Principles of Machine Learning
Kernel Machines

Vasant Honavar
Artificial Intelligence Research Laboratory
College of Information Sciences and Technology
Bioinformatics and Genomics Graduate Program
The Huck Institutes of the Life Sciences
Pennsylvania State University

vhonavar@ist.psu.edu
http://vhonavar.ist.psu.edu
http://faculty.ist.psu.edu/vhonavar

Maximum margin classifiers

- Discriminative classifiers that maximize the margin of separation
- Support Vector Machines
  - Feature spaces
  - Kernel machines
  - VC Theory and generalization bounds
  - Maximum margin classifiers
- Comparisons with conditionally trained discriminative classifiers

Perceptrons revisited

Perceptrons

- Can only compute threshold functions
- Can only represent linear decision surfaces
- Can only learn to classify linearly separable training data

How can we deal with non linearly separable data?

- Map data into a typically higher dimensional feature space where the classes become separable

Two problems must be solved:

- Computational problem of working with high dimensional feature space
- Overfitting problem in high dimensional feature spaces

Maximum margin model

We cannot outperform Bayes optimal classifier when

- The generative model assumed is correct
- The data set is large enough to ensure reliable estimation of parameters of the models

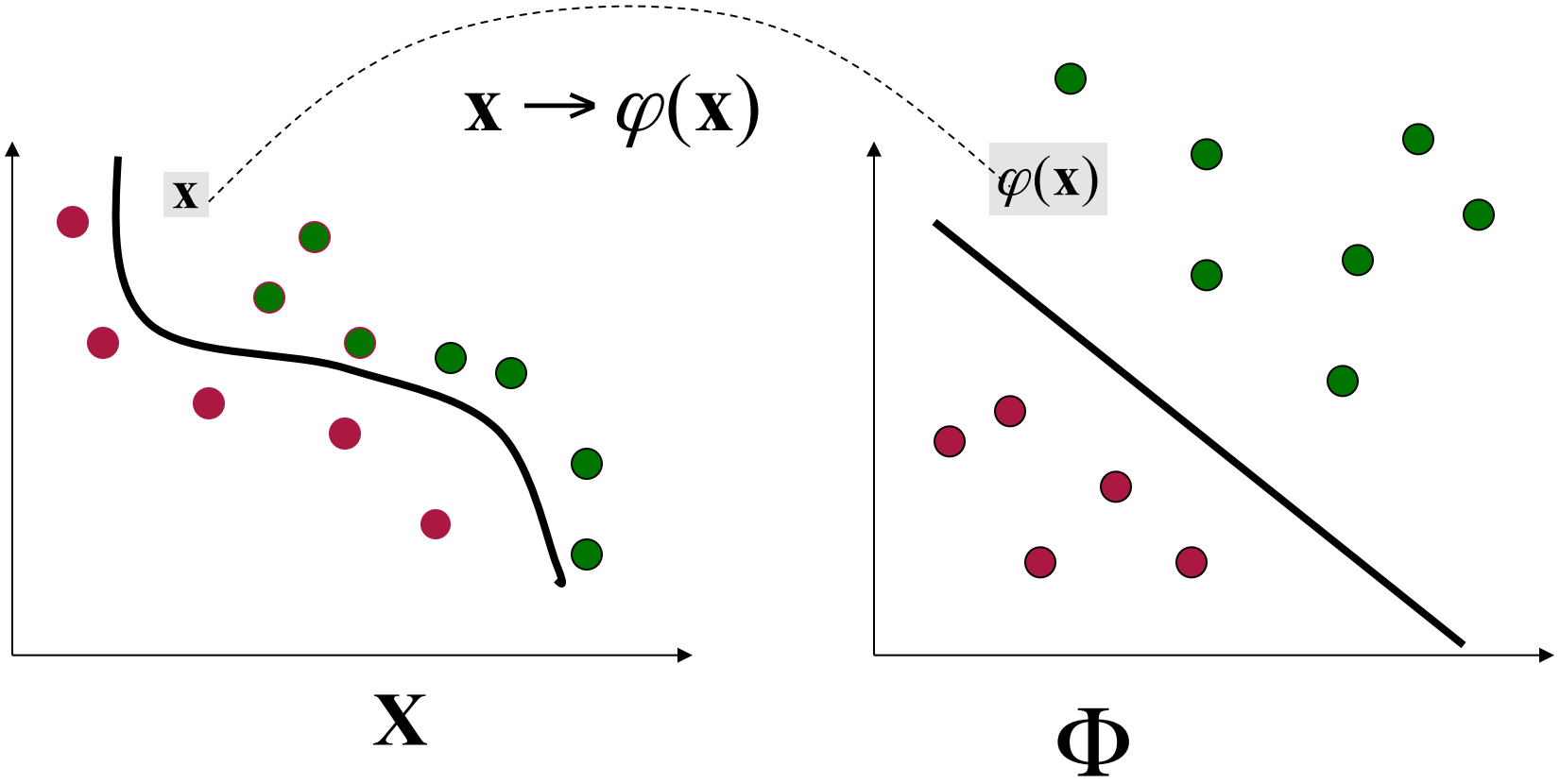
But discriminative models may be better than generative models when

- The correct generative model is seldom known
- The data set is often simply not large enough

Maximum margin classifiers are a kind of discriminative classifiers designed to circumvent the overfiting problem

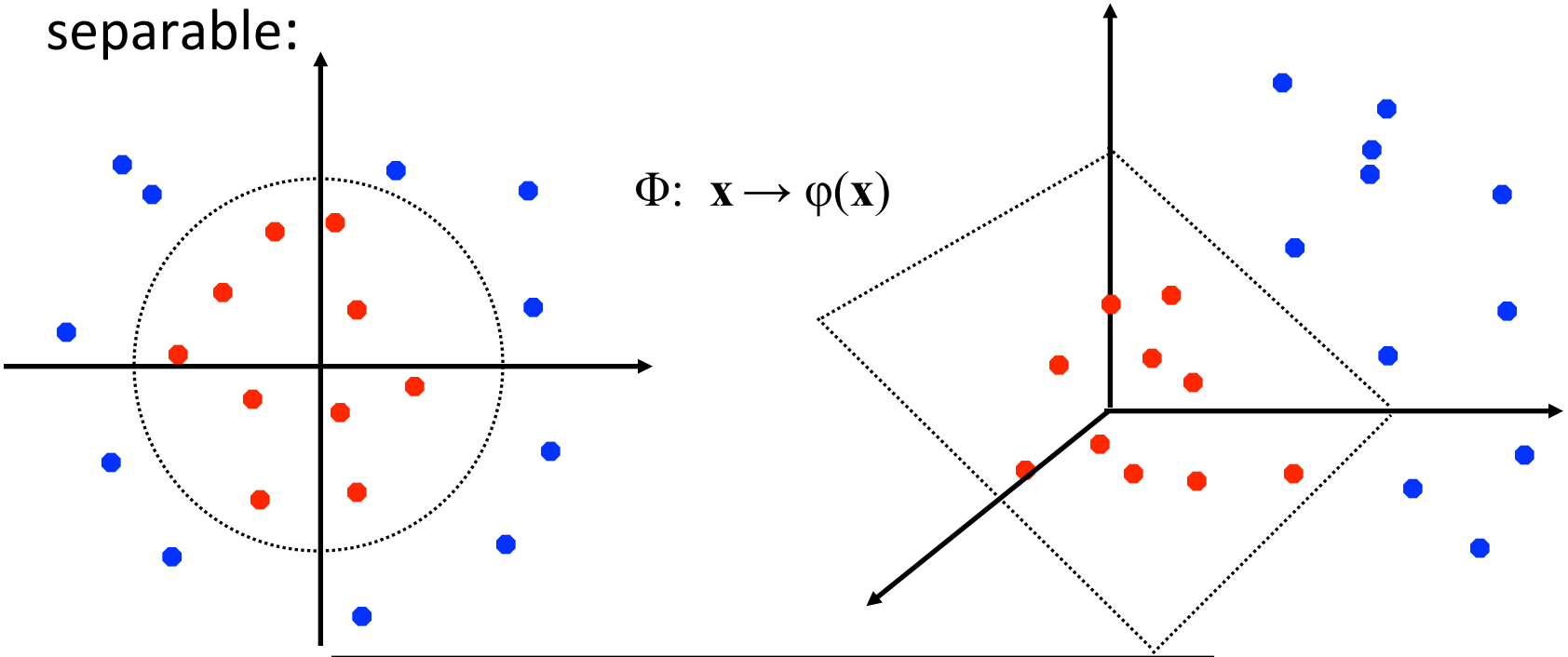
Extending Linear Classifiers – Feature spaces

Map data into a feature space where they are linearly separable



Linear Separability in Feature spaces

The original input space can always be mapped to some higher-dimensional feature space where the training data become separable:



$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

Learning in the Feature Spaces

High dimensional Feature spaces

$$\mathbf{x} = (x_1, x_2 \ldots . x_n) \rightarrow \varphi(\mathbf{x}) = (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots . \varphi_d(\mathbf{x}))$$

where typically $d >> n$ solve the problem of expressing complex functions

But this introduces a

- computational problem (working with very large vectors)
- generalization problem (curse of dimensionality)

SVM offer an elegant solution to both problems

The Perceptron Algorithm Revisited

- The perceptron works by adding misclassified positive or subtracting misclassified negative examples to an arbitrary weight vector, which (without loss of generality) we assumed to be the zero vector.
- The final weight vector is a linear combination of training points

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i,$$

where, since the sign of the coefficient of $\mathbf{x}_i$ is given by label y_i, the α_i are positive values, proportional to the number of times, misclassification of $\mathbf{x}_i$ has caused the weight to be updated. It is called the embedding strength of the pattern $\mathbf{x}_i$.

-

Dual Representation

The decision function can be rewritten as:

$$h(\mathbf{x}) = \text{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) = \text{sgn}\left(\left\langle \left(\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \right), \mathbf{x} \right\rangle + b\right)$$

$$= \text{sgn}\left(\sum_{j=1}^{l} \alpha_j y_j \langle \mathbf{x}_j, \mathbf{x} \rangle + b \right)$$

The update rule is:

on training example $(\mathbf{x}_i, y_i)$

if $y_i \left(\sum_{j=1}^{l} \alpha_j y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle + b \right) \leq 0,$

then $\alpha_i \leftarrow \alpha_i + \eta$

WLOG, we can take $\eta = 1.$

Implications of Dual Representation

When Linear Learning Machines are represented in the dual form

$$h(\mathbf{x}_i) = \text{sgn}\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) = \text{sgn}\left(\sum_{j=1}^{l} \alpha_j y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle + b\right)$$

Data appear only inside dot products (in decision function and in training algorithm)

The matrix $G = \left(\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle\right)_{i,j=1}^{l}$ which is the matrix of pair-wise dot products between training samples is called the Gram matrix

Implicit Mapping to Feature Space

Kernel Machines

- Solve the computational problem of working with many dimensions
- Can make it possible to use infinite dimensions efficiently
- Offer other advantages, both practical and conceptual

Kernel-Induced Feature Spaces

$$f(\mathbf{x}_i) = \langle \mathbf{w}, \varphi(\mathbf{x}_i) \rangle + b$$
$$h(\mathbf{x}_i) = \mathrm{sgn}(f(\mathbf{x}_i))$$

where $\varphi : X \rightarrow \Phi$ is a non-linear map from input space to feature space

In the dual representation, the data points only appear inside dot products

$$h(\mathbf{x}_i) = \mathrm{sgn}\left(\langle \mathbf{w}, \varphi(\mathbf{x}_i) \rangle + b\right) = \mathrm{sgn}\left(\sum_{j=1}^{l} \alpha_j y_j \langle \varphi(\mathbf{x}_j), \varphi(\mathbf{x}_i) \rangle + b\right)$$

Kernels

- Kernel function returns the value of the dot product between the images of the two arguments

$$K(x_1, x_2) = \langle \varphi(x_1), \varphi(x_2) \rangle$$

- When using kernels, the dimensionality of the feature space Φ is not necessarily important because of the special properties of kernel functions

- Given a function *K*, it is possible to verify that it is a kernel (we will return to this later)

Kernel Machines

We can use perceptron learning algorithm in the feature space by taking its dual representation and replacing dot products with kernels:

$$\langle x_1, x_2 \rangle \leftarrow K(x_1, x_2) = \langle \varphi(x_1), \varphi(x_2) \rangle$$

Example: Polynomial Kernels

$$x = (x_1, x_2);$$
$$z = (z_1, z_2);$$

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2$$
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2$$
$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle$$
$$= \langle \varphi(x), \varphi(z) \rangle$$

The Kernel Matrix

Kernel matrix is the Gram matrix in the feature space Φ

(the matrix of pair-wise dot products between feature vectors corresponding to the training samples)

$\mathbf{K}=$

$K(1,1)$	$K(1,2)$	$K(1,3)$	…	$K(1,l)$
$K(2,1)$	$K(2,2)$	$K(2,3)$	…	$K(2,l)$
…	…	…	…	…
$K(l,1)$	$K(l,2)$	$K(l,3)$	…	$K(l,l)$

Properties of Kernel Matrices

It is easy to show that the Gram matrix (and hence the kernel matrix) is

- A Square matrix
- Symmetric ($\mathbf{K}^T = \mathbf{K}$)
- Positive semi-definite (all eigenvalues of **K** are non-negative – Recall that Eigen values of a square matrix **A** are given by values of λ that satisfy $|\mathbf{A}-\lambda\mathbf{I}|=0$)

Any symmetric positive semi definite matrix can be regarded as a kernel matrix, that is, as an inner product matrix in *some* feature space Φ

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle$$

Mercer's Theorem: Characterization of Kernel Functions

A function $K : X \times X \rightarrow \Re$ is said to be (finitely) positive semi-definite if

- K is a symmetric function: $K(\mathbf{x},\mathbf{y}) = K(\mathbf{y},\mathbf{x})$
- Matrices formed by restricting K to any finite subset of the space X are positive semi-definite

Characterization of Kernel Functions

Every (finitely) positive semi definite, symmetric function is a kernel: i.e. there exists a mapping φ such that it is possible to write:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle$$

Properties of Kernel Functions - Mercer's Theorem

Eigenvalues of the Gram Matrix define an expansion of Kernels:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle = \sum_l \lambda_l \varphi_l(\mathbf{x}_i) \varphi_l(\mathbf{x}_j)$$

That is, the eigenvalues act as features!

Recall that the eigenvalues of a square matrix A correspond to solutions of

$$|A - \lambda I| = 0 \text{ where}$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Examples of Kernels

Simple examples of kernels:

$$K(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z} \rangle^d$$

$$K(\mathbf{x},\mathbf{z}) = e^{-\left(\frac{\langle \mathbf{x},\mathbf{z} \rangle}{\sigma}\right)}$$

Example: Polynomial Kernels

$$x = (x_1, x_2);$$
$$z = (z_1, z_2);$$

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2$$
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2$$
$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle$$
$$= \langle \varphi(x), \varphi(z) \rangle$$

Making Kernels – Closure Properties

The set of kernels is closed under some operations. If K_1 , K_2 are kernels over X×X, then the following are kernels:

$$K(\mathbf{x},\mathbf{z}) = K_1(\mathbf{x},\mathbf{z}) + K_2(\mathbf{x},\mathbf{z})$$

$$K(\mathbf{x},\mathbf{z}) = aK_1(\mathbf{x},\mathbf{z}); \quad a > 0$$

$$K(\mathbf{x},\mathbf{z}) = K_1(\mathbf{x},\mathbf{z})K_2(\mathbf{x},\mathbf{z})$$

$$K(\mathbf{x},\mathbf{z}) = f(\mathbf{x})f(\mathbf{z}); \quad f : \mathcal{X} \to \Re$$

$$K(\mathbf{x},\mathbf{z}) = K_3(\varphi(\mathbf{x}),\varphi(\mathbf{z}));$$

$$\left(\varphi : \mathcal{X} \to \Re^N; K_3 \text{ is a kernel over } \Re^N \times \Re^N\right)$$

$$K(\mathbf{x},\mathbf{z}) = \mathbf{x}^T\mathbf{B}\mathbf{z};$$

$$\left(\mathbf{B} \text{ is a symmetric positive definite } n \times n \text{ matrix and } \mathcal{X} \subseteq \Re^n\right)$$

We can make complex kernels from simple ones: modularity!

Kernels

- We can define Kernels over arbitrary instance spaces including
  - Finite dimensional vector spaces,
  - Boolean spaces
  - $\sum^*$ where $\sum$ is a finite alphabet
  - Documents, graphs, etc.
- Kernels need not always be expressed by a closed form formula.
- Many useful kernels can be computed by complex algorithms (e.g., diffusion kernels over graphs).

Kernels on Sets and Multi-Sets

Let $X \subseteq 2^V$ for some fixed domain V

Let $A_1, A_2 \in X$

Then $K(A_1, A_2) = 2^{|A_1 \cap A_2|}$ is a kernel.

Exercise: Define a Kernel on the space of multi-sets whose elements are drawn from a finite domain *V*

String Kernel (p-spectrum Kernel)

- The p-spectrum of a string is the histogram – vector of number of occurrences of all possible contiguous substrings – of length p
- We can define a kernel function $K(s, t)$ over $\sum^* \times \sum^*$ as the inner product of the p-spectra of s and t.

$$s = statistics$$

$$t = computation$$

$$p = 3$$

$$\text{Common substrings}: \ tat, ati$$

$$K(s,t) = 2$$

Kernel Machines

Kernel Machines are Linear Learning Machines that :

- Use a dual representation
- Operate in a kernel induced feature space (that is a linear function in the feature space implicitly defined by the gram matrix corresponding to the data set *K*)

$$h(\mathbf{x}_i) = \mathrm{sgn}\left(\langle \mathbf{w}, \varphi(\mathbf{x}_i)\rangle + b\right) = \mathrm{sgn}\left(\sum_{j=1}^{l} \alpha_j y_j \langle \varphi(\mathbf{x}_j), \varphi(\mathbf{x}_i)\rangle + b\right)$$

Kernels – the good, the bad, and the ugly

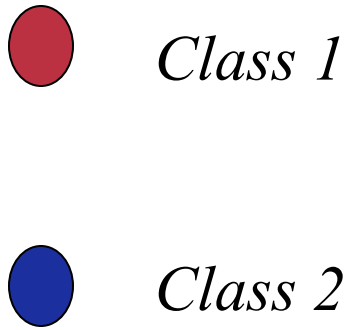
Bad kernel – A kernel whose Gram (kernel) matrix is mostly diagonal -- all data points are orthogonal to each other, and hence the machine is unable to detect hidden structure in the data

1	0	0	…	0
0	1	0	…	0
		1		
…	…	…	…	…
0	0	0	…	1

Kernels – the good, the bad, and the ugly

Good kernel – Corresponds to a Gram (kernel) matrix in which subsets of data points belonging to the same class are similar to each other, and hence the machine can detect hidden structure in the data

3	2	0	0	0
2	3	0	0	0
0	0	4	3	3
0	0	3	4	2
0	0	3	2	4



Kernels – the good, the bad, and the ugly

- In mapping in a space with too many irrelevant features, kernel matrix becomes diagonal
- Need some prior knowledge of target so choose a good kernel

Learning in the Feature Spaces

High dimensional Feature spaces

$$\mathbf{x} = (x_1, x_2 \ldots . x_n) \rightarrow \varphi(\mathbf{x}) = (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots . \varphi_d(\mathbf{x}))$$

where typically $d >> n$ solve the problem of expressing complex functions

But this introduces a

- computational problem (working with very large vectors) – solved by the kernel trick – implicit computation of dot products in kernel induced feature spaces via dot products in the input space
- generalization problem (curse of dimensionality)

The Generalization Problem

- The curse of dimensionality
  - It is easy to overfit in high dimensional spaces
- The Learning problem is ill posed
  - There are infinitely many hyperplanes that separate the training data
  - Need a principled approach to choose an optimal hyperplane

The Generalization Problem

- "Capacity" of the machine – ability to learn any training set without error – related to VC dimension
- Excellent memory is not an asset when it comes to learning from limited data

"A machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree"

C. Burges

History of Key Developments leading to SVM

1958 Perceptron (Rosenblatt)

1963 Margin (Vapnik)

1964 Kernel Trick (Aizerman)

1965 Optimization formulation (Mangasarian)

1971 Kernels (Wahba)

1992-1994 SVM (Vapnik)

1996 – present Rapid growth, numerous applications

Extensions to other problems

A Little Learning Theory

Suppose:

- We are given *l* training examples $(\mathbf{x}_i, y_i)$.
- Train and test points drawn randomly (i.i.d) from some unknown probability distribution $D(\boldsymbol{x}, y)$

- The machine learns the mapping $\mathbf{x}_i \rightarrow y_i$ and outputs a hypothesis $h(\mathbf{x}, \boldsymbol{\alpha}, b)$
- A particular choice of $(\boldsymbol{\alpha}, b)$ specifies "trained machine"
- The expectation of the test error or expected risk is

$$R(\boldsymbol{\alpha}, b) = \frac{1}{2}\int |y - h(\mathbf{x}, \boldsymbol{\alpha}, b)| dD(\mathbf{x}, y)$$

A Bound on the Generalization Performance

The empirical risk is:

$$R_{emp}(\boldsymbol{\alpha}, b) = \frac{1}{2l}\sum_{i=1}^{l}\left|y_i - h(\mathbf{x}, \boldsymbol{\alpha}, b)\right|$$

Choose some δ such that $0 < \delta < 1$ With probability $1-\delta$ the following bound – risk bound of $h(x,a)$ for distribution D – holds (Vapnik,1995):

$$R(\boldsymbol{\alpha}, b) \leq R_{emp}(\boldsymbol{\alpha}, b) + \sqrt{\left(\frac{d\left(\log\left(\frac{2l}{d}\right)+1\right) - \log\left(\frac{\delta}{4}\right)}{l}\right)}$$

where $d \geq 0$ is called VC dimension is a measure of "capacity" of machine.

A Bound on the Generalization Performance

The second term in the right-hand side is called *VC confidence*.

Three key points about the actual risk bound:

- It is independent of $D(\boldsymbol{x},y)$
- It is usually not possible to compute the left hand side.
- If we know d, we can compute the right hand side.

The risk bound gives us a way to compare learning machines!

The VC Dimension

Definition: the VC dimension of a set of functions $H = \{h(\mathbf{x},\boldsymbol{\alpha},b)\}$ is *d* if and only if there exists a set of points *d* data points that is shattered by *H*

That is, each of the 2^d possible labelings (dichotomies) of the *d* data points, can be realized using some member of *H* but that no set of size d+1 or greater satisfying this property exists.

The VC Dimension

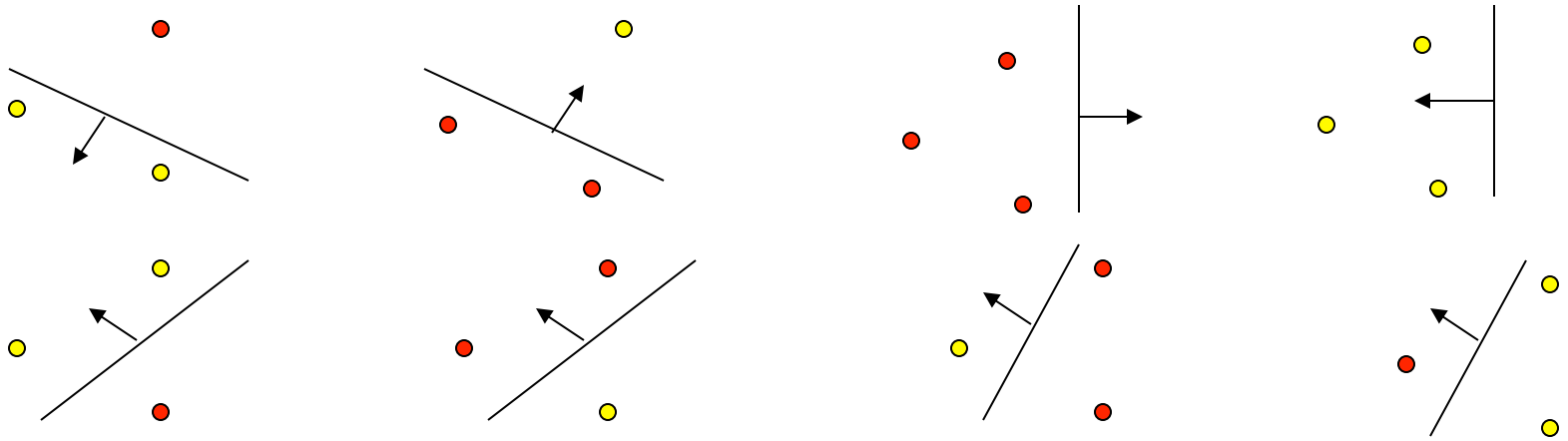
VC dimension of *H* is size of largest subset of *X* shattered by *H*. VC dimension measures the capacity of a set *H* of hypotheses (functions).

If for any number *N*, it is possible to find *N* points

$\mathbf{x}_1, .., \mathbf{x}_N$ that can be separated in all the 2^N possible ways, we will say that the VC-dimension of the set is infinite

The VC Dimension Example

- Suppose that the data live in $\mathbf{R}^2$ space, and the set $\{h(\mathbf{x}, \alpha)\}$ consists of oriented straight lines, (linear discriminants).
- It is possible to find three points that can be shattered by this set of functions
- It is not possible to find four.
- So the VC dimension of the set of linear discriminants in $\mathbf{R}^2$ is three.



The VC Dimension of Hyperplanes

Theorem 1 Consider some set of *m* points in $\mathbf{R}^n$. Choose any one of the points as origin. Then the *m* points can be shattered by oriented hyperplanes if and only if the position vectors of the remaining points are linearly independent.

Corollary: The VC dimension of the set of oriented hyperplanes in $\mathbf{R}^n$ is *n*+1, since we can always choose *n*+1 points, and then choose one of the points as origin, such that the position vectors of the remaining points are linearly independent, but can never choose *n*+2 points

VC Dimension

VC dimension can be infinite even when the number of parameters of the set $\{h(\mathbf{x},\alpha)\}$ of hypothesis functions is very small.

Example: $h(\mathbf{x},\alpha) \equiv \text{sgn}(\sin(\alpha \mathbf{x})), \quad \mathbf{x},\alpha \in \mathbf{R}$

For any integer l with any labels $y_1,\ldots,y_l, \quad y_i \in \{-1,1\}$

we can find l points $x_1,\ldots,x_l$ and parameter α such that those points can be shattered by $h(\mathbf{x},\alpha)$

Those points are: $x_i = 10^{-i}, \quad i = 1,\ldots,l.$

and parameter α is: $\alpha = \pi(1 + \sum_{i=1}^{l} \frac{(1-y_i)10^i}{2})$

Vapnik-Chervonenkis (VC) Dimension

- Let H be a hypothesis class over an instance space X.
- Both H and X may be infinite.
- We need a way to describe the behavior of H on a finite set of points $S \subseteq X$.

$$S = \{X_1, X_2 \ldots . X_m\}$$

- For any concept class H over X, and any $S \subseteq X$,

$$\Pi_H(S) = \{h \cap S : h \in H\}$$

- Equivalently, with a little abuse of notation, we can write

$$\Pi_H(S) = \{(h(X_1) \ldots\ldots h(X_m)) : h \in H\}$$

$\Pi_H(S)$ is the set of all dichotomies or behaviors on S that are induced or realized by H

Vapnik-Chervonenkis (VC) Dimension

- If $\Pi_H(S) = \{0,1\}^m$ where $|S| = m$, or equivalently, $|\Pi_H(S)| = 2^m$
- we say that S is shattered by H.

- A set S of instances is said to be *shattered* by a hypothesis class H if and only if for *every* dichotomy of S, there exists a hypothesis in H that is consistent with the dichotomy.

VC Dimension of a hypothesis class

Definition: The VC-dimension $V(H)$, of a hypothesis class *H* defined over an instance space *X* is the cardinality *d* of the largest subset of *X* that is shattered by *H*. If arbitrarily large finite subsets of *X* can be shattered by *H*, $V(H)=\infty$

How can we show that $V(H)$ is at least *d*?

- Find a set of cardinality at least *d* that is shattered by *H.*

How can we show that $V(H) = d$?

- Show that $V(H)$ is at least *d* and no set of cardinality $(d+1)$ can be shattered by H.

VC Dimension of a Hypothesis Class - Examples

- Example: Let the instance space *X* be the 2-dimensional Euclidian space. Let the hypothesis space H be the set of axis parallel rectangles in the plane.

- $V(H)$=4 (there exists a set of 4 points that can be shattered by a set of axis parallel rectangles but no set of 5 points can be shattered by H).

Some Useful Properties of VC Dimension

$$(H_1 \subseteq H_2) \Rightarrow V(H_1) \leq V(H_2)$$

If H is a finite concept class, $V(H) \leq \lg|H|$

$$\left(\overline{H} = \{X - h : h \in H\}\right) \Rightarrow V(H) = V\left(\overline{H}\right)$$

$$(H = H_1 \cup H_2) \Rightarrow V(H) \leq V(H_1) + V(H_2) + 1$$

If H_l is formed by a union or intersection of l concepts from H, $V(H_l) = O(V(H) l \lg l)$

If $VH = d, \Pi_H(m) = \max\{|\Pi_H(S)| : |S| = m\}$,

$\Pi_H(m) \leq \Phi_d(m)$ where

$\Phi_d(m) = 2^m$ if $m \geq d$ and $\Phi_d(m) = O(m^d)$ if $m < d$

Proof: Left as an exercise

Risk Bound

What is VC dimension and empirical risk of the nearest neighbor classifier?

Any number of points, labeled arbitrarily, can be shattered by a nearest-neighbor classifier, thus $d = \infty$ and empirical risk =0 .

So the bound provide no useful information in this case.

Minimizing the Bound by Minimizing d

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{d\left(\log\left(\frac{2l}{d} \right) + 1 \right) - \log\left(\frac{\delta}{4} \right)}{l} \right)}$$

- VC confidence (second term) depends on d/l
- One should choose that learning machine with minimal d
- For large values of d/l the bound is not tight

Bounds on Error of Classification

Vapnik proved that the error ε of of classification function *h* for separable data sets is

$$\varepsilon = O\left(\frac{d}{l}\right)$$

where *d* is the VC dimension of the hypothesis class and *l* is the number of training examples

The classification error

- depends on the VC dimension of the hypothesis class
- is independent of the dimensionality of the feature space

Structural Risk Minimization

Finding a learning machine with the minimum upper bound on the actual risk

- leads us to a method of choosing an optimal machine for a given task
- essential idea of the structural risk minimization (SRM).

Let $H_1 \subset H_2 \subset H_3 \subset ..$ be a sequence of nested subsets of hypotheses whose VC dimensions satisfy $d_1 < d_2 < d_3 < \ldots$

- SRM involves finding that subset of functions which minimizes the upper bound on the actual risk
- SRM involves training a series of classifiers, and choosing a classifier from the series for which the sum of empirical risk and VC confidence is minimal.

Margin Based Bounds on Error of Classification

Margin based bound

$$\varepsilon = O\left(\frac{1}{l}\left(\frac{L}{\bar{\gamma}}\right)^2\right)$$

$$L = \max_p \|\mathbf{X}_p\|$$

$$\bar{\gamma} = \min_i \frac{y_i f(\mathbf{x}_i)}{\|\mathbf{w}\|}$$

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

Important insight

Error of the classifier trained on a separable data set is inversely proportional to its margin, and is independent of the dimensionality of the input space!

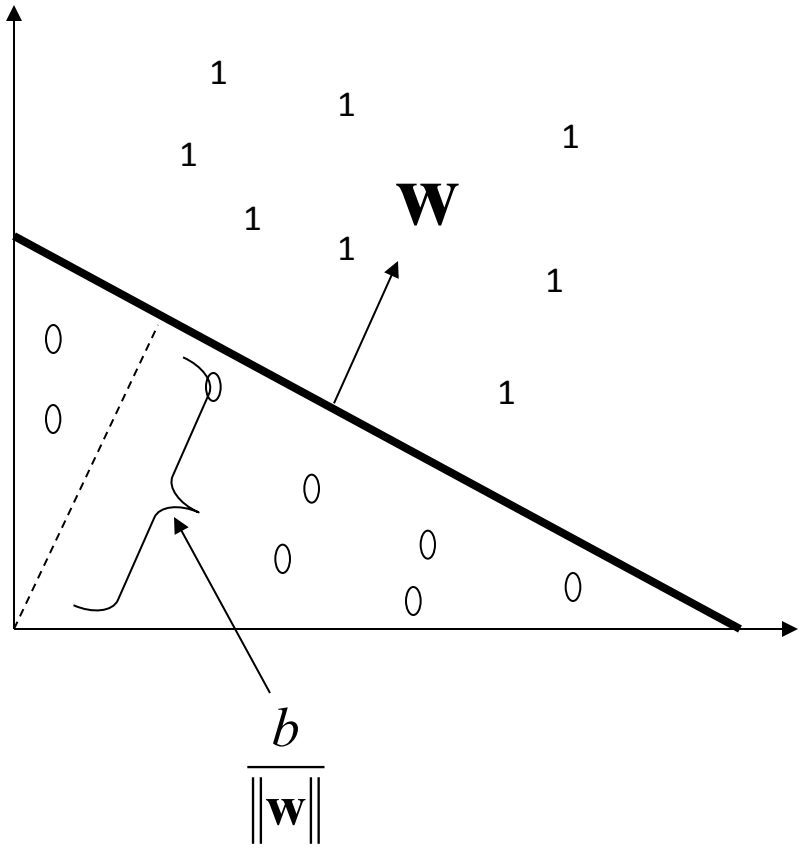
Maximal Margin Classifier

- The bounds on error of classification suggest the possibility of improving generalization by maximizing the margin
- Minimize the risk of overfitting by choosing the maximal margin hyperplane in feature space
- SVMs control capacity by increasing the margin, regardless of the dimensionality of the feature space

Linear separation of the input space

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

$$h(\mathbf{x}) = sign(f(\mathbf{x}))$$



Functional and Geometric Margin

- The functional margin of a linear discriminant (**w**,*b*) w.r.t. a labeled pattern $(\mathbf{x}_i, y_i) \in \Re^d \times \{-1,1\}$ is defined as

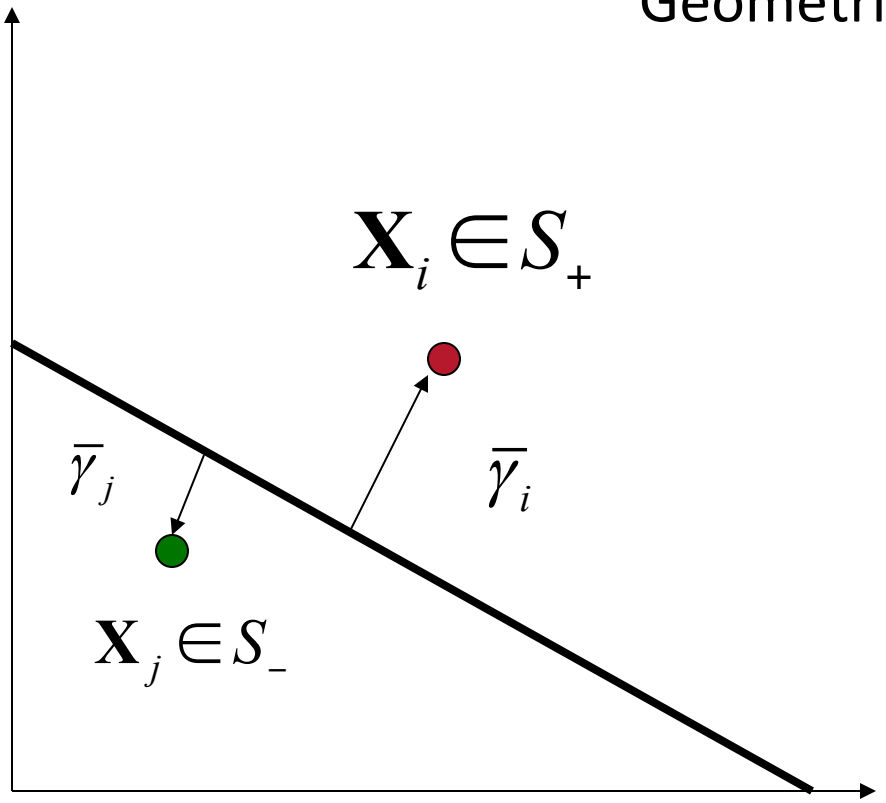
$$\gamma_i \equiv y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$$

- If the functional margin is negative, then the pattern is incorrectly classified, if it is positive then the classifier predicts the correct label.
- The larger $|\gamma_i|$ the further away $\mathbf{x}_i$ is from the discriminant
- This is made more precise in the notion of the geometric margin

$$\overline{\gamma}_i \equiv \frac{\gamma_i}{\|\mathbf{w}\|}$$

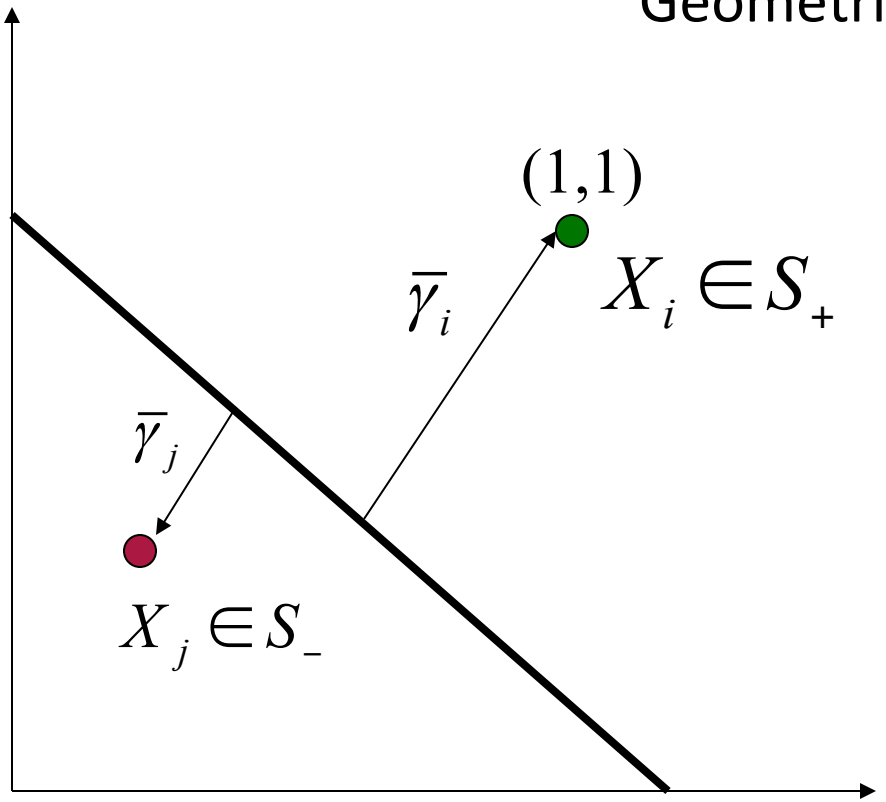
which measures the Euclidean distance of a point from the decision boundary.

Geometric Margin



The geometric margin of two points

Geometric Margin



Example

$$\mathbf{W} = (1 \quad 1)$$

$$b = -1$$

$$\mathbf{X}_i = (1 \quad 1)$$

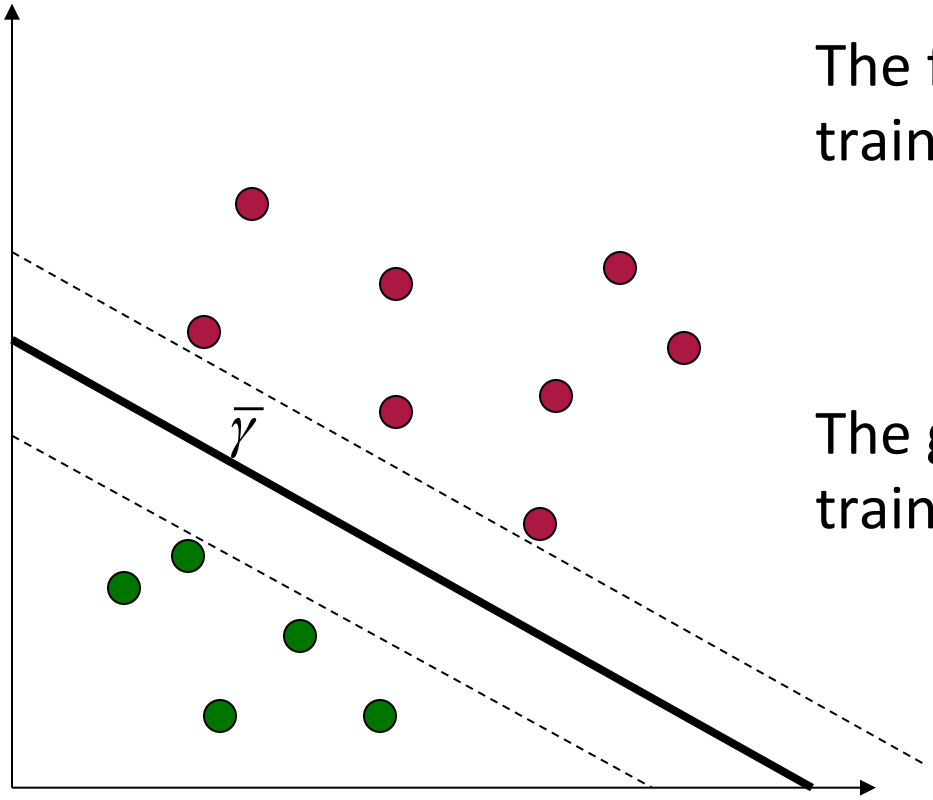
$$Y_i = 1$$

$$\gamma_i = (Y_i)((1)x_1 + (1).x_2 - 1)$$

$$= (1)(1+1-1) = 1$$

$$\bar{\gamma}_i = \frac{\gamma_i}{\|\mathbf{W}\|} = \frac{1}{\sqrt{2}}$$

Margin of a training set



The functional margin of a training set

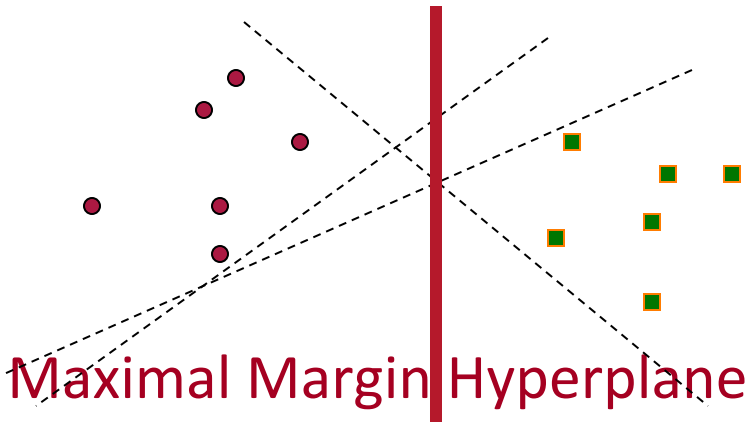
$$\gamma = \min_i \gamma_i$$

The geometric margin of a training set

$$\bar{\gamma} = \min_i \bar{\gamma}_i$$

Maximum Margin Separating Hyperplane

The margin of a training set *S* is the maximum geometric margin over all hyperplanes. A hyperplane realizing this maximum is a maximal margin hyperplane.



VC Dimension of Δ- margin hyperplanes

- If data samples belong to a sphere of radius *R*, then the set of Δ - margin hyperplanes has VC dimension bounded by

$$VC(H) \leq \min(R^2 / \Delta^2, n) + 1$$

- WLOG, we can make *R*=1 by normalizing data points of each class so that they lie within a sphere of radius *R.*
- For large margin hyperplanes, VC-dimension is controlled independent of dimensionality *n* of the feature space

Maximal Margin Classifier

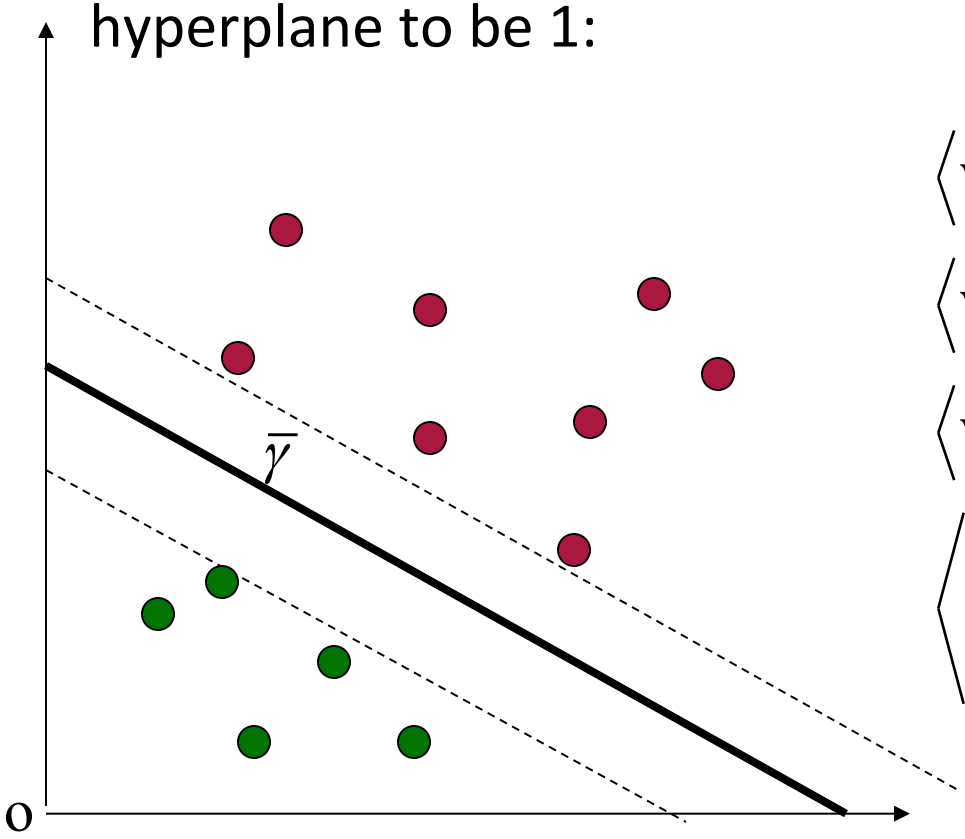
- The bounds on error of classification suggest the possibility of improving generalization by maximizing the margin
- Minimize the risk of overfitting by choosing the maximal margin hyperplane in feature space
- SVMs control capacity by increasing the margin, regardless of the dimensionality of the feature space

Maximizing Margin → Minimizing ||**W**||

- Definition of hyperplane (**w**,b) does not change if we rescale it to (σ**w**, σb), for σ>0
- Functional margin depends on scaling, but geometric margin $\bar{\gamma}$ does not
- If we fix (by rescaling) the functional margin to 1, the geometric margin will be equal 1/||**w**||
- We can maximize the margin by minimizing the norm ||**w**||

Maximizing Margin → Minimizing ||w||

Let $\mathbf{x}^+$ and $\mathbf{x}^-$ be the nearest data points in the sample space. Then we have, by fixing the distance from the hyperplane to be 1:



$$\langle \mathbf{w}, \mathbf{x}^+ \rangle + b = 1$$

$$\langle \mathbf{w}, \mathbf{x}^- \rangle + b = -1$$

$$\langle \mathbf{w}, (\mathbf{x}^+ - \mathbf{x}^-) \rangle = 1 - (-1) = 2$$

$$\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}^+ - \mathbf{x}^-) \right\rangle = \frac{2}{\|\mathbf{w}\|}$$

Learning as optimization

Minimize $\langle \mathbf{w}, \mathbf{w} \rangle$

subject to: $y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1$

Digression: Constrained Optimization

Constrained Optimization

- Primal optimization problem
- Given functions f, g_i, $i=1\ldots k$; h_j $j=1..m$; defined on a domain $\Omega \subseteq \Re^n$,

$$\begin{array}{lllll} \text{minimize} & f(\mathbf{w}) & \mathbf{w}\in\Omega & \left\{\right. & \text{objective function} \\ \text{subject to} & g_i(\mathbf{w})\leq 0 & i=1\ldots k, & \left\{\right. & \text{inequality constraints} \\ & h_j(\mathbf{w})=0 & j=1\ldots m & \left\{\right. & \text{equality constraints} \end{array}$$

Shorthand

$$g(\mathbf{w})\leq 0 \quad \text{denotes} \quad g_i(\mathbf{w})\leq 0 \quad i=1\ldots k$$
$$h(\mathbf{w})=0 \quad \text{denotes} \quad h_j(\mathbf{w})\leq 0 \quad j=1\ldots m$$

Feasible region

$$F=\left\{\mathbf{w}\in\Omega : g(\mathbf{w})\leq 0, h(\mathbf{w})=0\right\}$$

Optimization problems

- Linear program – objective function as well as equality and inequality constraints are linear
- Quadratic program – objective function is quadratic, and the equality and inequality constraints are linear
- Inequality constraints $g_i(\mathbf{w}) \leq 0$ can be active i.e. $g_i(\mathbf{w}) = 0$ or inactive i.e. $g_i(\mathbf{w}) < 0$.
- Inequality constraints are often transformed into equality constraints using slack variables

$$g_i(\mathbf{w}) \leq 0 \leftrightarrow \qquad g_i(\mathbf{w}) + \xi_i = 0 \text{ with } \xi_i \geq 0$$

- We will be interested primarily in convex optimization problems

Convex optimization problem

If function f is convex, any local minimum $\mathbf{w}^*$ of an unconstrained optimization problem with objective function f is also a global minimum, since for any $\mathbf{u} \neq \mathbf{w}^* \quad f(\mathbf{w}^*) \leq f(\mathbf{u})$

A set $\Omega \subset \mathbf{R}^n$ is called convex if, $\forall \mathbf{w}, \mathbf{u} \in \Omega$ and for any $\theta \in (0,1)$, the point $(\theta \mathbf{w} + (1-\theta)\mathbf{u}) \in \Omega$

A convex optimization problem is one in which the set Ω, the objective function and all the constraints are convex

Lagrangian Theory

Given an optimization problem with an objective function $f(\mathbf{w})$ and equality constraints $h_j(\mathbf{w}) = 0 \quad j = 1..m$, we define a Lagrangian as

$$L(\mathbf{w}, \boldsymbol{\beta}) = f(\mathbf{w}) + \sum_{j=1}^{m} \beta_j h_j(\mathbf{w})$$

where β_j are called the Lagrange multipliers. The necessary Conditions for $\mathbf{w}^*$ to be minimum of $f(\mathbf{w})$ subject to the Constraints $h_j(\mathbf{w}) = 0 \quad j = 1..m$ is given by

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\beta}^*)}{\partial \mathbf{w}} = 0, \quad \frac{\partial L(\mathbf{w}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{\beta}} = 0$$

The condition is sufficient if $L(\mathbf{w}, \boldsymbol{\beta}^*)$ is a convex function of $\mathbf{w}$

Lagrangian Theory Example

$$\text{Minimize } f(x,y) = x + 2y$$

$$\text{Subject to: } \quad x^2 + y^2 - 4 = 0$$

$$L(x,y,\boldsymbol{\lambda}) = x + 2y + \left(x^2 + y^2 - 4\right)\boldsymbol{\lambda}$$

$$\frac{\partial L(x,y,\boldsymbol{\lambda})}{\partial x} = 1 + 2\boldsymbol{\lambda}x = 0$$

$$\frac{\partial L(x,y,\boldsymbol{\lambda})}{\partial y} = 2 + 2\boldsymbol{\lambda}y = 0$$

$$\frac{\partial L(x,y,\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = x^2 + y^2 - 4 = 0$$

Solving the above, we have:

$$\boldsymbol{\lambda} = \pm\frac{\sqrt{5}}{4} \qquad x = \mp\frac{2}{\sqrt{5}} \qquad y = \mp\frac{4}{\sqrt{5}}$$

$$f \text{ is minimized when } x = -\frac{2}{\sqrt{5}},\ y = -\frac{4}{\sqrt{5}}$$

Lagrangian theory – example

Find the lengths *u, v, w* of sides of the box that has the largest volume for a given surface area *c*

minimize $-uvw$

subject to $wu + uv + vw = \frac{c}{2}$

$$L = -uvw + \beta\left(wu + uv + vw - \frac{c}{2}\right)$$

$$\frac{\partial L}{\partial w} = 0 = -uv + \beta(u+v); \frac{\partial L}{\partial u} = 0 = -vw + \beta(v+w); \frac{\partial L}{\partial v} = 0 = -wu + \beta(u+w);$$

$$\frac{\partial L}{\partial \beta} = 0 = wu + uv + vw - \frac{c}{2};$$

$$u = v = w = \sqrt{\frac{c}{6}}$$

Lagrangian Optimization - Example

- The entropy of a probability distribution $\mathbf{p}=(p_1 \ldots p_n)$ over a finite set $\{1, 2, \ldots n\}$ is defined as

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

- The maximum entropy distribution can be found by minimizing $-H(\mathbf{p})$ subject to the constraints

$$\sum_{i=1}^{n} p_i = 1$$

$$\forall i \quad p_i \geq 0$$

Generalized Lagrangian Theory

Given an optimization problem with domain $\Omega \subseteq \Re^n$

$$\begin{array}{lllll} \text{minimize} & f(\mathbf{w}) & \mathbf{w} \in \Omega & \left\{ \text{objective function} \right. \\ \text{subject to} & g_i(\mathbf{w}) \leq 0 & i = 1 \ldots k, & \left\{ \text{inequality constraints} \right. \\ & h_j(\mathbf{w}) = 0 & j = 1 \ldots m & \left\{ \text{equality constraints} \right. \end{array}$$

where f is convex, and g_i and h_j are affine, we can define the generalized Lagrangian function as:

$$L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{w}) + \sum_{i=1}^{k} \alpha_i g_i(\mathbf{w}) + \sum_{j=1}^{m} \beta_j h_j(\mathbf{w})$$

$F(x)$ is affine if $F(x)$=$G(x)$+b where $G(x)$ is a linear function of x and b is a constant

Generalized Lagrangian Theory: KKT Conditions

Given an optimization problem with domain $\Omega \subseteq \Re^n$

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{w}) && \mathbf{w} \in \Omega && \{ \text{objective function} \\ \text{subject to} \quad & g_i(\mathbf{w}) \leq 0 && i = 1 \ldots k, && \{ \text{inequality constraints} \\ & h_j(\mathbf{w}) = 0 && j = 1 \ldots m && \{ \text{equality constraints} \end{aligned}$$

where f is convex, and g_i and h_j are affine. The necessary and Sufficient conditions for $\mathbf{w}^*$ to be an optimum are the existence of $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ such that

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \mathbf{w}} = 0, \quad \frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{\beta}} = 0,$$

$$\alpha_i^* g_i(\mathbf{w}^*) = 0; g_i(\mathbf{w}^*) \leq 0; \alpha_i^* \geq 0; i = 1 \ldots k$$

Minimize $f(x,y) = (x-1)^2 + (y-3)^2$

Subject to : $x + y \le 2;\ x - y \le 0$

$$L = (x-1)^2 + (y-3)^2 + \lambda_1(x + y - 2) + \lambda_2(x - y)$$

$$\frac{\partial L}{\partial x} = 2(x-1) + \lambda_1 + \lambda_2 = 0$$

$$\frac{\partial L}{\partial y} = 2(y-3) + \lambda_1 - \lambda_2 = 0; \quad \lambda_1(x + y - 2) = 0;\ \lambda_2(x - y) = 0$$

$\lambda_1 \ge 0,\ \lambda_2 \ge 0,\ x + y \le 2;\ x - y \le 0$

Case 1 : $\lambda_1 = 0,\ \lambda_2 = 0; \quad \Rightarrow 2(x-1) = 0;\ 2(y-3) = 0 \Rightarrow x = 1;\ y = 3$

This is not a feasible solution because it violates $x + y \le 2$

Similarly, case 2 : $\lambda_1 = 0,\ \lambda_2 \ne 0$ does not yield a feasible solution

Case 3 : $\lambda_1 \ne 0,\ \lambda_2 = 0 \Rightarrow 2(x-1) + \lambda_1 = 0;\ 2(y-3) + \lambda_1 = 0;\ \lambda_1(x + y - 2) = 0$

which yields $x = \frac{2 - \lambda_1}{2};\ y = \frac{6 - \lambda_1}{2};\ \lambda_1\left(\frac{2 - \lambda_1}{2} + \frac{6 - \lambda_1}{2} - 2\right) = 0 \Rightarrow \lambda_1 = 2, x = 0; y = 2$

which is a feasible solution

Case 4 $\lambda_1 \ne 0,\ \lambda_2 \ne 0$ does not yield a feasible solution

Hence, the solution is $x = 0;\ y = 2$

Dual optimization problem

A primal problem can be transformed into a dual by simply setting to zero, the derivatives of the Lagrangian with respect to the primal variables and substituting the results back into the Lagrangian to remove dependence on the primal variables, resulting in $\theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \inf_{\mathbf{w} \in \Omega} L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

which contains only dual variables and needs to be maximized under simpler constraints

The infimum of a set *S* of real numbers is denoted by inf(*S*) and is defined to be the largest real number that is smaller than or equal to every number in *S*. If no such number exists then inf(*S*) = −∞.

$$\inf\{x \in \mathbb{R} \mid 0 < x < 1\} = 0$$

Maximum Margin Hyperplane

- The problem of finding the maximal margin hyperplane is a constrained optimization problem
- Use Lagrange theory (extended by Karush, Kuhn, and Tucker – KKT)

Lagrangian:

$$L_p(\mathbf{w}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_i \alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right]$$

$$\alpha_i \geq 0$$

From Primal to Dual

Minimize $L_p(\mathbf{w})$ with respect to $(\mathbf{w},b)$ requiring that derivatives of $L_p(\mathbf{w})$ with respect to all α_i vanish, subject to the constraints $\alpha_i \geq 0$

Differentiating $L_p(\mathbf{w})$:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0$$

$$\frac{\partial L_p}{\partial b} = 0$$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\sum_i \alpha_i y_i = 0$$

Substituting the equality constraints back into $L_p(\mathbf{w})$ we obtain a dual problem.

The Dual Problem

Maximize:

$$L_D(\mathbf{w}) = \frac{1}{2}\left\langle \left(\sum_i \alpha_i y_i \mathbf{x}_i \right) \left(\sum_j \alpha_j y_j \mathbf{x}_j \right) \right\rangle$$

$$- \sum_i \alpha_i \left[y_i \left(\left\langle \left(\sum_j \alpha_j y_j \mathbf{x}_j \right), \mathbf{x}_i \right\rangle + b \right) - 1 \right]$$

$$= \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i \mathbf{x}_j \right\rangle - \sum_{ij} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i \mathbf{x}_j \right\rangle - b \sum_i \alpha_i y_i + \sum_i \alpha_i$$

$$= -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_i \alpha_i$$

Subject to $\sum_i \alpha_i y_i = 0$ and $\alpha_i \geq 0$

b has to be solved using primal constraints

$$b = -\frac{\max_{y_i=-1}(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle) + \min_{y_i=1}(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle)}{2}$$

Karush-Kuhn-Tucker Conditions

$$L_p(\mathbf{w}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_i \alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right]$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

$$y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \geq 0$$

$$\alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right] = 0$$

$$\alpha_i \geq 0$$

Solving for the SVM is equivalent to finding a solution to the KKT conditions

Karush-Kuhn-Tucker Conditions for SVM

- The KKT conditions state that optimal solutions $\alpha_i(\mathbf{w}, b)$ must satisfy

$$\alpha_i\left[y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) - 1\right] = 0$$

- Only the training samples $\boldsymbol{x}_i$ for which the functional margin = 1 can have nonzero α_i. They are called Support Vectors.

- The optimal hyperplane can be expressed in the dual representation in terms of this subset of training samples – the support vectors

$$f(\mathbf{x}, \boldsymbol{\alpha}, b) = \sum_{i=1}^{l} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = \sum_{i \in \mathrm{SV}} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b$$

$$\langle \mathbf{w}, \mathbf{w} \rangle = \sum_{j=1}^{l} \sum_{i=1}^{l} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$= \sum_{j \in SV} \alpha_j y_j \sum_{i \in SV} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

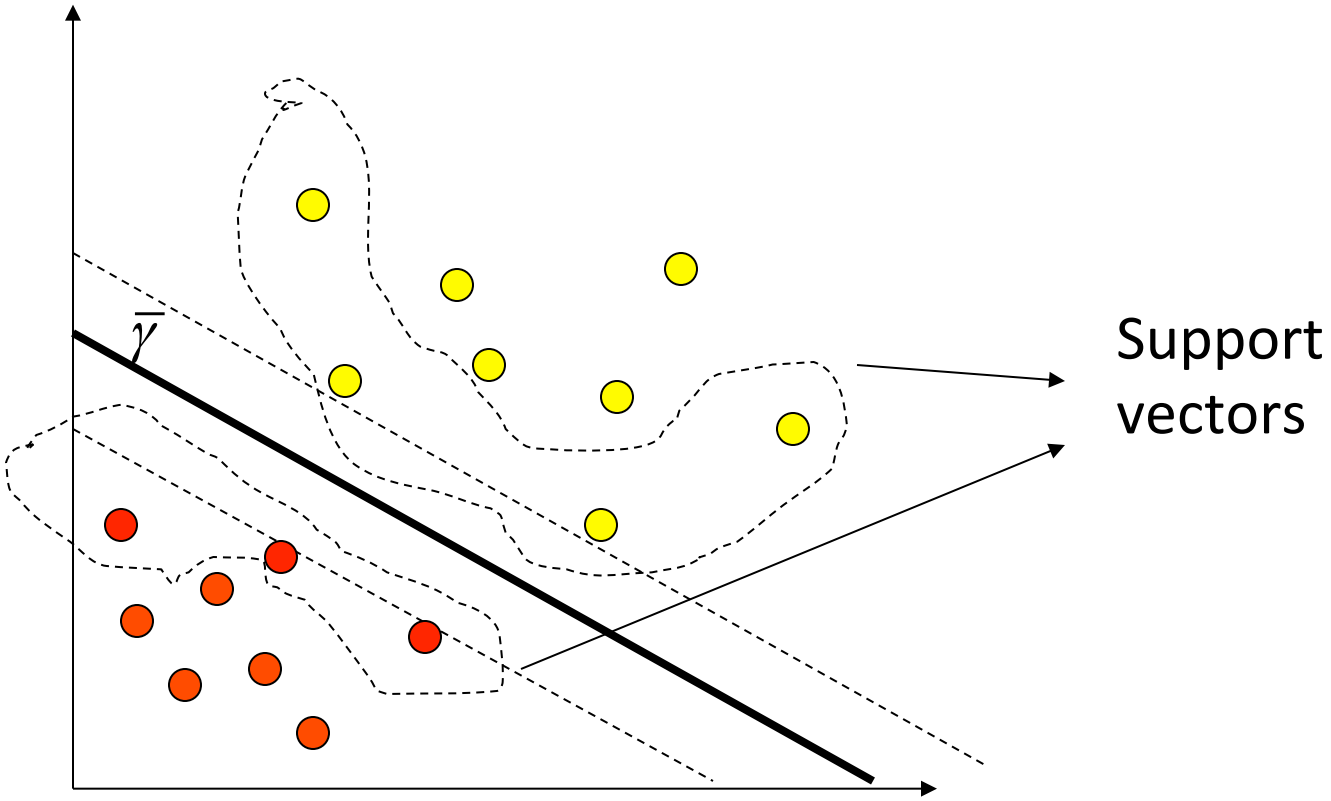
$$\because \mathbf{x}_j \in SV, \quad y_j \left(\sum_{i \in SV} y_i \alpha_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle + b \right) = 1 \Rightarrow y_j \sum_{i \in SV} y_i \alpha_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle = 1 - y_j b$$

$$\langle \mathbf{w}, \mathbf{w} \rangle = \sum_{j \in SV} \alpha_j (1 - y_j b) = b \left(\sum_{j \in SV} \alpha_j y_j + \sum_{j \notin SV} 0 y_j \right) + \sum_{j \in SV} \alpha_j$$

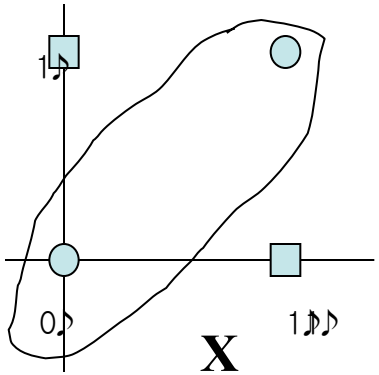
$$= b \underbrace{\left(\sum_{j \in SV} \alpha_j y_j + \sum_{j \notin SV} \alpha_j y_j \right)}_{0 \text{ because of KKT condns}} + \sum_{j \in SV} \alpha_j = \sum_{i \in SV} \alpha_i$$

So we have $$\bar{\gamma} = \frac{1}{\|\mathbf{w}\|} = \sqrt{\left(\sum_{\mathbf{x}_i \in SV} \alpha_i \right)}$$

Support Vector Machines Yield Sparse Solutions



Example: XOR



$$\Phi(\mathbf{w}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$L(\mathbf{w},b,\alpha) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

$$Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

$$K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^2$$

$$\varphi(\mathbf{x}) = [1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]^T$$

$\mathbf{x}$	y
[-1,-1]	-1
[-1,+1]	+1
[+1,-1]	+1
[+1,+1]	-1

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

Example: XOR

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \tfrac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_2^4)$$

Setting derivative of *Q* wrt the dual variables to 0

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$$
$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$$
$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$$
$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$$

$$\alpha_{o,1} = \alpha_{o,2} = \alpha_{o,3} = \alpha_{o,4} = \tfrac{1}{8}$$

$$Q_o(\alpha) = \tfrac{1}{4}$$

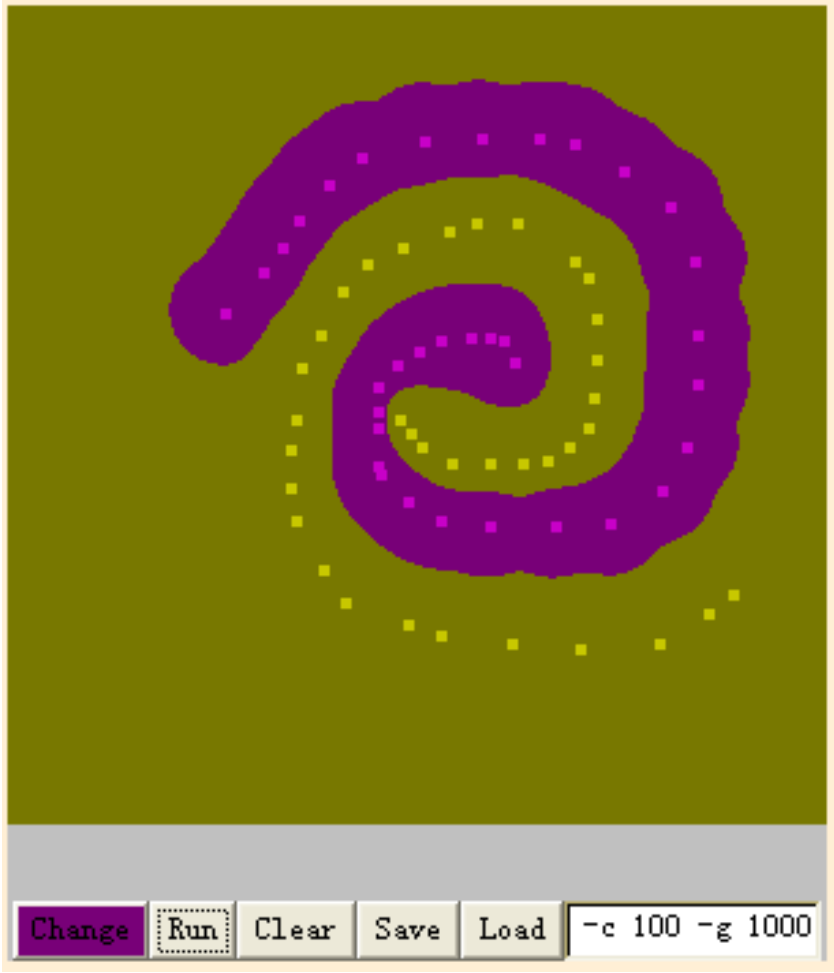
Not surprisingly, each of the training samples is a support vector

$$w_o = \sum_{i=1}^{N} \alpha_i y_i \varphi(\mathbf{x}_i)$$

$$= \begin{bmatrix} 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 \end{bmatrix}^T$$

$$\tfrac{1}{2}\|w_o\|^2 = \tfrac{1}{4}, \|w_o\| = \tfrac{1}{\sqrt{2}}$$

Example – Two Spirals



Summary of Maximal margin classifier

- Good generalization when the training data is noise free and separable in the kernel-induced feature space
- Excessively large Lagrange multipliers often signal outliers – data points that are most difficult to classify

  – SVM can be used for identifying outliers
- Focuses on boundary points If the boundary points are mislabeled (noisy training data)
  - the result is a terrible classifier if the training set is separable
  - Noisy training data often makes the training set non separable
  - Use of highly nonlinear kernels to make the training set separable increases the likelihood of over fitting – despite maximizing the margin

Non-Separable Case

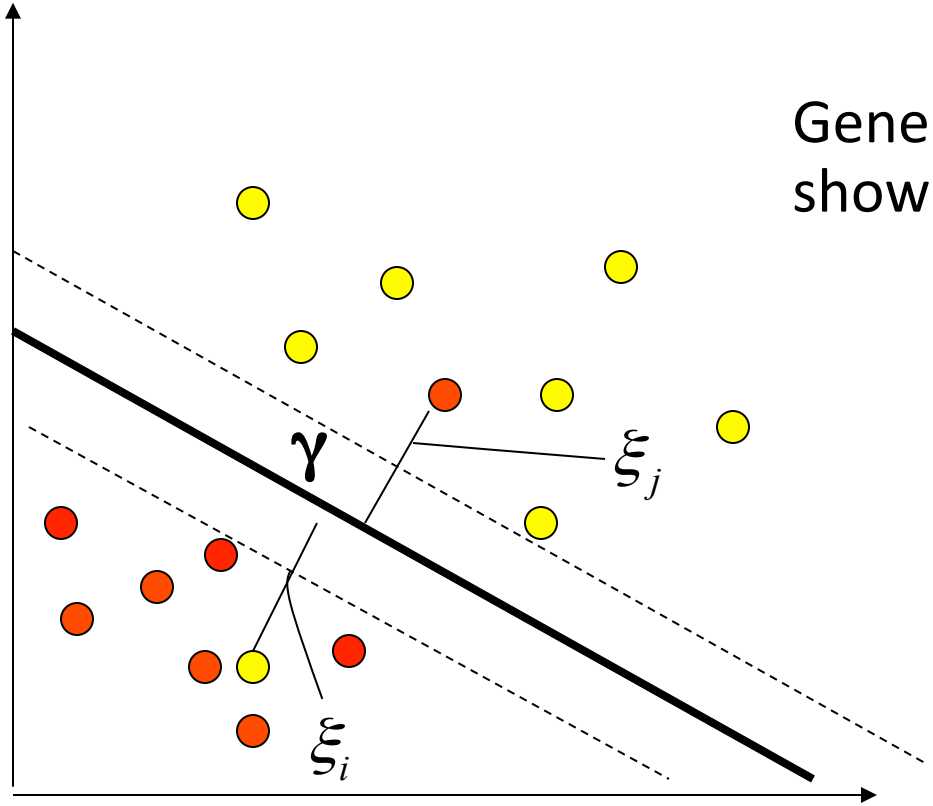
- In the case of non-separable data in feature space, the objective function grows arbitrarily large.
- Solution: Relax the constraint that all training data be correctly classified but only when necessary to do so
- Cortes and Vapnik, 1995 introduced slack variables in the constraints:

$$\xi_i, \quad i = 1,..,l$$

$$\xi_i = \max\left(0, \gamma - y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right)$$

Non-Separable Case

$$\xi_i = \max(0, \gamma - y_i(\langle \mathbf{w}, x_i \rangle + b)).$$



Generalization error can be shown to be

$$\varepsilon \leq \frac{1}{l}\left(\frac{R + \sqrt{\sum_i \xi_i^2}}{\gamma}\right)^2$$

Non-Separable Case

- For error to occur, the corresponding ξ_i must exceed γ (which was chosen to be unity)
- So $\sum_i \xi_i$ is an upper bound on the number of training errors.
- So the objective function is changed from $\frac{\|\mathbf{w}\|^2}{2}$ to

$$\left(\frac{\|\mathbf{w}\|^2}{2}\right) + C\sum_i \xi_i^k$$

where *C and k* are parameters

(typically *k*=1 or 2 and *C* is a large positive constant

The Soft-Margin Classifier

Minimize: $\frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_i \xi_i$

Subject to: $y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq \left(1 - \xi_i \right)$

$\xi_i \geq 0$

$$L_P = \underbrace{\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i}_{\text{objective function}} - \underbrace{\sum_i \alpha_i \left(y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 + \xi_i \right)}_{\text{inequality constraints involving } \xi_i} - \underbrace{\sum_i \mu_i \xi_i}_{\text{non-negativity constraints on } \xi_i}$$

The Soft-Margin Classifier: KKT Conditions

$$\xi_i \geq 0, \;\; \alpha_i \geq 0, \;\; \mu_i \geq 0$$

$$\alpha_i \left(y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 + \xi_i \right) = 0$$

$$\mu_i \xi_i = 0$$

$\alpha_i > 0$ only if

- $\mathbf{x}_i$ is a support vector i.e., $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = \pm 1$ and hence $\xi_i = 0$

or

- $\mathbf{x}_i$ is misclassified by $(\mathbf{w}, b)$ and hence $\xi_i > 0$

From primal to dual

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \left(y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) - 1 + \xi_i\right) - \sum_i \mu_i \xi_i$$

Equating the derivatives of L_P wrt primal variables to 0,

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow \alpha_i + \mu_i = C \Rightarrow 0 \leq \alpha_i \leq C$$

Substituting back into L_P, we get the dual L_D to be *maximized*

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

Implementation Techniques

Maximizing a quadratic function, subject to a linear equality and inequality constraints

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$0 \leq \alpha_i \leq C$$

$$\sum_i \alpha_i y_i = 0$$

$$\frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i} = 1 - y_i \sum_j \alpha_j y_j K(x_i, x_j)$$

On-line algorithm for the 1-norm soft margin (bias b=0)

Given training set D

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$

$$\eta_i = \frac{1}{K(\mathbf{x}_i, \mathbf{x}_i)}$$

repeat

for $i=1$ *to* l

$$\alpha_i \leftarrow \alpha_i + \eta_i (1 - y_i \sum_j \alpha_j y_j K(x_i, x_j))$$

$$\text{if} \quad \alpha_i < 0 \quad \text{then} \quad \alpha_i \leftarrow 0$$

else

$$\text{if} \quad \alpha_i > C \quad \text{then} \quad \alpha_i \leftarrow C$$

end for

until stopping criterion satisfied

return $\boldsymbol{\alpha}$

In the general setting, $b \neq 0$ and we need to solve for b

Implementation Techniques

- Use QP packages (MINOS, LOQO, quadprog from MATLAB optimization toolbox). They are not online and require that the data are held in memory in the form of kernel matrix
- Stochastic Gradient Ascent. Sequentially update 1 weight at the time. So it is online. Gives excellent approximation in most cases

$$\hat{\alpha}_i \leftarrow \alpha_i + \frac{1}{K(x_i, x_i)}\left(1 - y_i \sum_j \alpha_j y_j K(x_i, x_j)\right)$$

Chunking and Decomposition

Given training set D

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$

Select an arbitrary working set $\hat{D} \subset D$

repeat

solve optimization problem on $\hat{D}$

select new working set from data not satisfying KKT conditions

until stopping criterion satisfied

return $\boldsymbol{\alpha}$

Sequential Minimal Optimization S.M.O.

- At each step SMO: optimize two weights α_i, α_j simultaneously in order not to violate the linear constraint $\sum_i \alpha_i y_i = 0$
- Optimization of the two weights is performed analytically
- Realizes gradient descent without leaving the linear constraint (J.Platt)
- Online versions exist (Li, Long; Gentile)

The Perceptron Algorithm Revisited

- The perceptron works by adding misclassified positive or subtracting misclassified negative examples to an arbitrary weight vector, which (without loss of generality) we assumed to be the zero vector. $\alpha_i \leftarrow \alpha_i + \eta(d_i - y_i)\mathbf{x}_i$
- The final weight vector is a linear combination of training points

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i,$$

where, since the sign of the coefficient of $\mathbf{x}_i$ is given by label y_i, the α_i are positive values, proportional to the number of times, misclassification of $\mathbf{x}_i$ has caused the weight to be updated. It is called the embedding strength of the pattern $\mathbf{x}_i$.

-

From Perceptron Rule to SMO Rule

- Recall that SVM optimization problem has the added requirement that: $\sum_{i=1}^{N} y_i \alpha_i = 0$
- Therefore if we increase one α by an amount η, in either direction, then we have to change another α by an equal amount in the opposite direction (relative to class value). We can accomplish this by changing: $\alpha_i \leftarrow \alpha_i + \eta(d_i - y_i)\mathbf{x}_i$
- But because of the constraint, we also adjust α_j such that $y_i\alpha_i + y_j\alpha_j$ stays same

First of Two Other Changes

- We set $\eta = \frac{1}{K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)}$

- For a given difference in error between two examples, the step size is larger when the two examples are more similar. When x_1 and x_2 are similar, $K(x_1,x_2)$ is larger (for typical kernels, including dot product). Hence the denominator is smaller and step size is larger.

Second of Two Other Changes

- Recall our formulation had an additional constraint:
- Therefore, we "clip" any change we make to any α to respect this constraint $0 \leq \alpha_i \leq C, \forall i$
- This yields the SMO algorithm (Platt,

Problem of finding the p-norm maximum margin hyperplane [Mangasarian 99]

Given: (linearly separable) $S=((\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T))$,

Goal: Find an approximate solution of $(\mathbf{w}^*,b^*)$

$$(\boldsymbol{w}^*, b^*) = \arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_q^2$$

$$\text{sub. to:}$$

$$y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j + b) \geq 1 \quad (j = 1,\ldots T)$$

q-norm (dual norm)
1/p+1/q=1
E.g.
p=2, q=2
p=∞, q=1

We want an online alg. solving the problem with small # of updates.

ROMMA

(Relaxed Online Maximum Margin Algorithm)[Li&Long,'02]

Given: $S=((\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_{t-1},y_{t-1}))$, $\mathbf{x}_t$,

1. Predict $\hat{y}_t=\text{sign}(\mathbf{w}_t\cdot\mathbf{x}_t)$, and receive y_t
2. If $y_t(\mathbf{w}_t\cdot\mathbf{x}_t)<1-\delta$ (margin is "insufficient"),
3. update:

$$\boldsymbol{w}_{t+1} = \underset{\boldsymbol{w}}{\text{argmin}}. \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{sub.to}: \quad y_t(\boldsymbol{w}\cdot\boldsymbol{x}_t)\geq 1,$$

$$\boldsymbol{w}\cdot\boldsymbol{w}_t \geq \|\boldsymbol{w}_t\|_2^2$$

Constraint over the last Example which causes an update

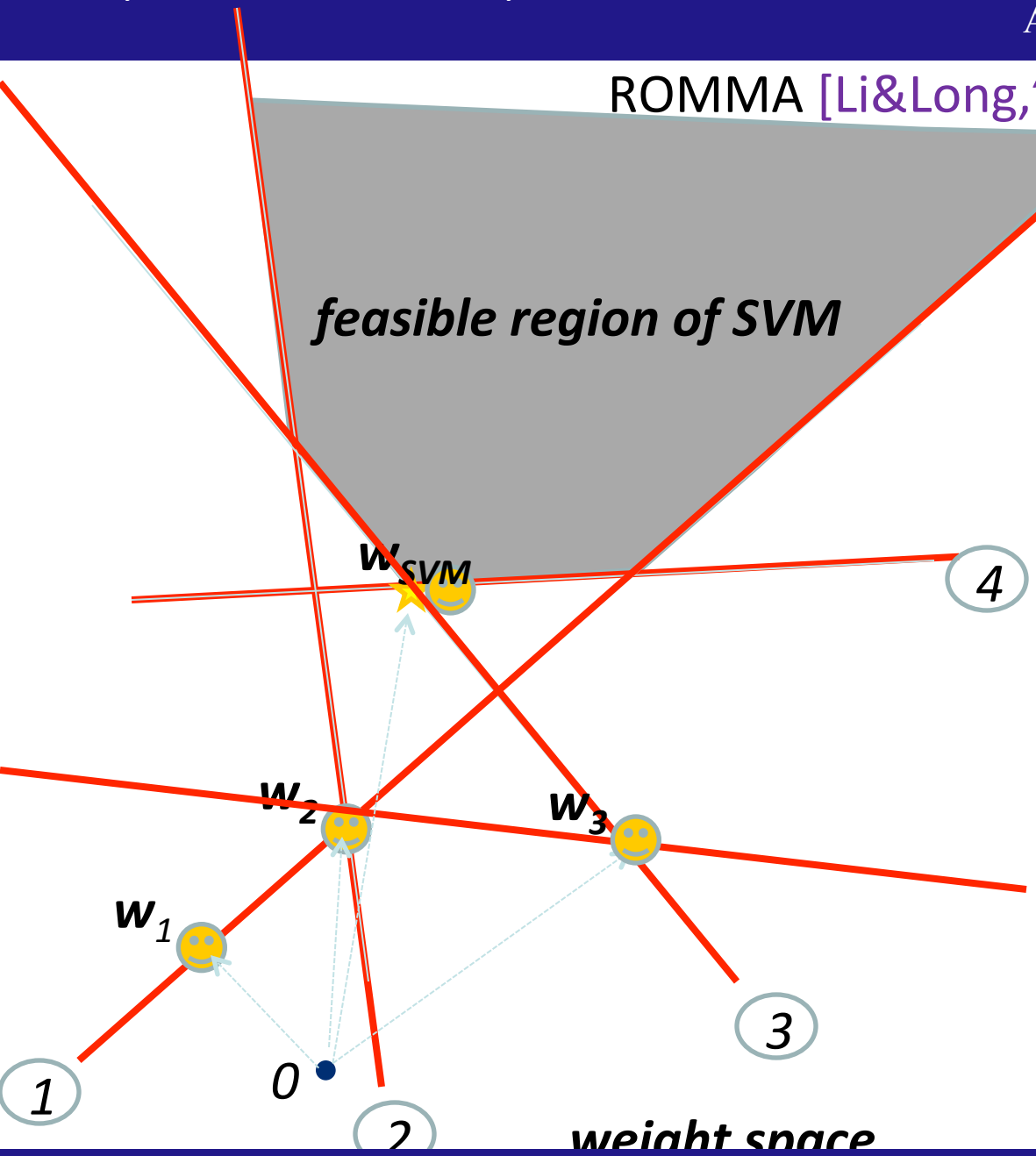
2 constraints

Constraint over the last hyperplane

4. Otherwise, $\mathbf{w}_{t+1}=\mathbf{w}_t$

NOTE: bias is fixed with 0

ROMMA [Li&Long,'02]



SVM (without bias)

$$\min_{w} \frac{1}{2}\|w\|_2^2$$

$$\text{sub.to: } y_j(w \cdot x_j) \geq 1,$$

$$(j = 1,\ldots,4)$$

ROMMA

$$\min_{w} \frac{1}{2}\|w\|_2^2$$

$$\text{sub.to: } y_t(w \cdot x_t) \geq 1,$$

$$w \cdot w_{t-1} \geq \|w\|_2^2$$

Solution of ROMMA

Solution of ROMMA is an additive update:

(i) If $\boldsymbol{w}_{t+1} \cdot \boldsymbol{w}_t > \|\boldsymbol{w}_t\|_2^2$,

$$\boldsymbol{w}_{t+1} = \alpha y_t \boldsymbol{x}_t, \text{ where } \alpha = \frac{1}{\|\boldsymbol{x}_t\|_2^2}.$$

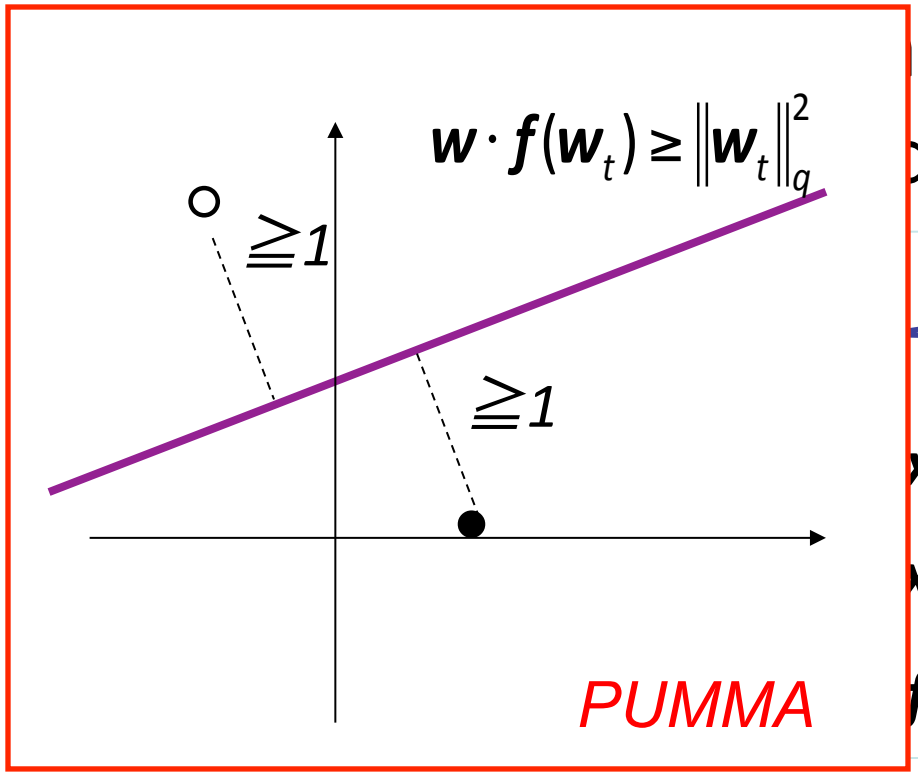
(ii) Otherwise,

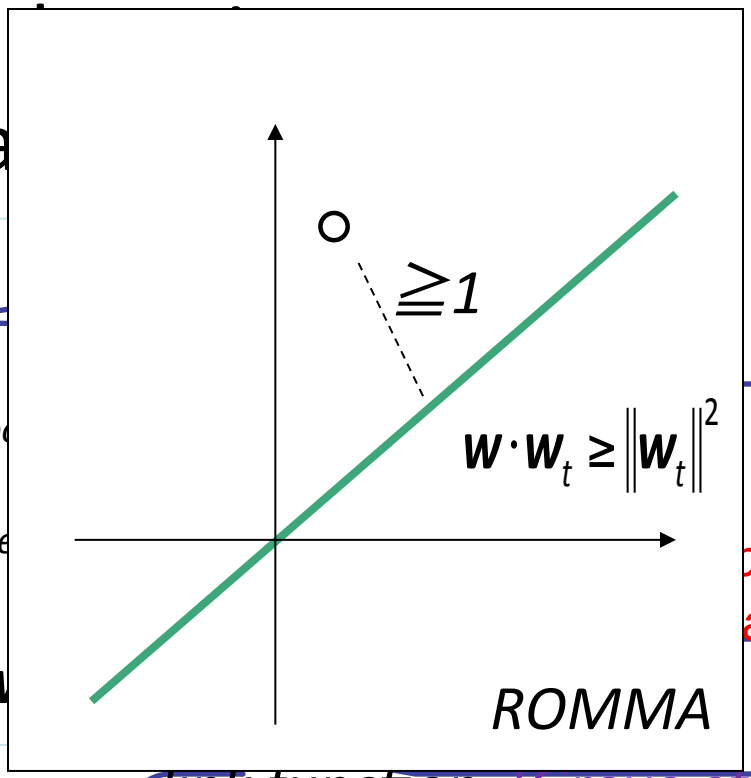
$$\boldsymbol{w}_{t+1} = \alpha y_t \boldsymbol{x}_t + \beta \boldsymbol{w}_t, \text{ where}$$

$$\alpha = \frac{\|\boldsymbol{w}_t\|_2^2 \left(1 - \boldsymbol{w}_t \cdot \boldsymbol{x}_t\right)}{\|\boldsymbol{w}_t\|_2^2 \|\boldsymbol{x}_t\|_2^2 - (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)^2}, \beta = \frac{\|\boldsymbol{w}_t\|_2^2 \|\boldsymbol{x}_t\|_2^2 - 2(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)}{\|\boldsymbol{w}_t\|_2^2 \|\boldsymbol{x}_t\|_2^2 - (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)^2}.$$

PUMMA [Hatano, 2008]

Given: $S=((\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_{t-1},y_{t-1}))$, $\boldsymbol{x}_t$,





and
bles
ates

link function [Grove et al. 97]

$$f(\boldsymbol{w})=\frac{sign(w_i)|w_i|_q^{q-1}}{\|\boldsymbol{w}\|^{q-2}}$$

Solution of PUMMA

Solution of PUMMA is found numerically:

(i) If $\boldsymbol{w}_{t+1} \cdot \boldsymbol{w}_t > \|\boldsymbol{w}_t\|_2^2$,

$$\boldsymbol{w}_{t+1} = \alpha y_t \boldsymbol{z}_t, \text{ where } \alpha = \frac{2}{\|\boldsymbol{z}_t\|_2^2} \text{ and } \boldsymbol{z}_t = \boldsymbol{x}_t^{pos} - \boldsymbol{x}_t^{neg}.$$

(ii) Otherwise,

$$\boldsymbol{w}_{t+1} = \alpha y_t \boldsymbol{z}_t + \beta \boldsymbol{w}_t, \text{ where}$$

$$(\alpha, \beta) = \text{argmin}. \frac{1}{2}\|\alpha \boldsymbol{z}_t + \beta \boldsymbol{f}(\boldsymbol{w}_t)\|_p^2 - 2\alpha - \beta \|\boldsymbol{f}(\boldsymbol{w}_t)\|_p^2,$$

which is solved by the Newton method.

In either cases,

$$b_{t+1} = \frac{-(\boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t^{pos} + \boldsymbol{w}_{t+1} \cdot \boldsymbol{x}_t^{neg})}{2}.$$

x^{pos}_t, x^{neg}_t
: last positive and negative examples which incur updates

Observation:
For p=2, the solution is the same as that of ROMMA for $z_t = x_t^{pos} - x_t^{neg}$.

SVM Implementations

- Details of optimization and Quadratic Programming
- SVMLight – one of the first practical implementations of SVM (Platt)
- Matlab toolboxes for SVM
- LIBSVM – one of the best SVM implementations is by Chih-Chung Chang and Chih-Jen Lin of the National University of Singapore http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- WLSVM – LIBSVM integrated with WEKA machine learning toolbox Yasser El-Manzalawy (at ISU AI lab)

http://www.codeforge.com/read/204900/WLSVM.java html

Why does SVM Work?

- The feature space is often very high dimensional. Why don't we have the curse of dimensionality?
- A classifier in a high-dimensional space has many parameters and is hard to estimate
- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is the capacity of a classifier
- Typically, a classifier with many parameters is very flexible, but there are also exceptions
  - Let $x_i=10^i$ where i ranges from 1 to n. The classifier $y = \text{sign}\left(\sin(\alpha x)\right)$ can classify all x_i correctly for all possible combination of class labels on x_i
  - This 1-parameter classifier is very flexible

Why does SVM work?

- Vapnik argues that the capacity of a classifier should not be characterized by the number of parameters, but by the capacity of a classifier
  - This is formalized by the "VC-dimension" of a classifier
- The minimization of $||w||^2$ subject to the condition that the geometric margin =1 has the effect of restricting the VC-dimension of the classifier in the feature space
- The SVM performs structural risk minimization: the empirical risk (training error), plus a term related to the generalization ability of the classifier, is minimized
- SVM loss function behaves like ridge regression.
  - The term $\frac{1}{2}||\mathbf{w}||^2$ "shrinks" the parameters towards zero to avoid overfitting

Choosing the Kernel Function

- Probably the most tricky part of using SVM
- The kernel function should maximize the similarity among instances within a class while accentuating the differences between classes
- A variety of kernels have been proposed (diffusion kernel, Fisher kernel, string kernel, …) for different types of data -
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try for data that live in a fixed dimensional input space
- Low order Markov kernels and its relatives are good ones to consider for structured data – strings, images, etc.

Other Aspects of SVM

- How to use SVM for multi-class classification?
  - One can change the QP formulation to become multi-class
  - More often, multiple binary classifiers are combined
  - One can train multiple one-versus-all classifiers, or combine multiple pairwise classifiers “intelligently”
- How to interpret the SVM discriminant function value as probability?
  - By performing logistic regression on the SVM output of a set of data that is not used for training
- Some SVM software (like libsvm) have these features built-in

Recap: How to Use SVM

- Prepare the data set
- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
  - You can use the values suggested by the SVM software
  - You can use a tuning set to tune C
- Execute the training algorithm and obtain α_i and b
- Unseen data can be classified using the α_i, the support vectors, and b

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optima
  - 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.

Recent developments

- Better understanding of the relation between SVM and regularized discriminative classifiers (logistic regression)
- Knowledge-based SVM – incorporating prior knowledge as constraints in SVM optimization (Shavlik et al)
- A zoo of kernel functions
- Extensions of SVM to multi-class and structured label classification tasks

Representative Applications of SVM

- Handwritten letter classification
- Text classification
- Image classification – e.g., face recognition
- Bioinformatics
  - Gene expression based tissue classification
  - Protein function classification
  - Protein structure classification
  - Protein sub-cellular localization
- ….

Kernel Logistic Regression

- We saw that under fairly general assumptions concerning the underlying generative model, the posterior probability of class given $\mathbf{x}$ can be expressed in the form of a logistic function of an affine or polynomial (in the simplest case, linear) function of $\mathbf{x}$ in the case of a binary classification task.

$$P(y=1 \mid \mathrm{x}) = \frac{1}{1+e^{-\langle \mathrm{w}, G(\mathrm{x}) \rangle}} = \frac{1}{1+e^{-\eta(\mathrm{x},\mathrm{w})}} = \mu(\mathrm{x},\mathrm{w})$$

where

$$\eta(\mathrm{x},\mathrm{w}) = \mathrm{w}^T G(\mathrm{x}) = \langle \mathrm{w}, G(\mathrm{x}) \rangle$$

$$\text{Simplest case}: \; G(\mathrm{x}) = \mathrm{x}$$

Logistic Regression

Note that the posterior probability of $Y=1$ is same as the conditional expectation of y given $\mathbf{x}$:

$$E(y \mid \mathrm{x}) = 1 \cdot P(y=1 \mid \mathrm{x}) + 0 \cdot P(y=0 \mid \mathrm{x})$$

$$= P(y=1 \mid \mathrm{x}) = \mu(\mathrm{x},\mathrm{w}) = (\mu(\mathrm{x},\mathrm{w}))^{y} (1-\mu(\mathrm{x},\mathrm{w}))^{1-y}$$

where

$$\mu(\mathrm{x},\mathrm{w}) = \frac{1}{1+e^{-\eta(\mathrm{x},\mathrm{w})}} = \frac{1}{1+e^{-\mathrm{w}^{T}\mathrm{x}}}$$

Hence estimating $\mathrm{P}(Y=1|\mathbf{x})$ is equivalent to performing logistic regression

Kernel logistic regression

$$\eta(\mathrm{x},\mathrm{w}) = \sum_{i=0}^{k} w_i \varphi_i(\mathrm{x})$$

$$\text{Linear regression}: \ \varphi_i(\mathrm{x}) = x_i$$

$$\eta(\mathrm{x},\mathrm{w}) = \sum_{i=0}^{k} w_i x_i = w_0 + \sum_{i=1}^{k} w_i x_i$$

$$\text{Kernel regression}: \ \varphi_i(\mathrm{x}) = K(\mathrm{x},\mathrm{x}_i)$$

$$\eta(\mathrm{x},\mathrm{w}) = w_0 + \sum_{i=1}^{N} w_i K(\mathrm{x},\mathrm{x}_i)$$

Maximum likelihood estimation of $\mathbf{w}$

$$D = \{ (\mathrm{x}_n, y_n) \}; \;\; y_n \in \{0,1\}; \;\; n = 1..N$$

$$\eta_n = \mathrm{w}^T \mathrm{x}_n; \quad \mu_n = \frac{1}{1+e^{-\eta_n}} = P(y_n = 1 \mid \mathrm{x}_n)$$

Under the iid assumption, Likelihood

$$P(y_1 \ldots y_N \mid \mathbf{x}_1 \ldots \mathbf{x}_N, \mathbf{w}) = \prod_{n=1}^{N} (\mu_n)^{y_n} (1-\mu_n)^{(1-y_n)}$$

Log likelihood

$$LL(\mathbf{w} : D) = \sum_{n=1}^{N} \{ y_n \log \mu_n + (1-y_n) \log(1-\mu_n) \}$$

We need to find $\mathbf{w}$ that maximizes log likelihood

Maximum likelihood estimation of $\mathbf{w}$

$$LL(\mathrm{w}:D) = \sum_{n=1}^{N} \left\{ y_n \log \mu_n + (1-y_n)\log(1-\mu_n) \right\}$$

$$\frac{\partial LL(\mathrm{w}:D)}{\partial w_i} = \sum_{n=1}^{N} \left(\frac{y_n}{\mu_n} - \frac{(1-y_n)}{(1-\mu_n)} \right) \left(\frac{\partial \mu_n}{\partial \eta_n} \right) \left(\frac{\partial \eta_n}{\partial w_i} \right) K(\mathrm{x}_i, \mathrm{x}_n)$$

$$= \sum_{n=1}^{N} \left(\frac{y_n - \mu_n}{\mu_n(1-\mu_n)} \right) \mu_n(1-\mu_n) K(\mathrm{x}_i, \mathrm{x}_n)$$

$$= \sum_{n=1}^{N} (y_n - \mu_n) K(\mathrm{x}_i, \mathrm{x}_n)$$

Regularized kernel logistic regression

- Maximum likelihood models can over-fit training data
- We can *regularize* the models by imposing a penalty in the estimation criterion that encourages $||\mathbf{w}||$ to remain small
- Maximum penalized log likelihood criterion:

$$LL(\mathrm{w}:D,\lambda) = \sum_{i=1}^{N} \log p(y_i \mid \mathrm{x}_i;\mathrm{w}) - \frac{\lambda}{2} \|\mathrm{w}\|^2$$

where larger values of λ impose stronger regularization.

$$w_i \leftarrow w_i + \rho \frac{\partial LL(\mathrm{w}:D)}{\partial w_i} - \lambda w_i$$

$$w_i \leftarrow w_i + \sum_{n=1}^{N} (y_n - \mu_n) K(\mathrm{x}_i, \mathrm{x}_n) - \lambda w_i$$