Support Vector Machine recap

As it was described on the lecture, the goal of support vector machines (SVMs) is to keep the same linear separator model we have been using since the beginning, but try to make it more robust by placing the separator where it has the widest margin on either one of its sides: that is, we want the *closest* data points to be as *far* away from the separator as possible. The intuition is that if we we make a mistake, and the learnt separator is a little bit off, then we still have some margin until it actually counts[1].

Again, let our model output $w^T x + b$ for some input x, and we decide based on the sign of this output. Furthermore, let's constrain w, so that for all positive samples x^+, $w^T x^+ + b \geq 1$, and for all negative samples x^-, $w^T x^- + b \leq -1$. If we represent the true labels by $y \in \{-1; 1\}$, then these two can be combined: $y(w^T x + b) \geq 1$. Call the left hand side of this inequality the *score*. We would like to go one step further, and state that the closest datapoints (having the smallest score) should have score 1. This basically constrains the size of w, but not its direction.

Now with some careful drawing of vectors and calculations, we can calculate the width of the margin to be $\frac{2}{\|w\|_2}$. Since we want it to be as large as possible, we have a goal:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|_2^2 \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 \; i = 1, \ldots, N. \end{aligned} \tag{1}$$

(There were some minor equivalent transformations in the objective formulation, don't worry about it.)

Equation 1 is the one which is nice because it has all this geometric interpretation behind it, but it's not the one we are going to use. With some algebraic and other derivations that we won't worry about for now[2], we can change it into an equivalent problem formulation (both of them are quadratic programming tasks):

$$\begin{aligned} \text{maximize w.r.t } \alpha \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j Q_{ij} \\ & \text{where } Q_{il} = y_i y_j x_i^T x_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \; i = 1, \ldots, N \\ & \sum_{i=1}^{N} \alpha_i y_i = 0, \end{aligned} \tag{2}$$

where C is some hyperparameter. The new variables α_i for optimisation will determine which of the original samples will be supporting vectors: if α_i is (significantly) nonzero, those x_i samples will affect the location of the separator and they are the support vectors, as can be seen by the construction of the parameters:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{3}$$

and

$$b = y_B - x_B^T w, \tag{4}$$

[1] This is of course provided that the two classes are separable

[2] but would involve dual problems and Lagrangian multipliers, if you are interested

where B is the index of a sample that lies exactly on the margin's decision boundary.

OK, I confess, I omitted a very important difference between Equations eq. 1 and eq. 2: the former specifies a hard SVM, where we don't allow any point to be within the margin region; and eq. 2 describes a soft SVM, where we allow some points to be in the margin, or even on the wrong side of the decision boundary[3].

The parameter C is to govern this balance (it is a regularisation parameter): with a large enough C, the SVM becomes a hard SVM, with smaller C, it tries to form a decision boundary (and corresponding margin) with more support vectors, some of which may be inside the margin (which can be thought of as the product of noise).

The α_is can be in three distinct categories:

1. $\alpha_i = 0$, this means the ith sample is not a support vector, it is outside of the margin (on the correct side of the decision boundary) (in some corner cases, these samples can be on the margin's boundary as well)
2. $0 < \alpha_i < C$, this means the ith sample is exactly on the margin's boundary
3. $\alpha_i = C$, this means the ith sample is within the margin

To classify a new, unknown input u, you can calculate

$$\hat{y} = \text{sign}\left\{\sum_{i=1}^{N} \alpha_i y_i x_i^T u + b\right\} \tag{5}$$

Nonlinearity in SVMs

We can do the same nonlinear feature transformation trick we are slowly getting used to, except now we can go a step further: what you would need to do for a transformation $\phi(x)$ is to change every occurence of x and u in eqns. 2, 5 to $\phi(x)$ and $\phi(u)$, respectively.

But you don't actually need that, only the inner product of the two transformed vectors! This gives the idea to substitute ϕ with some symmetric kernel function $K(x_i, x_j)$ that takes two input vectors and outputs a sort of "distance" between the two. (Remember that the inner product is in some sense a distance: it is an unnormalised angle.)

This is the so called *kernel trick*: instead of specifying a feature transformation function ϕ, you specify a kernel function (which can be thought of as a composition of the feature transformation with an inner product). This has computational advantages, since kernel functions are usually specified in a way that are easier to calculate than the transformations (see the examples below). Taken to the extreme, there are some kernel functions (eg. the Gaussian kernel), for which the transformed feature space would be infinite dimensional.

Example kernels are: $K(a, b) = (a^T b + 1)^k$ (polynomial kernel) or $K(a, b) = \exp(-\|a - b\|/\sigma)$ (Gaussian kernel).

Note that inference (eq. 5) is now:

$$\hat{y} = \text{sign}\left\{\sum_{i=1}^{N} \alpha_i y_i K(x_i, u) + b\right\}, \tag{6}$$

with

$$b = y_B - \sum_{i=1}^{N} \alpha_i y_i K(x_i, x_B). \tag{7}$$

[3] Question: can a hard SVM classify a dataset that is not linearly separable?