Support Vector and Kernel Methods for Pattern Recognition

Nello Cristianini
BIOwulf Technologies
nello@support-vector.net
http://**www.support-vector.net/tutorial.html**

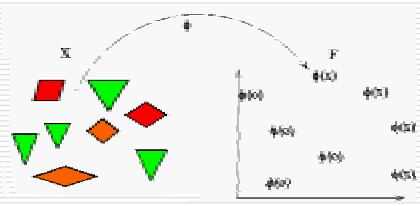
PSB 2002

A Little History

- Support Vector Machines (SVM) introduced in COLT-92 (conference on learning theory) greatly developed since then.
- Result: a class of algorithms for Pattern Recognition (Kernel Machines)
- Now: a large and diverse community, from machine learning, optimization, statistics, neural networks, functional analysis, etc. etc
- Centralized website: www.kernel-machines.org
- Textbook (2000): see www.support-vector.net

Basic Idea

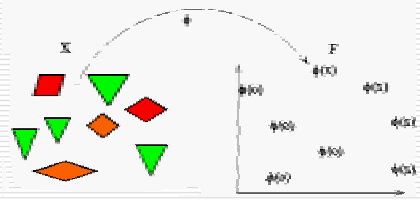
- Kernel Methods work by embedding the data into a vector space, and by detecting linear relations in that space
- Convex Optimization, Statistical Learning Theory, Functional Analysis are the main tools



www.support-vector.net

Basic Idea

- "Linear relations": can be regressions, classifications, correlations, principal components, etc.
- If the feature space chosen suitably, pattern recognition can be easy



www.support-vector.net

General Structure of Kernel-Based Algorithms

- Two Separate Modules:

Learning Module

A learning algorithm: performs the learning In the embedding space

Kernel Function

A kernel function: takes care of the embedding

Overview of the Tutorial

- Introduce basic concepts with extended example of Kernel Perceptron
- Derive Support Vector Machines
- Other kernel based algorithms (PCA;regression; clustering;…)
- Bioinformatics Applications

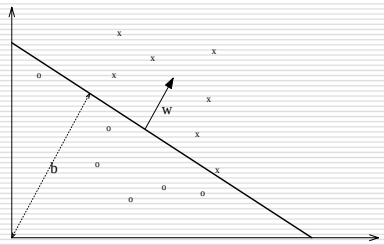
Just in case …

- Inner product between vectors

$$\langle \bar{x}, \bar{z} \rangle = \sum_i x_i z_i$$

- Hyperplane:

$$\langle w, x \rangle + b = 0$$



Preview

- Kernel methods exploit information about the inner products between data items
- Many standard algorithms can be rewritten so that they only require inner products between data (inputs)
- Kernel functions = inner products in some feature space (potentially very complex)
- If kernel given, no need to specify what features of the data are being used

Basic Notation

- Input space $x \in X$
- Output space $y \in Y = \{-1,+1\}$
- Hypothesis $h \in H$
- Real-valued: $f: X \to \mathrm{R}$
- Training Set $S = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots\}$
- Test error ε
- Dot product $\langle x, z \rangle$

Basic Example: the Kernel-Perceptron

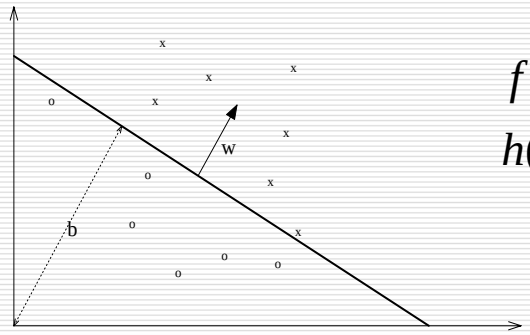
- We will introduce the main ideas of this approach by using an example: the simplest algorithm with the simplest kernel
- Then we will generalize to general algorithms and general kernels

Perceptron

- Simplest case: classification. Decision function is a hyperplane in input space
- The Perceptron Algorithm (Rosenblatt, 57)
- Useful to analyze the Perceptron algorithm, before looking at SVMs and Kernel Methods in general

Perceptron

- Linear Separation of the input space



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

$$h(x) = sign(f(x))$$

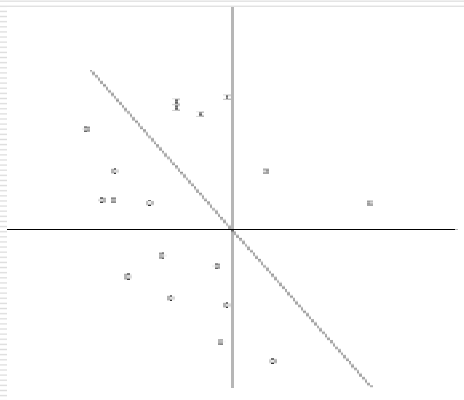
Perceptron Algorithm

Update rule (ignoring threshold):

- if $y_i(\langle w_k, x_i \rangle) \le 0$

  then $w_{k+1} \leftarrow w_k + \eta y_i x_i$

  $k \leftarrow k+1$



Observations

- Solution is a linear combination of training points $w = \sum \alpha_i y_i x_i$

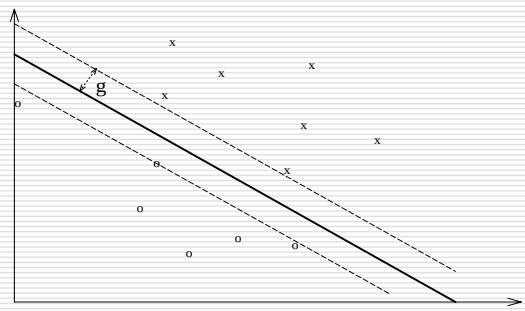
  $\alpha_i \ge 0$

- Only used informative points (mistake driven)
- The coefficient of a point in combination reflects its 'difficulty'

Observations - 2

- Mistake bound:

$$M \le \left(\frac{R}{\gamma}\right)^2$$



- coefficients are non-negative
- possible to rewrite the algorithm using this alternative representation

Dual Representation

IMPORTANT CONCEPT

The decision function can be re-written as follows:

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

$$w = \sum \alpha_i y_i x_i$$

Dual Representation

- And also the update rule can be rewritten as follows:
- If $y_i \sum_j \alpha_j y_j \langle x_j, x_i \rangle + b \leq 0$

  then $\alpha_i \leftarrow \alpha_i + \eta$
- Note: in dual representation, data appears only inside dot products

Duality: First Property of SVMs

- DUALITY is the first feature of Support Vector Machines (and KM in general)
- SVMs are Linear Learning Machines represented in a dual fashion

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

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

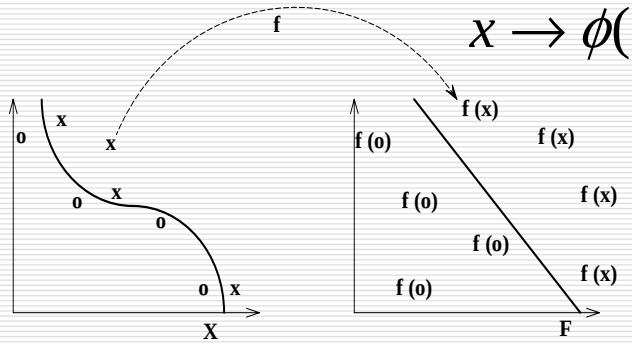
Limitations of Perceptron

- Only linear separations
- Only defined on vectorial data
- Only converges for linearly separable data

Learning in the Feature Space

- Map data into a feature space where they are linearly separable

$$x \rightarrow \phi(x)$$



Trick

- Often very high dimensional spaces are needed
- We can save computation by not explicitly mapping the data to feature space, but just working out the inner product in that space
- We will call this *implicit mapping*
- (many algorithms only need this information to work)

Kernel-Induced Feature Spaces

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

$$f(x) = \sum \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

- The dimensionality of space F not necessarily important. May not even know the map ϕ

Kernels

IMPORTANT CONCEPT

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

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

- Given a function K, it is possible to verify that it is a kernel

Kernels

- One can use LLMs in a feature space by simply rewriting it in dual representation and replacing dot products with kernels:

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

Example: Polynomial Kernels

$$x = (x_1, x_2);$$

$$z = (z_1, z_2);$$

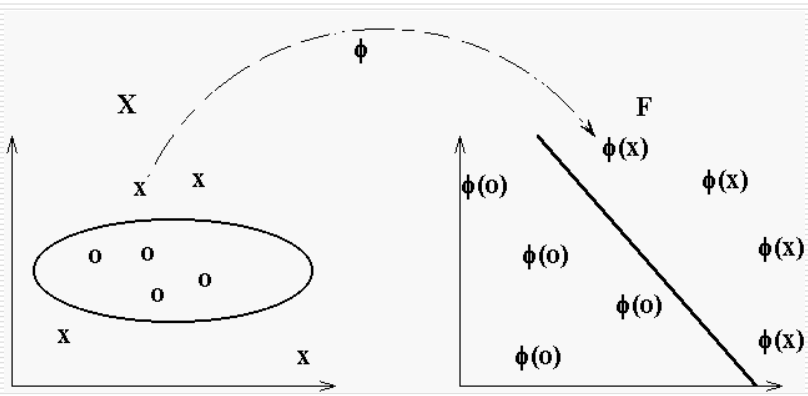
$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 =$$

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

$$= \langle \phi(x), \phi(z) \rangle$$

Example: Polynomial Kernels



The Kernel Matrix

IMPORTANT CONCEPT

- (aka the Gram matrix):

K=

K(1,1)	K(1,2)	K(1,3)	...	K(1,m)
K(2,1)	K(2,2)	K(2,3)	...	K(2,m)
...	...	...	...	...
K(m,1)	K(m,2)	K(m,3)	...	K(m,m)

The Kernel Matrix

- The central structure in kernel machines
- Information 'bottleneck': contains all necessary information for the learning algorithm
- Fuses information about the data AND the kernel
- Many interesting properties:

Mercer's Theorem

- The kernel matrix is Symmetric Positive Definite
  (has positive eigenvalues)
- Any symmetric positive definite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space

Mercer's Theorem

- Eigenvalues expansion of Mercer's Kernels:

$$K(x_1, x_2) = \sum_i \lambda_i \phi_i(x_1) \phi_i(x_2)$$

- The features are the eigenfunctions of the integral operator

$$(Tf)(x) = \int_X K(x, x') f(x') dx'$$

Examples of Kernels

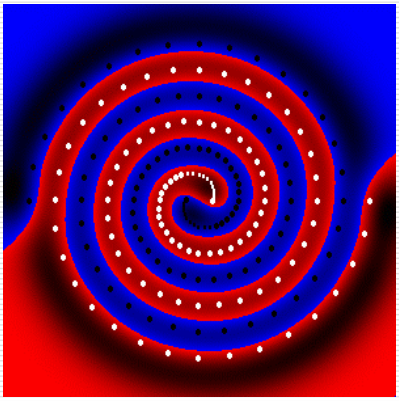
- Simple examples of kernels are:

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

$$K(x,z) = e^{-\|x-z\|^2/2\sigma}$$

Example: the two spirals

- Separated by a hyperplane in feature space (gaussian kernels)



Making kernels

- From kernels (see closure properties): can obtain complex kernels by combining simpler ones according to specific rules

Closure properties

- List of closure properties:

  if K_1 and K_2 are kernels, and $c>o$

$$K(x,z) = c \cdot K_1(x,z)$$
$$K(x,z) = c + K_1(x,z)$$
$$K(x,z) = K_1(x,z) + K_2(x,z)$$
$$K(x,z) = K_1(x,z) \cdot K_2(x,z)$$

- Then also K is a kernel

$$\forall f : X \to \Re$$
$$K(x,z) = f(x) \cdot f(z)$$

Some Practical Consequences

- if *K1* and *K2* are kernels, and *c>o d>0 integer*

$$K(x,z)=(K_1(x,z)+c)^d$$

$$K(x,z)=\exp\left(\frac{K_1(x,z)}{\sigma^2}\right)$$

$$K(x,z)=\exp\left(-\frac{K_1(x,x)+K_1(z,z)-2K_1(x,z)}{2\sigma^2}\right)$$

- Then also K is a kernel

$$K(x,z)=\frac{K_1(x,z)}{\sqrt{K_1(x,x)K_1(z,z)}}$$

Making kernels

- From features:
  start from the features, then obtain the kernel.
  Example: the polynomial kernel, the string kernel, …

Learning Kernels

- From data:
- either adapting parameters in a parametric family
- or modifying the kernel matrix (as seen below)
- Or training a generative model, then extract kernel as described before

Second Property of SVMs:

SVMs are Linear Learning Machines, that

- Use a dual representation

AND

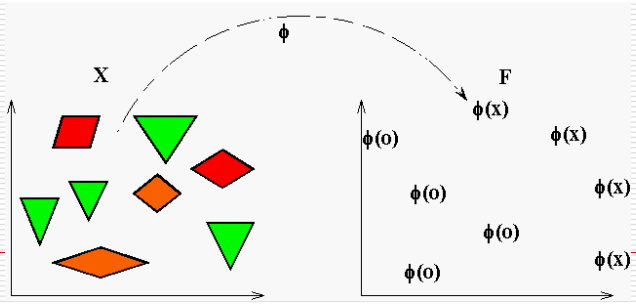
- Operate in a kernel induced feature space $f(x) = \sum \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$

(that is:

is a linear function in the feature space implicitely defined by K)

Kernels over General Structures

- Haussler, Watkins, etc: kernels over sets, over sequences, over trees, etc.
- Applied in text categorization, bioinformatics, etc



A bad kernel …

- … would be a kernel whose kernel matrix is mostly diagonal: all points orthogonal to each other, no clusters, no structure …

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

No Free Kernel

IMPORTANT CONCEPT

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

Other Kernel-based algorithms

- Note: other algorithms can use kernels, not just LLMs (e.g. clustering; PCA; etc). Dual representation often possible (in optimization problems, by Representer's theorem).

The Generalization Problem

NEW TOPIC

- The curse of dimensionality: easy to overfit in high dimensional spaces
  (=regularities could be found in the training set that are accidental, that is that would not be found again in a test set)
- The SVM problem is ill posed (finding one hyperplane that separates the data: many such hyperplanes exist)
- Need principled way to choose the best possible hyperplane

The Generalization Problem

- Many methods exist to choose a good hyperplane (inductive principles)
- Bayes, statistical learning theory / pac, MDL, …
- Each can be used, we will focus on a simple case motivated by statistical learning theory (will give the basic SVM)

Statistical (Computational) Learning Theory

- Generalization bounds on the risk of overfitting (in a p.a.c. setting: assumption of I.I.d. data; etc)
- Standard bounds from VC theory give upper and lower bound proportional to VC dimension
- VC dimension of LLMs proportional to dimension of space (can be huge)

Assumptions and Definitions

- distribution D over input space X
- train and test points drawn randomly (I.I.d.) from D
- training error of h: fraction of points in S misclassifed by h
- test error of h: probability under D to misclassify a point x
- VC dimension: size of largest subset of X shattered by H (every dichotomy implemented)

VC Bounds

$$\varepsilon = \tilde{O}\left(\frac{VC}{m}\right)$$

VC = (number of dimensions of X) +1

Typically VC >> m, so not useful

Does not tell us which hyperplane to choose

Margin Based Bounds

$$\varepsilon = \tilde{O}\left(\frac{(R/\gamma)^2}{m}\right)$$

$$\gamma = \min_i \frac{y_i f(x_i)}{\|f\|}$$

Note: also compression bounds exist; and online bounds.

Margin Based Bounds

IMPORTANT CONCEPT

- (The worst case bound still holds, but if lucky (margin is large)) the other bound can be applied and better generalization can be achieved:

$$\varepsilon = \tilde{O}\left(\frac{(R/\gamma)^2}{m}\right)$$

- Best hyperplane: the maximal margin one
- Margin is large is kernel chosen well

Maximal Margin Classifier

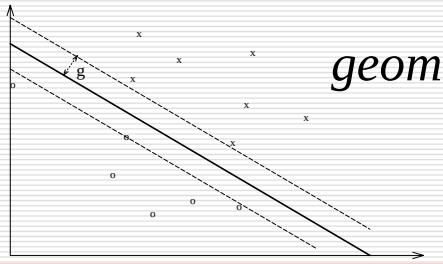
- Minimize the risk of overfitting by choosing the maximal margin hyperplane in feature space
- Third feature of SVMs: maximize the margin
- SVMs control capacity by increasing the margin, not by reducing the number of degrees of freedom (dimension free capacity control).

Two kinds of margin

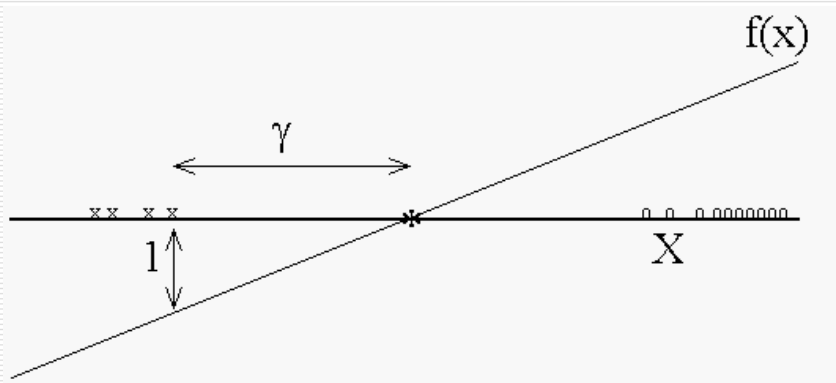
- Functional and geometric margin:

$$funct = \min y_i f(x_i)$$

$$geom = \min \frac{y_i f(x_i)}{\|f\|}$$



Two kinds of margin

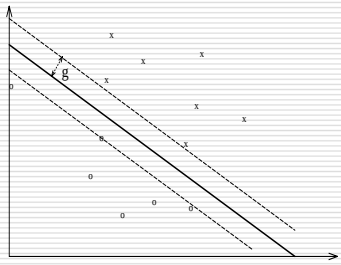


www.support-vector.net

Max Margin = Minimal Norm

Distance between

The two convex hulls



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

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

$$\langle w, (x^+ - x^-) \rangle = 2$$

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

www.support-vector.net

The primal problem

IMPORTANT STEP

- Minimize: $\langle w, w \rangle$

  subject to: $y_i \left[\langle w, x_i \rangle + b \right] \geq 1$

Optimization Theory

- The problem of finding the maximal margin hyperplane: constrained optimization (quadratic programming)
- Use Lagrange theory (or Kuhn-Tucker Theory)
- Lagrangian: $\frac{1}{2}\langle w, w \rangle - \sum \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right]$

  Lagrange Multipliers: $\alpha_i \geq 0$

From Primal to Dual

$$\frac{1}{2}\langle w, w\rangle - \sum \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right]$$

$$\alpha_i \geq 0$$

Differentiate and substitute:

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

$$\frac{\partial L}{\partial w} = 0$$

From Primal to Dual

$$\frac{\partial L}{\partial w} = w - \sum y_i \alpha_i x_i = 0$$

$$\frac{\partial L}{\partial b} = \sum y_i \alpha_i = 0$$

$$w = \sum y_i \alpha_i x_i$$

$$\sum y_i \alpha_i = 0$$

The Dual Problem

IMPORTANT STEP

$$\frac{1}{2}\langle w,w\rangle - \sum \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right]$$

$$\alpha_i \geq 0$$

PRIMAL

$$w = \sum y_i \alpha_i x_i$$

$$\sum y_i \alpha_i = 0$$

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

$$\alpha_i \geq 0$$

DUAL

$$\sum \alpha_i y_i = 0$$

Dual

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

$$\alpha_i \geq 0$$

$$\sum \alpha_i y_i = 0$$

Notice:
quadratic function
linear equality constraint coming from optimizing *b*
Positive quadrant

PROPERTIES OF THE SOLUTION

IMPORTANT CONCEPT

Convexity

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

$$\alpha_i \geq 0$$

$$\sum \alpha_i y_i = 0$$

- This is a Quadratic Optimization problem: convex, no local minima (second effect of Mercer's conditions)
- Solvable in polynomial time …
- (convexity is another fundamental property of SVMs)

PROPERTIES OF THE SOLUTION

Kuhn-Tucker Theorem

- Hyperplane is linear combination of training vectors
- KKT conditions:
- Sparseness: only the points nearest to the hyperplane (margin = 1) have positive weight
- They are called support vectors

$$w = \sum \alpha_i y_i x_i$$

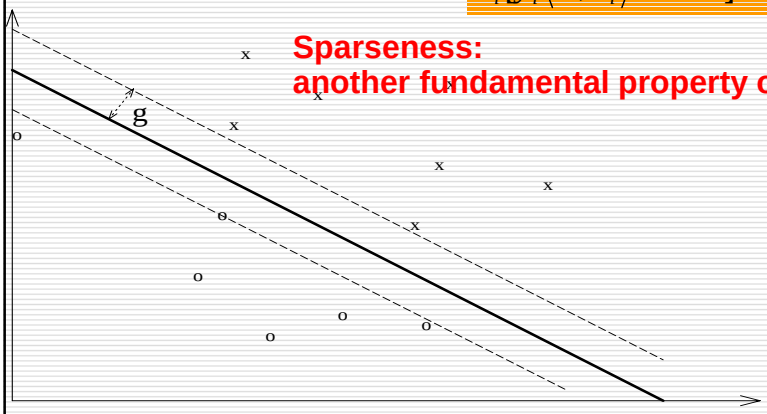
$$\alpha_i \left[y_i \langle w, x_i \rangle + b - 1 \right] = 0, \forall i$$

KKT Conditions Imply Sparseness

PROPERTIES OF THE SOLUTION

$$\alpha_i \left[y_i \langle w, x_i \rangle + b - 1 \right] = 0, \forall i$$

Sparseness:
another fundamental property of SVMs



Properties of SVMs Summary

PROPERTIES OF THE SOLUTION

- ✓ Duality
- ✓ Kernels
- ✓ Margin
- ✓ Convexity
- ✓ Sparseness

Soft Margin Classifier

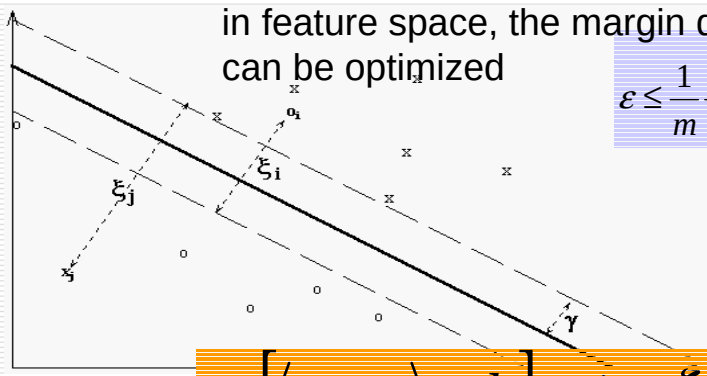
New Topic

- Problem: non-separable data (in feature space)
- We could always separate it with a 'finer' kernel, but that is not a good idea
- Better to have an algorithm that tolerates mislabeled points

Dealing with noise

In the case of non-separable data in feature space, the margin distribution can be optimized

$$\varepsilon \le \frac{1}{m}\frac{\left(R+\sqrt{\sum \xi^2}\right)^2}{\gamma^2}$$



$$y_i\left[\langle w, x_i\rangle + b\right] \ge 1-\xi_i$$

The Soft-Margin Classifier

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

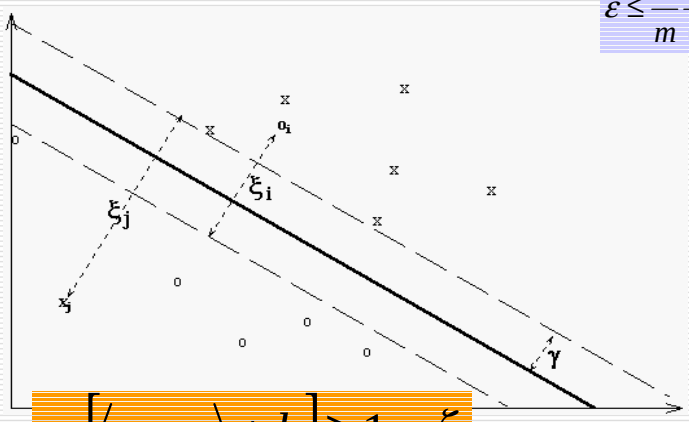
Or: $$\frac{1}{2}\langle w, w\rangle + C\sum_i \xi_i^{\,2}$$

Subject to: $$y_i\left[\langle w, x_i\rangle + b\right] \ge 1-\xi_i$$

www.support-vector.net

Slack Variables

$$\varepsilon \le \frac{1}{m}\frac{\left(R+\sqrt{\sum \xi^2}\right)^2}{\gamma^2}$$



$$y_i\left[\langle w, x_i\rangle + b\right] \ge 1-\xi_i$$

Soft Margin-Dual Lagrangian

- Box constraints

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

$$0 \le \alpha_i \le C$$

$$\sum_i \alpha_i y_i = 0$$

- Diagonal

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \frac{1}{2C}\sum \alpha_j\alpha_j$$

$$0 \le \alpha_i$$

$$\sum_i \alpha_i y_i \ge 0$$

Soft Margin

- Second problem equivalent to replacing K with $K+\lambda I$ $(\lambda=1/2C)$.
- Both formulations aim at reducing role of outliers, preventing (or discouraging) points from having too large a α
- Remember that in kernel perceptron, outliers would have unbounded α

Regression

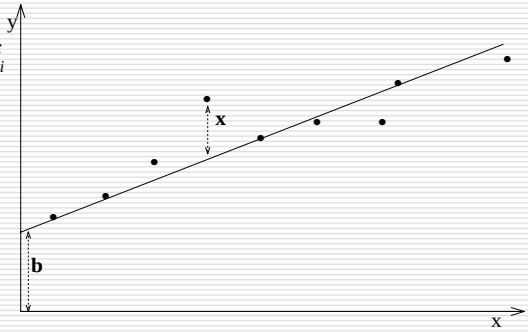
New Topic

- Now we see how the same ideas can be applied to regression
- Later also to PCA, clustering, etc.

Kernel Ridge Regression

$$\text{minimize } \lambda\|w\|^2 + \sum \xi_i^2$$

$$\text{subject to: } y_i - \langle w, x_i \rangle = \xi_i$$



Optimization problem

$$\text{minimize } L = \lambda \|w\|^2 + \sum \xi_i^2 + \sum \alpha_i \left(y_i - \langle w, x_i \rangle - \xi_i \right)$$

Imposing Optimality:

$$w = \frac{1}{2\lambda} \sum \alpha_i x_i$$

$$\xi_i = \frac{\alpha_i}{2}$$

We can now substitute them back in, and obtain a DUAL problem

The Dual

$$\text{maximize } W(\alpha) = \sum y_i \alpha_i - \frac{1}{4\lambda} \sum \alpha_i \alpha_j \langle x_i, x_j \rangle - \frac{1}{4} \sum \alpha_i^2$$

In matrix notation:

$$W(\alpha) = y'\alpha - \frac{1}{4\lambda} \alpha' K \alpha - \frac{1}{4} \alpha' \alpha$$

The Solution

$$-\frac{1}{2\lambda}K\alpha - \frac{1}{2}\alpha + y = 0$$

$$\alpha = 2\lambda(K + \lambda I)^{-1} y$$

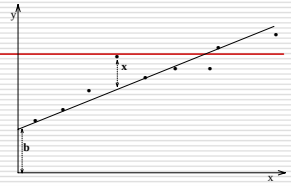
$$\rightarrow \quad f(x) = \langle w, x \rangle = y'(K + \lambda I)^{-1} k$$

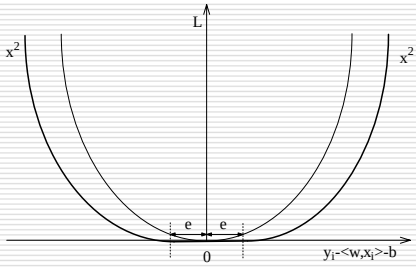
Ridge Regression

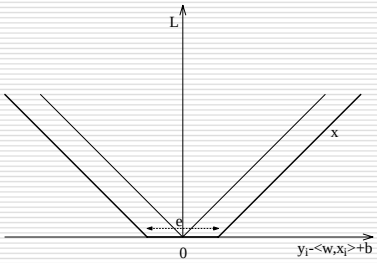
- So we can kernelize Ridge Regression
- More complex loss functions can be used than the square loss
- One problem: all alphas are positive. No more sparseness.
- Vapnik proposed the epsilon-insensitive loss, to obtain sparse solutions

SVM Regression

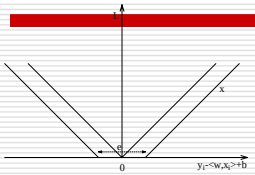
- Using the following loss:

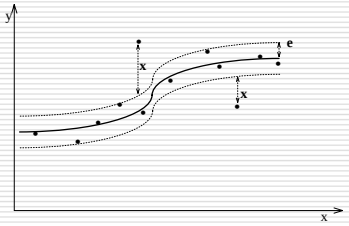


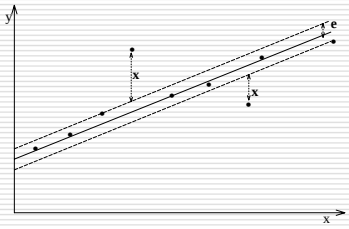




Insensitive Loss







If the points are close enough to the Function, they ‘pay no loss’.
If they are out of the insensitive region They pay in proportion (linear or quadratic)

This gives sparsity back:
points in the insensitive region will have zero alpha …

SVM Regression

PRIMAL

minimize:

$$\|w\|^2 + C\sum_i (\xi_i + \hat{\xi}_i)$$

subject to:

$$(\langle w, x_i \rangle + b) - y_i \le \varepsilon + \xi_i$$

$$y_i - (\langle w, x_i \rangle + b) \le \varepsilon + \hat{\xi}_i$$

$$\hat{\xi}_i, \xi_i \ge 0$$

DUAL

maximize:

$$\sum(\hat{\alpha}_i - \alpha_i) y_i - \varepsilon \sum(\hat{\alpha}_i + \alpha_i) - \frac{1}{2}\sum(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_i - \alpha_i) K(x_i, x_j)$$

subject to:

$$0 \le \alpha_i, \hat{\alpha}_i \le C$$

$$\sum(\hat{\alpha}_i - \alpha_i) = 0$$

Parzen Windows

New Topic

One can re-derive Parzen Windows from simple considerations…

Given two classes of points, find their centers of mass, and label new points according to the nearest center of mass

Parzen Windows

- Center of mass of a class is: $C = \frac{1}{n}\sum \phi(x_i)$

- Decision function will be:

$$\mathrm{sign}\left(\|x - c_-\|^2 - \|x - c_+\|^2\right)$$

$$\mathrm{sign}\left(\|x\|^2 + \|c_-\|^2 - 2\langle x, c_-\rangle - \|x\|^2 - \|c_+\|^2 + 2\langle x, c_+\rangle\right)$$

$$\mathrm{sign}\left(\underbrace{\|c_-\|^2 - \|c_+\|^2}_{2b} + 2\left(\langle x, c_-\rangle - \langle x, c_+\rangle\right)\right)$$

$$\alpha_i = \frac{1}{n_{class(i)}}$$

$$\mathrm{sign}\left(\sum \alpha_i y_i \langle x_i, x\rangle - b\right)$$

Example: from Schoelkopf's book

Other algorithms

New Topic

- K nearest neighbor

  ...

- K means clustering

  ...

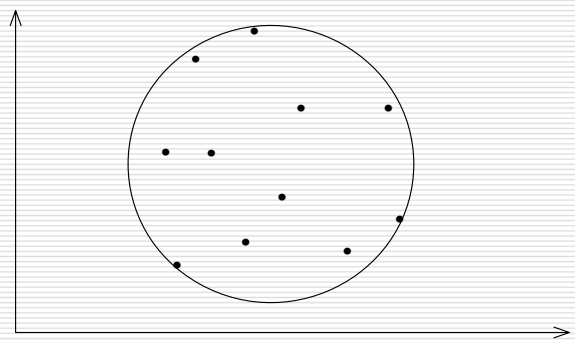
- Just use this:

$$\|\phi(x) - \phi(z)\|_2^2 = \langle \phi(x), \phi(x)\rangle + \langle \phi(z), \phi(z)\rangle - 2\langle \phi(x), \phi(z)\rangle =$$
$$= K(x,x) + K(z,z) - 2K(x,z)$$

Novelty Detection

New Topic

- Estimating the support of the distribution



Kernel PCA

Goal:
extract the principal components of a data vector.
Project it onto eigenvectors of dataset …

New Topic

- Standard PCA (primal, dual):

$\sum x_i = 0$ Assume data are centered

$C = \frac{1}{m} \sum_i x_i x_i^T$ Define covariance

$\lambda v = Cv$ Define eigenvectors of covariance

Kernel PCA

$\sum x_i = 0$

$C = \frac{1}{m}\sum_i x_i x_i^T$

$\lambda v = Cv$

Combining them, we obtain:

All solutions with Nonzero λ lie in the span of $X_1,\ldots,X_m$

$$\lambda v = Cv = \frac{1}{m}\sum_j \langle x_j, v\rangle x_j$$

$$\lambda\langle x_i, v\rangle = \langle x_i, Cv\rangle$$

$$\text{for all } i = 1,\ldots, m$$

We know that eigenvectors can be expressed
As lin comb of images of training vectors
We will characterize them by the corresponding
α vectors

Kernel PCA

$\lambda v = Cv = \frac{1}{m}\sum_j \langle x_j, v\rangle x_j$

$\lambda\langle x_i, v\rangle = \langle x_i, Cv\rangle$

for all $i = 1,\ldots, m$

$\sum \phi(x_i) = 0$

$C = \frac{1}{m}\sum_i \phi(x_i)\phi(x_i)^T$

$\lambda\, \mathbf{v} = \mathbf{Cv}$

$\lambda\langle \phi(x_n), v\rangle = \langle \phi(x_n), Cv\rangle$

$v = \sum_i \alpha_i \phi(x_i)$

Eigenvectors (with nonzero λ) can be written in dual form. The eigenvectors equation can be rewritten as follows:

$$\lambda \sum_i \alpha_i \langle \phi(x_n), \phi(x_i)\rangle = \frac{1}{m}\sum_i \alpha_i \left\langle \phi(x_n), \sum_j \phi(x_j)\langle \phi(x_j), \phi(x_i)\rangle \right\rangle$$

$m\lambda K\alpha = K^2\alpha$

$m\lambda\alpha = K\alpha$

$$K_{i,j} = \langle \phi(x_i), \phi(x_j)\rangle$$

Kernel PCA

- In order to find the dual coordinates α of the eigenvectors in feature space, we solve this problem:
  where α is a column vector of *m* entries (*m* = sample size)
- We also want to normalize the eigenvectors…

$$m\lambda\alpha = K\alpha$$

Kernel PCA - solution

- Normalize the eigenvectors: require that

$$\langle v^n, v^n \rangle = 1$$

$$1 = \sum_{i,j} \alpha^n{}_i \alpha^n{}_j \langle \phi(x_i), \phi(x_j) \rangle = \sum_{i,j} \alpha^n{}_i \alpha^n{}_j K_{ij}$$

$$1 = \langle \alpha^n, K\alpha^n \rangle = \lambda_n \langle \alpha^n, \alpha^n \rangle$$

- How to deal with a new point *x*:

$$\langle v^n, \phi(x) \rangle = \sum_i \alpha^n{}_i \langle \phi(x_i), \phi(x) \rangle$$

These are the principal components, or features of X in feature space

Summary of Kernel PCA

- K
- Center K
- Find eigenvectors of K
- Normalize alpha coefficients
- Extract PCs of new points by:

$$\left\langle v^n, \phi(x) \right\rangle = \sum_i \alpha^n{}_i \left\langle \phi(x_i), \phi(x) \right\rangle$$

Discussion ...

Like normal PCA, also kernel PCA has the property that the most information (variance) is contained in the first principal components (projections on eigenvectors)

Etc,. Etc

Spectral Methods

New Topic

- Semisupervised learning:
  given a partially labeled set, complete the labeling (TRANSDUCTION)
- Many possibilities:
  use the labels to learn a kernel,
  then use the kernel to label the data

Kernel Alignment

IMPORTANT CONCEPT

- Notion of similarity between kernels:
  Alignment (= similarity between Gram matrices)

$$A(K1,K2) = \frac{\langle K1,K2\rangle}{\sqrt{\langle K1,K1\rangle\langle K2,K2\rangle}}$$

Kernel Alignment

$$A(K1,K2) = \frac{\langle K1,K2\rangle}{\sqrt{\langle K1,K1\rangle\langle K2,K2\rangle}}$$

Where we use the Frobenius inner product:

$$\langle K_1,K_2\rangle = \sum_{i,j} K_1(i,j)K_2(i,j)$$

It is a similarity measure between kernel matrices.
That is: it depends on the sample.
A more general version can naturally be defined, using the input distribution. We could call the general one 'alignment', and the one defined here 'empirical alignment'.

Kernel Alignment Properties

- Ranges between -1 and 1 [0,1] for positive definite matrices
- Sharply concentrated around expected value E[A] (expectation is wrt random choice of sample): can reliably estimate it from one sample

$$P(|A(S)-E[A]|>\varepsilon) < f(S)e^{-(\varepsilon^2/m)}$$

where f is some function of the sample...
(omit for simplicity, pls check paper)

Used McDiarmid theorem to prove concentration

Kernel Selection or Combination

- Choose K_1 from a set so to optimize:

$$A(K_1, YY') = \frac{\langle K_1, YY' \rangle}{\sqrt{\langle K_1, K_1 \rangle \langle YY', YY' \rangle}}$$

- If set is convex, this leads to a convex optimization problem
- We will see one way to obtain convex family of kernels
- Before we will need another remark …

Interesting Analogy

$$K = \sum_i \lambda_i y_i y_i'$$

$$K = \sum_i \lambda_i v_i v_i'$$

Eigendecomposition of kernel matrix K

Thresholding the eigenvectors of *K* we can obtain many different labelings of the sample, and then we can consider the set of their convex combinations

Fixed K, choose best Y

- Choosing the labels: a clustering problem
- Optimizing over all possible labelings is a complex task
- We will relax the constraints, and approximate it with a convex problem

The ideal kernel

YY'=

1	1	-1	...	-1
1	1	-1	...	-1
-1	-1	1		1
...	...	...	...	...
-1	-1	1	...	1

Spectral Machines

- Can (approximately) maximize the alignment of a set of labels to a given kernel
- By solving this problem: $y = \arg\max \frac{yKy}{yy'}$

$$y_i \in \{-1,+1\}$$

- Approximated by principal eigenvector (thresholded) (see courant-fischer theorem)

Courant-Fischer theorem

- *A:* symmetric and positive definite,
- Principal Eigenvalue / Eigenvector characterized by:

$$\lambda = \max_v \frac{vAv}{vv'}$$

Optimizing Kernel Alignment

- Approximately find alignment set of labels by thresholding the first eigenvector of the kernel matrix
- More sophisticated methods exist (see website): using the Laplacian; or using SDP …

Using the alignment for Kernel Adaptation

- We can decompose the kernel matrix in eigen-components, then re-weight the coefficients so to optimize the alignment with the (available) labels

$$K = \sum_i \lambda_i v_i v_i'$$

It becomes a QP problem: Learning the kernel can be convex !

Recap: on Kernel-Based Algorithms

- We have seen:
  - Perceptron
  - SVMs (hard and soft margin)
  - Ridge Regression
  - Novelty detection
  - Kernel PCA
  - Spectral Clustering
- Also exist:
  - Fisher discriminant
  - ICA
  - Several Clustering algorithms
  - ...

Kernel Methods Recap

- They all work by:
  - Mapping the data into a space
  - Using algebra, optimization, statistics to extract patterns
  - Most of them: convex optimization problems
  - Designed to deal with high dimensional, noisy data
  - Computationally efficient; generalization-wise very effective

Modularity

IMPORTANT CONCEPT

- Any kernel-based learning algorithm composed of two modules:
  - A general purpose learning machine
  - A problem specific kernel function
- Any K-B algorithm can be fitted with any kernel
- Kernels themselves can be constructed in a modular way
- Great for software engineering (and for analysis)

BIOINFO APPLICATIONS

NEW TOPIC !

- In this last part we will review applications of Kernel Methods to bioinformatics problems
- Mostly Support Vector Machines, but also transduction methods, and others.
- Gene expression data; mass spectroscopy data; QSAR data; protein fold prediction;…

Diversity of Bioinformatics Data

- Gene Expression
- Protein sequences
- Phylogenetic Information
- Promoters
- Mass Spec
- QSAR

About bioinformatics problems

- Types of data:
  sequences (DNA; or proteins)
  gene expression data
  SNP; proteomics; etc. etc
- Types of tasks:
  diagnosis; gene function prediction
  protein fold prediction; drugs design; …
- Types of problems:
  high dimensional; noisy; very small or very large datasets; heterogeneous data; …

Gene Expression Data

- Measure expression level of thousands of genes simultaneously, in a cell or tissue sample
  (genes make proteins by producing RNA; a gene is expressed when its RNA is present…)
- Very high dimensionality; noise
- Can either characterize tissues or genes (transposing matrix)

Gene Function Prediction

- Predict functional roles for yeast genes based on their expression profiles
- Given set of 2467 genes, observed their expression under 79 conditions (from Eisen et al.)
- Assigned genes to 5 functional classes (from MIPS yeast genome database).
  TCA cycle; respiration; cytoplasmic ribosomes; proteasome; histones.
- SVM: learn to predict class based on expression profile.

Gene Function Prediction

- SVMs compared with 5 other algorithms, performed best (parzen windows; fisher discriminant; decision trees; etc).
- Also used to assign to their functional class 'new' genes.
- Often mistakes have biological interpretation …. See paper (and website).

- Brown, Grundy, Lin, Cristianini, Sugnet, Furey, Ares, Haussler, "Knowledge Based Analysis of Miroarray Gene Expression Data Using Support Vector Machines", PNAS
- www.cse.ucsc.edu/research/compbio

Gene Function Prediction

- Notice: not all functional classes can be expected to be predicted on the basis of expression profiles
- The 5 classes were chosen using biological knowledge: expected to show correlation.
- Also: chosen for control a class not expected to have correlation: helix-turn-helix.

Heterogeneous Information

- Diverse sources can be combined. An example.
- Phylogenetic Data obtained from comparison of a given gene with other genomes
- Simplest Phylogenetic Profile: a bit string in which each bit indicates whether the gene of interest has a close homolog in the corresponding genome
- More detailed: negative log of the lowest E value reported by BLAST in a search against a complete genome
- Merged with Expression data to improve performance in Function Identification

Heterogeneous Data

- Similar pattern of occurrence across species could indicate 1) functional link (they might need each other to function, so they occur together). Could also simply indicate 2) sequence similarity
- Used 24 genomes from the Sanger Centre website
- Again: only <u>some</u> functional classes can benefit from <u>this</u> type of data.
- Generalization improves, but mostly for effect 2): a way to summarize sequence similarity information

- Pavlidis, Weston, Cai, Grundy, "Gene Functional Classification from Heterogeneous Data", International Conference on Computational Molecular Biology, 2001

Cancer Detection

- Task: automatic classification of tissue samples
- Case study: ovarian cancer
- Dataset of 97808 cDNAs for each tissue !
  (each of which may or may not correspond to a gene)
- Just 31 tissues of 3 types: ovarian cancer; normal ovarian tissue; other normal tissues. (15 positive and 16 negatives)

- Furey, Cristianini, Duffy, Bednarski, Schummer, Haussler, "Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data" Bioinformatics

Ovarian Cancer

- Main goal: decide whether a given sample is cancerous or not
- Secondary goal: locate genes potentially responsible for classification
- Problem: overfitting due to curse of dimensionality

Results

- Cross validation experiments (l.o.o.).
- Located a consistently misclassified point. The sample was considered cancerous by the SVM (and dubious by humans that originally labelled it as OK). Re-labelled.
- The only non -ovarian tissue is also misclassified consistently. Removed.
- After its removal: perfect generalization
- Attempt to locate most correlated genes provides less interesting results (used Fisher score for ranking, independence assumption).
- Only 5 of the top 10 are actually genes, only 3 cancer related.

Protein Homology

- Special kernels can be designed for comparing protein sequences, based on HMMs
- The generative model used as 'feature extractor' for designing a kernel ('Fisher kernel').
- Successfully used to detect remote protein homology
- Jaakkola, Diekhans, Haussler "Using the Fisher Kernel Method to Detect Remote Protein Homologies", AAAI press

Promoters

- Similar technology used to classify genes based on the patterns in their regulatory region
- Task: identify co-regulated genes based on promoter sequences

- Pavlidis, Furey, Liberto, Haussler, Grundy, "Promoter Region-Based Classification of Genes", Pacific Symposium on Biocomputing, 2001

Promoters

- Simple way to represent promoters: presence of motifs that function as binding sites of TFs
- Small size and other problems make approach very noisy
- More abstract features: exploit presence of multiple copies and combinations of motifs; spacing between two motifs; sequence flanking the motifs; presence of more general – less conserved – patterns.
- Features of the promoter region not only the TFBSs motifs themselves.

Fisher Kernels

- Capture information about presence and relative position of motifs
- 1) build a motif-based HMM from a collection of TFBS motifs
- 2) extract Fisher kernel and use in in SVM
- 3) discriminate between a given set of promoters from co-regulated genes and a second set of negative example promoters
- Result: predicted coregulation of unannotated genes. Predictions validated with expression profiles or other annotation sources.

String Matching Kernels

- Different approach, very promising: dynamic programming method to detect similarity between strings
- So far: used in text categorization. Being tested on protein data.
- LATER MORE ON THIS
- Other work, with different kernels: detection of translation initiation sites.
- Lodhi, Cristianini, Watkins, Shawe-Taylor "String Matching Kernels for Text Categorization" NIPS 2000

More on Bioinformatics

- Different types of data,
  very noisy and from different sources
- Problem:
  How to combine them ?
- One possible answer:
  kernel combination ...

Transcription Initiation Site

- Parts of DNA are junk, others encode for proteins. They are transcribed into RNA and then translated into proteins
- The transcription starts at ATG; but not all ATGs are transcription initiation sites …
- Problem: predict if a given ATG is a TIS based on its neighbors ….

SVMs

- Encoding: window of 200 nucleotides each side around the candidate ATG
- Each nucleotide encoded with a 5 bits word (00001, 00010, 00100, 01000, 10000) (for A,C,G,T, and N – unknown).
- Comparisons of these 1000-dim bitstrings should reveal which ones contain actual TIS

Naïve Approach

- Linear kernels:
  $\langle x,z \rangle$

- Polynomial kernels:
  $\langle x,z \rangle^d$

Special Kernels

- Poly kernels consider all possible k-ples, even very distant ones
- We assume that only short range correlations matter

- We need a kernel that discards long range correlations

‘locality improved’ kernels

- First consider a window of length *2l+1* around each position. We will compare two sequences ‘locally’, by moving this window along them …

$$win_p(x,z) = \left(\sum_{j=-l}^{l} w_j match_{p+j}(x,z)\right)^{d_1}$$

$$K(x,z) = \left(\sum_{p=1}^{l} win_p(x,z)\right)^{d_2}$$

Notice: these are all kernel-preserving operations on basic kernels Hence the result is still a valid kernel. Weights chosen to penalize long range correlations.

TIS detection with Locality Improved kernels

- Performed better than polynomial kernels
- Better than best neural network (state of the art on that benchmark)

Engineering support vector machine kernels that recognize translation initiation sites, A. Zien, G. Ratsch, S. Mika, B. Scholkopf, T. Lengauer, and K.-R. Muller, BioInformatics, 16(9):799-807, 2000.

Protein Fold

- Problem is: given sequence of aminoacids forming a protein, predict which overall shape the molecule will assume
- Problem: defining the right set of features, the right kernel
- Work in progress

KDD 2001 Cup: Thrombin Binding

Data:

1909 known molecules, 42 actively binding to thrombin.

636 new compounds, unknown binding.

Each compound: 139,351 binary features of 3D structure.

(Data provided by DuPont Pharmaceuticals.)

Entrants: 114. (~10% used SVMs)

The winner: 68% prediction accuracy.

Weston, et al, Oct. 2001:

SVM, with transduction,+ feature selection.

82% prediction accuracy.

See more on the web…

- www.support-vector.net/bioinformatics.html
- (a non exhaustive list also attached to your handouts, just to give an idea of the diversity of applications)

Conclusions:

- Much more than just a replacement for neural networks
- General and rich class of pattern recognition methods
- Very effective for wide range of bioinformatics problems

Links, References, Further Reading:

Book on SVMs: www.support-vector.net
This tutorial: www.support-vector.net/tutorial.html
References:
www.support-vector.net/bioinformatics.html
Kernel machines website: www.kernel-machines.org
More slides: www.cs.berkeley.edu/~nello