Support Vector Machines

Outline

1. Overview
2. Support Vector Machines
   – The linear, separable case
   – Extension to non-separable data
   – Beyond linear classification with kernels
3. Getting practical
   – The “cookbook approach”
4. Conclusions

Why SVMs?

1. Conceptually simple
2. Powerful and elegant
3. Fast

Cronology

1979
- Underlying theory developed, enunciation of the Structural Risk Minimization principle: *Vapnik, "Estimation of Dependences Based on Empirical Data", Moscow, 1979.*

1992
- SVMs introduced in COLT-92: *Boser, Guyon, Vapnik, "A training algorithm for optimal margin classifiers", Computational Learning Theory, Pittsburgh,1992.*

1995
- Framework extended to non-separable problems: *Cortes, Vapnik, "Support Vector Networks", Machine Learning, 1995*.

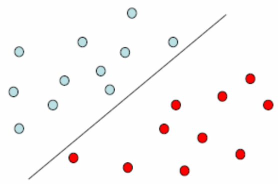
1999
- A fast algorithm for training and testing is proposed: *Platt, "Fast training of support vector machines using sequential minimal optimization", in "Advances in Kernel Methods", MIT Press, Cambridge, MA, 1999.*

Linear SVMs

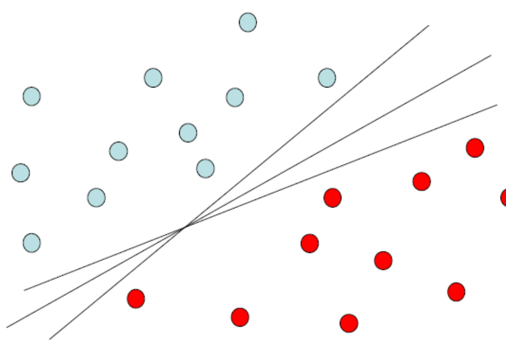
- Binary classifier
- Data is linearly separable
  - exists a hyperplane that divides the classes
  - given a set $\{x_i, y_i\}$ of tuples and their labels

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i \in \{1, \ldots, m\}$$

$$\mathbf{w} \in R^m, \quad \mathbf{x}_i \in R^m, \quad b \in R, \quad y_i \in \{+1, -1\}$$

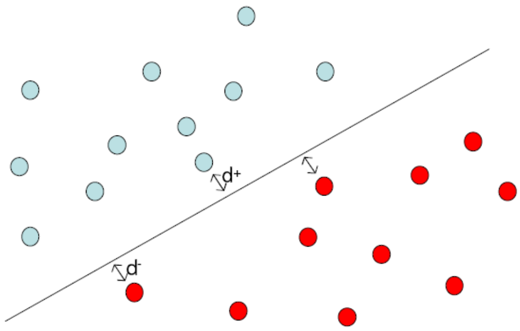


Finding the hyperplane



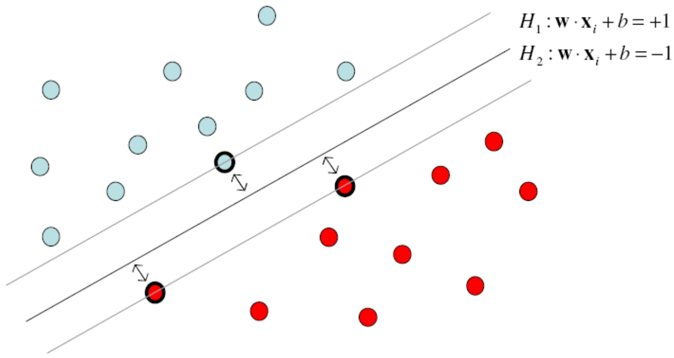
- What is the best separating hyperplane?

Maximizing the margin



- SVMs maximize the margin $d = d^+ + d^-$
- d^+ (d^-) is the minimum distance to the nearest positive (negative) sample

Support vectors



- The samples which lie on the lines H_1 and H_2 are called *Support Vectors*

Finding the hyperplane

- If ||w|| is the euclidian norm of w, then:

$$d^+ = d^- = \frac{1}{\|\mathbf{w}\|}$$

- Maximizing the margin $d = d^+ + d^- = \frac{2}{\|\mathbf{w}\|}$

  is equivalent to minimize the quantity: $\min \frac{1}{2} \mathbf{w} \cdot \mathbf{w}$

  with $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1..n\}$

Solving for the hyperplane

- The problem outlined is an instance of a constrained quadratic optimization and hence can be solved using the technique of Lagrange multipliers:

  $$L_p(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \right)$$

  to be minimised wrt *primal variables* ***w*** and *b*, and maximised wrt the dual variables α_i

Solving for the hyperplane

- Conditions for the saddle point are:

$$\frac{\partial}{\partial b} L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \qquad \frac{\partial}{\partial \mathbf{w}} L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$$

- which lead to:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \qquad \text{and} \qquad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

- The solution vector has then an expansion in terms of a subset of training patterns, namely those patterns whose α_i is non-zero, called *support vectors*

Solving for the hyperplane

- Since $\alpha_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right) = 0, \quad i = 1, \ldots, m$
  the support vectors lie on the margin and all remaining examples of the training set are irrelevant (i.e., the contraints does not play a role in the optimization).
- Substituting the equation in the previous slide into L_p, primal variables are eliminated and the dual form of the optimization problem is found:

$$L_D(\boldsymbol{\alpha}) = \sum_i \alpha_i - \tfrac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

- subject to: $\alpha_i \geq 0, \quad i = 1, \ldots, m, \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0$

Solving for the hyperplane

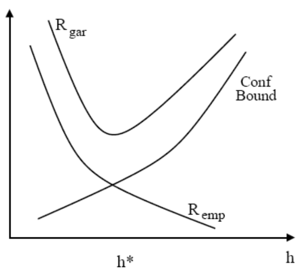
- **w** is a linear combination of all training data for which $\alpha_i \neq 0$, who are the support vectors
- An optimal solution always exists since in a quadratic optimization problem there are no local minima (convex problem)
- Number of variables is equal to be number of training vectors: ad-hoc algorithms have been devised to limit the computational cost
- Classification of new vector **x** is obtained computing

$$f(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

- where *b* is computed by using $\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0$

Structural Risk Minimization (SRM)

- For obtaining the minimum risk, both the empirical risk (measured on the training set) and the ratio between the Vapnik-Chervonenkis dimension (denoted by *h,* it is a measure of the classifier complexity) and the number of points (i.e. h/ℓ) must be minimized
- The empirical risk depends on *h*. It is typically a decreasing function with respect to *h*

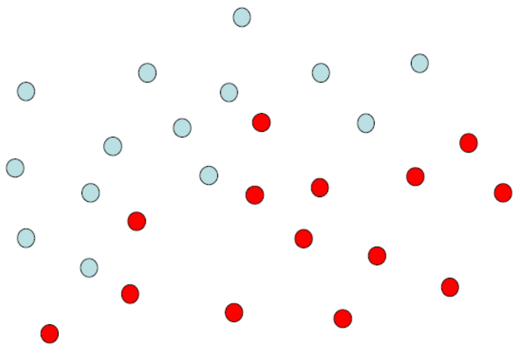


- It is necessary to find a trade-off when minimizing the two quantities
  - the structural risk minimization (SRM) principle followed by the SVMs

Summary: linear SVMs

- Binary classifier
- Finds the best separating hyperplane
- The hyperplane is described by a small subset of training data, called *support vectors*
- The procedure is fast (polynomially bounded)
- We are guaranteed to find an optimal solution
- The method is statistical, not probabilistic
- Structural Risk Minimization

Not linearly separable classes

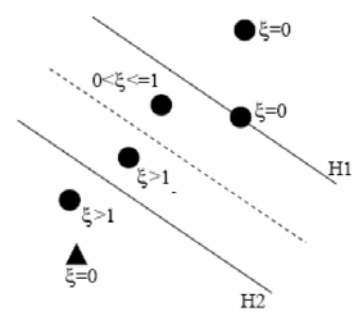


- What do we do if data are not linearly separable?

Slack variables

- Quadratic optimization does not converge for non linearly separable data
- We modify the constraints adding "slack" variables, which enable vectors to cross the margin

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$$



- The value of the ξ_i indicates the position of the vector respect to the hyperplane

Optimization with slack variables

- The new objective function is

$$\min \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \cdot \left(\sum_1^m \xi_i \right)$$

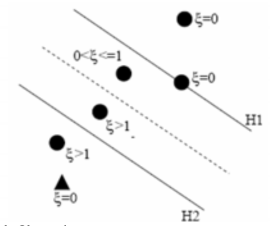
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

- *C* is a user-specified parameter which represents the cost for misclassified data
- *C* determines the sensibility of the classifier to errors and its generalization performance

Solution for NLS data

- The constrained system can be solved in its dual form, solution is again

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$



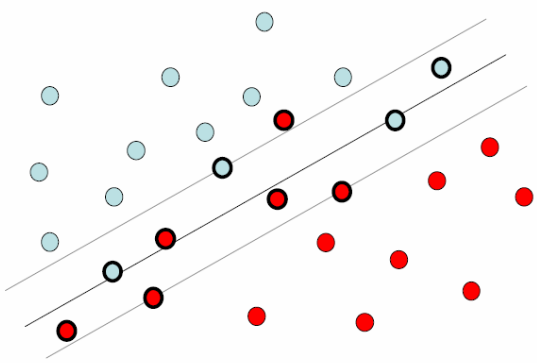
- Alpha values can be interpreted

$\alpha_i = 0$ point is correctly classified

$0 < \alpha_i < C$ correctly classified but outside H_i

$\alpha_i = C$ incorrectly classified, error

Support vectors in the NLS case



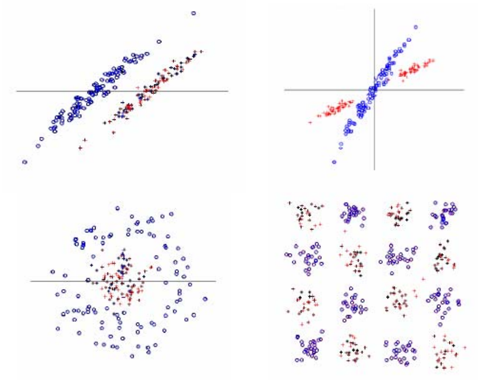
- Support vectors are all points for which $\alpha \neq 0$ (misclassified data are also support vectors)

Summary: SVMs in the NLS case

- Slack variables are introduced to allow vectors to cross the margin
- Support vectors contain every vector which lies on or *beyond* the margin
- We now need a free arbitrary parameter, the misclassification cost C

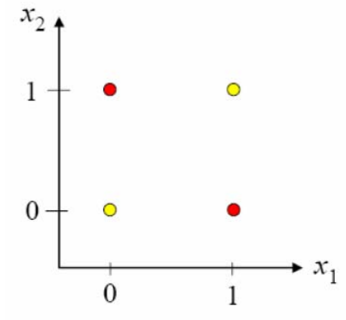
Complex examples

- What do we do when linear hyperplanes don't suffice?



A simpler example: XOR

- Not linearly separable (Minsky & Papert '69)
- *How can we tackle a problem like this with a SVM?*



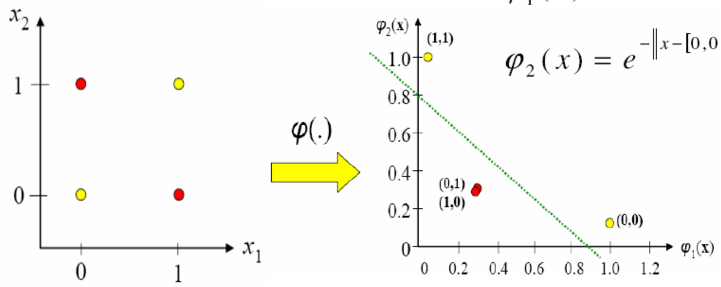
Data mapping

- Could it be separable when mapped into another space?

$$x = [x_1, x_2]$$

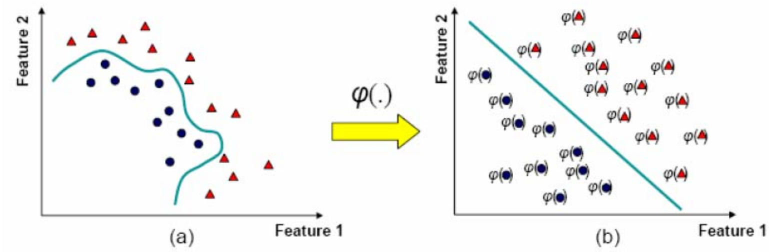
$$\varphi_1(x) = e^{-\left\| x - [1,1]^T \right\|}$$

$$\varphi_2(x) = e^{-\left\| x - [0,0]^T \right\|}$$



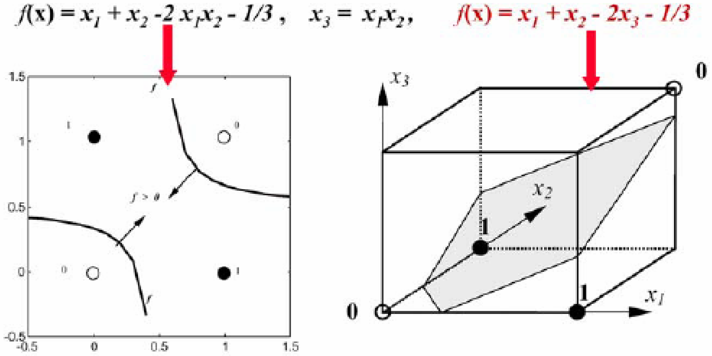
Data mapping: idea

- A non linearly separable dataset can be mapped to another space (possibly of higher dimension), where a separating hyperplane exist



Mapping: one last example

$f(\mathbf{x}) = x_1 + x_2 - 2\,x_1 x_2 - 1/3\,, \quad x_3 = x_1 x_2, \quad f(\mathbf{x}) = x_1 + x_2 - 2x_3 - 1/3$



Problems of the mapping idea

1. Mappings can have huge dimensionality (even infinite)
2. Mappings are in general difficult to compute
3. It is not clear how to find the proper mapping that will separate the data

Applying mapping to SVMs

- Crucial observation is that feature vectors in SVM training appear only in the form of dot products ($\mathbf{x}_i \cdot \mathbf{x}_j$)
- After applying a map, the training will be carried over the product $\varphi(x_i) \cdot \varphi(x_j)$
- If we could find a function *K* defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

  we could train the classifier without even ever bothering to explicitly compute the mapping φ.
- We call this function **Kernel** (and the technique *kernel trick*)

The kernel concept

- Using a kernel, a SVM can work in a space of huge dimensionality while using the original algorithms
- The mapping to the target space is never calculated esplicitly, so it can be arbitrarily complicated
- We avoid the curse of dimensionality because the resulting classification algorithm is indipendent from the size of the target space
- Kernels can be seen as a problem specific module fitted in a general purpose algorithm

How to find a kernel function?

- A valid kernel satisfies the Mercer condition:
  - defined the Gram matrix G as

$$G = \begin{bmatrix} \langle \Phi(x_1), \Phi(x_1) \rangle & \cdots & \langle \Phi(x_1), \Phi(x_j) \rangle & \cdots & \langle \Phi(x_1), \Phi(x_m) \rangle \\ \vdots & \ddots & & & \vdots \\ \langle \Phi(x_i), \Phi(x_1) \rangle & & \langle \Phi(x_i), \Phi(x_j) \rangle & & \langle \Phi(x_i), \Phi(x_m) \rangle \\ \vdots & & & \ddots & \vdots \\ \langle \Phi(x_m), \Phi(x_1) \rangle & \cdots & \langle \Phi(x_m), \Phi(x_j) \rangle & \cdots & \langle \Phi(x_m), \Phi(x_m) \rangle \end{bmatrix}$$

- ϕ is a kernel iff G is symmetric and semidefinite positive (all eigenvalues ≥ 0)

Standard kernels

- Linear $K(x,z) = \langle x, z \rangle$
- Polynomial $K(x,z) = \left(\langle x, z \rangle + 1\right)^p$
- Radial basis functions $K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$
- Sigmoid $K(x,z) = \tanh(a\langle x, z \rangle + b)$

Kernel selection

- Linear kernel
  - Used when the feature space is huge (for example in text classification, which uses individual word counts as features)
  - Shown to be a special case of the RBF kernel
  - No additional parameters
- Polynomial
  - Has numerical difficulties approaching 0 or infinity
  - A good choice for well known and well conditioned tasks
  - One additional parameter (degree *p*)

Kernel selection

- Radial basis functions
  - Indicated in general as the best choice in the literature
  - One additional parameter (σ)
- Sigmoid
  - Two additional parameters (a and b)
  - For some values of a and b, the kernel doesn't satisfy the Mercer condition
  - From neural networks
  - Not recommended in the literature
- *Choosing the right kernel is still an art!*

SVM Generalization

- Given a training set ℓ, let be P(error) the risk evaluated on a set of ℓ-1 samples and $E[P(\text{error})]$ the average value evaluated on all the possible choice of ℓ (leave-one-out error). Let E[#support vector] be the average number of SVs on the ℓ SVM trainings.
- The following relation holds (Vapnik leave-one-out bound 1995):
  - $E[P(\text{error})] \leq E[\#\text{support vector}] / \ell$
- The generalization performance is obtainable after training, by counting the number of SVs

Comments

- Kernel selection and parameter tuning are critical
- Cost *C* has a huge impact on the generalization ability
- Lowering degree or sigma can avoid overfitting
- Number of support vector is a measure of generalization performance

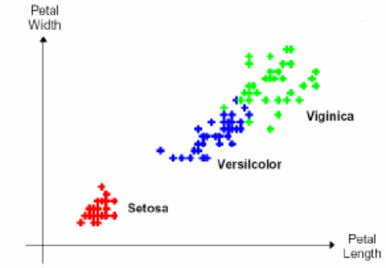
Cookbook approach

1. Conduct simple scaling on the data
2. Consider RBF kernel
3. Use cross-validation to find the best parameter C and σ
4. Use the best C and σ to train the whole training set
5. Test

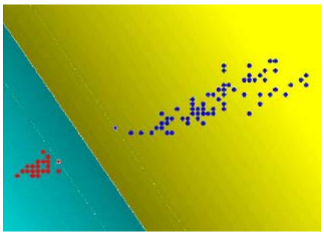
Cookbook problems

- Parameter search can be very time consuming
  Solution: conduct parameter search hierarchically
- RBF kernels are sometimes subject to overfitting
  Solution: use high degree polynomials kernels
- Parameter search must be repeated for every chosen features; there no reuse of computations
  - Solution: compare features on random subsets of the entire dataset to contain computational cost
- Search ranges for C and σ are tricky to choose
  Solution: literature suggest using exponentially growing values like $C = 2^{[-5..15]}$ and $\sigma = 2^{[-15..5]}$

An Example: IRIS data

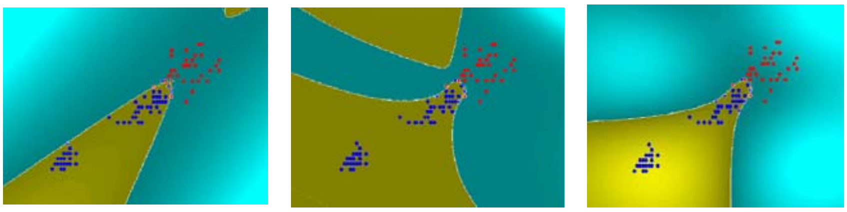


- Setosa and Versicolor classes are linearly separable



An Example: IRIS data

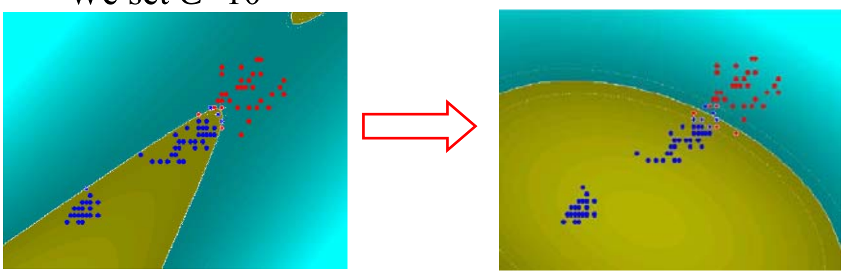
- Virginica e Versicolor are non linearly separable in the feature space
- Let us analyze, in the feature space, the decision boundaries obtained with different kernels



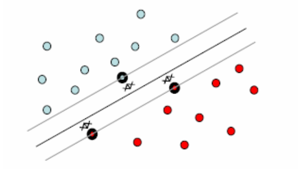
- From left to right:
  - Polynomial degree 2
  - Polynomial degree 10
  - RBF σ=1.0

An Example: IRIS data

- How does decision boundary change for a polynomial kernel of degree equal to 2 when accepting misclassifications on the training set?
  - We set C=10

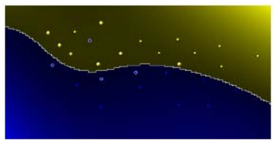


Summary



$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$

- Support Vector Machines, two key ideas
  – Margin maximization
  – Kernel trick
- Training
  – a quadratic optimization problem
  – always convergent to the optimal solution
  – solvable in polynomial time
- Free parameters
  – The cost *C*
  – The kernel type
  – The kernel parameters



Advantages

- The SVM theory is an elegant and highly principled
- SVMs have a simple geometric interpretation
- The use of kernels provides a efficient solution to non-linear classification and dispels the ”curse of dimensionality”
- Convergence to the solution is guaranteed
- Support vectors give a compact representation of the entire dataset; their number is a measure of the generalization perfomance

Disadvantages

- Kernel and parameter choice is crucial
- Training can sometimes be tricky and time consuming
- Training is not incremental; the whole dataset must be processed for every new addition
- There are no optimized extension to multi-class problems; a problem with *N* classes requires *N* classifiers