

Support Vector Machines

A Tutorial Introduction

James M. Hogan
Queensland University of Technology
Faculty of Information Technology

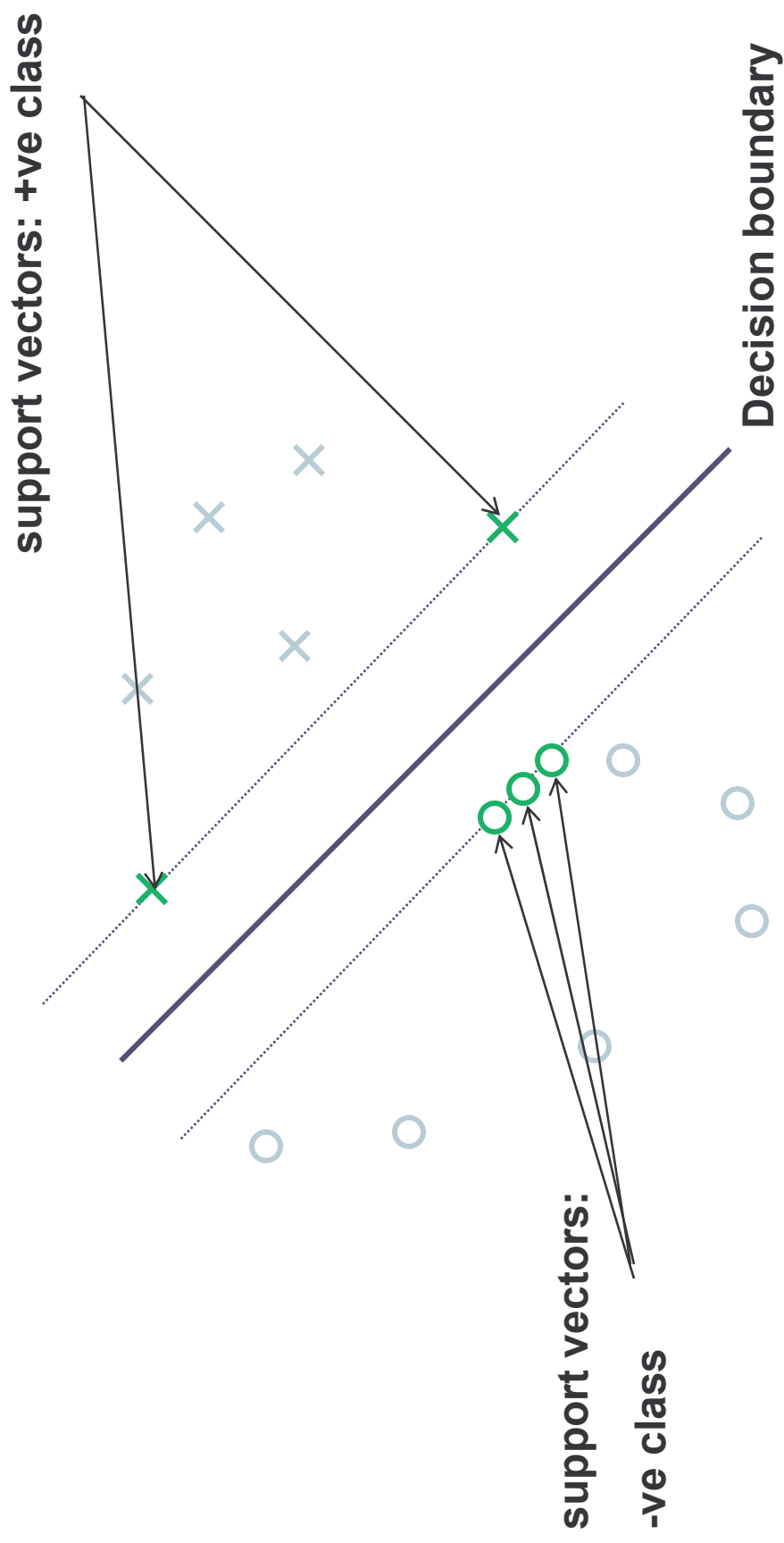
Acknowledgements

- ❑ Slides stolen from
 - ❑ Earlier talks of mine
 - ❑ Lecture notes due to Frederic Maire
- ❑ Sources include
 - ❑ Tutorial due to Chris Burges
 - ❑ Tutorial due to Scholkopf
 - ❑ Papers due to Shawe-Taylor et. al.

Outline

- The New Religion
- Statistical Learning Theory
 - Lots of risk and how to manage it
- The Support Vector Machine
 - Risk minimisation and large margins
 - Formulation, Lagrangian, Dual, QP
 - The Kernel trick, standard kernels
- More information

THE NEW RELIGION



The New Religion

- The Support Vector Machine
 - A simple decision surface between +ve and –ve examples
 - The classifier uses only those training points which lie closest to the decision boundary – the **support vectors**
 - The inner product of an example with a linear combination of the support vectors is sufficient to determine the classification – based on its sign for binary classification:

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^t \mathbf{x} + b$$

where w is ultimately dependent upon the support vectors

Statistical Learning Theory

- Based around the problem of how rapidly, and under what conditions, an empirical quantity based upon l observations, converges uniformly to the value it would hold when l is infinite
- In the pattern classification problem, one has n observations of a pattern and its associated class:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$$

which we assume to be drawn IID according to the joint distribution generating the data

Classification and Risk

- Our task is to train a machine or classifier $f(\mathbf{x}; \boldsymbol{\alpha})$ through selection of an appropriate parameter vector $\boldsymbol{\alpha}$ so that the classifier performs well on unseen data.
- Now, if we assume that the unseen or test data are drawn IID from the same distribution as the training set, we can bound the expected error over such unseen examples, known as the [expected, actual] *risk* of the machine with respect to the loss functional L :

$$R(\boldsymbol{\alpha}) = E[L(y, f(\mathbf{x}; \boldsymbol{\alpha}))]$$

where the expectation is taken with respect to the joint distribution

$$P(\mathbf{x}, y)$$

Empirical Risk

- The error on the training set is usually known as the empirical risk, and is a fixed real number for a particular training set and machine.
- In our case we shall assume that f and y can take on values in $\{-1, 1\}$, and we shall measure the error as the so-called 0-1 loss. Then, the empirical risk is just:

$$R_{emp}(\mathbf{\alpha}) = \frac{1}{2l} \sum_{i=1}^l |y_i - f(\mathbf{x}_i; \mathbf{\alpha})|$$

which is basically the fraction of mistakes.

Structural Risk

- The function f is drawn from an hypothesis class H , with some known complexity level or capacity
- This capacity is usually expressed in classification problems through the non-negative integer h , a measure of the capacity of the function class containing f and known as the Vapnik-Chervonenkis dimension
- The greater the complexity of the model class, the easier it is reduce the empirical risk
- And of course, the easier it is to overfit the training data

Bounding the Risk

- Then Vapnik (1995 and earlier), established the result that for any $0 \leq \eta \leq 1$, the bound

$$R(\boldsymbol{\alpha}) \leq R_{emp}(\boldsymbol{\alpha}) + \varphi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right)$$

holds with probability $1 - \eta$

Bounding the Risk

- Here, $\varphi(\cdot, \cdot)$ is a positive function known as the VC confidence, which tends to zero with increasing l
- Note also that the VC confidence increases monotonically with h , so we cannot support hopes of a strong performance on the generalisation set if we allow the capacity of the classifier to grow too rapidly
- [Note that the lower bounds on the actual risk have a similar functional form]
- [By the way, has anyone seen a distribution recently?]

Dangerous ground

- I have a training set of observations, so I choose the *perfect* classifier – one which gets everything right on my training set – thus delivering zero empirical risk
- But, if I undertake this approach blindly, I might find that the model I have selected has an astonishingly high capacity, thus yielding an enormous VC confidence
- I can thus have no expectation that my classifier will perform well on unseen examples from the same distribution

Risk Averse Classifier Selection

- Selection of a classifier is thus a balancing act:
 - Goal of minimising error on the training set (*low empirical risk*)
 - Goal of limiting model capacity to avoid overfitting (*low structural risk*)
 - Need to keep both low to limit the generalisation error or *actual risk*
 - **SVMs are very good at doing this even when there are lots of features around**

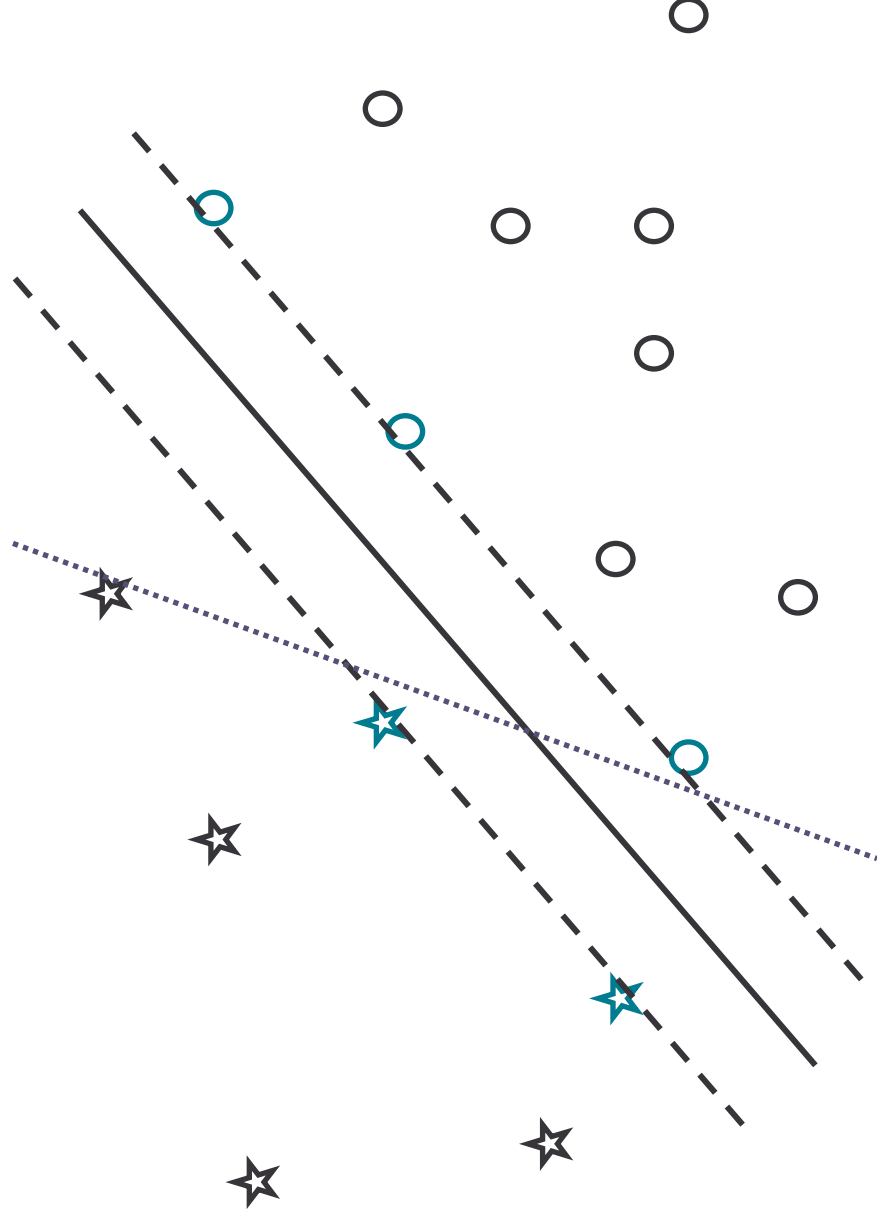
The Support Vector Machine

- The main idea of an SVM;
 - construct an hyperplane as the decision surface
 - Maximise the margin of separation between positive and negative examples – for SVM-like classifiers, the large margin acts as a capacity control
 - SVM is an approximate implementation of the method of structural risk minimization.

Risk Minimisation

- In the case of separable patterns, a SVM produces a value of zero for the empirical risk, while selecting the lowest structural risk classifier able to deliver this result
- [The precise idea of Structural Risk Minimization is discussed in the references at the end of the talk]

The Optimal hyperplane – linearly separable case



Finding the hyperplane

- Problem; find the parameters \mathbf{w}_0 and b_0 for the optimal hyperplane, given the training set $T = \{\{\mathbf{x}_i, y_i\}\}$
- The pair (\mathbf{w}_0, b_0) must satisfy the constraints of correct classification:

$$\begin{cases} \mathbf{w}_0^T \mathbf{x}_i + b_0 \geq +1 & \text{for } y_i = +1 \\ \mathbf{w}_0^T \mathbf{x}_i + b_0 \leq -1 & \text{for } y_i = -1 \end{cases}$$

- The particular data points $\{\mathbf{x}_i, y_i\}$ for which we have equality are called support vectors. Note that the support vectors do *not* lie on the decision surface, but rather at the extremes of the class

The Margin:

- The margin of separation between the two classes is equal to

$$\rho = \frac{2}{\|\mathbf{w}_0\|}$$

- Maximising the margin of separation between classes is equivalent to minimising the Euclidean norm of the weight vector \mathbf{w}_0
- The optimal hyperplane is unique in the sense that the optimum weight vector \mathbf{w}_0 provides the maximum possible separation between positive and negative examples.

Quadratic Optimisation:

Minimise the cost function

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

subject to the constraints

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq +1 \quad \text{for } i = 1, K, l$$

The Lagrangian Formulation:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^l \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

- The Lagrangian formulation has two key advantages
 - The constraints are replaced by simpler constraints on the Lagrange multipliers when we transform to the dual
 - The training data will then only appear in the problem through inner products between the training vectors – something which facilitates the kernel trick as we shall see later

Primal Optimality conditions

$$\left\{ \begin{array}{l} \mathbf{w} = \sum_{i=1}^l \alpha_i \gamma_i \mathbf{X}_i \\ \sum_{i=1}^l \alpha_i \gamma_i = 0 \end{array} \right.$$

$$\alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1] = 0 \quad \text{for } i = 1, K, l$$

The Wolfe Dual Problem

Find the Lagrange multipliers that maximise

$$Q(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j \gamma_i \gamma_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to the constraints

$$\left\{ \begin{array}{l} \sum_{i=1}^l \alpha_i \gamma_i = 0 \\ \alpha_i \geq 0 \quad \text{for } i = 1, \dots, l \end{array} \right.$$

Some comments

- The value of the Lagrange multiplier indicates whether or not the original constraint is active: positive α means that we have a support vector.
- Usually, only a small fraction of the training set constitute support vectors, and the solution machine is sparse, and rapidly evaluated (similar ideas emerge in the quadratic optimisation solvers: active set methods)
- If all other training points were removed, or numerous others added outside the margin tube, the same hyperplane would be found

What if the data aren't separable?

- Introduce slack variables to the optimisation problem, relaxing the classification constraints on the training set
- But, one must penalise relaxation so that relaxation is used only when necessary...
- There is then a trade-off between the margin and the cost imposed for the constraint violation

Constraint Relaxation

- Slack variables

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, K, l$$

- New primal cost function

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i$$

- C determines the penalty imposed for too much relaxation; an extreme value of C restores the hard-margin

Some comments

- The slack variables are there to handle patterns that would otherwise result in a classification error.
- $|\xi_i| \geq 1 \Rightarrow$ a genuine misclassification
- So the penalty term in the cost function on the previous slide is in fact a measure of the number of misclassifications scaled by the cost one is assigning to them...
- The value of C is thus an important parameter...
- [With the simple penalty term, the slacks and their LMs don't appear in the Wolfe Dual]

Revised Dual Problem

Find the Lagrange multipliers that maximize

$$Q(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to the constraints

$$\left\{ \begin{array}{l} \sum_{i=1}^l \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \quad \text{for } i = 1, K, l \end{array} \right.$$

Extension to non-linear decision functions

- Thus far we have worked with a decision function based around an inner product in the pattern space
- But, as noted earlier, the use of the Wolfe Dual for the Lagrangian means that the pattern vectors only appear in the data through their inner products...
- So if we were to transform the pattern vectors into a higher dimensional space – where the classes might be better separated – we might still find a linear decision boundary

A note of extreme caution

- In many applications in which the pattern space is sparse, we need not continue too far down this path
- If there is sufficient room to allow the placement of an adequate hyperplane then the transformation is unnecessary
- Many document processing tasks fall into this category, and there is some evidence in both NLP and bioinformatics that more complex Mercer kernels – polynomials and radial basis functions offer at best modest performance improvements.

A note of extreme caution

- These comments apply particularly to the bag of words based models such as those of Joachims considered below.
- BUT they do not imply that the search for addition NLP kernels is in vain; as we shall see in Lecture 3, the key is to tailor application and structure specific kernels as a pre-processing step, and to exploit the SV framework by applying a linear kernel to the new feature vector.
- Minimally, however, you *must* use a linear kernel to baseline your work.

We return to the non linear classifier

- Non-linear transformation $x \rightarrow \Phi(x)$ into a high dimensional feature space H
 - Use linear classifier in H
 - Corresponds to non-linear classifier in pattern space
- Classifier relies only on the inner products

$$f(x) = \sum_{i \in S} \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b$$

- But the inner products may be prohibitively expensive...

Unless we use The Kernel Trick

- Using the mapping into the (possibly infinite dimensional) space H , we have expensive inner products of the form: $\varphi(x_i)^T \varphi(x_j)$
- What if there existed a Kernel function in the pattern space such that $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$
- Never need to leave R^n and need not determine φ

Mercer's Posthumous Fame

- For which kernel $K(x_i, x_j)$ does there exist a pair $\{H, \varphi\}$?
- Mercer's condition (1905) : there exists a mapping φ and an expansion

$$K(x_i, x_j) = \sum_k \varphi(x_i)_k^T \varphi(x_j)_k$$

$$\text{iff, } \int g(\mathbf{x})^2 dx < \infty \Rightarrow \int K(x_i, x_j) g(x_i) g(x_j) \geq 0$$

More general problems

- Note that there need not be an inner product defined in the original pattern space. The transformation may be defined so as to embed notions of similarity between patterns in the framework of an inner product
- The Best examples here are String and Graph based kernels for language processing and bioinformatics.
- This makes the approach incredibly general, but the magic of Mercer's theorem is diminished.

Some Comments

- Mercer's condition guarantees the existence of a transformation and an appropriate Hilbert space, but it doesn't allow characterisation of either – determination of the ideal kernel for a particular situation is a research problem
- Transformation to a high dimensional Hilbert space suggests that we should be overfitting like crazy; some SVMs have infinite VC dimension, or unbounded capacity
- Maximising the margin controls the capacity of the classifier, and prevents overfitting

Practical SVM training relies on a kernel

Find the Lagrange multipliers that maximize

$$Q(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to the constraints

$$\left\{ \begin{array}{l} \sum_{i=1}^l \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \dots, l \end{array} \right.$$

With the resulting classifier

- Classifier:
$$f(x) = \sum_{i \in S} \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b$$
- Using the kernel:
$$f(x) = \sum_{i \in S} \alpha_i y_i K(x_i, x) + b$$
- May never leave the pattern space, but the kernel represents the relative position of all the training vectors in the feature space H

Standard Kernels:

- Linear, Polynomial: $K(x_i, x_j) = (x_i \cdot x_j + c)^p$
 - Take $p=1$ and $c=0$ and one has the inner product of the pattern space, the simple linear kernel
- RBF: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|_2^2)$
 - SVs are the RBF centres, selected automatically
- Sigmoid: $K(x_i, x_j) = \tanh(ax_i \cdot x_j - c)$
 - Essentially a 2-layer neural network; careful with the parameters, as one may violate Mercer's condition

A bag of words application

- The Reuters “corporate acquisitions” task
- Thorsten Joachims’ dissertation work
- Illustrates profound advantages over neural network and some other approaches
- Here our task is to classify documents according to their relationship to the subject area of “corporate acquisitions”
- The original “bag of words” model

A bag of words application

- Each Reuters' story is considered a separate document, and represented as a single pattern vector.
- Here, we define the dimension of our pattern vectors to be the size of the lexicon, and each document is modelled by a sparse vector reflecting the normalised counts of each possible word stem. $N=9947$ in the example
- Each vector is real valued and a variety of standard kernels are employed

A bag of words application

The first few values from the first line of the test set:

+1 6:0.034259 26:0.148286 27:0.057004 31:0.037309

A positive example

stem 6: said

Stems 26,27, 31: share,
Normalised pct, corp
frequency

A revolutionary application

- Astonishingly rapid training: under 5 seconds on a standard PC.
- Remarkable performance: under 5% generalisation error
- Could not have been attempted with a neural network approach
- How are these values affected by choice of kernel and scaling parameters? (see the exercises)

More General Advantages

- Model selection is much easier than for backprop and variations, with implicit capacity control
- Convexity of the optimisation problem ensures a global solution, without the local minima that plague neural network training [Solution is not unique, but there is no better value attainable]
- SVMs allow consideration of problems of incredibly high dimension – well beyond the limit for backprop neural networks. The Reuters task is a perfect example.

Some more general difficulties

- Selection of an appropriate kernel
- Intuitions take longer to develop than in the neural network case, and for some problems the results may be enormously sensitive to parameter selection
- [Remember to scale the input where necessary]
- Experiment with different values of the cost C until you have achieved an appropriate balance between training errors and generalisation performance
- Not every problem is noise free 😊

Some Ideas Related to the SV methods:

- Support Vector Regression
- Bayes Point Methods
- Kernel PCA, clustering, LSA/LSI
- Specialist problem kernels:
 - Incorporation of prior knowledge
 - Specialist kernels for image processing
 - Specialist kernels for language processing and bioinformatics (see Lecture 3)

Current work:

- I have this classification problem, which kernel is the best for me to use?
- Kernel-based density estimation
- Combination of SV classifiers using voting methods
- Automatic parameter selection – the so-called ν -SVMs
 - These are now quite mature but are better viewed after the vanilla SVM approaches have been discussed

Where to get more information:

- Key website:
 - www.kernel-machines.org
 - Books
 - Tutorial articles
 - Software Links
 - Applications
- Google searches for support vector machines
 - Include “citeseer” in the search terms

Books:

- N. Cristianini and J. Shawe-Taylor
AN INTRODUCTION TO SUPPORT VECTOR MACHINES
(and other kernel-based learning methods)
Cambridge University Press
2000 ISBN: 0 521 78019 5
www.support-vector.net
- Bernhard Schölkopf and Alex Smola. **[Learning with Kernels](#)**.
MIT Press, Cambridge, MA, 2002. *An introduction and overview
over SVMs. A free sample of one third of the chapters
(Introduction, Kernels, Loss Functions, Optimization, Learning
Theory Part I, and Classification) is available on the book website.
([650 pages, \\$60](#)).*
www.learning-with-kernels.org

Tutorial Articles (linked from site):

- C. J. C. Burges. [A Tutorial on Support Vector Machines for Pattern Recognition](#). *Knowledge Discovery and Data Mining*, 2(2), 1998.
- Bernhard Schölkopf. [Statistical learning and kernel methods](#). MSR-TR 2000-23, Microsoft Research, 2000
- A. J. Smola and B. Schölkopf. [A Tutorial on Support Vector Regression](#). NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

Software (linked from site):

- [SVMLight](#). by [Thorsten Joachims](#).
 - Very convenient; compiles anywhere
 - Very, very fast – even for large problems
- [SVMTorch](#). Support Vector Machine for Large-Scale Regression and Classification Problems.
- [Torch](#). a new machine learning library in C++/GPL including MLP, RBF, SVM, GMM, HMM, KNN, Parzen...
- Numerous others, including matlab libraries

The Religious Experience

- It is a very demanding religion...
- The catechumenate is long and challenging
 - Hopefully I have enabled you to encounter the main revelations in the one gulp
 - Often, these come unexpectedly
- But, the tutorial papers are very much worth a look, as is the software due to Joachims and others