Kernel Methods for General Pattern Analysis

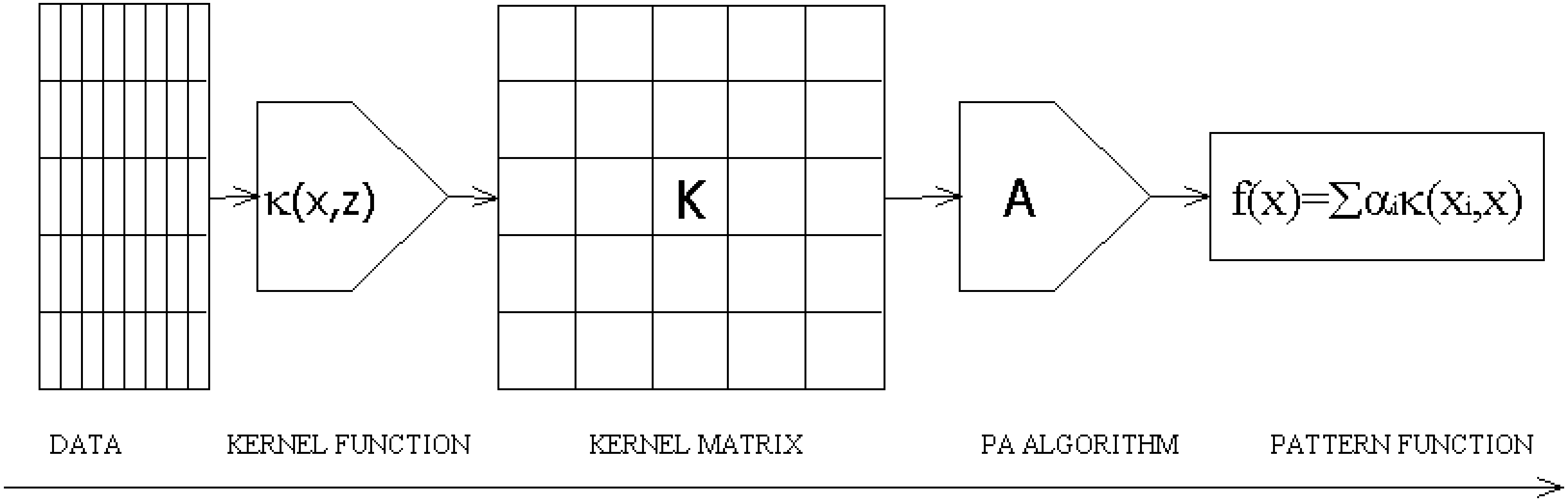
Nello Cristianini
University of California, Davis
nello@support-vector.net

Overview

- Kernel Methods are a new class of pattern analysis algorithms which can operate on very general types of data and can detect very general types of relations.
- Correlation, factor, cluster and discriminant analysis are just some of the types of pattern analysis tasks that can be performed on data as diverse as sequences, text, images, graphs and of course vectors.
- The method provides also a natural way to merge and integrate different types of data.

Overview

- Kernel methods offer a modular framework.
- In a **first step**, a dataset is processed into a kernel matrix. Data can be of various types, and also heterogeneous types.
- In a **second step**, a variety of kernel algorithms can be used to analyze the data, using only the information contained in the kernel matrix



Overview

- Computationally most kernel-based learning algorithms reduce to optimizing convex cost functions or to computing generalized eigenvectors of large matrices.
- Kernel design is based on various methods. For discrete data (eg sequences) often use methods like dynamic programming, branch and bound, discrete continuous optimization…
- Combination is very efficient but still computationally challenging, for our ambitions …

Overview

- The flexible combination of appropriate kernel design and relevant kernel algorithms has given rise to a powerful and coherent class of methods, whose computational and statistical properties are well understood,
- Increasingly used in applications as diverse as biosequences and microarray data analysis, text mining, machine vision and handwriting recognition.

Overview

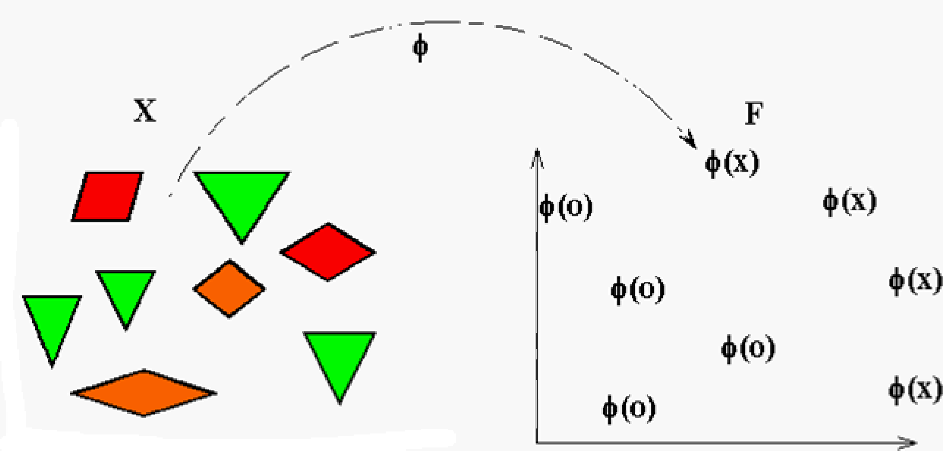
- General introduction to kernel methods
- Discussion of specific kernel-based algorithms
- Discussion of specific kernels for strings

- Few <u>examples</u> based on text and sequence data
- Discussion of how to combine heterogeneous data types (maybe)
- Focus on eigen- algorithms from classic multivariate stats, combined with kernels …

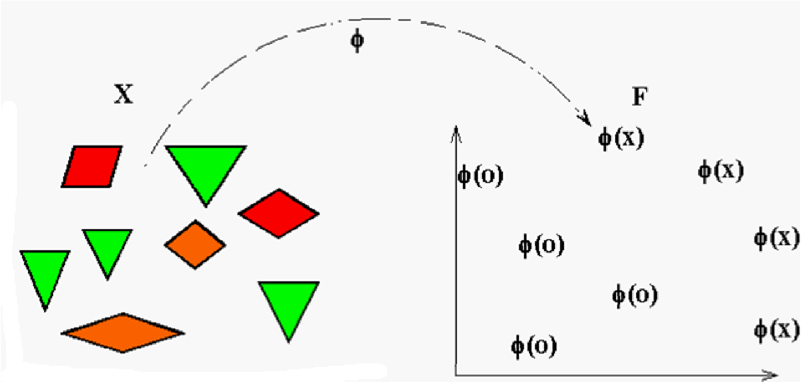
The Basic Idea: $x \rightarrow \phi(x)$

- 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

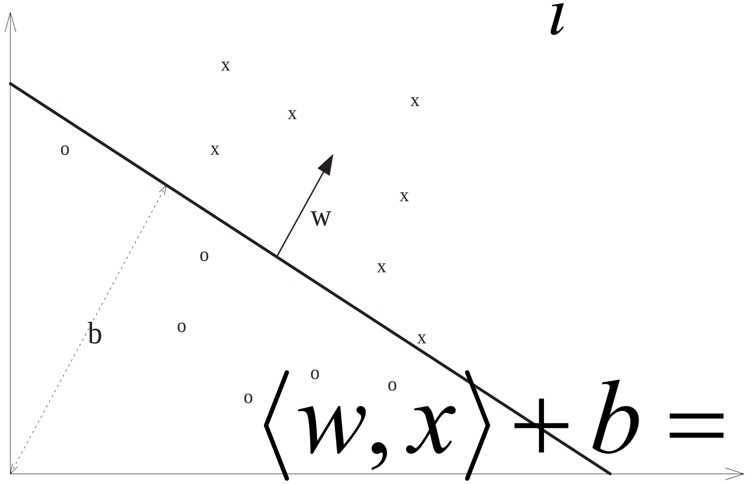
Algorithms

- Algorithms that can be used with inner product information:
  - Ridge Regression
  - Fisher Discriminant
  - Principal Components Analysis
  - Canonical Correlation Analysis
  - Spectral Clustering
  - Support Vector Machines
  - Lots more … …

Notation

- For now let us assume we have a vector space X where the data lives, and where we can define inner products … $\langle \bar{x}, \bar{z} \rangle = \sum_i x_i z_i$

- And we also have a <u>sample S of points</u> from that space…



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

Dual Representation

We consider linear functions like this: they could be the result of Fisher Discriminant Analyis, Ridge regression, PCA, etc …

$$f(x) = w' x + b$$

We consider re-writing them as a function of the data points (see perceptron as example)

$$w = \sum_i \alpha_i x_i$$

(we have a sample S of data points)

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

More generally …

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

See Wahba’s Representer theorem for theory behind …

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

Not crucial for this talk
for all algorithms we consider
this is always valid

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

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

Dual Representation

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

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

- This representation of the linear function only needs inner products between data points (not their coordinates!)

- JUST <u>REWRITTEN</u> THE SAME THING
  from #parameters w = dimension
  to #parameters α = sample size !

- If we want to work in the embedding space just need to know this:

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

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

Pardon my notation

Kernels

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

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

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

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

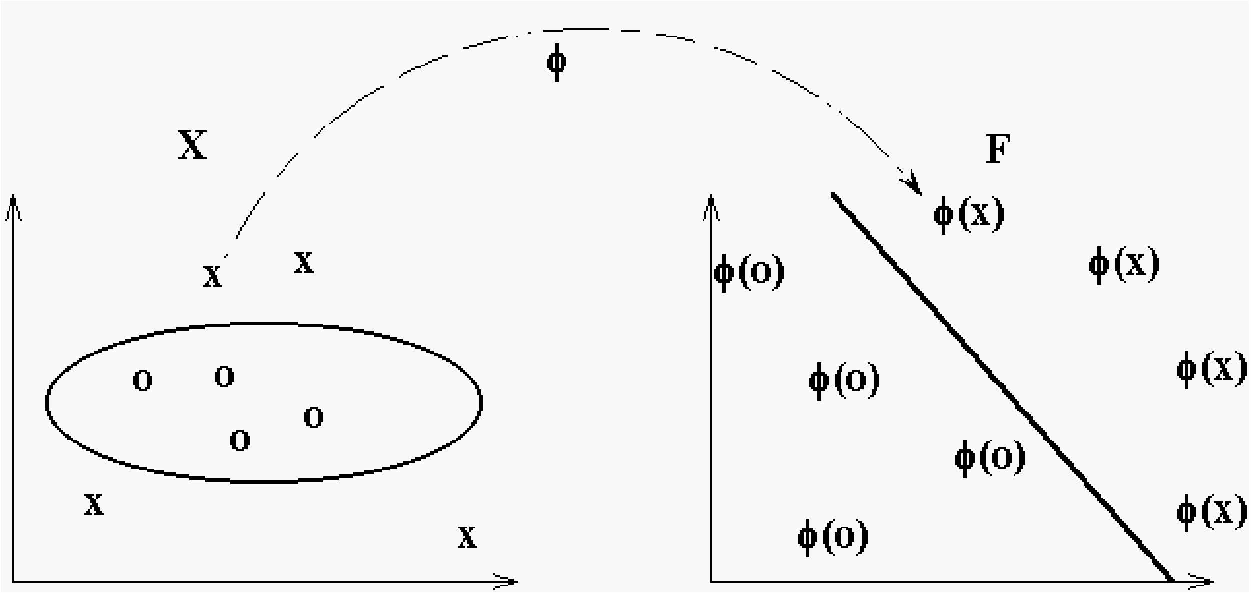
$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

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

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

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

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).

www.kernel-methods.net

More Important than Nonlinearity…

- Kernels can be defined on general types of data (as long as few conditions are satisfied – see later)
- Many classical algorithms can naturally work with general, non-vectorial, data-types !
- We can represent general types of relations on general types of data …

- Kernels exist to embed sequences, trees, graphs, general structures
- Semantic Kernels for text, Kernels based on probabilistic graphical models
- etc

The Kernel Matrix

- Given a sample of data, one can define the kernel matrix

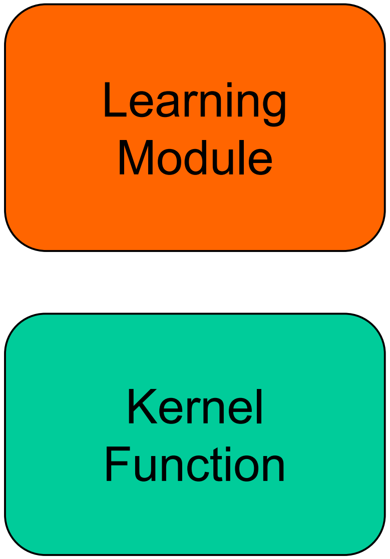
<x1,x1>	.	<x1,x2>	<x1,xn>
<x2,x1>	.	<x2,x2>	<x2,xn>
.	.	.	.
<xn,x1>	.	<xn,x2>	<xn,xn>

- Mercer Theorem: The kernel matrix is Symmetric Positive Definite

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

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**



Learning Module

Kernel Function

$$f(x)=\sum_i \underset{\uparrow}{\alpha_i} K(\underset{\uparrow}{x_i},x)$$

Algorithms

(to give you an idea)

Computation 2.5 Ridge regression 30
Computation 5.14 Regularised Fisher discriminant 131
Computation 5.15 Regularised kernel Fisher discriminant 133
Computation 6.3 Maximising variance 141
Computation 6.18 Maximising covariance 154
Computation 6.30 Canonical correlation analysis 163
Computation 6.32 Kernel CCA 165
Computation 6.34 Regularised CCA 169
Computation 6.35 Kernel regularised CCA 169
Computation 7.1 Smallest enclosing hypersphere 193
Computation 7.7 Soft minimal hypersphere 199
Computation 7.10 nu-soft minimal hypersphere 202
Computation 7.19 Hard margin SVM 209
Computation 7.28 1-norm soft margin SVM 216
Computation 7.36 2-norm soft margin SVM 223
Computation 7.40 Ridge regression optimisation 229
Computation 7.43 Quadratic e-insensitive SVR 231
Computation 7.46 Linear e-insensitive SVR 233
Computation 7.50 nu-SVR 235
Computation 8.8 Soft ranking 254
Computation 8.17 Cluster quality 261
Computation 8.19 Cluster optimisation strategy 265
Computation 8.25 Multiclass clustering 272
Computation 8.27 Relaxed multiclass clustering 273
Computation 8.30 Visualisation quality 277

- Algorithm 5.1 Normalisation 110
Algorithm 5.3 Centering data 113
Algorithm 5.4 Simple novelty detection 116
Algorithm 5.6 Parzen based classifier 118
Algorithm 5.12 Cholesky decomposition or dual Gram–Schmidt 126
Algorithm 5.13 Standardising data 128
Algorithm 5.16 Kernel Fisher discriminant 134
Algorithm 6.6 Primal PCA 143
Algorithm 6.13 Kernel PCA 148
Algorithm 6.16 Whitening 152
Algorithm 6.31 Primal CCA 164
Algorithm 6.36 Kernel CCA 171
Algorithm 6.39 Principal components regression 175
Algorithm 6.42 PLS feature extraction 179
Algorithm 6.45 Primal PLS 182
Algorithm 6.48 Kernel PLS 187
Algorithm 7.2 Samllest hypersphere enclosing data 194
Algorithm 7.8 Soft hypersphere minimisation 201
Algorithm 7.11 nu-soft minimal hypersphere 204
Algorithm 7.21 Hard margin SVM 211
Algorithm 7.26 Alternative hard margin SVM 214
Algorithm 7.29 1-norm soft margin SVM 218
Algorithm 7.32 nu-SVM 221
Algorithm 7.37 2-norm soft margin SVM 225
Algorithm 7.41 Kernel ridge regression 229
Algorithm 7.45 2-norm SVR 232
Algorithm 7.47 1-norm SVR 234
Algorithm 7.51 nu-support vector regression 236
Algorithm 7.52 Kernel perceptron 237
Algorithm 7.59 Kernel adatron 242
Algorithm 7.61 On-line SVR 244
Algorithm 8.9 nu-ranking 254
Algorithm 8.14 On-line ranking 257
Algorithm 8.22 Kernel k-means 269
Algorithm 8.29 MDS for kernel-embedded data 276
Algorithm 8.33 Data visualisation 280

Kernels

(to give you an idea)

- Definition 9.1 Polynomial kernel 286
  Computation 9.6 All-subsets kernel 289
  Computation 9.8 Gaussian kernel 290
  Computation 9.12 ANOVA kernel 293
  Computation 9.18 Alternative recursion for ANOVA kernel 296
  Computation 9.24 General graph kernels 301
  Definition 9.33 Exponential difiusion kernel 307
  Definition 9.34 von Neumann difiusion kernel 307
  Computation 9.35 Evaluating difiusion kernels 308
  Computation 9.46 Evaluating randomised kernels 315
  Definition 9.37 Intersection kernel 309
  Definition 9.38 Union-complement kernel 310
  Remark 9.40 Agreement kernel 310
  Section 9.6 Kernels on real numbers 311
  Remark 9.42 Spline kernels 313
  Definition 9.43 Derived subsets kernel 313
  Definition 10.5 Vector space kernel 325
  Computation 10.8 Latent semantic kernels 332
  Definition 11.7 The p-spectrum kernel 342
  Computation 11.10 The p-spectrum recursion 343
  Remark 11.13 Blended spectrum kernel 344
  Computation 11.17 All-subsequences kernel 347
  Computation 11.24 Fixed length subsequences kernel 352

- Computation 11.33 Naive recursion for gap-weighted subsequences kernel 358
  Computation 11.36 Gap-weighted subsequences kernel 360
  Computation 11.45 Trie-based string kernels 367
  Algorithm 9.14 ANOVA kernel 294
  Algorithm 9.25 Simple graph kernels 302
  Algorithm 11.20 All–non-contiguous subsequences kernel 350
  Algorithm 11.25 Fixed length subsequences kernel 352
  Algorithm 11.38 Gap-weighted subsequences kernel 361
  Algorithm 11.40 Character weighting string kernel 364
  Algorithm 11.41 Soft matching string kernel 365
  Algorithm 11.42 Gap number weighting string kernel 366
  Algorithm 11.46 Trie-based p-spectrum kernel 368
  Algorithm 11.51 Trie-based mismatch kernel 371
  Algorithm 11.54 Trie-based restricted gap-weighted kernel 374
  Algorithm 11.62 Co-rooted subtree kernel 380
  Algorithm 11.65 All-subtree kernel 383
  Algorithm 12.8 Fixed length HMM kernel 401
  Algorithm 12.14 Pair HMM kernel 407
  Algorithm 12.17 Hidden tree model kernel 411
  Algorithm 12.34 Fixed length Markov model Fisher kernel 427

www.kernel-methods.net

Examples of Kernels

- Simple examples of kernels are:

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

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

Obtaining Information about the Embedding

- What do we know about the geometry of the data sample embedded in the feature space ? We have only the kernel matrix …

- Eg we know all pairwise distances

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

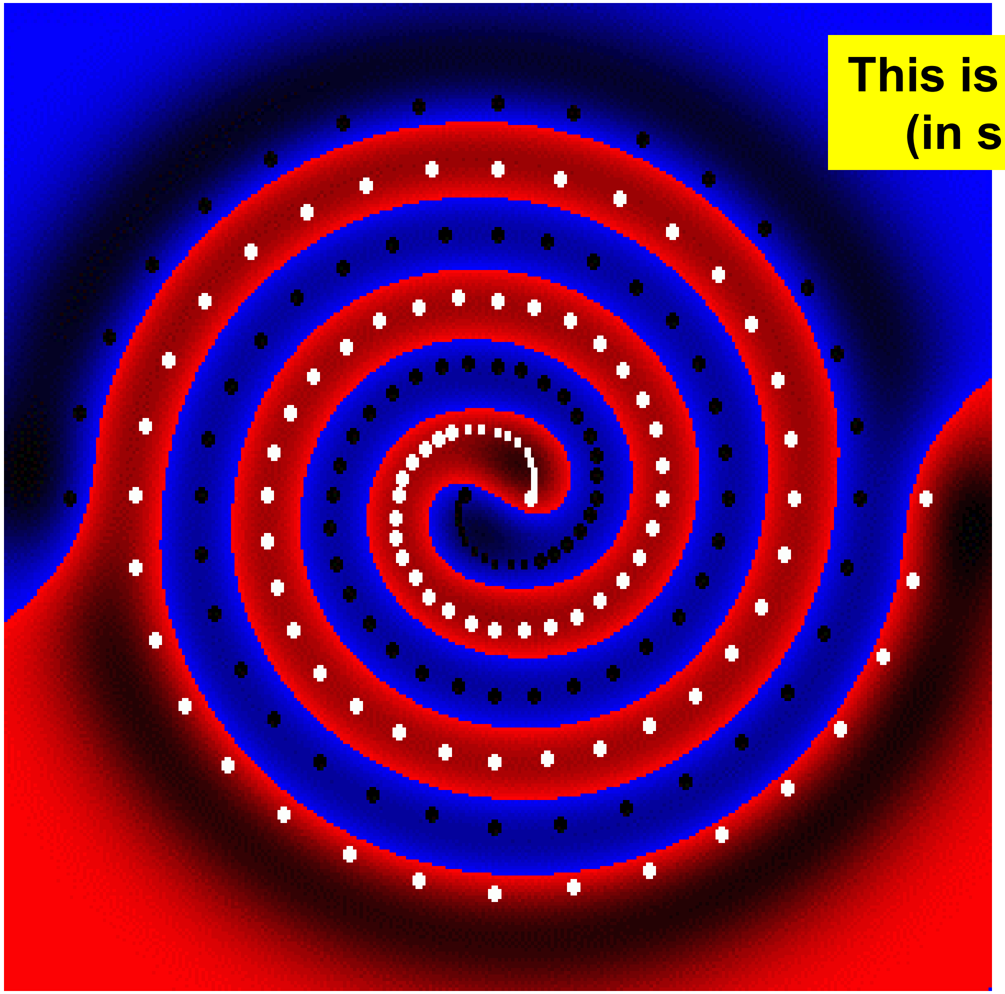
$$=K(x,x)+K(z,z)-2K(x,z)$$

- We can know distance of points from centre of mass of a set, and this can also lead to fisher-discriminant type of classifiers … etc etc etc

Etc etc etc …

- We will see that we can also know the principal axes,
- we can perform regression and discriminant analysis,
- as well as factor and cluster analysis…
- All in the kernel-induced space without need to use data coordinates explicitly…

Flexibility of KMs…

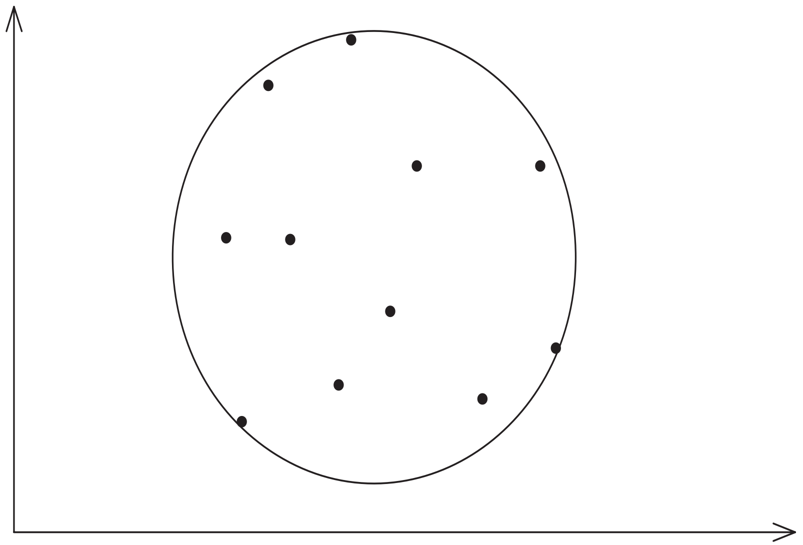


This is a hyperplane!
(in some space)

www.kernel-methods.net

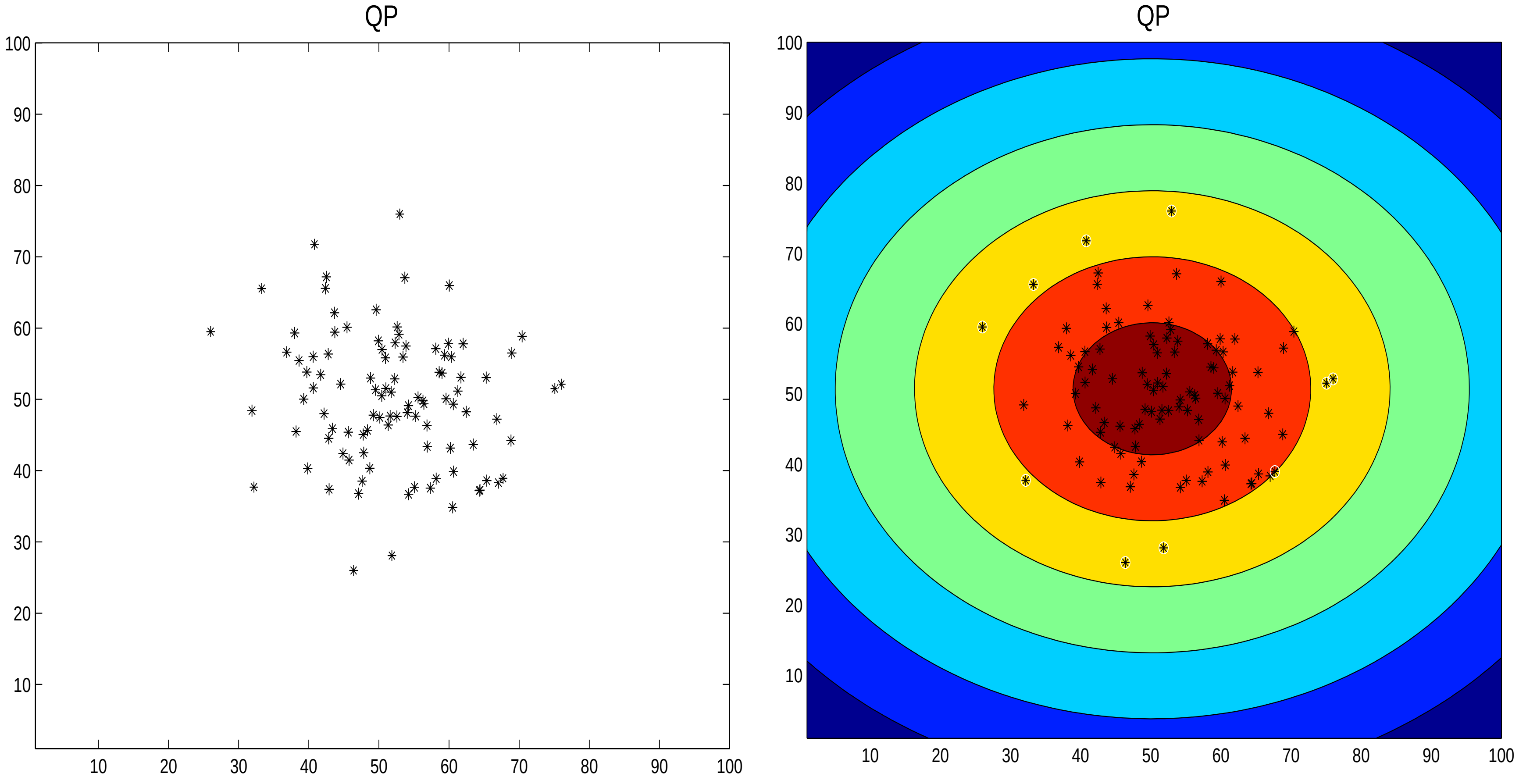
Smallest Sphere

- **Quick example of application: find smallest sphere containing all the data in the embedding space (a QP problem)**
- **Similarly: find small sphere that contains a given fraction of the points**

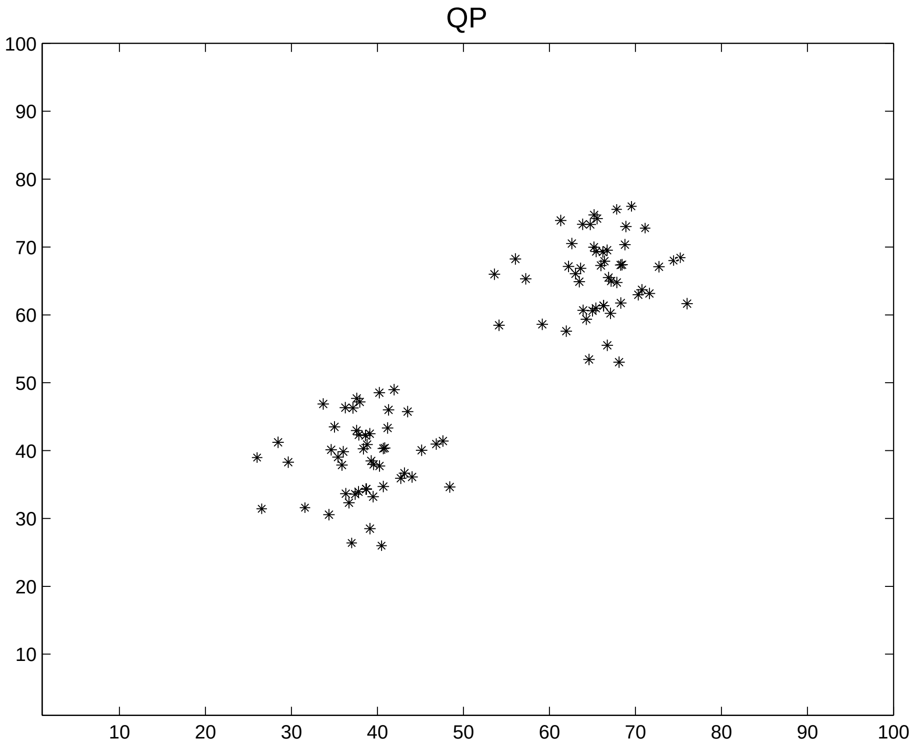


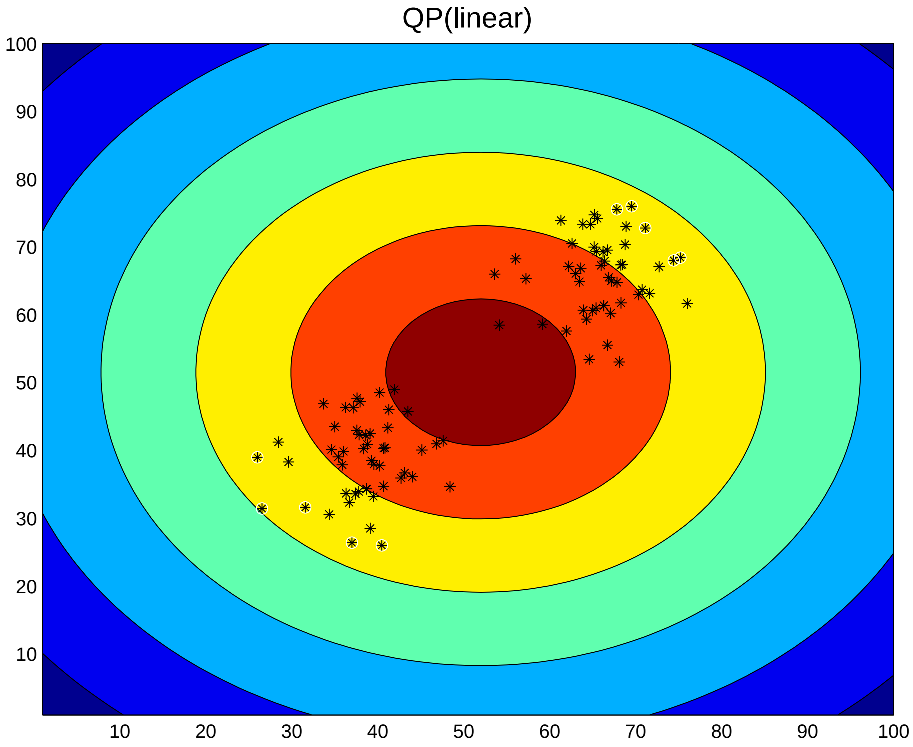
(Tax and Duin)

Example

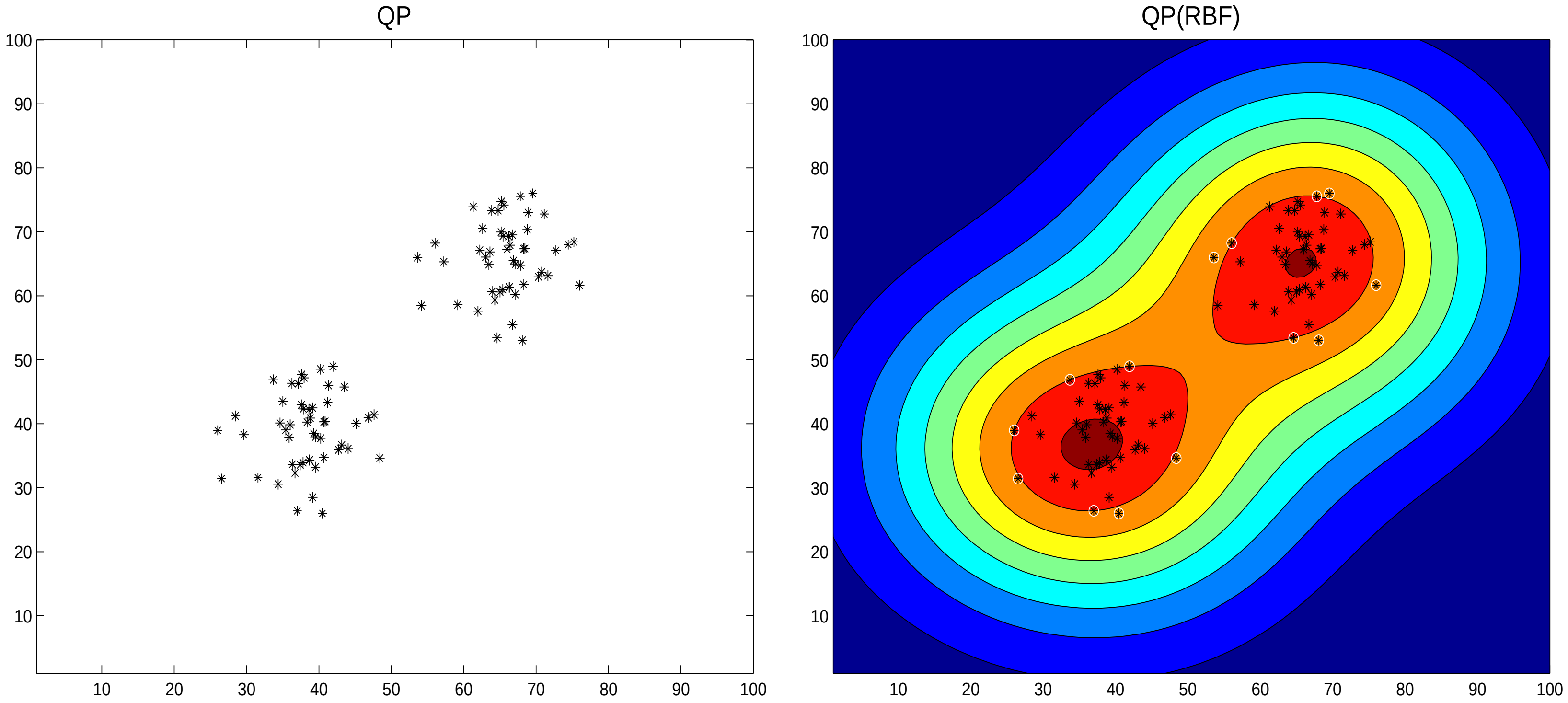

QP
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70
80
90
100
QP
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70
80
90
100

Example


QP
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70
80
90
100


QP(linear)
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70
80
90
100

Effect of Kernels



Generalized Eigenproblems

- Many multivariate statistics algorithms based on generalized eigenproblems can be ‘kernelized’ ..
  - PCA
  - CCA
  - FDA
  - Spectral Clustering
- (See website for free code…)

Generalized Eigenproblems

- $Ax=\lambda Bx$
- With A, B symmetric, B invertible

- Many classical multivariate statistics problems can be reduced to this

- But here matrices have same size as sample

Fisher Linear Discriminant

- In FDA the parameters are chosen so as to optimize a criterion functional that depends on the distance between the means of the two populations being separated, and on their variances.
- More precisely, consider projecting all the multivariate (training) data onto a generic direction $\boldsymbol{w}$, and then separately observing the mean and the variance of the projections of the two classes.

Maximal separation

- If we denote μ_+ and μ_- the means of the projections of the two classes on the direction ***w***, and s_+ and s_- their variances, the cost function associated to the direction ***w*** is then defined as

$$C(\mathbf{w}) = \frac{(\mu_+ - \mu_-)^2}{s_+{}^2 + s_-{}^2}$$

. FDA selects the direction ***w*** that maximizes this measure of separation.

Re-formulation

$$C(\mathbf{w}) = \frac{(\mu_+ - \mu_-)^2}{s_+{}^2 + s_-{}^2}$$

- This mathematical problem can be rewritten in the following way. Define the scatter matrix of the *i-th* class as (where by we denote the mean of a set of vectors) and define the total within-class scatter matrix as . In this way one can write the denominator of the criterion as and similarly, the numerator can be written as: where S_B is the between-class scatter matrix.

$$s_+{}^2 + s_-{}^2 = \mathbf{w}^T S_W \mathbf{w}$$

$$S_i = \sum_{x \in class(i)} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)^T$$

$$S_W = S_1 + S_{-1}$$

$$S_B = (\bar{\mathbf{x}}_\mathbf{1} - \bar{\mathbf{x}}_{-\mathbf{1}})(\bar{\mathbf{x}}_\mathbf{1} - \bar{\mathbf{x}}_{-\mathbf{1}})^\mathbf{T}$$

$$(\mu_+ - \mu_-)^2 = \mathbf{w}^\mathbf{T}(\bar{\mathbf{x}}_\mathbf{1} - \bar{\mathbf{x}}_{-\mathbf{1}})(\bar{\mathbf{x}}_\mathbf{1} - \bar{\mathbf{x}}_{-\mathbf{1}})^\mathbf{T}\mathbf{w} = \mathbf{w}^T S_B \mathbf{w}$$

⇒

$$C(w) = \frac{\mathbf{w}' S_B \mathbf{w}}{\mathbf{w}' S_W \mathbf{w}}$$

Fisher Discriminant

- The direction that maximizes the separation measure given above is the same that maximizes the cost function:

$$C(w) = \frac{\mathbf{w}' S_B \mathbf{w}}{\mathbf{w}' S_W \mathbf{w}}$$

- This ratio (Rayleigh quotient) is related to generalized eigenproblems…

Another eigenproblem

- In this way the criterion function becomes (an expression known as Reyleigh quotient) and its maximization amounts to solving the generalized eigenproblem .

$$C(w) = \frac{\mathbf{w}' S_B \mathbf{w}}{\mathbf{w}' S_W \mathbf{w}} \quad \Longrightarrow \quad S_B \mathbf{w} = \lambda S_w \mathbf{w}$$

PCA

Spectral Clustering

CCA

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) eigen-problem…
- **We will use this later** for visualization of the embedding space: projecting data onto a 2-dim plane

Kernel PCA

The eigenvectors can be written as:

$\boldsymbol{w}=\Sigma\alpha_i\boldsymbol{x}_i$

Problem: find the alphas

- Primal PCA:

$$\sum x_i = 0$$

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

$$\lambda v = Cv$$

- Dual PCA:

$$\sum \phi(x_i) = 0$$

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

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

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

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

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

Schoelkopf et al

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

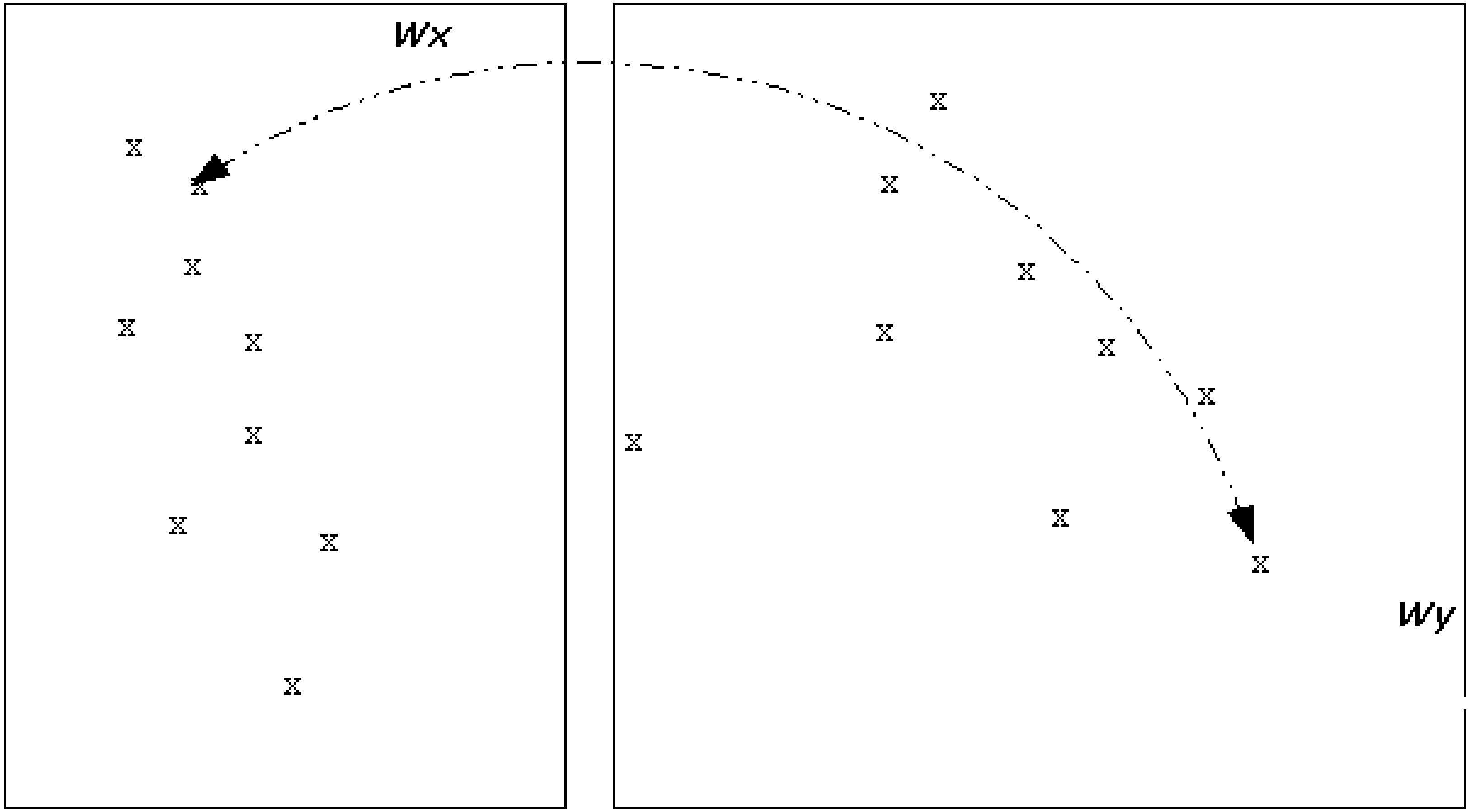
We will use it just to visualize a ‘slice’ of the embedding space, later…

Kernel CCA

- Similarly can be done for CCA (*)

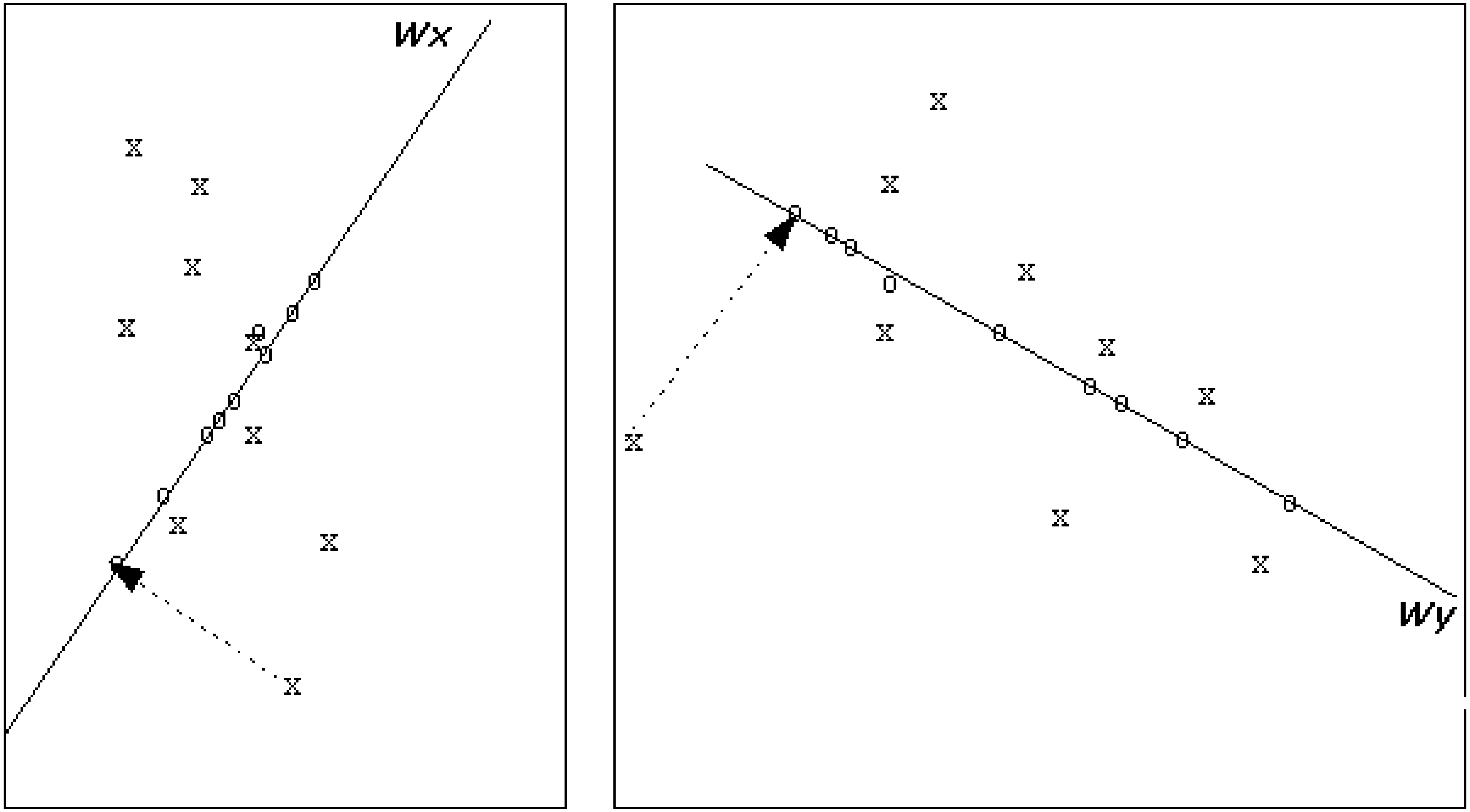
- (*) was done here in Berkeley !
  (Bach and Jordan)

CCA in pictures

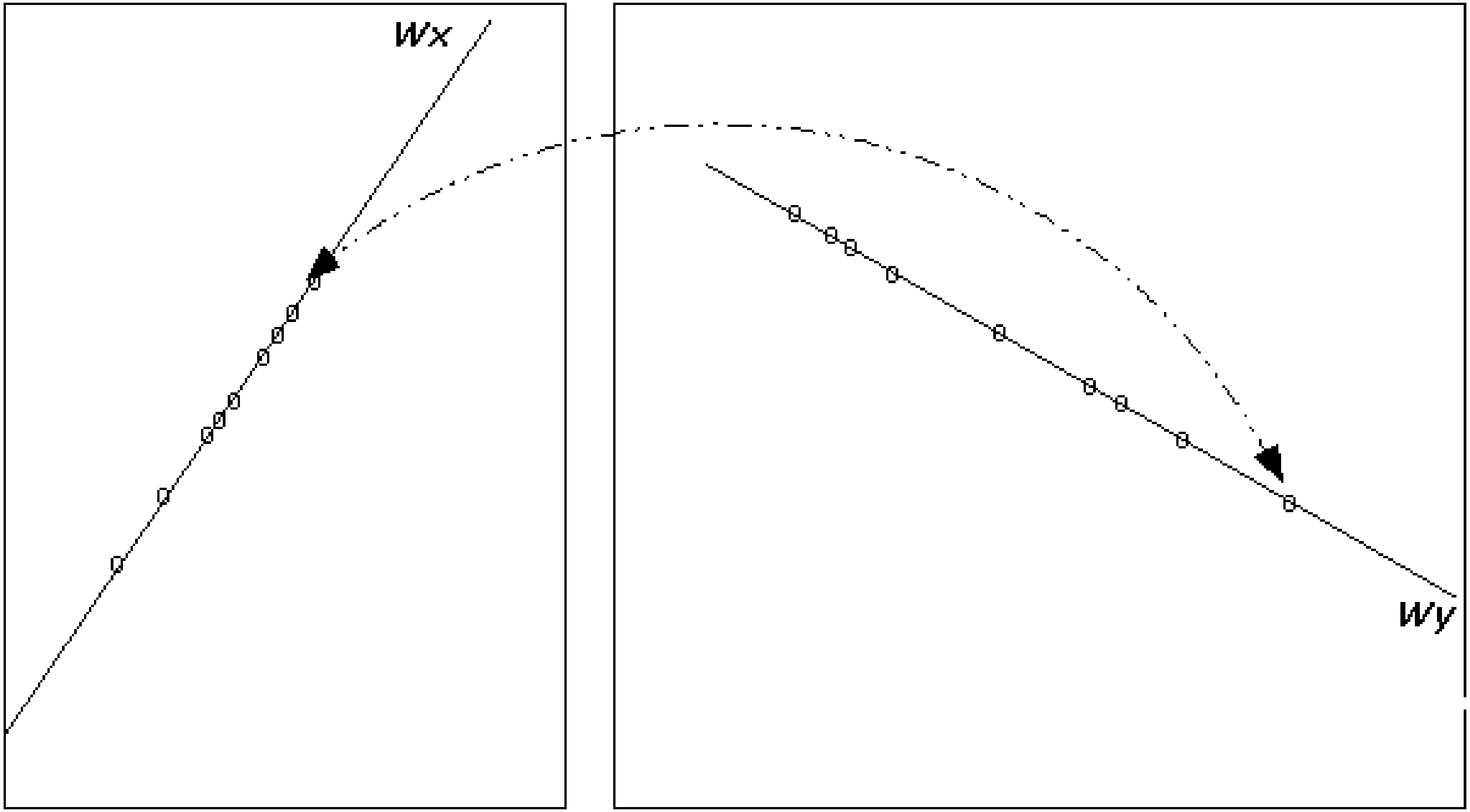


English corpus French corpus

CCA in pictures

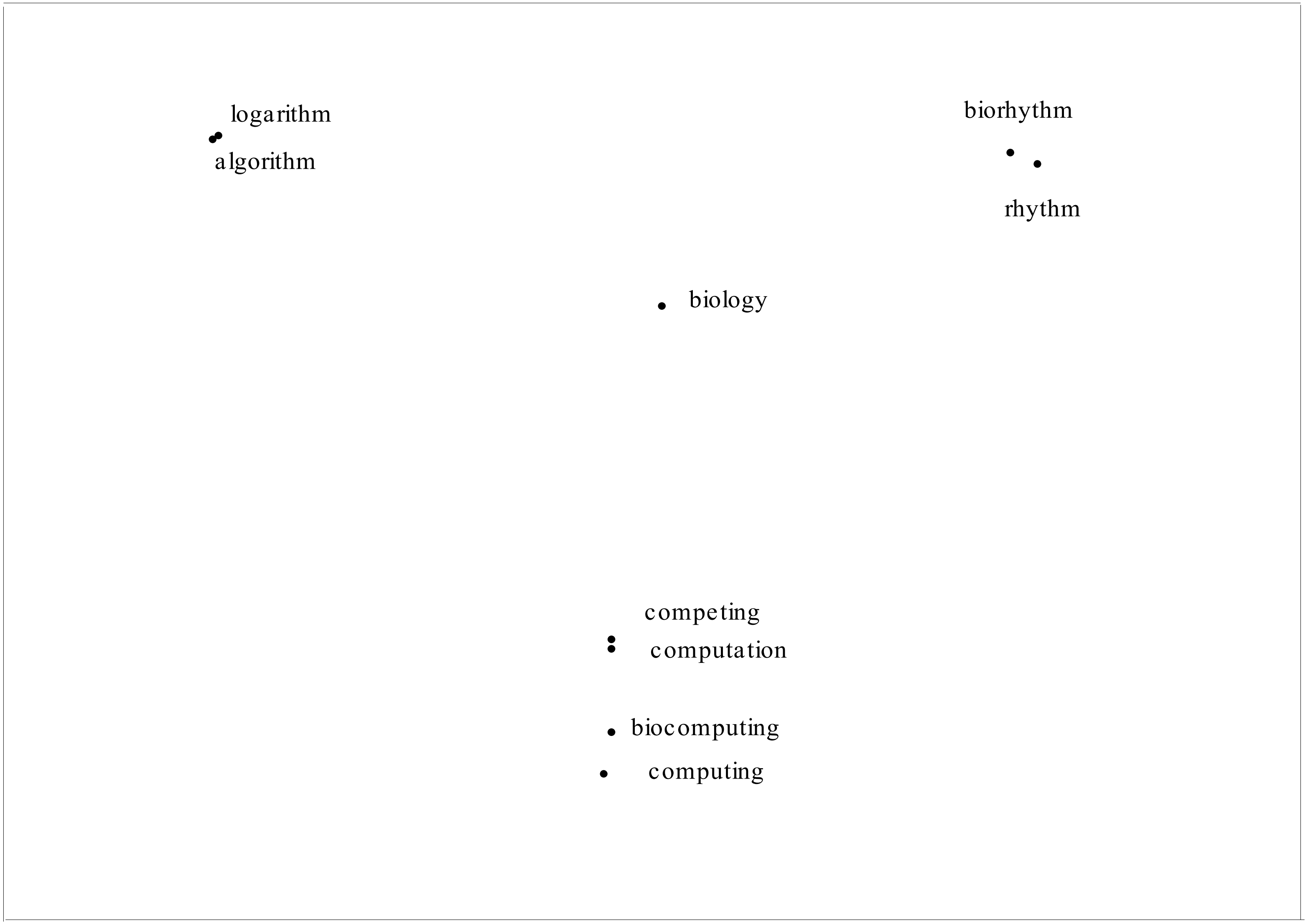


CCA in pictures



Kernels for Structured Data

- We will show how to use these algorithms on non-vector data
- Kernel functions do not need to be defined over vectors
- As long as a kernel function is guaranteed to always give rise to a symmetric positive definite matrix, it is fair game …
- As an example, we consider kernels over strings, to embed text documents in a space spanned by all k-mers from a given alphabet


logarithm
algorithm
biorhythm
rhythm
biology
competing
computation
biocomputing
computing

String Kernels

- A first generation of string kernels was based on dynamic programming ideas, and had a cost quadratic in the length of the sequences
- More recent implementations, based on structures similar to Tries and Suffix Trees, can achieve linear complexity in the total length of the corpus

Suffix Tree Kernels

For each document a feature vector is constructed indexed by all possible length-*k* strings (k-mer) of the given alphabet; the value of these entries is equal to the number of times this substring occurs in the given text.
→ how many ???

The kernel between two texts is then computed in the usual way, as the inner product of their corresponding feature vectors.

This can be done in a highly effective way by using a recursive relation and the appropriate data structures

(christina leslie)

kernel

- The kernels we use here are the 2-mer kernel, the 4-mer kernel (and gappy 4-mers and wildcard 4-mers, these count also partial matches, in different ways)

Features and Embedding

- The feature space is that of all k-grams…
- The embeddings:
  - Exact match
    (count how many times each k-gram appears)
  - Wildcard
    (count how many times, allowing some mismatches, parameters can be used to tune the cost of mismatches)
  - Gappy kernel
    (allows gaps, parameters tune cost of gaps)
- Cost of computation … (depends on tolerance, linear in length of sequence)

Putting it all together

- We can implicitly/virtually embed sequence data into a high dimensional space spanned by all k-grams, and look for linear relations in that space
- We can do factor analysis, correlation analysis, clustering, discrimination, regression and many other things … ON STRINGS !
- Let’s try with some text data … (fun example, not real application)

Suffix Trees

- ST are a *magic* data structure: constructed in linear time (*) can then be used as oracles to answer a series of queries in constant time (*):
  - Fill an entry of Kernel Matrix (for p-spectrum kernels and similar)
  - Position, frequency, … of words (also with mismatches at an additional cost)
  - …
- (*) in length of the corpus

Suffix Trees

- A suffix tree of a string s is a compacted trie of all suffixes of s
- Powerful data structure, many applications
- Typical way to answer queries: depth first traversals
- Construction: Ukkonen’s algorithm is linear (not simple, can give you a simple quadratic one)

Swiss Constitution

- The articles of the Swiss constitution, which is available in 4 languages: English, French, German and Italian.
- The articles are divided into several parts, called 'Titles' (in the English translation). All data can be found online at `http://www.admin.ch/ch/e/rs/c101.html`.

Swiss Constitution

- A few articles were omitted in this case study (some because they do not have an exact equivalent in the different languages, 2 others because they are considerably different in length than the bulk of the articles), leaving a total 195 articles per language (few hundreds symbols long)
- The texts are processed by removing punctuation and by stemming.
  (this software also online)

Examples

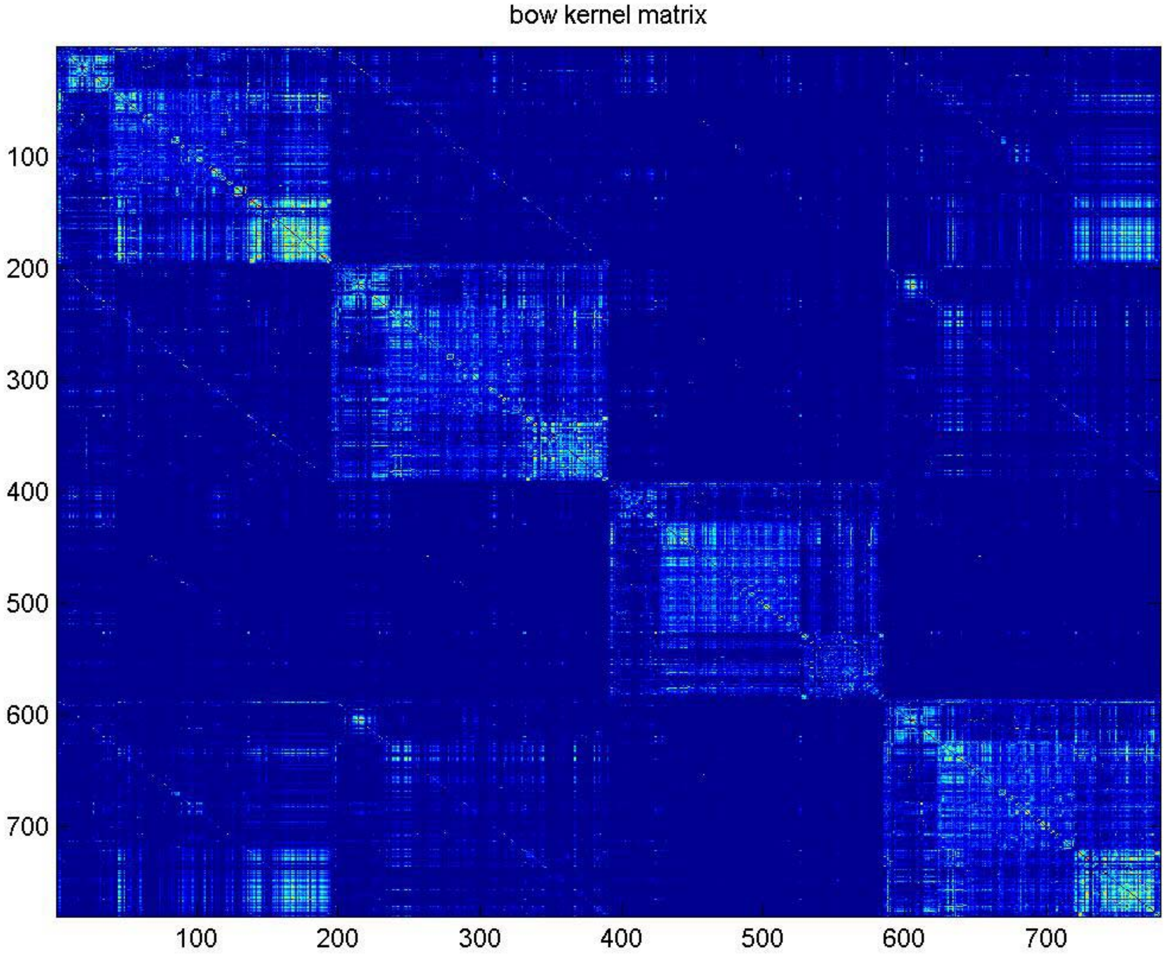
- We used various sequence kernels on those articles, as well as classic ‘bag of words’ kernels
- We experimented with different choices of parameters We applied various algorithms – no attempts to optimize performance – just demonstrations of feasibility

Examples

- First let’s see how kernel matrices look like (here the data set contains all 4 versions of the constitution, see just as a set of sequences)

- (all experiments by Tijl de Bie, KU Leuven)

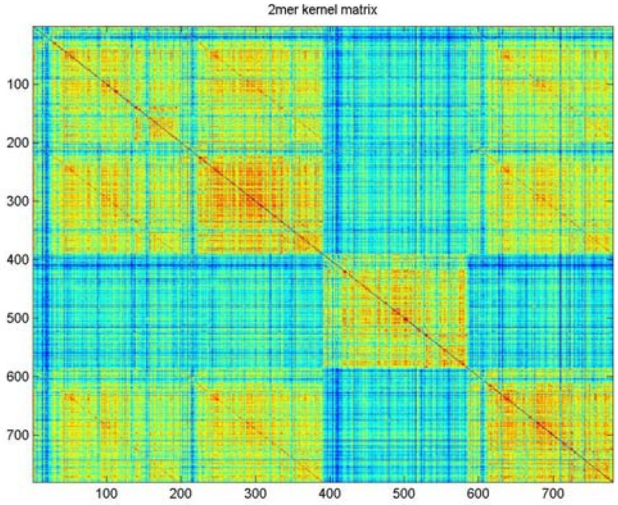
Kernel Matrix - Bag of Words



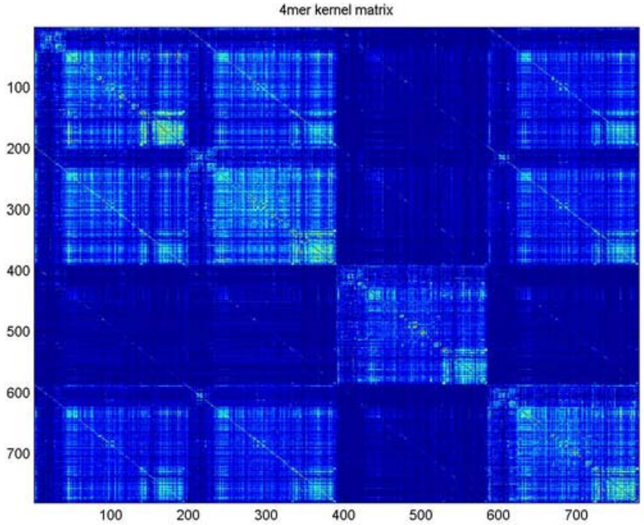
Languages:
English
French
German
Italian

Kernel Matrices

2-mer kernel



4-mer kernel



www.kernel-me

Examples

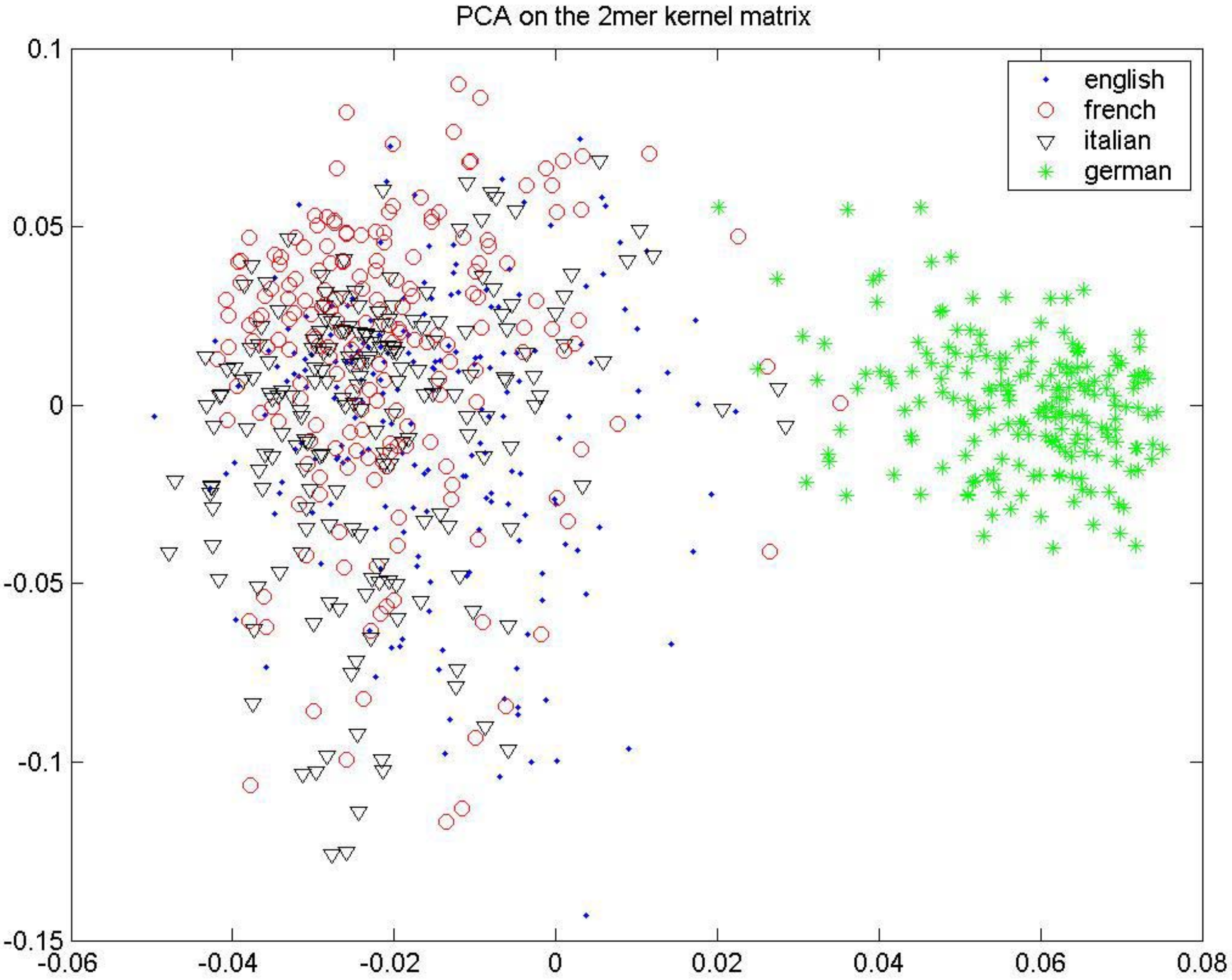
- Here we removed the diagonal in order to improve visibility
- Different colors are largely an artifact of matlab
- Notice that some structure appears in the kernel matrix, but not always distinguishable
- We used several different kernel functions…

Examples

- We played both with the possibility of distinguishing languages based on counting and comparing k-mers, and with the possibility of distinguishing topics (parts of the constitution known as titles)

- We tried first PCA and CCA …

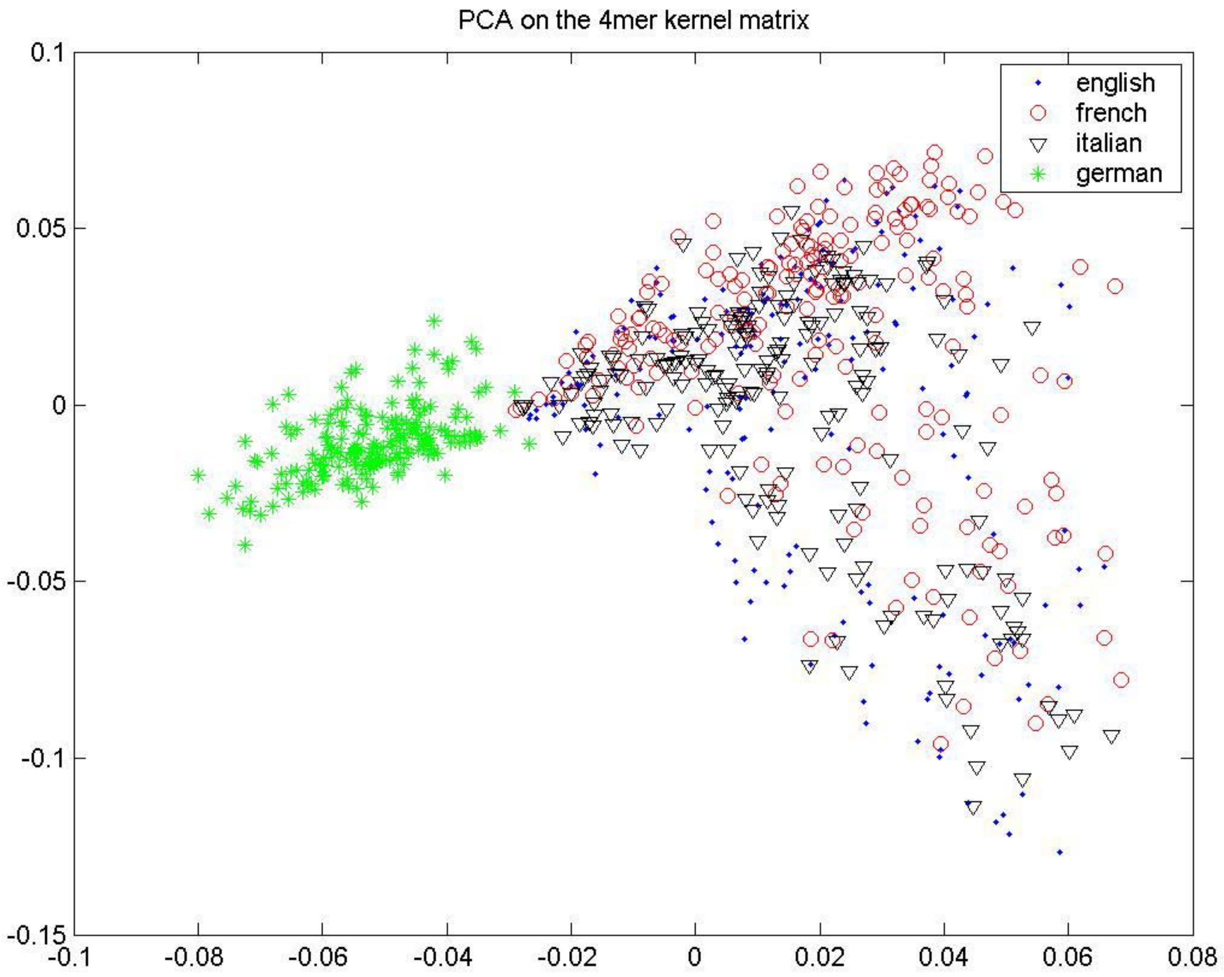
Principal Components Analysis



2-PCA on the total sample. 2mer kernel. Colored later according to language.

Only german stands out, for this simple kernel

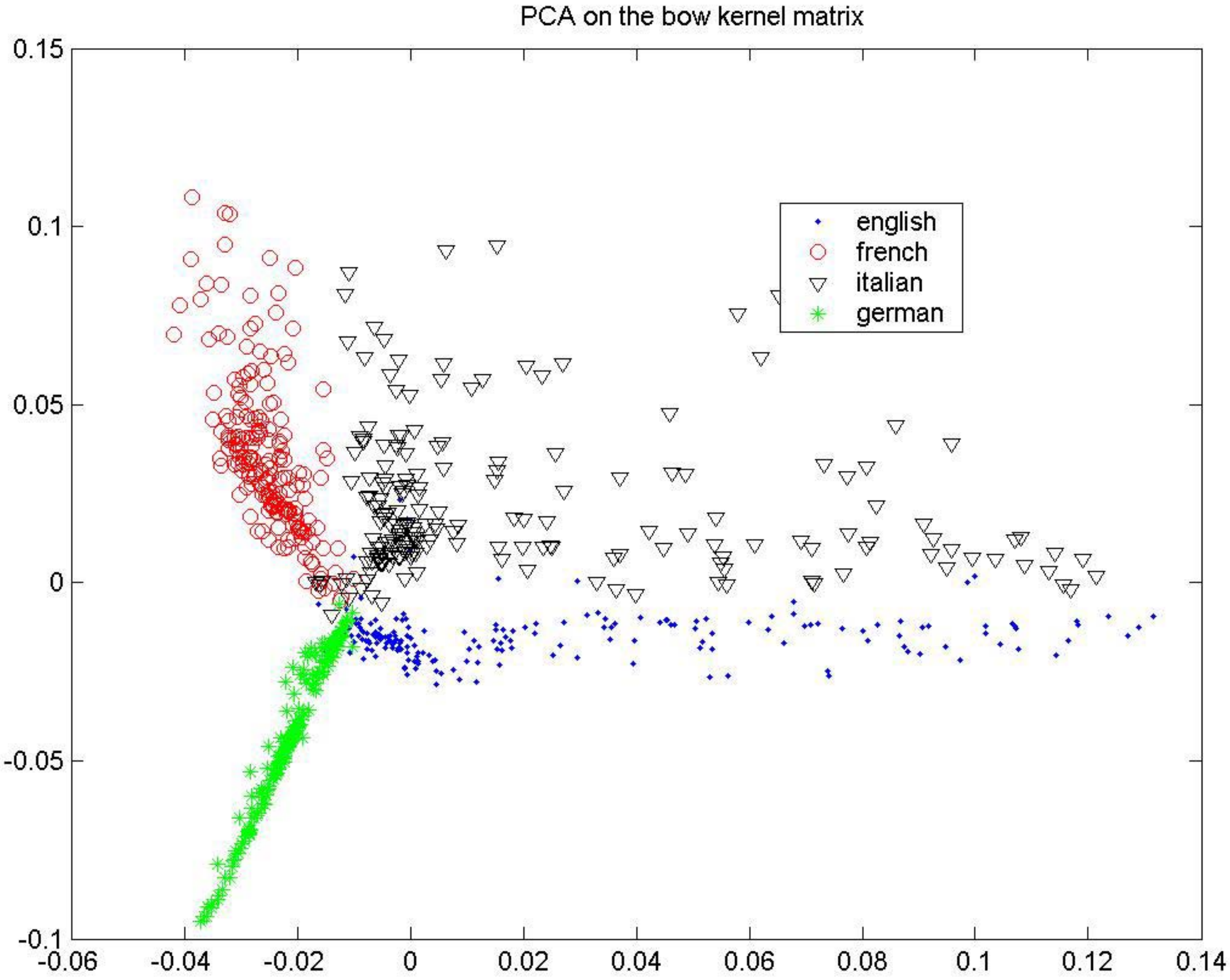
Principal Components Analysis



2-PCA on the total sample. 4mer kernel. Colored later according to language.

Better kernel,
Better embedding

Principal Components Analysis

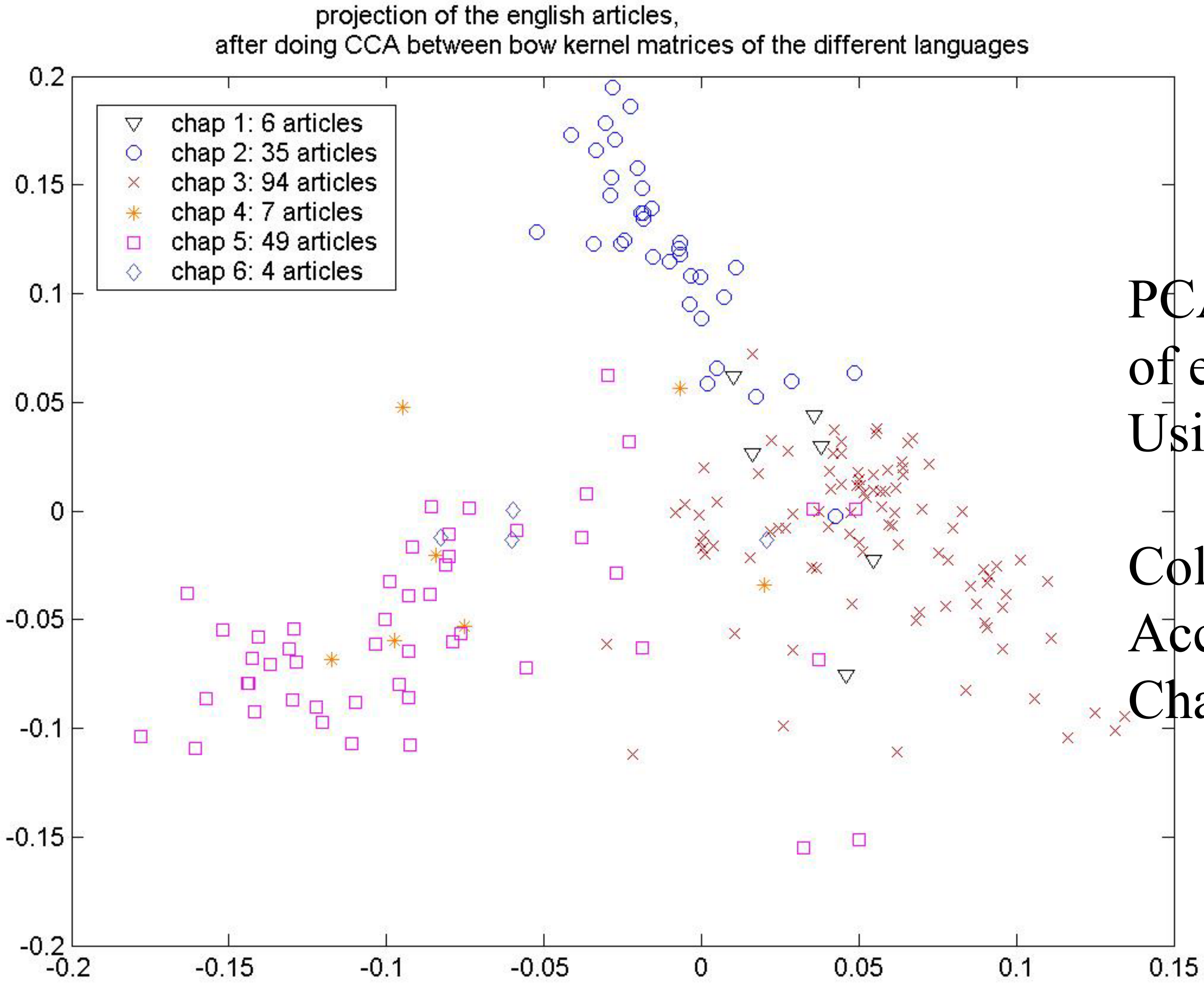


2-PCA on the total sample. Bag of Words kernel.

As expected, words are better features than 4-grams

Compare german with italian… ☺

Semantic Information - english: distinguishing Titles (chapters)



PCA projection
of english articles
Using BOW.

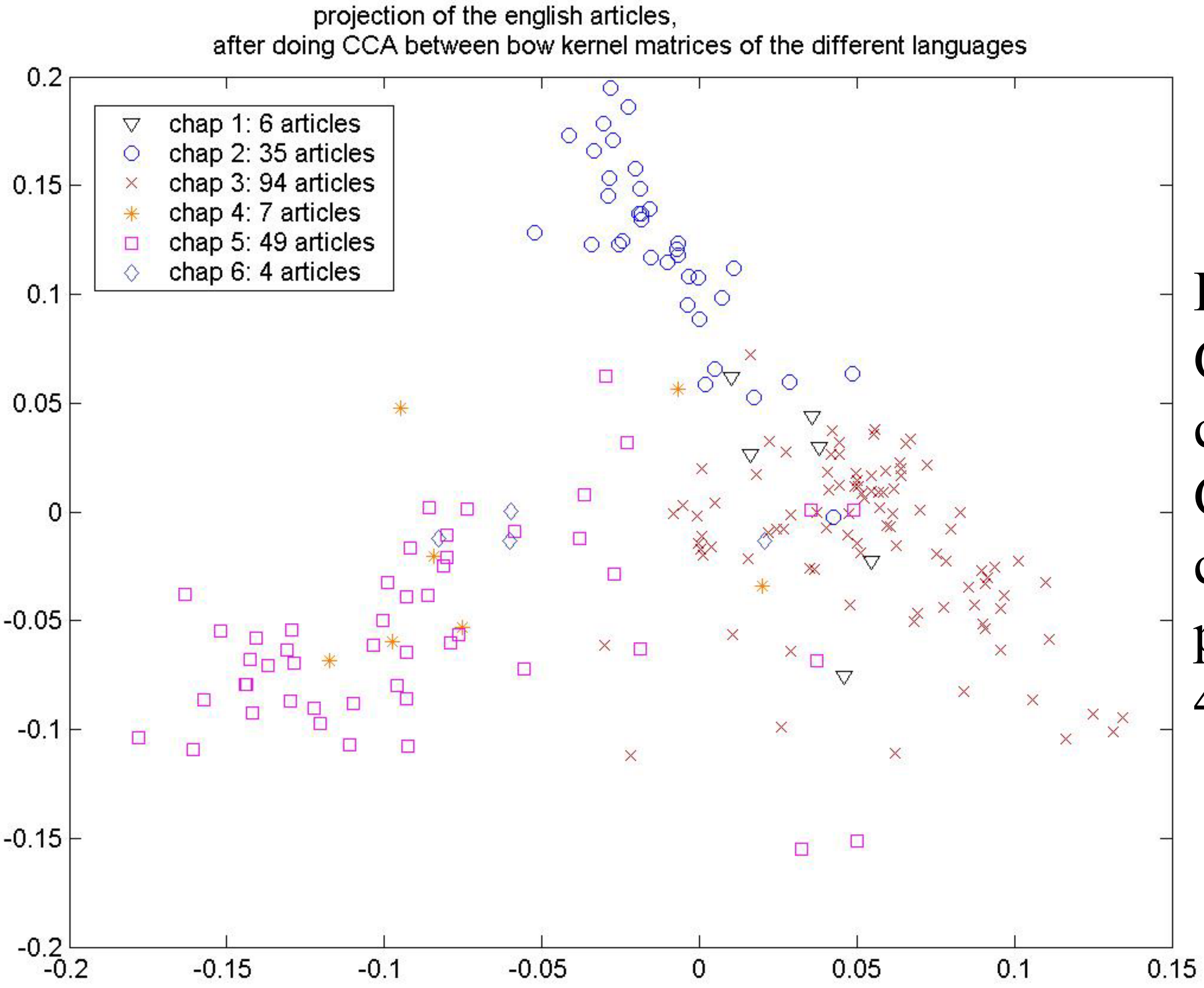
Colored later
According to
Chapter (topic)

Multi-way kCCA

- One can define Canonical Correlation Analysis among more than 2 spaces (*)
- Much more ‘supervised’ choice of projection
- We have chosen the regularization parameters to emphasize difference from randomized data
- Projected, then colored according to chapter …

- (*) Bach and Jordan

Projecting on first 2 english components of 4-way kCCA on BOW



Projection on 2 CCA components Colored by chapter CCA performed on all 4 languages

Other examples

- I have done similar work last year on canadian parliament data, significant semantic relations were extracted across french and english

- We then tried various methods of discriminant and cluster analysis with these same kernels and data … see all of them online…
  - (fisher vs svm)
  - (k-means vs spectral clustering)

Summary of classification results

	English vs French	English vs German	English vs Italian
SVM	3.3%±0.2	.12% ±.12	1.7% ±0.2
FDA	5.0% ±0.2	0 ±0	1.2% ±0.1

Noise free classification error rates, averaged over 100 randomizations with balanced 80/20 splits. The 2-mer kernel is used.

NOTE: when supervised learning is used on all dimensions, task is easy

Other kernels for structured data

- Kernels can be defined based on Probabilistic Graphical Models
- Marginalization kernels and Fisher kernels allow us to insert significant domain knowledge in the embedding
- …

Scaling things up

- We processed 800 sequences, of 300 symbols…
- I would like to do 80000000 sequences of 3000 symbols …. Can we ?

Statistical Aspects

- All these methods require very careful regularization
- Our models involve use of Rademacher Complexity and other uniform convergence arguments
- Have papers on reducing problems to eigenproblems
- Have papers on statistical stability and generalization power of eigenproblems

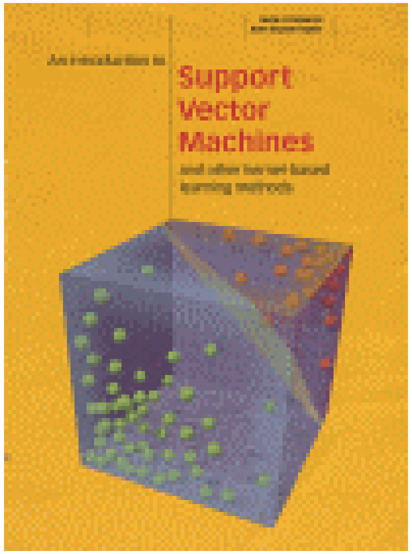
Combining Heterogeneous Data

- An important problem is to combine different data sources, or different representations of the same data
- (eg: in bioinformatics: combine sequence information with gene expression information and protein-interaction…)

Conclusions

- Kernel methods a fun set of methods for finding various types of patterns on structured data
- Allow us to apply well known set of statistical procedures to a whole new set of problems
- Potential to scale up to massive sample sizes, for the first time in pattern recognition…
- Can use Graphical Models as kernels easily…

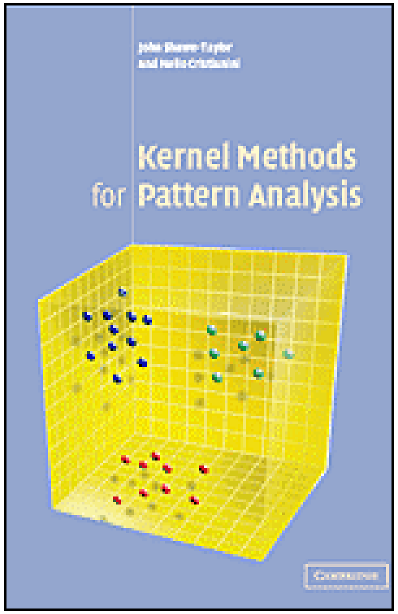
Conclusions



If you want to find out more, here websites and books …

- SUPPORT-VECTOR.NET
  (cristianini and shawe-taylor)

- Finally:
  New book on
  “Kernel methods for pattern analysis”
  CUP Press 2004 →
- KERNEL-METHODS.NET


John Shawe-Taylor and Nello Cristianini
Kernel Methods for Pattern Analysis
Cambridge