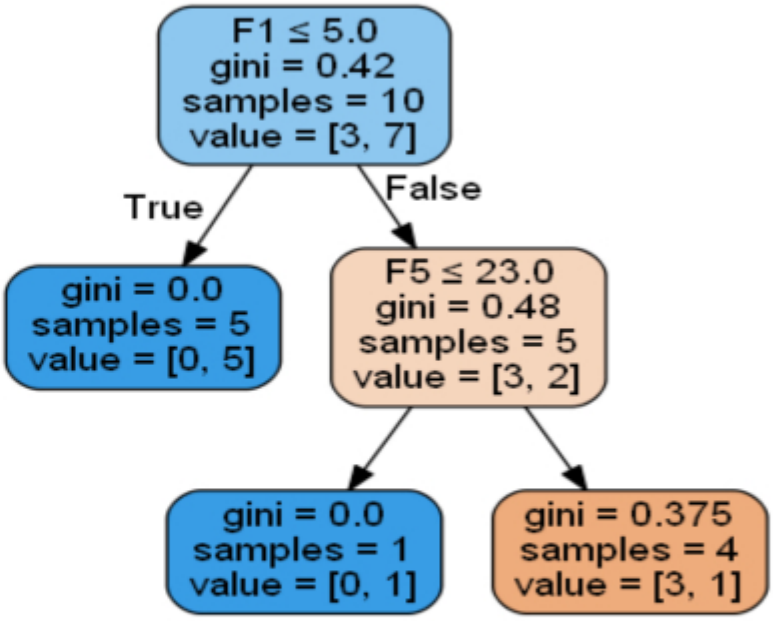
F1	F2	F3	F4	F5	T
7	18	7	11	22	B
1	5	1	18	36	B
0	15	0	2	4	B
7	5	7	12	24	B
1	15	1	12	24	A
3	20	3	6	12	B
0	5	0	18	36	B
7	10	7	12	24	A
10	8	10	20	40	B
9	20	9	17	34	A
4	14	4	9	18	B
1	19	1	9	18	B
10	19	10	18	36	A
10	14	10	7	14	B
9.75	9	9.75	16	32	A



Feature	Precision	Recall	F1	Support
A	0	0	0	2
B	0.6	1.0	0.75	3

Pre-Pruned Tree with Depth Limit

F1	F2	F3	F4	F5	T
7	18	7	11	22	B
1	5	1	18	36	B
0	15	0	2	4	B
7	5	7	12	24	B
1	15	1	12	24	A
3	20	3	6	12	B
0	5	0	18	36	B
7	10	7	12	24	A
10	8	10	20	40	B
9	20	9	17	34	A
4	14	4	9	18	B
1	19	1	9	18	B
10	19	10	18	36	A
10	14	10	7	14	B
9.75	9	9.75	16	32	A



Feature	Precision	Recall	F1	Support
A	1.0	1.0	1.0	2
B	1.0	1.0	1.0	3

Post-Pruning in Decision Trees

- Post-pruning first builds the decision tree, and then checks if any subtrees can be reduced to leaves.
- Reduced Error Pruning maintains a subset of the training data exclusively for pruning.
- Subtrees are pruned to create leaves based on the errors they produce on the pruning set.
- A slightly more complex variation of Reduced Error Pruning is called Cost Complexity Pruning.

Issues with Large Datasets

- The entire dataset needs to be present in the main memory for decision tree construction, which causes issues on large datasets.
  - Solution: Sample the dataset to construct a smaller approximation of the data for building the decision tree
  - Eg: CLOUDS, BOAT
- Finding the Ideal Splitting point requires sorting the data at every split, which leads to increased complexity on large datasets.
  - Solution: Presorting the dataset.
  - Eg: SLIQ, SPRINT, RAINFOREST

SLIQ - Decision Trees with Presorting

- SLIQ (Supervised Learning In Quest) imposes no restrictions on the size of the dataset for learning Decision Trees.
- It constructs a presorted list of attribute values to avoid sorting overheads during constructing the decision tree and maintains a record of where each record resides during the construction.

RID	Age	Salary	CLass
1	30	65	G
2	23	15	B
3	40	75	G
4	55	40	B
5	55	100	G
6	45	60	G

Attribute and Record Lists in SLIQ

Age	Class	RID
23	B	2
30	G	1
40	G	3
45	G	6
55	G	5
55	B	4

Salary	Class	RID
15	B	2
40	B	4
60	G	6
65	G	1
75	G	3
100	G	5

RID	Leaf
1	N1
2	N1
3	N1
4	N1
5	N1
6	N1

SLIQ - Decision Trees with Presorting

- The use of presorted attribute lists means that a single scan of the attribute list is sufficient to generate the adequate splitting point, compared to the multiple scans required in the CART algorithm.
- At each stage, the record list is updated to record where the records reside during the construction process.

CLOUDS - Decision Trees with Sampling

- While SLIQ works reasonably well, it still encounters certain issues while dealing with extremely large datasets, when the pre-sorted tables cannot reside in memory.
- A solution for this is to sample the dataset, and hence reduce the number of data items.
- Two points related to the Gini Index aid in this sampling:
  - Gini Index grows slowly, so there is a lesser chance of missing out on the splitting point.
  - The minimum value of the Gini index for the splitting point is lower than most of the other points.
- Classification for Large OUt of core DataSets (CLOUDS) provides a decision tree algorithm with sampling.

CLOUDS - Decision Trees with Sampling

- The CLOUDS algorithm seeks to eliminate the overhead of calculating the Gini Index at every splitting point, by narrowing down the number of splitting points.
- This is done by dividing the data into multiple quantiles, and calculating the Gini Index only at the boundaries.
- Now the search for the best splitting point is restricted to the quantile boundaries.
- This approach significantly reduces the complexity of the algorithm, but also introduces the possibility of errors. Errors can arise if the actual best splitting point resided **inside** one of the quantiles.
- To overcome this, each quantile computes a Gini Estimate within its boundaries. If the Gini Estimate is less than the Minimum Gini Value computed using the quantile boundaries, the quantile is discarded.
- If the Gini Estimate is less than the value calculated at the boundaries, then the quantile is inspected in detail to select the splitting point.

Advantages and Disadvantages of Decision Trees

- Advantages of Decision Trees
  - Invariant under scaling and various other transformations of feature values.
  - Robust to inclusion of irrelevant features i.e. has embedded feature selection.
  - Produces inspectable models which is simple to code and readily understandable by humans.
- Disadvantages of Decision Trees
  - Not always very accurate.
  - Prone to overfitting.

The "Wisdom of the Crowd" - Random Forests

- To improve the performance of decision trees, often a number of uncorrelated decision trees are used together. This is called the **Random Forest Classifier**.
- All the trees are individually used to classify an incoming data sample, and use an internal voting to decide on the final class value.
- This is an example of an **Ensemble Classifier** and relies on the wisdom of the crowd to select the best class.
- The key here is that the **trees in the forest should be uncorrelated**.

Why "Random" Forests?

- In order to avoid correlation between decision trees in the forest, two main modifications are done to the tree building process in a random forest classifier:
  - **Random selection of data subset**: For training the individual trees, a random subset of the data with replacement is used - a concept called as **Bagging** (Bootstrapped Aggregation).
  - **Random selection of feature subset for splitting**: At each node, only a subset of features are used to split the node. The value is usually set to the square root of the number of features.
- Random Forests help to reduce the effects of overfitting in trees without having to prune the trees.
- They provide a more reliable classifier than individual decision trees.

Rule based Classification

Dr. Amit Praseed

Rule based Classifier

- A **rule-based classifier** uses a set of IF-THEN rules for classification.
- Eg: IF *age* = *youth* AND *student* = *yes*
  THEN *buys computer* = *yes*.
- The IF part is called the antecedent
- The THEN part is called the consequent

Coverage and Accuracy

- The coverage of a rule is the number of tuples it triggers
- The accuracy of a rule is the fraction of its coverage where the predicted class label is also correct

Tid	Refund	Marital Status	Taxable Income	Class
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**

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

Conflict Resolution

- Rules used should ideally be:
  - Mutually exclusive
  - Complete
- In case a particular antecedent triggers more than one rule, we resort to conflict resolution strategies:
  - Size based ordering: Choose the rule which has the strictest antecedent
  - Rule based Ordering: Select the first rule which satisfies the antecedent
  - Voting: Select the consequent triggered by majority of the triggered rules
- If a particular antecedent does not trigger any rule, we resort to a default class label

Sequential Covering Algorithms

- Sequential covering algorithms generate a set of rules for classification directly from the dataset
  - An alternate approach is to generate a decision tree and then generate rules from the tree
- Sequential covering algorithms sequentially add more and more predicates to an existing rule as long as the error rate does not increase
- Once a rule is formed, the tuples satisfying that rule are removed from the database

Example

Object	Height	Hair	Eyes	Class
1	Short	Blonde	Blue	C1
2	Short	Blonde	Brown	C2
3	Tall	Red	Blue	C1
4	Tall	Dark	Blue	C2
5	Tall	Dark	Blue	C2
6	Tall	Blonde	Blue	C1
7	Tall	Dark	Brown	C2
8	Short	Blonde	brown	C2

Find the first rule for C1

- We have to find a rule of the form ____ → C1.
- What are the antecedents we can have?

Find the first rule for C1

- We have to find a rule of the form ____ → C1.
- What are the antecedents we can have?

α_x	C1	S	$P(C1 \mid \alpha_x)$
Height=short	1	3	1/3 = 0.33
Height=tall	2	5	2/5 =0.4
Hair=blonde	2	4	2/4 = 0.5
Hair=red	1	1	1
Hair=dark	0	3	0
Eyes=blue	3	5	3/5 = 0.6
Eyes=brown	0	3	0

Find the first rule for C1

- Create subset $S_{hair=red}$. It contains only one element, so output the rule
  - (Hair=red) →C1

α_x	C1	S	P(C1|α_x)
Height=short	1	3	1/3 = 0.33
Height=tall	2	5	2/5 =0.4
Hair=blonde	2	4	2/4 = 0.5
Hair=red	1	1	1
Hair=dark	0	3	0
Eyes=blue	3	5	3/5 = 0.6
Eyes=brown	0	3	0

Remove Learned Tuple

Object	Height	Hair	Eyes	Class
1	Short	Blonde	Blue	C1
2	Short	Blonde	Brown	C2
~~3~~	~~Tall~~	~~Red~~	~~Blue~~	~~C1~~
4	Tall	Dark	Blue	C2
5	Tall	Dark	Blue	C2
6	Tall	Blonde	Blue	C1
7	Tall	Dark	Brown	C2
8	Short	Blonde	brown	C2

Recalculate probability values

- We now try to find rule R2

α_x	C1	S	$P(C1 \mid \alpha_x)$
Height=short	1	3	1/3 = 0.33
Height=tall	1	4	1/4 =0.25
Hair=blonde	2	4	2/4 = 0.5
Hair=dark	0	3	0
Eyes=blue	2	4	2/4 = 0.5
Eyes=brown	0	3	0

Select hair=blonde as the first term

Construct $S_{hair=blonde}$. Here two items belong to class C2, so we need to refine the rule further. So we try to add a new term to the antecedent

Object	**Height**	**Hair**	**Eyes**	**Class**
1	Short	Blonde	Blue	C1
2	Short	Blonde	Brown	C2
6	Tall	Blonde	Blue	C1
8	Short	Blonde	brown	C2

Find the next term

Select eyes=blue as the second term as it applies to more rows

α_x	C1	S	$P(C1 \mid \alpha_x)$
Height=short	1	3	1/3 = 0.33
Height=tall	1	1	1/1 = 1
Eyes=blue	2	2	2/2 = 1
Eyes=brown	0	2	0

Output the second rule

- The second rule becomes
  - R2: (Hair=blonde ^ eyes=blue) →C1
- Because the tuples selected by this antecedent both belong to class C1
- Remove the selected tuples
- All instances of C1 are done.
- Repeat for C2

Object	Height	Hair	Eyes	Class
2	Short	Blonde	Brown	C2
4	Tall	Dark	Blue	C2
5	Tall	Dark	Blue	C2
7	Tall	Dark	Brown	C2
8	Short	Blonde	brown	C2

Final Set of Rules

Sequential Cover

- (Hair=Red) → C1
- (Hair=blonde ^ Eyes=blue) →C1
- (Eyes=brown)→C2
- (Hair=dark)→C2

ID3 Decision Tree

- (Hair=Red) → C1
- (Hair=blonde ^ Eyes=blue) →C1
- (Hair=blonde ^ Eyes=brown) →C2
- (Hair=dark)→C2

Question – Select the first antecedent

Object	Height	Class
1	Tall	C1
2	Short	C1
3	Tall	C1
4	Tall	C2
5	Tall	C1
6	Tall	C1
7	Tall	C1
8	Tall	C1

Find the first rule for C1

- Normally the antecedent height=short will be selected because it has 100% accuracy while height=tall has only 85% accuracy
- But height=tall has better coverage than height=short
- Instead of considering the accuracy, we can consider a combination of accuracy and coverage

FOIL Gain

$$FOIL - Gain = pos' * \left(log \frac{pos'}{pos' + neg'} - log \frac{pos}{pos + neg} \right)$$

- pos/neg is the number of positive or negative instances covered by R
- pos'/neg' is the number of positive or negative instances covered by R'
- It favours rules having high accuracy and covers many positive instances
- Used in FOIL and RIPPER algorithms

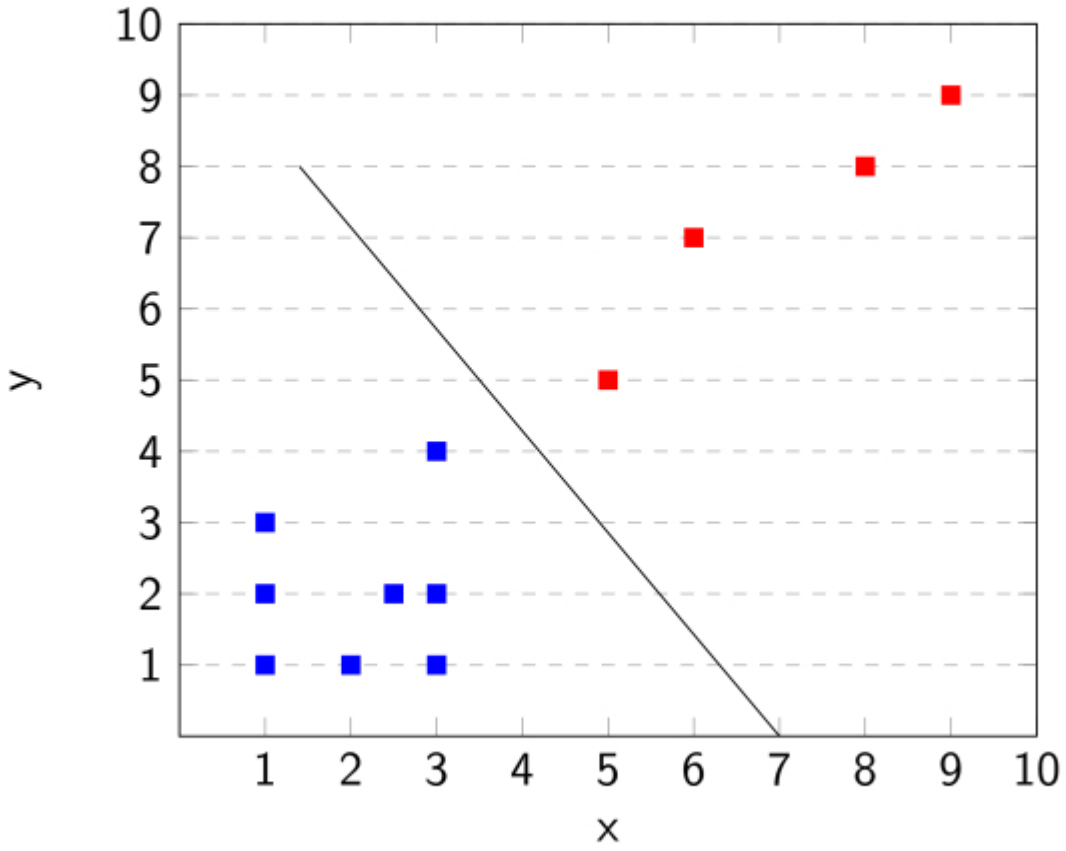
Support Vector Machines

October 11, 2022

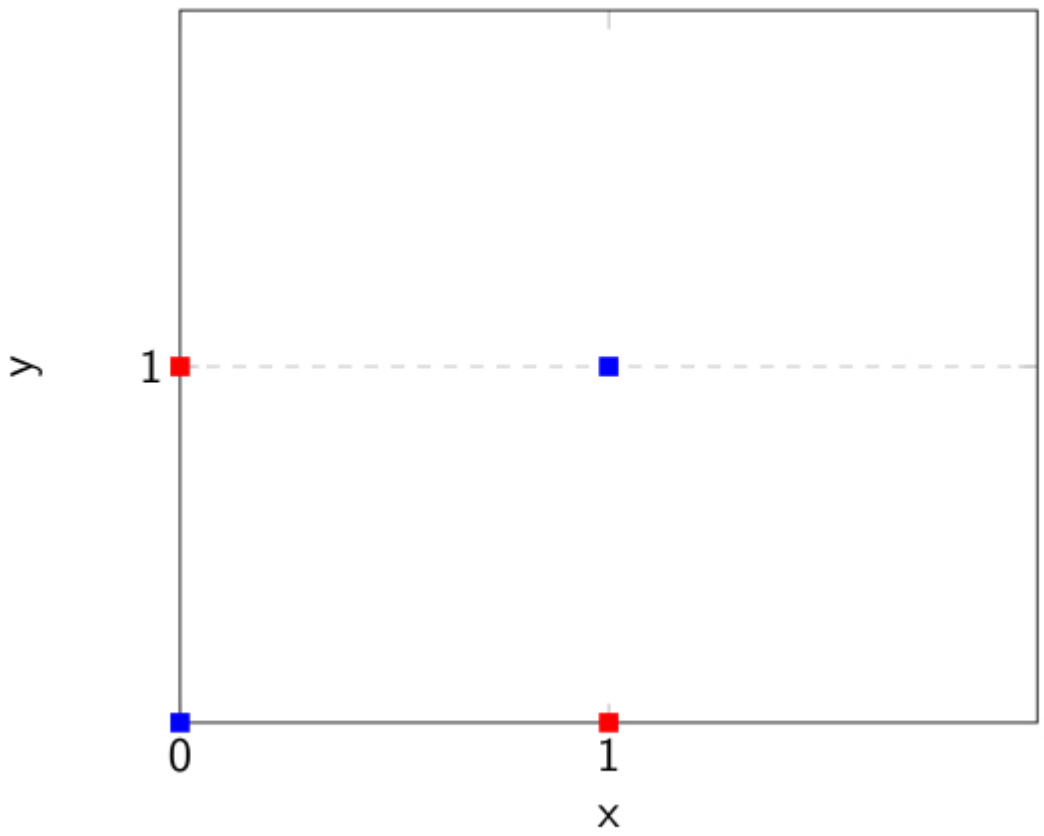
Introduction

- Support Vector Machines (SVMs) are arguably the most famous machine learning tool.
- They are extensively used in text and image classification.

Linear Separability

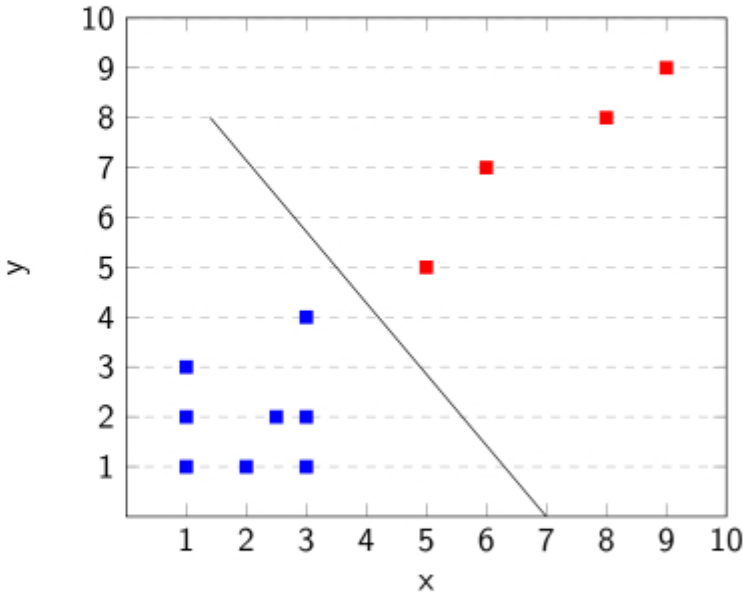


Linear Inseparability (The XOR Problem)

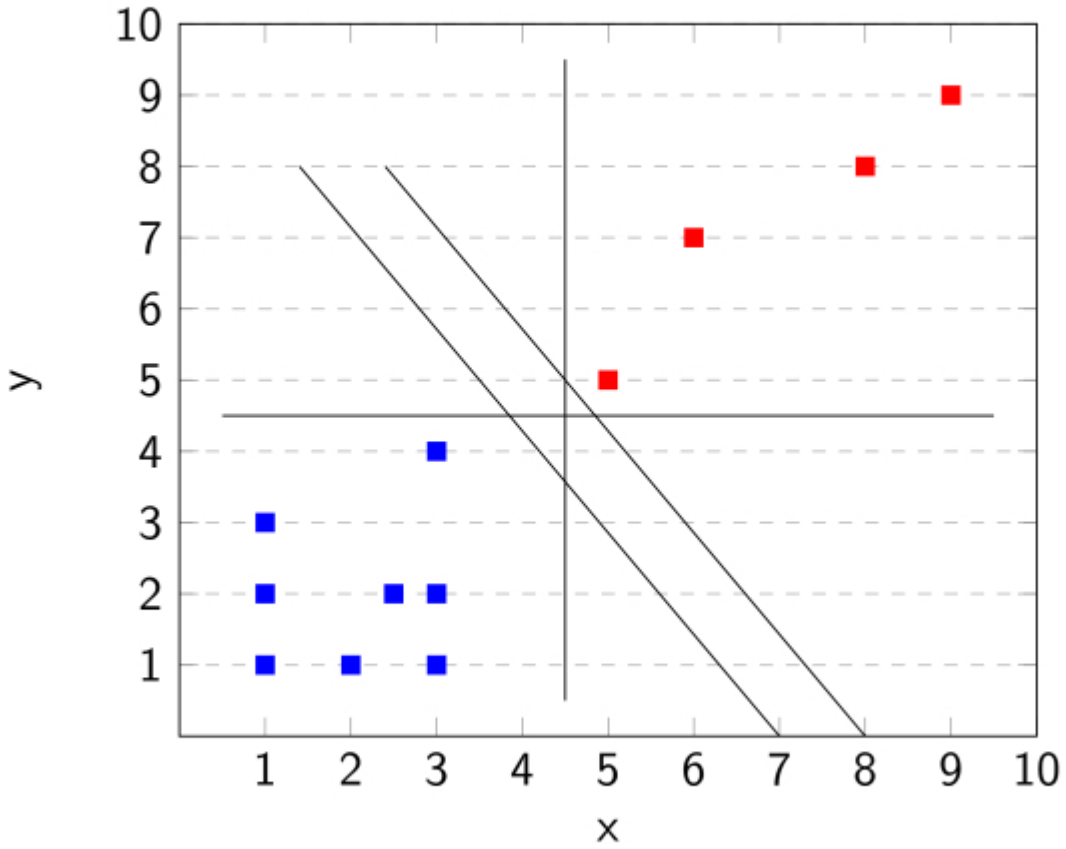


SVM for Linearly Separable Problems

- For now, let us consider only linearly separable cases.
- The only thing SVM does is find the hyperplane (in n-dimensions; in two dimensions it simply finds the line) that separates the two classes.



Which is the Optimal Hyperplane?



The Optimal Hyperplane

- The optimal Hyperplane should be maximally separated from the closest data points ("maximize the margin").
- The optimal hyperplane should completely separate the data points into two perfectly pure classes.

SVM in 2 Dimensions

- In two dimensions, the problem reduces to finding the optimal line separating the two data classes. The conditions for the optimal hyperplane still hold in two dimensions.
- Let the line we have to find be $y = ax + b$.

$$y = ax + b$$

$$ax - y + b = 0$$

- Let $\vec{X} = [x \quad y]$ and $\vec{W} = [a \quad -1]$, then the equation for the line becomes

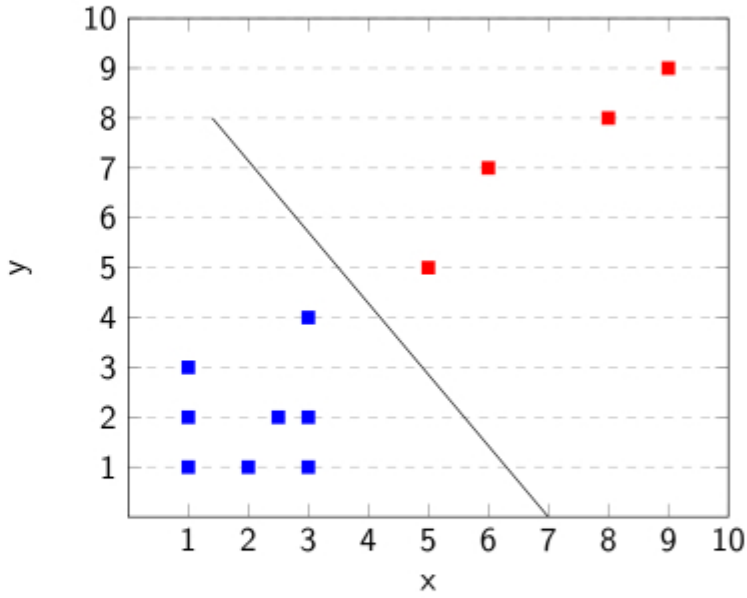
$$\vec{W}.\vec{X} + b = 0$$

which incidentally is the equation for a hyperplane as well, when generalized to n dimensions. Here $\vec{W}$ is a vector perpendicular to the plane.

SVM in 2 Dimensions

- Let us denote the two classes by integer numbers. The SVM will output -1 for all the blue data points and $+1$ for the red data points.
- It can also be noted that for all blue points which are on the left of the separating line, $\vec{W}.\vec{X} + b \leq 0$. For all the red points, $\vec{W}.\vec{X} + b \geq 0$.
- Denote

$$h(x_i) = \begin{cases} +1, \vec{W}.\vec{X} + b \geq 0 \\ -1, \vec{W}.\vec{X} + b \leq 0 \end{cases}$$



SVM in 2 Dimensions

- The sign of $\vec{W}.\vec{X} + b$ gives the class label.
- The magnitude of $\vec{W}.\vec{X} + b$ gives the distance of the data point from the separating line.
- Let β_i denote the distance from a data point to the separating hyperplane P. Then the closest point to the plane P will have the lowest value of β. Let us denote this value as the functional margin F.
- However, the distance measures on the blue side will be negative while on the red side they will be positive!!! To avoid the difficulty in comparing, we can simply multiply with the class labels, because all blue points have class labels as -1 and all red points have class labels $+1$.

$$F = min\ \beta_i$$

$$F = min\ y_i(\vec{W}.\vec{X_i} + b)$$

SVM in 2 Dimensions

- From a potentially infinite number of hyperplanes, an SVM selects the one which maximizes the margin, or in other words which maximally separates the two closest data points to it.
- Put in a mathematical sense, SVM selects the hyperplane which maximizes the functional margin.
- However, the functional margin is not scale invariant.
- For instance consider two planes given by $\vec{W_1} = (2, 1), b_1 = 5$ and $\vec{W_2} = (20, 10), b_1 = 50$. Essentially they represent the same plane because $\vec{W}$ is a vector orthogonal to the plane P, and the only thing that matters is the direction of $\vec{W}$. However, the functional margin still multiplies the distance of a data point from P by a factor of 10, which renders it ineffective.
- We compute a scale invariant distance measure, given by

$$\gamma_i = y_i(\frac{\vec{W}}{|\vec{W}|}.\vec{X_i} + \frac{b}{|\vec{W}|})$$

SVM in 2 Dimensions

- We define a scale invariant geometric margin as

$$M = min\ \gamma_i$$

$$M = min\ y_i(\frac{\vec{W}}{|\vec{W}|}.\vec{X_i} + \frac{b}{|\vec{W}|})$$

- The SVM problem reduces to finding a hyperplane with the maximum geometric margin, or in other words,

$$max\ M$$

$$subject\ to\ \gamma_i \geq M, i = 1, 2..\ m$$

SVM in 2 Dimensions

- Now, we can rewrite the above problem by recalling that $M = \frac{F}{|\vec{W}|}$.

$$max \ \frac{F}{|\vec{W}|}$$

$$subject \ to \ \frac{f_i}{|\vec{W}|} \geq \frac{F}{|\vec{W}|}, i = 1, 2.. \ m$$

- Since we are maximizing the geometric margin, which is scale invariant, we can set $\vec{W}$ and b to make $F = 1$.

$$max \ \frac{1}{|\vec{W}|}$$

$$subject \ to \ f_i \geq 1, i = 1, 2.. \ m$$

SVM in 2 Dimensions

- We can express the same as a minimization problem

$$min\ |\vec{W}|$$

$$subject\ to\ y_i(\vec{W}.\vec{X_i} + b) \geq 1, i = 1, 2..\ m$$

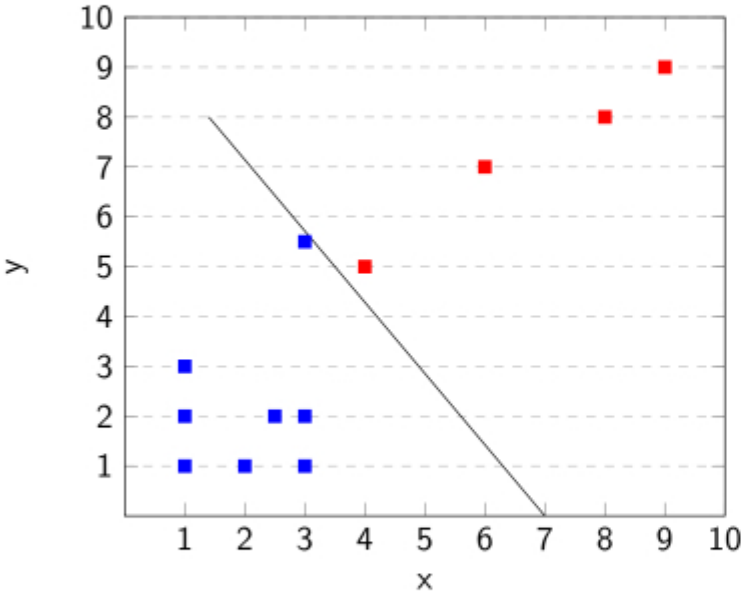
- The final optimization problem seen in literature is a simple modification of this

$$min\ \frac{1}{2}|\vec{W}|^2$$

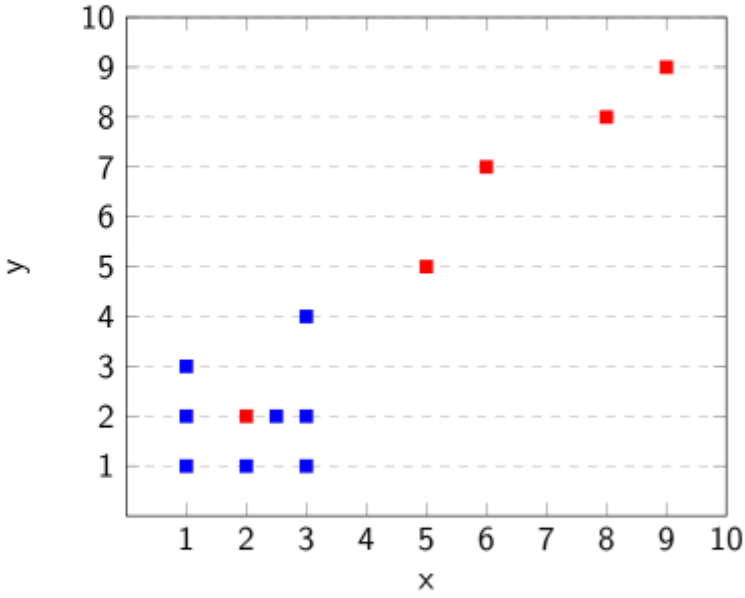
$$subject\ to\ y_i(\vec{W}.\vec{X_i} + b) - 1 \geq 0, i = 1, 2..\ m$$

- This problem resembles the Convex Quadratic Optimization Problem which can be solved using Lagrange multipliers.

Issues with Hard Margin SVMs



Very Small Margin due to Outliers



Linearly Inseparable due to Outliers

Soft Margin SVMs

- Hard Margin SVMs have a strict rule that all of the training dataset must be classified properly.
- Soft Margin SVMs allow some data points in the training data set to be misclassified if it allows for a better margin and a better decision boundary.
- They introduce a **slack variable** to allow for this, so that the constraint becomes

$$y_i(\vec{W}.\vec{X_i} + b) \geq 1 - \zeta_i, i = 1, 2.. \ m$$

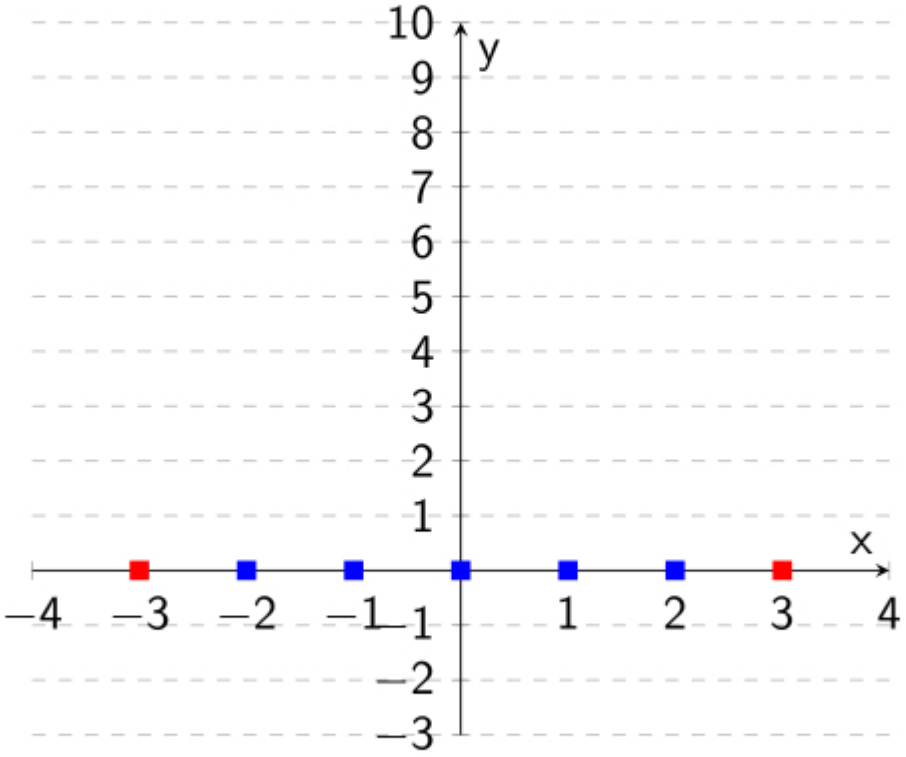
- The complete minimization problem for a soft margin SVM becomes

$$min \ \frac{1}{2}|\vec{W}|^2 + C\sum_{i=1}^{m}\zeta_i$$

$$subject \ to \ y_i(\vec{W}.\vec{X_i} + b) \geq 1 - \zeta_i, i = 1, 2.. \ m$$

- The additional term of $C\sum_{i=1}^{m}\zeta_i$ in the minimization function is meant to penalize the choice of a large ζ value which could lead to all constraints being satisfied, but would lead to more misclassification error.

A Look Back at Linear Separability

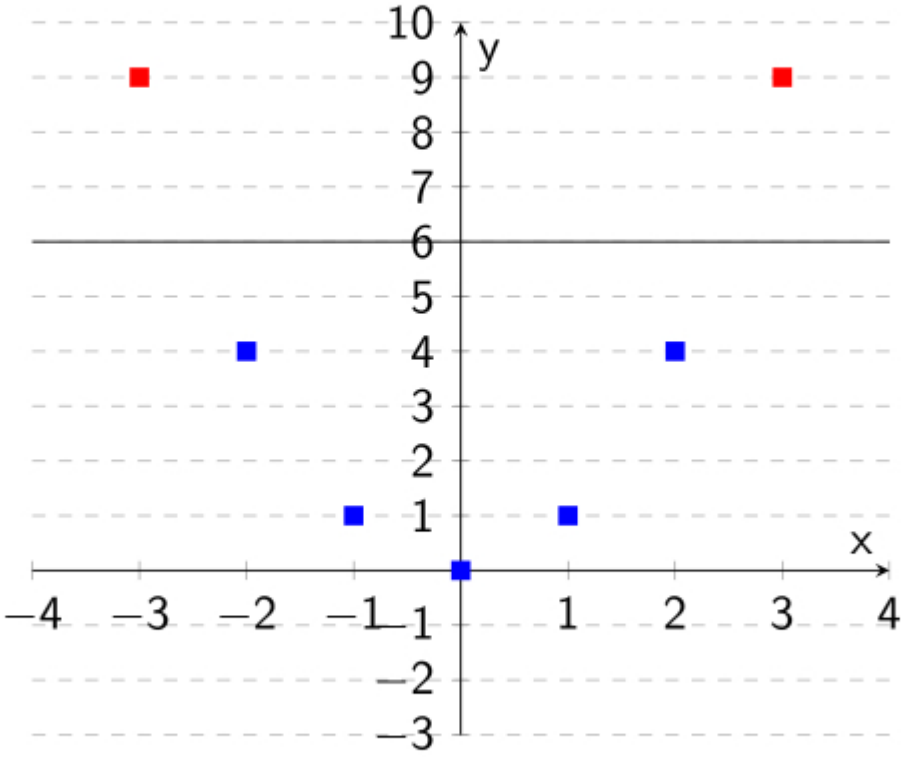


The Kernel Trick

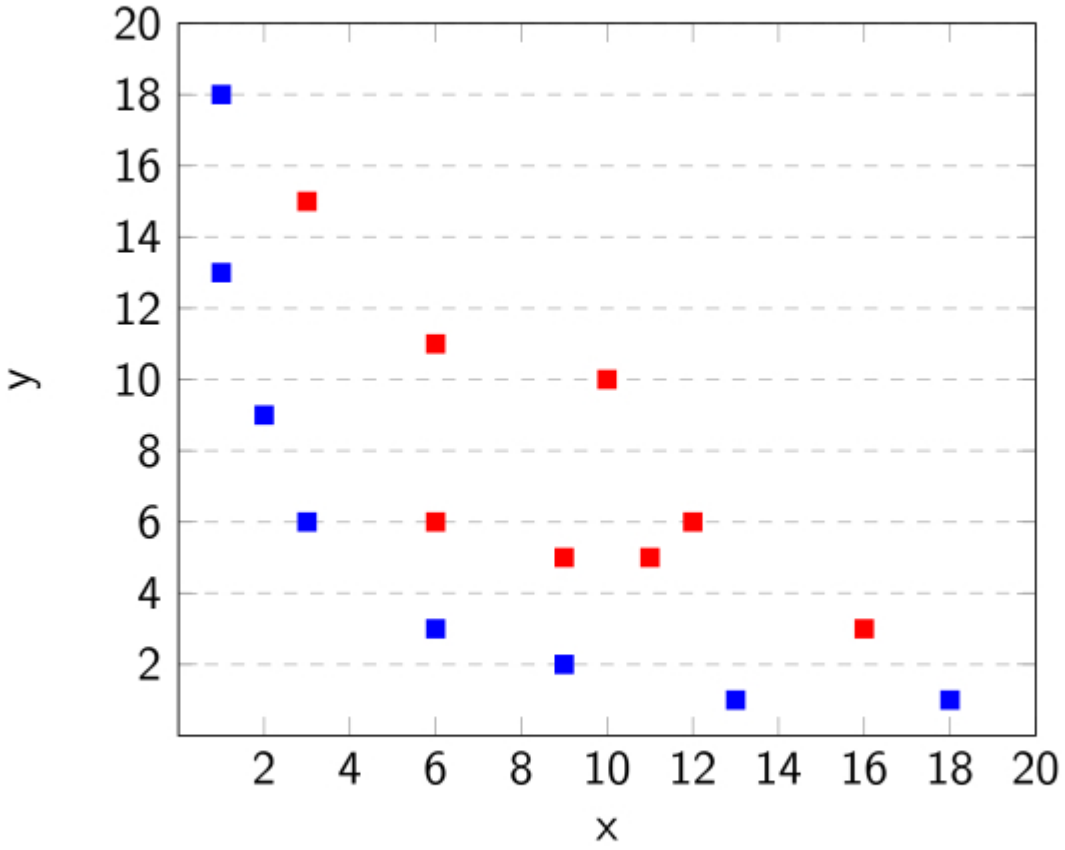
- A data set that appears to be linearly inseparable in a lower dimensional space can be made linearly separable by projecting onto a higher dimensional space.
- Instead of transforming all data points into the new higher dimensional space, SVM uses a kernel that computes the distances between the data points as if they belong to a higher dimensional space. This is often called the **Kernel Trick**.
- For the earlier dataset, let us apply a mapping $\phi : \mathbb{R} \to \mathbb{R}^2$

$$\phi(x, y) = (x, x^2)$$

Linearly Separable in 2D

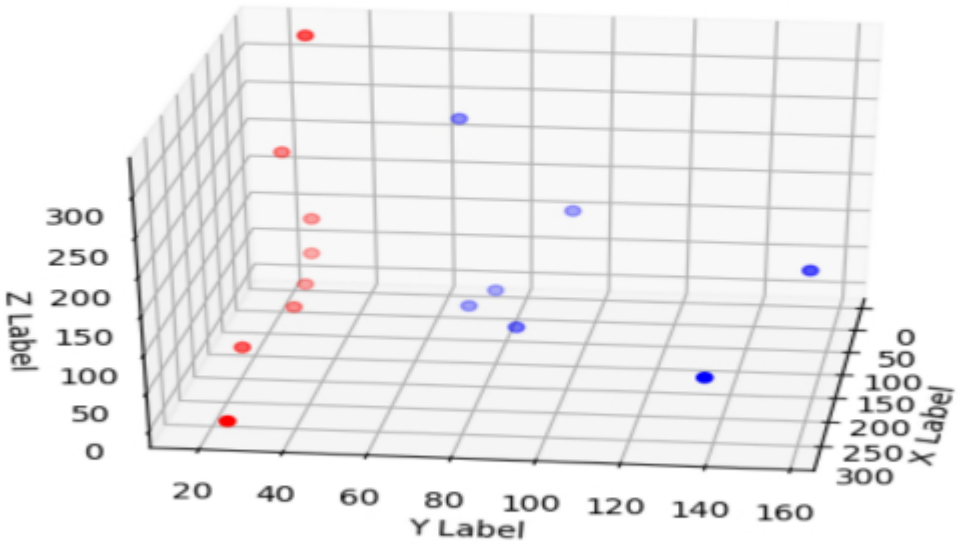


Another Example



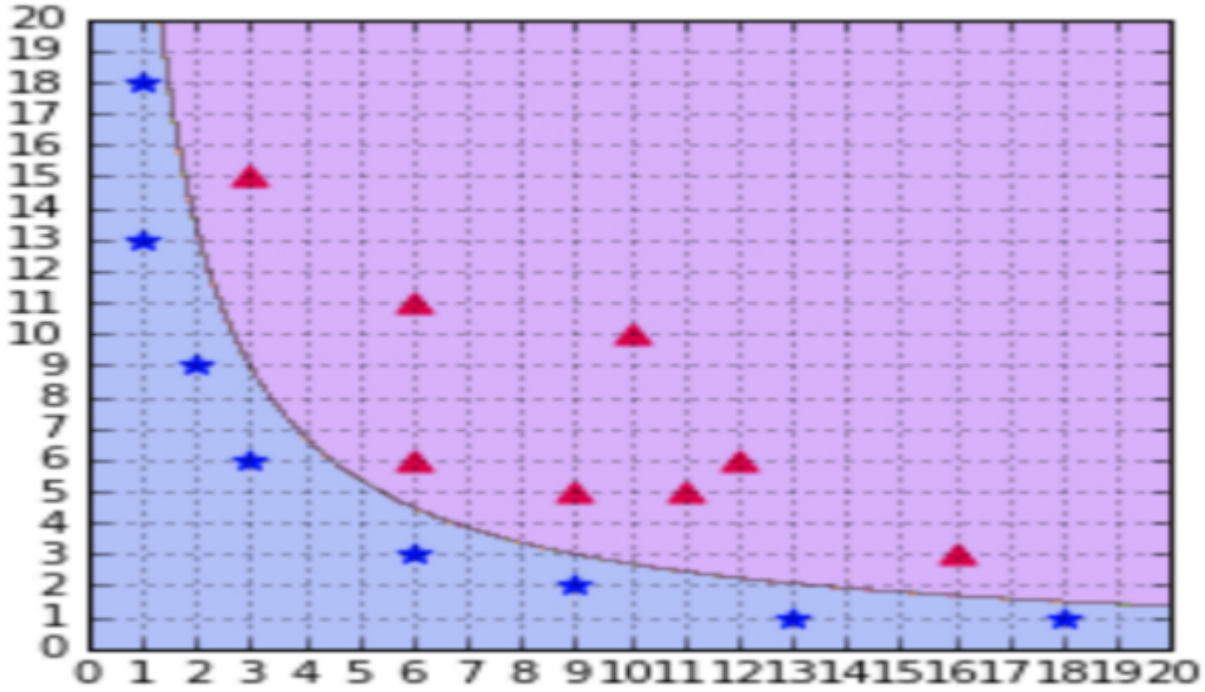
$$\phi(x, y) = (x^2, \sqrt{2}xy, y^2)$$

The Kernel Trick



The data when projected in 3 dimensions is linearly separable

The Kernel Trick



Types of Kernels

- Linear Kernel
- Polynomial Kernel
- RBF or Gaussian Kernel : An RBF Kernel theoretically transforms data into an infinite dimensional space for classification. It is usually recommended to try classification using an RBF kernel because it usually gives good results.

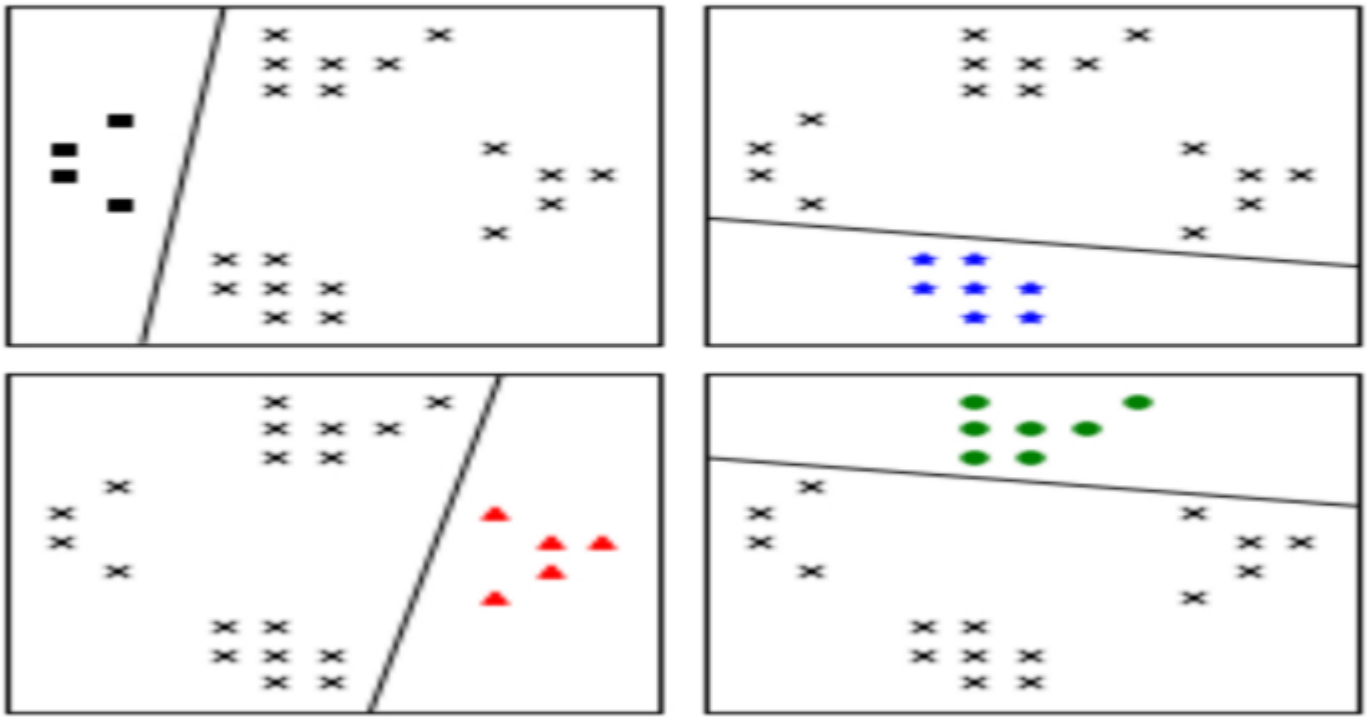
Multi-class Classification using SVM

- SVMs are primarily built for binary classification.
- However, two approaches can be used to extend an SVM to classify multiple classes.
  - One against All
  - One against One

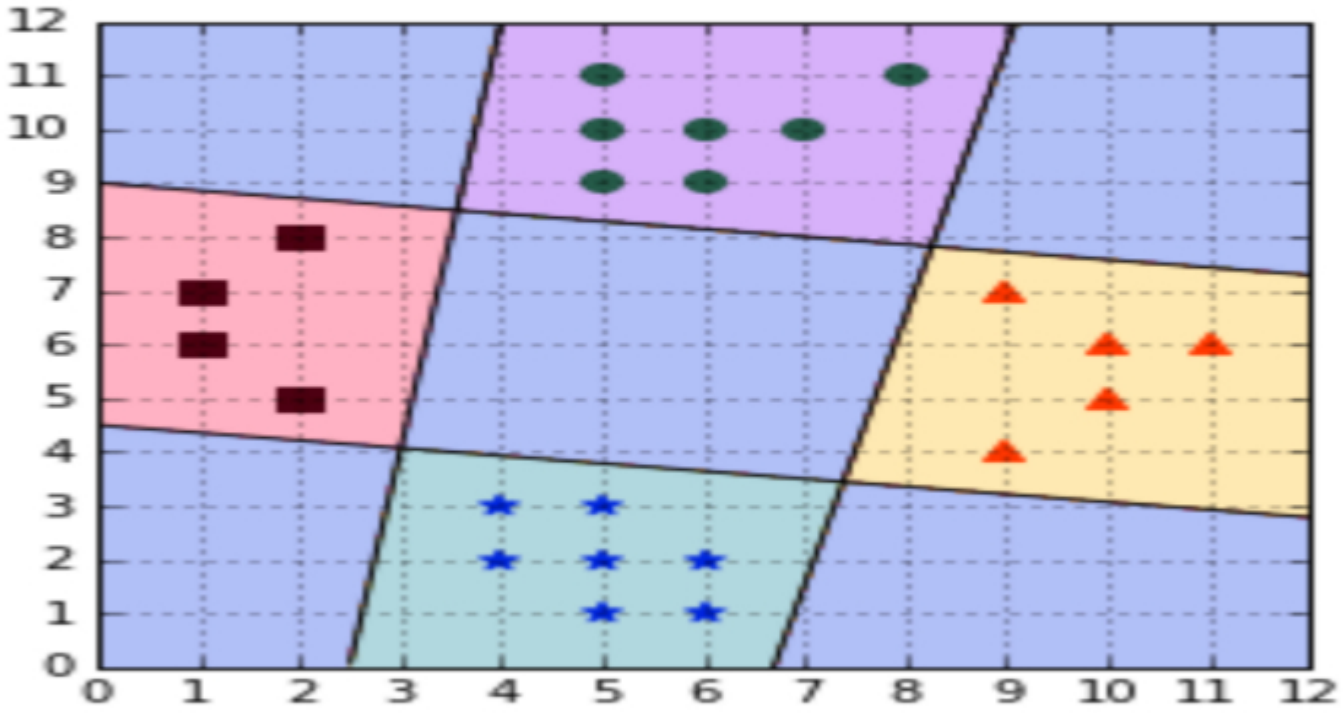
One against All SVM

- In One against All SVM, an n class classification problem is decomposed into n binary classification problems.
- For example, a problem involving three classes A,B and C is decomposed into three binary classifiers:
  - A binary classifier to recognize A
  - A binary classifier to recognize B
  - A binary classifier to recognize C
- However, this approach can lead to inconsistencies.

One against All SVM



One against All leads to Ambiguity

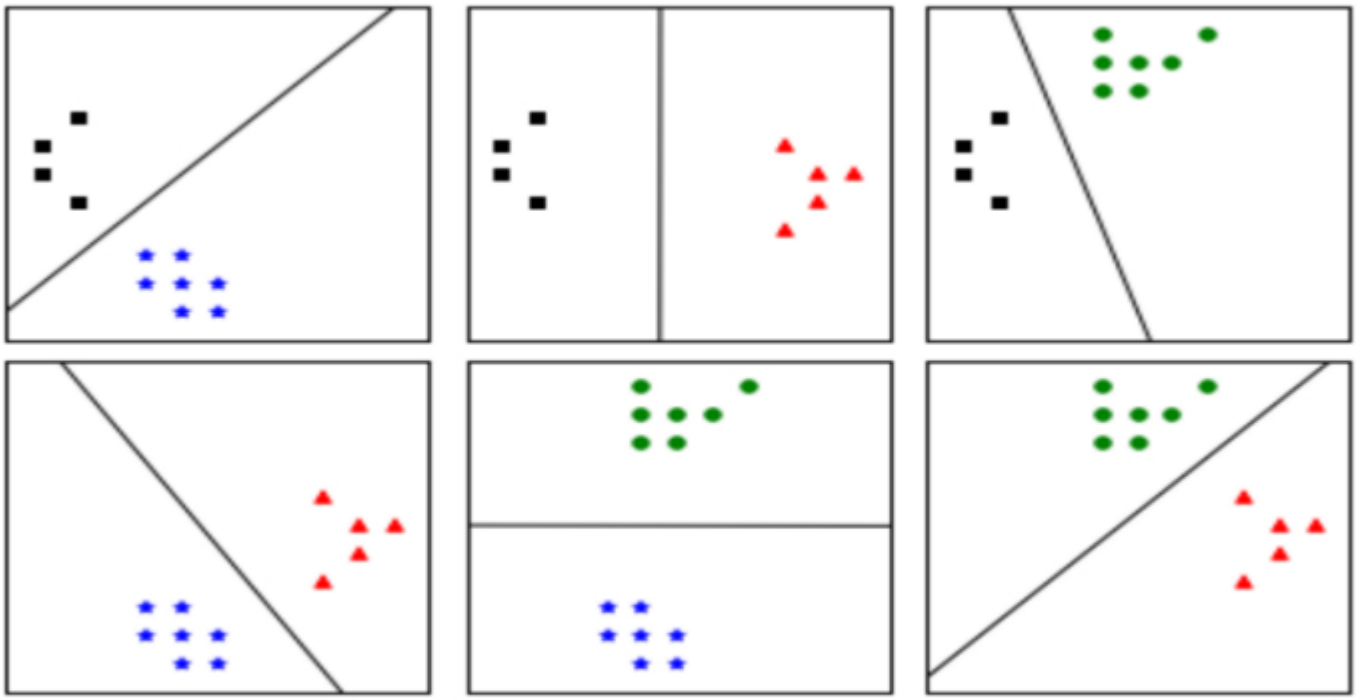


Data in the blue regions will be classified into two classes simultaneously

One against One SVM

- In One against One SVM, instead of pitting one class against all the others, classes are pitted against each other.
- This leads to a total of $\frac{k(k-1)}{2}$ classifiers for a k-class classification problem.
- The class of the data point is decided by a simple majority voting.

One against One SVM



One against one still leads to ambiguity if two classes get the same amount of votes.

Anomaly Detection

- The fundamental notion of classification is that there are two or more classes that are well represented in the training data, and the classifier must assign one of the data labels to the incoming testing data.
- However in a number of scenarios, only one class of data is available. The other class is either not available at all, or has very few data samples.
- Eg: Security related domains, for example detecting cyber-attacks, or detecting abnormal program execution.
- In such scenarios, obtaining the training data corresponding to attacks or abnormal runs is expensive or infeasible.
- The commonly used approach is to train the classifier on the available data which models the **normal** data, and use the classifier to identify any **anomalies** in the testing data.
- This is often called anomaly detection.

One Class SVMs

- SVMs can be used for anomaly detection by modifying the constraints and the objective function.
- Instead of maximizing the margin between classes as in a normal SVM, the One Class SVM tries to form a hypersphere (with radius R and centre a) around the training data, and tries to minimize the radius of the sphere.

$$min\ R^2 + C\sum_{i=1}^{N}\zeta_i$$

$$subject\ to$$

$$||x_i - a||^2 \leq R^2 + \zeta_i$$

$$\zeta_i \geq 0$$