Kernel Methods for Pattern Analysis

John Shawe-Taylor
University College London

Acknowledgements

- Including work (and slides) from
  - Nello Cristianini
  - Blaž Fortuna
  - David Hardoon
  - Yaoyong Li
  - Huma Lodhi
  - Craig Saunders
  - Yoram Singer
  - Sandor Szedmak
  - Alexei Vinokourov

Aims of presentation

- Give intuition behind the approach – not the technical details
- Give broad picture to make clear when applicable and how to begin to use the methods
- Emphasise plug and play feature
- Details can always be found if needed

Motivation behind approach

- Linear learning typically has nice properties
  - Unique optimal solutions
  - Fast learning algorithms
  - Better statistical analysis
- But one big problem
  - Insufficient capacity

Historical perspective

- Minsky and Pappert highlighted the weakness in their book Perceptrons
- Neural networks overcame the problem by glueing together many linear units with non-linear activation functions
  - Solved problem of capacity and led to very impressive extension of applicability of learning
  - But ran into training problems of speed and multiple local minima

Kernel methods approach

- The kernel methods approach is to stick with linear functions but work in a high dimensional feature space:

$$\phi : x \in X \longmapsto \phi(x) \in \mathcal{F}$$

- The expectation is that the feature space has a much higher dimension than the input space.

Example

- Consider the mapping

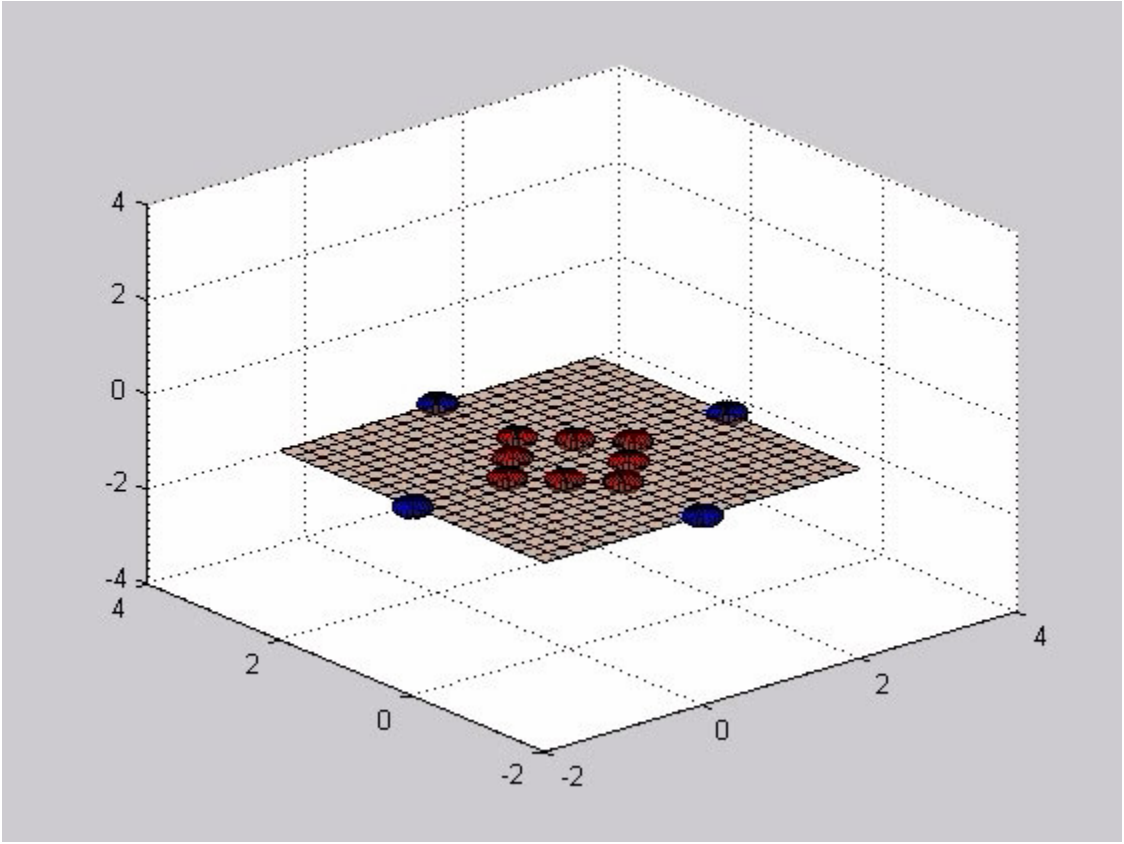
$$(x_1, x_2) \mapsto (x_1^2, x_1 x_2, x_2 x_1, x_2^2)$$

- If we consider a linear equation in this feature space:

$$ax_1^2 + bx_2^2 = c$$

- We actually have an ellipse – i.e. a non-linear shape in the input space.

Quadratic learning example

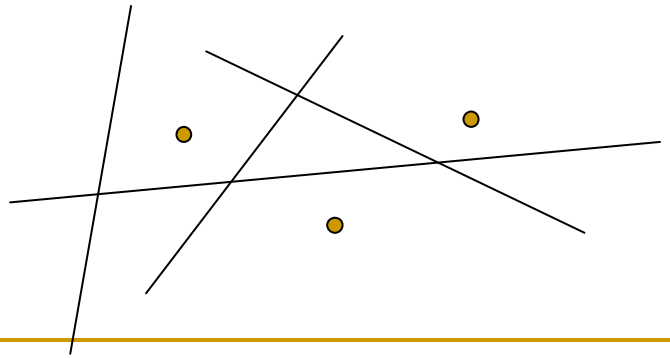


Capacity of feature spaces

- The capacity is proportional to the dimension – for example:

Theorem 1 *Given $m+1$ examples in general position in an m dimensional space, we can generate every possible classification with a thresholded linear function.*

- 2-dim:



Form of the functions

- So kernel methods use linear functions in a feature space:

$$x \longmapsto \langle \mathbf{w}, \phi(x) \rangle$$

- For regression this could be the function
- For classification require thresholding

$$x \longmapsto \mathrm{sign}\left(\langle \mathbf{w}, \phi(x) \rangle + b\right)$$

Problems of high dimensions

- Capacity may easily become too large and lead to overfitting: being able to realise every classifier means unlikely to generalise well
- Computational costs involved in dealing with large vectors

Capacity problem

- What do we mean by generalisation?

- Assume that we have a learning algorithm $\mathcal{A}$ that chooses a function $\mathcal{A}_{\mathcal{F}}(S)$ from a function space $\mathcal{F}$ in response to the training set S.

- From a statistical point of view the quantity of interest is the random variable:

$$\epsilon(S, \mathcal{A}, \mathcal{F}) = \mathbb{E}_{(\mathbf{x},y)}\left[\ell(\mathcal{A}_{\mathcal{F}}(S), \mathbf{x}, y)\right],$$

Generalisation of a learner

- For example, in the case of classification ℓ is 1 if the two disagree and 0 otherwise, while for regression it could be the square of the difference between $\mathcal{A}_{\mathcal{F}}(S)(\mathbf{x})$ and y.

- We refer to the random variable $\epsilon(S, \mathcal{A}, \mathcal{F})$ as the generalisation of the learner.

Example of Generalisation

- We consider the Breast Cancer dataset from the UCIrepository
- Use the simple Parzen window classifier: weight vector is

$$\mathbf{w}^+ - \mathbf{w}^-$$

where $\mathbf{w}^+$ ($\mathbf{w}^-$) is the average of the positive (negative) training examples.

- Threshold is set so hyperplane bisects the line joining these two points.

Example of Generalisation

- By repeatedly drawing random training sets S of size m we estimate the distribution of

$$\epsilon(S, \mathcal{A}, \mathcal{F}) = \mathbb{E}_{(\mathbf{x},y)}\left[\ell(\mathcal{A}_{\mathcal{F}}(S), \mathbf{x}, y)\right],$$

  by using the test set error as a proxy for the true generalisation
- We plot the histogram and the average of the distribution for various sizes of training set

  648, 342, 273, 205, 137, 68, 34, 27, 20, 14, 7.

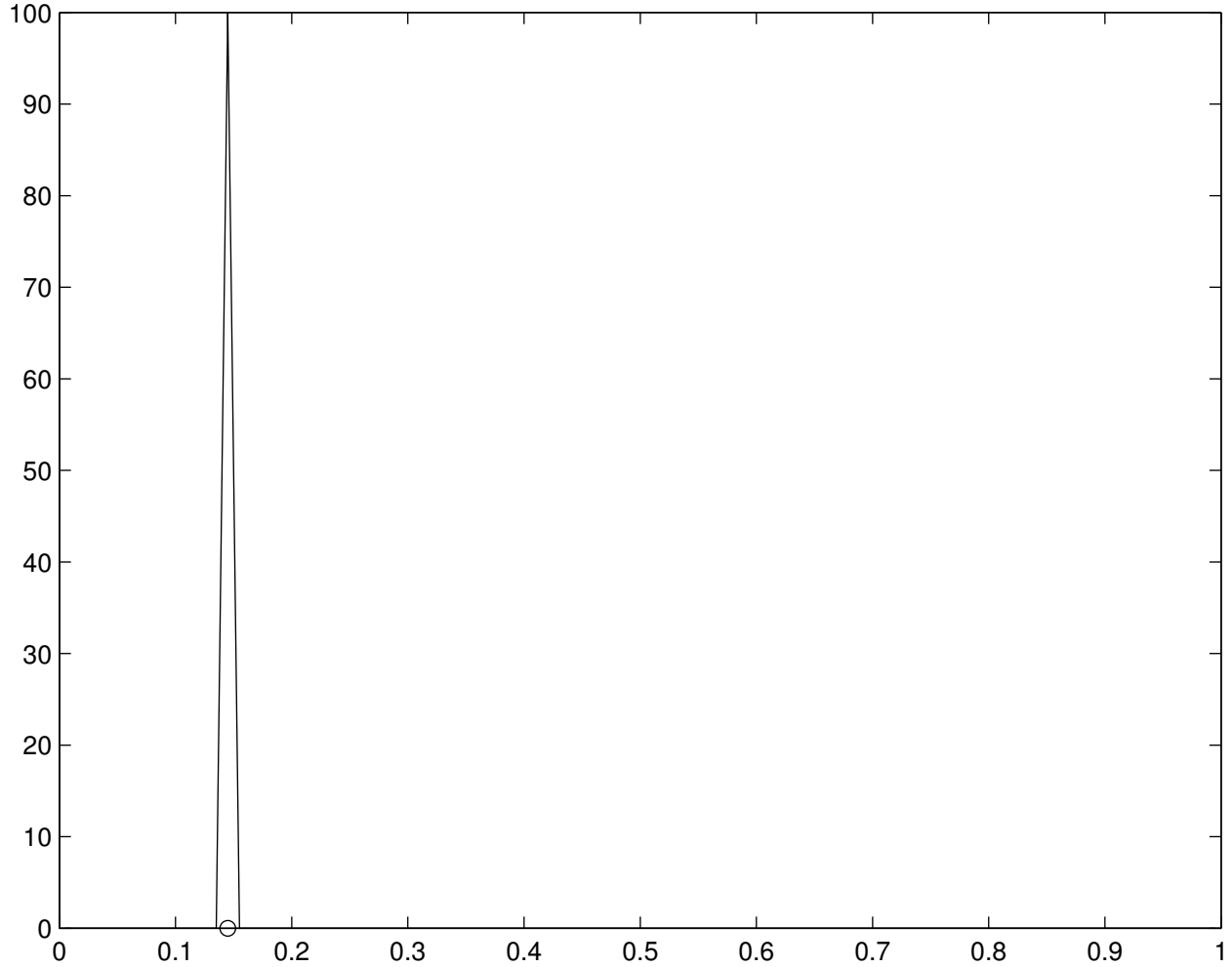
Example of Generalisation

- Since the expected classifier is in all cases the same

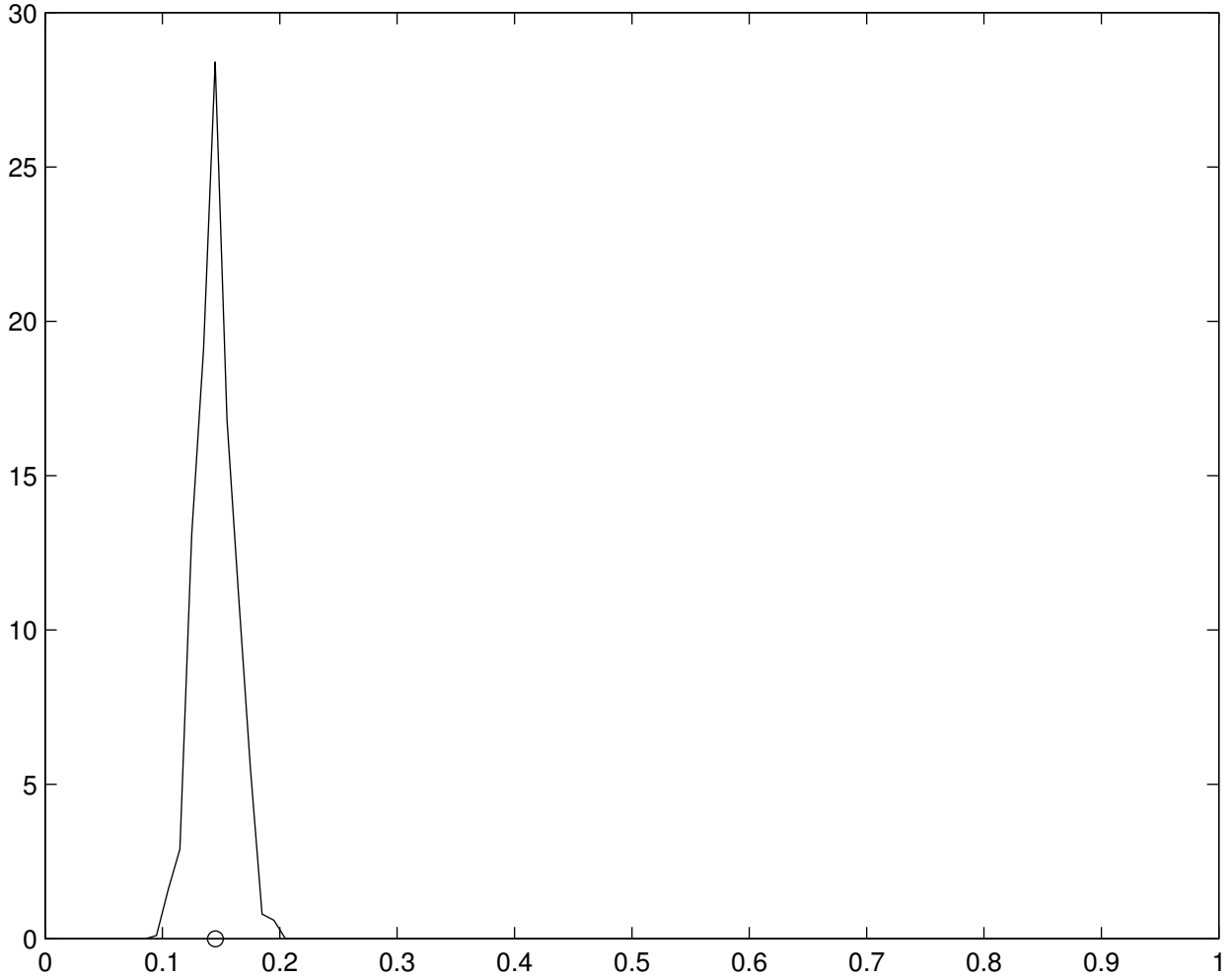
$$\begin{aligned}\mathbb{E}\left[\mathcal{A}_{\mathcal{F}}(S)\right] &= \mathbb{E}_S\left[\mathbf{w}_S^+ - \mathbf{w}_S^-\right] \\ &= \mathbb{E}_S\left[\mathbf{w}_S^+\right] - \mathbb{E}_S\left[\mathbf{w}_S^-\right] \\ &= \mathbb{E}_{y=+1}\left[\mathbf{x}\right] - \mathbb{E}_{y=-1}\left[\mathbf{x}\right],\end{aligned}$$

we do not expect large differences in the average of the distribution, though the non-linearity of the loss function means they won't be the same exactly.

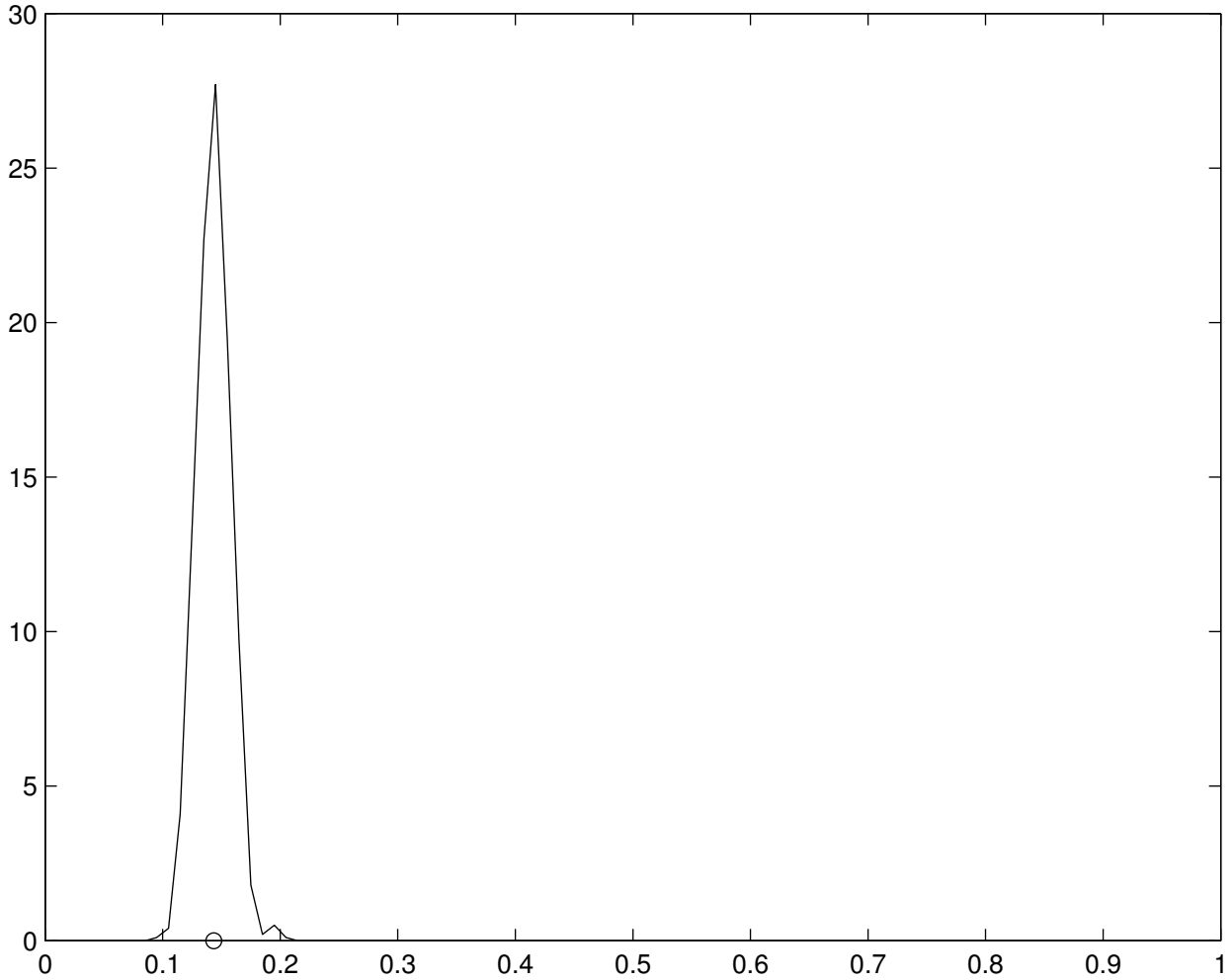
Error distribution: full dataset



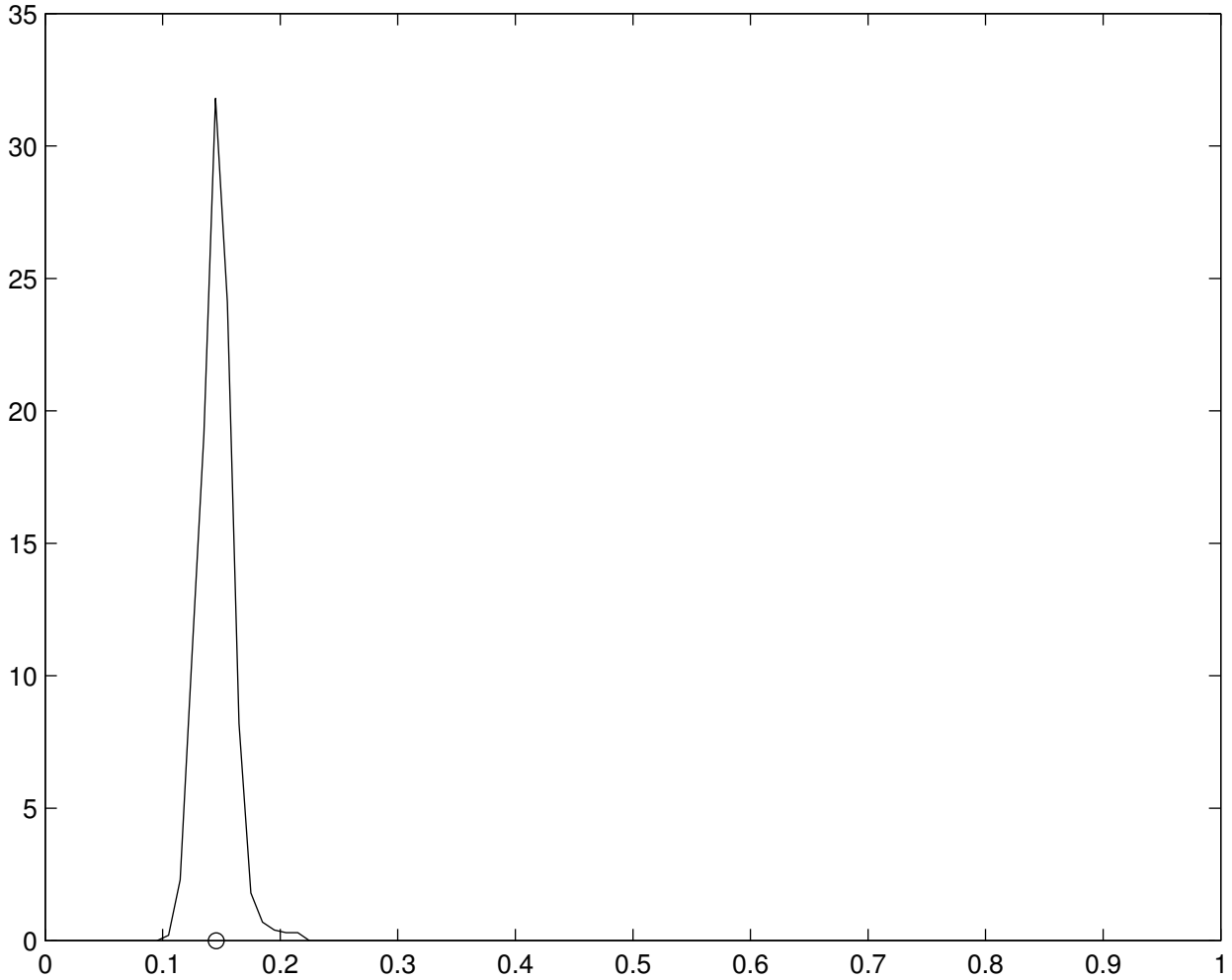
Error distribution: dataset size: 342



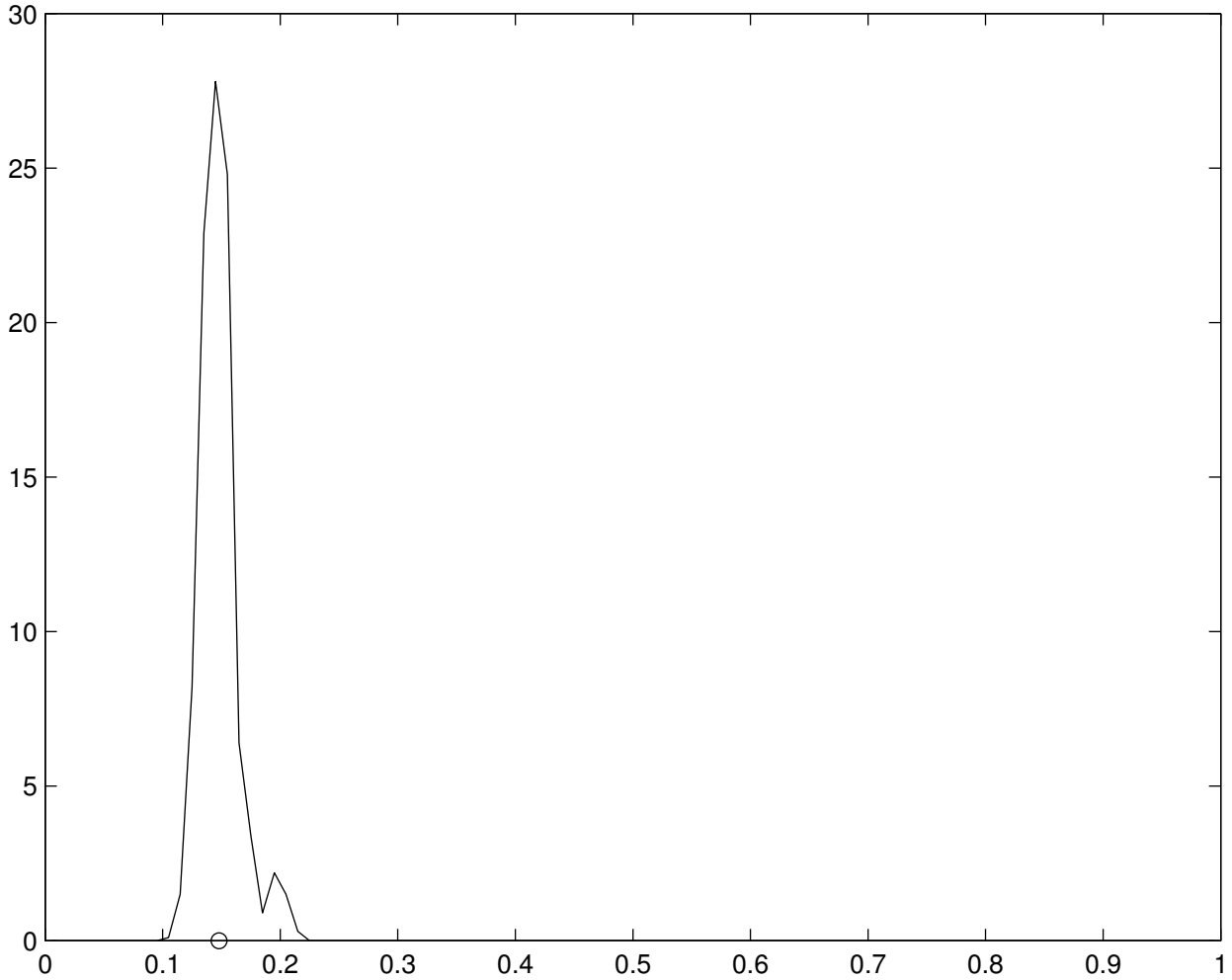
Error distribution: dataset size: 273



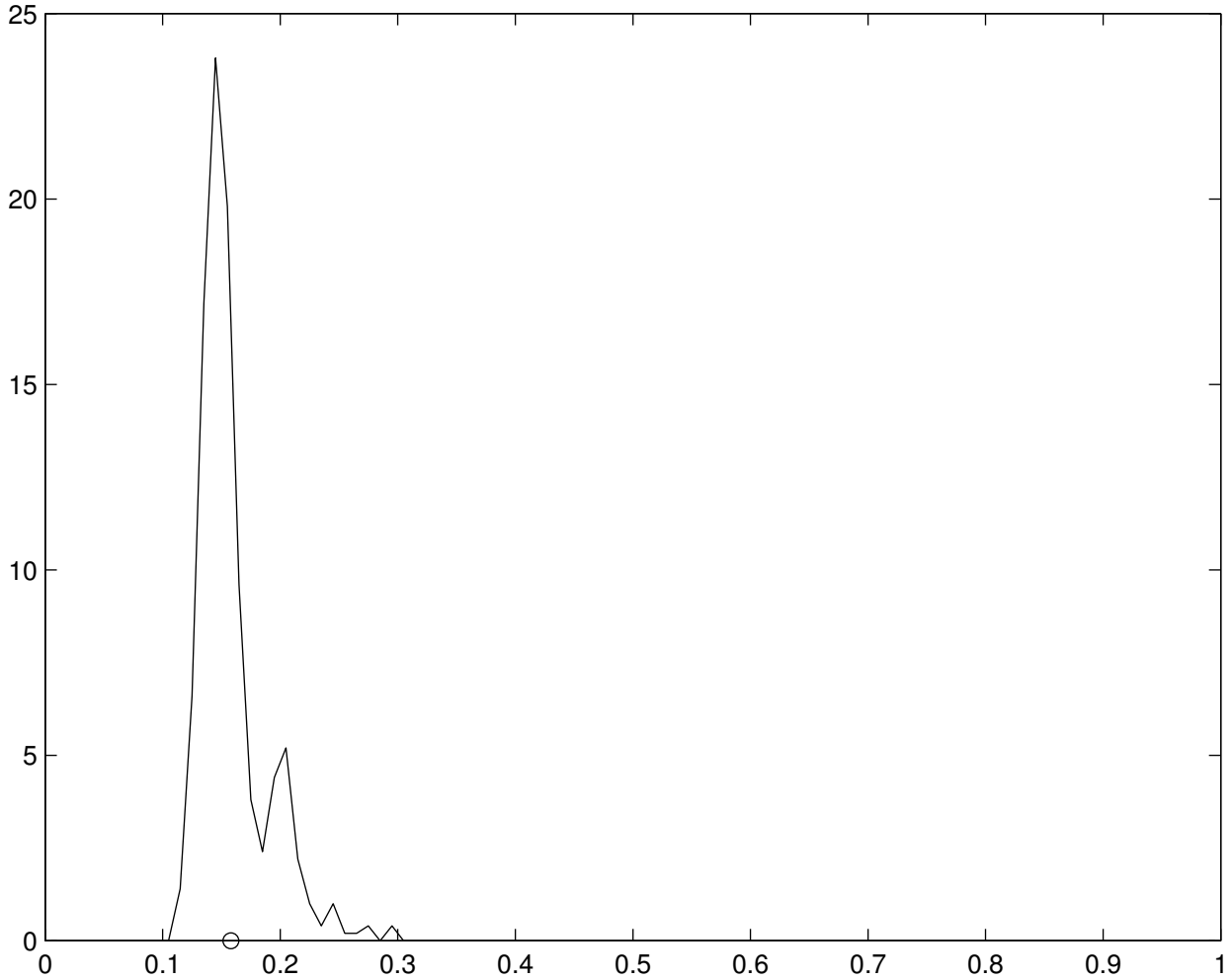
Error distribution: dataset size: 205



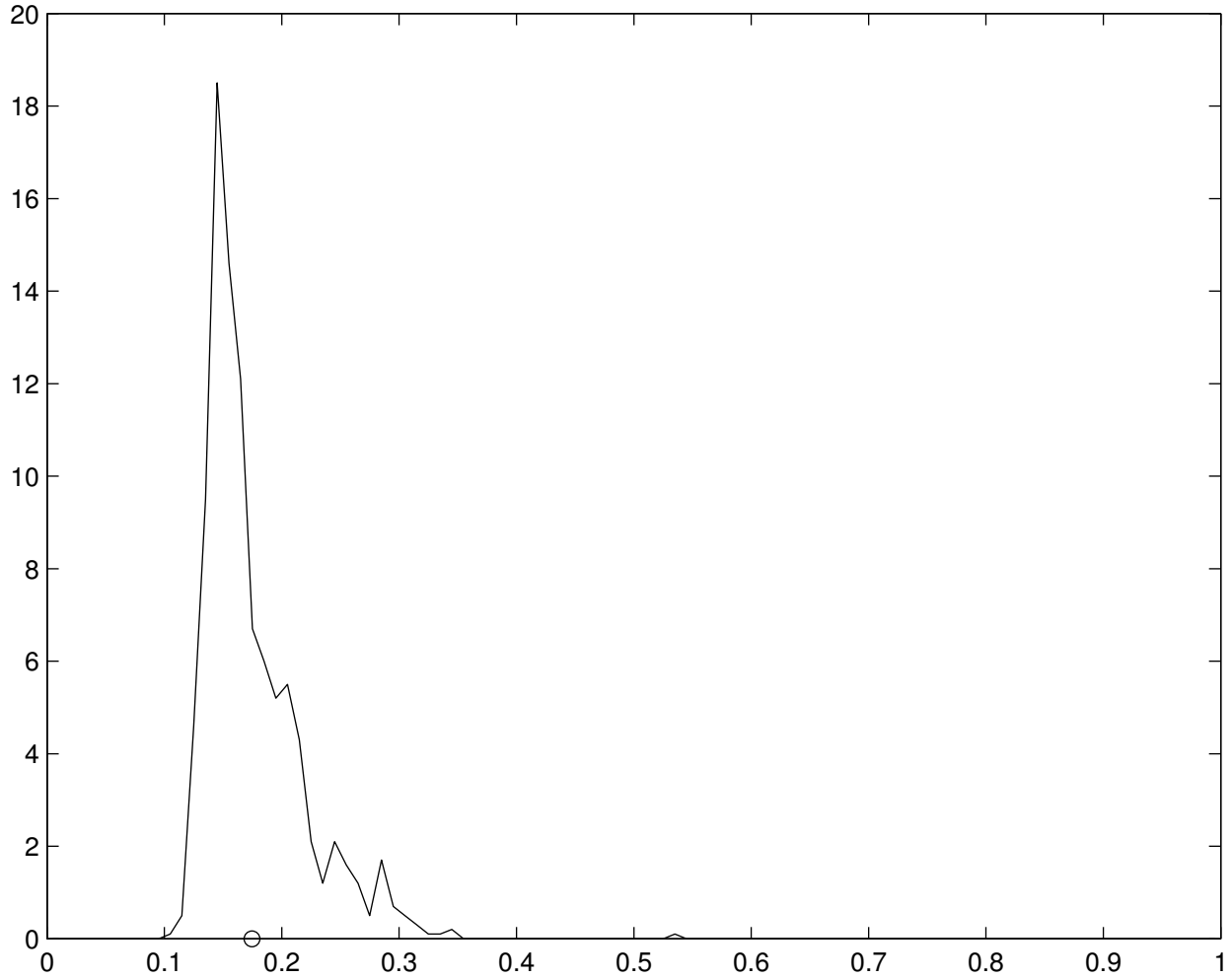
Error distribution: dataset size: 137



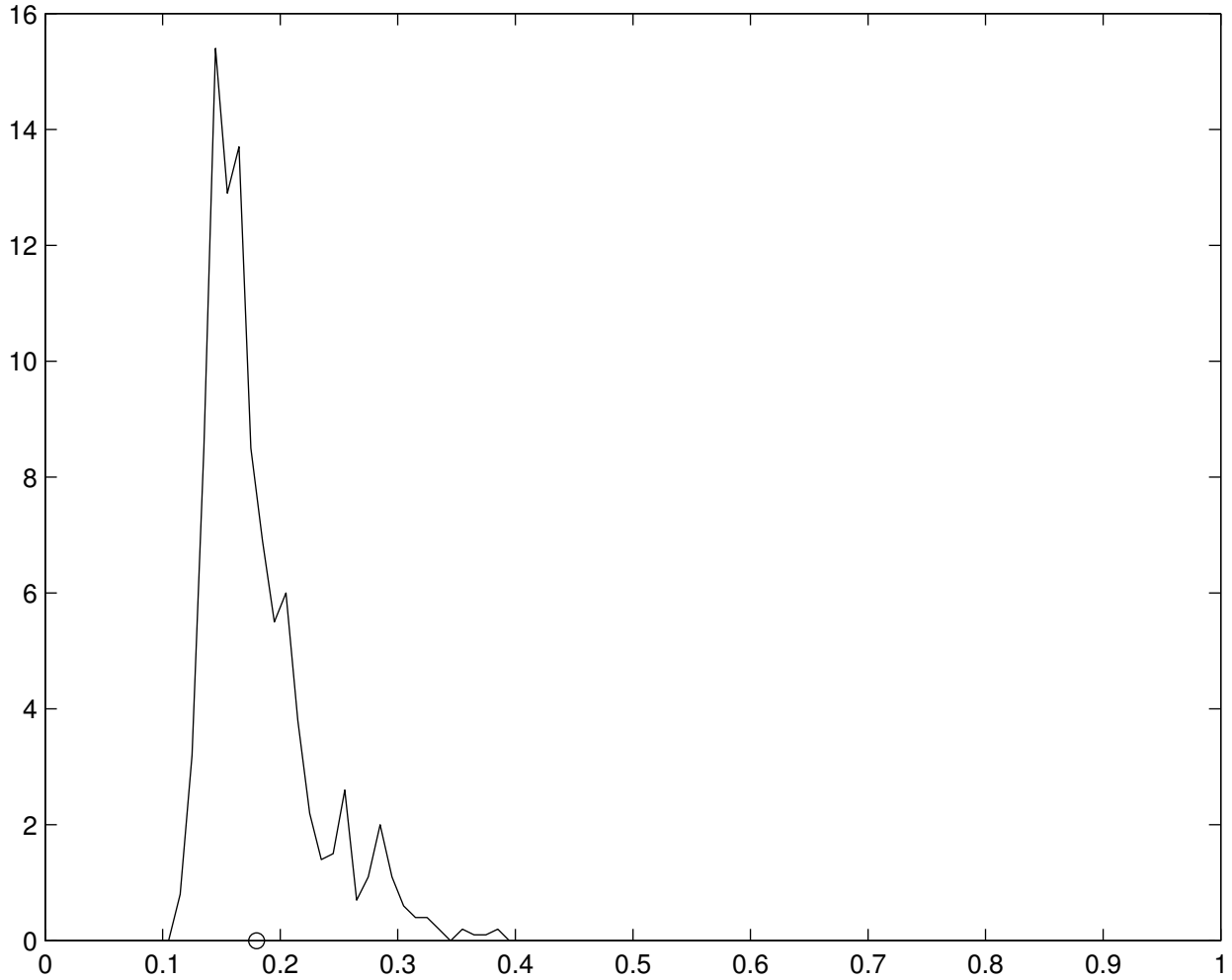
Error distribution: dataset size: 68



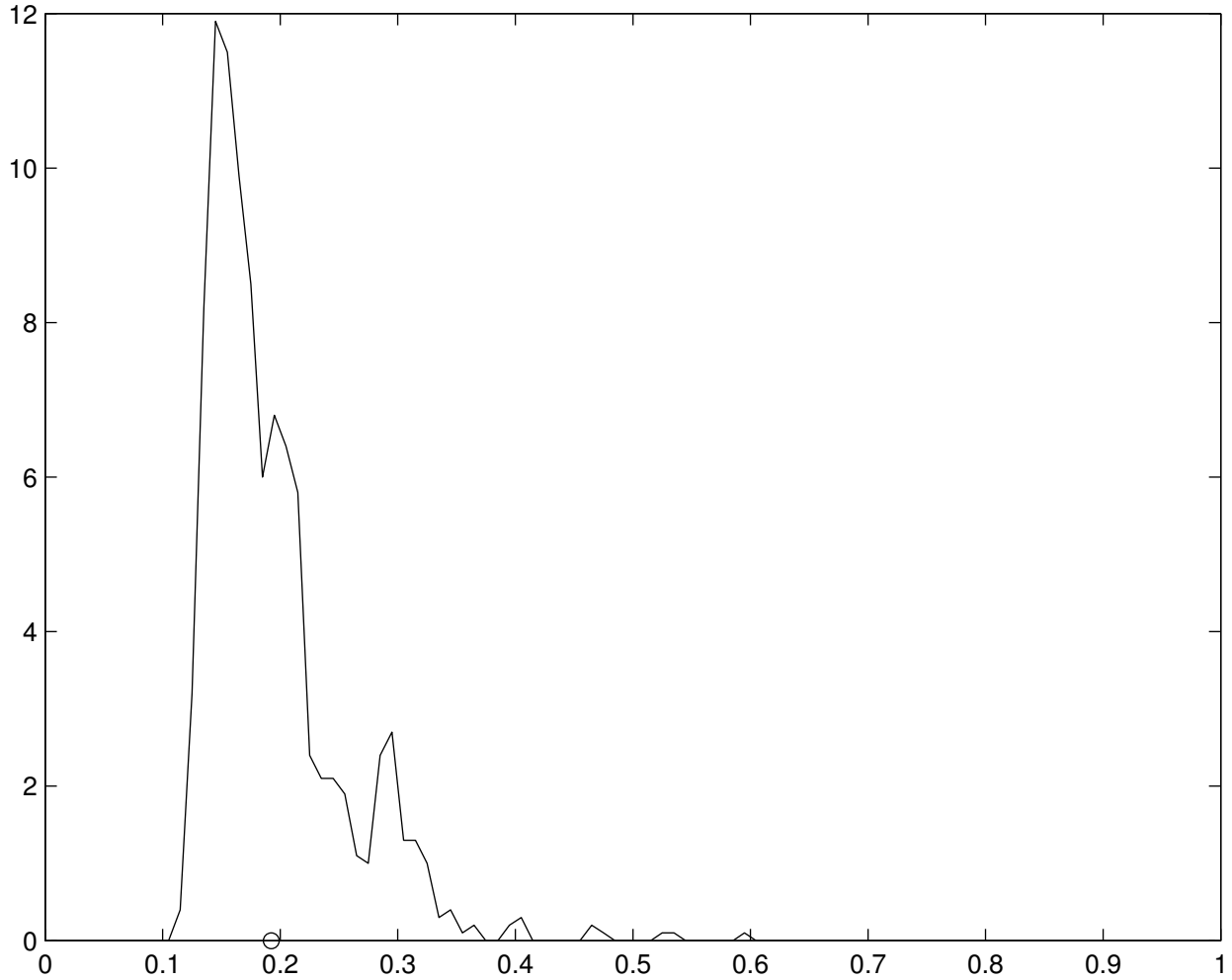
Error distribution: dataset size: 34



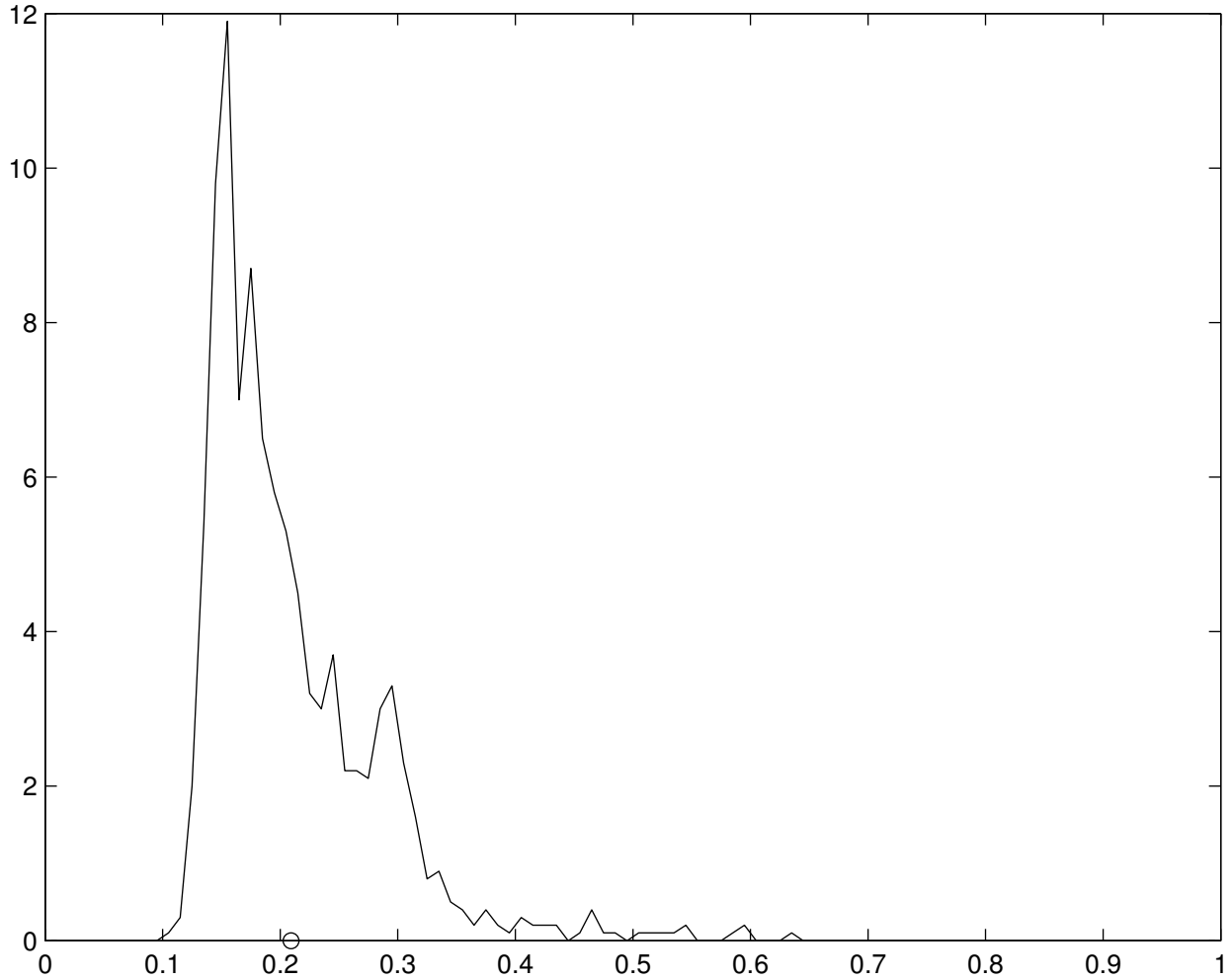
Error distribution: dataset size: 27



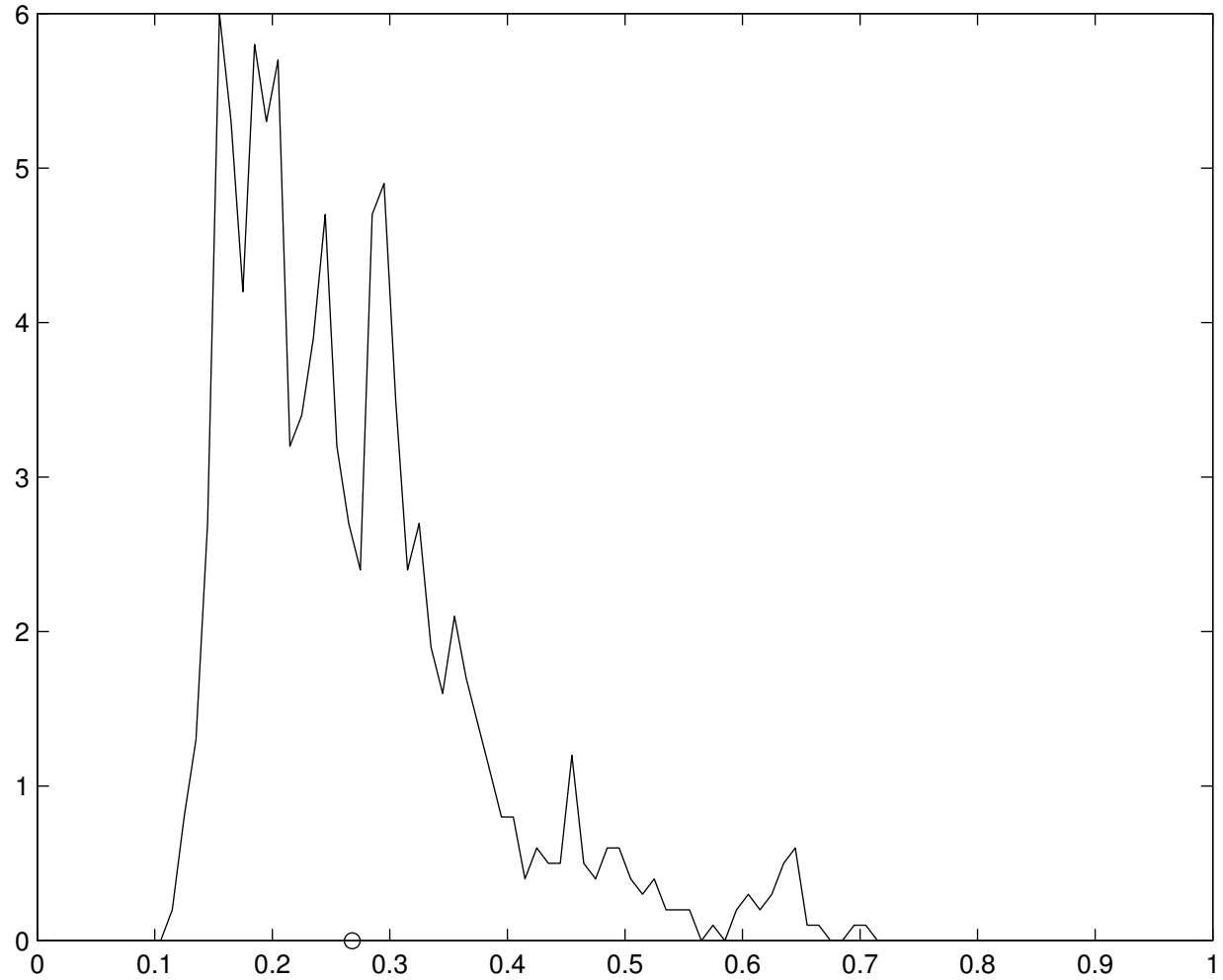
Error distribution: dataset size: 20



Error distribution: dataset size: 14



Error distribution: dataset size: 7

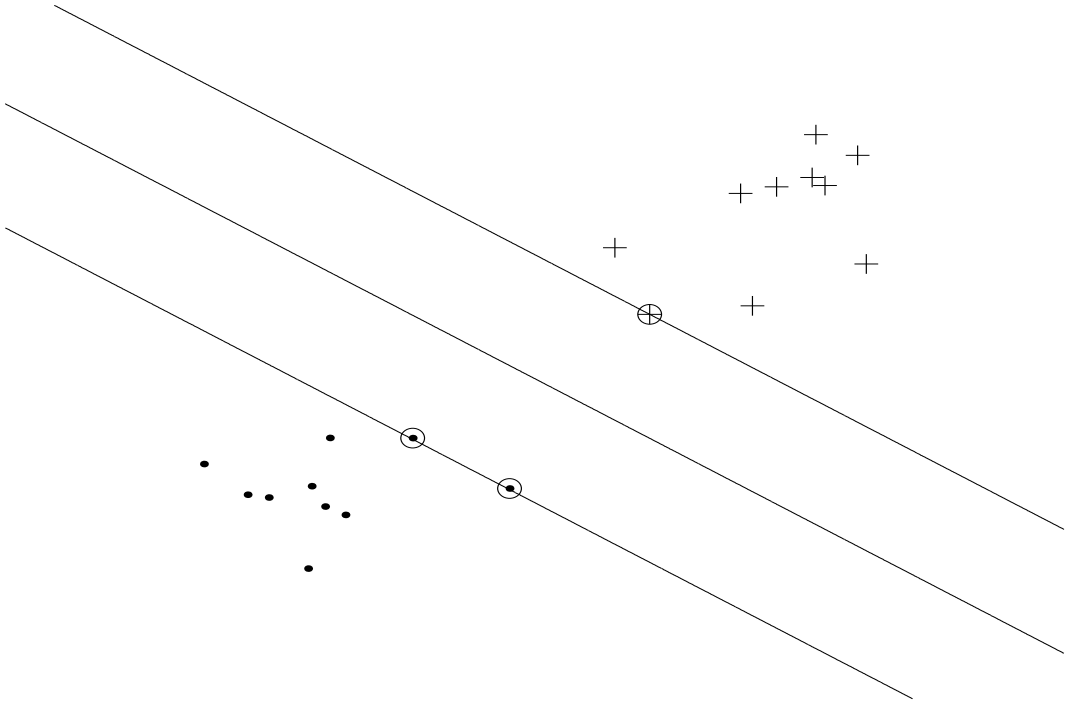


Observations

- Things can get bad if number of training examples small compared to dimension (in this case input dimension is 9)
- Mean can be bad predictor of true generalisation – i.e. things can look okay in expectation, but still go badly wrong
- Critical ingredient of learning – keep flexibility high while still ensuring good generalisation

Controlling generalisation

- The critical method of controlling generalisation is to force a large margin on the training data:



Intuitive explanations

- Makes classification robust to uncertainties in inputs
- Can randomly project into lower dimensional spaces and still have separation – so effectively low dimensional
- Rigorous statistical analysis shows effective dimension

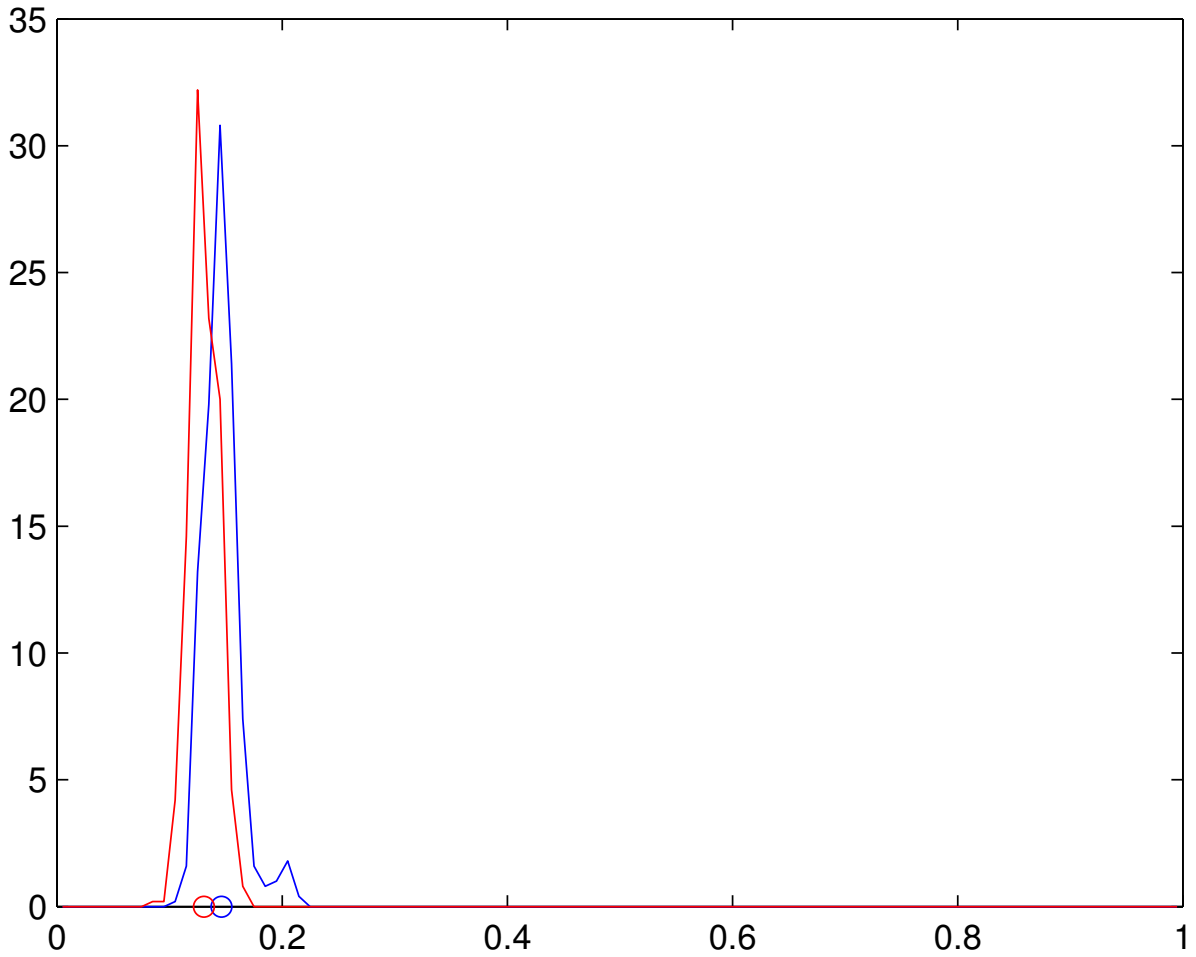
$$\propto 1/\gamma^2$$

Regularisation

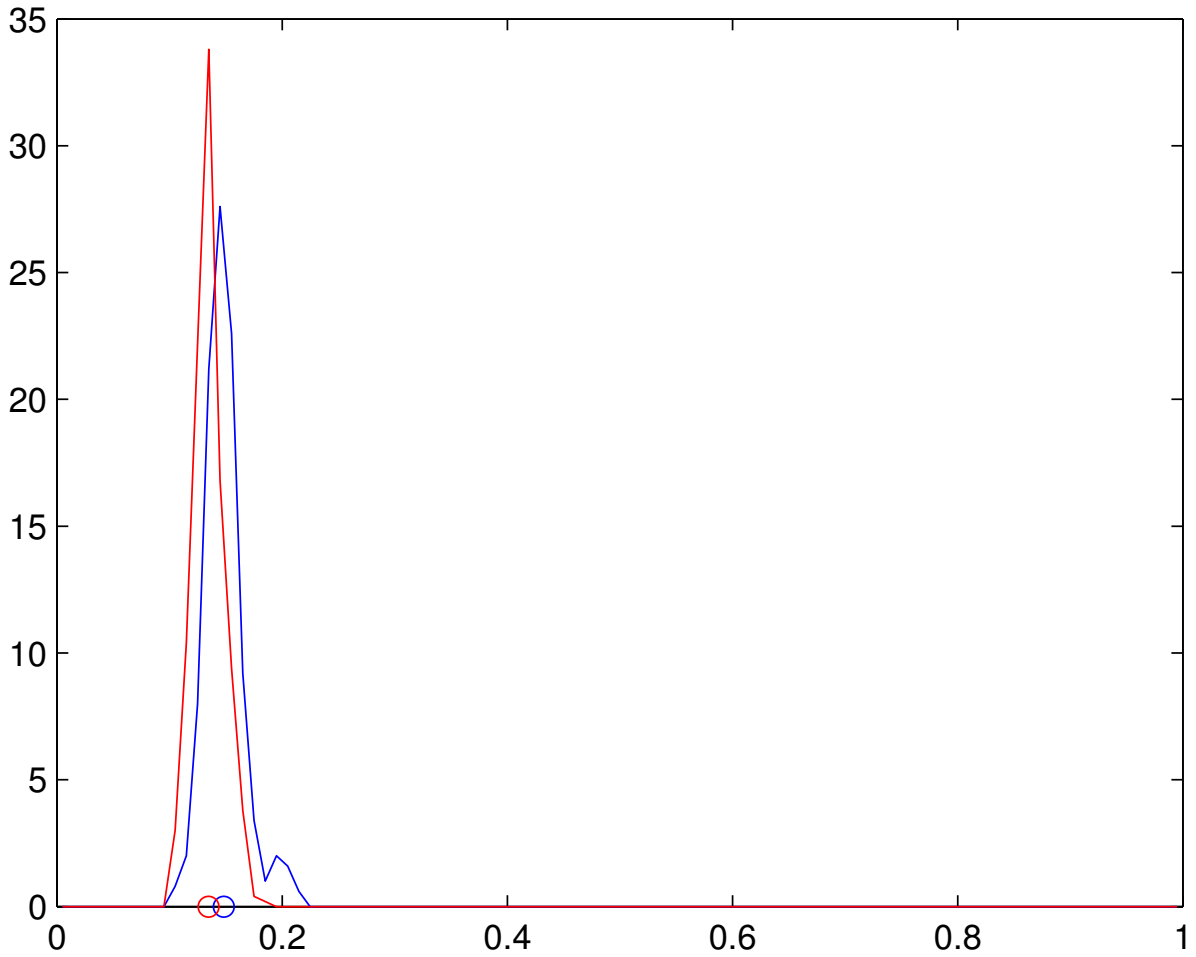
- Keeping a large margin is equivalent to minimising the norm of the weight vector while keeping outputs above a fixed value
- Controlling the norm of the weight vector is also referred to as regularisation, c.f. weight decay in neural network learning
- This is not structural risk minimisation since hierarchy depends on the data: data-dependent structural risk minimisation

  see S-T, Bartlett, Williamson & Anthony, 1998

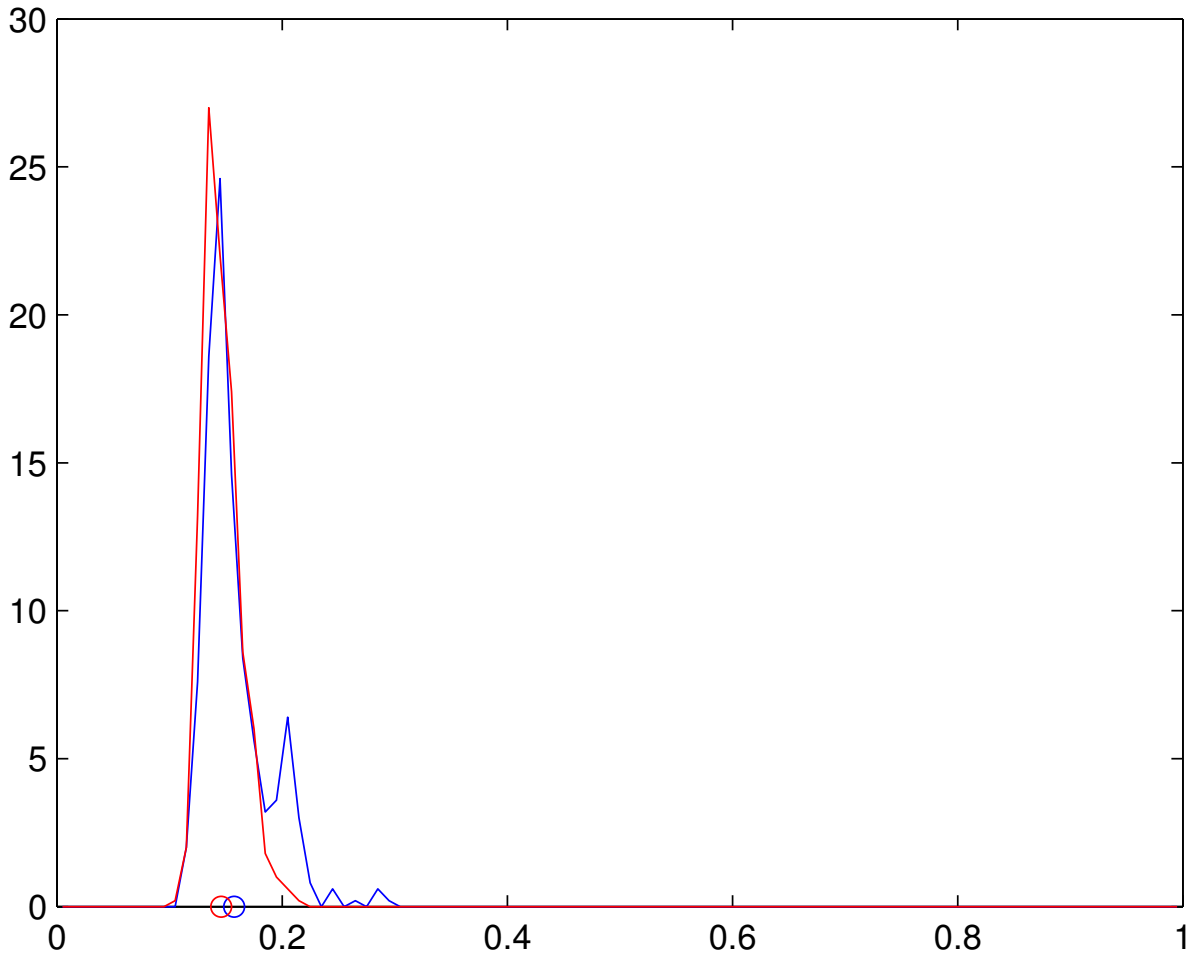
Error distribution: dataset size: 205



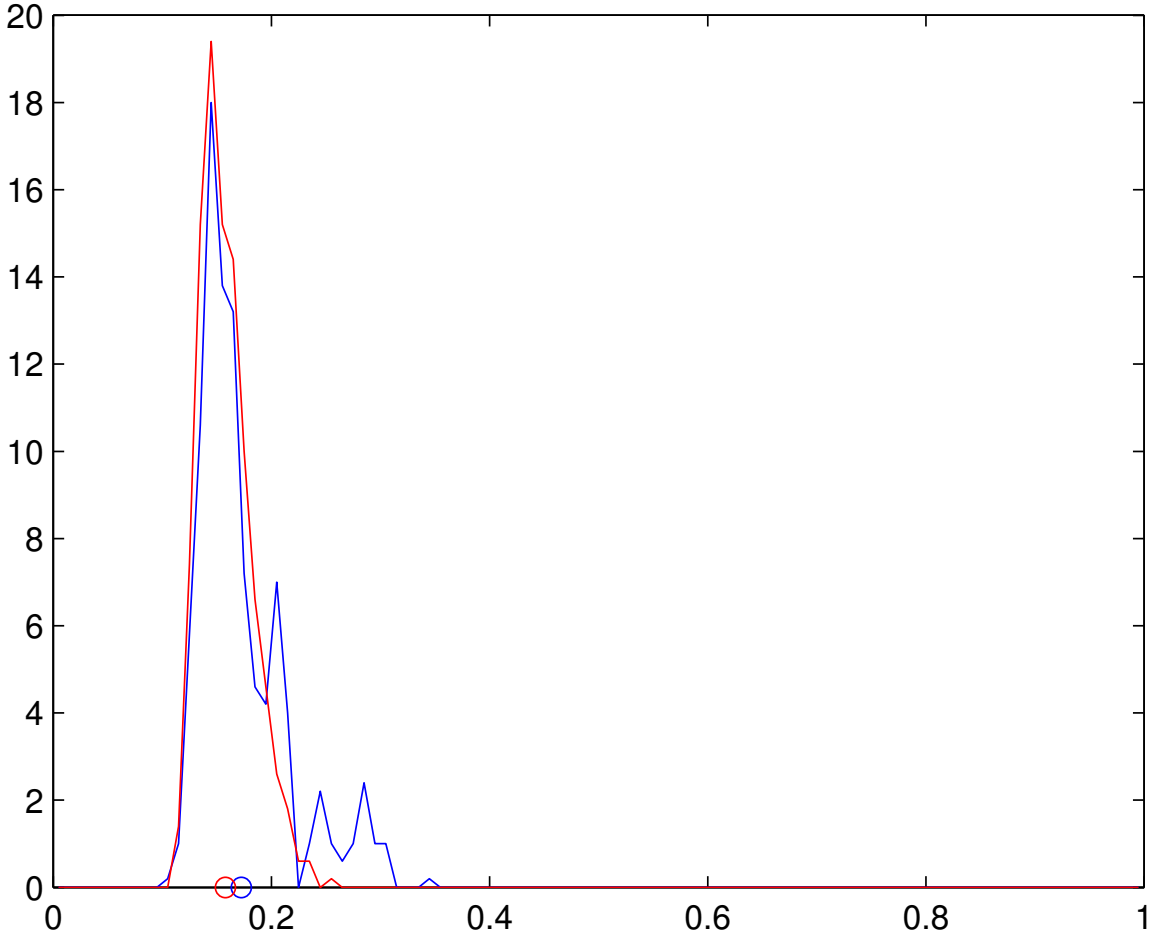
Error distribution: dataset size: 137



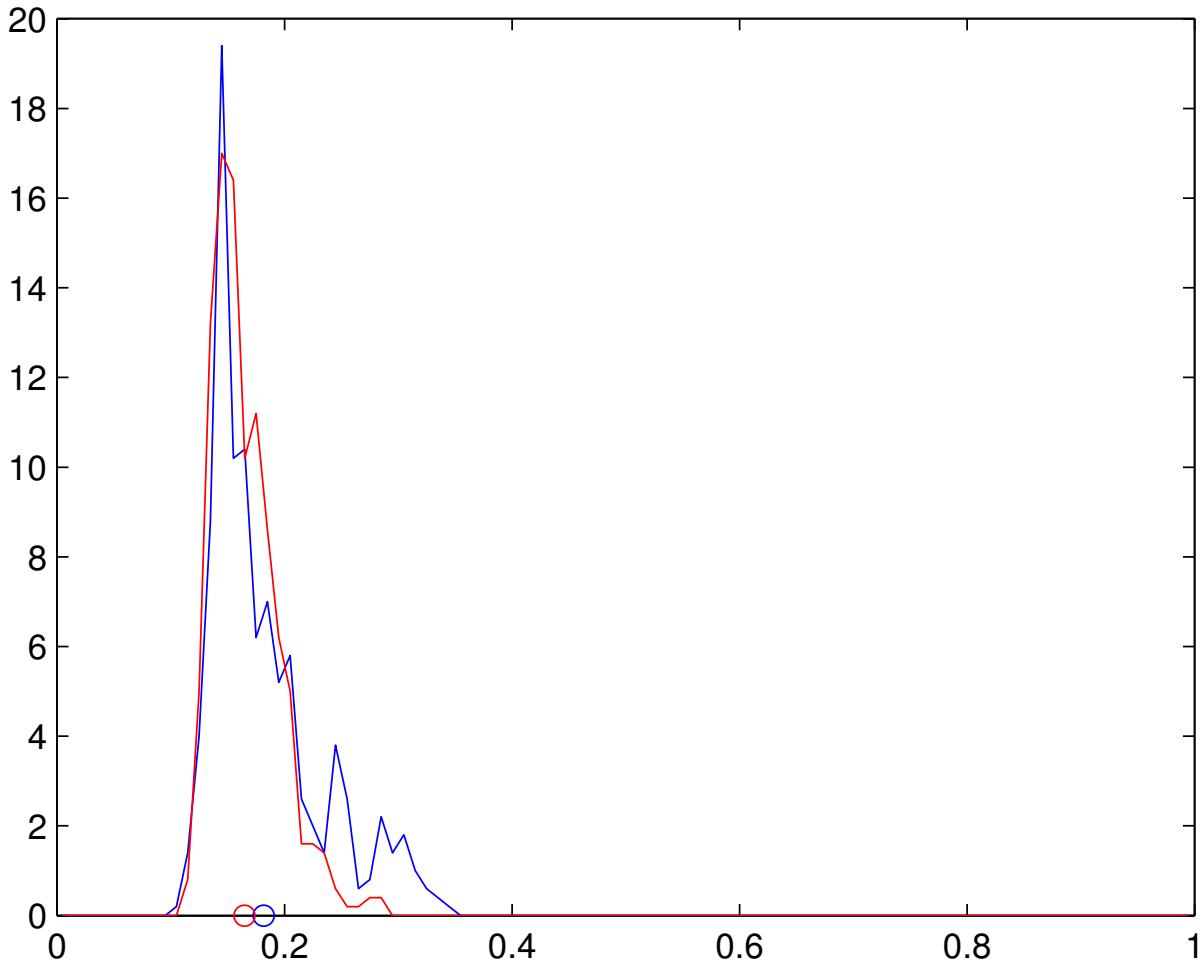
Error distribution: dataset size: 68



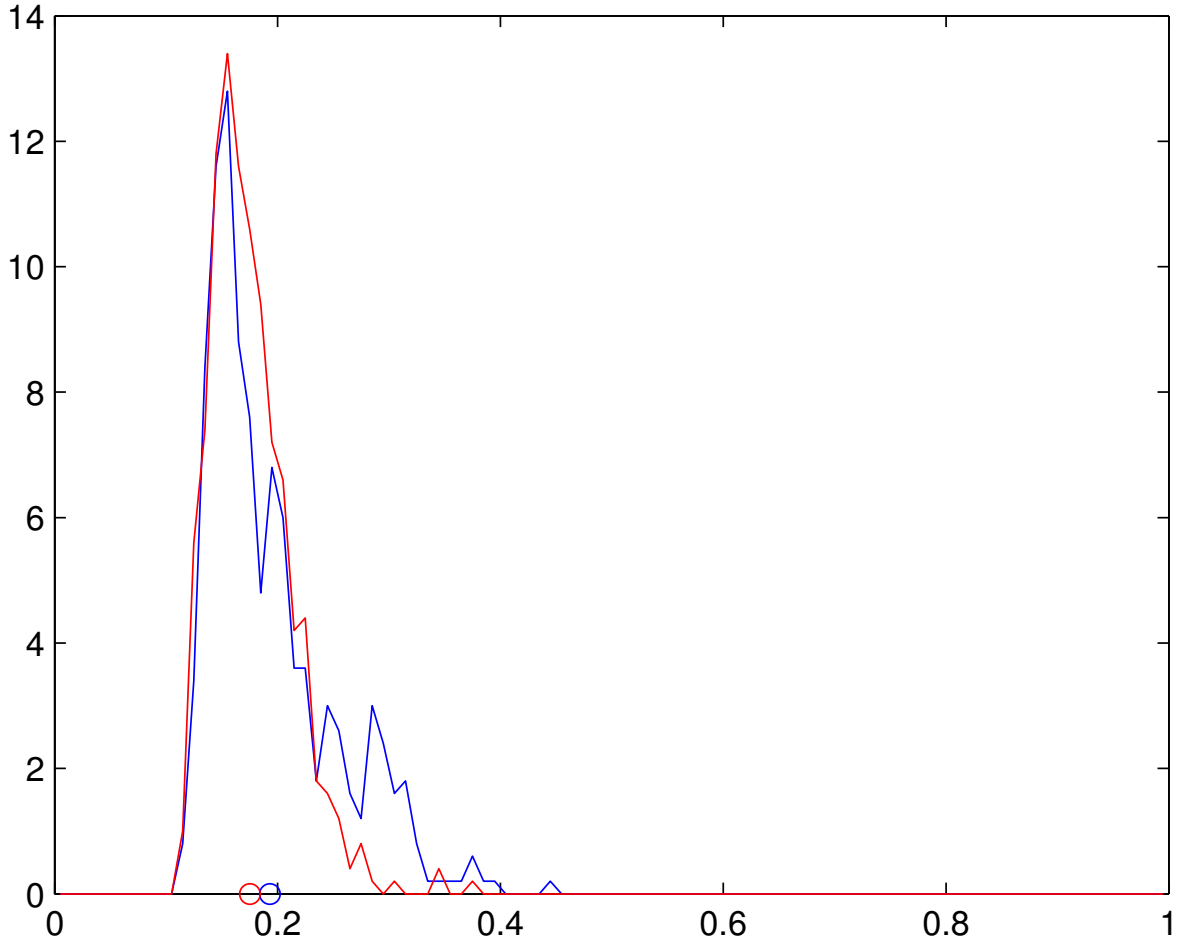
Error distribution: dataset size: 34



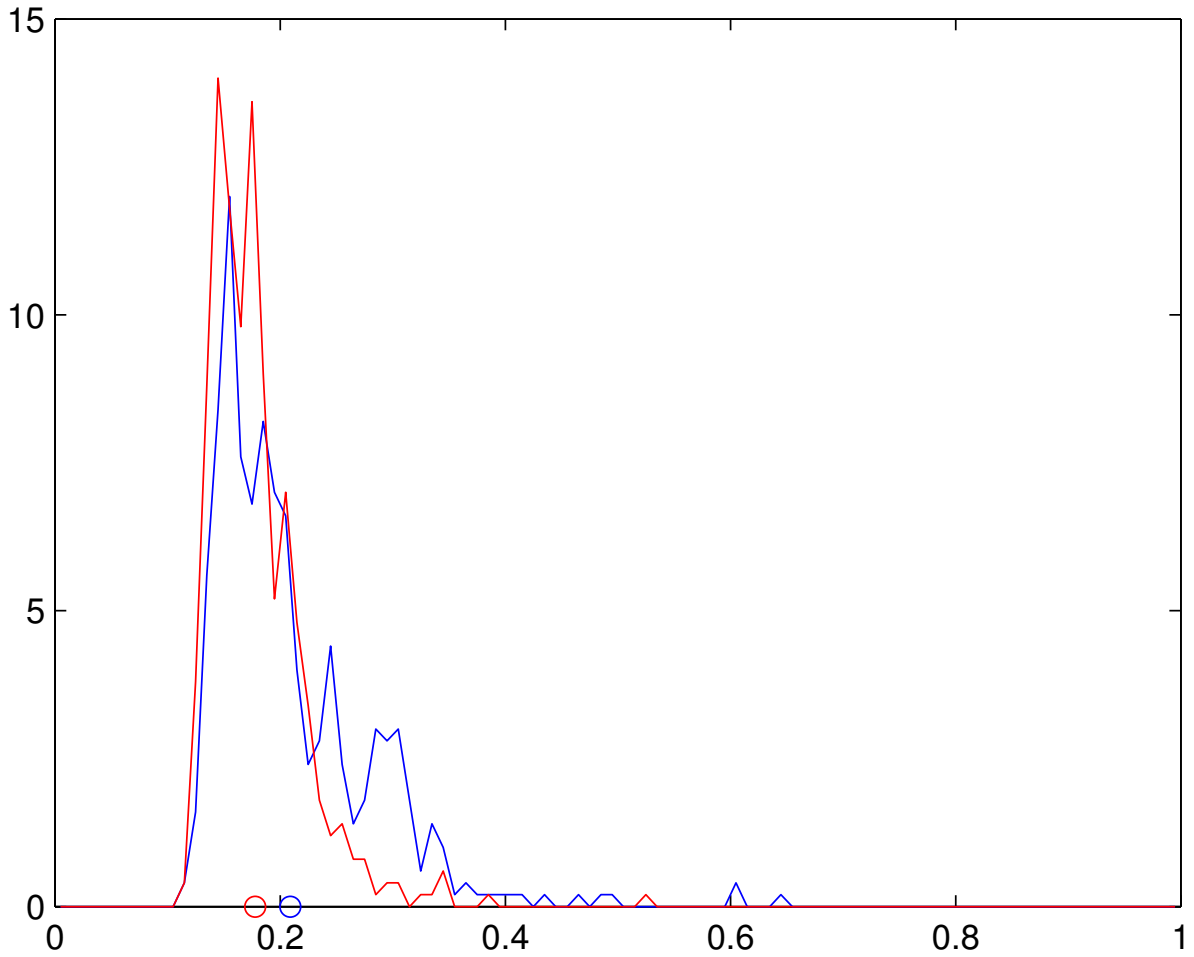
Error distribution: dataset size: 27



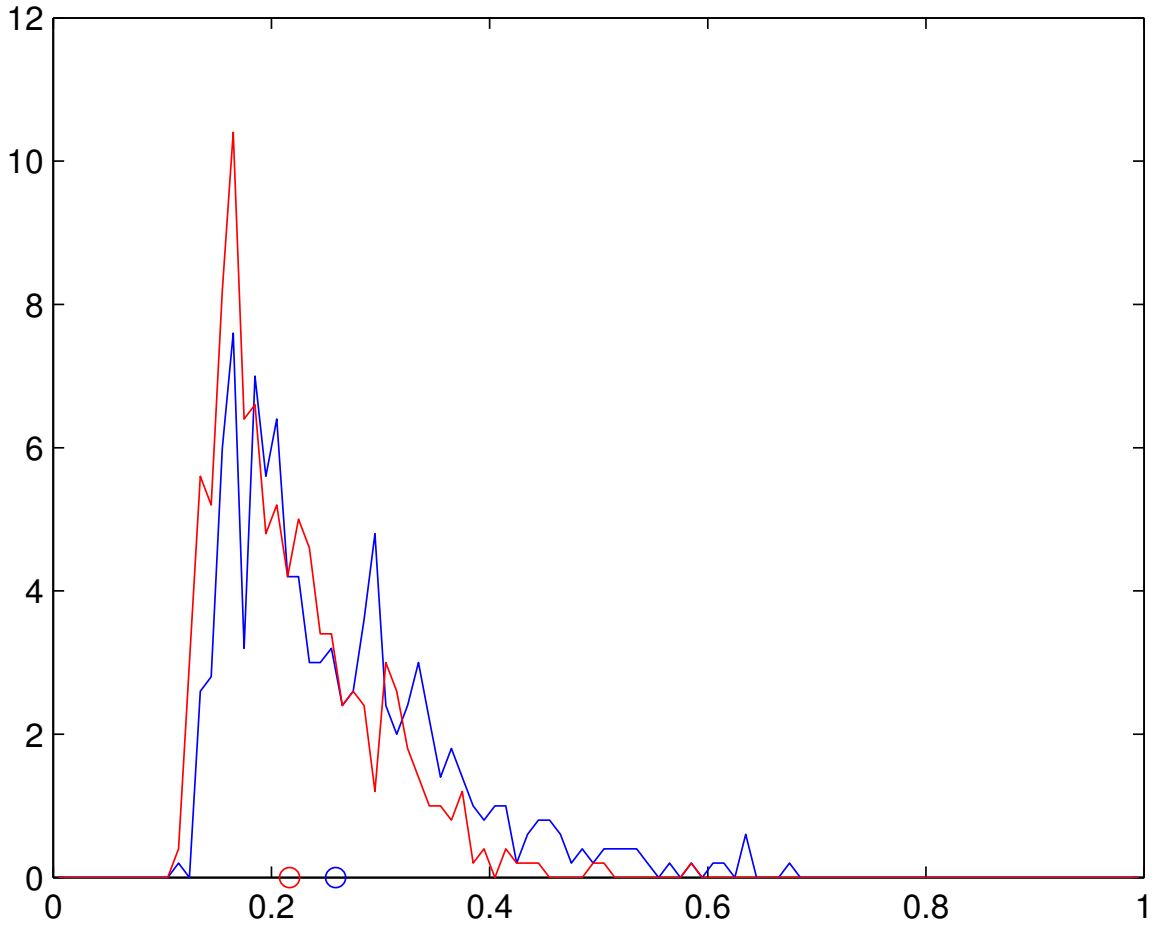
Error distribution: dataset size: 20



Error distribution: dataset size: 14



Error distribution: dataset size: 7



Support Vector Machines

- SVM optimisation

$$\min_{\mathbf{w},b,\xi} \quad 0.5\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{subj to:} \quad y_i\left(\langle \mathbf{w}, \phi(x_i)\rangle + b\right) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- Addresses generalisation issue but not the computational cost of dealing with large vectors

Complexity problem

- Let's apply the quadratic example

$$(x_1, x_2) \mapsto (x_1^2, x_1 x_2, x_2 x_1, x_2^2)$$

  to a 20x30 image of 600 pixels – gives approximately 180000 dimensions!

- Would be computationally infeasible to work in this space

Dual representation

- Suppose weight vector is a linear combination of the training examples:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \phi(x_i)$$

- can evaluate inner product with new example

$$\langle \mathbf{w}, \phi(x) \rangle = \left\langle \sum_{i=1}^{m} \alpha_i \phi(x_i), \phi(x) \right\rangle = \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x) \rangle$$

Learning the dual variables

- The α_i are known as dual variables
- Since any component orthogonal to the space spanned by the training data has no effect, general result that weight vectors have dual representation: the representer theorem.
- Hence, can reformulate algorithms to learn dual variables rather than weight vector directly

Dual form of SVM

- The dual form of the SVM can also be derived by taking the dual optimisation problem! This gives:

$$\max_{\alpha} \quad \sum_{i=1}^{m} \alpha_i - 0.5 \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle \phi(x_i), \phi(x_j) \right\rangle$$

$$\text{subj to:} \quad \sum_{i=1}^{m} y_i \alpha_i = 0, 0 \leq \alpha_i \leq C \text{ for all } i.$$

- Note that threshold must be determined from border examples

Using kernels

- Critical observation is that again only inner products are used
- Suppose that we now have a shortcut method of computing:

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

- Then we do not need to explicitly compute the feature vectors either in training or testing

Kernel example

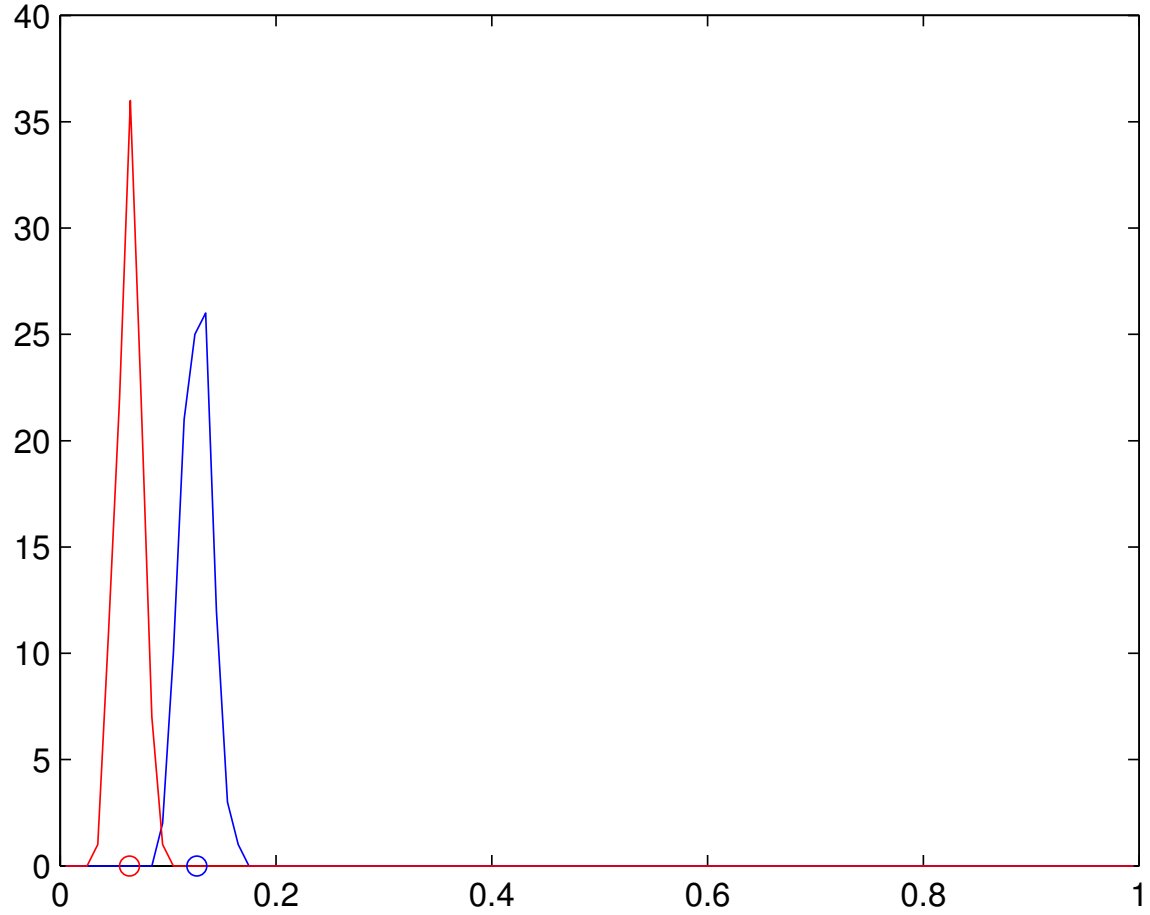
- As an example consider the mapping

$$\phi : (x_1, x_2) \mapsto (x_1^2, x_1 x_2, x_2 x_1, x_2^2)$$

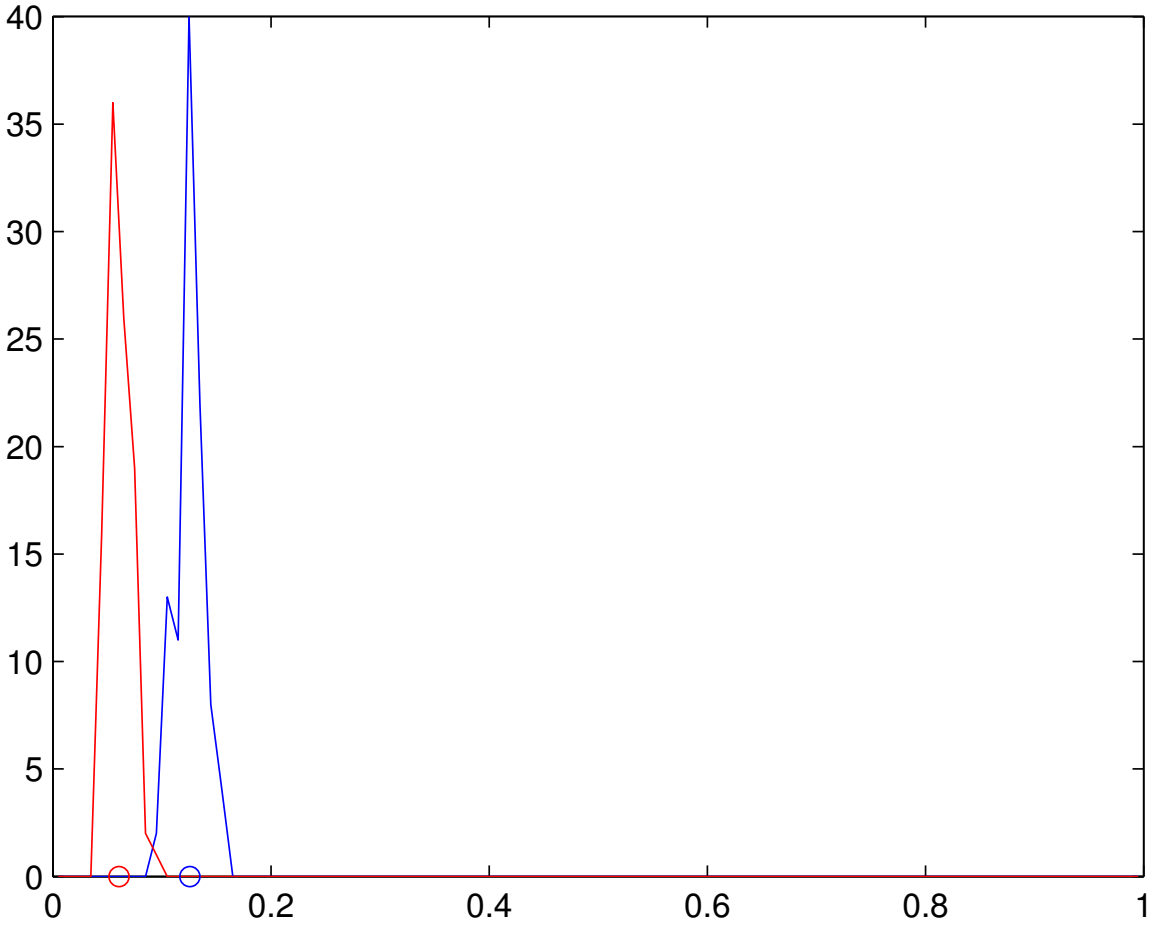
- Here we have a shortcut:

$$\begin{aligned} \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \left\langle \mathbf{x}\mathbf{x}', \mathbf{z}\mathbf{z}' \right\rangle_F \\ &= \mathrm{tr}(\mathbf{x}\mathbf{x}'\mathbf{z}\mathbf{z}') \\ &= \langle \mathbf{x}, \mathbf{z} \rangle^2 \end{aligned}$$

Using gaussian kernel for Breast: 273



Data size 342



Efficiency

- Hence, in the pixel example rather than work with 180000 dimensional vectors, we compute a 600 dimensional inner product and then square the result!
- Can even work in infinite dimensional spaces, eg using the Gaussian kernel:

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right)$$

Constraints on the kernel

- There is a restriction on the function:

$$\begin{aligned}\sum_{i,j=1}^{m} \beta_i \beta_j \kappa(x_i, x_j) &= \sum_{i,j=1}^{m} \beta_i \beta_j \langle \phi(x_i), \phi(x_j) \rangle \\ &= \left\langle \sum_{i=1}^{m} \beta_i \phi(x_i), \sum_{j=1}^{m} \beta_j \phi(x_j) \right\rangle \\ &= \left\| \sum_{i=1}^{m} \beta_i \phi(x_i) \right\|^2 \geq 0.\end{aligned}$$

- This restriction for any training set is enough to guarantee function is a kernel

What have we achieved?

- Replaced problem of neural network architecture by kernel definition
  - Arguably more natural to define but restriction is a bit unnatural
  - Not a silver bullet as fit with data is key
  - Can be applied to non- vectorial (or high dim) data
- Gained more flexible regularisation/ generalisation control
- Gained convex optimisation problem
  - i.e. NO local minima!

Brief look at algorithmics

- Have convex quadratic program
- Can apply standard optimisation packages – but don't exploit specifics of problem and can be inefficient
- Important to use chunking for large datasets
- But can use very simple gradient ascent algorithms for individual chunks

Kernel adatron

- If we fix the threshold to 0 (can incorporate learning by adding a constant feature to all examples), there is a simple algorithm that performs coordinate wise gradient descent:

$$\frac{\partial W(\alpha)}{\partial \alpha_i} = 1 - y_i \sum_{j=1}^{m} \alpha_j y_j \kappa(x_i, x_j) = 1 - y_i g(x_i)$$

$$\text{update: } \alpha_i \leftarrow \alpha_i + \kappa(x_i, x_i)^{-1}(1 - y_i g(x_i))$$

$$\text{then move } \alpha_i \text{ into interval } [0, C]$$

$$\text{where } g(x) = \sum_{j=1}^{m} \alpha_j y_j \kappa(x, x_j) \text{ is the learned function.}$$

Sequential Minimal Optimisation (SMO)

- SMO is the adaptation of kernel Adatron that retains the threshold and corresponding constraint:

$$\sum_{i=1}^{m} y_i \alpha_i = 0$$

by updating two coordinates at once.

Support vectors

- At convergence of kernel Adatron:

  either: $1 - y_i g(x_i) = 0$ or

  $\alpha_i = 0$ and $1 - y_i g(x_i) \leq 0$ or

  $\alpha_i = C$ and $1 - y_i g(x_i) \geq 0.$

- This implies sparsity:

$$y_i g(x_i) > 1 \;\Rightarrow\; \alpha_i = 0$$

$$y_i g(x_i) < 1 \;\Rightarrow\; \alpha_i = C$$

- Points with non-zero dual variables are Support Vectors – on or inside margin

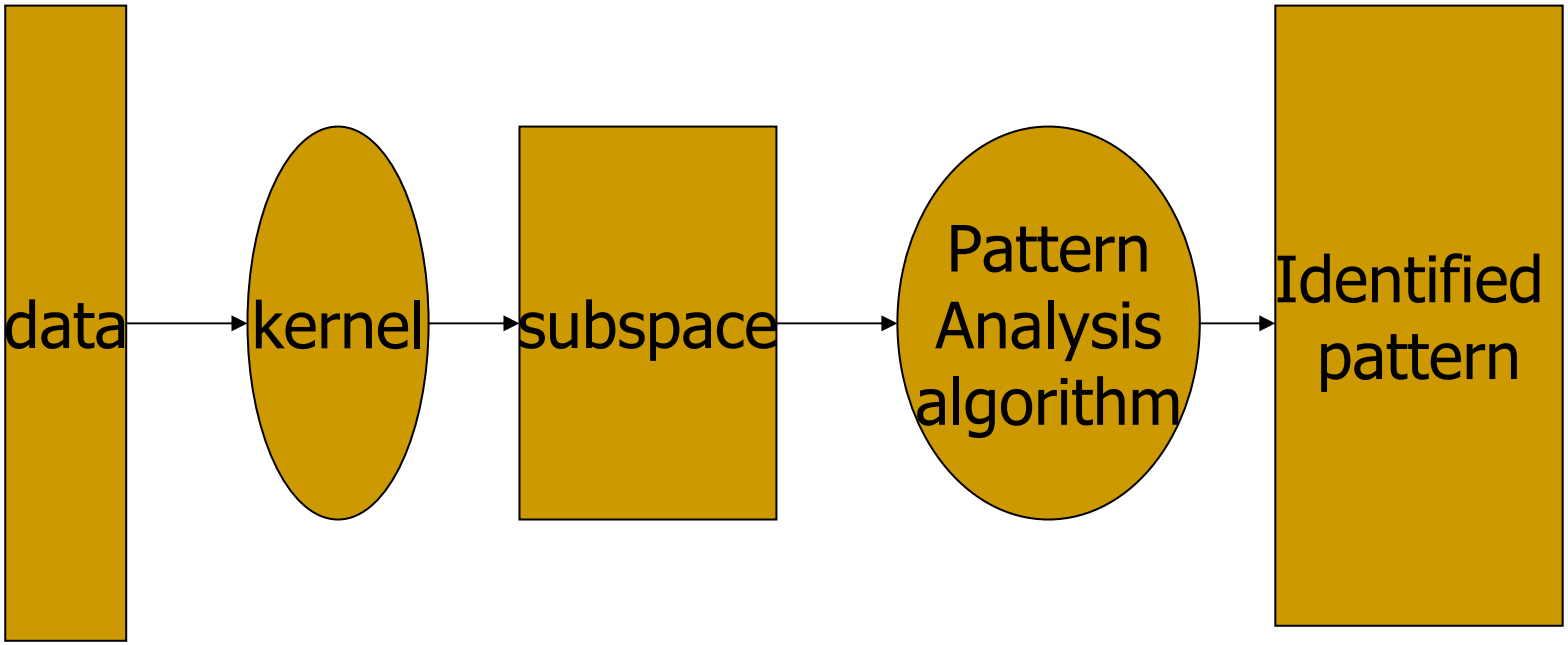
Issues in applying SVMs

- Need to choose a kernel:
  - Standard inner product
  - Polynomial kernel – how to choose degree
  - Gaussian kernel – but how to choose width
  - Specific kernel for different datatypes
- Need to set parameter C:
  - Can use cross-validation
  - If data is normalised often standard value of 1 is fine

Kernel methods topics

- Kernel methods are built on the idea of using kernel defined feature spaces for a variety of learning tasks – issues:
  - Kernels for different data
  - Other learning tasks and algorithms
  - Subspace techniques such as PCA for refining kernel definitions and CCA for combining two views of the data

Kernel methods: plug and play

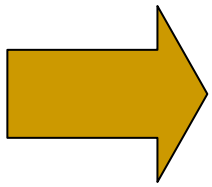


Kernels for other data

Kernels for text

- Bag of words model – Vector of term weights

The higher minimum wage signed into law… will be welcome relief for millions of workers … . The 90-cent-an-hour increase for … .



$x \in \mathbb{R}^n$

for	2
into	1
law	1
the	2
.	.
.	.
wage	1

IDF Weighting

- Term frequency weighting gives too much weight to frequent words
- Inverse document frequency weighting of words developed for information retrieval:

$$\mathrm{idf}(w) = \ln\left(\frac{m}{m_w}\right)$$

where m_w is no. of docs containing w out of m docs $\Rightarrow$ the removal of stop words.

Alternative: string kernel

- Features are indexed by k-tuples of characters
- Feature weight is count of occurrences of k-tuple as a subsequence down-weighted by its length
- Can be computed efficiently by a dynamic programming method

Example

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
ϕ(cat)	λ^2	λ^3	λ^2	0	0	0	0	0
ϕ(car)	λ^2	0	0	0	0	λ^3	λ^2	0
ϕ(bat)	0	0	λ^2	λ^2	λ^3	0	0	0
ϕ(bar)	0	0	0	λ^2	0	0	λ^2	λ^3

Example of recursion/dynamic programming

Without gap penalties and including all subsequences up to a given length is simpler:

$$\begin{aligned}
\kappa_0(s,t) &= 1, \\
\kappa_p(\varepsilon,s) &= 0, \text{ for } p > 0, \\
\kappa_p(sa,t) &= \kappa_p(s,t) + \sum_{k:t_k=a} \kappa_{p-1}(s, t(1:k-1)).
\end{aligned}$$

Other kernel types

- Kernels for structured data: eg trees, graphs, etc.
  - Can compute inner products efficiently using dynamic programming techniques even when an exponential number of features included
- Kernels from probabilistic models: eg Fisher kernels, P-kernels
  - Fisher kernels used for smoothly parametrised models: computes gradients of log probability
  - P-kernels consider family of models with each model providing one feature equal to the probability of the example in that model

Other learning tasks

Regression

- Supervised learning with real valued outputs
- Simplest is to consider least squares with regularisation:
  - Ridge regression
  - Gaussian process
  - Krieking
  - Least squares support vector machine

$$\min_{\mathbf{w}} \quad 0.5\lambda\|\mathbf{w}\|^2 + \sum_{i=1}^{m} \xi_i^2$$

$$\text{subj to:} \quad y_i - \langle \mathbf{w}, \phi(x_i) \rangle = \xi_i$$

Dual soln for Ridge Regression

- Simple derivation gives:

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

$$f(x) = \sum_{i=1}^{m} \alpha_i \kappa(x, x_i)$$

- We have lost sparsity – but with GP view gain useful probabilistic analysis, eg variance, evidence, etc.
- Support vector regression regains sparsity by using ε-insensitive loss:

$$\min_{\mathbf{w}} \quad 0.5\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{m} \xi_i^2$$

$$\text{subj to:} \quad \xi_i = \max(0, \text{abs}(y_i - f_{\mathbf{w}}(x_i)) - \varepsilon)$$

Other tasks

- Novelty detection, eg condition monitoring, fraud detection: possible solution is so-called one class SVM, or minimal hypersphere containing the data
- Ranking, eg recommender systems: can be made with similar margin conditions and generalisation bounds
- Clustering, eg k-means, spectral clustering: can be performed in a kernel defined feature space

Subspace techniques

Subspace techniques

- Classical method is principle component analysis – looks for directions of maximum variance, given by eigenvectors of covariance matrix

$$C = \sum_{i=1}^{m} \phi(x_i)\phi(x_i)' = \mathbf{X}'\mathbf{X}$$

where $\mathbf{X}$ has rows $\phi(x_i)$.

Dual representation of PCA

- Eigenvectors of kernel matrix give dual representation:

$$\begin{aligned} K\mathbf{v} &= \mathbf{X}\mathbf{X}'\mathbf{v} = \lambda\mathbf{v} \\ \Rightarrow C(\mathbf{X}'\mathbf{v}) &= \mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{v}) = \lambda\mathbf{X}'\mathbf{v} \\ \text{so that } \mathbf{u} &= \mathbf{X}'\mathbf{v} \text{ is PCA vector} \end{aligned}$$

- Means we can perform PCA projection in a kernel defined feature space: kernel PCA

Kernel PCA

- Need to take care of normalisation to obtain:

$$\langle \mathbf{u}, \phi(x) \rangle = \frac{1}{\sqrt{\lambda}} \sum_{i=1}^{m} \mathbf{v}_i \kappa(x_i, x)$$

where λ is the corresponding eigenvalue.

Issues

- How reliable are estimates obtained from a sample when working in such high dimensional spaces
- Some analysis showing that if low dimensional projection captures most of the variance in the training set, will do well on test examples as well
- Analysis of other subspace methods not always available

Related techniques

- Number of related techniques:
  - Probabilistic LSI (pLSI)
  - Non-negative Matrix Factorisation (NMF)
  - Multinomial PCA (mPCA)
  - Discrete PCA (DPCA)
- All can be viewed as alternative decompositions:

$$D' \approx CM \text{ (in LSI/PCA} \approx U_k(\Sigma_k V_k')\text{)}$$

where k columns of C are underlying components and M gives mixing to create different documents.

Different criteria

- Vary by:
  - Different constraints (eg non-negative entries)
  - Different prior distributions (eg Dirichlet, Poisson)
  - Different optimisation criteria (eg max likelihood, Bayesian)
- Unlike LSI typically suffer from local minima and so require EM type iterative algorithms to converge to solutions

Other subspace methods

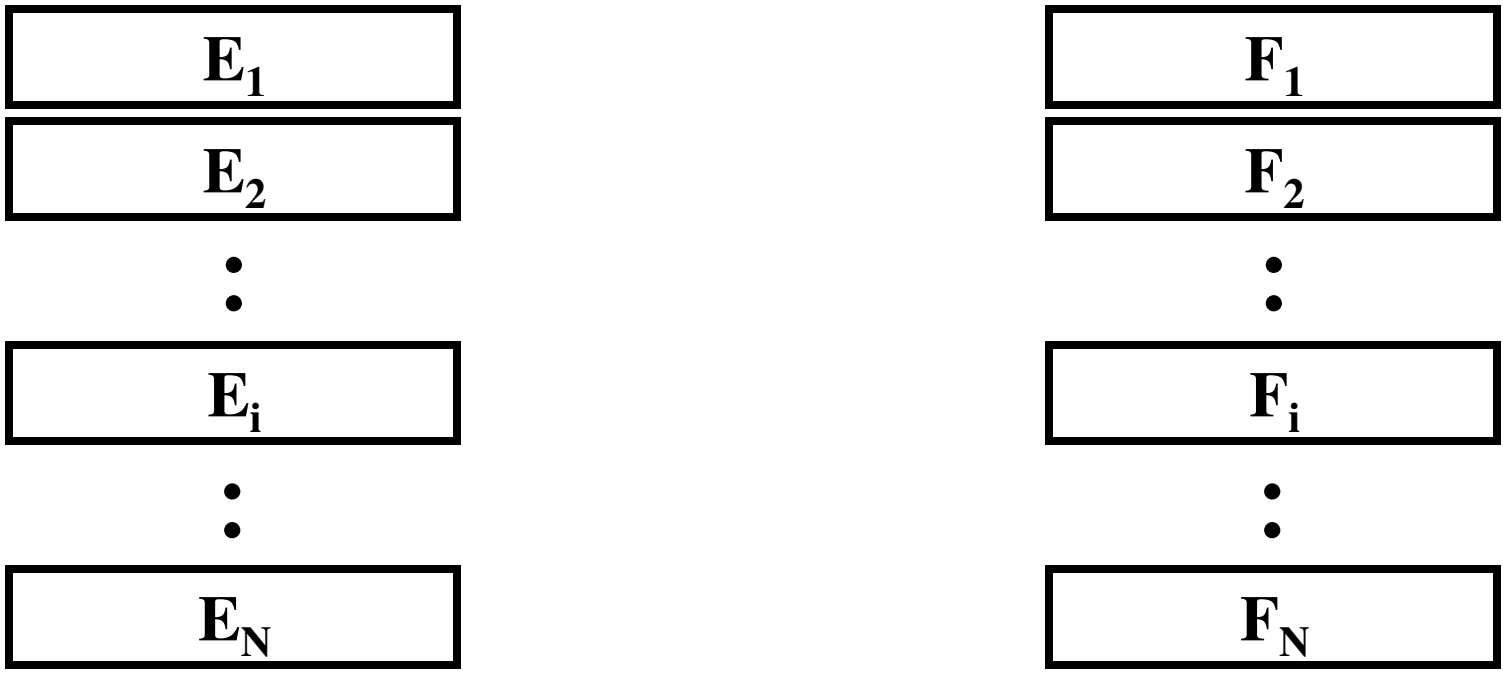
- Latent Semantic kernels equivalent to kPCA
- Kernel partial Gram-Schmidt orthogonalisation is equivalent to incomplete Cholesky decomposition – greedy kernel PCA
- Kernel Partial Least Squares implements a multi-dimensional regression algorithm popular in chemometrics – takes account of labels
- Kernel Canonical Correlation Analysis uses paired datasets to learn a semantic representation independent of the two views

Paired corpora

- Can we use information from paired corpora to extract more information?
- Two views of same semantic object – hypothesise that both views contain all of the necessary information, eg document and translation to a second language:

$$\phi_a(d) \longleftarrow d \longrightarrow \phi_b(d)$$

aligned text



Canadian parliament corpus

LAND MINES
Ms. Beth Phinney (Hamilton Mountain, Lib.):
Mr. Speaker, we are pleased that the Nobel peace prize has been given to those working to ban land mines worldwide.
We hope this award will encourage the United States to join the over 100 countries planning to come to …

E_{12}

LES MINES ANTIPERSONNEL
Mme Beth Phinney (Hamilton Mountain, Lib.):
Monsieur le Président, nous nous réjouissons du fait que le prix Nobel ait été attribué à ceux qui oeuvrent en faveur de l'interdiction des mines antipersonnel dans le monde entier.
Nous espérons que cela incitera les Américains à se joindre aux représentants de plus de 100 pays qui ont l'intention de venir à …

F_{12}

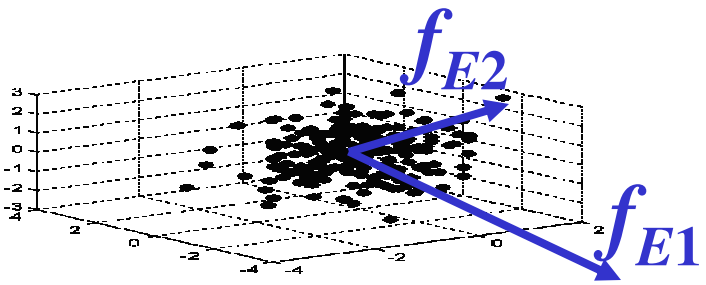
cross-lingual lsi via svd

$$D' = \begin{pmatrix} D'_E \\ D'_F \end{pmatrix} = \begin{pmatrix} U_E \\ U_F \end{pmatrix} \Sigma V^T$$

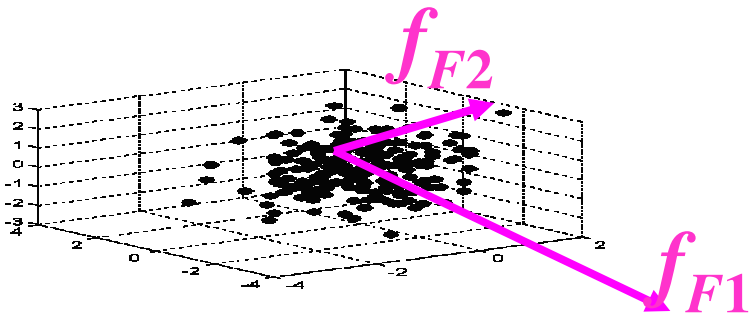
M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette, editor, *Cross-language information retrieval*. Kluwer, 1998.

cross-lingual kernel canonical correlation analysis

feature "English" space

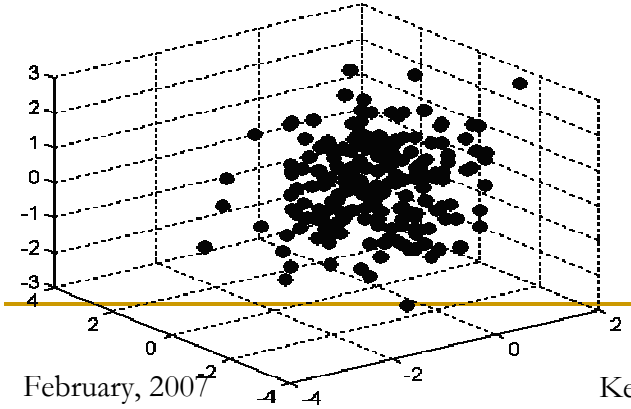


feature "French" space

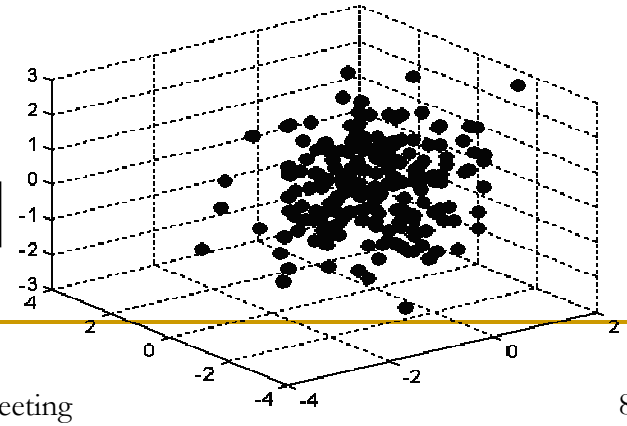


$\Phi(x)$

input "English" space



input "French" space



kernel canonical correlation analysis

$$\max_{f_E, f_F} corr(< f_E, \Phi_E(E) >, < f_F, \Phi_F(F) >)$$

$$f_E = \sum_l \alpha_l \Phi_E(E_l) \qquad f_F = \sum_l \beta_l \Phi_F(F_l)$$

$$\boxed{B\xi = \rho D\xi}$$

$$B = \begin{pmatrix} O & K_E K_F \\ K_F K_E & O \end{pmatrix} \quad D = \begin{pmatrix} K_E^2 & O \\ O & K_F^2 \end{pmatrix} \quad \xi = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

regularization

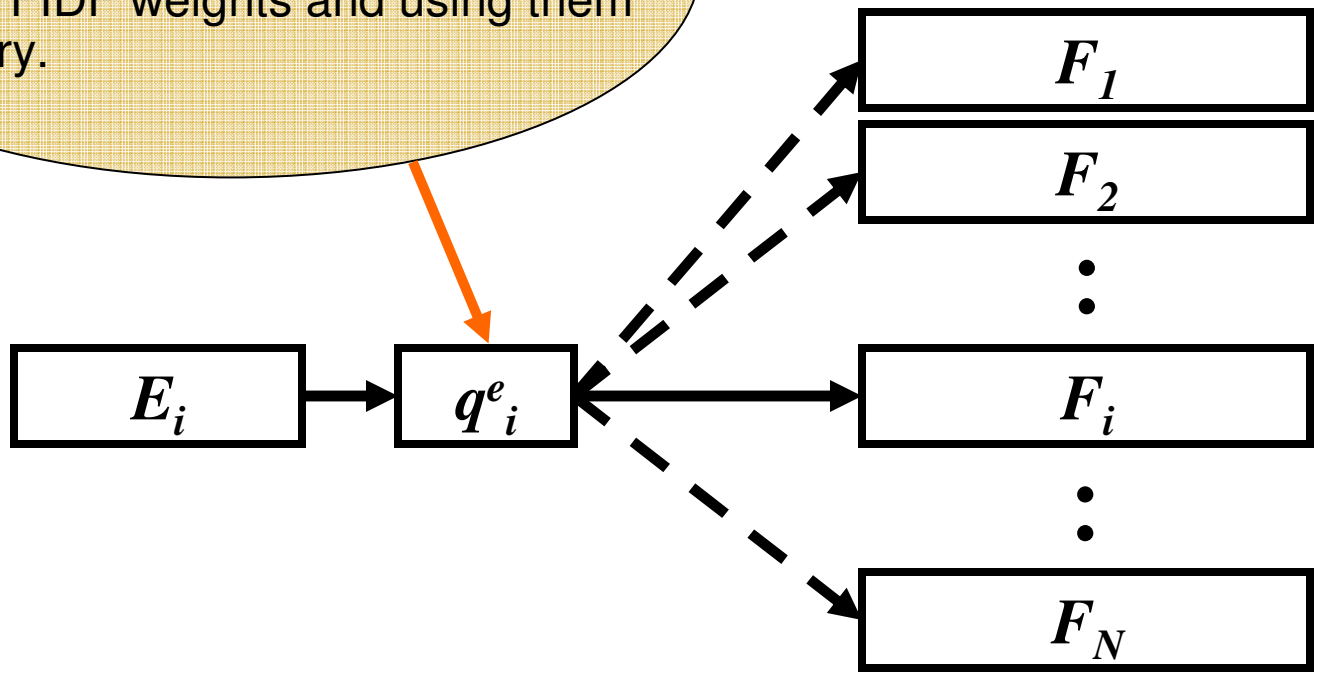
- using kernel functions may result in overfitting
- Theoretical analysis shows that provided the norms of the weight vectors are small the correlation will still hold for new data
- need to control flexibility of the projections f_E and f_F: add diagonal to the matrix D:

$$D = \begin{pmatrix} (K_E + \kappa I)^2 & O \\ O & (K_F + \kappa I)^2 \end{pmatrix}$$

κ is the regularization parameter

pseudo query test

Queries were generated from each test document by extracting 5 words with the highest TFIDF weights and using them as a query.



Experimental Results

The goal was to retrieve the paired document.

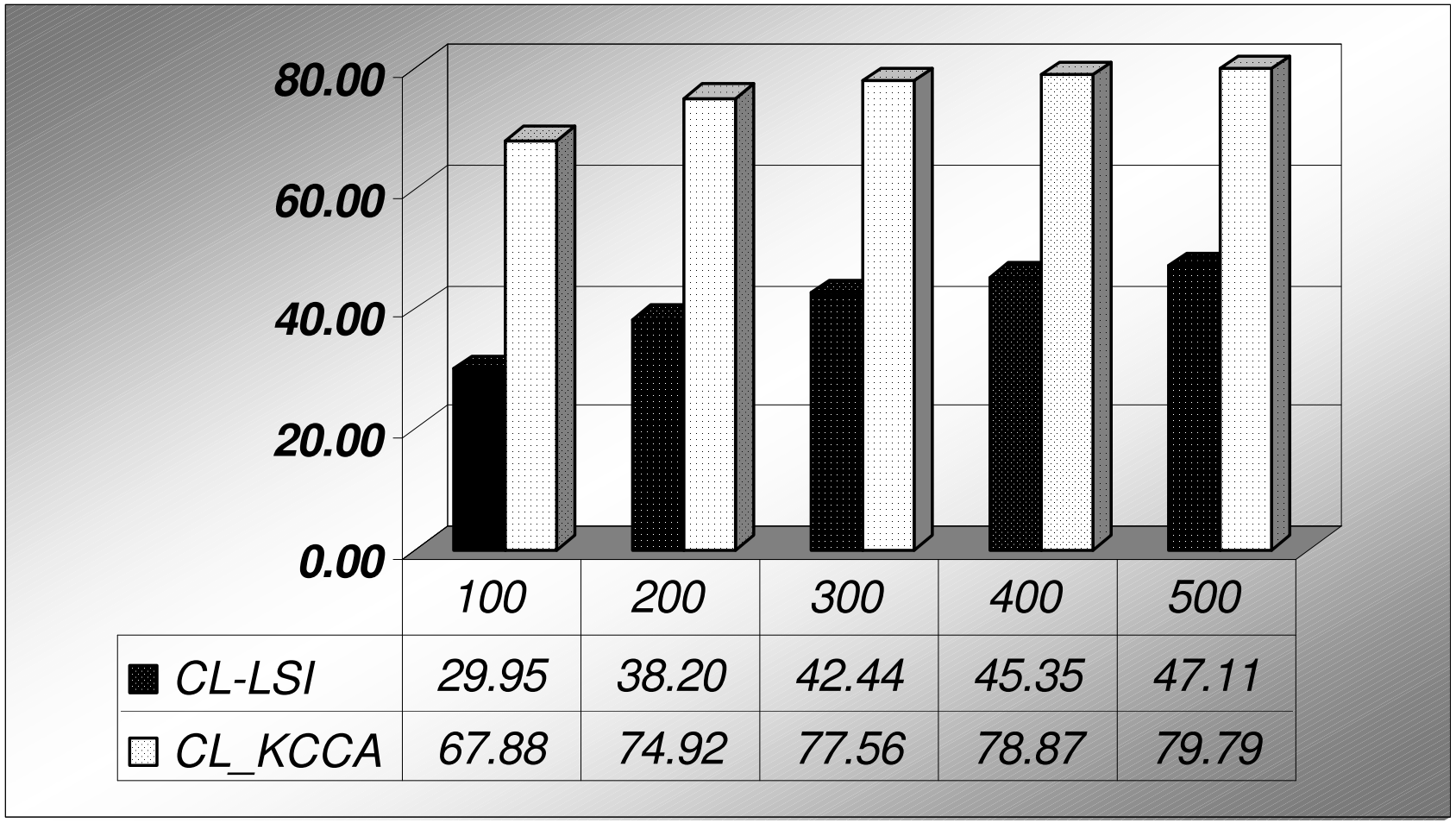
Experimental procedure:

(1) LSI/KCCA trained on paired documents,

(2) All test documents projected into the LSI/KCCA semantic space,

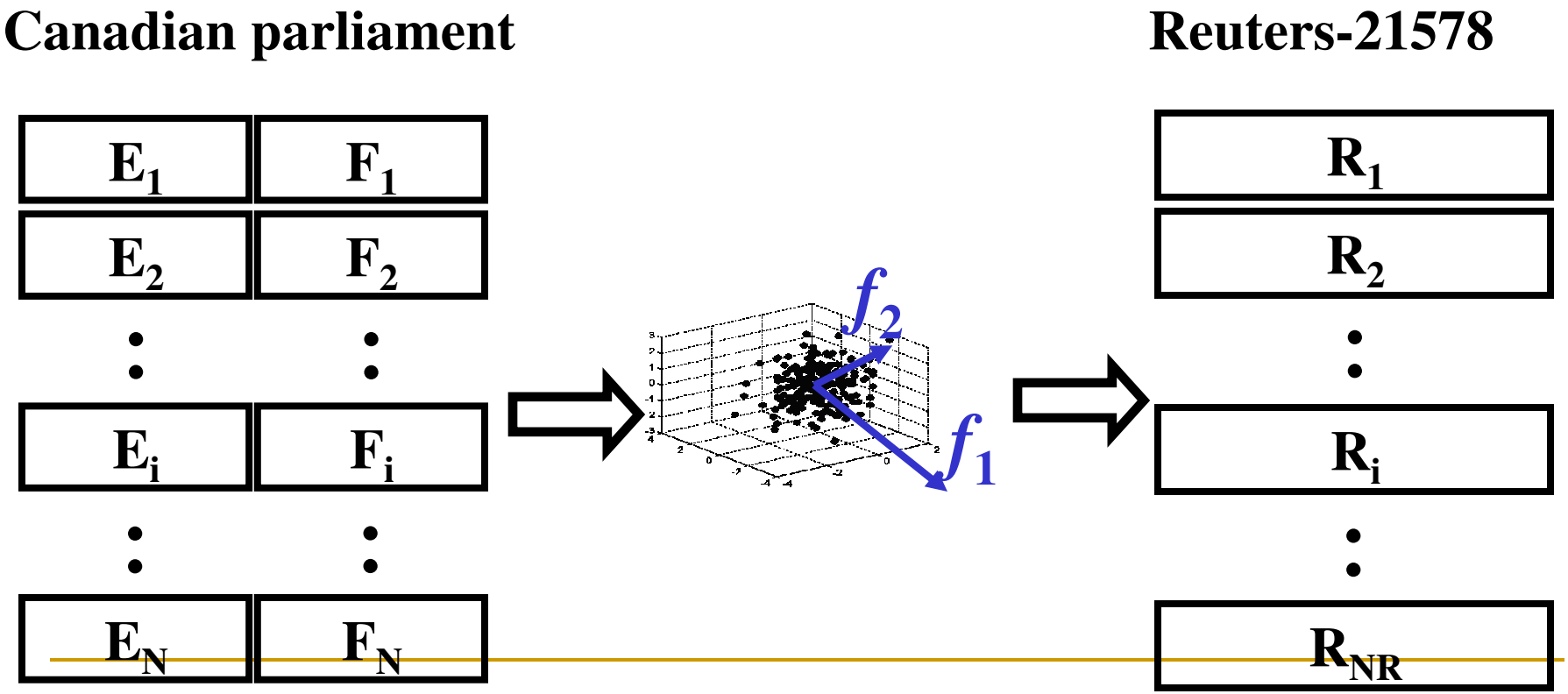
(3) Each query was projected into the LSI/KCCA semantic space and documents were retrieved using nearest neighbour based on cosine distance to the query.

English-French retrieval accuracy, %



	100	200	300	400	500
CL-LSI	29.95	38.20	42.44	45.35	47.11
CL_KCCA	67.88	74.92	77.56	78.87	79.79

using semantics, extracted from aligned corpus, for completely different corpora



Classification Procedure

Experimental procedure:

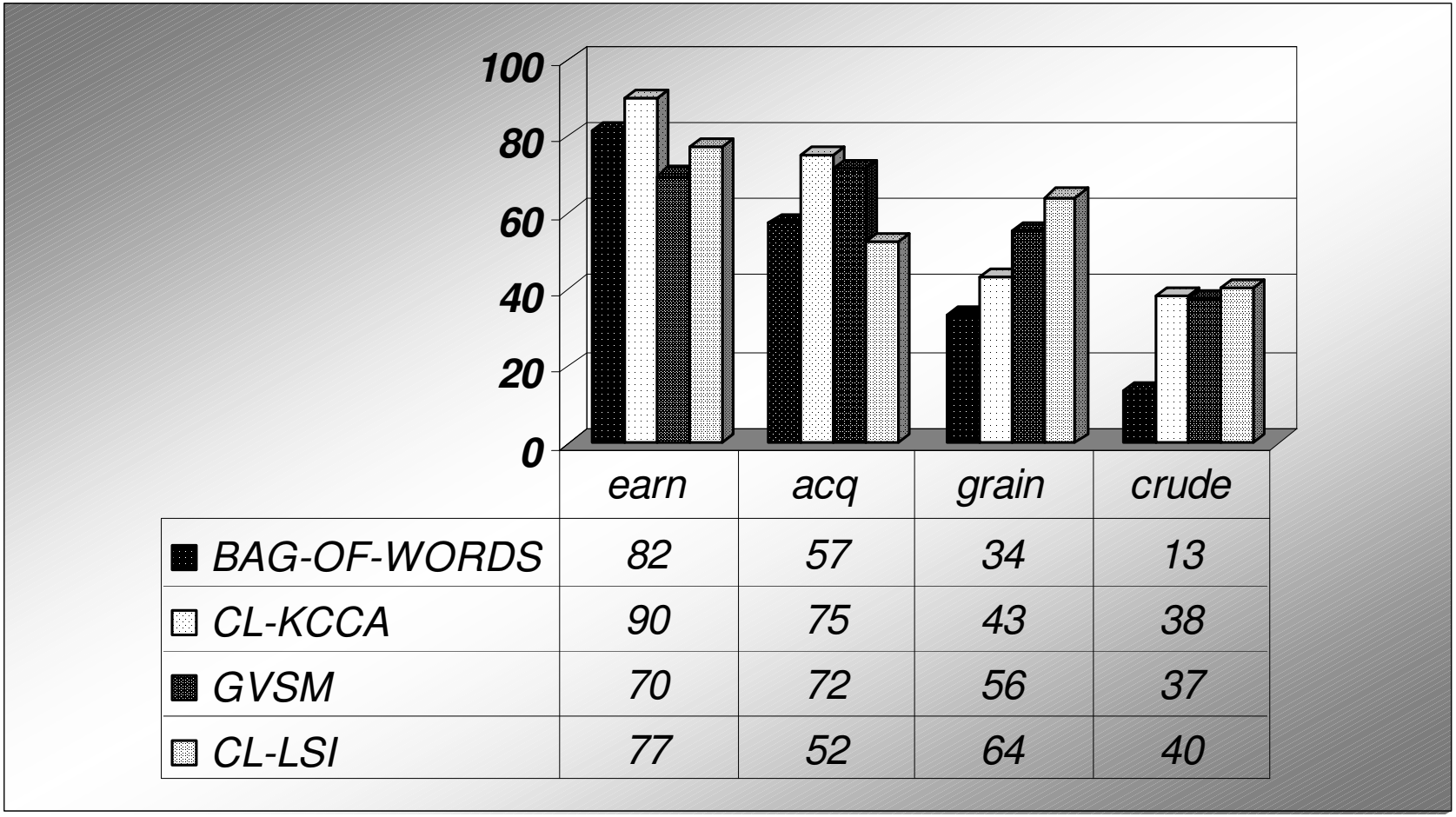
(1) LSI/KCCA trained on paired documents,

(2) Whole Reuters corpus was projected into the LSI/KCCA semantic space,

(3) Linear SVM classifier was trained in the LSI/KCCA semantic space on a subset of documents and tested on a separate test set.

(4) Same procedure used for Generalised Vector Space Model (GVSM) – representation of a document is vector of inner products with the training set of appropriate language.

SVM classification with Reuters 21578 with 5% training

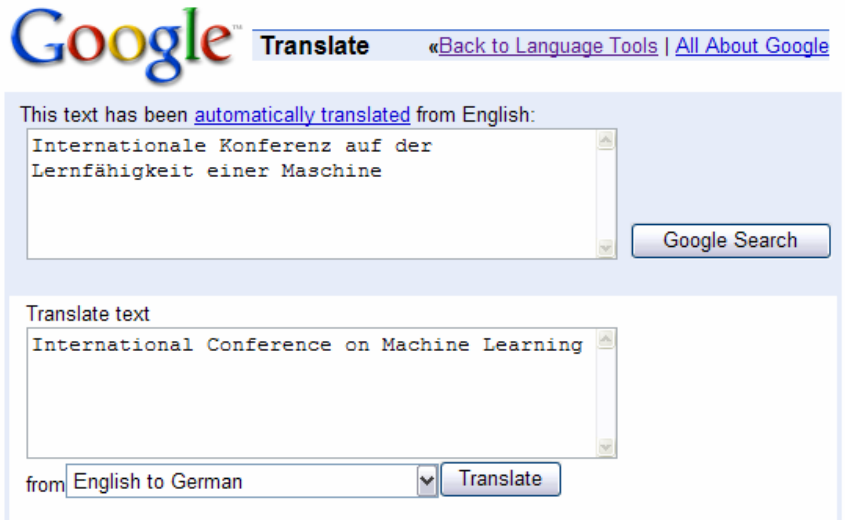


	earn	acq	grain	crude
BAG-OF-WORDS	82	57	34	13
CL-KCCA	90	75	43	38
GVSM	70	72	56	37
CL-LSI	77	52	64	40

Paired training set and machine translation

KCCA needs paired dataset for training. When there is no paired dataset available we have two options:

- We use human made dataset from some other domain.
  - This could be unreliable because of a big semantic and vocabulary gap.
- We use machine translation tools to generate paired dataset.
  - In our experiments we used Google Language Tools for translating documents.



Artificial paired corpora

We compared two paired corpora:

- *Hansard* corpus: aligned pairs of text chunks from the official records of the 36th Canadian Parliament Proceedings.

  [Germann, 2001]

- *Artificial corpus*: half of the English and half of the French translations from *Hansard* corpus were replaced by machine translation.

Results

For 65% of queries the correct document appeared on the first place.
For 95% of queries the correct document appeared among first 10 results.

	En-En	En-Fr	Fr-En	Fr-Fr
Hansard	87 / 99	66 / 96	65 / 95	84 / 99
Artificial	86 / 99	58 / 91	59 / 90	83 / 99

There is no difference when query and document are in the same language

When query and document are from different languages, there is around 5-10% drop in retrieval accuracy

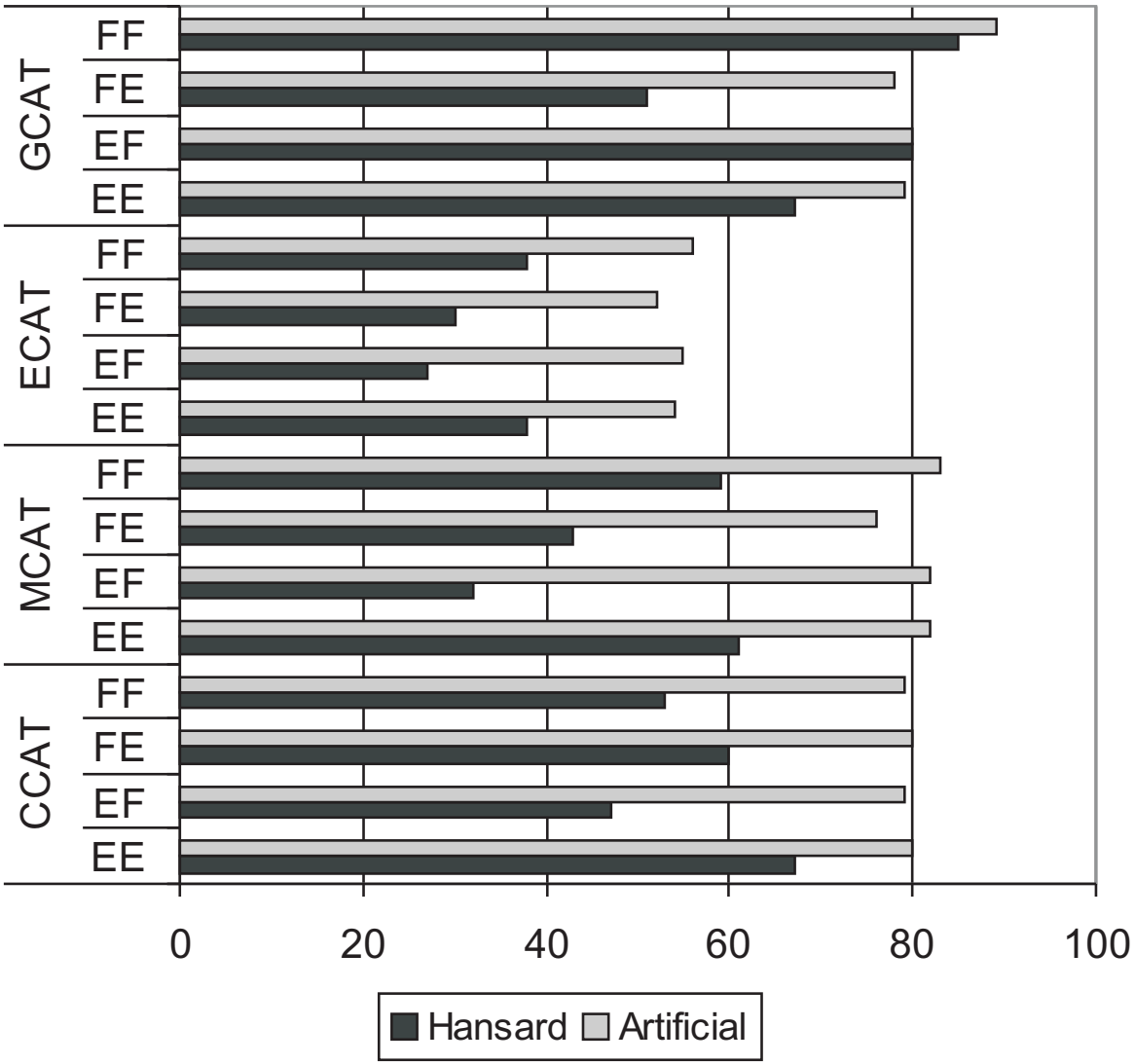
Classification with artificial PC

Reuters multilingual corpora (English and French) was used as a dataset. [Reuters, 2004]

- First paired train set, *Hansard*, was taken from previous experiment; different domain than news articles.
- Second paired train set was generated from the Reuters dataset using machine translation (Google).

- Results are averaged over 5 random splits.

Results



#KCCA dimensions: 800

FE … French training set, English testing set.

Artificial paired training set generates significantly better semantic space than train set taken from a different domain!

Applying to different data types

- Data
  - Combined image and associated text obtained from the web
  - Three categories: sport, aviation and paintball
  - 400 examples from each category (1200 overall)
  - Features extracted: HSV, Texture , Bag of words
- Tasks
  1. Classification of web pages into the 3 categories
  2. Text query -> image retrieval

Classification error of baseline method

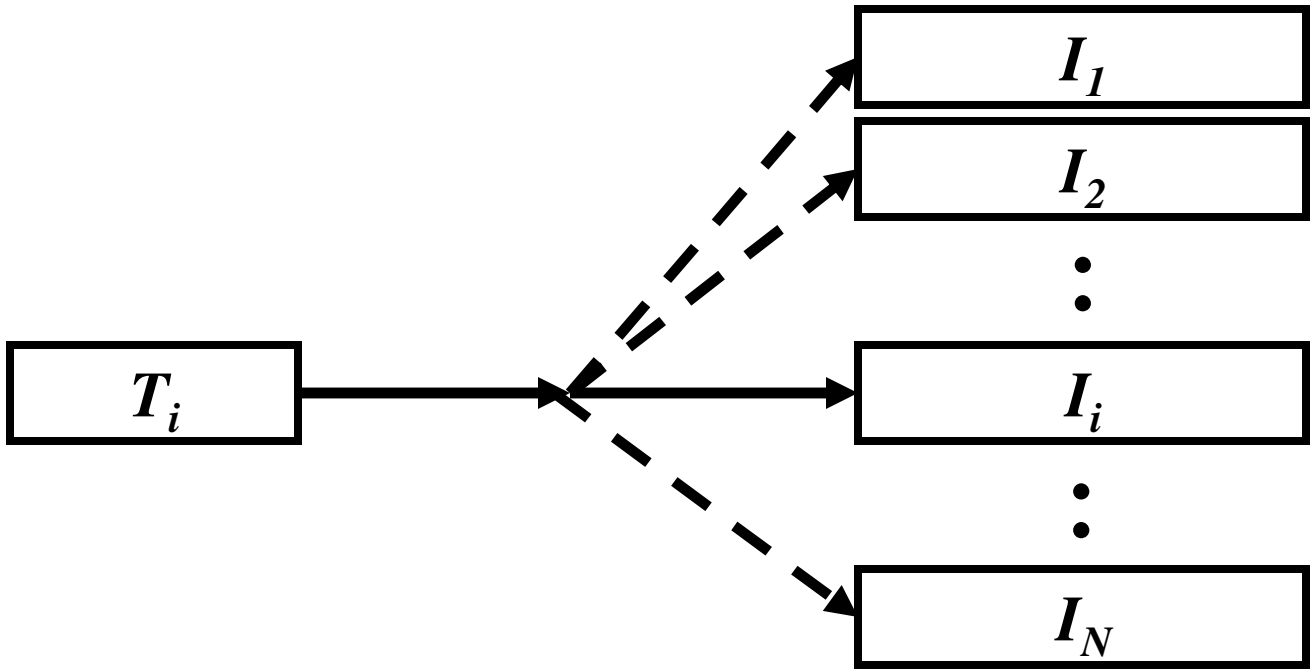
- Previous error rates obtained using probabilistic ICA classification done for single feature groups

Colour	Texture	Colour + Texture	Text	All
22.9%	18.3%	13.6%	9.0%	3.0%

Classification rates using KCCA

K	Error rate
Plain SVM	2.13%±0.23%
KCCA-SVM (150)	1.36%±0.15%
KCCA-SVM (200)	1.21%±0.27%

Query Test for Image Retrieval



% Success of partner image in Top n rated images

Success rate	Top 10	Top 30
GVSM	1.27%	5%
KCCA	17.34%	30.34%

Example

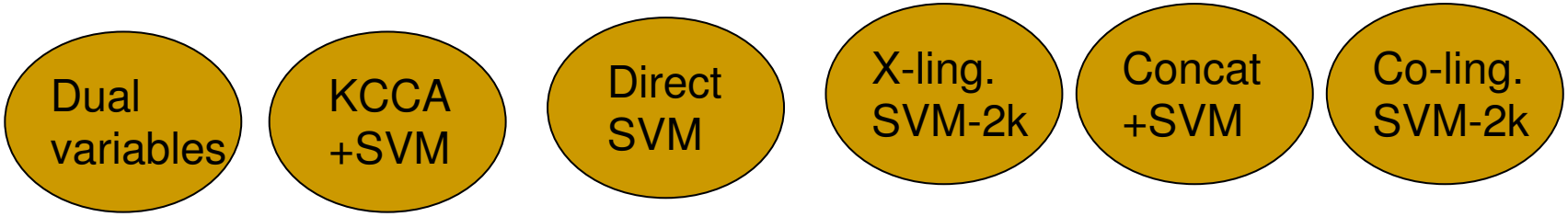
Height: 7-0 weight: 225 lbs position: center born: august 5, 1962, Kingston, Jamaica college: Georgetown

Actual Match

Classification with multi-views

- If we use KCCA to generate a semantic feature space and then learn with an SVM, can envisage combining the two steps into a single 'SVM-2k'
- learns two SVMs one on each representation, but constrains their outputs to be similar across the training (and any unlabelled data).
- Can give classification for single mode test data – applied to patent classification for Japanese patents
- Again theoretical analysis predicts good generalisation if the training SVMs have a good match, while the two representations have a small overlap

Results



	pSVM	kcca_SVM	SVM	SVM_2k_j	Concat	SVM_2k
1	59.4±3.9	60.3±2.8	66.6±2.8	66.1± 2.6	67.5±2.3	67.5±2.1
2	71.1±4.5	68.4±4.4	73.0±4.0	74.8±4.7	73.9±4.0	75.1±4.1
3	16.7±1.2	13.1±1.0	18.8±1.6	20.8±1.9	21.5±1.9	22.5±1.7
7	74.9±1.8	76.0±1.2	76.7±1.3	77.5±1.4	79.0±1.2	80.7±1.5
12	75.0±0.8	73.6±0.8	76.8±1.0	77.6±0.7	76.8±0.6	78.4±0.6
14	76.0±1.6	71.5±1.5	80.9±1.3	82.2±1.3	81.4±1.4	82.7±1.3

Conclusions

- SVMs are well-founded in frequentist statistics and lead to convex quadratic programs that can be solved with simple algorithms
- Allow use of high dimensional feature spaces but control generalisation using data dependent structural risk minimisation
- Kernels enable efficient implementation through dual representations
- Kernel design can be extended to non-vectorial data and complex models

Conclusions

- Same approach can be used for other learning tasks: eg regression, ranking, etc.
- Subspace methods can often be implemented in kernel defined feature spaces using dual representations
- Overall gives a generic plug and play framework for analysing data, combining different data types, models, tasks, and preprocessing

SMART research topics

- Extensions of kernel design for specific data types often inspired by probabilistic models, eg
  - finite state automata based models of language
  - Tree based kernels capturing possible parsings,
  - slot phrases capturing context, etc.
- Extensions to learning data with output structure that can be used to inform the learning, eg
  - maximum margin Markov
  - Maximum margin robot
  - Density learning
- Choosing kernels as part of the learning process
- Critical role of scaling to substantial datasets