Data Mining - Classification I

Nhat-Quang Doan

University of Science and Technology of Hanoi

doan-nhat.quang@usth.edu.vn

Overview

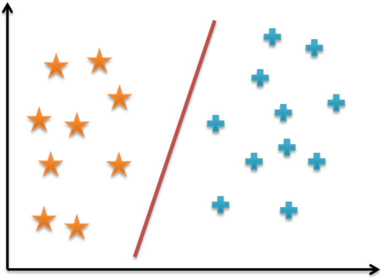
Classification vs Clustering

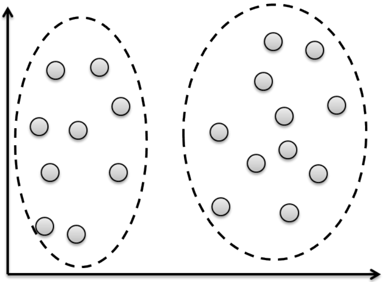
Classification

- Labeled data objects
- Assign a label to new objects

Clustering

- Data is not labeled
- Identify structure in data and group objects

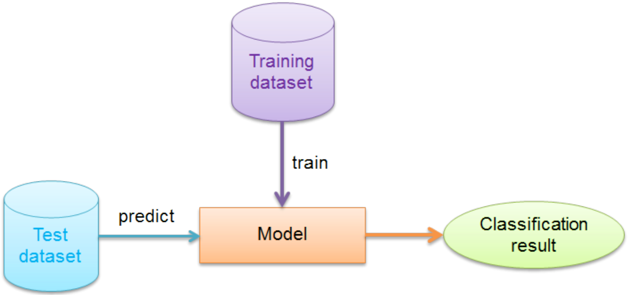




Introduction

Objectives

- Classification is the supervised learning task of data mining that predicts categorical class labels (discrete or nominal)
- A model is built based on the training set and the class labels. This model helps in classifying new data



Introduction

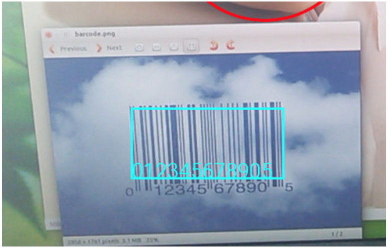
Applications

- Banking: card fraud detection

Introduction

Applications

- Pattern Recognition: Plate Recognition, Optimal Character Recognition



Introduction

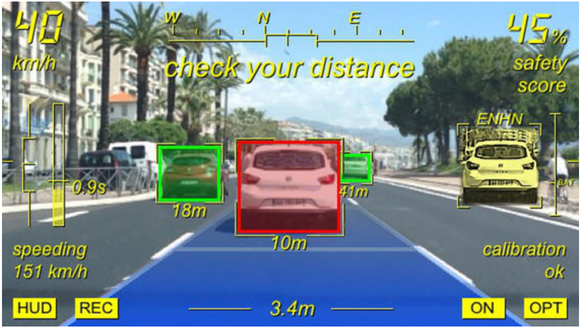
Applications

- Security: Finger Print, Spam Filter.

Introduction

Applications

- Transport: Self-driving vehicules.



Problematics

Difficulties

- Training datasets: overfitting, imbalance of data, ...
- Information about data class
- Classification algorithms: complexity, training time, etc...
- Big data
- Performance

Notions

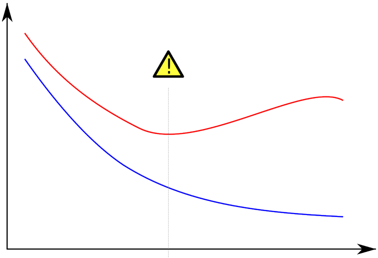
Input set: $(\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_i, y_i)\}, \forall i = 1, .., n$: a set of n training objects
Object: $\mathbf{x}_i \in \mathbb{R}^d$ with $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,d})$, $y_i \in \mathcal{Y}$ (y_i is the class label of $\mathbf{x}_i$
Model: Model $\mathcal{M}$ allows to classify $\mathcal{X}$ into many groups according to its $\mathcal{Y}$.
Problem: For new object (test data) $\mathbf{x}_0 \notin \mathcal{X}$ and unknown y_0, the model $\mathcal{M}$ is employed to predict y_0.

Overfitting

Definition

In data mining, overfitting problem occurs:

- A model is excessively complex, such as having too many parameters relative to the number of training objects.
- A model has poor predictive performance, as it overreacts due to the training data.



Overfitting

Overfitting

Many approaches to manage training datasets have been proposed to reduce variance and avoid overfitting:

- cross validation: leave-p-out , k-fold cross-validation
- bagging or bootstrap agrgregating
- and many more....

Training dataset generation

Cross-validation

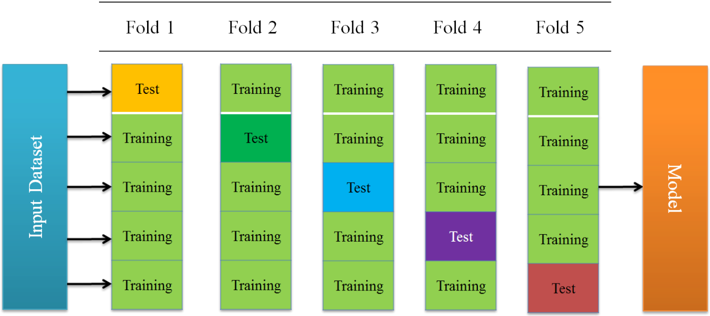
This technique is a statistics model:

- generalize to an independent data set
- involve partitioning a sample of data into complementary subsets: building the model from one subset (the training set), and validating the model on the other subset (the testing set)

Training dataset generation

k-fold cross validation

- generalize to k subsets
- $k-1$ subsets for training and 1 subset for testing



5-fold cross validation

Training dataset generation

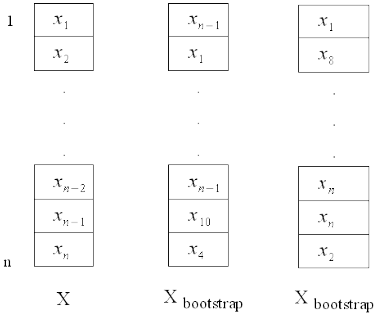
Leave-p-out

- Leave-p-out take p data out from the input dataset of n data, $n - p$ for training and p for testing
- This cross-validation requires to learn and validate C_n^p times (if $n = 100$ and $p = 30$, $C_{100}^{30} = 3 * 10^{25}$)
- Leave-one-out is often used as for $C_n^1 = n$

Training dataset generation

Bagging

- consider input set $\mathcal{X}$ with n training objects
- create k new training sets $\mathcal{X}_i$ by drawing n objects with replacement



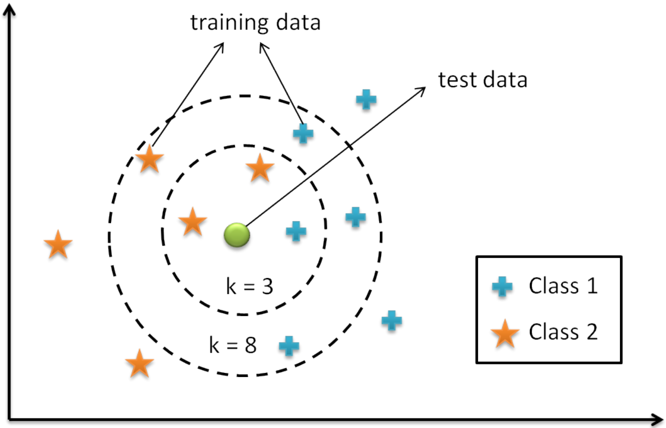
Definition

k-NN

k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method often used for classification.

- The input consists of the k closest training examples in the feature space.
- The output is a class label. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.
- k is a positive integer, typically small for example $k = 1, 3, 5, 10$, etc.

k-NN



Pros and Cons

Advantages

- No input parametric
- No training step required
- Simple to understand and to interpret

Disadvantages

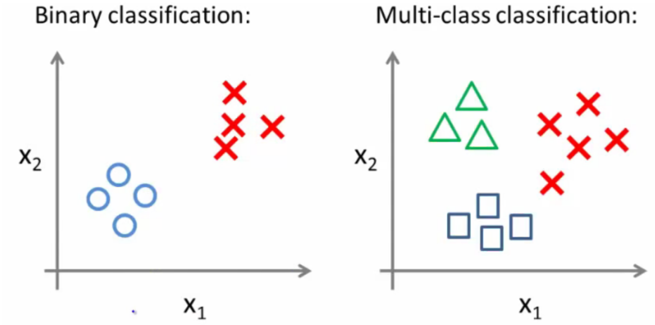
- Require many operations for testing
- Low accuracy comparing to Decision Tree, Random Forest, SVM, Neural Networks, etc...

Binary Classification

Input set: $(\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_i, y_i)\}, \forall i = 1, .., n$: a set of N training objects
Object: $\mathbf{x}_i \in \mathbb{R}^d$ with $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,d})$, $y_i \in \mathcal{Y}$ (y_i is $\{-1; +1\}$
Model: A learn classifier $f(\mathbf{x}_i)$ such that

$$f(\mathbf{x}_i) = \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

Binary Classification



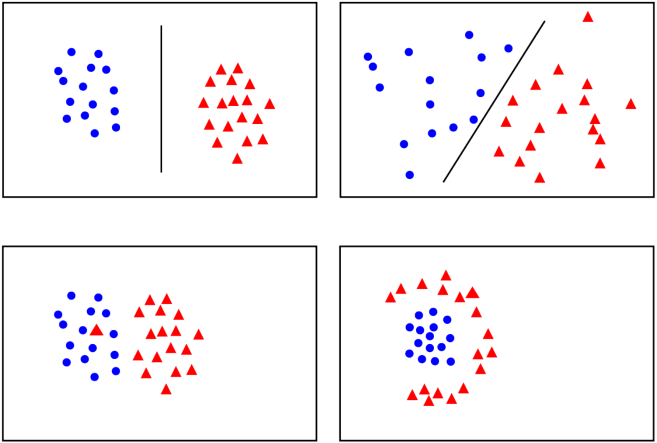
Binary vs Multi-class classification

A multi-class classifier is able to classify into more 2 classes.
Binary classifier can only deal with 2-class problems

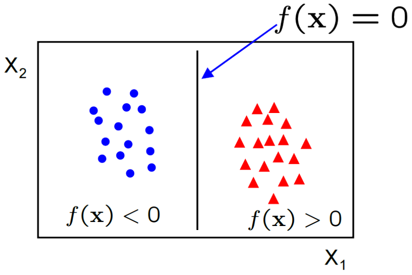
Binary Classification

linearly separable

not linearly separable



Binary Classification



A linear classifier has the form:

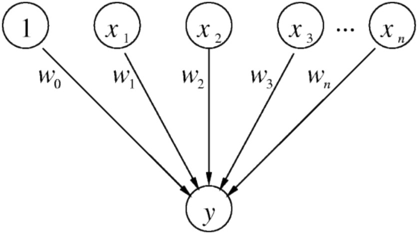
$$f(\mathbf{x}) = \mathbf{w}\mathbf{x}^T + b$$

- in 2D the discriminant is a line, in kD ($k \geq 3$), is a hyperplane denoted by $\Delta(\mathbf{v}, a) = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{v}\mathbf{x}^T + a = 0\}$
- **w** is the normal to the line, and b the bias
- **w** is known as the weight vector

Perceptron Classifier

Perceptron

- An algorithm for supervised learning of binary classifiers.
- A type of linear classifier.



$$y = w_0 + \sum_{i=1}^{n} x_i w_i$$

Perceptron Classifier

Given linearly separable data $\mathcal{X}$ labelled into two categories $y_i = \{-1, 1\}$, find a weight vector **w** such that the discriminant function:

$$f(\mathbf{x}) = \mathbf{w}\mathbf{x}^T$$

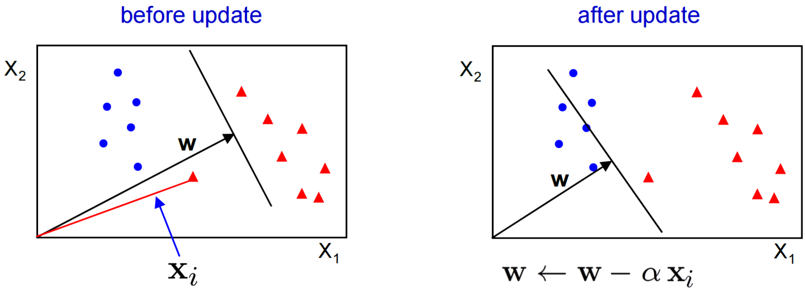
Perceptron Algorithm

- intialize $\mathbf{w}_0 = \mathbf{0}$ (or close to $\mathbf{0}$)
- for all $\mathbf{x}_i \in \mathcal{X}$, if $\mathbf{x}_i$ is misclassified

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \alpha sign(f(\mathbf{x}_i))\mathbf{x}_i$$

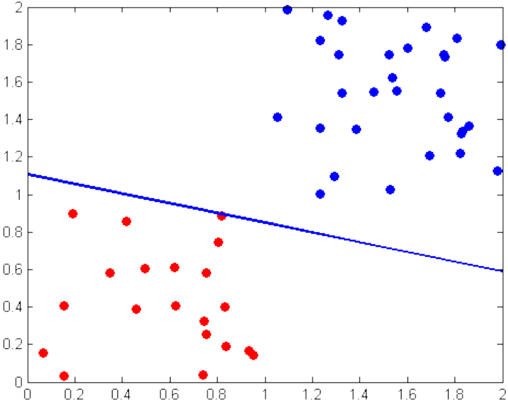
- until all the data is correctly classified

Perceptron Classifier



if the data is linearly separable, then the algorithm will converge.
After the convergence, $\mathbf{w} = \alpha \sum \mathbf{x}_i$.

Perceptron Classifier



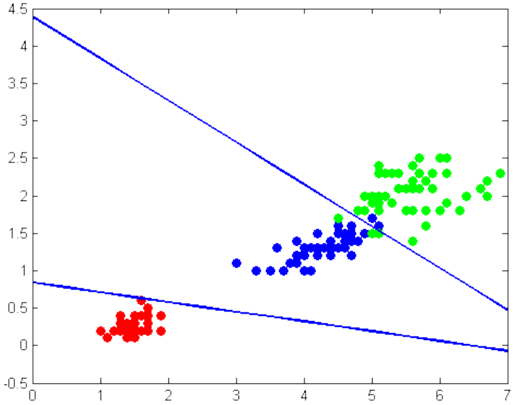
Perceptron Classifier for 2-class dataset

Perceptron Classifier

- Convergence can be slow and Perceptron algorithm depends heavily on input parameters $\mathbf{w}_0$ and α
- Perceptron is able to handle 2-class problems.

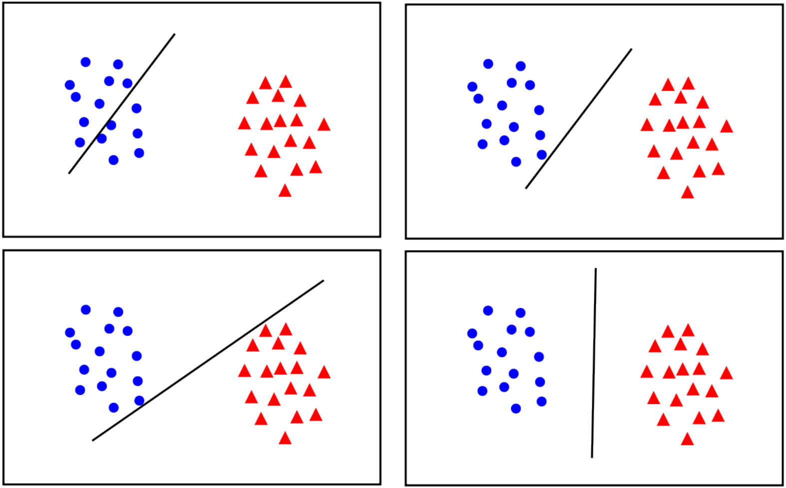
- How to use Perceptron for multi-class problems?
- What if we have 3 classes such as Iris dataset?

Perceptron Classifier



Perceptron Classifier for Iris dataset

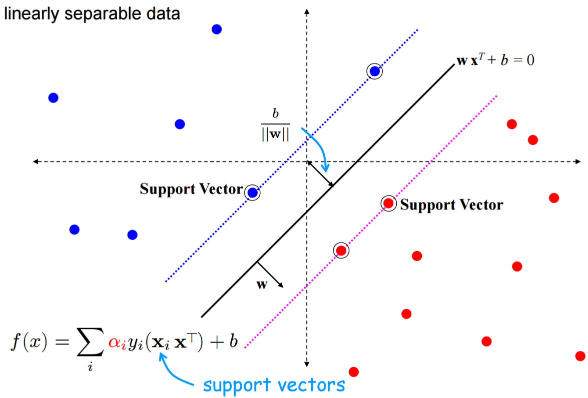
Binary Classification



Are all decision boundaries equally good? or what is the best **w**?

Support Vector Machines

The decision boundary or the hyperplan should be as far away from the data of both classes as possible.



Support Vector Machines

A support vector machine (SVM) is a supervised learning technique from the field of machine learning applicable to both classification and regression.

- Rooted in the Statistical Learning Theory developed by Vladimir Vapnik and co-workers at AT&T Bell Laboratories in 1995
- Very famous due to its success for hand writting recognitions (or image classification in general)

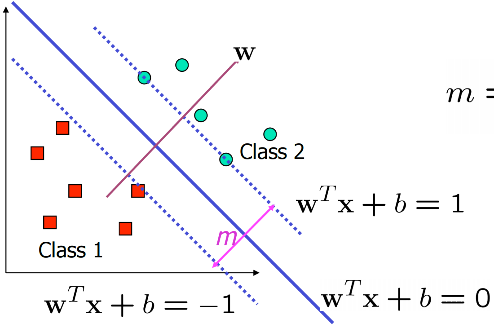
Support Vector Machines

- SVMs are based on the principle of Structural Risk Minimization.
- Non-linearly map the input space into a very high dimensional feature space in order to construct an optimal separating hyperplane in this space (a maximal margin classifier)

Support Vector Machines

The margin, the smallest distance between the decision boundary $\Delta(\mathbf{v}, a)$ and the examples $\mathbf{x}_i$, must be maximized.

$$m = min_{i=1..n} dist(\mathbf{x}_i, \Delta(\mathbf{v}, a))$$



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

Support Vector Machines

Let $\mathbf{x}$ be a vector in $\mathbb{R}^d$ and $\Delta(\mathbf{v}, a) = \{\mathbf{s} \in \mathbb{R}^d | \mathbf{v}\mathbf{s}^T + a = 0\}$ an hyperplane. The distance between $\mathbf{x}$ and the hyperplan $\Delta(\mathbf{v}, a)$ is

$$dist(\mathbf{x}, \Delta(\mathbf{v}, a)) = \frac{|\mathbf{v}\mathbf{x}^T + a|}{||v||}$$

Let $\mathbf{s}_x$ be the closest point to $\mathbf{x} \in \Delta, \mathbf{s}_x = argmin||\mathbf{x} - \mathbf{s}||$,

$$\mathbf{x} = \mathbf{s}_x + m\frac{\mathbf{v}}{||\mathbf{v}||} \iff \mathbf{x} - \mathbf{s}_x = m\frac{\mathbf{v}}{||\mathbf{v}||}$$

So that, taking the scalar product with vector $\mathbf{v}$ we have:

$$\mathbf{v} \times m\frac{\mathbf{v}^T}{\mathbf{v}} = \mathbf{v}(\mathbf{x}^T - \mathbf{s}_x^T) = \mathbf{v}\mathbf{x}^T - \mathbf{v}\mathbf{s}_x^T = \mathbf{v}\mathbf{x}^T + a - (\mathbf{v}\mathbf{s}_x^T + a) = \mathbf{v}\mathbf{x}^T + a$$

$\mathbf{v}\mathbf{s}_x^T + a = 0$ because $\mathbf{s}_x \in \Delta$

Support Vector Machines

So, therefore

$$\mathbf{v} \times m\frac{\mathbf{v}^T}{||\mathbf{v}||} = m\frac{||\mathbf{v}||^2}{||\mathbf{v}||} = m||\mathbf{v}|| = \mathbf{v}\mathbf{x}^T + a$$

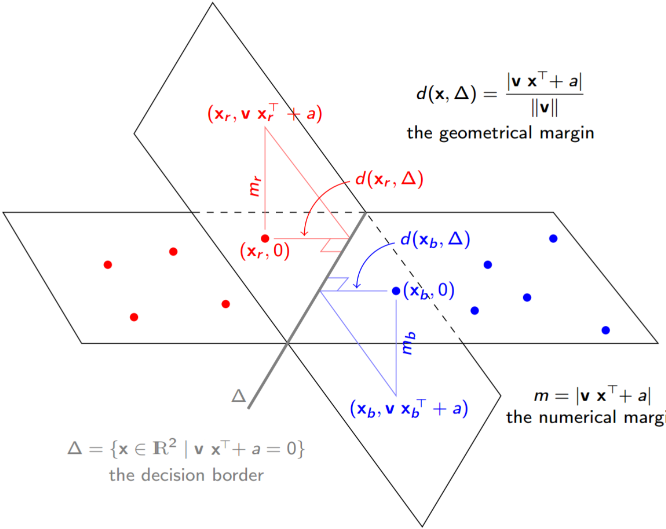
Thus,

$$m = \frac{\mathbf{v}\mathbf{x}^T + a}{||v||}$$

and

$$dist(\mathbf{x}, \Delta(\mathbf{v}, a)) = \frac{|\mathbf{v}\mathbf{x}^T + a|}{||v||}$$

Support Vector Machines



$$d(\mathbf{x}, \Delta) = \frac{|\mathbf{v}\ \mathbf{x}^\top + a|}{\|\mathbf{v}\|}$$

the geometrical margin

$$m = |\mathbf{v}\ \mathbf{x}^\top + a|$$

the numerical margin

$$\Delta = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{v}\ \mathbf{x}^\top + a = 0\}$$

the decision border

Support Vector Machines

The decision hyperplane $\Delta(\mathbf{v}, a) = \{\mathbf{s} \in \mathbb{R}^d | \mathbf{v}\mathbf{s}^T + a = 0\}$, we have to maximize the margin

$$\max_{\mathbf{v},a} \min_{i \in [1..n]} dist(\mathbf{x}_i, \Delta(\mathbf{v}, a))$$

Maximize the confidence

$$\begin{cases} \max_{\mathbf{v},a} m \\ \text{with } \min_{i \in [1..n]} \frac{|\mathbf{v}\mathbf{x}^T + a|}{||v||} \geq m \end{cases} \quad \begin{cases} \max_{\mathbf{v},a} m \\ \text{with } \frac{y_i(\mathbf{v}\mathbf{x}^T + a)}{||v||} \geq m \end{cases}$$

Support Vector Machines

Change variable $\mathbf{w} = \frac{\mathbf{v}}{m||\mathbf{v}||} \implies ||\mathbf{w}|| = \frac{1}{m}$ and $b = \frac{a}{m||\mathbf{v}||}$
Maximize the confidence

$$\begin{cases} \max_{\mathbf{v},a} & m \\ \text{with} & \frac{y_i(\mathbf{v}\mathbf{x}^T + a)}{||v||} \geq m \end{cases}$$

$$\begin{cases} \max_{\mathbf{w},b} m \\ \text{with } y_i(\mathbf{w}\mathbf{x}^T + b) \geq 1; i = 1..n \\ \text{and } m = \frac{1}{||\mathbf{w}||} \end{cases} \qquad \begin{cases} \min_{\mathbf{w},a} ||\mathbf{w}|| \\ \text{with } y_i(\mathbf{w}\mathbf{x}^T + b) \geq 1; i = 1..n \end{cases}$$

Support Vector Machines

Linear SVMs are the solution of the following problem (called primal)

Let $\{(\mathbf{x}_i, y_i); i = 1..n\}$ be a set of labelled data with $\mathbf{x} \in \mathbb{R}^d, y_i \in \{1, -1\}$. A support vector machine is a linear classifier associated with the following decision function: $D(x) = sign(\mathbf{wx}^T + b)$ where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ a given thought the solution of the following problem:

$$\begin{cases} \min_{\mathbf{w} \in \mathbf{R}^d, b \in \mathbb{R}} & \frac{1}{2}||\mathbf{w}||^2 \\ \text{with} & y_i(\mathbf{wx}^T + b) \geq 1; i = 1..n \end{cases}$$

Support Vector Machines

Minimize $\frac{1}{2}||\mathbf{w}||$
subject to $1 - y_i(\mathbf{w}\mathbf{x}^T + b) \leq 0$ for $i = 1..n$
The Lagrangian is:

$$\mathcal{L} = \frac{1}{2}\mathbf{w}\mathbf{w}^T + \sum_{i=1}^{n} \alpha_i(1 - y_i(\mathbf{w}\mathbf{x}^T + b))$$

Note that $||\mathbf{w}|| = \mathbf{w}\mathbf{w}^T$ and $\alpha_I \geq 0$

Support Vector Machines

$$\mathcal{L} = \frac{1}{2}\mathbf{w}\mathbf{w}^T + \sum_{i=1}^{n} \alpha_i (1 - y_i(\mathbf{w}\mathbf{x}^T + b))$$

Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero, we have:

$$\frac{\delta \mathcal{L}}{\delta \mathbf{w}} =$$

$$\frac{\delta \mathcal{L}}{\delta b} =$$

Support Vector Machines

If we substitue $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ to $\mathcal{L}$, we have

$$\mathcal{L} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j^T$$

- Note that $\sum_{i=1}^{n} \alpha_i y_i = 0$
- This is a function of α
- It is known as the dual problem: if we know $\mathbf{w}$, we know all α; or if we know all α, we know $\mathbf{w}$

Support Vector Machines

Thue dual problem is thus:

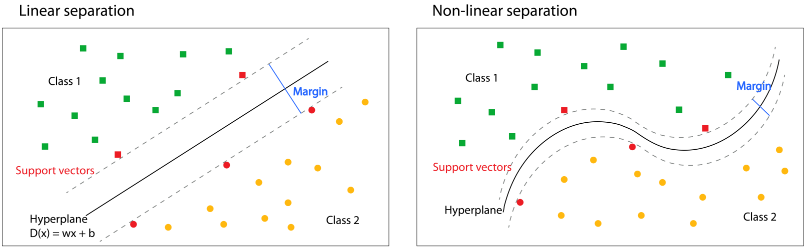
$$max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j^T$$

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

- $\mathbf{x}_i$ with non-zero α_i are called support vectors (SV)
- $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ is linear combination of a small number of data points
- To test new data $\mathbf{z}$
    - compute $\mathbf{w}\mathbf{z}^T + b = \sum_{j=1}^{s} \alpha_j y_j (\mathbf{x}_j \mathbf{z}^T) + b$
    - if the sum is positive, $\mathbf{z}$ is classified as class 1 or class 2 otherwise.

Support Vector Machines

- Up to now, we have only seen large-margin classifier with a linear decision boundary. How to do if our problem is nonlinear?



Support Vector Machines

- To handle non-linearly separable problems, we use *error* ξ

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- ξ_i are slack variables in optimization, $\xi = 0$ if there is no error for $\mathbf{x}_i$

We want to minimize

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

Support Vector Machines

- The dual of this new problem is:

$$max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j^T$$

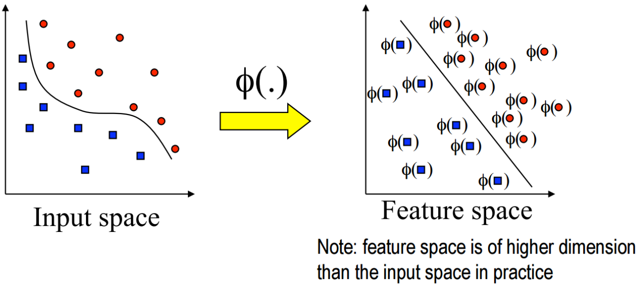
subject to $C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$

- This is similar to the problem in the linear separable case.

Support Vector Machines

- Up to now, we have only seen large-margin classifier with a linear decision boundary. How to do if our problem is nonlinear?
- Idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"

Support Vector Machines



- Computation in the feature space is costly because it is high dimensional
- Kernel functions are used to overcome this.

Support Vector Machines

- The dual of this new problem is:

$$max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j^T$$

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

- We just calculate the inner product without explicite the mapping of original data to the feature space.
- Geometric similarity measures can be expressed by inner products.
- Kernal function is often defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^T$$

Support Vector Machines

Examples of kernel functions

- Linear kernels

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}\mathbf{y}^T$$

- Polynomial kernels with degree d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\mathbf{y}^T + 1)^d$$

- Gaussian kernels

$$K(\mathbf{x}, \mathbf{y}) = e^{\frac{-||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}}$$

Support Vector Machines

Special transformation from $\mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\phi(\mathbf{x}) = \phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\mathbf{x})\phi(\mathbf{y})^T = (\mathbf{x}\mathbf{y}^T)^2$$

Evaluation

Confusion Matrix

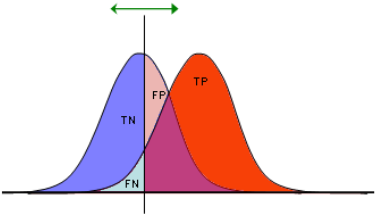
		Prediction outcome		
		Positive	**Negative**	**total**
Actual value	**Positive**	True Positive	False Negative	P′
	Negative	False Positive	True Negative	N′
	total	P	N	

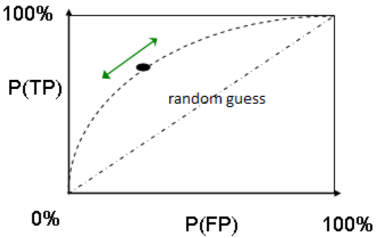
$$Precision = \frac{\mathbf{TP}}{\mathbf{P}}$$

$$Recall = \frac{\mathbf{TP}}{\mathbf{P'}}$$

$$Accuracy = \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{P} + \mathbf{N}}$$

Evaluation





A receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system.

- True positive rate (TPR) vs False positive rate (FPR) at various threshold settings.
- Cost vs benefit analysis of decision making.

Evaluation

To compare different binary classifiers, AUC (Area under ROC curve) becomes a good criterion.