Short course

A vademecum of statistical pattern recognition and machine learning

The support vector machine (SVM) and structural SVM

Massimo Piccardi
University of Technology, Sydney, Australia

Agenda

- Introductory notes
- Preliminary geometry
- The support vector machine
- The soft-margin case
- Multi-class SVM as combination of binary classifiers
- Multi-class SVM
- Structural SVM
- Example: sequential labeling with Hamming loss
- Structural SVM with latent variables

The support vector machine

- The support vector machine (SVM) is a classifier based on the notion of maximum margin between classes
- It was invented by Vladimir Vapnik and often credited by reference [Cortes and Vapnik 1995]. Major contributions from Shawe-Taylor, Cristianini, Schölkopf, Smola and apologies to the many others not cited
- This presentation is aimed to introduce the *structural* (aka *structured-output*) *SVM* which is useful, amongst others, in problems with time series and spatial structure

Corinna Cortes and Vladimir Vapnik, "Support-vector networks," Machine Learning, 20 (3): 273-297, 1995

The support vector machine

The topic is vast. I dare say that there are at least two ways to present it:

- A more theoretical point of view which emphasises the *margin* between classes and its maximisation. For this reason, the SVM is often referred to as a "maximum-margin" or "large-margin" classifier
- A more "practical" point of view based on the notion of *regularised empirical classification error* and its minimisation. For this reason, the SVM is increasingly referred to as "minimum empirical risk" classifier. The analogy with CRFs is striking

A quick review of analytic geometry

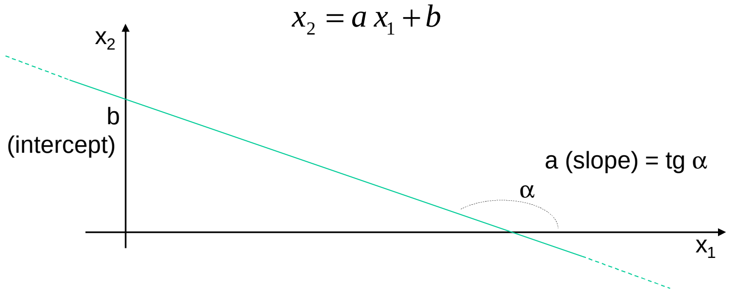
- Before discussing notions of classification, we recall a few concepts of analytic geometry:
  - the implicit equation of a straight line in the 2D plane
  - the half planes
  - the normal vector to the line
  - the (signed) distance of the line from the origin
  - the (signed) distance of any other point from the line
  - the (signed) distance between any two parallel lines
- Once these concepts are clear per se, the introduction of the SVM is greatly simplified

© Massimo Piccardi, UTS 5

Straight line in the 2D plane

- A straight line in a 2D plane of coordinates (x_1,x_2) can be expressed by the *slope-intercept* (aka *explicit*) equation:

$$x_2 = a\, x_1 + b$$



- Any possible straight line in 2D can be represented by an appropriate choice of a, b

© Massimo Piccardi, UTS 6

Straight line: implicit equation

- Please note that the following equation represents exactly the same straight line (i.e. it is satisfied by the same set of (x_1,x_2) points):

$$kx_2 = ka\,x_1 + kb \qquad (k \neq 0)$$

- Parameter k is redundant, in that it does not extend the set of representable lines. Now, manipulate the above as:

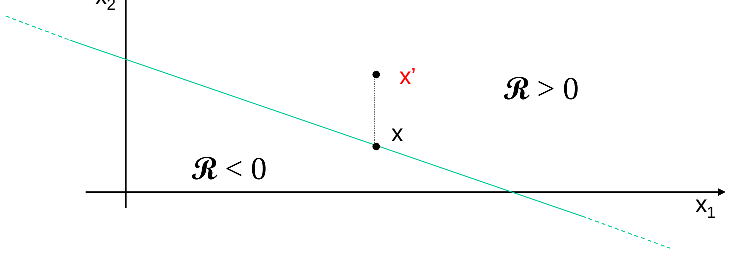
$$kx_2 - ka\,x_1 - kb = 0$$
$$\rightarrow w_2 x_2 + w_1 x_1 + w_0 = 0$$

with obvious positions. The above is known as *implicit equation*

Half planes

- Let us assume for convenience that w_2 is > 0. If we consider point x' = (x'_1,x'_2) in figure, we can be sure that:

$$w_2 x'_2 + w_1 x'_1 + w_0 > 0$$



because x satisfies the equation and x' has a larger second co-ordinate. The line cuts the plane in two *half planes*, $\mathcal{R} > 0$ and $\mathcal{R} < 0$

Extension to N dimensions

- We are working in 2D because it's easy to visualise, but exactly the same considerations can be applied to N dimensions. In that case, the plane becomes an N-D space and the straight line a (N-1)-D subspace (a *hyperplane*; the half planes become *half spaces*)
- It is common to compact the notation using w = $[w_1,..w_N]^T$ and x = $[x_1,..x_N]^T$:

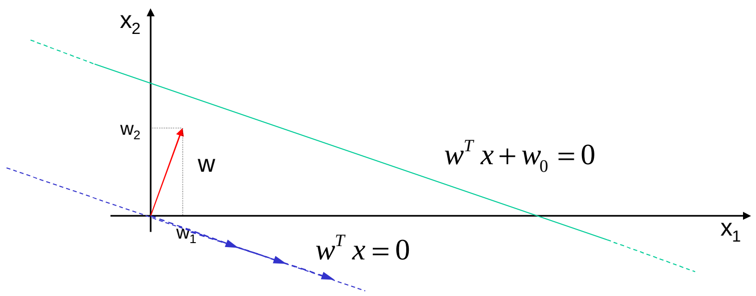
$$w^T x + w_0 = 0$$

- At times, you can see x extended with a pseudo-coordinate of constant value 1 as in x' = $[1,x_1,..x_N]^T$ and w' = $[w_0,w_1,..w_N,]^T$. In this case, the equation simply becomes:

$$w'^T x' = 0$$

Properties

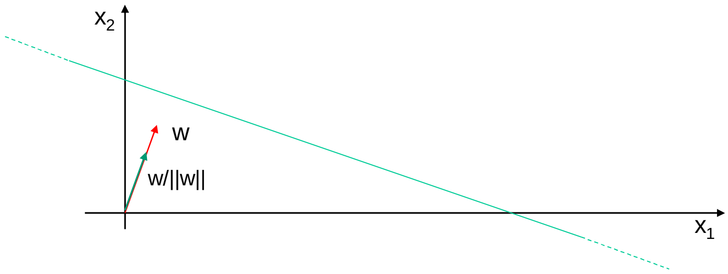
- Vector w is normal to the line:



- Consider line w^T x = 0, parallel to w^T x + w_0 = 0 and passing by the origin: w must be orthogonal to all its points

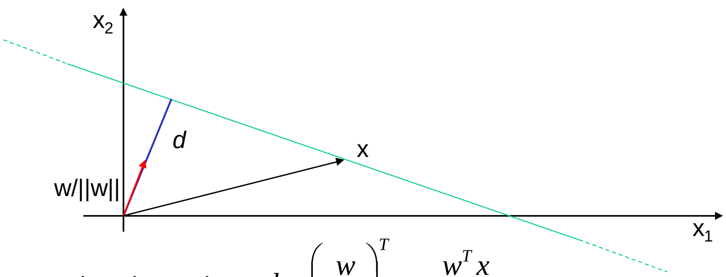
Properties

- The norm of vector w is noted as ||w||. w/||w|| is the unit vector with the same direction as w



Properties

- The signed distance, d, between the line and the origin is given by the inner product between any x ∈ line and unit vector w/||w|| :



- We then have that: $d = \left(\frac{w}{\|w\|}\right)^T x = \frac{w^T x}{\|w\|}$

- In addition, given that $w^T x + w_0 = 0$, we also have that $\mathbf{d = -w_0 / \|w\|}$ (a constant)

Properties

- **Please note!** that the geometric distance between the line and the origin does not change if we scale w and w_0 by k:

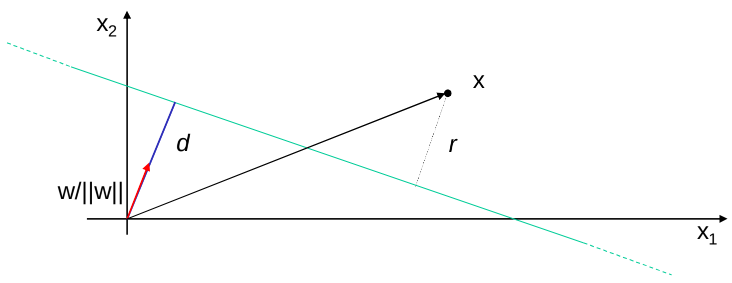
$$d = \frac{-k w_0}{\|k w\|} = \frac{-w_0}{\|w\|}$$

this is the consequence of the redundant parameter in the implicit equation

Properties

- The signed distance, r, between any point x in the plane and the straight line is given by:

$$r = \frac{w^T}{\|w\|} x - d = \frac{w^T x + w_0}{\|w\|}$$

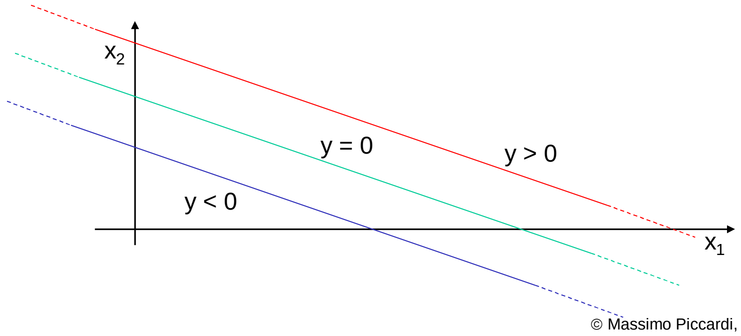


Properties

- We can extend the implicit equation to any arbitrary value, y:

$$y = w^T x + w_0$$

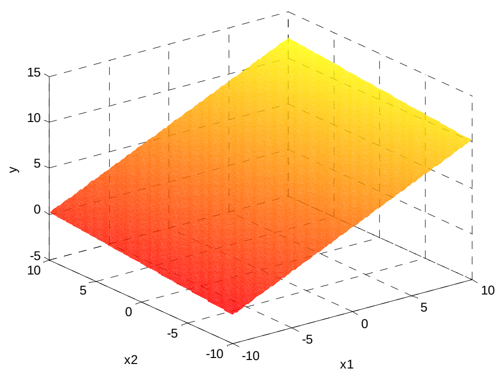
- For any given y, the solution is a straight line parallel to that for y = 0:



© Massimo Piccardi, UTS 15

Properties

- The same equation, but visualised in 3D:
  ($w_2 = 0.1$, $w_1 = 0.6$, $w_0 = 5.2$)

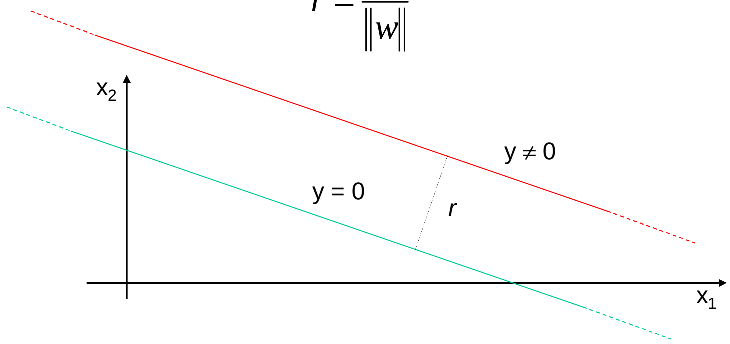


© Massimo Piccardi, UTS 16

Properties

- The signed distance, r, between any line $w^T x + w_0 = y$ and line $w^T x + w_0 = 0$ is given by:

$$r = \frac{y}{\|w\|}$$

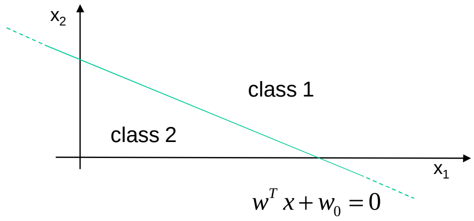


 17

Two-class linear classifier

- Note that our model, (w, w_0), is a two-class (i.e., binary) linear classifier:

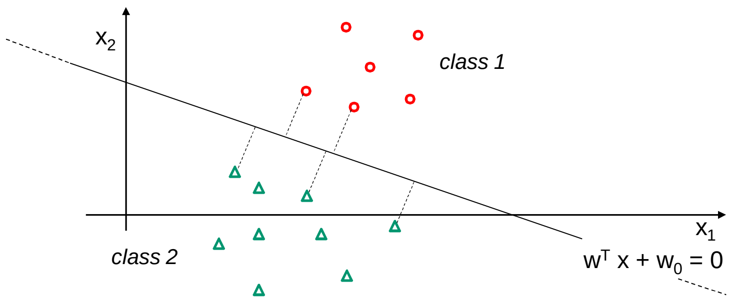
$$w^T x + w_0 \begin{cases} \geq 0 & \rightarrow class\ 1 \\ < 0 & \rightarrow class\ 2 \end{cases}$$



 18

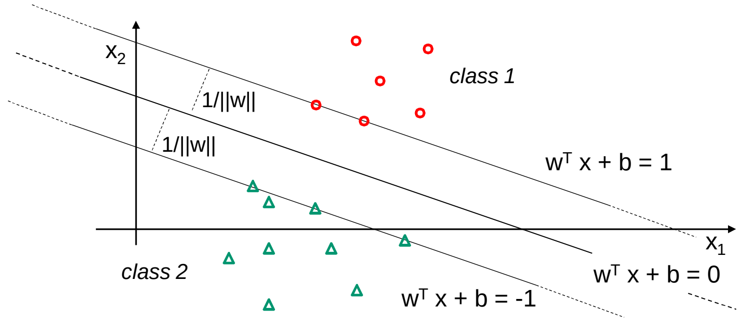
The support vector machine

- Given two sets of samples from two classes, the support vector machine aims to determine a separating hyperplane which is at the **maximum distance** from the **closest points** of both classes:



The support vector machine

- We replace w_0 with b to follow common notations
- We assume the arbitrary scale of w, b to be such that the closest points lie on $w^T x + b = 1$ and $w^T x + b = -1$



The objective function

- Therefore, the objective function of SVM can be expressed as:

$$w^*, b^* = \arg\max_{w,b} \frac{1}{\|w\|} \quad s.t.$$
$$w^T x_i + b \geq 1 \quad \forall x_i \in class1$$
$$w^T x_i + b \leq -1 \quad \forall x_i \in class2$$

The objective function

- For convenience, we can express the two class labels as y = {+1, -1} and compact the constraints. We can also manipulate the objective to obtain the equivalent:

$$w^*, b^* = \arg\min_{w,b} \frac{1}{2}\|w\|^2$$
$$s.t. \quad y_i\left(w^T x_i + b\right) \geq 1 \quad \forall x_i$$

- The objective is a convex function (a quadratic) subject to linear inequality constraints (it is solvable)

Inference

- As said previously, the inference of the class (aka prediction, classification) for a new point, x, is given by:

$$w^T x + b \underset{class\,2}{\overset{class\,1}{\gtrless}} 0$$

- An alternative notation:

$$y^* = \arg\max_y y\left(w^T x + b\right)$$

Learning: the primal problem

- Minimising the objective function subject to the inequality constraints can be done in terms of a Lagrangian equation:

$$L(w, b, \alpha_{1:N}) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i\left(w^T x_i + b\right) - 1\right]$$

$$p^* = \min_{w,b} \max_{\alpha_{1:N} \geq 0} L(w, b, \alpha_{1:N})$$

- This problem is known as the *primal problem*; *p** is the sought constrained minimum and $w^*, b^*, \alpha^*_{1:N}$ are the arguments of L() where it occurs

Learning: the primal problem

Some references:

- Olivier Chapelle, Training a Support Vector Machine in the Primal, Neural Computation 19(5): 1155-1178 (2007)
- SVM^{perf}: Joachims, T., “Training linear SVMs in linear time,” KDD, 2006.
- BMRM (Bundle Methods for Regularized Risk Minimization): C. H. Teo, Q. Le, A. J. Smola and S. V. N. Vishwanathan, A Scalable Modular Convex Solver for Regularized Risk Minimization, KDD, 2007
- Yu, J., Vishwanathan, S. V. N., Günter, S., and Schraudolph, N. N., “A quasi-Newton approach to nonsmooth convex optimization,” ICML 2008

Learning: the dual problem

- It can be proven that the same maximum, p*, and the same argmax, $w^*, b^*, \alpha^*_{1:N}$, can be obtained by solving the following dual problem:

$$d^* = \max_{\alpha_{1:N} \geq 0} \min_{w,b} L(w, b, \alpha_{1:N})$$

- Learning using the dual problem has various advantages (as we will see, it allows us to use kernels and simplifies the treatment of the non-separable case)

Learning: the dual problem

- We derive the internal minimisation:

$$\frac{\partial L(w,b,\alpha_{1:N})}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\frac{\partial L(w,b,\alpha_{1:N})}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

- Function L is convex and there is no need to compute the second derivatives

Learning: the dual problem

- Replacing these results in L we obtain:

$$L(w,b,\alpha_{1:N}) = \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y_i x_i\right)^T \left(\sum_{i=1}^{N} \alpha_i y_i x_i\right)$$

$$- \sum_{i=1}^{N} \alpha_i y_i \left(\sum_{i=1}^{N} \alpha_i y_i x_i\right)^T x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i =$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1 \ldots N$$

Note page

Learning: the dual problem

- The dual problem becomes:

$$\arg\max_{\alpha_{1:N}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \quad i = 1 \ldots N$$

- From the solution of the dual problem, $\alpha_{1:N}$*, it is then easy to obtain w*,b* (see Ng and Bishop in the references)

Inference in the dual

- It can be shown that this optimisation satisfies the Karush-Kuhn-Tucker conditions: the α_i of all points x_i not lying at the minimum distance from the boundary must be 0!

- The inference with the dual formulation becomes:

$$\sum_{i=1}^{N} \alpha_i y_i \langle x_i, x \rangle + b \ \underset{class\,2}{\overset{class\,1}{\gtrless}} \ 0$$

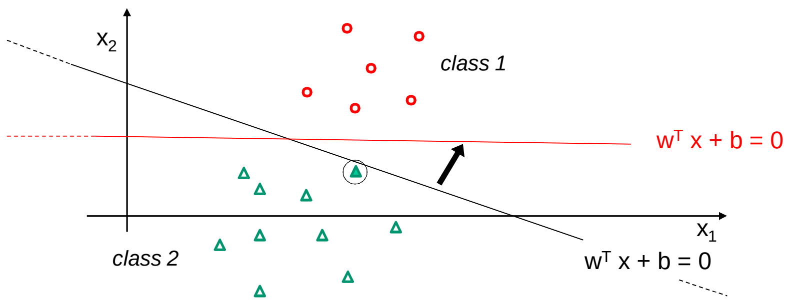
since only a few $\alpha_i \neq 0$, the dual inference is efficient (it is a sparse non-parametric classifier)

Kernels

- In the dual formulation, both learning and inference depend on the data only through the inner product, <x,x'>
- We can then replace the inner product by some other non-linear kernels such as the Gaussian and X^2 kernel, and the mechanics of inference and learning are unchanged
- In data space, this equates to a non-linear classifier
- Obviously, we can also manipulate the data into other features, $x \rightarrow \phi(x)$, prior to applying the SVM to perform it in ϕ-space rather than x-space

Soft-margin SVM

- In some cases, maximising the margin between the closest points of the two classes may not be an ideal strategy. A single "outlier" (▲) can affect the separating hyperplane significantly:



Soft-margin SVM

- In such cases, one can modify the constraints so as to tolerate a few points that do not meet the " ≥ 1 " constraint. For each such point, a penalty is accrued to the objective. The objective becomes a trade-off between minimising $||w||^2$ and minimising the total penalty:

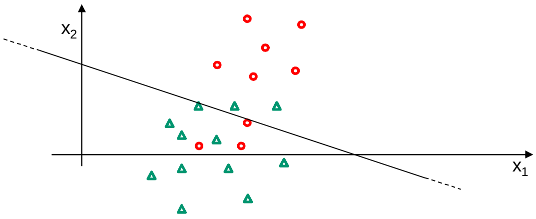
$$w^*, b^* = \arg\min_{w,b,\xi_{1:N}} \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \right)$$

$$s.t. \quad y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \quad \xi_i \geq 0,\ i = 1 \ldots N$$

- C is an arbitrary weight. The ξ_i are called *slack variables*. It is possible to use ξ_i^2 as penalty to discourage large individual errors

Soft-margin SVM

- Exactly the same constrained objective can be used for the much more realistic case of non-separable classes: classes for which there exists no hyperplane that can separate all points from both classes:



- In this case, the penalty is added for all violating points. The heavier the violation, the larger is ξ_i

Slack variables

- We have said that the slack variables, ξ_i, must satisfy constraints:

$$y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \quad \xi_i \geq 0,\, i = 1 \ldots N$$

  and that we aim to minimise their sum

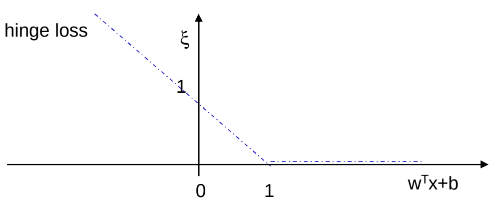
- Let us therefore satisfy the constraints with the smallest possible ξ_i:

$$\xi_i = \max\left(0, 1 - y_i\left(w^T x_i + b\right)\right),\ \ i = 1 \ldots N$$

Hinge loss

- This function is known as the *hinge loss*. Consider, for example, a point x belonging to class y = 1:

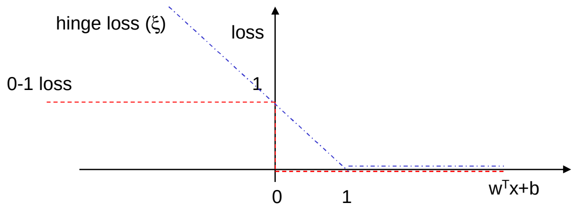
$$\xi = \max\left(0, 1 - \left(w^T x + b\right)\right)$$



- It is visibly a convex function of the score (and of w,b)

The hinge loss and the classification error

- The *empirical error* (i.e., the classification error on the training set) can be quantified in terms of the 0-1 loss, $\Delta(y_{true}, y_{predicted})$
- We can compare the 0-1 loss with the hinge loss: the latter is an upper bound! (and convex)



- The 0-1 loss is not a convex function of the score (nor it is of w,b)

 37

SVM objective rephrased

- The SVM objective is:

$$w^*, b^* = \underset{w,b,\xi_{1:N}}{\arg\min} \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \right)$$

$$s.t. \quad y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \qquad \xi_i \geq 0,\, i = 1 \ldots N$$

  – the second term is an upper bound of the empirical error (or loss, or risk)

  – the first term is a regulariser, like with regularised likelihood

→ SVM can be referred to as a "minimum regularised empirical risk" classifier

 38

Convexity of the objective

- We have seen that each ξ_i is a convex function of the score, $w^T x + b$
- The score is a linear function (both convex and concave) of w,b. Therefore, each ξ_i , is a convex function of w,b
- The sum of the ξ_i, $\Sigma_{i=1..N}\, \xi_i$, is also convex
- $||w||^2$ is obviously convex
- The overall SVM objective, $\frac{1}{2}\, ||w||^2 + C\, \Sigma_{i=1..N}\, \xi_i$, is a positive combination of convex terms, therefore convex overall. A global optimum exists

Dual for soft-margin SVM

- The dual problem for the soft-margin SVM does not contain the slack variables:

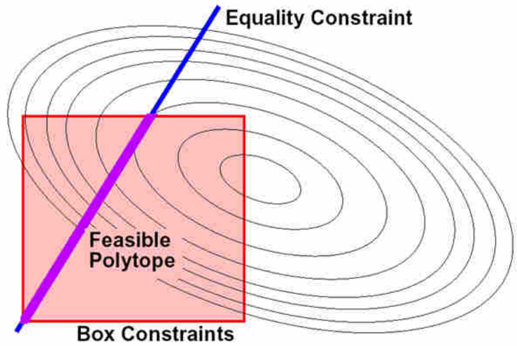
$$\arg\max_{\alpha_{1:N}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$C \geq \alpha_i \geq 0, \quad i = 1 \ldots N$$ *("box" constraints)*

Dual for soft-margin SVM

- The feasible polytope for the dual problem (2D illustration):



courtesy of Léon Bottou

Multi-class SVM as combination of binary classifiers

- The original formulation of the SVM is as a binary classifier. Multi-class classification over K classes can be obtained by combining multiple, binary SVMs
- The main techniques are known as:
  – "one vs all" (aka "one vs rest")
  – "one vs one" (aka "all-pairs")
  – DAGSVM (directed acyclic graph SVM)
- These techniques can be used to combine any binary classifier, not just the SVM

Multi-class SVM as combination of binary classifiers

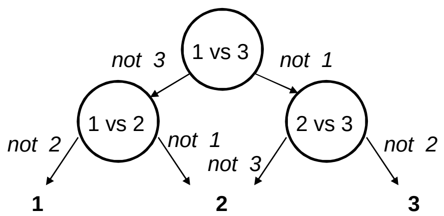
- In *one-vs-all*, one trains K binary classifiers, of the type class 1 versus not class 1, … class K versus not class K
- To classify a new sample, x, all classifiers are applied. The classification is not necessarily consistent: the sample may be assigned to more than one class, or none. Noting the class as y, a common classification rule is then:

$$y^* = \arg\max_{y=1\ldots K} \left(w_y x + b_y \right)$$

- One-vs-all classification may be seen as challenging since a single hyperplane must separate class k from other K-1 distributions. However, Rifkin and Klautau, JMLR 2004, have reported a strong experimental performance

Multi-class SVM as combination of binary classifiers

- In *one-vs-one*, one trains K(K-1)/2 binary classifiers, of the type class 1 vs class 2, class 1 vs class 3, … class K-1 versus class K: all pairs
- To classify a new sample, all classifiers are applied. The class that gets the highest number of votes is selected
- In *DAGSVM*, one trains the same classifiers, but uses a minimal number for classification:



“True” multi-class SVM

- Multi-class SVM as a single machine was proposed by [Weston and Watkins 1999]. An alternative formulation was given by [Crammer and Singer 2001]:

$$w^*, b^* = \arg\min_{w,b,\xi_{1:N}} \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \right)$$
$$s.t. \quad w_{y_i}^T x_i + b_{y_i} - \left(w_k^T x_i + b_k\right) \geq 1 - \xi_i$$
$$\forall k \neq y_i, \ \xi_i \geq 0, \ i = 1 \ldots N$$

where w is the concatenation of individual class’ models, $\mathbf{w}^T = [\mathbf{w}_1^T \ldots \mathbf{w}_K^T]$, and $\mathbf{b} = [\mathbf{b}_1 \ldots \mathbf{b}_K]$

Multi-class SVM

$$w^*, b^* = \arg\min_{w,b,\xi_{1:N}} \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \right)$$
$$s.t. \quad w_{y_i}^T x_i + b_{y_i} - \left(w_k^T x_i + b_k\right) \geq 1 - \xi_i$$
$$\forall k \neq y_i, \ \xi_i \geq 0, \ i = 1 \ldots N$$

- There are N (K-1) constraints: for every sample x_i, there is a constraint between its ground-truth class, y_i, and every other class
- Like in the binary case, there is only one slack variable per sample, set by the “most violating” constraint
- The models, w_k, b_k, are **absolute**, not differential anymore. In fact, in the case of K = 2, we have w = ($w_1 - w_2$) and b = ($b_1 - b_2$) where w, b are the parameters of the binary SVM

Inference in multi-class SVM

- Once training is completed, classification is obtained as:

$$y^* = \arg\max_{y=1\ldots K}\left(w_y x + b_y\right)$$

- This time, the models are properly comparable since they have been jointly trained and the classification rule, exact (it is equivalent to a MAP rule with a probabilistic classifier)

Analogy with probabilistic models

- Let us consider a class-posterior probability from the exponential family (like in CRF), $p(y|x) = \exp(w_y^T x + b_y)/Z$
- For a given sample, x_i, **the ratio of the posterior probabilities** for the ground-truth class and another class is:

$$\frac{p(y_i \mid x_i)}{p(k \mid x_i)} = \frac{e^{w_{y_i}^T x_i + b_{y_i}}}{e^{w_k^T x_i + b_k}}$$

- If we apply the natural logarithm to the above, we obtain the difference in scores requested by the SVM constraints:

$$w_{y_i}^T x_i + b_{y_i} - \left(w_k^T x_i + b_k\right)$$

Analogy with probabilistic models

- Even more important, a comparison of the objectives:
- SVM: minimum regularised empirical risk (upper bound):

$$\min_{w,b,\xi_{1:N}} \left(\|w\|^2 + C\sum_{i=1}^{N} \xi_i \right) \qquad C > 0$$

- Discriminative probabilistic training: minimum regularised negative conditional log-likelihood:

$$\min_{w,b} \left(\|w\|^2 - C\sum_{i=1}^{N} \ln p(y_i \mid x_i, w, b) \right) \qquad C > 0$$

rephrased to illustrate the analogy. The second term is also called *logistic loss*

Margin-rescaled multi-class SVM

- In margin-rescaled multi-class SVM [Tsochantaridis et al. 2005], the constraints are changed as follows:

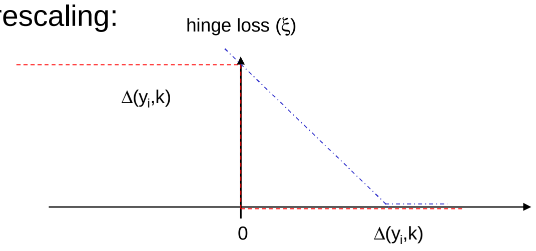
$$w_{y_i}^T x_i + b_{y_i} - \left(w_k^T x_i + b_k \right) \geq \boxed{\Delta(y_i, k) - \xi_i}$$

where $\Delta(y_i,k)$ is a loss function other than the zero-one loss

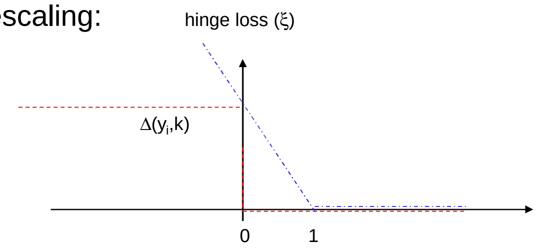
- The principle of margin rescaling is to create larger margins with the classes of most undesirable misclassification
- For the zero-one loss, it goes back to the standard multi-class
- A *slack rescaling,* $1 - \xi_i/\Delta(y_i,k)$, is also possible

Loss rescaling

- Margin rescaling:

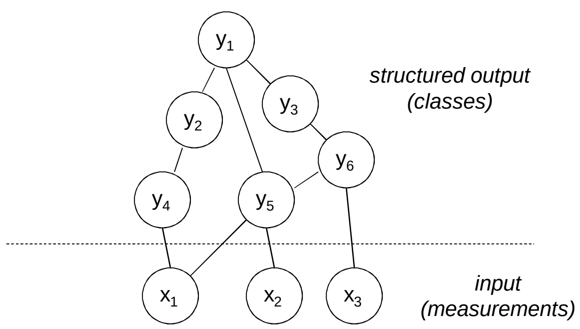


- Slack rescaling:



Structural SVM

- Structural SVM (originally called *structured-output SVM* and, by some, *structured SVM*) is a classification by SVM where the classes enjoy some graphical structure:



Structural SVM

- **Structural classification** (aka structured prediction) provides the labels of all variables, $y = \{y_1, y_2, \dots\}$, jointly based on a set of measurements, $x = \{x_1, x_2, \dots\}$
- The number of the possible class assignments is too large ($K_1 * K_2 * \dots$) to allow for a standard multi-class implementation, but the structure in the predicted labels can be leveraged
- The common approach is to find a sub-set of the constraints ensuring a sufficiently accurate solution. This approach is known as the *cutting-plane method*

Augmented inference

- Strategy: identifying the most violated constraint for each sample without having to try all possible class assignment
- In the case of margin rescaling, we will see that this equates to finding:

$$y_i^* = \arg\max_{y=1\dots K}\left(score(x_i, y) + \Delta(y_i, y)\right)$$

  the above is known as *augmented inference*

- If the violation is > current $\xi_i + \varepsilon$, add the constraint to the working set for the sample, S_i (Tsochantaridis et al. 2005)

Generalised linear models

- Depending on the structure over the classes and the edges between classes and measurements, the score function can take significantly different forms. It is therefore convenient to introduce a common notation: we call **w** the vector of all parameters, and **Ψ(x,y)** an arrangement of the classes and the measurements such that the score is simply given by $\mathbf{w}^T \, \Psi(\mathbf{x},\mathbf{y})$
- This notation is sometimes referred to as *generalised linear model*. It can be used for multi-class, HMM, HCRF, switching HMMs and any other structured output

An example: multi-class

- Example with multi-class SVM:

w	w_1	b_1	…	w_k	b_k	…	w_K	b_K
Ψ(x, y = k)	0 0 0 …		… 0 0 0	x	1	0 0 0 …		… 0 0 0

- In this way, $w^T \, \Psi(x,y) = w_k^T \, x + b_k$!

Most violated constraint

- At every iteration of the solver, find the most violated constraint for each sample (generalised linear model, margin re-scaling):

$$
\begin{aligned}
& w^T\Psi(x_i, y_i) - w^T\Psi(x_i, y) \geq \Delta(y_i, y) - \xi_i \quad \forall y \\
& \rightarrow \xi_i \geq -w^T\Psi(x_i, y_i) + w^T\Psi(x_i, y) + \Delta(y_i, y) \quad \forall y \\
& \rightarrow \xi_i = \max_y\left(-w^T\Psi(x_i, y_i) + w^T\Psi(x_i, y) + \Delta(y_i, y)\right), \\
& \quad y_i^* = \arg\max_y\left(w^T\Psi(x_i, y) + \Delta(y_i, y)\right)
\end{aligned}
$$

- The value of ξ_i is set by the most violating labeling, y_i*. We choose to satisfy the inequality with "=" since there is no reason to exceed it. If ξ_i > previous ξ_i + ε for that sample, add y_i* to the sample's working set, S_i

Structural SVM: objective (margin re-scaling)

$$
w^* = \arg\min_{w, \xi_{1:N}}\left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i\right)
$$

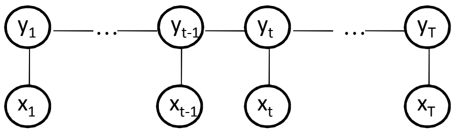
$$
s.t.
$$

$$
w^T\Psi(x_i, y_i) - w^T\Psi(x_i, y) \geq \Delta(y_i, y) - \xi_i \quad \forall i, \forall y \in S_i
$$

- S_i is the working set for the i-th sample and grows as explained
- The number of constraints to achieve ε–accuracy is $O(1/\varepsilon^2)$ (Tsochantaridis et al. 2005)

An example: sequential labelling

- We can approach the problem of sequential labelling (aka tagging) with a structure equivalent to the HMM. NB: the states are **supervised** (i.e., known) during training:



- Let us recall the generative model and apply the logarithm:

$$p(x,y) = p(y_1)\prod_{t=2}^{T} p(y_t \mid y_{t-1})\prod_{t=1}^{T} p(x_t \mid y_t) \rightarrow$$

$$\ln p(x,y) = \ln p(y_1) + \sum_{t=2}^{T} \ln p(y_t \mid y_{t-1}) + \sum_{t=1}^{T} \ln p(x_t \mid y_t)$$

 59

An example: sequential labelling

- Let us ignore $p(x_1)$ for simplicity and assume all distributions are from the exponential family:

$$\ln p(x,y) = \sum_{t=2}^{T} \ln p(y_t \mid y_{t-1}) + \sum_{t=1}^{T} \ln p(x_t \mid y_t) \rightarrow$$

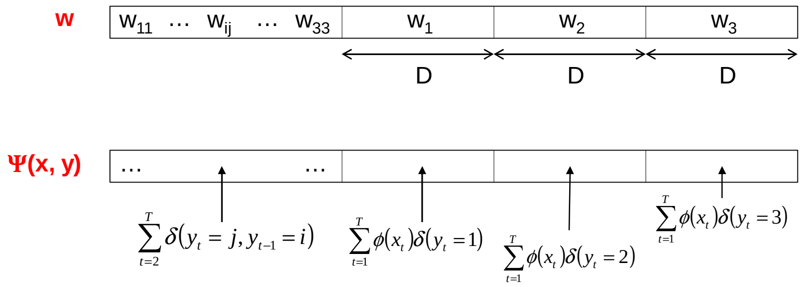
$$w^T \Psi(x,y) = \sum_{t=2}^{T} w_{ij}\delta(y_t = j, y_{t-1} = i) + \sum_{t=1}^{T} w_i^T \phi(x_t)\delta(y_t = i)$$

- All the scores become products of weights and functions:
  - the $\delta()$ function is 1 when its argument is true, 0 otherwise
  - the $\phi(x)$ function consist of features computed from x. For instance, $\phi(x) = [x_1, x_2, \ldots x_D, x_1^2, x_2^2, \ldots x_D^2]$ containing all the dimensions of x and their squares is comparable to a Gaussian emission model with diagonal covariance

 60

Example

- For an HMM with K = 3 states and D-dimensional observations:



- It is easy to verify that $w^T \Psi(x,y)$ returns the desired score

An example: sequential labelling

- Once trained, the Viterbi algorithm can be used for classification:

$$y^* = \arg\max_y w^T \Psi(x, y)$$

- Training requires finding the most violated constraint, i.e.:

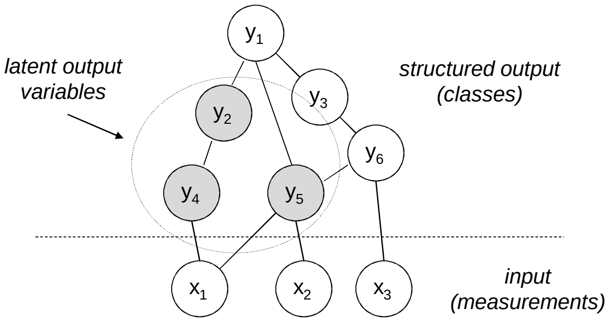
$$y^* = \arg\max_y \left(w^T \Psi(x_i, y) + \Delta(y_i, y) \right)$$

- This problem depends on the choice for $\Delta(y_i, y)$. A plausible choice is the Hamming distance which can be easily computed with Viterbi by augmenting the model with a term of the type:

$$\sum_{t=1}^{T} 1 \cdot \delta\left(y_t^i \neq y_t\right)$$

Latent structural SVM

- In latent structural SVM, some of the output variables are unsupervised, i.e. we do not know their value during training (nor are we interested in their value at run-time):



Latent structural SVM

- We note the output variables collectively as {y,h}. With this notation, the constraints would become:

$$w^T\Psi(x_i, y_i, h_i) - w^T\Psi(x_i, y, h) \geq \Delta((y_i, h_i),(y, h)) - \xi_i$$

- The problem is that h_i is unknown. The strategy of [Yu & Joachims 2009] is to alternate a step of inference for latent variables h_i with a step of SVM optimisation, until convergence

Latent structural SVM

- Step 1: SVM optimisation using the inferred h_i:

$$w^* = \arg\min_w \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \right)$$

$$s.t. \quad w^T \Psi(y_i, h_i^*, x_i) - w^T \Psi(y, h, x_i) \geq \Delta(y_i, y, h) - \xi_i$$
$$\forall y, h, \quad i = 1 \ldots N$$

- Step 2: infer h_i again:

$$h_i^* = \arg\max_h w^{*T} \; \Psi(y_i, h, x_i)$$

- Alternate between step 1 and step 2 until convergence
- h_i* needs to be removed from the loss function to ensure convergence

Latent structural SVM

- Training latent structural SVM is not a convex problem: the solution is a local minimum that depends on the initialisation. This is in common with any models containing latent variables due to the possible, consequent multi-modality of the objective function
- Training can be initialised either by arbitrary h_i* in step 1 or an arbitrary parameter vector, w, in step 2
- Unsupervised and semi-supervised training of SVM is still a current research problem

Inference

- The inference is as usual, with all the output variables, {y,h}, inferred and then h discarded:

$$y^*, h^* = \arg\max_{y,h} w^T \Psi(x, y, h)$$

Thorsten Joachims' *Latent SVM^struct*

- `void classify_struct_example(PATTERN x, LABEL *y, LATENT_VAR *h, STRUCTMODEL *sm, STRUCT_LEARN_PARM *sparm)`

$$y^*, h^* = \arg\max_{y,h} w^T \Psi(x, y, h)$$

- `LATENT_VAR infer_latent_variables(PATTERN x, LABEL y, STRUCTMODEL *sm, STRUCT_LEARN_PARM *sparm)`

$$h^* = \arg\max_{h} w^T \Psi(x, y, h)$$

- `void find_most_violated_constraint_marginrescaling(PATTERN x, LABEL y, LABEL *ybar, LATENT_VAR *hbar, STRUCTMODEL *sm, STRUCT_LEARN_PARM *sparm)`

$$y^*, h^* = \arg\max_{y,h} \left(w^T \Psi(x_i, y, h) + \Delta(y_i, y) \right)$$

Main references for these notes

- Andrew Ng, Support Vector Machines, CS229 Lecture notes, Stanford University
- Christopher Bishop, Pattern Recognition and Machine Learning, Chapter 7, Springer, 2006
- Crammer and Singer, On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines, JMLR, 2001
- Tsochantaridis, Joachims, Hofmann, Altun, Large margin methods for structured and interdependent output variables, JMLR, 2005
- Chun-Nam John Yu, Thorsten Joachims, Learning structural SVMs with latent variables, ICML 2009
- Sebastian Nowozin, Christoph H. Lampert: Structured Learning and Prediction in Computer Vision. Foundations and Trends in Computer Graphics and Vision 6(3-4): 185-365 (2011) http://pub.ist.ac.at/~chl/papers/nowozin-fnt2011.pdf

