Separating hyperplanes

Two class problem. Data: \( \{ x_i, y_i \}, \quad i=1..n, \quad y_i \in \{-1,1\}, \quad x_i \in \mathbb{R}^d \)
Assume two classes are linearly separable.
Infinite number of separating hyperplanes.
Many 'thin' planes, few 'thick' planes. One 'thickest' plane.
Points on a separating hyperplane satisfy \( x \cdot w + b = 0 \)
\( w \) is normal to hyperplane
\( \frac{|b|}{||w||} \) is perpendicular distance from hyperplane to origin
\( d_+, d_- \) distance of nearest points to hyperplane.
Separating margin is \( d_+ + d_- \)
SVM seeks to maximise this
Separable problem

Separating hyperplanes

Suppose data satisfy following (i.e. set scale for $w, b$)

\begin{align*}
   x_i \cdot w + b & \geq +1 \quad \text{for} \quad y_i = +1 \\
   x_i \cdot w + b & \leq -1 \quad \text{for} \quad y_i = -1
\end{align*}

Equality satisfied for point(s) nearest boundary (on the margin).

First case is $H_1$. Distance from this to origin is

$$
\|x\| = \frac{|1-b|}{\|w\|} \quad \text{so} \quad d_+ = \frac{1}{\|w\|} \quad \text{and similar for second case, } H_2
$$

$H_1, H_2$ and separating hyperplane are parallel.

Thus margin size is $d_+ + d_- = \frac{2}{\|w\|}$

Therefore, optimal separating hyperplane is found by minimizing $\|w\|$ subject to the constraint

$$
y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i \quad \text{(combined inequality from above)}
$$

Note $\|w\| = \sqrt{w \cdot w}$. Can equally well (and more easily) minimize $\|w\|^2$. 
Lagrangian formulation

Minimize $||w||^2$ subject to $y_i(x_i \cdot w + b) - 1 \geq 0$

Introduce Lagrange multipliers $\alpha_i \geq 0 \ \forall i$

$$L_P = \frac{1}{2} ||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i(x_i \cdot w + b) - 1)$$

Minimize $L_P$ w.r.t $w, b$ and maximize w.r.t $\alpha_i$.

This is a convex quadratic programming problem.

$$\frac{\partial L}{\partial w} = \left( w - \sum_{i=1}^{n} \alpha_i y_i x_i \right) = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

Thus we see that the plane is a linear combination of the training vectors

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


---

Lagrangian dual

$$L_P = \frac{1}{2} ||w||^2 - \sum_{i=1}^{n} \alpha_i y_i(x_i \cdot w + b) + \sum_{i=1}^{n} \alpha_i$$

Substitute for $w$ and use $\sum_i \alpha_i y_i = 0$

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

The (Wolfe) 'dual' problem is to maximize $L_D$ w.r.t the $\alpha_i$ subject to

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

There is a Lagrange multiplier for every training point.

Points with $\alpha_i > 0$ are called support vectors and lie on $H_1$ or $H_2$.

Points with $\alpha_i = 0$ lie beyond margin planes and are irrelevant to solution.

This comes from the so-called 'Karush-Kuhn-Tucker' conditions for constrained optimization.

One such condition is $\alpha_i |y_i(x_i \cdot w + b) - 1| = 0 \ \forall i$ $\implies$ $\alpha_i \neq 0$ only for points on plane

Applying the classifier

Let $s_i = x_i$ be the $N$ support vectors.
The boundary is $x \cdot w + b = 0$
The classification of a new point, $x_j$, is given by the sign of

$$f(x_j) = x \cdot w + b = \sum_{i=1}^{N} \alpha_i y_i s_i \cdot x_j + b$$

Non-separable case

Maximize margin size as before, but allow some points to be on wrong
side of margin. Thus modify constraints with slack variables, $\xi_i$

$$x_i \cdot w + b \geq 1 - \xi_i \quad \text{for} \quad y_i = +1$$

$$x_i \cdot w + b \leq -1 + \xi_i \quad \text{for} \quad y_i = -1 \quad \xi_i \geq 0$$

Errors only occur for $\xi_i > 1$ and $\sum_i \xi_i$ is (upper bound on) total error.

Assign a cost, $C$, to errors and now minimize $||w||^2 + C \sum_i \xi_i$

subject to $y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0$ and $\xi_i \geq 0 \ \forall \ i$

The Lagrangian (primal) for this is

$$L_p = \frac{1}{2} ||w||^2 - \sum_{i=1}^{n} \alpha_i [y_i(x_i \cdot w + b) - (1 - \xi_i)] + C \sum_i \xi_i - \sum_i \mu_i \xi_i$$

where the $\mu_i$ are the Lagrange multipliers for the slack variables.

$\mu_i \geq 0, \xi_i \geq 0 \ \forall \ i$
Non-separable case

Take derivatives w.r.t parameters as for separable case to find the dual formulation.

The dual Lagrangian (objective function) is the same as before

\[ L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \]

This we maximize w.r.t the \( \alpha_i \) but now subject to

\[ \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C \]

i.e. there is an upper limit on the \( \alpha_i \) which limits influence of any one point.
Non-separable case

Solution is otherwise the same as the separable case

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

but now the $\alpha_i$ have an upper bound, $C$.

Sum is over all support vectors, which again are those points with $\alpha_i > 0$.

These are over all support vectors, which again are those points with $\alpha_i > 0$.

$C$ controls the solution complexity, so generally

larger $C$ => more flexibility on solutions ($\alpha_i$) => less regularization

=> smaller error on training set (see p. 15 for a discussion)

\[ \xi_i = 0 \text{ if } \alpha_i < C \]

Note that the $\xi_i$ and $\mu_i$ do not appear in the solution.

Two 'Karush-Kuhn-Tucker' conditions in this case are $C - \alpha_i = \mu_i$ and $\mu_i \xi_i = 0$ $\forall i$.

Therefore $\xi_i = 0 \text{ if } \alpha_i < C$.

Nonlinear functions via kernels

Generate nonlinear combinations of inputs and fit model to this higher dimensional space: $\mathbf{x} \rightarrow \Phi(\mathbf{x})$, $\mathbb{R}^d \rightarrow H$.

$\mathbf{x}$ only appear in algorithm in dot products, $\mathbf{x}_i \cdot \mathbf{x}_j$.

Thus if we could use a kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x})$ we would only need to know $K$ and never need to explicitly know $\Phi$ e.g.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp \left( -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right)$$

$H$ is infinite dimensional (think series expansion) so $\Phi$ an infinite function.

To apply the classifier we did

$$f(\mathbf{x}_j) = \sum_{i=1}^{N_s} \alpha_i y_i \mathbf{s}_i \cdot \mathbf{x}_j + b$$

now insert kernel

$$f(\mathbf{x}_j) = \sum_{i=1}^{N_s} \alpha_i y_i \Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{x}_j) + b = \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}_j) + b$$
Nonlinear functions via kernels

\[
K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \quad \text{RBF kernel}
\]
\[
K(x_i, x_j) = (x_i \cdot x_j + 1)^p \quad \text{degree } p \text{ polynomial kernel}
\]
\[
K(x_i, x_j) = \tanh(\kappa x_i \cdot x_j - \delta) \quad \text{sigmoidal kernel}
\]

Cf. neural network: in RBF case, number of centers \(N_s\), centres \(s_i\) and weights \(\alpha_i\) found automatically by the SVN.

Support Vector Machines

- for nonlinear, multidimensional two class problems
- *implicit* projection into a higher (possibly infinite) dim. space
  - where we hope to get closer to linear separation
- free parameters \(\alpha_i\) (nonzero only for the support vectors)
  - problem is a convex linear programming problem
  - so the solution is unique (still needs numerical methods)
- solution is a combination of the support vectors (only)
  - solution is sparse in these, so suitable for high-dim. problems with limited training data (does not have scaling problem like an MLP)
- two control parameters \(C\) and \(\gamma = 1/\sigma^2\)
  - \(C\) controls regularization
  - Generally: larger \(C\) \(\Rightarrow\) error term dominates \(\Rightarrow\) less regularization
  - \(1/\gamma\) is a correlation/relevance length scale in the data space

\[
E = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N_s} \xi_i
\]
The margin, cost and no. of SVs in the non-separable case

We are still trying to maximize the margin width, even though the margin is no longer defined as the region void of data points (as in the separable case). Rather, the margin is defined as the region which contains all points relevant to (i.e. included in) the solution, that is, the support vectors, even if they are on the correct side of the separating hyperplane. However, support vectors only contribute to the errors (i.e. have $\xi>1$) if they are on the wrong side of the separating hyperplane. We are maximizing the margin width with a cost penalty. If we increase the cost, $C$, then errors incur a higher penalty and so tend to make the margin smaller. If the separating hyperplane were fixed this would generally reduce the number of support vectors. But of course it is not fixed (we change $C$ in order to find a different solution!), so the support vectors, and their number, will change, and the new margin is not necessarily narrower. Thus the number of support vectors and size of the margin in the solution do not have a monotonic dependence on $C$ (although they might for some problems). Generally, a larger cost produces a more flexible solution (as we are encouraging fewer error) and thus a smaller error on the training data, and we would normally associate a more complex solution with more support vectors. But this is depends very much on the specific problem, i.e. how well the data can be separated. Moreover, there is no guarantee that a larger cost produces a better solution.

SVM application: star-galaxy separation with PS1

- 4 noise-free simulated colours
- package libsvm
  - C++, Java
  - R import in svm{e1071}
- see R scripts for example and syntax
SVM application

For
\[ i-z = 0.2 \]
\[ z-y = 0.2 \]

Multiclass problems

- with K classes, either solve
  - K models, each of type 1 class vs. combined K-1 classes, or
  - \( K(K-1)/2 \) models, each with 1 class vs. 1 class (“pairwise coupling”)
Summary

- SVMs operate on principle of separating hyperplanes
  - maximize margin
  - only data points near margin are relevant (the support vectors)
- Nonlinearity via kernels
  - possible because training data appear only as dot products
  - project data into a higher dimensional space where more separation is possible
  - projection is only implicit (big computational saving)
- Unique solution (convex objective function)
  - optimization via the Lagrangain dual
- Various extensions
  - multiclass case. No natural probabilistic interpretation, but we can remap outputs
  - regression