An Introduction to Kernel-Based Learning Algorithms [1]

Klaus-Robert Müller, Sebastian Mika, Gunnar Rätcsch, Koji Tsuda, and Bernhard Schölkopf

Saurabh V. Pendse

North Carolina State University

April 8, 2013

[1] IEEE Transactions on Neural Networks, 2001, Cited By 2367

Introduction

- Review paper on kernel-based learning methods.
- Basic concepts of learning theory.
  1. Support Vector Machines (SVM)
  2. Kernel Fischer Discriminant analysis (KFD)
  3. Kernel Principal Component Analysis (k-PCA)
- Emphasis on the above and their connections to boosting.
- Supervised and unsupervised learning.
- Model selection, parameter tuning.
- Recent and interesting applications.

What is classification? I

- The task of classification is to find a rule, which, based on external observations, assigns an object to one of several classes.
- Formally, the task is to estimate a function $f : \mathbb{R}^N \longrightarrow \{-1, +1\}$, using i.i.d input-output training data pairs according to an unknown probability distribution $P(\mathbf{x}, y)$

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots (\mathbf{x_n}, y_n) \in R^N \times Y, \quad Y = \{-1, 1\}$$

such that f will correctly classify unseen examples $(\mathbf{x}, y)$.

The test examples are assumed to be generated from the same probability distribution $P(\mathbf{x}, y)$ as the training data.

What is classification? II

Minimize the expected risk :

$$R[f] = \int l(f(\mathbf{x}), y) dP(\mathbf{x}, y) \tag{1}$$

where l denotes a suitably chosen loss function (e.g. squared loss).

- $R[f]$ cannot be minimized directly since $P(\mathbf{x}, y)$ is unknown.
- Minimize the empirical risk based on the training data as follows :

$$R_{emp}[f] = \frac{1}{n} \sum_{i=1}^{n} l(f(\mathbf{x_i}, y_i)) \tag{2}$$

- As $n \longrightarrow \infty$, R_{emp} converges toward R.

What is classification? III

- Overfitting may occur for small sample sizes.

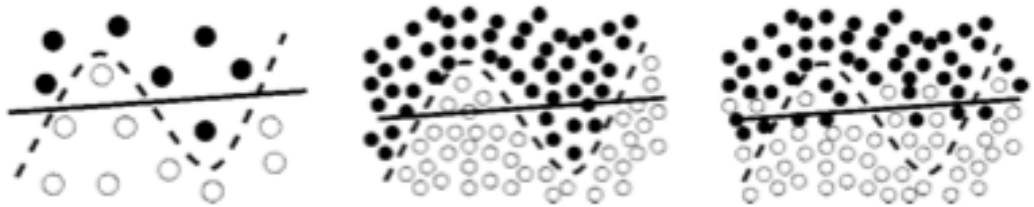


Figure : The overfitting dilemma

- Goal is to minimize the generalization error, not simply the training error.
- Restrict the complexity of the function class F (from which f is chosen).
- A "simple" (e.g. linear) function that explains most of the data is preferable to a complex one.
- Introduce a *regularization term* to limit the complexity of the function class F.

Vapnik-Chervonenkis (VC) Dimension I

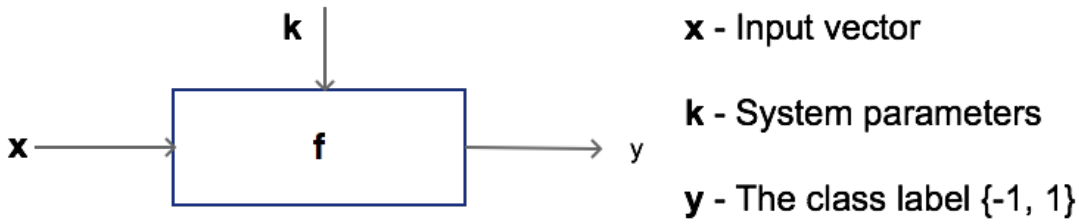


Figure : A classification machine

Given r data points there are 2^r possible training sets in the case of binary classification.

Shattering

A machine f can shatter a set of points $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_r}$ if and only if for every possible training set of the form $(\mathbf{x_1}, y_1), \ldots (\mathbf{x_r}, y_r)$, there exists some system configuration $\mathbf{k}$ that gets a zero training error.

Vapnik-Chervonenkis (VC) Dimension II

Shattering guarantees the number of points that can be reliably separated for a given function f.
For simplicity, consider a 2-dimensional input space :

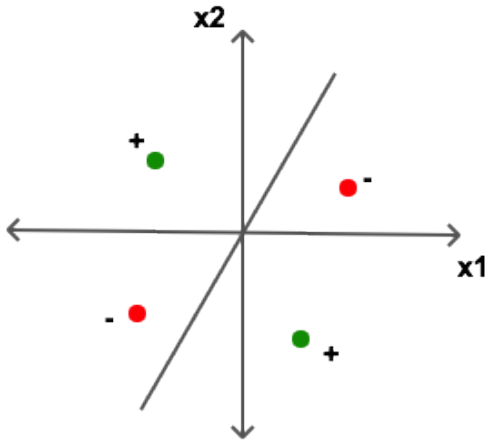


Figure : Linear f

- $f = \text{sgn}(\mathbf{w}^T \cdot \mathbf{x} + b)$
- Can shatter at the most 3 points.

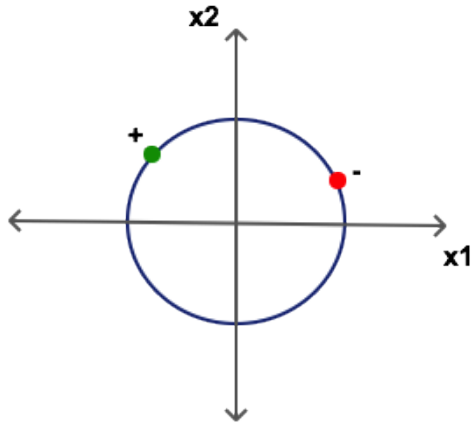


Figure : Quadratic f

- $f = \text{sgn}(\mathbf{x}^T \cdot \mathbf{x} - b)$
- Can shatter at most 1 point.

In general, the VC dimension of a linear function f, given an N dimensional input is $N + 1$.

Vapnik-Chervonenkis (VC) Dimension III

- A specific way of controlling the complexity of a function class.
- VC dimension h of the function class F that the estimate f is chosen from.

Theorem (Vapnik95, Vapnik97)

Let h denote the VC dimension of the function class F and let R_{emp} defined in 2 using the 0/1-loss. For all $\delta > 0$ and $f \in F$,

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h\left(\log\frac{2n}{h} + 1\right) - \log(\frac{\delta}{4})}{n}} \tag{3}$$

holds with probability of at least $1 - \delta$ for $n > h$.

Vapnik-Chervonenkis (VC) Dimension IV

VC Dimension in practice

- The bound on $R[f]$ is neither easily computable nor very helpful.
- Most often the upper bound on the expected test error might be trivial, or the VC dimension of the function class may be unknown or infinite.
- Such bounds offer helpful theoretical insights into the nature of learning problems.

Vapnik-Chervonenkis (VC) Dimension V

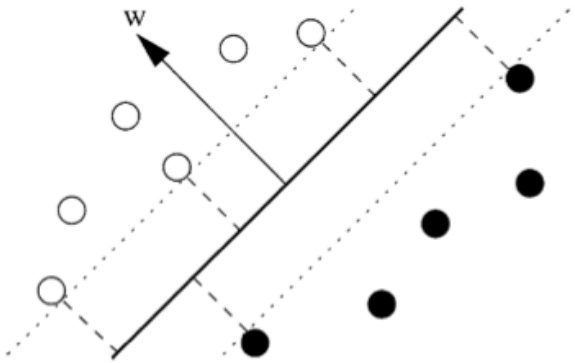


Figure : Linear classifier and margins : $f(\mathbf{x}) = \text{sgn}((\mathbf{w} \cdot \mathbf{x}) + b)$.

- The margin of the linear classifier is the minimal distance of any training point to the hyperplane. ($\frac{2}{||\mathbf{w}||}$ in this case).
- $h \leq ||\mathbf{w}||_2^2 R^2 + 1$, where R is the radius of the smallest sphere around the data.

Thus, if we bound the margin of a function class F from below, say by $\frac{2}{||\mathbf{w}||}$, we can control its VC dimension and hence its classifying power.

Kernel Functions I

- Algorithms in non-linear feature spaces :

$$\begin{aligned} \Phi : \mathbb{R}^N &\longrightarrow \mathcal{F} \\ \mathbf{x} &\longrightarrow \Phi(\mathbf{x}) \end{aligned}$$

  Data : $(\Phi(\mathbf{x_1}), y_1), \ldots, (\Phi(\mathbf{x_n}), y_n)$.
- Given this mapped representation, a simple classification or regression in $\mathcal{F}$ is to be found.

Curse of Dimensionality

Estimation problem increases drastically with the dimension N of the space. (exponentially many patterns to sample the space properly.)

Simple Function Classes

Simple class of decision rules. All variability and richness that one needs to have a powerful function class can be introduced into the mapping Φ.

Kernel Functions II

If $k : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{R}$ is a continuous and positive semi-definite kernel, then there exists a space $\mathcal{F}$ and a mapping $\Phi : \mathbb{R}^N \longrightarrow \mathcal{F}$ such that $k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$.

Takeaway

Every (linear) algorithm that only uses scalar products can implicitly be executed in $\mathcal{F}$ by using kernels i.e. one can very elegantly construct a non-linear version of a linear algorithm.

Gaussian RBF	$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{c}\right)$
Polynomial	$((\mathbf{x} \cdot \mathbf{y}) + \theta)^d$
Sigmoidal	$\tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \theta)$
inv. multiquadric	$\frac{1}{\sqrt{\|\mathbf{x} - \mathbf{y}\|^2 + c^2}}$

Support Vector Machines I

- VC dimension of linear system can be upper bounded in terms of the margin.
- Perfect classification for hyperplane classifiers :

$$y_i((\mathbf{w} \cdot \mathbf{x_i}) + b) \geq 1 \qquad i = 1, 2, \ldots, n \tag{4}$$

- Linear classifiers are not rich enough in practice. Hence we use the kernel trick.

$$y_i((\mathbf{w} \cdot \Phi(\mathbf{x_i})) + b) \geq 1 \qquad i = 1, 2, \ldots, n \tag{5}$$

Support Vector Machines II

- Minimize expected risk : Keep empirical risk zero and minimize the complexity term.
- Complexity term is monotonically increasing function of the VC dimension, h, which in turn is bounded according to $h \leq ||\mathbf{w}||^2 R^2 + 1$.
- Thus, the optimization problem can be formulated as :

$$\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||^2 \tag{6}$$

- $\mathbf{w}$ lies in the feature space $\mathcal{F}$. Hence we cannot directly solve 6.

Support Vector Machines III

- Introduce Lagrangian multipliers for each of the constraints in 5.

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i((y_i((\mathbf{w} \cdot \Phi(\mathbf{x_i})) + b) - 1)) \quad (7)$$

- Minimize 7 w.r.t. $\mathbf{w}, b$ and maximize it w.r.t. α_i.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial b} &= 0 \text{ and } \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \\ \sum_{i=1}^{n} \alpha_i y_i &= 0 \text{ and } \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x_i}) \end{aligned} \quad (8)$$

Support Vector Machines IV

- Obtain the coefficients $\alpha_i, i = 1, \ldots, n$ by solving the dual quadratic optimization problem :

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{(\Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j}))}_{k(\mathbf{x_i}, \mathbf{x_j})} \\ \text{subject to} \quad & \alpha_i \geq 0, i = 1, \ldots, n \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

-

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{n} y_i \alpha_i \underbrace{(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x_i}))}_{k(\mathbf{x} \cdot \mathbf{x_i})} + b \right) \tag{9}$$

Support Vector Machines V

- In case of noisy data, slack variables are introduced to avoid overfitting.

$$y_i((\mathbf{w} \cdot \Phi(\mathbf{x_i})) + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0, i = 1, \ldots, n$$

$$\min_{\mathbf{w}, b, \xi} \qquad \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i$$

- The dual optimization problem is given by (notice the constraints on α_i):

$$\max_{\alpha} \qquad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j})$$

$$\text{subject to} \qquad 0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

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

Support Vector Machines VI

- **Sparsity** - The SVM solution is sparse in α i.e. most patterns are outside the margin area and the optimal α_i's are zero. Without this property, SVMs would be hardly practical for large datasets.
- **v-SVMs** - This modification to the SVM algorithm introduces an upper bound on the number of margin errors, and a lower bound on the number of Support Vectors.
- **Computing the threshold** - The threshold b can be computed by exploiting the fact that for all the support vectors $\mathbf{x_i}$, with $0 < \alpha_i < C$, the slack variable ξ_i is zero. (KKT conditions).

Support Vector Machines VII

- **A Geometrical explanation** Consider a normalized weight vector $||\mathbf{w}|| = 1$ and $b = 0$. The version space $\mathcal{V}$ is :

$$\mathcal{V} = \{\mathbf{w} | y_i f(\mathbf{x_i}) > 0; i = 1, \ldots, n, ||\mathbf{w}||_2 = 1\}$$

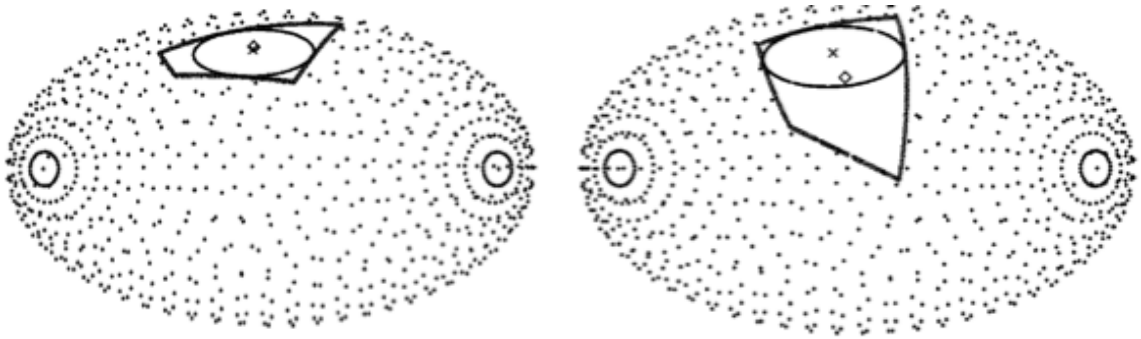
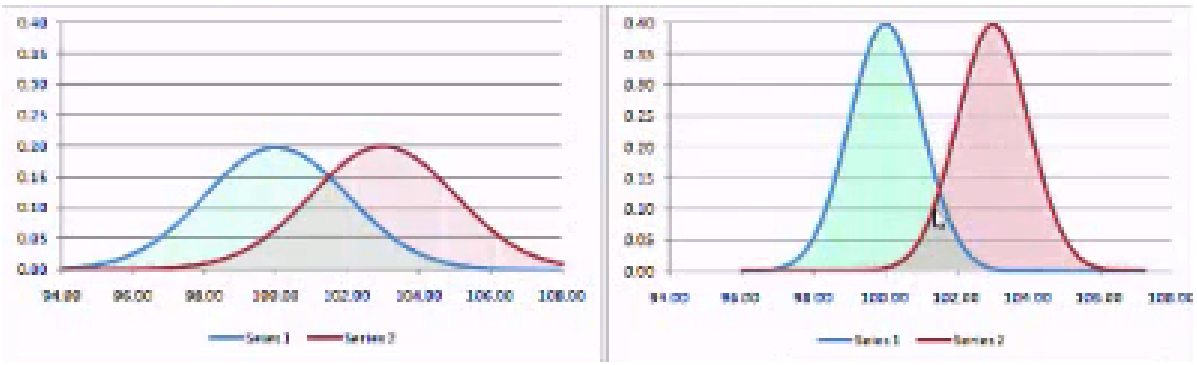


Figure : SVM Solution vs Bayes Point and SVM performance

- SVM Solution - Tchebycheff-center of the version space.
- Theoretical optimal point - Bayes point (closely approximated by the center of mass).

Kernel Fischer Discriminant I

- Solve the problem of Fisher's linear discriminant in a kernel feature space $\mathcal{F}$.
- Fisher's linear discriminant aims at finding linear projections such that the classes are well separated.



- This can be achieved by maximizing the Rayleigh coefficient :

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{10}$$

S_B is the between-class variance, whereas S_W is the within-class variance.

Kernel Fischer Discriminant II

$$S_B = (\mathbf{m_2} - \mathbf{m_1})(\mathbf{m_2} - \mathbf{m_1})^T \quad (11)$$

$$S_W = \sum_{k=1,2} \sum_{i \in \mathcal{I}_k} (\mathbf{x_i} - \mathbf{m_k})(\mathbf{x_i} - \mathbf{m_k})^T \quad (12)$$

Here $\mathbf{m_k}$ and $\mathcal{I}_k$ denote the sample mean and index set for the class k.

- In the kernel feature space $\mathcal{F}$, $\mathbf{w}$ can be expressed as (similar to SVMs) :

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x_i}) \quad (13)$$

Kernel Fischer Discriminant III

- With this, the KFD optimization problem can be stated as:

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T M \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T N \boldsymbol{\alpha}} \tag{14}$$

- Given the solution for $\boldsymbol{\alpha}$, the projection of a new data point is given by :

$$y(\mathbf{x}) = (\mathbf{w} \cdot \Phi(x)) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x_i}, \mathbf{x}) \tag{15}$$

- The class label of a new input can then be determined using :

$$f(\mathbf{x}) = arg\min_{j} \; D(y(\mathbf{x}), \bar{\mathbf{y}}_\mathbf{j}) \tag{16}$$

where $\bar{\mathbf{y}}_\mathbf{j}$ is the projected mean for class j and $D(.,.)$ is the distance function.

Boosting I

- Meta-algorithm for reducing bias in supervised learning.
- Can a set of weak learners create a strong learner?
- Most boosting algorithms : Iteratively learn weak classifiers w.r.t a distribution and add them to a final strong classifier.

Arc-GV

$$\max_{w \in \mathcal{F}, \rho \in R_+} \quad \rho$$

$$\text{subject to} \quad y_i \sum_{j=1}^{J} w_j h_j(\mathbf{x_i}) \geq \rho \qquad i = 1, \ldots, n$$

$$||\mathbf{w}||_1 = 1$$

ρ is the margin, and J is the number of weak learners (iterations).

Boosting II

SVM minimization

$$\max_{w \in \mathcal{F}, \rho \in R_+} \quad \rho$$

$$\text{subject to} \quad y_i \sum_{j=1}^{N} w_j P_j(\Phi(\mathbf{x_i})) \geq \rho \qquad i = 1, \ldots, n$$

$$||\mathbf{w}||_2 = 1$$

$N = dim(\mathcal{F})$, P_j is the operator projecting onto the jth coordinate in $\mathcal{F}$.

- Boosting is a SV approach in a high-dimensional feature space spanned by the base hypothesis of a function set H.
- SVMs and KFD are a boosting approach in a high-dimensional space.

Unsupervised Learning

- Only the data $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n} \in \mathbb{R}^N$ is given.
- Labels are missing.
- More relevant to many real-world problems, since it is difficult to obtain labelled data.
- Clustering, density estimation and data description.
- Kernel trick can be applied here as well : if the base algorithm can be written in terms of scalar products.

Kernel PCA I

Linear PCA

Basic idea : For N-dimensional data, compute a set of orthogonal directions - capturing most of the variance in the data.

- The first k projections allow the reconstruction with minimal quadratic error.
- Useful for dimensionality reduction, pattern recognition and data compression.
- Linear version cannot be used to extract nonlinear structures in the data.

Kernel PCA II

- Map the data $\mathbf{x_1}, \ldots, \mathbf{x_n} \in \mathbb{R}^N$ into a feature space $\mathcal{F}$.
- Compute the covariance matrix

$$C_{ij} = \Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})$$

- Solve the Eigenvalue problem for C. i.e. Find $\lambda > 0, \mathbf{V} \neq 0$,

$$\lambda \mathbf{V} = C\mathbf{V} = \frac{1}{n} \sum_{j=1}^{n} (\Phi(\mathbf{x_j}) \cdot \mathbf{V}) \Phi(\mathbf{x_j}) \tag{17}$$

Kernel PCA III

- Also, all eigenvectors with non-zero eigenvalue must be in the span of the mapped data.

$$\begin{aligned} \mathbf{V} &= \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x_i}) \\ \therefore \lambda(\Phi(\mathbf{x_k}) \cdot \mathbf{V}) &= (\Phi(\mathbf{x_k}) \cdot C\mathbf{V}) \quad k = 1, \ldots, n \\ \therefore \lambda\boldsymbol{\alpha} &= K\boldsymbol{\alpha} \end{aligned} \tag{18}$$

  Here $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$ and 18 refers to the eigenvalue problem for $\boldsymbol{\alpha}$.

- The k^{th} nonlinear principal component of $\mathbf{x}$ is the projection of $\Phi(\mathbf{x})$ onto V^k is given by :

$$(\mathbf{V}^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{n} \alpha_i^k k(\mathbf{x_i}, \mathbf{x}) \tag{19}$$

Single-Class Classification I

Density estimation

Density estimation : Assuming unlabeled observations $\mathbf{x_1}, \ldots, \mathbf{x_n}$ were generated i.i.d. according to some unknown distribution $P(\mathbf{x})$, estimate the density.

- Density might not even exist.
- Exact estimation is a hard task.
- Often enough to estimate the support (i.e. quantiles of a multivariate distribution) of a data distribution instead of the full density.
- Single-class SVMs are often used for this purpose.

Single-Class Classification II

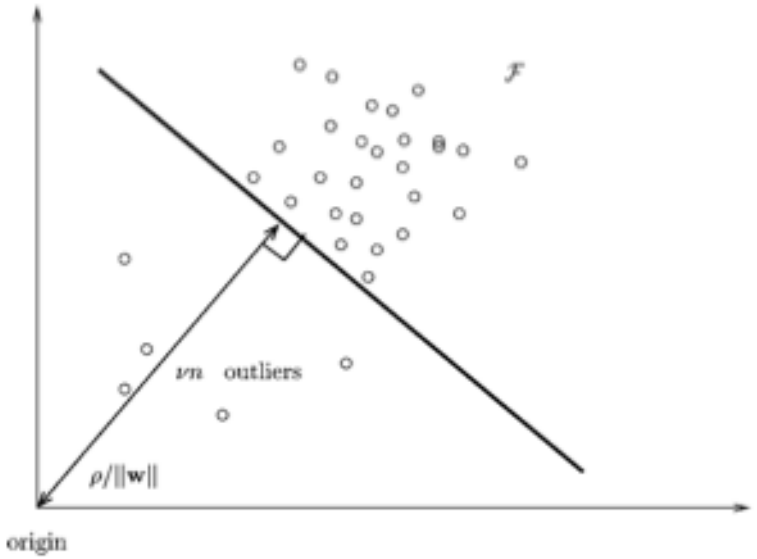


Figure : The single-class idea (I) : A hyperplane is constructed in $\mathcal{F}$ that maximizes the distance to the origin while allowing for νn outliers.

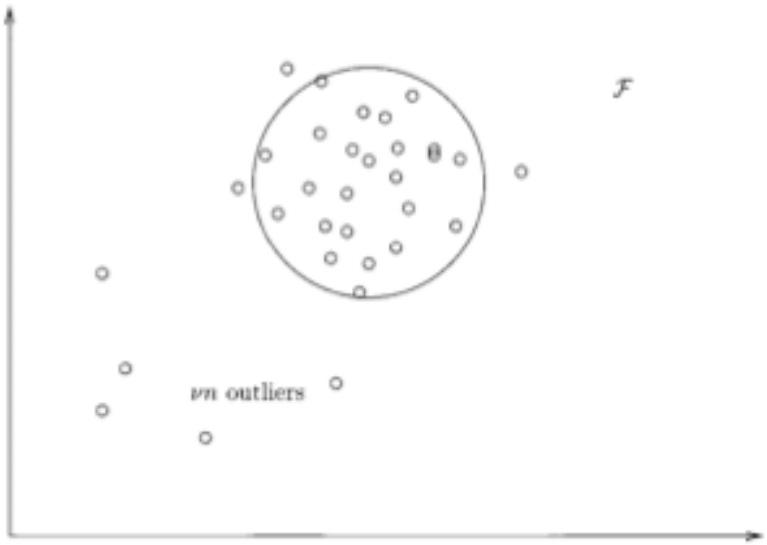


Figure : The single-class idea (II) : Construction of the smallest soft sphere in $\mathcal{F}$ that contains the data.

Model Selection

- In kernel methods, the choice of kernels is very crucial.
- Choosing the wrong kernel for a problem leads to poor performance.
- Model selection provides principled methods to select a proper kernel.
- Candidates for the optimal kernels are prepared using some heuristic rules.
- The one which minimizes a given criterion or measure is chosen.

Methods for Model Selection I

Bayesian evidence framework

The training of a SVM is interpreted as Bayesian inference, and the model selection is done by maximizing marginal likelihood. *Relevance Vector Machines (RVMs) - probabilistic classification.*

Probably Approximately Correct (PAC)

The generalization error is upper bounded using a capacity measure depending on the weights and the model, and these are optimized to minimize the bound. *Take into account the distribution of the input data by considering the eigenvalue distribution of the Gram matrix of the data. i.e. perform Kernel PCA + SVMs*

Methods for Model Selection II

Cross validation

- The training samples are divided into k subsets, each of which have the same number of samples.
- The classifier is trained k times. In the i^{th} iteration, the classifier is trained on all subsets except the i^{th} one. The classification error is computed for the i^{th} subset.
- The average of these k errors is a rather good estimate of the generalization error.
- Leave-one-out cross validation.

Other techniques : AIC, NIC (require large amounts of samples), span bound (invariant support vectors).

Applications : Supervised Learning - OCR

Standardized Optical Character Recognition benchmarks such as USPS and MNIST.

linear PCA & linear SVM (Schölkopf et. al. [11])	8.7%
k-Nearest Neighbor	5.7%
LeNet1 (LeCun et. al. [132], [133], [134])	4.2%
Regularized RBF Networks (Rätsch [128])	4.1%
Kernel-PCA & linear SVM (Schölkopf et. al. [11])	4.0%
SVM (Schölkopf et. al. [120])	4.0%
Virtual SVM (Schölkopf [4])	3.0%
Invariant SVM (Schölkopf et. al. [131])	3.0%
Boosting (Drucker et. al. [137])	2.6%
Tangent Distance (Simard et. al. [135], [136])	2.5%
Human error rate	2.5%

Figure : Classification Error in % for off-line handwritten character recognition on the USPS dataset with 7291 and 2007 training and test patterns respectively.

Applications : Supervised Learning - Analyzing DNA Data I

- Coding sequences (CDS) encode proteins.
- Necessary to recognize the translation initiation sites (TIS) from which coding starts to determine which parts of a sequence will be translated and which not.
- A potential start codon is typically a ATG triplet.
- A classification task to determine whether or not a binary coded (fixed length) sequence window around the ATG indicates a true TIS.
- Knowledge priors prove to be very useful e.g. specialized kernel functions for SVMs.

Applications : Supervised Learning - Analyzing DNA Data II

algorithm	parameter setting	overall error
neural network		15.4%
Salzberg method		13.8%
SVM, simple polynomial	d=1	13.2%
SVM, locality-improved kernel	d_1=4,l=4	11.9%
SVM, codon-improved kernel	d_1=2,l=3	12.2%
SVM, Salzberg kernel	d_1=3,l=1	11.4%

Figure : Comparison of classification errors (measured on the test sets) achieved with different learning algorithms. (11000 training samples, 3000 test samples)

The engineered kernel functions clearly outperform the neural network or the Salzberg method.

Benchmarks I

UCI, DELVE and STATLOG standardized benchmark repositories. However :

- Unclear model selection.
- Size of training and test samples not always stated.
- No information about reliability of results (error bars or confidence intervals).
- Data might need preprocessing.
- These are often multi-class problems.

IDA repository - A very clean repository; contains 13 artificial and real-world datasets collected from the above repositories.

Benchmarks II

	SVM	KFD	RBF	AB	AB_R
Banana	11.5±0.07	**10.8±0.05**	**10.8±0.06**	12.3±0.07	*10.9±0.04*
B.Cancer	*26.0±0.47*	**25.8±0.46**	27.6±0.47	30.4±0.47	26.5±0.45
Diabetes	*23.5±0.17*	**23.2±0.16**	24.3±0.19	26.5±0.23	23.8±0.18
German	**23.6±0.21**	*23.7±0.22*	24.7±0.24	27.5±0.25	24.3±0.21
Heart	**16.0±0.33**	*16.1±0.34*	17.6±0.33	20.3±0.34	16.5±0.35
Image	*3.0±0.06*	3.3±0.06	3.3±0.06	**2.7±0.07**	**2.7±0.06**
Ringnorm	1.7±0.01	**1.5±0.01**	1.7±0.02	1.9±0.03	*1.6±0.01*
F.Sonar	**32.4±0.18**	*33.2±0.17*	34.4±0.20	35.7±0.18	34.2±0.22
Splice	10.9±0.07	10.5±0.06	*10.0±0.10*	10.1±0.05	**9.5±0.07**
Thyroid	4.8±0.22	**4.2±0.21**	4.5±0.21	*4.4±0.22*	4.6±0.22
Titanic	**22.4±0.10**	23.2±0.20	23.3±0.13	*22.6±0.12*	*22.6±0.12*
Twonorm	3.0±0.02	**2.6±0.02**	2.9±0.03	3.0±0.03	*2.7±0.02*
Waveform	*9.9±0.04*	*9.9±0.04*	10.7±0.11	10.8±0.06	**9.8±0.08**

Figure : Comparison between SVM, KFD, RBF, AdaBoost and Regularized Adaboost on 13 benchmark datasets of the IDA repository.

Applications : Unsupervised Learning - Kernel PCA I

- **Denoising :** Given a noisy $\mathbf{x}$, map it into $\Phi(\mathbf{x})$, discard the higher components to obtain $P_k\Phi(\mathbf{x})$, and then compute a preimage $\mathbf{z}$. The hope is that the main structure of the dataset is captured in the first k directions, and the remaining components merely pick up the noise.
- **Compression :** Given the eigenvectors α^i and a small number of features β_i of $\Phi(\mathbf{x})$, but not $\mathbf{x}$, compute a preimage that is the approximate reconstruction of $\mathbf{x}$.
- **Interpretation :** Visualize a nonlinear feature extractor $\mathbf{V}^i$ by computing a preimage. The preimage might not exist, and if it exists, might not be unique.

Applications : Unsupervised Learning - Kernel PCA II

Saurabh V. Pendse

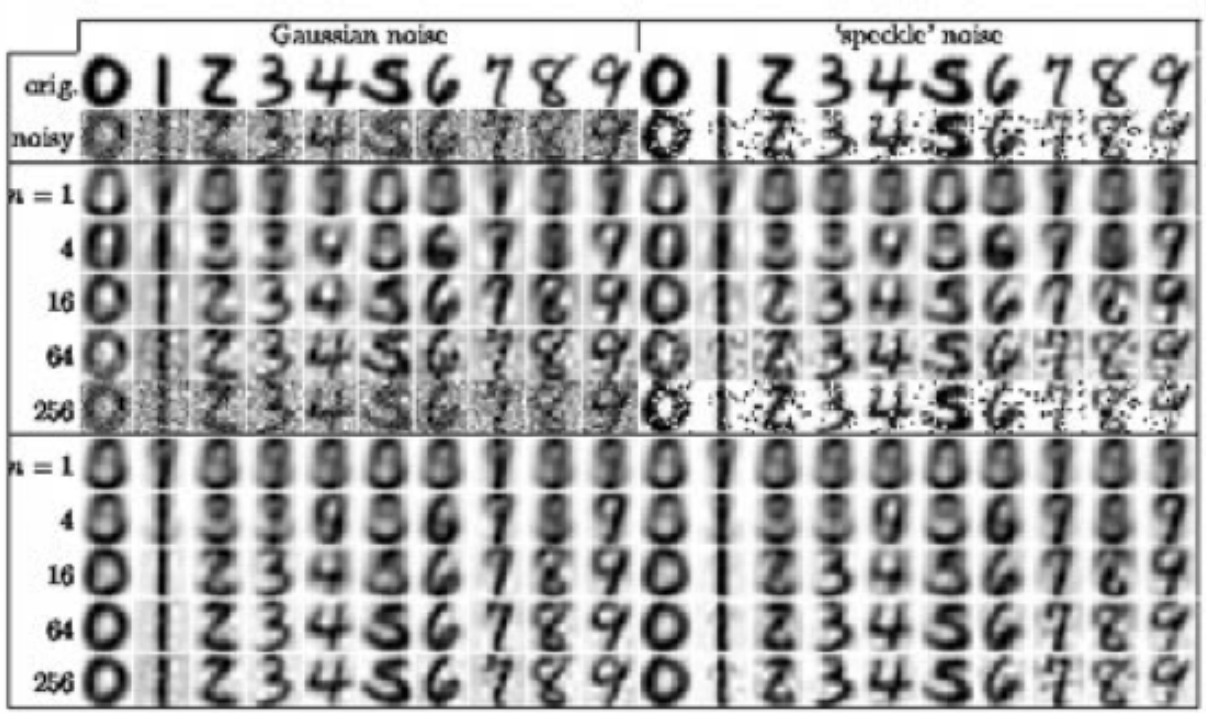


Figure : Denoising of the USPS dataset. Orig. dataset, followed by noisy version. The next five rows : reconstruction for linear PCA using different k values. The last five rows : reconstruction using the approximate preimage approach with kernel PCA.

Summary & Conclusion

- Goal was to give a simple introduction into the field of kernel-based learning methods.
- Proposed a conceptual framework for KFD, boosting and SVM w.r.t handling of high dimensionality of kernel feature spaces.
- Reviewed kernel PCA for finding projections that give useful nonlinear descriptors of the data.
- A single class SVM algorithm for density estimation.
- Selected real-world applications showed that kernel-based learning methods exhibit a good performance on a variety of problems with different characteristics.

Transform linear scalar product based algorithms to non-linear algorithms to obtain further powerful kernel-based learning machines.

Thank you.