Support vector machines

Cavan Reilly

October 16, 2019

Table of contents

K-nearest neighbor classification

Suppose we have a collection of measurements x_i and class labels y_i for $i = 1, \ldots, n$ and our goal is to predict the class label for a new collection of measurements x_{new}.

A conceptually simple way to acheive this is to measure the distance between x_{new} and x_i for all i and then predict the class for the new observation to be the most common class label among the K nearest neighbors (where $K = 1, 3, 5, 7, \ldots$).

The value of K can be determined by cross validation, however the method for computing distances now becomes critical.

K-nearest neighbor classification

Moreover rarely can a scientific understanding of the problem ever contribute to the decision regarding what distance to use (unlike the use of a likelihood).

Nonetheless, given a method for computing distances, the data x_i and y_i and K, the space in which the data reside (the data space) gets partitioned into regions where the prediction for new observations is for one class or the other.

Unless the data really do separate out well, this partition is likely to be rather complicated.

This makes prediction sensitive to the values of x_{new} and consequently the predictions would frequently be incorrect.

Support vector machines

Support vector machines are a class of algorithms which were designed for classification of 2 groups, however these methods have been extended to other settings (e.g. multigroup classification and regression).

The starting point for these methods is that if the data from the 2 groups can be separated then one should pick a boundary in the data space and use this boundary for classifying future observations.

If we pick a linear boundary, then there are still an infinte number of possible boundaries.

Support vector machines

In support vector machines the linear boundary is selected to maximize the margin betweem the 2 groups.

Only a subset of observations will be involved in determining exactly where this boundary lies: such observations are called support vectors.

In most applications the data from the different groups do not separate out well, so simply finding a linear boundary in the data space will not result in a good classification scheme.

In this case we could try to map our data from data space to another space, called *feature space* in such a way that a linear boundary separates the 2 groups in this other space.

Support vector machines

Recall multidimensional scaling: map to a lower dimensional space to see if the data separate in this lower dimensional space.

In the support vector machine literature we pursue the opposite: map to a higher dimensional space.

Of course, in practice one can always map to a higher dimensional space and separate the data from the 2 groups perfectly.

In fact one can always do this by mapping to a space with just 1 more dimension.

Support vector machines

So given a new observation, we would map this to the higher dimensional space and see which side of our boundary the new observation maps to.

We can do this by looking at the angle between where our point gets mapped and the boundary (well really the vector that is normal to the boundary).

It turns out that, if I define my boundary by maximizing the margin, the angle between where the new observation gets mapped and the vector that is normal to the boundary only depends on an *inner product* in the space where I map the data.

Moreover I can represent the boundary just using such inner products.

Support vector machines

So rather than specifying feature space and a mapping, I can just specify an inner product in feature space.

There are well established characterizations of functions of 2 vectors in arbitrary spaces that are valid as inner products (this characterization is known as Mercer's theorem).

Such functions are called *kernels* in this context, hence if we specify a kernel we are ready to do classification.

Support vector machines: slack variables and cost parameters

Although we can always seprate the data in a higher dimensional space, if we try to do so by specifying a kernel rather than a mapping, we may not acheive perfect separation in feature space.

In this event there will be no margin to maximize, hence we need to allow for errors in the classification process.

Formally we do this by introducing "slack variables", one for each observation in our data set, and we pay a certain cost depending on how much slack is in the data.

When we "train our classifier" we must select a value for this cost, and sometimes other parameters too.

Support vector machines: kernels

Users of SVMs spend a considerable amount of effort on "training" the classifier by trying to optimize the performance on the training data set, typically in terms of cross validated errors.

This may involve trying different kernels, different costs and perhaps parameters that enter the kernel.

Some common kernels are

1. the Gaussian radial basis kernel:
   $k(x_1, x_2) = \exp\{-c\,||x_1 - x_2||^2\}$
2. the simple kernel $k(x_1, x_2) = \sum_i x_{1i} x_{2i}$
3. the polynomial kernel $k(x_1, x_2) = (c_1 \sum_i x_{1i} x_{2i} + c_2)^{c_3}$.

Support vector machines: kernels

The first is useful when little is known about what might work well, the second for text processing and the third for classifying images.

There are a number of packages that implement support vector machines in R and there are publications that compare them in terms of generality, speed and quality.

We will use the kernlab package: it has good speed, a numerically stable algorithm and is more extensible than some of the others.

In particular, you can make up your own kernels in this package.