IAML: Support Vector Machines II

Nigel Goddard
School of Informatics

Semester 1

In SMV I

We saw:

- Max margin trick
- Geometry of the margin and how to compute it
- Finding the max margin hyperplane using a constrained optimization problem
- Max margin = Min norm

This Time

- Non separable data
- The kernel trick

The SVM optimization problem

- Last time: the max margin weights can be computed by solving a constrained optimization problem

$$\min_{\mathbf{w}} \ ||\mathbf{w}||^2$$
$$\text{s.t. } \ y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq +1 \qquad \text{for all } i$$

- Many algorithms have been proposed to solve this. One of the earliest efficient algorithms is called SMO [Platt, 1998]. This is outside the scope of the course, but it does explain the name of the SVM method in Weka.

Finding the optimum

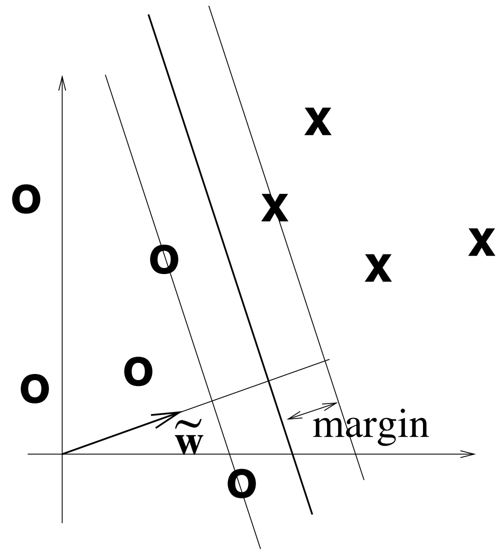
- If you go through some advanced maths (Lagrange multipliers, etc.), it turns out that you can show something remarkable. Optimal parameters look like

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- Furthermore, solution is sparse. Optimal hyperplane is determined by just a few examples: call these *support vectors*

Why a solution of this form?

If you move the points not on the marginal hyperplanes, solution doesn't change - therefore those points don't matter.



Finding the optimum

- If you go through some advanced maths (Lagrange multipliers, etc.), it turns out that you can show something remarkable. Optimal parameters look like

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- Furthermore, solution is sparse. Optimal hyperplane is determined by just a few examples: call these *support vectors*
- $\alpha_i = 0$ for non-support patterns
- Optimization problem to find α_i has no local minima (like logistic regression)
- Prediction on new data point $\mathbf{x}$

$$\begin{aligned} f(\mathbf{x}) &= \mathrm{sign}((\mathbf{w}^\top \mathbf{x}) + w_0) \\ &= \mathrm{sign}(\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + w_0) \end{aligned}$$

Non-separable training sets

- If data set is not linearly separable, the optimization problem that we have given has *no solution*.

$$\begin{aligned} &\min_{\mathbf{w}} \ ||\mathbf{w}||^2 \\ &\text{s.t.} \ \ y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq +1 \qquad \text{for all } i \end{aligned}$$

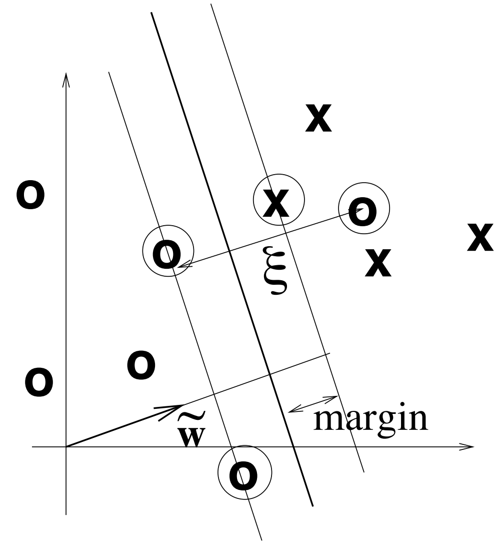
- Why?

Non-separable training sets

- If data set is not linearly separable, the optimization problem that we have given has *no solution*.

$$\min_{\mathbf{w}} \ ||\mathbf{w}||^2$$
$$\text{s.t. } y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq +1 \qquad \text{for all } i$$

- Why?
- Solution: Don't require that we classify all points correctly. Allow the algorithm to choose to ignore some of the points.
- This is obviously dangerous (why not ignore all of them?) so we need to give it a penalty for doing so.



Slack

- Solution: Add a "slack" variable $\xi_i \geq 0$ for each training example.
- If the slack variable is high, we get to relax the constraint, but we pay a price
- New optimization problem is to minimize

$$||\mathbf{w}||^2 + C(\sum_{i=1}^{n} \xi_i^k)$$

subject to the constraints

$$\mathbf{w}^\top \mathbf{x}_i + w_0 \geq 1 - \xi_i \quad \text{for } y_i = +1$$
$$\mathbf{w}^\top \mathbf{x}_i + w_0 \leq -1 + \xi_i \quad \text{for } y_i = -1$$

- Usually set $k = 1$. C is a trade-off parameter. Large C gives a large penalty to errors.
- Solution has same form, but support vectors also include all where $\xi_i \neq 0$. Why?

Think about ridge regression again

- Our max margin + slack optimization problem is to minimize:

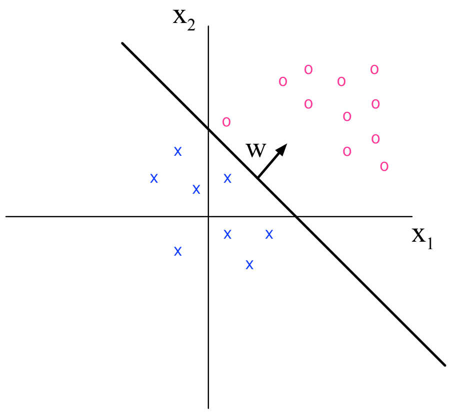
$$||\mathbf{w}||^2 + C(\sum_{i=1}^{n} \xi_i)^k$$

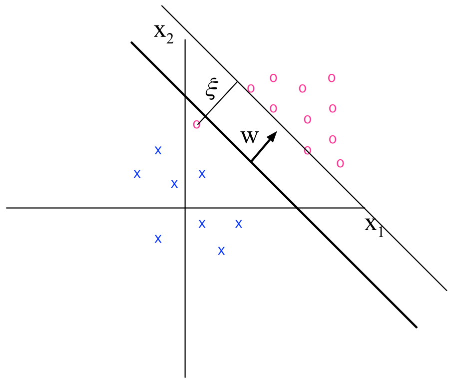
subject to the constraints

$$\mathbf{w}^\top \mathbf{x}_i + w_0 \geq 1 - \xi_i \quad \text{for } y_i = +1$$
$$\mathbf{w}^\top \mathbf{x}_i + w_0 \leq -1 + \xi_i \quad \text{for } y_i = -1$$

- This looks a even more like ridge regression than the non-slack problem:
  - $C(\sum_{i=1}^{n} \xi_i)^k$ measures how well we fit the data
  - $||\mathbf{w}||^2$ penalizes weight vectors with a large norm
- So C can be viewed as a regularization parameters, like λ in ridge regression or regularized logistic regression
- You're allowed to make this tradeoff even when the data set is separable!

Why you might want slack in a separable data set





Non-linear SVMs

- SVMs can be made nonlinear just like any other linear algorithm we've seen (i.e., using a basis expansion)
- But in an SVM, the basis expansion is implemented in a very special way, using something called a *kernel*
- The reason for this is that kernels can be faster to compute with if the expanded feature space is very high dimensional (even infinite)!
- This is a fairly advanced topic mathematically, so we will just go through a high-level version

Kernel

- A kernel is in some sense an alternate "API" for specifying to the classifier what your expanded feature space is.
- Up to now, we have always given the classifier a new set of training vectors $\phi(\mathbf{x}_i)$ for all i, e.g., just as a list of numbers. $\phi : \mathbb{R}^d \to \mathbb{R}^D$
- If D is large, this will be expensive; if D is infinite, this will be impossible

Non-linear SVMs

- Transform $\mathbf{x}$ to $\phi(\mathbf{x})$
- Linear algorithm depends only on $\mathbf{x}^\top \mathbf{x}_i$. Hence transformed algorithm depends only on $\phi(\mathbf{x})^\top \phi(\mathbf{x}_i)$
- Use a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ such that

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- (This is called the "kernel trick", and can be used with a wide variety of learning algorithms, not just max margin.)

Example of kernel

- Example 1: for 2-d input space

$$\phi(\mathbf{x}_i) = \begin{pmatrix} x_{i,1}^2 \\ \sqrt{2} x_{i,1} x_{i,2} \\ x_{i,2}^2 \end{pmatrix}$$

then

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j)^2$$

Kernels, dot products, and distance

- The Euclidean distance squared between two vectors can be computed using dot products

$$\begin{aligned} d(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T (\mathbf{x}_1 - \mathbf{x}_2) \\ &= \mathbf{x}_1^T \mathbf{x}_1 - 2\mathbf{x}_1^T \mathbf{x}_2 + \mathbf{x}_2^T \mathbf{x}_2 \end{aligned}$$

- Using a linear kernel $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$ we can rewrite this as

$$d(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_1, \mathbf{x}_1) - 2k(\mathbf{x}_1, \mathbf{x}_2) + k(\mathbf{x}_2, \mathbf{x}_2)$$

- Any kernel gives you an associated distance measure this way. Think of a kernel as an indirect way of specifying distances.

Support Vector Machine

- A **support vector machine** is a kernelized maximum margin classifier.
- For max margin remember that we had the magic property

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- This means we would predict the label of a test example $\mathbf{x}$ as

$$\hat{y} = \text{sign}[\mathbf{w}^T \mathbf{x} + w_0] = \text{sign}[\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0]$$

- Kernelizing this we get

$$\hat{y} = \text{sign}[\sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b]$$

Prediction on new example

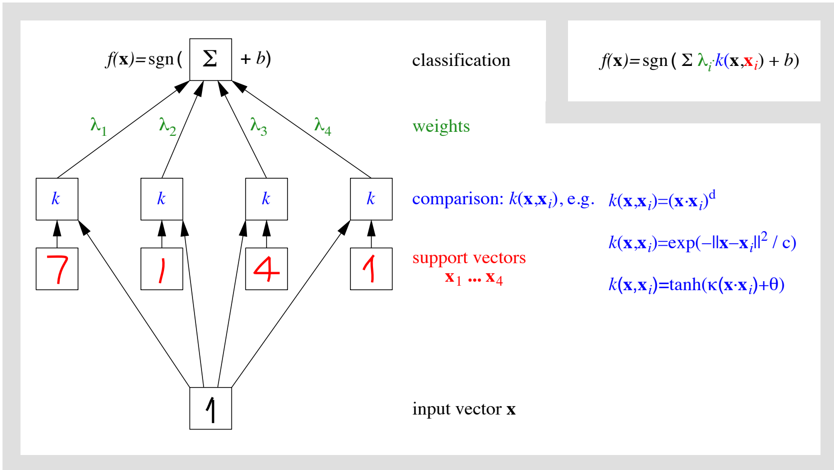


Figure Credit: Bernhard Schoelkopf

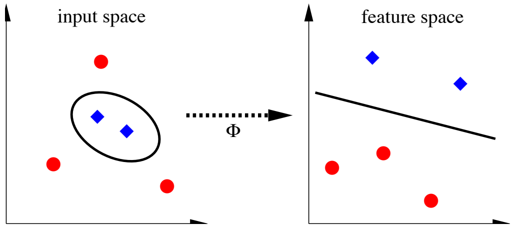


Figure Credit: Bernhard Schoelkopf

- Example 2

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp -||\mathbf{x}_i - \mathbf{x}_j||^2/\alpha^2$$

  In this case the dimension of ϕ is infinite. i.e., It can be shown that no ϕ that maps into a finite-dimensional space will give you this kernel.
- We can *never* calculate $\phi(\mathbf{x})$, but the algorithm only needs us to calculate k for different pairs of points.

Choosing ϕ, C

- There are theoretical results, but we will not cover them. (If you want to look them up, there are actually upper bounds on the generalization error: look for VC-dimension and structural risk minimization.)
- However, in practice cross-validation methods are commonly used

Example application

- US Postal Service digit data (7291 examples, 16×16 images). Three SVMs using polynomial, RBF and MLP-type kernels were used (see Schölkopf and Smola, *Learning with Kernels*, 2002 for details)
- Use almost the same ($\simeq$ 90%) small sets (4% of data base) of SVs
- All systems perform well ($\simeq$ 4% error)
- Many other applications, e.g.
  - Text categorization
  - Face detection
  - DNA analysis

Comparison with linear and logistic regression

- Underlying basic idea of linear prediction is the same, but error functions differ
- Logistic regression (non-sparse) vs SVM (“hinge loss”, sparse solution)
- Linear regression (squared error) vs ϵ-insensitive error
- Linear regression and logistic regression can be “kernelized” too

SVM summary

- SVMs are the combination of max-margin and the kernel trick
- Learn linear decision boundaries (like logistic regression, perceptrons)
  - Pick hyperplane that maximizes margin
  - Use slack variables to deal with non-separable data
  - Optimal hyperplane can be written in terms of support patterns
- Transform to higher-dimensional space using kernel functions
- Good empirical results on many problems
- Appears to avoid overfitting in high dimensional spaces (cf regularization)
- Sorry for all the maths!