Kernel Methods for Pattern Analysis

Nello Cristianini
nello@support-vector.net

- Bernhard has presented Support Vector Machines
- Three 'innovations' in that method:
  use of kernels
  use of optimization
  use of statistical learning theory
- Can we export these 3 ideas to other algorithms?
- → kernel methods!

In this talk…

- Review the main ideas of kernel based learning algorithms, and their application to pattern analysis tasks
- Demonstrate their flexibility by giving examples of the diverse types of data and tasks they can handle
- Present some recent results on learning kernels

Kernel Methods

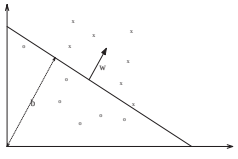
- rich family of '*pattern analysis*' algorithms, whose best known element is the Support Vector Machine
- very general task: given a set of data (any form, not necessarily vectors), find patterns (= any relations).
- (Examples of relations: classifications, regressions, principal directions, correlations, clusters, rankings, etc.…)
- (Examples of data: gene expression; protein sequences; heterogeneous descriptions of genes; text and hypertext documents; etc. etc.)

Basic Notation

- Given a set *X* (the input set), not necessarily a vector space…
- And a set *Y* (the output set) eg *Y={-1,+1}*
- Given a finite subset $S \subseteq (X \times Y)$ (usually: iid from an unknown distribution)
- Elements $(x_i, y_i) \in S \subseteq (X \times Y)$
- Find a function *y=f(x)* that 'fits' the data (minimizes some cost function, etc…)

Just in case …

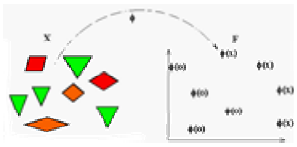
- Inner product between vectors
$$\langle \bar{x}, \bar{z} \rangle = \sum_i x_i z_i$$
- Hyperplane:
$$\langle w, x \rangle + b = 0$$



The Main Idea: $x \to \phi(x)$

- Kernel Methods work by:

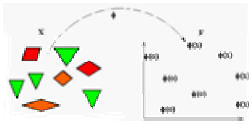
  1-embedding data in a vector space
  2-looking for (linear) relations in such space
- If map chosen suitably, complex relations can be simplified, and easily detected



Main Idea / two observations

- 1- Much of the geometry of the data in the embedding space (relative positions) is contained in all pairwise inner products*

  We can work in that space by specifying an inner product function between points in it (rather than their coordinates)
- 2- In many cases, inner product in the embedding space very cheap to compute



<x1,x1>	.	<x1,x2>	<x1,xn>
<x2,x1>	.	<x2,x2>	<x2,xn>
.	.	.	.
<xn,x1>	.	<xn,x2>	<xn,xn>

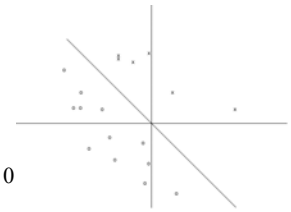
* Inner products matrix

Example: Linear Discriminant

- Data $\{x_i\}$ in vector space X, divided into 2 classes $\{-1,+1\}$
- Find linear separation: a hyperplane

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

- (Eg: the perceptron)



Dual Representation of Linear Functions

$$f(x) = w'x = \sum_{x_i \in S} \alpha_i x_i' x$$

$$w = \sum_{x_i \in S} \alpha_i x_i + \sum_{x_i \perp span(S)} \alpha_i x_i$$

$$f(x_j) = \sum_{x_i \in S} \alpha_i x_i' x_j + \sum_{x_i \perp span(S)} \alpha_i x_i' x_j = \sum_{x_i \in S} \alpha_i x_i' x_j + 0 = \sum_{x_i \in S} \alpha_i x_i' x_j$$

The linear function *f(x)* can be written in this form
Without changing its behavior on the sample

See Wahba's Representer's Theorem for more considerations

Dual Representation

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

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

- It only needs inner products between data points (not their coordinates!)
- If I want to work in the embedding space $x \to \phi(x)$ just need to know this: $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$

Pardon my notation:
x,w vectors, α,y scalars

Kernels

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

Kernels are functions that return inner products between the images of data points in some space.

By replacing inner products with kernels in linear algorithms, we obtain very flexible representations

Choosing K is equivalent to choosing Φ (the embedding map)

Kernels can often be computed efficiently even for very high dimensional spaces – see example

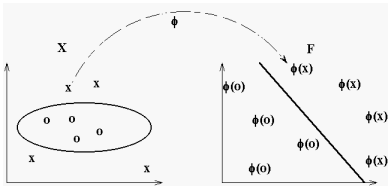
Classic Example Polynomial Kernel

$x = (x_1, x_2);$

$z = (z_1, z_2);$

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

Can Learn Non-Linear Separations



$$f(x) = \sum_i \alpha_i K(x_i, x)$$

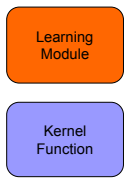
By combining a simple linear discriminant algorithm with this simple Kernel, we can learn nonlinear separations (efficiently).

More Important than Nonlinearity…

- Can naturally work with general, non-vectorial, data types !
- Kernels exist to embed sequences
  (based on string matching or on HMMs; see: haussler; jaakkola and haussler; bill noble; …)
- Kernels for trees, graphs, general structures
- Semantic Kernels for text, etc. etc.
- Kernels based on generative models
  (see phylogenetic kernels, by J.P. Vert)

The Point

- More sophisticated algorithms* and kernels** exist, than linear discriminant and polynomial kernels
- The idea is the same: *modular systems*, a general purpose **learning module**, and a problem specific **kernel function**

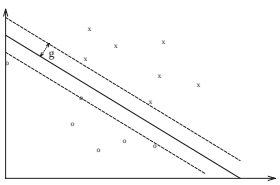


$$f(x) = \sum_i \alpha_i K(x_i, x)$$

*PCA, CCA, ICA, RR, Fisher Discriminant, TDλ, etc. etc.
** string matching; HMM based; etc. etc

Eg: Support Vector Machines

- *Maximal margin* hyperplanes in the embedding space
- Margin: distance from nearest point (while correctly separating sample)
- Problem of finding the optimal hyperplane reduces to Quadratic Programming (convex !) once fixed the kernel
- Extensions exist to deal with noise.



$$f(x) = \sum_i \alpha_i K(x_i, x)$$

Large margin bias motivated by statistical considerations (see Vapnik's talk)
leads to a convex optimization problem (for learning α)

A QP Problem

(we will need dual later)

$$\frac{1}{2}\langle w, w \rangle - \sum \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right]$$
$$\alpha_i \geq 0$$

PRIMAL

$$w = \sum y_i \alpha_i x_i$$
$$\sum y_i \alpha_i = 0$$

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$
$$\alpha_i \geq 0$$
$$\sum \alpha_i y_i = 0$$

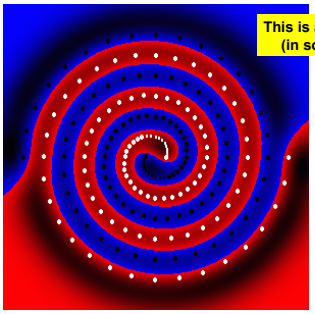
DUAL

Support Vector Machines

- No local minima:
  (training = convex optimization)
- Statistically well understood

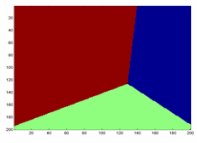
- Popular tool among practitioners
  (introduced in COLT 1992, by Boser, Guyon, Vapnik)
- State of the art in many applications…

Flexibility of SVMs…



Example: Voronoi Partitioning

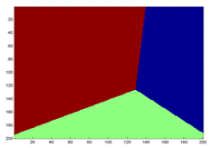
- Fix *k* centers, label test points according to label of *nearest* center
- Creates 'voronoi' partitioning of the input space (cells with linear boundaries)

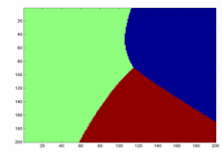


Voronoi Partitioning of feature space

- Use distance in feature space:

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





Examples of Applications…

- Remote protein homology detection…
  (HMM based kernels; string matching kernels; …)
- Text Categorization …
  (vector space representation + various types of semantic kernels; string matching kernels; …)
- Gene Function Prediction, Transcription Initiation Sites, etc. etc. …

Examples of Kernels

- Simple examples of kernels are:

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

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

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

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 $c>0$

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

- 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>0* *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

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

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

Remarks

- SVMs just an instance of the class of Kernel Methods
- Other types of linear discriminant can be kernelized
  (eg fisher, bayes, least squares, etc)
- Other types of linear analysis (other than 2-class discrimination) possible (eg PCA, CCA, novelty detection, etc) → NEXT !
- Kernel representation: efficient way to deal with high dimensionality
- Use well-understood linear methods in a non-linear way
- Convexity, concentration results, guarantee computational and statistical efficiency.

Kernel Methods

- General class, interpolates between statistical pattern recognition, neural networks, splines, *structural* (syntactical) pattern recognition, etc. etc
- We will see some examples and open problems…

Principal Components Analysis

- Eigenvectors of the data in the embedding space can be used to detect directions of maximum variance
- We can project data onto principal components by solving a (dual) eigenproblem…
- **We will use this later** for visualization of the embedding space: projecting data onto a 2dim plane

Kernel PCA

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

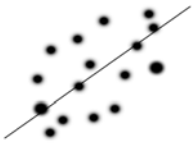
New Topic

- 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:

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

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

for all i = 1,..., m

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

The eigenvectors can be written as: $w=\Sigma\alpha_i x_i$

Problem: find the alphas

Kernel PCA

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

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

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

for all i = 1,..., m

$\sum \phi(x_i) = 0$

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

$\lambda \mathbf{v} = \mathbf{C}\mathbf{v}$

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

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

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:

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

Discussion …

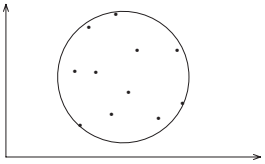
Like normal PCA, also kernel PCA has the property that the most information (variance) is contained in the first principal components (projections on eigenvectors)

Etc etc

We will use it just to visualize a 'slice' of the embedding space, later…

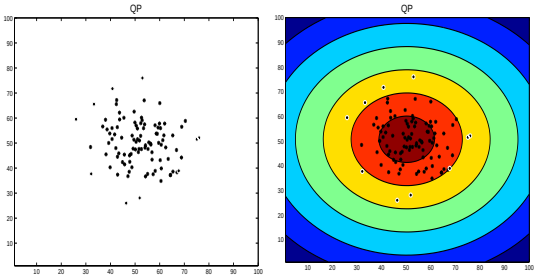
Novelty Detection

- **Another QP problem: find smallest sphere containing all the data (in the embedding space)**

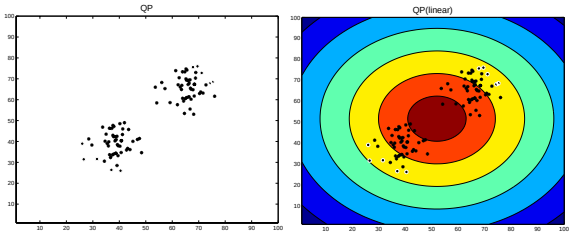


- **Similar: find small sphere that contains a given fraction of the points**

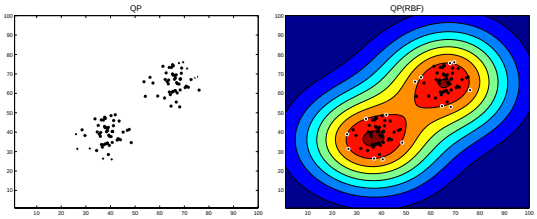
Novelty Detection: example



Example 2



Effect of Kernels



Smallest Sphere: Continued

- This method can be used to define a class of data (a subset of the input domain)
- Eg: if defined over set of symbol sequences, can be used to define/learn formal languages (see next) …
- (a task of syntactical pattern analysis)

A simple kernel for sequences

Consider a space with dimensions indexed by all possible finite substrings from alphabet A.

Embedding: if a certain substring *i* is present once in sequence *s*, then $\phi_i(s)=1$

Inner product: counts common substrings

Exponentially many coordinates, but can compute the inner product in such space in LINEAR time by using a recursive relation

Sequence-Kernel-recursion

It starts by computing kernels of small prefixes, then uses them for larger prefixes, etc

$$K(s,\Omega)=1$$

$$K(sa,t)=K(s,t)+\sum_i K(s,t[1:i-1])[t_i=a]$$

- Where *s,t* are generic sequences, *a* is a generic symbol, Ω is the empty sequence, …
- Analogous relation for *K(s,ta)* by symmetry…
- Dynamic programming techniques evaluate this in linear time !

Example

$$K(s,\Omega)=1$$

$$K(sa,t)=K(s,t)+\sum_i K(s,t[1:i-1])[t_i=a]$$

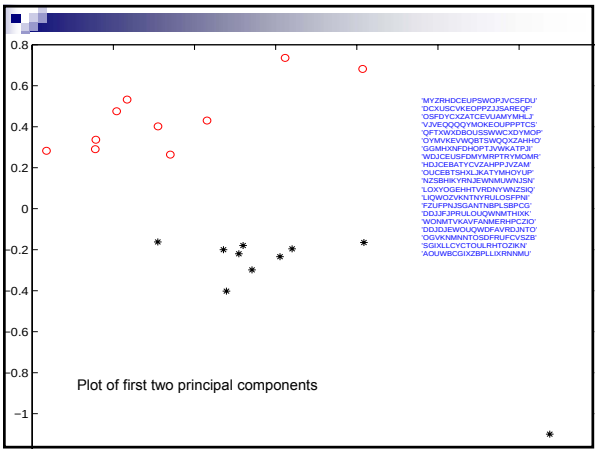
S=**ABBCBBC**A

T=**BBABBCAB**

Dynamic programming:
stored in table all the kernels for all smaller prefixes
The computation of the sum is just a matter of looking them up

More advanced sequence kernels…

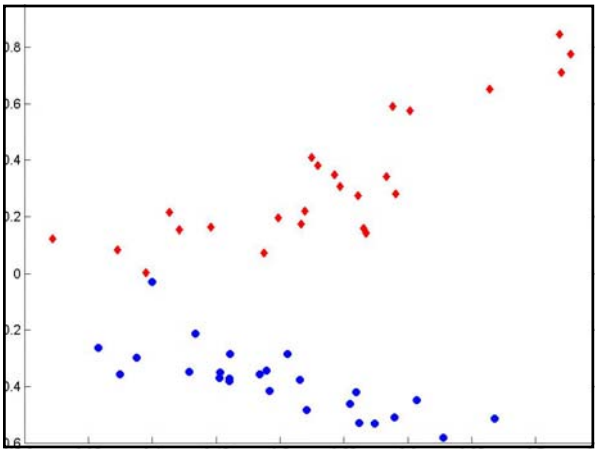
- Compare substrings of length *k*, and tolerate insertions …
- Similar (but more complicated) recursions…
- Demonstrated on sets of strings (generated by 2 different markov sources)



2 markov processes…

'VBNEUJVQRNEVIC,JWRVINE'
'SWMPURKPZPZWZUJKDFJZ'
'FESPZUJEUJVDHEWDFDVIC'
'TJWBQSTLMKBHRRBOMGMGV'
'HRPYEVBQBFNMCGNEMOIQW'
'AQXRZUCNEVFJVFJVBOICG'
'HHHHRBQWNESNEWFGVBKGD'
'NMYTJVDODHHHBHBZPZWNM'
'RPUEVEWTCXDDKPBKPFJUY'
'MDDCFEMYUJVQBOGNEWFDH'
'OIINTDODVQXCMKBQXOIBF'
'CLVFDVIIFDVFMCJVDFDDD'
'RBHBQSPZUESTPZWFMKBSP'
'EWBKLJWRGLBZHHGAQBSNA'
'XRBHOGUESNEXRQBOMOGVB'
'OESPFBODDMDVOIIFMDKBB'
'VDOGVIIBNEVWDHRNEUDAQ'
'QRRNESWZHBHESTLBODHEV'
'SPZIXRZIJKBZIJEVIBHBOZ'
'JVBSTBHBSPYTBHRNESWFG'
'HBFJKGTJUEXCXRBIJVLJV'
'BHOEIIBHRZESTBKGAIIJZ'
'KBOEXCJUESWFEVBKDHHHB'
'NEXKURKBHRBHHHEWTJEVD'
'DLLMCMQBHBKPYKLAAQQBH'

'LMMUCXWLFHOFVIOFNPHLG'
'DMMPBZVJKIGNFMZWBEYMB'
'RTRAACUXSACLMMZWKGLGO'
'IBQSRHOKRTTSTFGVJKEHI'
'VRHJFMZXSRTQFNPFHNPHJ'
'RLFHVJQFZZVJTFMBZZXBZ'
'LWKCLFVRAWBEHVJKLFVNP'
'WBENPJQFMMZUXMUXTROOF'
'PHOFHLMZUCLMZWCVJQFGN'
'ACVJFZWKCLQIUOBACGENF'
'SRHOFJXWBEIGNPHLYYPJY'
'LMBEYYCTUKLQDMLQMBEYP'
'NPHVIOFGEIJYNPJWLFWSA'
'MZAROKZVXMPHLBENPJXWC'
'YVXVJWIUCMMBEHLMPHLMZ'
'KXSCYRHLFWCPHOFGVRTFW'
'JWKGVXWOIGEAGROFGENPJ'
'LMPFZVINPHLMZWBEHJXMB'
'JKZZUCVNPKAOBJQDBENJF'
'YYMBGDXBENPHOFGECYYMZ'
'ZVNYMMBADMPJWLMZUSAWD'
'KVJKGOFMBEYYVJQFTSVJY'
'DXMZWOFGEHNFZUGEYYRTT'
'GDFHJFZVNYPFMUXVNPAGE'
'ZWKIOFTVRRHOWCXOFNPJX'

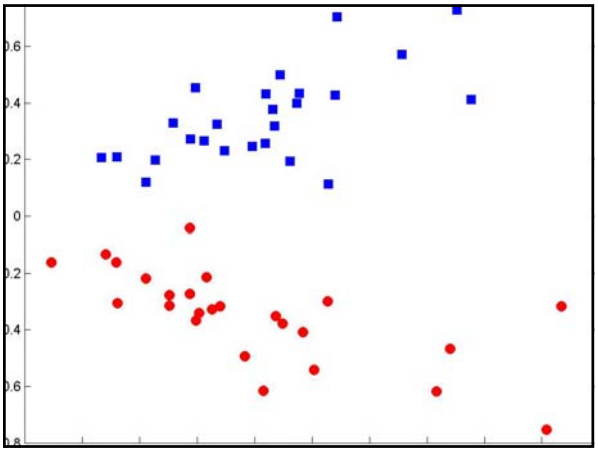


One more time …

- 'JXQXFUHKTOIUSDQXDOMIL'
- 'LNOMDWAXFWAYRYRELVQHK'
- 'OMPGHKTIXGJXFNAWAYZCV'
- 'HZYJXFWWAQXELQXPFNZOI'
- 'TIREGJUSVSJUASFPMXFKZ'
- 'DGTAELZJUAEGMHAGTIUIU'
- 'ETIWHAEVPJZYRYRELZKXF'
- 'FUHZJZNNAGUMYRMPFPJUL'
- 'LVHZYRDCDQBZCQIUAXQBP'
- 'UAQEGNEVTAGTTAYRELHDG'
- 'THATTTPMOMILXFPVVHWWW'
- 'ELWWTPGNAXFPGJXFNLZYI'
- 'NEFKKTNWHZJZYILXFKOON'
- 'JMXFTATHKTPVPGHAYRYZJ'
- 'WPHAXEGHKOMETTONPGJFL'
- 'FPGHJXELWWBPGTAYRMPMX'
- 'CFWBZJHDNQVSGJUYRDOIY'
- 'IWIULZNNALZOOOOIRJZJX'
- 'GJUHZOXJUILHAYDWHMRYR'
- 'LVJHZBYRYRHJUSJZYRJZY'
- 'EVJULQYRDCVQELWKXFNII'
- 'FKTTINILVJXFNIYEFKXEG'
- 'TAKJXFLWBPGTIYRMMMRDO'
- 'PJUAYRMPSFMMPJZYREVJZ'
- 'KTAYRDBZYRDJUATIWGTPJ'

- 'TQZDJHMUIVIIVKAFXYFRL'
- 'OFBUIGRTQJOFCGRZBBQNT'
- 'EASNSNEGRLUVKLZMUIWDP'
- 'IVKASNSNGRTUNSFRKVIVK'
- 'TUMNRUXHSHFEAYOZDAFRH'
- 'IVXHFBHBIKHBTGIIIVXHMJ'
- 'SMHMLICGRKLZWPEVDCKAFF'
- 'DUNTIWDSWQXYZBHCIGRKH'
- 'JYDAYDJYDDJTIIVXWFAIv'
- 'SWFXBUXWFEGPKGRTIVXYD'
- 'XJHBBKGRLZZZDSMUCLXWR'
- 'EGZMRQJHSMUXJUNERLMZM'
- 'MUCIVKASHCIWQJUNNTIVO'
- 'PKYDSMCBGPKAFXBTGWRQU'
- 'YDAKTQUXBKAKBGRLAKTGZ'
- 'GRKBKHCDPKBKHGRHSHGWR'
- 'BHCGWZYQNTJNEOFROAIIG'
- 'QXPKHMEGZQXWRKAKBHMUV'
- 'UIWFCKYDJTGPQAFOIGRUV'
- 'DDASNRQCKHCKYDAYDAIVO'
- 'AYUXYYQXHMHCVIIIVKYU'
- 'VDAIWFFEGRKHLUXBGRUSM'
- 'TGRQZBKBHMUBBTINTGRLU'
- 'CGIWDAFUIWPQABBGWFFCE'
- 'TFZZMUVXCLOQLZCGRQNEO'

www.support-vector.net/nello.html



Details…

- For each class, the markov processes given by random transition matrices, with 2 dominant transitions each (randomly chosen)

www.support-vector.net/nello.html

(CCA) Canonical Correlation Analysis

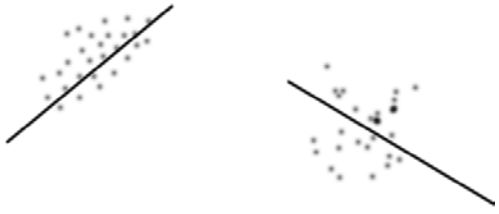
- 1- correlation between 2 random variables:
  (assume observations 1…m performed)

$$a = (a_1, a_2, ..., a_m)$$
$$b = (b_1, b_2, ..., b_m)$$
$$\text{zero mean, } \sigma = 1$$

$$C(a,b) = \frac{\langle a,b \rangle}{\sqrt{\langle a,a \rangle}\sqrt{\langle b,b \rangle}}$$

- 2- here: random variable is projection of data point *x* onto a given direction *w*

www.support-vector.net/nello.html



CCA

- Given 2 sets of points in bijection (or a set of pairs of points, generally in different spaces X1 and X2)
- Find a direction w1 in X1 and w2 in X2, such that the projection of the datasets onto the respective directions is maximally correlated

www.support-vector.net/nello.html

Formally…

$$\max_{w_1,w_2} C(\langle w_1, x^i{}_1\rangle, \langle w_2, x^i{}_2\rangle)$$

- Maximize correlation of random variables <w1,x1> and <w2,x2> over choice of w1 and w2
- This leads to a generalized eigenvalue problem

Generalized eigen-problem

- This leads to a generalized eigenvalue problem, both in the primal and in the dual…
  - Av=λBv
- We skip all the details, we give directly the dual problem (leading to the α coefficients for the directions w1 and w2)

CCA

Let *x* and *y* be random variables with zero mean
And x=w_x'x and y=w_y'y be their projections in the directions w_x and w_y

$$C = \begin{pmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{pmatrix} = E\left\{ \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}^T \right\}$$

$$\rho = \frac{E\{xy\}}{\sqrt{E\{xx\}E\{yy\}}} = \frac{E\{\hat{w}_x{}^T x y^T \hat{w}_y\}}{\sqrt{E\{\hat{w}_x{}^T x x^T \hat{w}_x\}E\{\hat{w}_y{}^T y y^T \hat{w}_y\}}} = \frac{w_x{}^T C_{xy} w_y}{\sqrt{w_x{}^T C_{xx} w_x w_y{}^T C_{yy} w_y}}$$

Skipping some steps…

$$\begin{cases} C_{xy}\hat{w}_y = \rho\lambda_x C_{xx}\hat{w}_x \\ C_{yx}\hat{w}_x = \rho\lambda_y C_{yy}\hat{w}_y \end{cases} \qquad \lambda_x = \lambda_y{}^{-1} = \sqrt{\frac{w_y{}^T C_{yy} w_y}{w_x{}^T C_{xx} w_x}}$$

$$\begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \mu \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix}$$

kCCA

- This can be kernelized, (by replacement w=Kα) and the dual is:

$$\begin{pmatrix} 0 & K_1 K_2 \\ K_2 K_1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \rho \begin{pmatrix} K_1{}^2 & 0 \\ 0 & K_2{}^2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}$$

- At least 4 authors have done this independendly in the last year or so! (I used Bach & Jordan)

kCCA

- Very promising method, when used in conjunction with kernels
- Tomorrow we will see an application of this to cross-language analysis
- Work in progress in bioinformatics

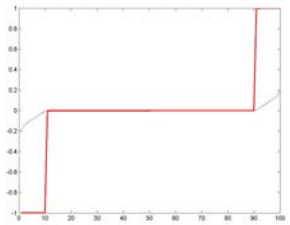
kCCA

- Important to understand its 'overfitting' behaviour (to avoid it).
- Usually $B \leftarrow B+\lambda I$
- This constrains the norm of vectors w, making the system less flexible …

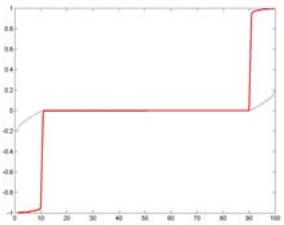
Some artificial examples…

- 50 random points in 10 dimensions
- Correlated with itself
- And with randomized version of self
- We expect only 10 positive eigenvalues
- If full freedom is given, they will be =1
- We can reduce their freedom …

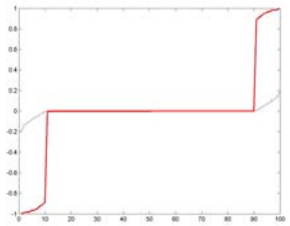
Regularization of kCCA…



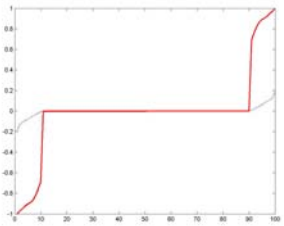
Regularization of kCCA…



Regularization of kCCA…

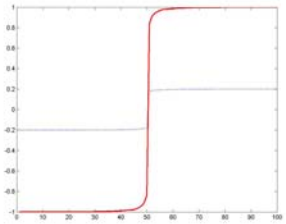


Regularization of kCCA…

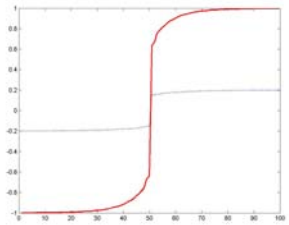


- 50 random points in 100 dimensions…

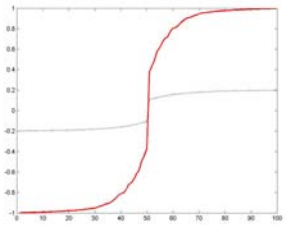
www.support-vector.net/nello.html



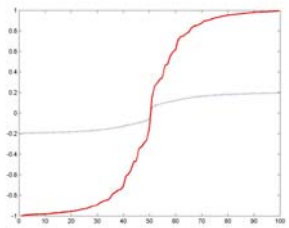
www.support-vector.net/nello.html



www.support-vector.net/nello.html

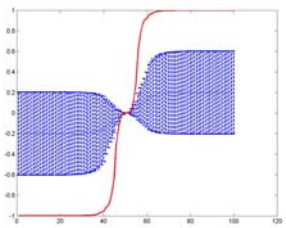


www.support-vector.net/nello.html

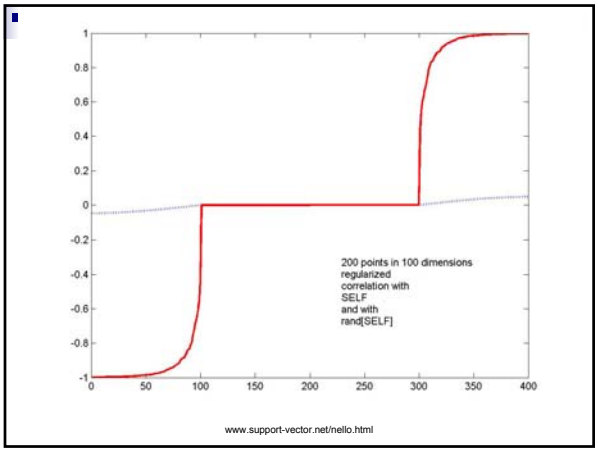


www.support-vector.net/nello.html

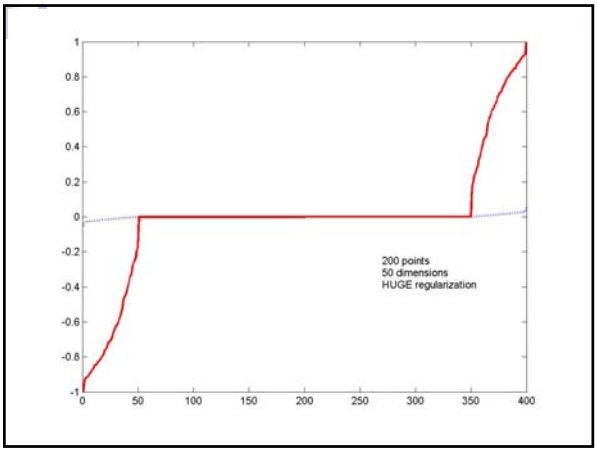
50 random points
in 50 dimensions…



www.support-vector.net/nello.html



www.support-vector.net/nello.html



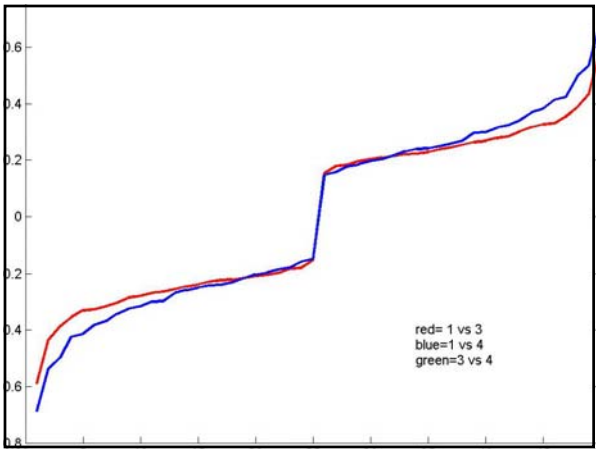
Correlation between strings…

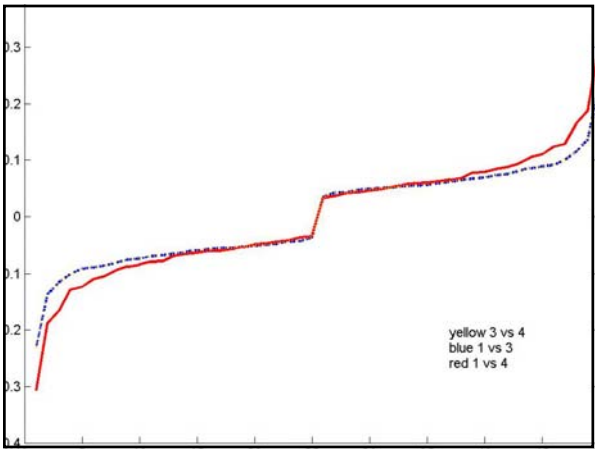
- Generated 3 sets of strings with markov process
- 2 of them are just transliterations of each other, the third is a different process

www.support-vector.net/nello.html

Can you find them ?

1	3	4
'VBNEUJVQRNEVICJWRVINE'	'LMMUCXWLFHOFVIOFNPHLG'	'WCOFVKWRSOFWJDKXSWJOF'
'SWMIPURKPZPZWZUJKDFJZ'	'DMMPBZVJKIGNFMZWBEYMB'	'TXNJQVSLQAQAXAVKLEGKA'
'FESPZIJEUJVDHEWDFDVIC'	'RTRAACUXSACLMMZWKGLGO'	'GFTQAVKFVKWEIFXEGEWJD'
'TJWBQSTLMXBHRRBOMGMGV'	'IBQSRHOKRTTSTFGVJKEHI'	'UKXCRTUMMLCISSCPNHNHW'
'HRPYEVBQBFNMCGNEMOIQW'	'VRHJFMZXSRTQFNPFHNPHJ'	'ISQZFWCRCGONDHOFNPJRX'
'AQXRZUCNEVFJVFJVBOICG'	'RLFHVJQFZZVJTFMBZZXBZ'	'BRYSAVDOFWGKWGKWCPJDH'
'HHHHRBQWNESNEWFGVBKGD'	'LWKCLFVRAWBEHVJKLFVNP'	'IIIISCRXQFTOFXGHWCLHE'
'NMYTJVDODHHHBHBZPZWNM'	'WBENPJQFMMZUXMUXTROOF'	'ONZUKWEPEIIICICAQAXON'
'RPUEVEWTCXDCKPBKPFJUY'	'PHOFHLMZUCLMZWCVJQFGN'	'SQVFWFXUDYEDLGCLQGKVZ'
'MDDCFEMYUJVQBOGNEWFDH'	'ACVJFZWKCLQIUOBACGENF'	'NEEDGFNZVKWRCPHOFXGEI'
'OIINTDODVQXCMKBQXOIBF'	'SRHOFJXWBEIGNPHLYYPJY'	'PJJOUEPEWRYDNLCRYPJCG'
'GLVFDVIIFDVFMCJVDFDDD'	'LMBEYYCTUKLQDMLQMBEYP'	'HMWGEWJJGEWGNDKWEGEEE'
'RBHBQSPZUESTPZWFMKBSP'	'NPHVIOFGEUYNPJWLFWSA'	'SCICRTQAVFTUQAXGNLCTQ'
'EWBKLJWRGLBZHYGAQBSNA'	'MZAROKZVJXMPHLBENPJXWC'	'FXCLMKXSHMCAIIHBRCTOB'
'XRBHOGUESNEXRQBOMOGVB'	'YVXVJWIUCMMBEHLMPHLMZ'	'YSCIPHVFTOFYSRCPNPHWC'
'OESPFBODDMDVOIIFMDKBB'	'KXSCYRHLFWCPHOFGVRTFW'	'PFTQGCPEPNEWPJJGNELCC'
'VDOGVIIBNEVWDHRNEUDAQ'	'JWKGVXWOIGEAGROFGENPJ'	'WEPHWJJCOFWXEISOFVEBR'
'QRRNESWZHBHESTLBODHEV'	'LMPFZVINPHLMZWBEHJXMB'	'RSSOFTXAICIFTUMCPEIFW'
'SPZIXRZUJKBZUEVIBHBOZ'	'JKZZUCVNPKAOBJQDBENJF'	'TQAJYSAVKLCAVFWJCICPA'
'JVBSTBHBSPYTBHRNESWFG'	'YYMBGDXBENPHOFGECYYMZ'	'KWCTUCICTOZUCISOFTXGH'
'HBFJKGTJUEXCXRBUJVLJV'	'ZVNYMMBADMPJWLMZUSAWD'	'ICGKUHJKVFYDYSCVKWMKW'
'BHOEIIBHRZESTBKGAIIIZ'	'KVJKGOFMBEYYVJQFTSVJY'	'CIPFJJCISAFTUCLHBJJJA'
'KBOEXCJUESWFEVBKDHHHB'	'DXMZWOFGEHNFZUGEYYRTT'	'LCPFYDKVFTXGFWCLEIIIC'
'NEXKURKBHRBHHHEWTJEVD'	'GDFHJFZVNYPFMJXVNPAGE'	'OFYLVSLCISCIIIFXUKFWE'
'DLLMCMQBHBKPYKLAAQQBH'	'ZWKIOFTVRRHOWCXOFNPJX'	'EMMNDNRCICLQZLMBBRRCI'





The idea…

- We can define sets, detect correlations, find clusters…
- On data such as strings, graphs, documents, …
- This is the main message of the talk…

On detecting stable patterns…

- We want relations that are not the effect of chance
  (i.e. that can be found in any random subset of the data, whp)
- Results from the theory of *empirical processes* can be used to guarantee this
- We do not discuss this here
  (but will give an example in tomorrow's talk)

Putting it all together …

- We can define sets
  (eg smallest enclosing sphere)
  on general types of data
  (eg strings)
- We can detect correlations between
  different datasets
  (tomorrow we will study correlation between english and french versions of same text)
- We can learn classifications, principal components, rankings …
- On strings, trees, images, text, …
- VERSATILITY, MODULARITY: the main features of KERNEL METHODS for pattern analysis

On kernel methods …

- Interpolate between syntactical and statistical pattern recognition
- Can naturally address many pattern analysis tasks that otherwise require specialized techniques …

Practical Applications of KMs

- Text Categorization:
  semantic kernels, etc…→ see tomorrow !
- Bioinformatics:
  gene function prediction; cancer type; diagnosis… → see papers/links on:
  www.support vector.net/bioinformatics.html

More…

- More advanced algorithms and kernels have been proposed, to deal with very general types of data, to insert domain knowledge, and to detect very general types of relations (eg: learning to rank phylogenetic trees; or detecting correlations in bi- lingual text corpora; etc. etc.)
- Now, however, we turn to another problem …
  … **on learning kernels !**

About Kernels…

- Let S be a set of points x_i
- Any function *K(x,z)* that creates a symmetric, positive definite matrix
  $K_{ij}= K(x_i, x_j)$
  is a valid kernel
  (= an inner product somewhere).
- **The kernel matrix** contains all the information produced by the kernel+data, and is passed on to the learning module
- Completely specifies *relative* positions of points in embedding space

K(1,1)	…		K(1,n)
…			
		K(i,j)	
K(n,1)			K(n,n)

Valid Kernels

- We can characterize kernel functions
- We can also give simple closure properties (kernel combination rules that preserve the kernel property)
- Simple example: *K=K1+K2* is a kernel if *K1* and *K2* are. Its features $\{\phi_i\}$ are the union of their features
- A simple convex class of kernels: $K = \sum_i \lambda_i K_i$ (more general classes are possible)
- Kernels form a cone

Last part of the talk…

- All information needed by kernel methods is in the kernel matrix
- Any kernel matrix corresponds to a specific configuration of the data in the feature space
- Usually a kernel function is used to obtain the matrix – but not necessary!
- We look at directly obtaining a kernel matrix (without kernel function)

The idea…

- Any symmetric positive definite matrix specifies an embedding of the data in some feature space
- Cost functions can be defined to assess the quality of a kernel matrix (wrt data)
  (alignment; margin; margin + spectral properties; etc).
- **Semi-Definite Programming** (SDP) deals with optimizing over the cone of positive (semi) definite matrices
- If cost function is convex, the problem is convex

What is SDP ?
(semi-definite programming)

Optimization of convex functions over the convex cone $\mathcal{P} = \{X \in \mathbb{R}^{p\times p} | X = X^T, X \succeq 0\}$, or subsets of this cone.

$$\min_x c^T x \quad \text{s.t.} \quad A(x) = A_0 + \sum_{i=1}^{n} x_i A_i \succeq 0 \quad (A_i = A_i^T \in \mathbb{R}^{p\times p})$$
$$Fx = g$$

- The Linear Matrix Inequality (LMI) $A(x) \succeq 0$ restricts $A(x)$ to be contained in the positive semi-definite cone $\mathcal{P}$
- Optimization variable is vector $x \in \mathbb{R}^p$
- Objective linear in x
- Finite number of LMI and equality constraints, linear in x

The Idea

- Perform kernel selection in "non-parametric" + convex way
- We can handle only the transductive case
- Interesting duality theory
- Problem: high freedom, high risk of overfitting
- Solutions: the usual …
  (bounds – see yesterday- and common sense)

Learning the Kernel (Matrix)

- We first need a *measure of fitness* of a kernel
- This depends on the task:
  we need a measure of agreement between a kernel and the labels
- *Margin* is one such measure
- We will demonstrate the use of SDP on case of hard margin SVMs. More general cases are possible (follow link below for paper).

Reminder:

QP for hard margin SVM classifiers

Given a linearly separable labelled sample $S_l = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the hyperplane $(\mathbf{w}, b)$ that solves the optimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{w},b} & \langle \mathbf{w}, \mathbf{w} \rangle \\ \text{subject to} & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1, i = 1, \ldots, n \end{array}$$

realizes the maximal margin classifier with a geometric margin $\gamma = 1/\|\mathbf{w}^*\|_2$ (hard margin).

Transform into corresponding dual problem:

$$\begin{aligned} w(K) &= 1/\gamma^2 \\ &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle \\ &= \max_{\alpha} 2\alpha^T e - \alpha^T G(K)\alpha \ : \ \alpha \geq 0, \ \alpha^T y = 0 \end{aligned}$$

$\alpha \in \mathbb{R}^n$, $G_{ij}(K) = [K]_{ij} y_i y_j = k(\mathbf{x}_i, \mathbf{x}_j) y_i y_j$.

A Bound Involving Margin and Trace

(ignore the details in this talk…)

The margin γ can be used to bound the generalization performance of SVMs (assumption of IID data): for a thresholded version of $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$, the proportion of errors on the test data is, with probability $1 - \delta$, bounded by

$$\frac{1}{n}\sum_{i=1}^{n} \phi(Y_i f(X_i)) + \frac{1}{\sqrt{n}}\left(4 + \sqrt{2\log(1/\delta)} + \sqrt{\frac{BC}{n\gamma^2}}\right)$$

where $C < n$ and $\text{trace}(K) \leq B$. This is valid when considering a kernel matrix of the form $K = \sum_{i=1}^m \mu_i K_i$ for a fixed set $\{K_1, \ldots, K_m\}$.

→ Inspires us to try to find the kernel matrix K for which the corresponding embedding shows maximal margin γ, keeping the trace of K constant

Optimal K: a convex problem

- Assume all labels are known, for simplicity
- Find the embedding (kernel matrix K) which shows maximal margin, keeping the trace of K constant:

$$\min_{K \geq 0} w(K) \quad \text{s.t. } \text{trace}(K) = c$$

or

$$\min_{K \geq 0} \max_{\alpha} 2\alpha^T e - \alpha^T G(K)\alpha \ : \ \alpha \geq 0, \ \alpha^T y = 0, \ \text{trace}(K) = c$$

- Note that $w(K)$ is convex in K (it is the pointwise maximum of affine functions of K). Given the convex constraint, the optimization problem is thus certainly convex in K. To express it as an SDP:

$$\min_{K \geq 0, t} t \ : \ t \geq \max_{\alpha} 2\alpha^T e - \alpha^T G(K)\alpha, \ \alpha \geq 0, \ \alpha^T y = 0, \ \text{trace}(K) = c$$

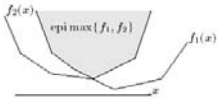
and express the constraints as a Linear Matrix Inequalty (LMI).

Just in case…

- linear and affine functions are convex and concave
- pointwise maximum:

f_1, f_2 convex $\Longrightarrow \max\{f_1(x), f_2(x)\}$ convex

(corresponds to intersection of epigraphs)



From Primal to Dual…

$$\min_{K \geq 0, t} t \ : \ t \geq \max_{\alpha} 2\alpha^T e - \alpha^T G(K)\alpha, \ \alpha \geq 0, \ \alpha^T y = 0, \ \text{trace}(K) = c$$

- Lagrangian of the maximization problem:

$$\mathcal{L}(\alpha, \nu, \lambda) = 2\alpha^T e - \alpha^T G(K)\alpha + 2\nu^T \alpha + 2\lambda y^T \alpha.$$

- By duality:

$$w(K) = \max_{\alpha} \min_{\nu \geq 0, \lambda} \mathcal{L}(\alpha, \nu, \lambda) = \min_{\nu \geq 0, \lambda} \max_{\alpha} \mathcal{L}(\alpha, \nu, \lambda)$$

- Since $G \succ 0$, at the optimum, we have

$$\alpha = G(K)^{-1}(e + \nu + \lambda y).$$

- Form the dual problem

$$w(K) = \min_{\nu \geq 0, \lambda} (e + \nu + \lambda y)^T G(K)^{-1}(e + \nu + \lambda y).$$

An SDP trick
Schur complement lemma

Consider the partitioned symmetric matrix

$$X = X^T = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

then

$$S = C - B^T A^{-1} B$$

is the Schur complement of A in X (provided $\det(A) \neq 0$)

Schur complement lemma:

if $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$.

The SDP Constraint

- Dual problem:

$$w(K) = \min_{\nu \geq 0,\, \lambda} (e + \nu + \lambda y)^T G(K)^{-1} (e + \nu + \lambda y).$$

- For any $t > 0$, we have $w(K) \leq t$ if and only if $\exists \nu \geq 0, \lambda \in \mathbb{R}$:

$$(e + \nu + \lambda y)^T G(K)^{-1} (e + \nu + \lambda y) \leq t,$$

- Using Schur complement lemma, we can write this as an LMI:

$$\begin{bmatrix} G(K) & e + \nu + \lambda y \\ (e + \nu + \lambda y)^T & t \end{bmatrix} \succeq 0$$

Maximizing Margin over K:
final (SDP) formulation

Find the embedding (kernel matrix K) which shows maximal margin, keeping the trace of K constant:

$$\min_{K \succeq 0} w(K) \quad \text{s.t. } \operatorname{trace}(K) = c$$

SDP !

$$\begin{array}{ll} \min\limits_{K, t, \lambda, \nu} & t \\ \text{subject to} & \operatorname{trace}(K) = c, \\ & K \succeq 0, \\ & \begin{pmatrix} G(K) & e + \nu + \lambda y, \\ (e + \nu + \lambda y)^T & t \end{pmatrix} \succeq 0, \\ & \nu \geq 0. \end{array}$$

Summarizing…

- Besides technicalities, the message is:
  given a set of labeled data,

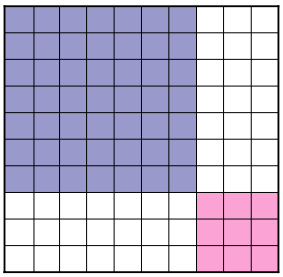
  we can find an embedding of the data in a space where the margin is maximized
  (with constant trace) and this is an SDP problem

- How to use this for machine learning ?

This does not learn …

- Of course this is not yet a learning algorithm
- Transductive setting: find a kernel matrix that has large margin w.r.t the available labels
- In general this would overfit.
- We can make it into a learning algorithm, by suitably restricting the hypothesis space (=feasible region)

Transduction



Test points are known beforehand

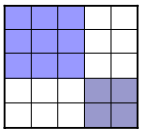
How to restrict the hypothesis space

- Important to 'entangle' training part with test part of the matrix
- One obvious choice (others should be possible):

$$K = \sum_i c_i K_i$$

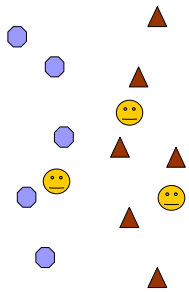
Learning the Kernel Matrix

- Transduction:
  given a partially labeled dataset, complete the labeling (train is labeled, test unlabeled).
- Use the labeled part to learn the geometry of the space.
  Warp the space,
  moving also the unlabeled points …
- All this by just adapting the kernel matrix.

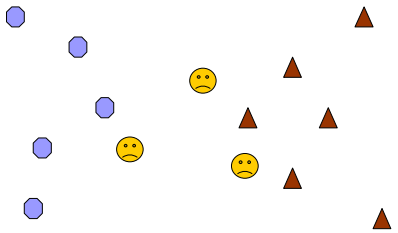


$$K = \sum_i \lambda_i K_i$$

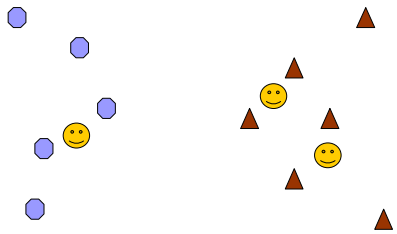
Learning the embedding



Learning the embedding (overfitting)



Learning the embedding



Some other examples of SDP problems

- Find matrix with optimal alignment
  (a measure of clustering)
- Find kernel matrix with maximal margin
- Find kernel matrix with maximal margin and minimal trace
  (done here)
- Minimize: Log det inv(M)
- Matrix completion:
  find 'best' completion of a partially filled kernel matrix:
  given a partially filled kernel matrix,
  1-find all legal completions,
  2-find a completion that has some extremal properties

Soft Margin Extension

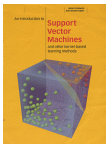
- Dealing with noise
- Standard solution in SVM literature: tolerate outliers by maximizing 'soft margin'
- Both 1-norm and 2-norm soft margin give rise to SDP problems
- (see paper for details)

CONCLUSIONS

- Kernel methods much more than maximal margin classification with gaussian kernels …
- Very rich, applicable, flexible family of pattern analysis methods
- Computationally efficient, statistically stable, and especially: versatile.

Links…

- Site for book, papers + THESE SLIDES: **www.support-vector.net**
- **www.kernel-machines.org** site: papers, links, events, news, …
- FInally: New book on "Kernel methods for pattern analysis" (CUP Press) – ***coming soon***!!



References (for SDP part)

- *Learning the Kernel Matrix with Semi-Definite Programming* (Lanckiert, Cristianini, Bartlett, El Ghaoui, Jordan) ICML – 2002
- Journal version (for JMLR) in preparation…
- (credit: some of the SDP slides prepared by Gert Lanckriet)