UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

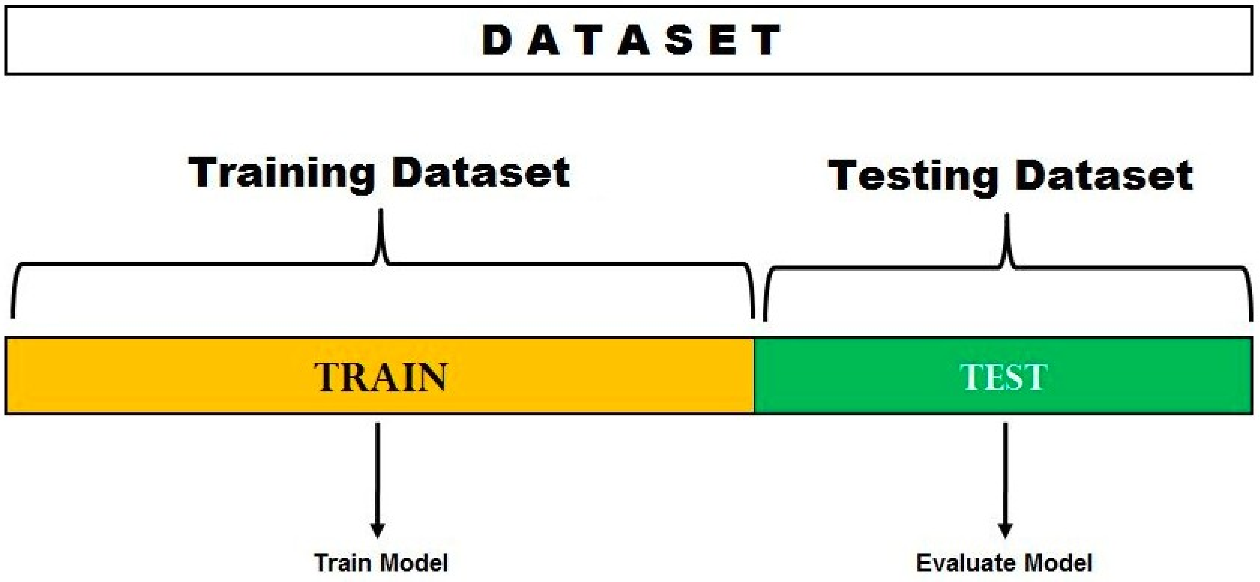
Course Name : AI & ML

Training a Model (For Supervised Learning)

- Holdout Method
- The hold-out method for training a machine learning model is the process of splitting the data in different splits and using one split for training the model and other splits for validating and testing the models. The hold-out method is used for both **model evaluation** and **model selection.**

Process of using the hold-out method for model evaluation:

- Split the dataset into two parts (preferably based on 70-30% split; However, the percentage split will vary)
- Train the model on the training dataset; While training the model, some fixed set of hyper parameters is selected.
- Test or evaluate the model on the held-out test dataset
- Train the final model on the entire dataset to get a model which can generalize better on the unseen or future dataset.


DATASET
Training Dataset
Testing Dataset
TRAIN
TEST
Train Model
Evaluate Model

Hold-out method for Model Selection

•The hold-out method can also be used for model selection or hyperparameters tuning. As a matter of fact, at times, the model selection process is referred to as hyper-parameters tuning. In the hold-out method for model selection, the dataset is split into three different sets – training, validation,and test dataset.

Process represents the hold-out method for model selection:

•Split the dataset in three parts – Training dataset, validation dataset and test dataset.

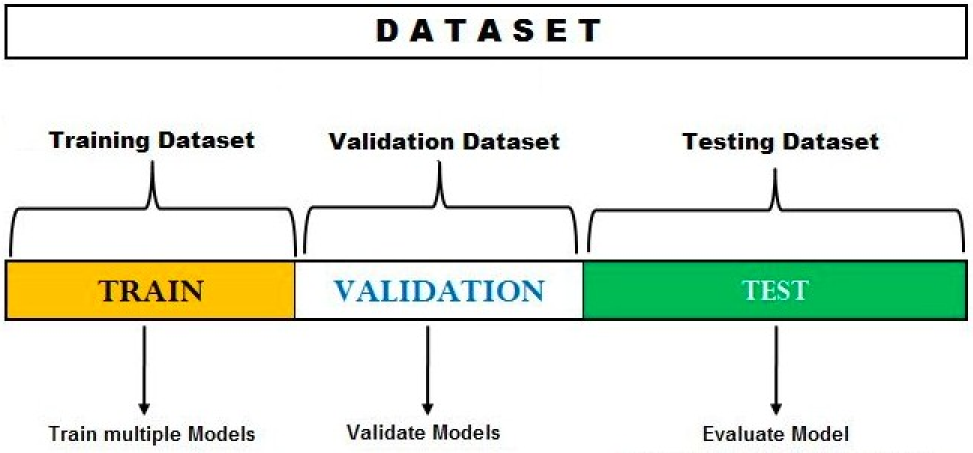
•Train different models using different machine learning algorithms. For example, train the classification model using logistic regression, random forest, XGBoost.

•For the models trained with different algorithms, tune the hyper-parameters and come up with different models. For each of the algorithms mentioned in step 2, change hyper parameters settings and come with multiple models.

•Test the performance of each of these models (belonging to each of the algorithms) on the validation dataset.

•Select the most optimal model out of models tested on the validation dataset. The most optimal model will have the most optimal hyper parameters settings for specific algorithm. Going by the above example, lets say the model trained with XGBoost with most optimal hyper parameters gets selected.

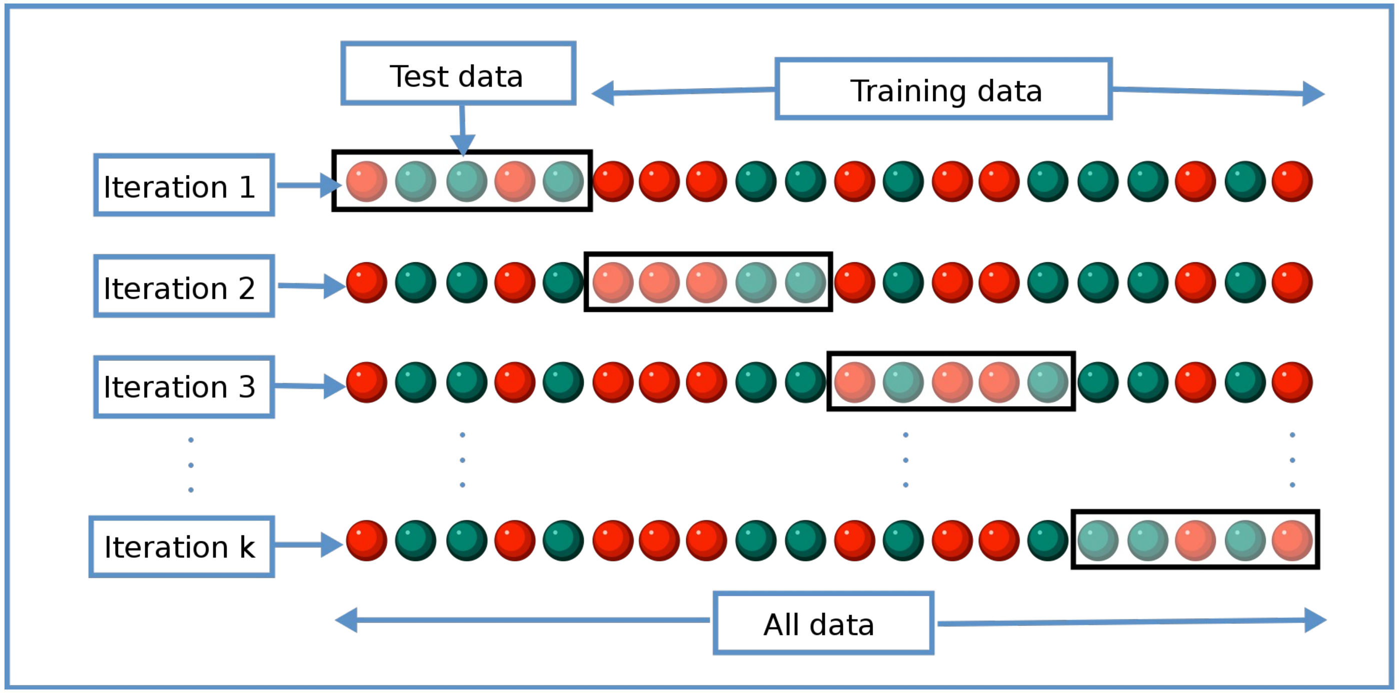
•Test the performance of the most optimal model on the test dataset.


DATASET
Training Dataset
Validation Dataset
Testing Dataset
TRAIN
VALIDATION
TEST
Train multiple Models
Validate Models
Evaluate Model

k-Fold Cross-Validation

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
- The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

- The general procedure is as follows:
- Shuffle the dataset randomly.
- Split the dataset into k groups
- For each unique group:
  - Take the group as a hold out or test data set
  - Take the remaining groups as a training data set
  - Fit a model on the training set and evaluate it on the test set
  - Retain the evaluation score and discard the model
- Summarize the skill of the model using the sample of model evaluation scores
- Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.


Test data
Training data
Iteration 1
Iteration 2
Iteration 3
Iteration k
All data

What is a Feature Variable in Machine Learning?

- A feature is a measurable property of the object you're trying to analyze. In datasets, features appear as columns.

What is feature Engineering?

- Process of translating a data set into features such that these features are able to represent the data set more effectively and result in a better learning performance.

Elements of Feature Engineering

- i) feature transformation
- ii) feature subset selection

Feature Transformation

Transforms the data into a set of features which can represent underlying problem into a ML problem.

Two types of feature transformation :

i) Feature construction: Discovers missing information about the relationships between features and create additional features.

ii) Feature Extraction: Process of extracting or creating new features from the original set of features.

When feature construction is an essential?

- When features are categorical but input requires numerical value
- When features are numeric but continuous value but input requires ordinal value
- When text specific feature construction is required

Feature Extraction

New features are created from a combination of original features. Some commonly used operators are:

For Boolean features: AND , OR, NOT

For nominal features: Cartesian product

For numerical features: Min, Max and other arithmetic operations.

Most popular feature extraction algorithms are:

Principal Component Analysis (PCA)

Singular Value Decomposition (SVD)

Linear Discriminant Analysis (LDA)

Effect of Feature

- Feature contains information about the target
- Classification model is a function of features
- Naïve view: More features

=> More information

=> More discrimination power.

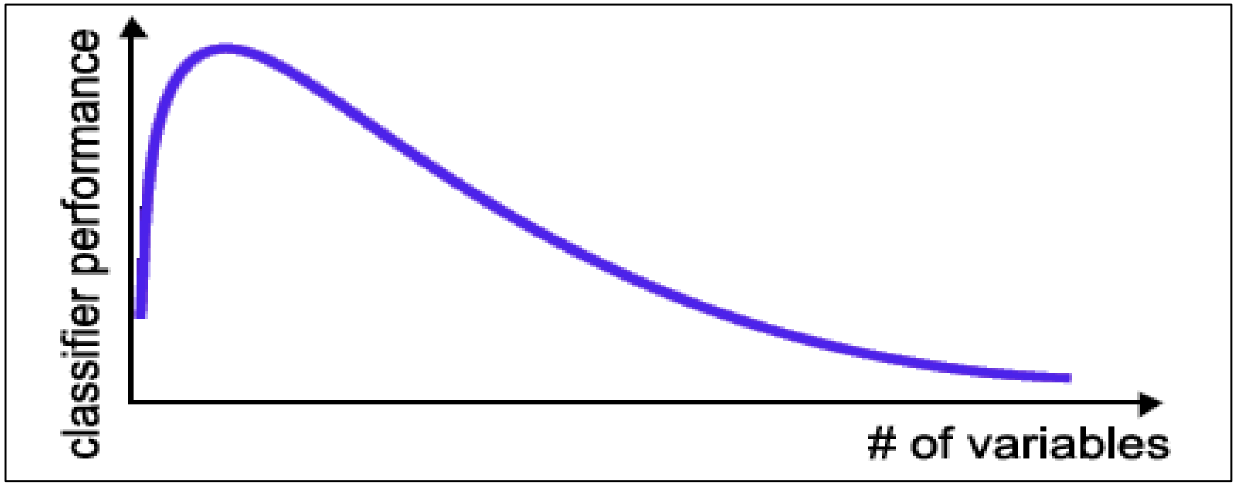
- In practice:

many reasons why this is not the case!

Curse of Dimensionality

- number of training examples is fixed
- Training set is not extremely large

=> the classifier's performance usually will degrade for a large number of features!



Feature Reduction in ML

- Irrelevant and redundant features
- In algorithms like KNN irrelevant feature can introduce noise
- can confuse learning algorithm
- Limited training data.
- Limited computational resources.
- **Curse of dimensionality**.

- Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.
- It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

•**Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.

•**Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

•**Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

•**Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.

•**Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA

- The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:
- i) The principal component must be the linear combination of the original features.
- ii) These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- Iii) The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and nth PC will have the least importance.

Steps for PCA algorithm

•**Getting the dataset**

Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

•**Representing data into a structure**

Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

•**Standardizing the data**

In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.
If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

•**Calculating the Covariance of Z**

To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

- **Calculating the Eigen Values and Eigen Vectors**
  Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
- **Sorting the Eigen Vectors**
  In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
- **Calculating the new features Or Principal Components**
  Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
- **Remove less or unimportant features from the new dataset.**
  The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Singular Value Decomposition (SVD)

SVD is a matrix factorization technique commonly used in linear algebra.

A=UWV^T

Where U and V are orthonormal matrices.

U is an m X m matrix of the orthonormal eigen vectors of AA^T

V is transpose of a n X n matrix containing orthonormal eigenvectors of AA^T

W is n X n diagonal matrix of the singular values which are the square roots of the eigenvalues of AA^T

23

- **Linear Discriminant Analysis** is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.
  For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping. So, we will keep on increasing the number of features for proper classification.
- Two criteria are used by LDA to create a new axis:
- Maximize the distance between means of the two classes.
- Minimize the variation within each class

Steps of LDA

- Calculate the mean vectors for the individual classes
- Calculate intra-class and inter-class scatter matrices
- Calculate eigenvalues and eigenvectors for $1/S_w$ and S_B.
- First one is the intra-class scatter and second one is inter-class scatter matrix. (calculations are in next slide)
- Identify the top ‘k’ eigenvectors and eigenvalues

- .

$$S_W = \sum_{i=1}^{c} S_i$$

$$S_i = \sum_{x \in D_i}^{n} (x - m_i)(x - m_i)^T$$

$m_i \Rightarrow$ mean vector of ith class

$$S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$$

Key drivers of feature selection

- Feature relevance
- Feature redundancy

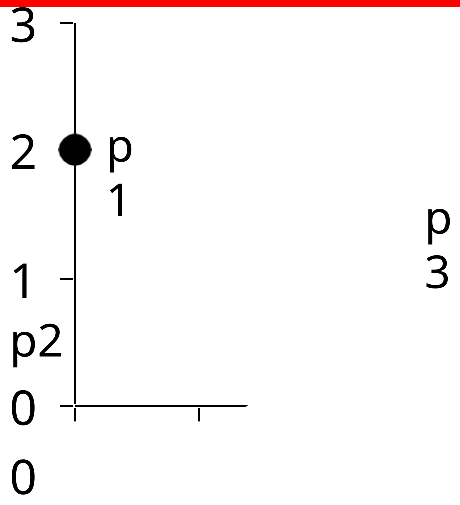
Measures of feature redundancy

- Correlation based measure
- Distance based measure
- Other coefficient based measure

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

d(x,y) is Euclidean distance, where *n* is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) or data objects **x** and **y**.

- Standardization is necessary, if scales differ.



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix

- Minkowski Distance is a generalization of Euclidean Distance

$$d(F_1, F_2) = \sqrt[r]{\sum_{i=1}^{n} (F_1 - F_2)^r}$$

Where r is a parameter, n is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) or data objects $\boldsymbol{x}$ and $\boldsymbol{y}$.

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance.

- $r = 2$. Euclidean distance

- $r \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.

– This is the maximum difference between any component of the vectors

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1	p1	p2	p3	p
p1	0	4	4	
p2	4	0		
p3	4			
p4				

L2	p1	p2	p3	p
p1	0	2.828	3	
p2	2.828			
p3	3			
p4				

L_∞	p1	p2	p3	p
p1	0	2		
p2	2			
p3				
p				

Distance Matrix

- Manhattan Distance between p1and p2
  - $D_{L1}(p1,p2) = |0-2|+|2-0|=4$
- Euclidean Distance
  - $D_{L2}(p1,p2) =(| 0-2|^2+|2-0|^2)^{0.5}=2.828$
- L_{max} norm or L_{∞} norm Distance
  - $D_{L\,\infty}(p1,p2)=Max\{|0-2|, |2-0|\}=2$

- Calculate the distances between the following data using Minkowski family of distances
- (-14.2,-3) and (12.2,1)
- (2,1) and (-3.66,-2.66)
- (-1.5,-1) and (-4.5,-3.5)
- (7.45,6.99) and (13.25,12.45)
- (0.22,-0.2) and (1.5,-1.5)

Hamming Distance

- To calculate distance between two binary vectors hamming distance is used. Measures difference of values position wise.
- Ex: 01101011 and 11001001 having hamming distance 3.

Other similarity measures

- **Cosine Similarity**
- If $\mathbf{d}_1$ and $\mathbf{d}_2$ are two document vectors, then

$\cos(\mathbf{d}_1, \mathbf{d}_2) = \langle\mathbf{d}_1,\mathbf{d}_2\rangle / ||\mathbf{d}_1|| \, ||\mathbf{d}_2||$,

where $\langle\mathbf{d}_1,\mathbf{d}_2\rangle$ indicates inner product or vector dot product of vectors, $\mathbf{d}_1$ and $\mathbf{d}_2,$ and $||\ \mathbf{d}\ ||$ is the length of vector $\mathbf{d}$.

- Example:

$\mathbf{d}_1 =$ **3 2 0 5 0 0 0 2 0 0**

$\mathbf{d}_2 =$ **1 0 0 0 0 0 0 1 0 2**

$\langle\mathbf{d}_1, \mathbf{d2}\rangle = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$

$|\ \mathbf{d}_1 || = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$

$||\ \mathbf{d}_2 || = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.449$

$\cos(\mathbf{d}_1, \mathbf{d}_2) = 0.3150$

Jaccard similarity

The Jaccard similarity index (sometimes called the Jaccard similarity *coefficient*) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Numerical Problem

A simple example using set notation: How similar are these two sets?

A = {0,1,2,5,6}

B = {0,2,3,4,5,7,9}

Solution: $J(A,B) = |A \cap B| / |A \cup B| = |\{0,2,5\}| / |\{0,1,2,3,4,5,6,7,9\}| = 3/9 = 0.33$.

Notes:

The cardinality of A, denoted |A| is a count of the number of elements in set A.

Although it's customary to leave the answer in decimal form if you're using set notation, you could multiply by 100 to get a similarity of 33.33%.

Feature Selection

- The accuracy of the classifier depends not only on the classification algorithm but also on the feature selection method used. Selection of irrelevant and inappropriate features may confuse the classifier and lead to incorrect results.

Feature Selection

Problem of selecting some subset of features, while ignoring the rest .

Feature Extraction

- Project the original x_i, $i=1,\ldots,d$ dimensions to new $k < d$ dimensions, z_j, $j=1,\ldots,k$

Criteria for selection/extraction:
either improve or maintain the classification accuracy, simplify classifier complexity.

Feature Selection - Definition

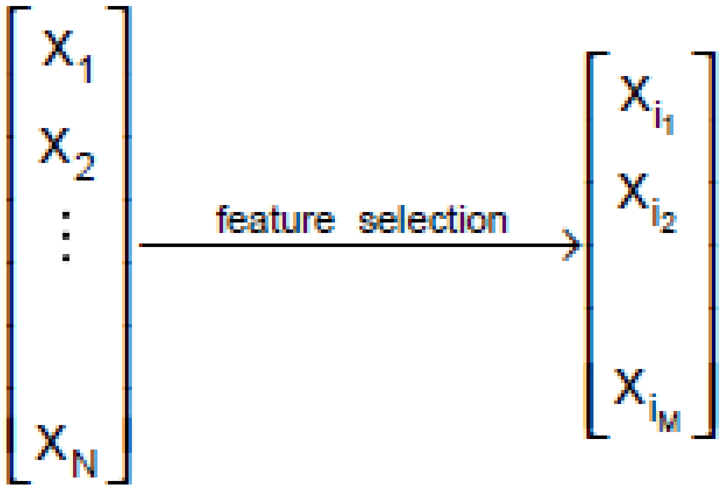
- Given a set of features $F = \{x_1, \dots, x_n\}$

the Feature Selection problem is

to find a subset $F' \subseteq F$ that maximizes

the learners ability to classify patterns.

- Formally F' should maximize some scoring function



$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \\ X_N \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} X_{i_1} \\ X_{i_2} \\ \\ X_{i_M} \end{bmatrix}$$

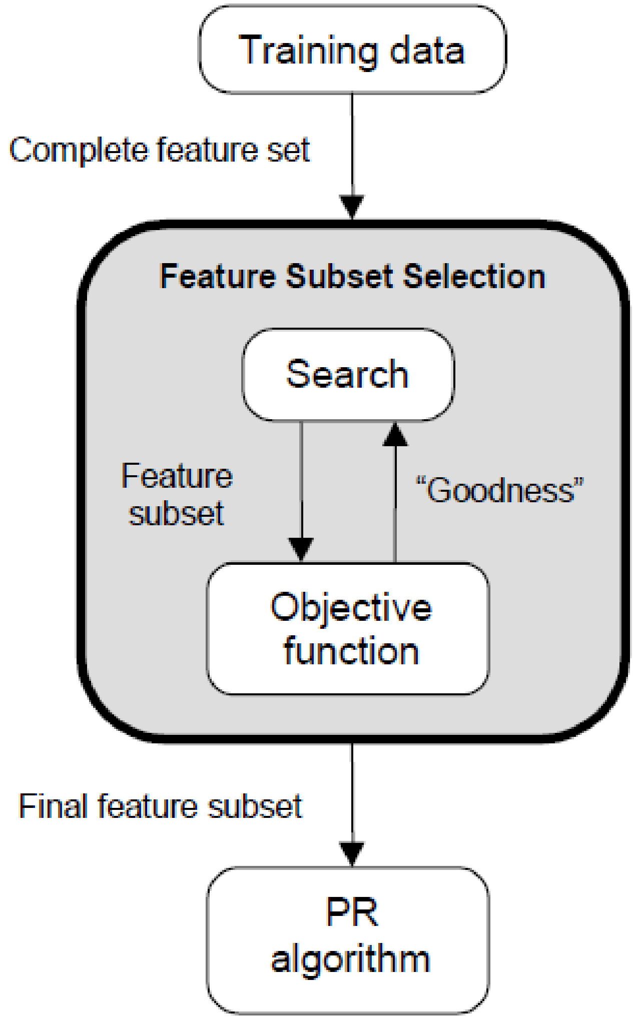
Optimization criteria

- Improve or maintain classifier accuracy
- Simplify classifier complexity

Feature Selection Steps

Feature selection is an **optimization** problem.

- Step 1: Search the space of possible feature subsets.
- Step 2: Pick the subset that is optimal or near-optimal with respect to some objective function.



Evaluating feature subset

- Supervised (wrapper method)
  - Train using selected subset
  - Estimate error on validation dataset

- Unsupervised (filter method)
  - Look at input only
  - Select the subset that has the most information

Embedded Approaches

- Feature selection occur naturally as a part of data mining algorithm
- During the operation of data mining algorithm, the algorithm itself decides which attribute to use and which to ignore
- Algorithm to built decision tree classifier use such approach

Filter approaches

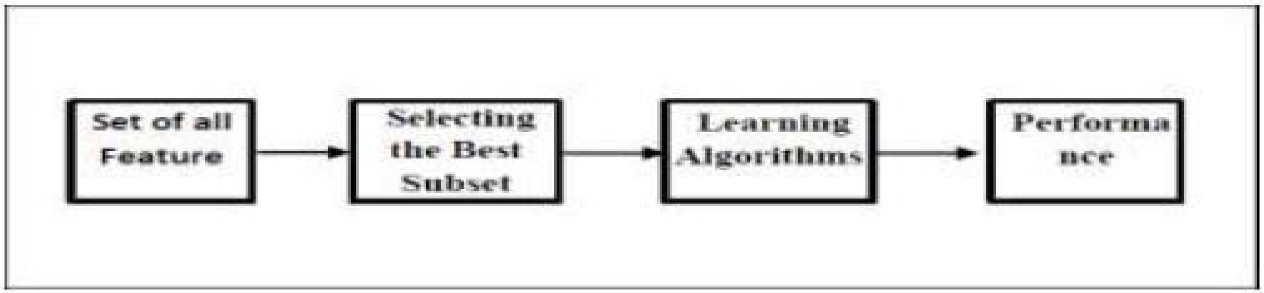
- Filter approaches: This method selects the feature without depending upon the data mining task (the type of classifier used).
- It is selected before the data mining algorithm run. We might select set wise attributes whose pair-wise correlation is low

Advantage of the method

- It is simple and independent of the type of classifier used so feature selection need to be done only once.

Drawback of the method

- It ignores the interaction with the classifier ignores the feature dependencies, and each feature considered separately



Feature selection

Univariate (looks at each feature independently of others)

– Pearson correlation coefficient

- F-score
- Chi-square
- Signal to noise ratio
- mutual information

Univariate methods measure some type of correlation between two random variables

•the label (y_i) and a fixed feature (x_{ij} for fixed j)

- Rank features by importance
- Ranking cut-off is determined by user

Pearson correlation coefficient

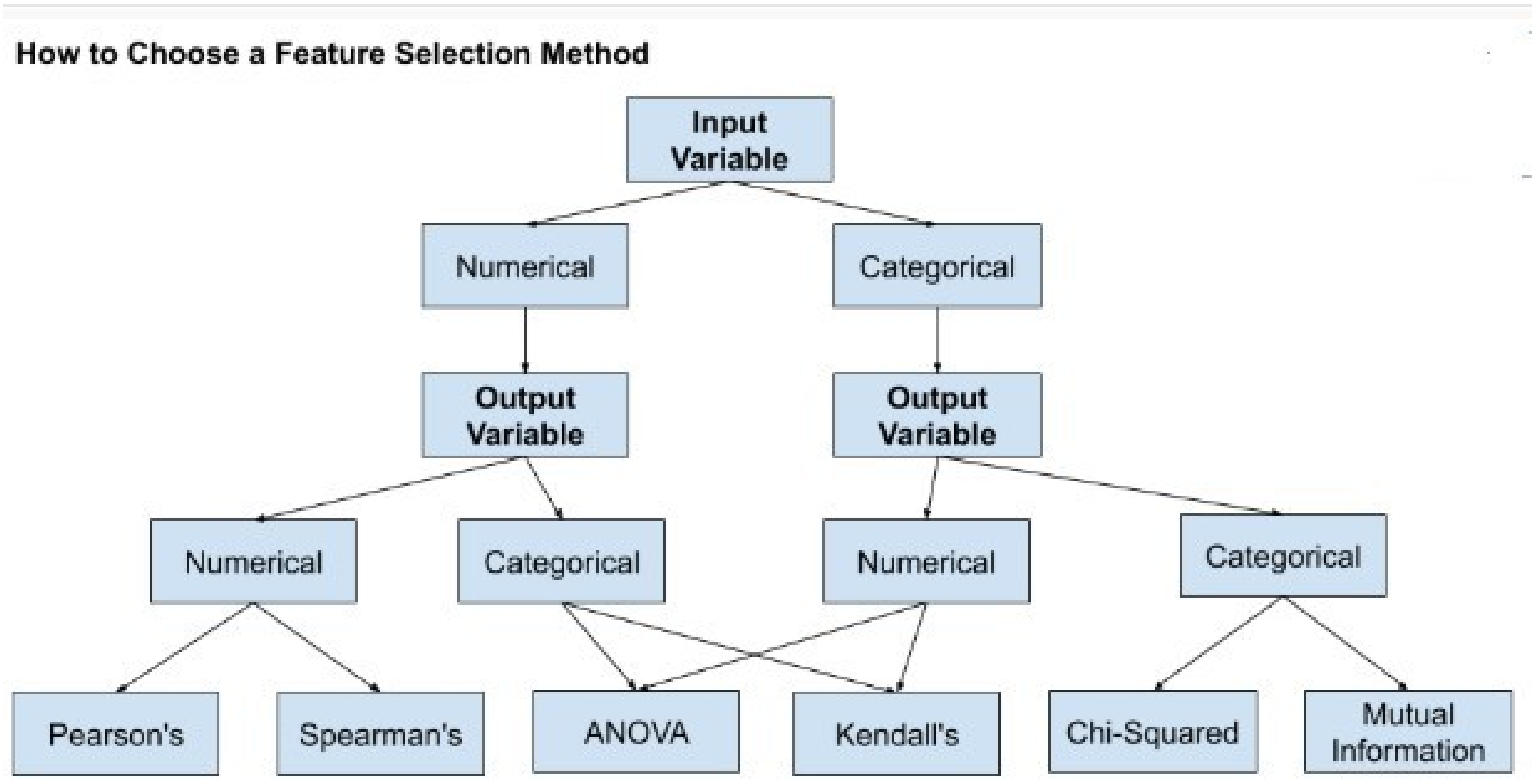
- Measures the correlation between two variables
- Formula for Pearson correlation =

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

- The correlation r is between $+1$and $-1.$

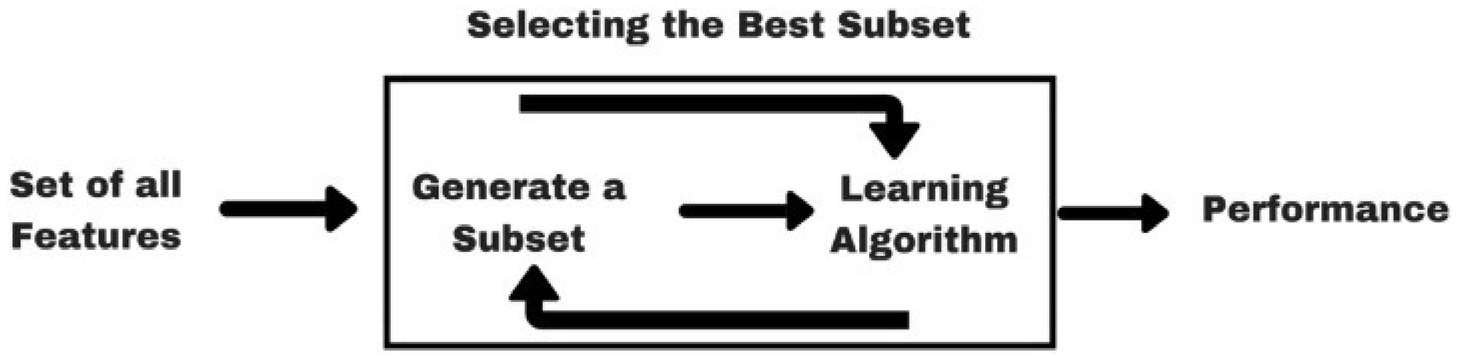
- $+1$means perfect positive correlation
  - -1in the other direction

How to choose a Feature Selection Method



Wrapper method

- These methods use the target data mining algorithm as a black box to find best subset of attributes.
- Typically without enumerating all possible subsets



Wrapper method

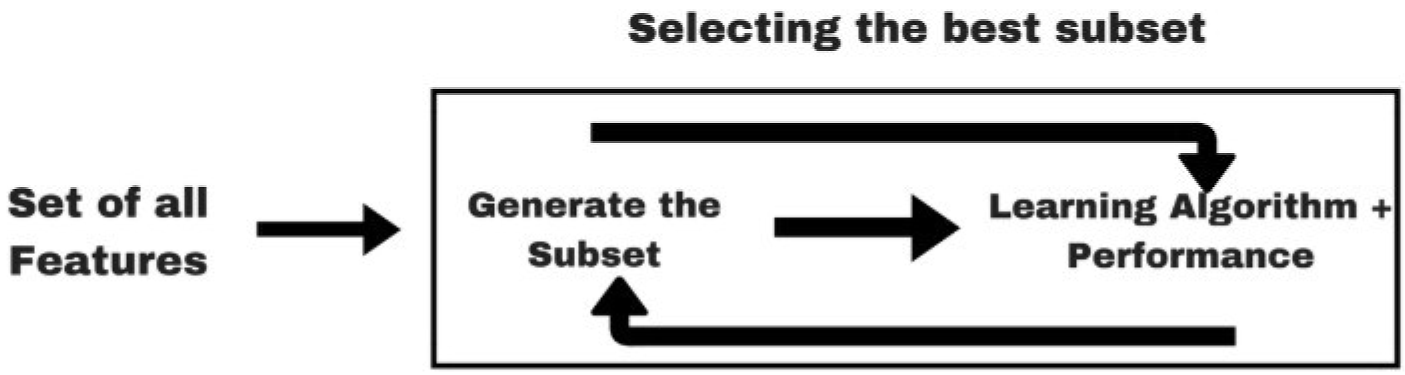
- This method the feature is dependent upon the classifier used, i.e. it uses the result of the classifier to determine the goodness of the given feature or attribute.
- **The advantage** of this method is that it removes the drawback of the filter method, i.e. It includes the interaction with the classifier and also takes the dependencies
- **Drawback** of this method is that it is slower than the filter method because it takes the dependencies also. The quality of the feature is directly measured by the performance of the classifier.

Subset selection

- Select uncorrelated features
- Forward search
  - Start from empty set of features
  - Try each of remaining features
  - Estimate classification/regression error for adding specific feature
  - Select feature that gives maximum improvement in validation error
  - Stop when no significant improvement
- Backward search
  - Start with original set of size *d*
  - Drop features with smallest impact on error

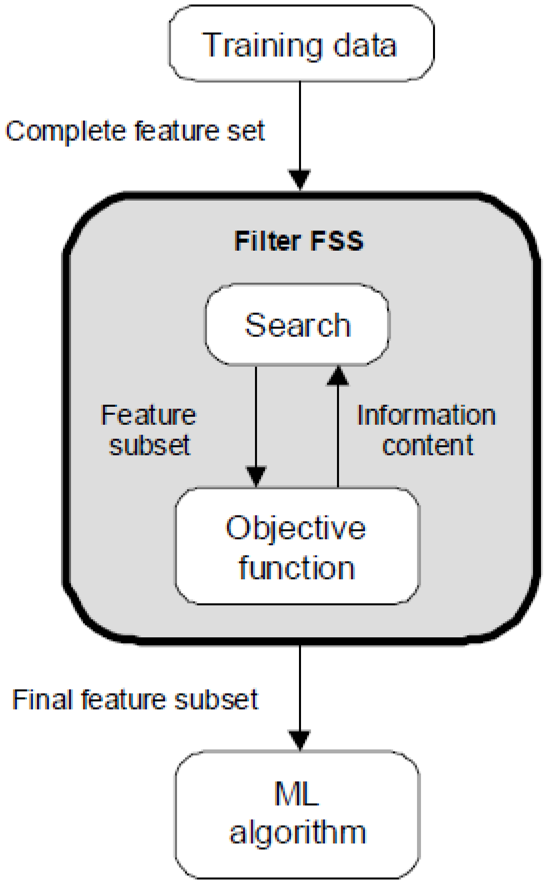
Embedded Approaches

- Such as, it searches for an optimal subset of features that is built into the classifier construction.
- The advantage of this method is that it is less computationally intensive than a wrapper approach.

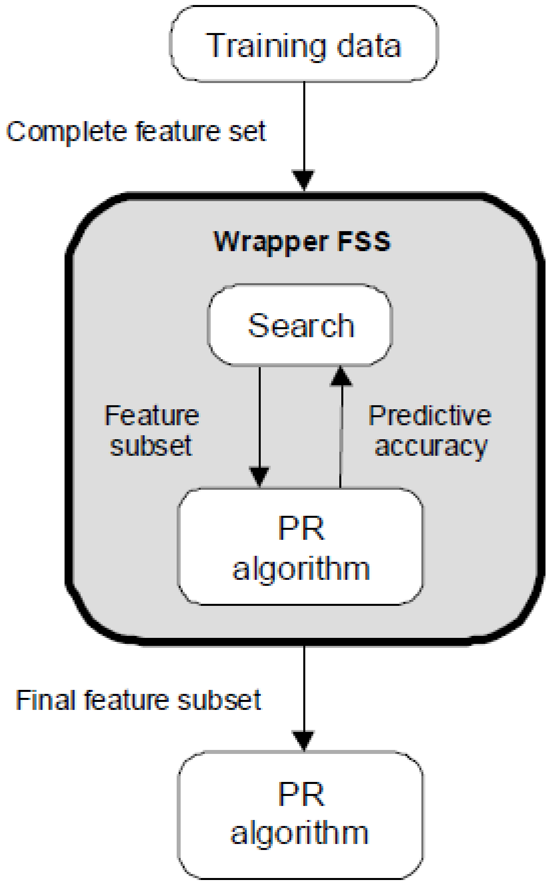


Evaluation Strategies

Filter Methods



Wrapper Methods



Data preprocessing

- Data integration and Cleaning
- Feature selection
- Dimension reduction
- Normalization

- ❑ Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an important equally important attribute(on lower scale) because of other attribute having values on larger scale.

Example: -10, 201, 301, -401, 501, 601, 701,1234

- ❑ Standard method for normalization: Z-score normalization, Decimal Scaling Method For Normalization

Z-score normalization

✓Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an important equally important attribute(on lower scale) because of other attribute having values on larger scale.

✓In this technique, values are normalized based on mean and standard deviation of the data A. The formula used is:

$$\boxed{v' = \frac{v - \overline{A}}{\sigma_A}}$$

Decimal Scaling Method For Normalization –

It normalizes by moving the decimal point of values of the data. To normalize the data by this technique, we divide each value of the data by the maximum absolute value of data. The data value, v_i, of data is normalized to v_i' by using the formula below –

$$\boxed{v_i' = \frac{v_i}{10^j}}$$

where j is the smallest integer such that $\max(|v_i'|)<1$

Let the input data is: -10, 201, 301, -401, 501, 601, 701 To normalize the above data,
Step 1: Maximum absolute value in given data(m): 701
Step 2: Divide the given data by 1000 (i.e j=3)
Result: The normalized data is: -0.01, 0.201, 0.301, -0.401, 0.501, 0.601,
0.701

1. Normalize the following 7 numbers using Z-Score normalization.
   The numbers are as follows:

N1: A random number between 10 to 98 N2: 10*N1

N3: 10*N2 + N2

N4: 12

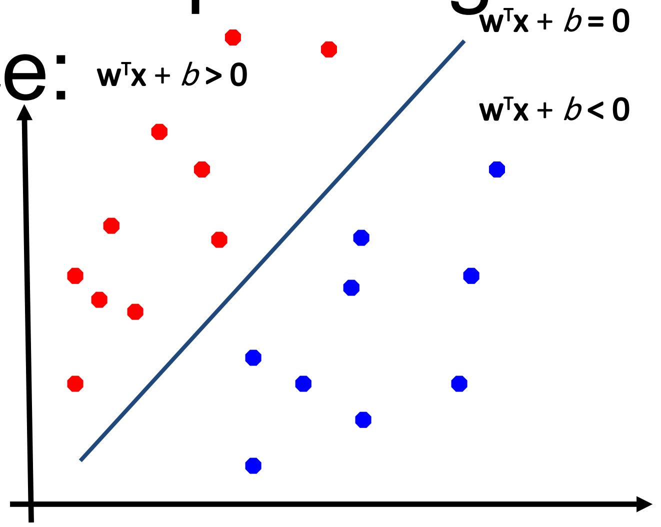
N5: N4*N1*100 + N3 N6: N4+100

N7: N3*100

Support Vector Machine

Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:

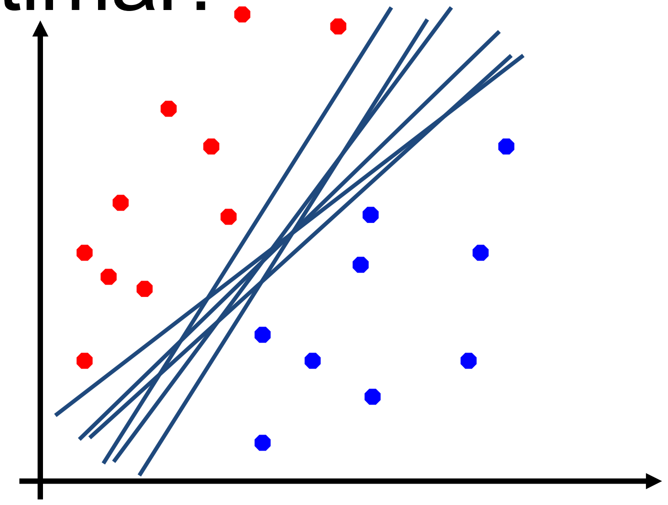


$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b > 0$
$\mathbf{w}^T\mathbf{x} + b < 0$

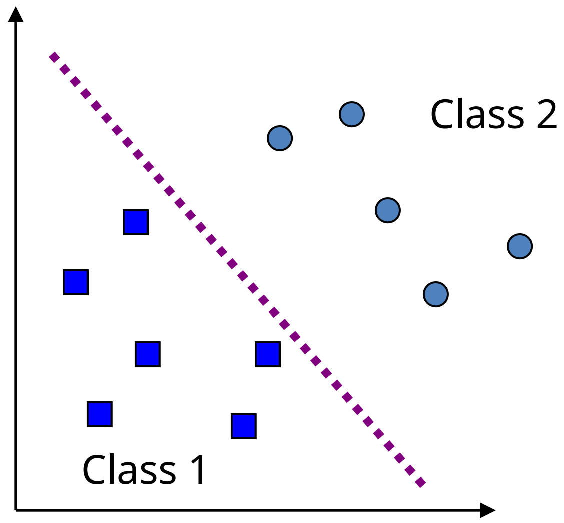
$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

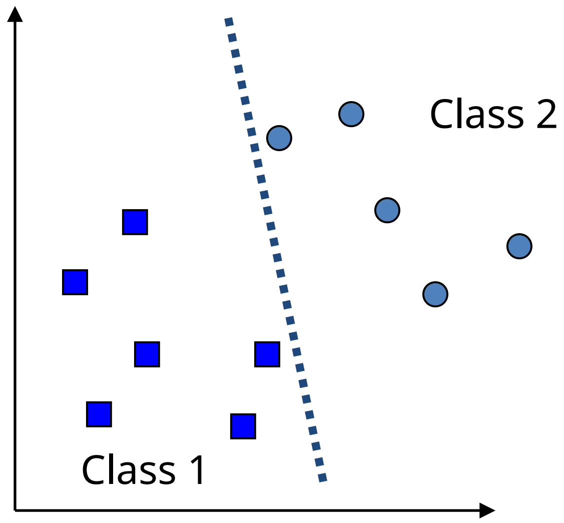
Linear Separators

- Which of the linear separators is optimal?



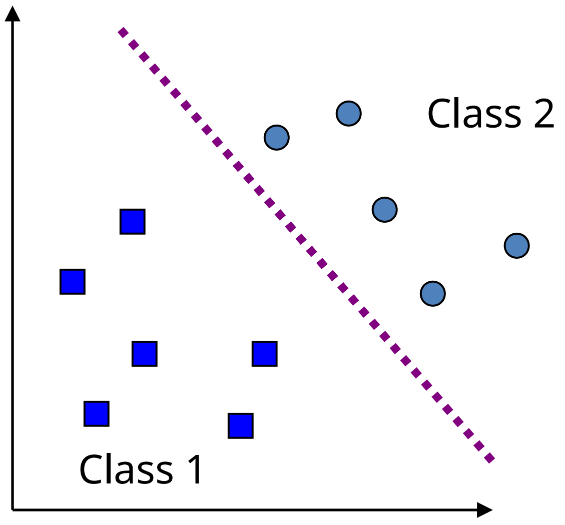
Examples of Bad Decision Boundaries





What is a good Decision Boundary?

- Many decision boundaries!
  - The Perceptron algorithm can be used to find such a boundary
- Are all decision boundaries equally good?



Finding the Decision Boundary

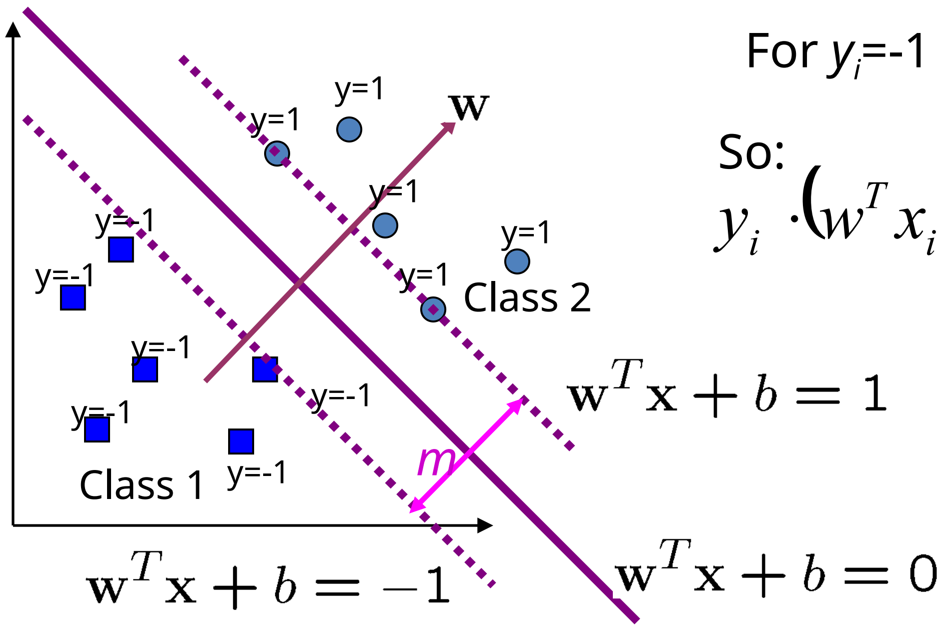
- Let {x_1, ..., x_n} be our data set and let y_i Î {1,-1} be the class label of x_i

For y_i=1 $w^T x_i + b \geq 1$

For y_i=-1 $w^T x_i + b \leq -1$

So:

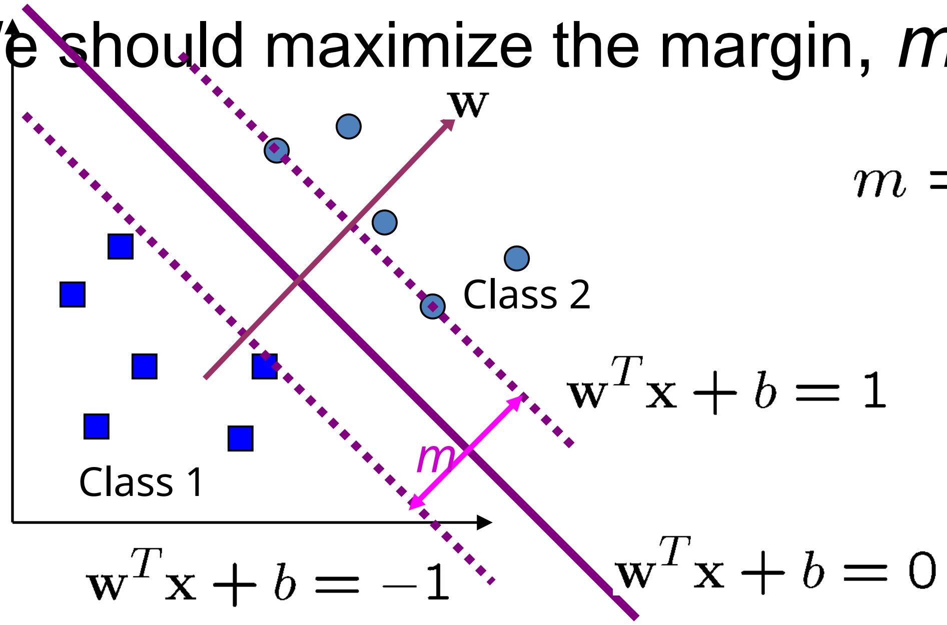
$$y_i \cdot (w^T x_i + b) \geq 1, \forall (x_i, y_i)$$



y=1
y=1
$\mathbf{w}$
y=1
y=1
y=1
Class 2
y=-1
y=-1
y=-1
y=-1
y=-1
y=-1
Class 1
m
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*



$\mathbf{w}$
Class 2
$\mathbf{w}^T\mathbf{x}+b=1$
m
Class 1
$\mathbf{w}^T\mathbf{x}+b=-1$
$\mathbf{w}^T\mathbf{x}+b=0$

$$m = \frac{2}{||\mathbf{w}||}$$

Finding the Decision Boundary

- The decision boundary should classify all points correctly Þ

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

- The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- This is a constrained optimization problem. Solving it requires to use Lagrange multipliers

What is a Support Vector Machine?

- It is a supervised machine learning problem where we try to find a hyperplane that best separates the two classes.
- SVM and logistic regression both the algorithms try to find the best hyperplane (decision boundary), but the main difference is logistic regression is a probabilistic approach whereas support vector machine is based on statistical approaches.
- SVM finds maximum margin between the hyperplanes that means maximum distances between the two classes.

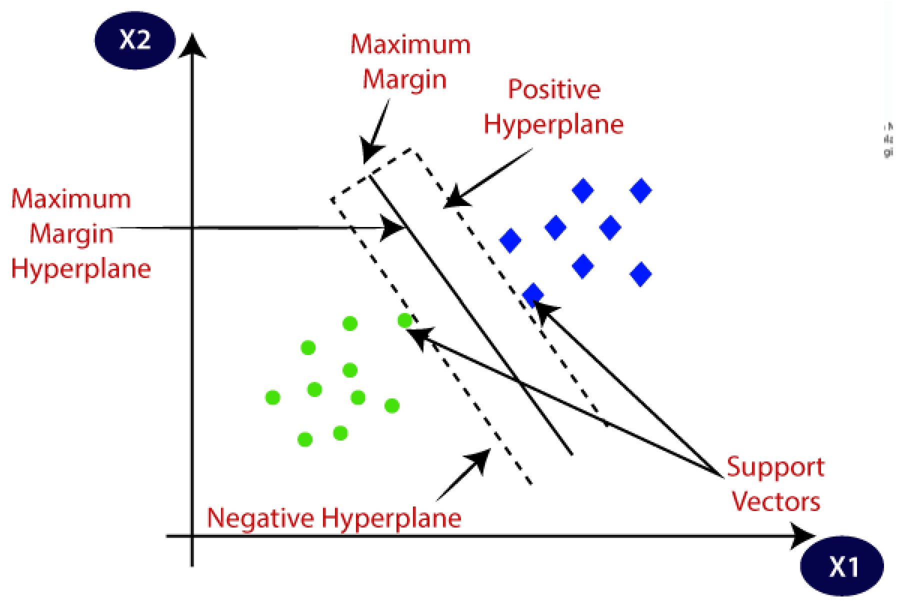
Types of Support Vector Machine

- **Linear SVM**
- When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).
- **Non-Linear SVM**
- When the data is not linearly separable then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a straight line (if 2D) then we use some advanced techniques like kernel tricks to classify them. In most real-world applications we do not find linearly separable datapoints hence we use kernel trick to solve them.

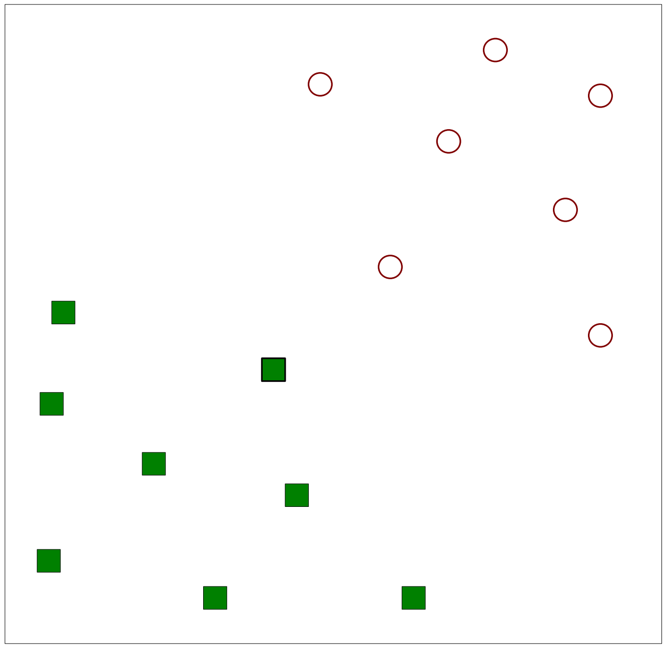
Few mostly used terms

- **Support Vectors:** These are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points.
- **Margin:** it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. There are two types of margins **hard margin** and **soft margin.**

SVM in figure

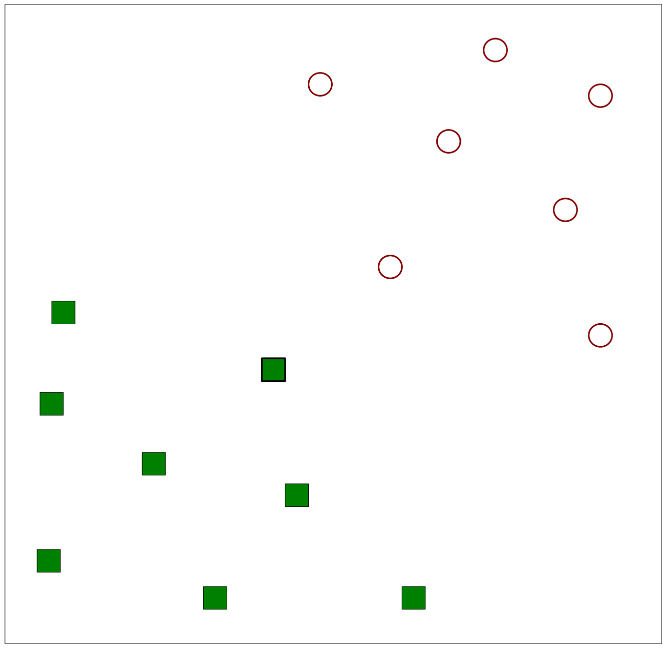


Support Vector Machines



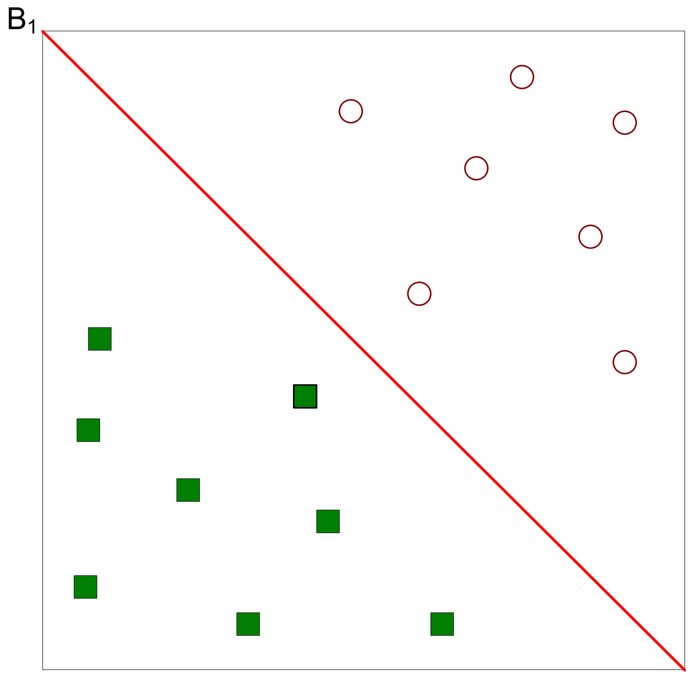
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



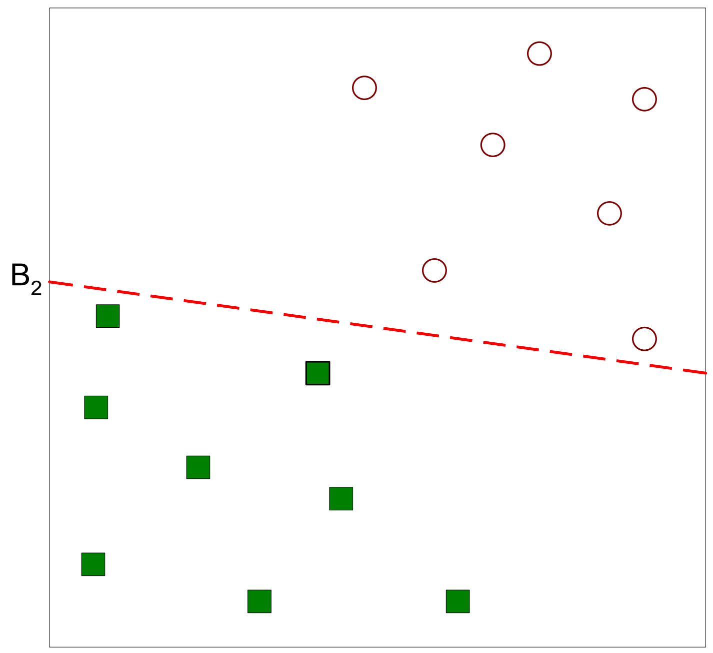
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



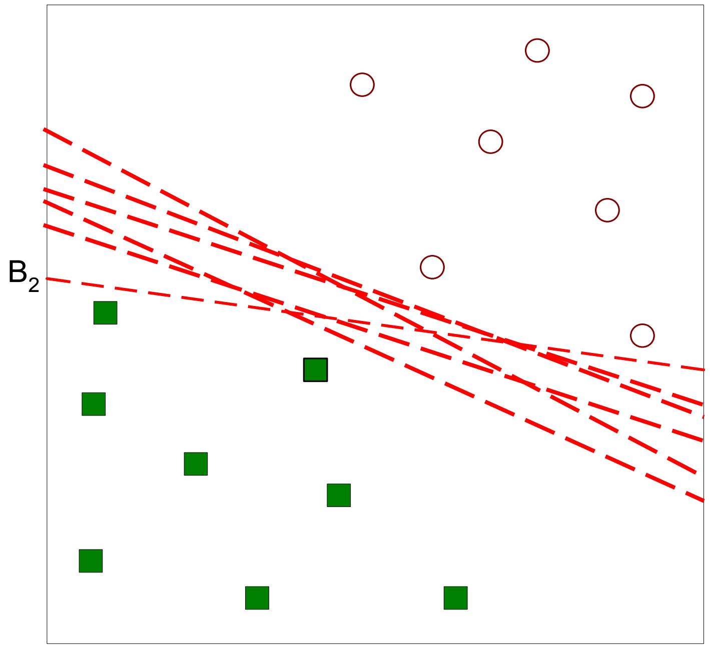
- One Possible Solution

Support Vector Machines



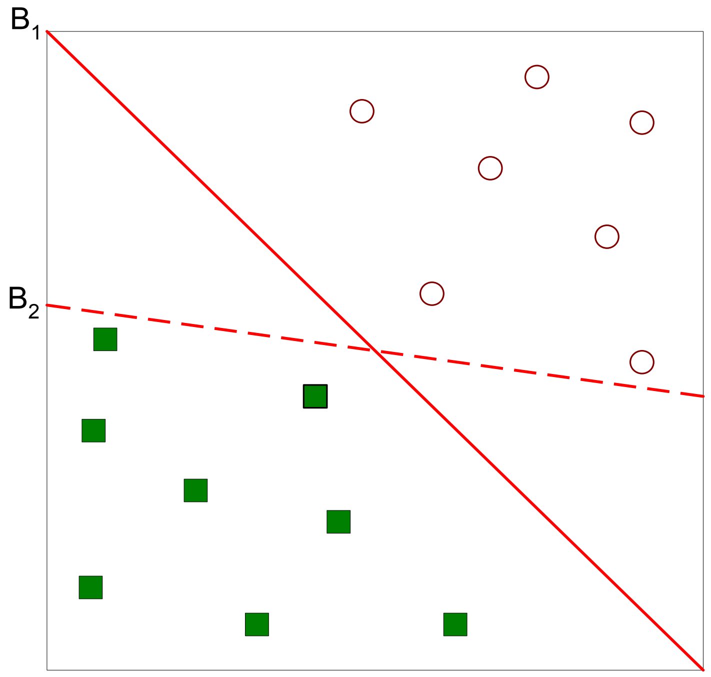
- Another possible solution

Support Vector Machines



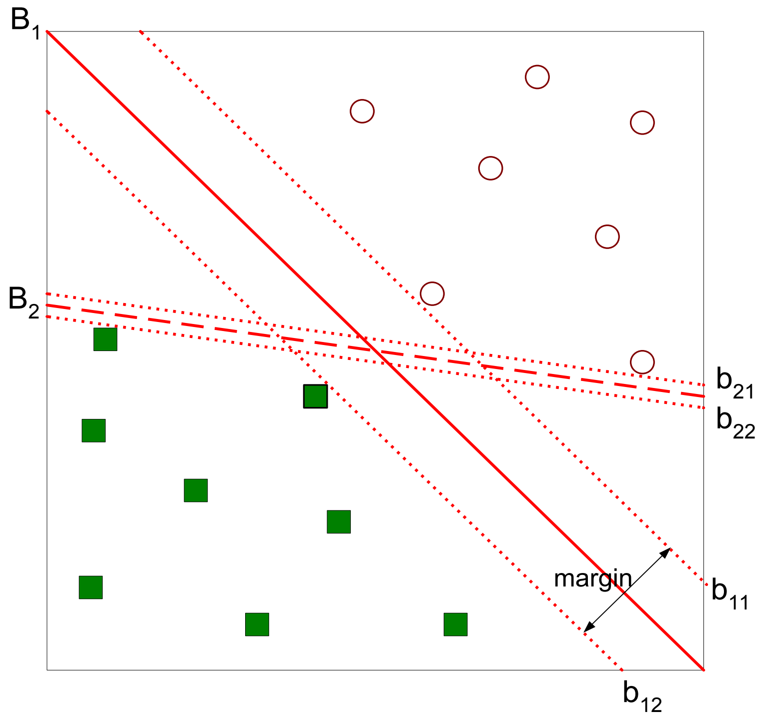
- Other possible solutions

Support Vector Machines



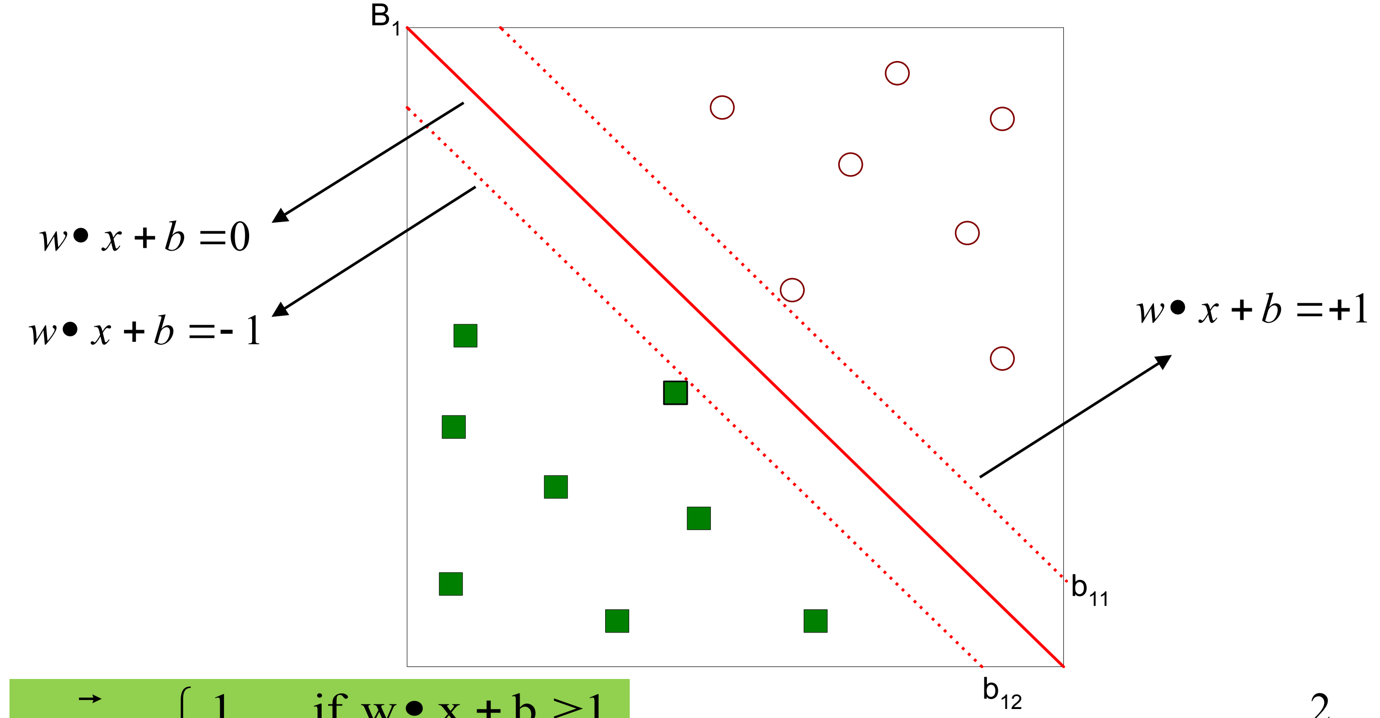
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \ge 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \le -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

Support Vector Machines

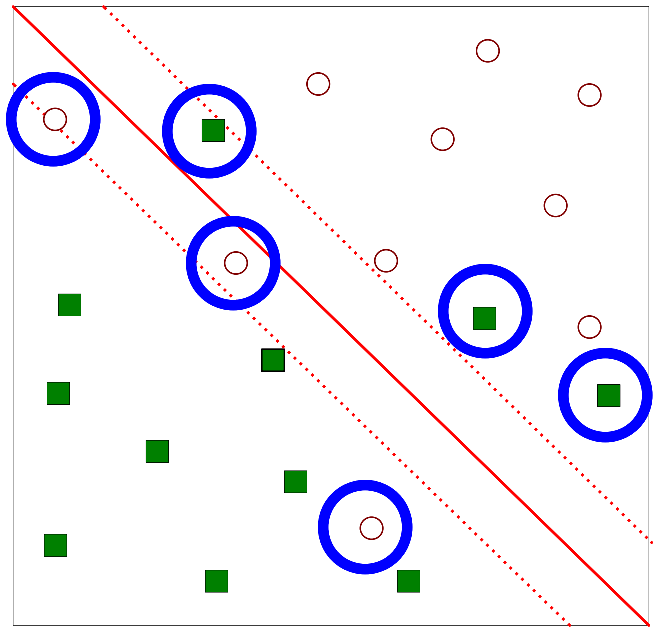
- We want to maximize: $\text{Margin} = \frac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing: $L(w) = \frac{\|w\|^2}{2}$
  - But subjected to the following constraints:

$$\vec{w} \cdot \vec{x_i} + b \geq 1 \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x_i} + b \leq -1 \text{ if } y_i = -1$$

    - This is a constrained optimization problem
      - Numerical approaches to solve it (e.g., quadratic programming)

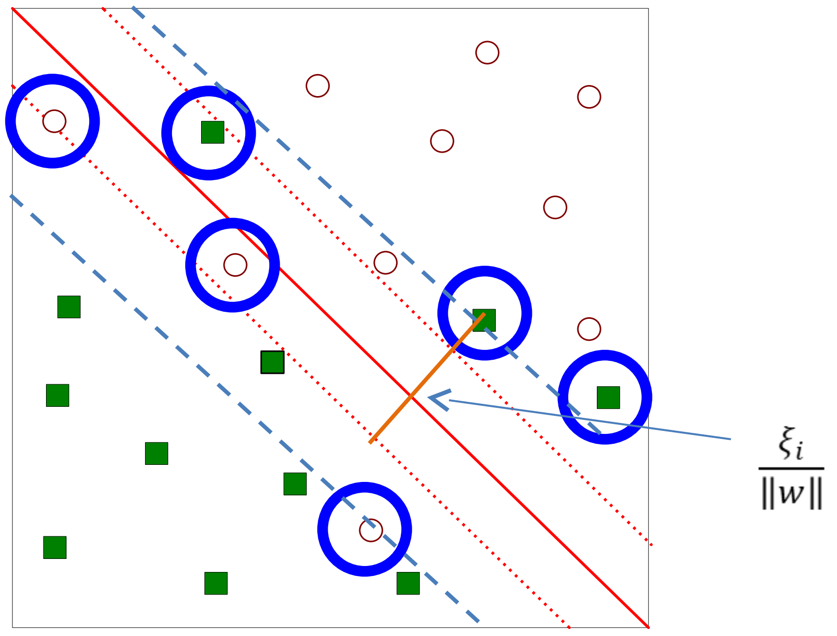
Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:

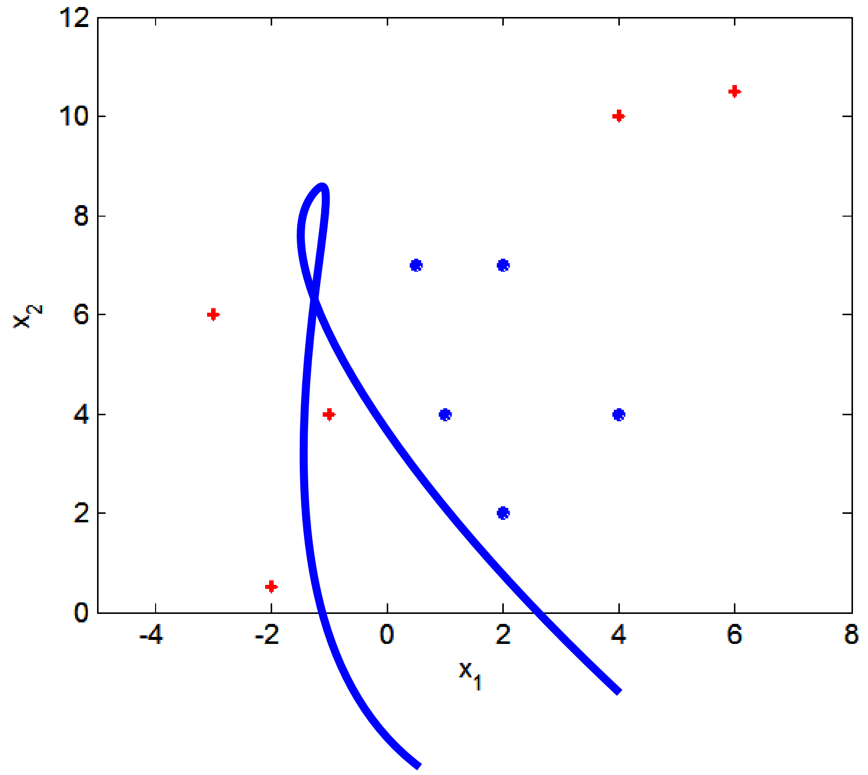
$$L(w) = \frac{\|w\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

Subject to:

$$\vec{w} \cdot \vec{x_i} + b \geq 1 - \xi_i \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x_i} + b \leq -1 + \xi_i \text{ if } y_i = -1$$

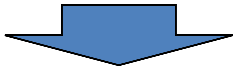
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



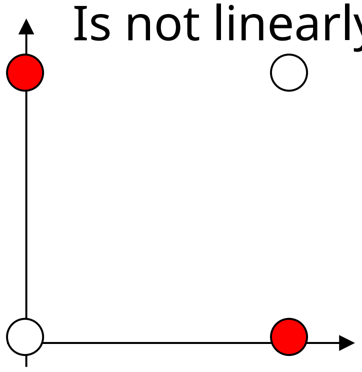
XOR

X	Y	
0	0	0
0	1	1
1	0	1
1	1	0

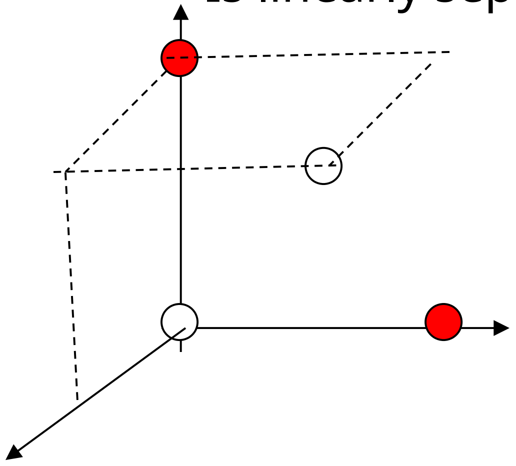


X	Y	XY	
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Is not linearly separable



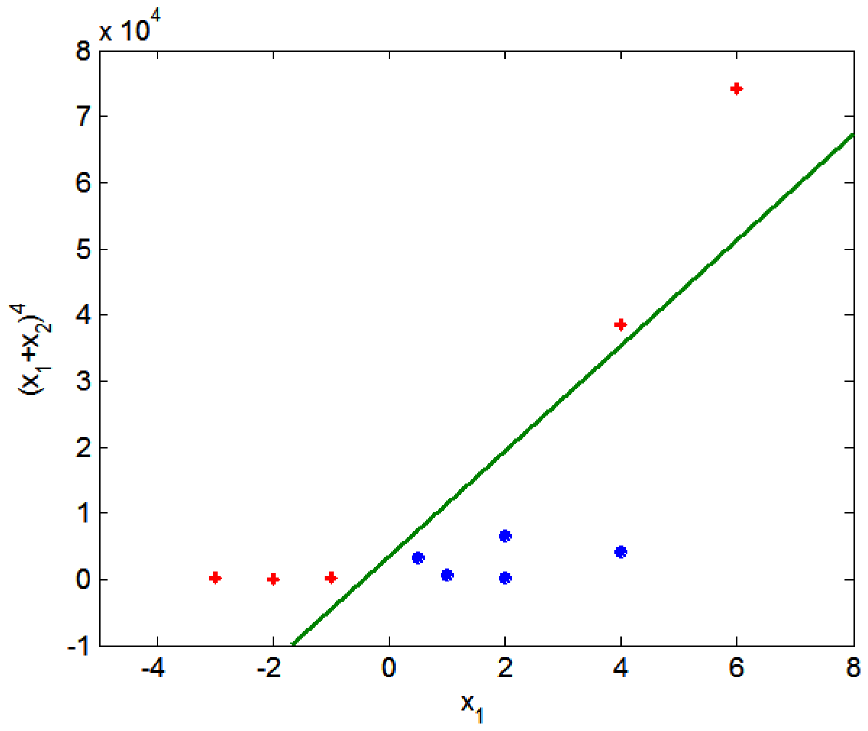
Is linearly separable



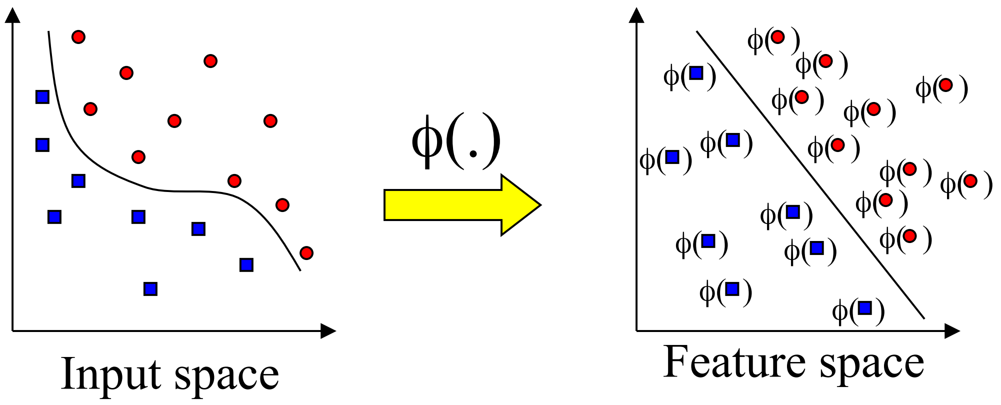
Nonlinear Support Vector Machines

- Transform data into higher dimensional space

Use the Kernel Trick



Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

Kernel trick

- **Kernel:** A kernel is a method of placing a two dimensional plane into a higher dimensional space, so that it is curved in the higher dimensional space. (In simple terms, a kernel is a function from the low dimensional space into a higher dimensional space.)

The Kernel Trick

- Recall the SVM optimization problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kern $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need to choose a “good” kernel function.

Thank You