ANALYTIXLABS

Machine Learning: Support Vector Machines (SVM)

What we will cover today

1. **SVM: Basics**
2. **How SVM Works**
   - The Support Vector Classifier (Linear Classifier)
   - The Support Vector Machine Classifier (Non Linear Classifier)
   - Kernels

ANALYTIXLABS

SVM—Support Vector Machines

- SVM is a supervised learning method
- A relatively new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)
- Computationally intensive and usually gives higher accuracy

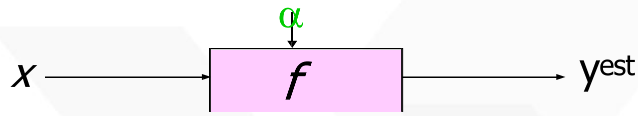
ANALYTIXLABS

SVM—History and Applications

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and regression
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, text mining, benchmarking time-series prediction tests

ANALYTIXLABS

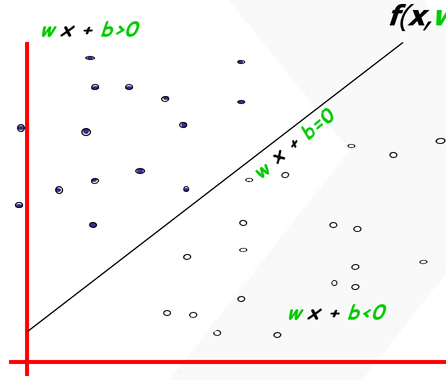
Linear Classifier



• denotes +1

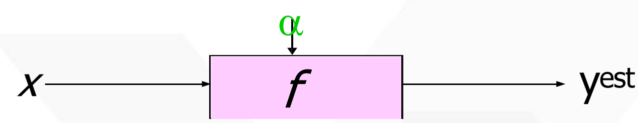
◦ denotes -1

f(**x**,**w**,*b*) = *sign*(**w x** + *b*)



How would you classify this data?

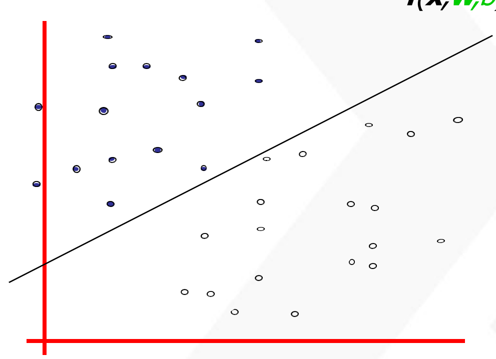
Linear Classifier



• denotes +1

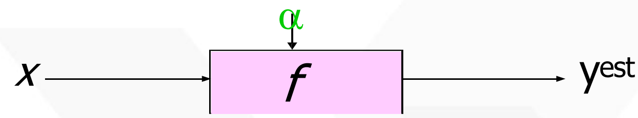
◦ denotes -1

f(**x**,**w**,*b*) = *sign*(**w x** + *b*)



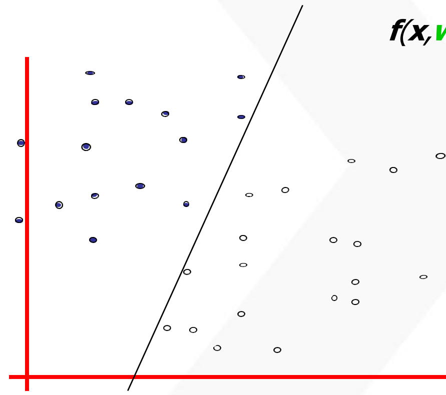
How would you classify this data?

Linear Classifier



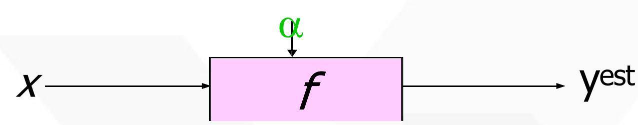
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- denotes +1
- denotes -1



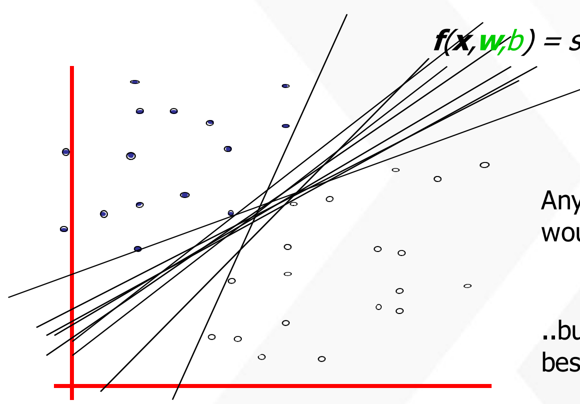
How would you classify this data?

Linear Classifier



$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\ \mathbf{x} + b)$

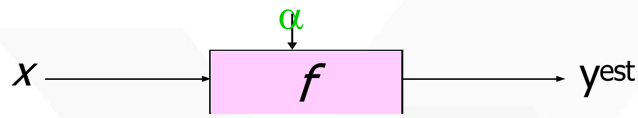
- denotes +1
- denotes -1



Any of these would be fine..

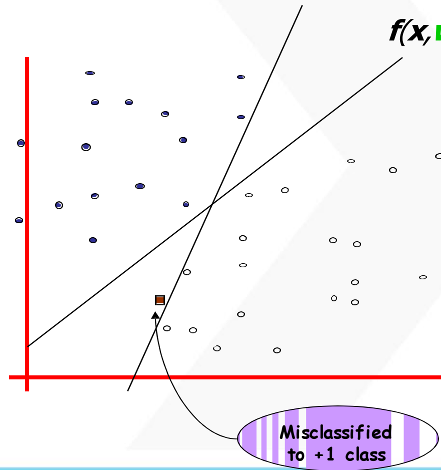
..but which is best?

Linear Classifier



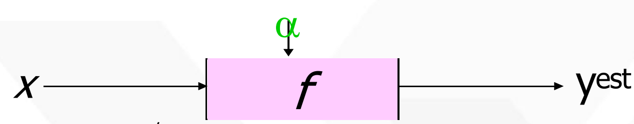
- denotes +1
- denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$



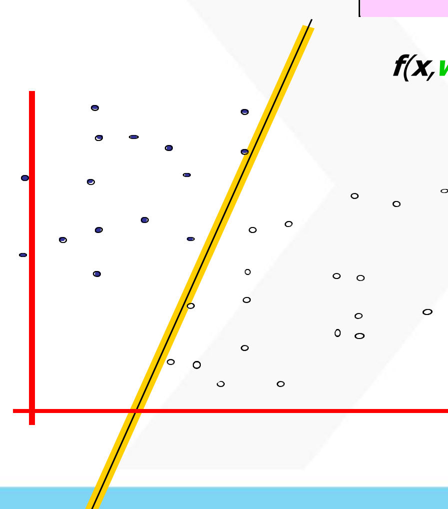
How would you classify this data?

Classifier Margin



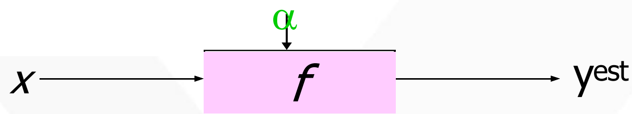
- denotes +1
- denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

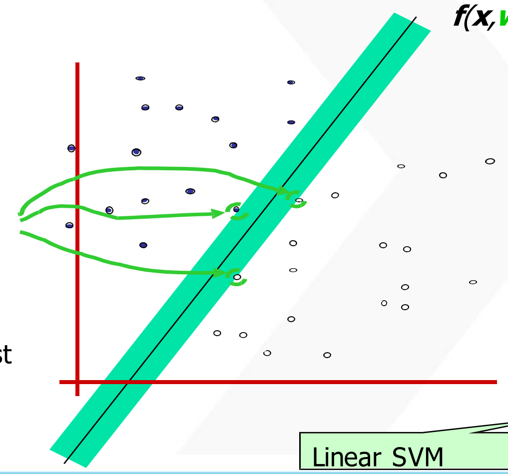
Classifier Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- denotes +1
- denotes -1

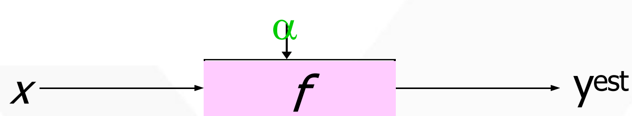
Support Vectors are those datapoints that the margin pushes up against



The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

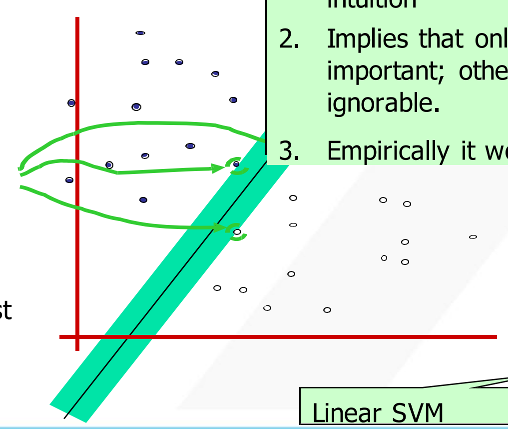
Classifier Margin



- denotes +1
- denotes -1

1. Maximizing the margin is good according to intuition
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against



with the maximum margin.

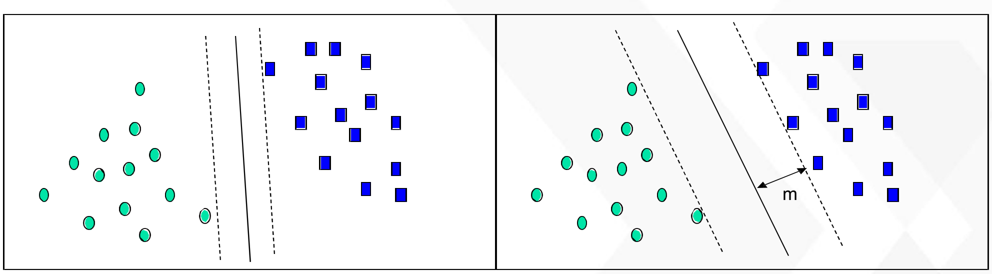
This is the simplest kind of SVM (Called an LSVM)

Separable Hyperplanes

- Imagine a situation where you have a two class classification problem with two predictors X_1 and X_2.
- Suppose that the two classes are "linearly separable" i.e. one can draw a straight line in which all points on one side belong to the first class and points on the other side to the second class.
- Then a natural approach is to find the straight line that gives the biggest separation between the classes i.e. the points are as far from the line as possible
- This is the basic idea of a support vector classifier.

ANALYTIXLABS

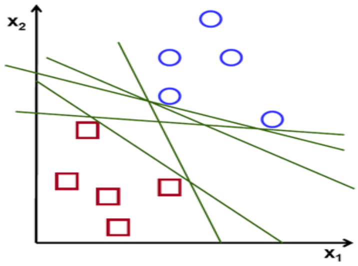
SVM—When Data Is Linearly Separable

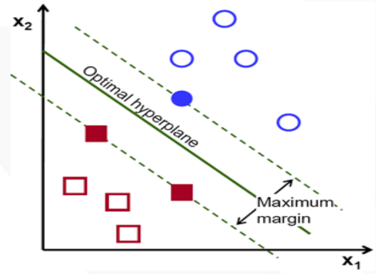


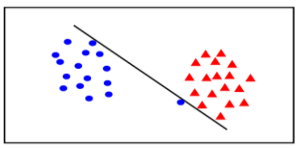
- ✓ Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i
- ✓ There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)
- ✓ SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane (MMH)**

ANALYTIXLABS

Large Margin Intuition Effect

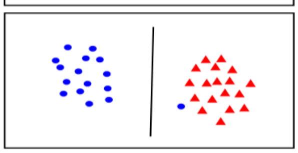






• the points can be linearly separated but there is a very narrow margin

C is large then – overfitting



• but possibly the large margin solution is better, even though one constraint is violated

C is low then – underfitting

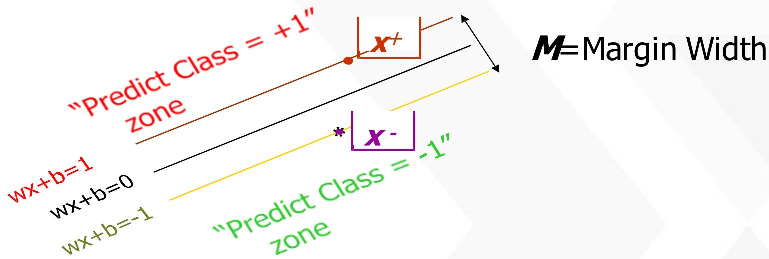
ANALYTIXLABS

SVM—Linearly Separable

- A separating hyperplane can be written as W • X + b = 0
  - where W={w1, w2, …, wn} is a weight vector and b a scalar (bias)
  - For 2-D it can be written as w0 + w1 x1 + w2 x2 = 0
  - The hyperplane defining the sides of the margin:
  - H1: w0 + w1 x1 + w2 x2 ≥ 1 for yi = +1, and
  - H2: w0 + w1 x1 + w2 x2 ≤ – 1 for yi = –1
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints -> *Quadratic Programming (QP) -> Lagrangian multiplier*

ANALYTIXLABS

Linear SVM Mathematically



What we know:

- $w \cdot x^{+} + b = +1$
- $w \cdot x^{-} + b = -1$
- $w \cdot (x^{+} - x^{-}) = 2$

$$M = \frac{(x^{+} - x^{-}) \cdot w}{|w|} = \frac{2}{|w|}$$

ANALYTIXLABS

Linear SVM Mathematically

- Goal: **1) Correctly classify all training data**

$$wx_i + b \geq 1 \quad \textit{if } y_i = +1$$
$$wx_i + b \leq -1 \quad \textit{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all } i$$

2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**
- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \quad \forall i$

ANALYTIXLABS

Solving the Optimization Problem

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **Need to optimize a *quadratic* function subject to *linear* constraints.**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Find $\alpha_1 \dots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

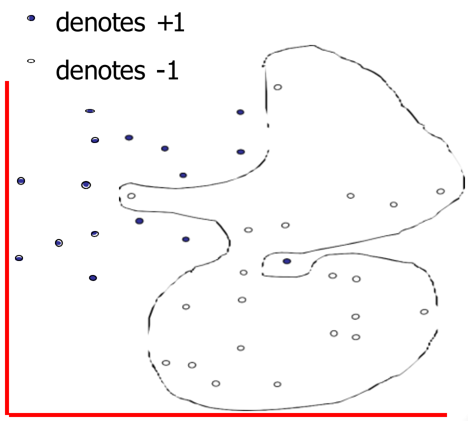
ANALYTIXLABS

Optimizing a Quadratic function with linear Constraints

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

ANALYTIXLABS

Dataset with noise



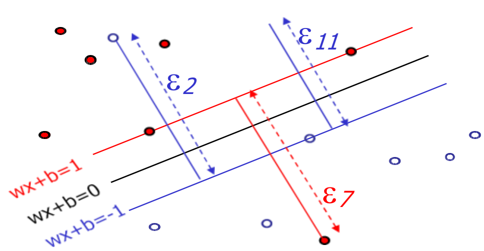
- **Hard Margin:** So far we require all data points be classified correctly
  - No training error
- **What if the training set is noisy?**

OVERFITTING!

- **Solution: use very powerful kernels**

Soft Margin Classification

***Slack variables** ξi* **can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}^T.\mathbf{w} + C \sum_{k=1}^{R} \varepsilon_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **The new formulation incorporating slack variables:**

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter *C* can be viewed as a way to control overfitting.**

ANALYTIXLABS

Linear SVMs: Overview

- **The classifier is a *separating hyperplane.***
- **Most "important" training points are support vectors; they define the hyperplane.**
- **Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.**
- **Both in the dual formulation of the problem and in the solution training points appear only inside dot products:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i^T x_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$

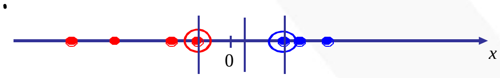
ANALYTIXLABS

Why Is SVM Effective on High Dimensional Data?

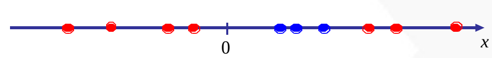
- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM—Linearly Inseparable

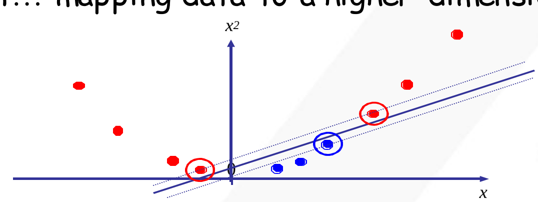
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

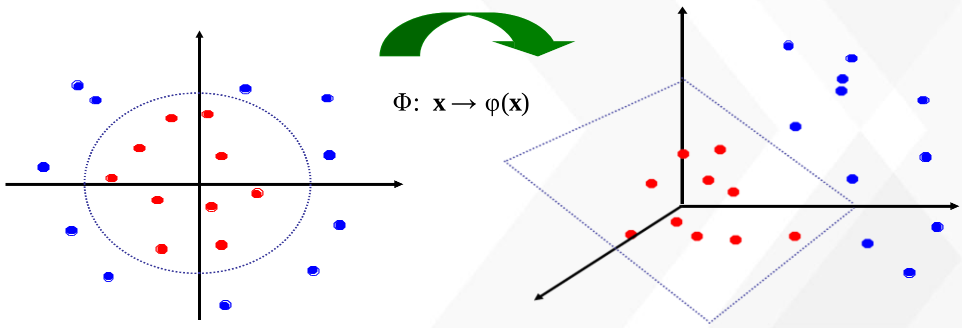


- How about… mapping data to a higher-dimensional space:

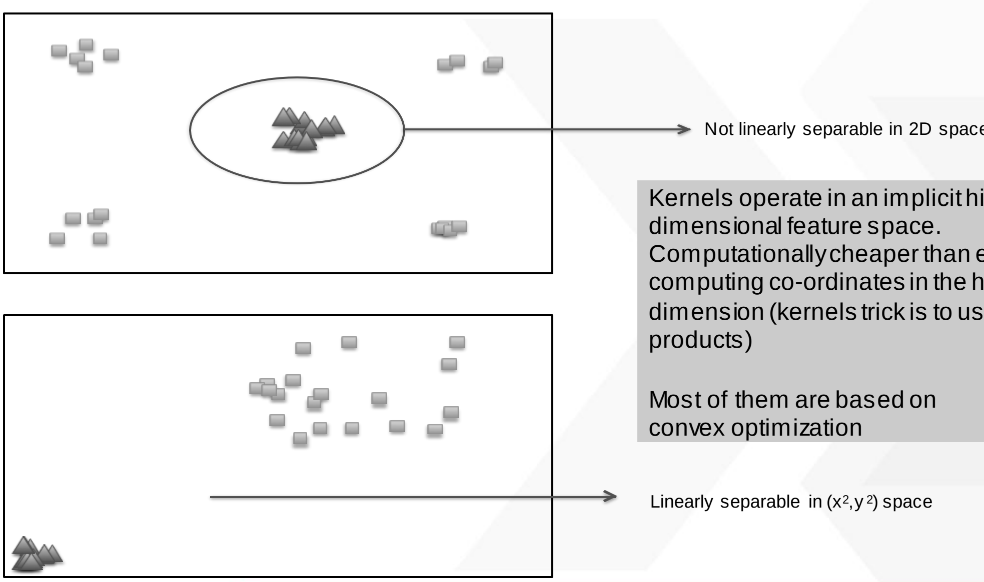


SVM—Linearly Inseparable

- General idea: The original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Kernels - Motivation



Kernels operate in an implicit higher-dimensional feature space. Computationally cheaper than explicitly computing co-ordinates in the higher dimension (kernels trick is to use inner products)

Most of them are based on convex optimization

Non-Linear Classifier: Basis Function

- The support vector classifier is fairly easy to think about. However, because it only allows for a linear decision boundary it may not be all that powerful.
- We extended linear regression to non-linear regression using a basis function i.e.

$$Y_i = \beta_0 + \beta_1 b_1(X_i) + \beta_2 b_2(X_i) + \cdots + \beta_p b_p(X_i) + \varepsilon_i$$

ANALYTIXLABS

Non Linear Classifier: A Basis Approach

- Conceptually, we can take a similar approach with the support vector classifier.
- The support vector classifier finds the optimal hyper-plane in the space spanned by $X_1, X_2, \ldots, X_p$.
- Instead we can create transformations (or a basis) $b_1(x), b_2(x), \ldots, b_M(x)$ and find the optimal hyper-plane in the space spanned by $b_1(\mathbf{X}), b_2(\mathbf{X}), \ldots, b_M(\mathbf{X})$.
- This approach produces a linear plane in the transformed space but a non-linear decision boundary in the original space.
- This is called the Support Vector Machine Classifier.

ANALYTIXLABS

In Reality: Kernal Function=Basis Function

- While conceptually the basis approach is how the support vector machine works, there is some complicated math (which I will spare you) which means that we don't actually choose $b_1(x), b_2(x), \ldots, b_M(x)$.
- Instead we choose something called a Kernel function which takes the place of the basis.
- Common kernel functions include
  - Linear
  - Polynomial
  - Radial Basis
  - Sigmoid

ANALYTIXLABS

Non Linear Classifier: Kernal Function in mathematical way

- **The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \phi(x)$, the dot product becomes:**

  $$K(x_i,x_j)= \phi(x_i)^T\phi(x_j)$$
- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example: 2-dimensional vectors $x=[x_1 \;\; x_2]$; let $K(x_i,x_j)=(1 + x_i^T x_j)^2$, Need to show that $K(x_i,x_j)= \phi(x_i)^T\phi(x_j)$:**

  $$K(x_i,x_j) = (1 + x_i^T x_j)^2,$$

  $$= 1+ x_{i1}^2 x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

  $$= \varphi(x_i)^T\varphi(x_j), \quad \text{where } \varphi(x) = [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]$$

ANALYTIXLABS

What Functions are Kernels?

- For some functions $K(x_i,x_j)$ checking that

  $K(x_i,x_j)= \phi(x_i)^T\phi(x_j)$ can be cumbersome.
- Mercer's theorem: Every semi-positive definite symmetric function is a kernel
- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_N)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_N)$
…	…	…	…	…
$K(\mathbf{x}_N,\mathbf{x}_1)$	$K(\mathbf{x}_N,\mathbf{x}_2)$	$K(\mathbf{x}_N,\mathbf{x}_3)$	…	$K(\mathbf{x}_N,\mathbf{x}_N)$

ANALYTIXLABS

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x}_i,\mathbf{x}_j)= \tanh(\beta_0\mathbf{x}_i^T\mathbf{x}_j + \beta_1)$

ANALYTIXLABS

SVMs Training with Kernels

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

- Now, we minimize using f as the feature vector instead of x
- By solving this minimization problem you get the parameters for your SVM

- Linear Kernel (Means no kernel)
  - Appropriate when high dimensional features and few examples
  - Avoid overfitting

- Gaussian Kernel
  - When low dimensional features and N is large
  - Variable normalization (Feature Scaling)

- Polynomial Kernel, String Kernel, Chi-squared Kernel, Histogram intersection Kernel

ANALYTIXLABS

Nonlinear SVM – Overview Summary

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

ANALYTIXLABS

SVM (Good to Know Things)

- If n (features) is large vs. m (training set)
- e.g. text classification problem
  - Feature vector dimension is 10 000
  - Training set is 10 - 1000
  - Then use logistic regression or SVM with a linear kernel
- If n is small and m is intermediate
  - n = 1 - 1000
  - m = 10 - 10 000
- Gaussian kernel is good
- If n is small and m is large
  - n = 1 - 1000
  - m = 50 000+
- SVM will be slow to run with Gaussian kernel
- In that case
  - Manually create or add more features
  - Use logistic regression of SVM with a linear kernel
- Logistic regression and SVM with a linear kernel are pretty similar
- SVM has a convex optimization problem - so you get a global minimum

ANALYTIXLABS

Weakness of SVM

- **It is sensitive to noise**

  - A relatively small number of mislabeled examples can dramatically decrease the performance
- **It only considers two classes**

  - how to do multi-class classification with SVM?

  - Answer:

  1) with output Classes m, learn m SVM's
  - **SVM 1 learns "Output==1" vs "Output != 1"**
  - **SVM 2 learns "Output==2" vs "Output != 2"**
  - **:**
  - **SVM m learns "Output==m" vs "Output != m"**

  (This strategy for prediction of multi-class problems using binary classifiers is known as One-against-all)

  2) To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

ANALYTIXLABS

Some Issues

- **Choice of kernel**
  - **Gaussian or polynomial kernel is default**
  - **if ineffective, more elaborated kernels are needed**
  - **domain experts can give assistance in formulating appropriate similarity measures**

- **Choice of kernel parameters**
  - **e.g. σ in Gaussian kernel**
  - **σ is the distance between closest points with different classifications**
  - **In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.**

- **Optimization criterion** – Hard margin v.s. Soft margin
  - **a lengthy series of experiments in which various parameters are tested**

ANALYTIXLABS

SVM vs. Neural Network

- Neural Network
  - Relatively old
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (not that trivial)

- SVM
  - Relatively new concept
  - Deterministic algorithm
  - Nice Generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

ANALYTIXLABS