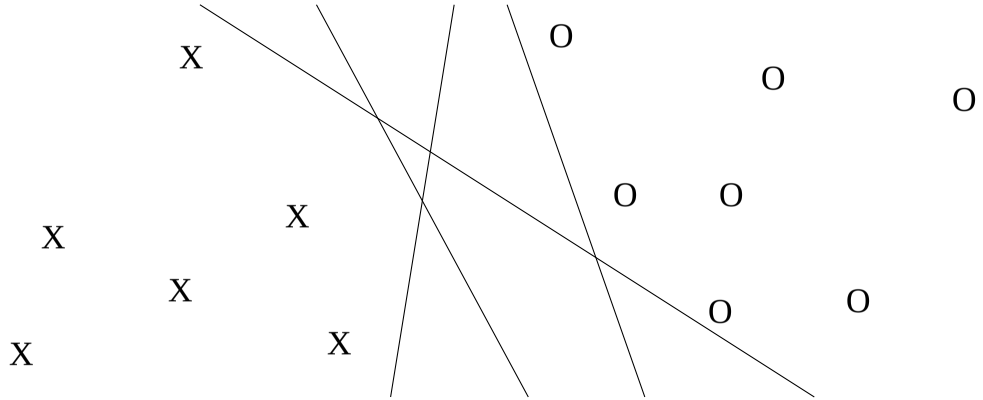
Separating Hyperplanes

The training set for a binary classification problem is *linearly separable* if there is a hyperplane in the input space that separates training cases of one class from those of the other class.

With p inputs, the hyperplane will be of dimension $p-1$. For example, with two inputs, the hyperplane is a line:



If the cases are linearly separable at all, there will be an infinite number of hyperplanes that separate them.

Equations for Separating Hyperplanes

A hyperplane of dimension $p-1$ consists of the points, x, satisfying some equation of the form

$$\beta_0 \; + \; \beta^T x \quad = \quad 0$$

Where β is a non-zero vector of length p, and β_0 is a scalar. Multiplying β_0 and β by any non-zero constant would give another equation for the same hyperplane.

Points on one side of the hyperplane will satisfy $\beta_0 + \beta^T x \; > \; 0$; for those on the other side, $\beta_0 + \beta^T x \; < \; 0$. Conventionally, the > 0 side corresponds to a response of $y = +1$, the < 0 side to $y = -1$.

A case with inputs x and response y is correctly classified if

$$y \, (\beta_0 + \beta^T x) \quad > \quad 0$$

We may be interested in *how far* a point, x, is from the hyperplane. This is

$$y \, (\beta_0 + \beta^T x) \; / \; ||\beta||$$

Where $||\beta||$ is the length of β. This distance is positive if the point is correctly classified, negative if not.

Finding Separating Hyperplanes

Suppose that the training data is linearly separable. How can/should we find a separating hyperplane?

Note that although all separating hyperplanes classify all the *training* cases correctly, they might perform very differently when classifying *test* cases.

We'd like to somehow choose a separating hyperplane that will do well on test cases.

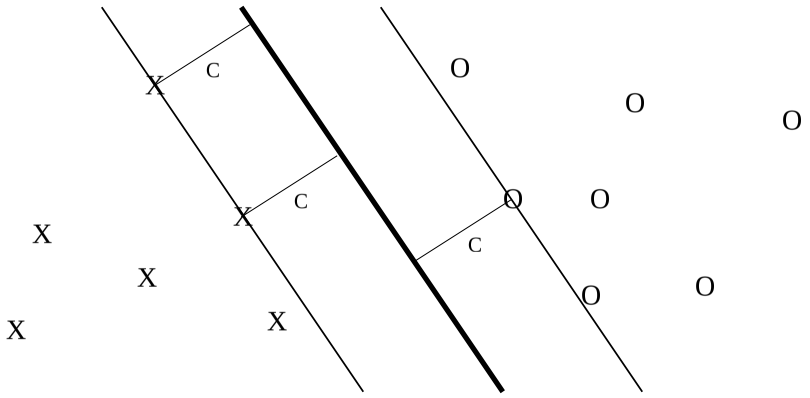
- The *perceptron learning* algorithm will find a separating hyperplane (eventually), using updates that look at just one training case at a time. This is an historically interesting method, but not used much in practice.
- We can use *logistic regression*. Maximizing the log likelihood minus $\lambda||\beta||^2$ for some very small λ will give us a well-defined hyperplane.
- We can find the hyperplane with *maximum margin*. This turns out to be equivalent to logistic regression with a very small penalty.

The last two methods produce a uniquely-determined separating hyperplane.

Maximum Margin Separating Hyperplanes

The "margin" of a separating hyperplane is the distance from the hyperplane to the closest training case. One idea is that performance on test cases will be good if we choose the separating hyperplane that has the largest margin.

Here's an example of a maximum margin separating hyperplane (in bold):



Two cases in the X class and one in the O class are a distance C from this separating hyperplane. There will always be at least one case in each class at the minimum distance (C) from the maximum margin separating hyperplane.

The training cases at the minimum distance are called the *support vectors*. They determine the solution — inputs for the other training cases could be changed slightly without changing the maximum margin separating hyperplane.

Finding the Maximum Margin Separating Hyperplane

We can formalize the problem as finding the values of β_0 and β that maximize C, subject to the constraint that

$$y_i \left(\beta_0 + \beta^T x_i\right) \;/\; ||\beta|| \;\geq\; C, \qquad \text{for } i = 1, \ldots, N$$

Here, i indexes training cases.

Since multiplying β_0 and β by any positive constant changes nothing, we can require that $||\beta|| = 1/C$. With this change, we can formulate the problem as

$$\text{Minimize } ||\beta||, \qquad \text{subject to } y_i \left(\beta_0 + \beta^T x_i\right) \;\geq\; 1, \text{ for } i = 1, \ldots, N$$

Minimizing $||\beta||$ is the same as minimizing $||\beta||^2$, so we can write this as

$$\text{Minimize } ||\beta||^2, \qquad \text{subject to } y_i \left(\beta_0 + \beta^T x_i\right) \;\geq\; 1, \text{ for } i = 1, \ldots, N$$

This is a *quadratic programming* problem — minimizing a quadratic function subject to linear constraints. There are (fairly) efficient algorithms for solving such problems.

More General "Support Vector Machines"

This method is sometimes called the *Support Vector Machine (SVM)*, because of the prominent role of the support vectors in the algorithm for solving the quadratic programming problem.

For most classification problems, the classes aren't linearly separable. There is a generalization of support vector machines that allows for this. There's also a version for regression.

Most interestingly, there is a generalization of SVMs that allows the inputs to be easily replaced by many functions of the inputs (eg, x, x^2, x^3, x^4) — even an infinite number of functions. This is useful when hyperplanes aren't the right way to separate classes.