Support Vector Machines

In this lecture. . .

- Dividing samples by hyperplanes
- Hard margins and soft margins
- Gradient descent for finding parameters

Introduction

Support vector machines are a supervised-learning classification technique. You have classified data represented by vectors of features. This method divides data according to which side of a hyperplane in feature space each point lies.

What Is It Used For?

Support Vector Machines are used for

- Classifying data based on a set of numerical features
- Example: Identify plant types from features of their petals
- Example: Estimate the risk of prostate cancer from MRI images
- Example: Find likely customers from subjects mentioned in their Twitter posts

The Support Vector Machine is another supervised-learning classification technique.

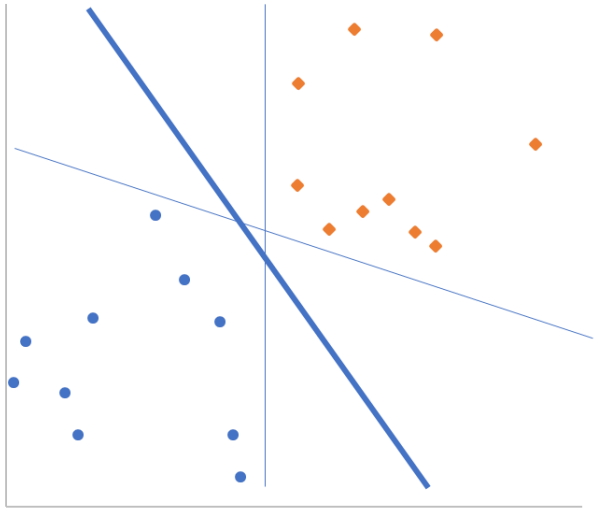
It's still implementable in Excel, at least as far as learning the technique is concerned if you don't have too much data.

As with all the methods we discuss if you want to use them professionally then you are going to have to code them up properly.

Hard Margins

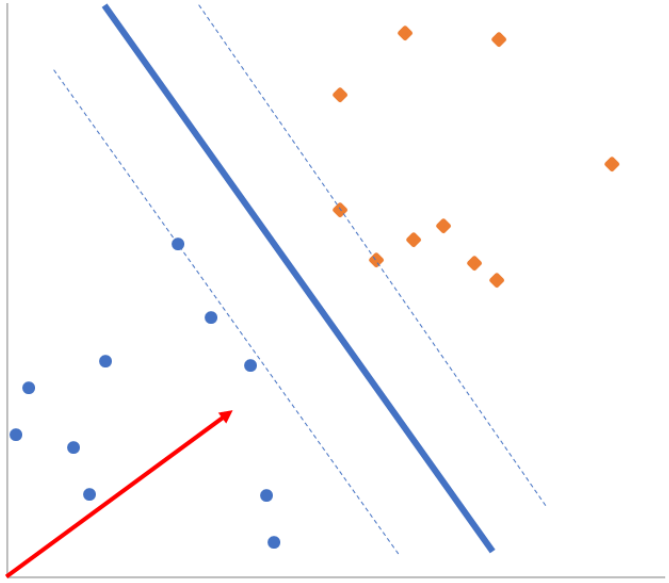
In the figure are shown sample vectors divided into two classes (the circles and the diamonds).

These classes can be divided using a straight line, they are linearly separable. This dividing straight line can then tell us to which class a new, unclassified, sample belongs, depending on whether it is on the circle or the diamond side of the line. But which is the *best* line?



One definition is that the best line is the one that has the largest margins between it and the samples. This is shown as the bold line in the figure.

These margins are shown below, where I've also shown the vector that is orthogonal to the hyperplane dividing the two classes. (I say 'hyperplane' because we will generally be in many dimensions.)



Our goal is to find the hyperplane such that the margins are furthest apart. The cases that define the margins are called the 'support vectors.'

You'll notice that this section is called 'Hard Margins.' This is a reference to there being a clear boundary between the two classes.

If one sample strays into the wrong region, a diamond over to the circle side, say, then we will need to do something slightly different from what immediately follows. And we'll look at that in the 'Soft Margins' and 'Kernel Trick' sections.

I am going to use $\boldsymbol{\theta}$ to denote the vector shown in the figure that is orthogonal to the hyperplane.

It therefore has M dimensions. Similarly $\mathbf{x}^{(n)}$ is the M-dimensional vector representing the n^{th} sample. I shall use $y^{(n)}$ as the class label, being $+1$ for all diamonds and -1 for all circles.

We want some symmetry to keep the mathematics elegant and that's why we use labels ± 1.

The hyperplane is all points such that

$$\boldsymbol{\theta}^T \mathbf{x} + \theta_0 = 0,$$

where we have to find the vector $\boldsymbol{\theta}$ and the scalar θ_0.

I want the margins to be represented by $\mathbf{x}^{\pm}$ where

$$\boldsymbol{\theta}^T \mathbf{x}^{\pm} + \theta_0 = \pm 1. \tag{1}$$

So

$$\boldsymbol{\theta}^T (\mathbf{x}^{+} - \mathbf{x}^{-}) = 2. \tag{2}$$

The $+$ refers to the margin close to the diamonds, and the $-$ to the margin close to the circles. Moreover, for the diamonds (labelled $y^{(n)} = +1$) we will have

$$\boldsymbol{\theta}^T \mathbf{x}^{(n)} + \theta_0 \geq 1 \tag{3}$$

and for the circles (labelled $y^{(n)} = -1$)

$$\boldsymbol{\theta}^T \mathbf{x}^{(n)} + \theta_0 \leq -1. \tag{4}$$

Multiplying these last two equations by the label $y^{(n)}$ they can both be expressed as simply

$$y^{(n)}\left(\boldsymbol{\theta}^T\mathbf{x}^{(n)}+\theta_0\right)-1\geq 0. \tag{5}$$

We also know from Equation (2) that

$$\text{margin width}=\frac{2}{|\boldsymbol{\theta}|}.$$

And so maximizing the margin is equivalent to finding $\boldsymbol{\theta}$ and θ_0 to minimize $|\boldsymbol{\theta}|$ or

$$\min_{\boldsymbol{\theta},\theta_0}\frac{1}{2}|\boldsymbol{\theta}|^2.$$

Of course, subject to the constraint (5) for all n.

Once we have found $\boldsymbol{\theta}$ and θ_0 and given a new, unclassified, data point, $\mathbf{u}$, say, we just plug everything into

$$\boldsymbol{\theta}^T \mathbf{u} + \theta_0, \tag{6}$$

and depending on whether this is positive or negative we have a diamond or a circle. This is known as the *primal version* of the classifier.

Example: Irises

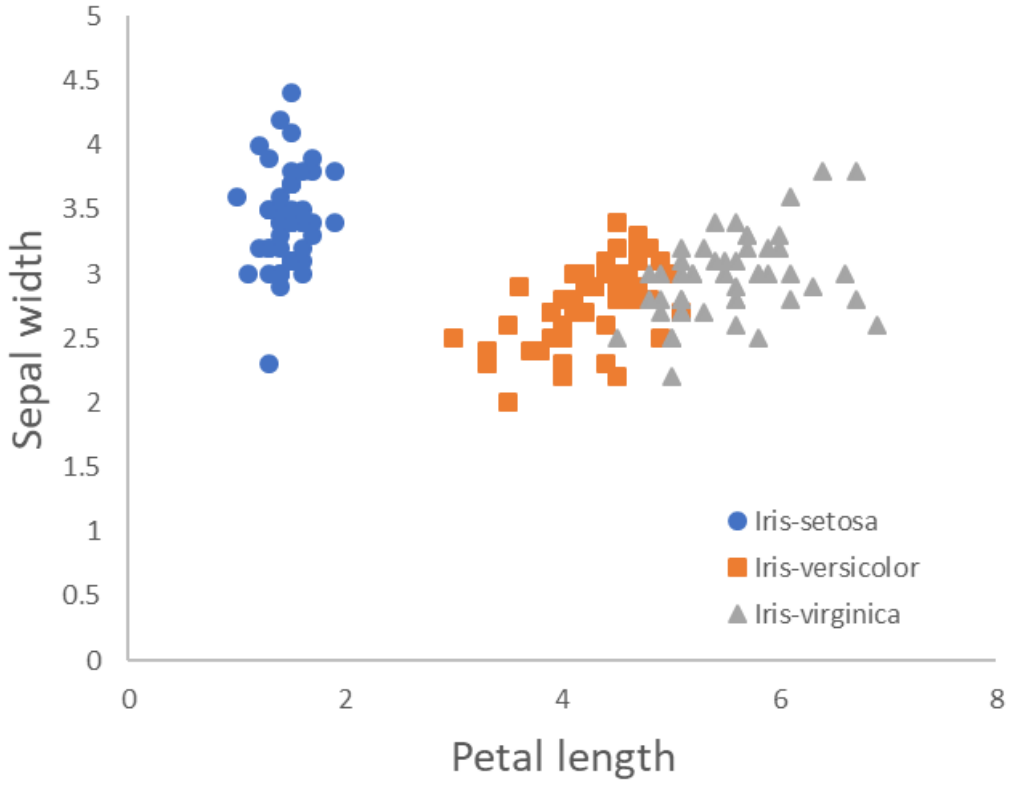
There is a popular data set often used in machine learning, a dataset of measurements for different types of iris, the flower. You can get the dataset here: `https://www.kaggle.com/uciml/iris`.

There are three types of iris in the data: Setosa; Versicolor; Virginica. And there are 50 samples of each, with measurements for sepal length and width, petal length and width.

The first few lines of raw data in the csv file look like this:

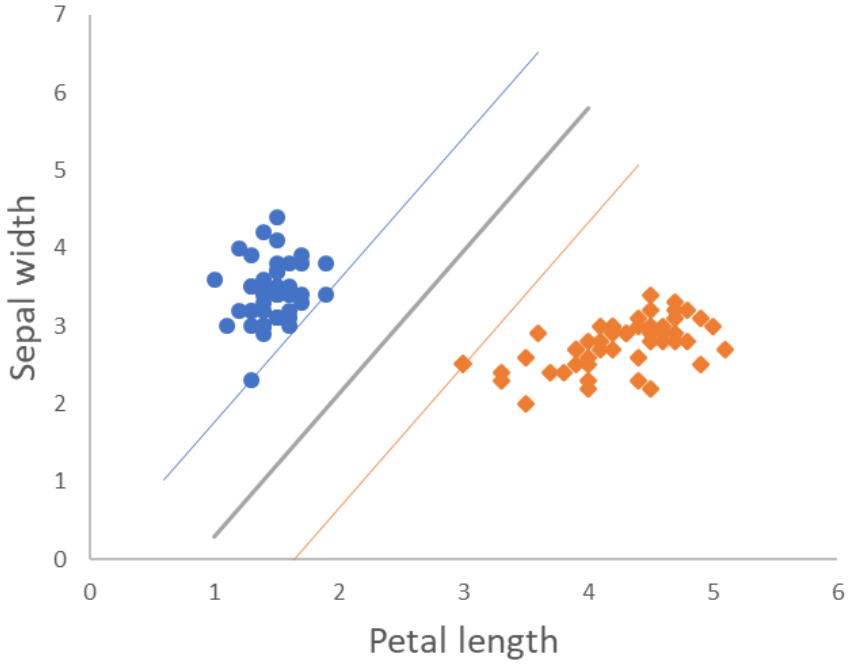
```
SepalLength,SepalWidth,PetalLength,PetalWidth,Name
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5,3.6,1.4,0.2,Iris-setosa
```

In the figure are shown the 50 samples for each type of iris, the data being just petal length and sepal width. (I can't plot in four dimensions!)



I am going to make this as simple as possible and just use the setosa and versicolor varieties since eyeballing them shows a very clear demarcation.

The results are shown in the figure.



Not the most challenging example but Excel's Solver did the job!

I found the hyperplane for the two-dimensional, two-classes, iris problem to be

$$0.686 \times \text{sepal width} - 1.257 \times \text{petal length} + 1.057 = 0.$$

Now given a new iris with values for sepal width and petal length we can determine whether it is setosa or versicolor by plugging the numbers into Expression (6), or equivalently the left-hand side of the above, and seeing whether the number that comes out is positive or negative. A positive number for the left-hand side means you have a setosa.

If the number has absolute value less than one it means we have an iris in the middle, the no-man's land. We will still have a classification but we won't be confident in it.

As a little experiment see what happens if you move one of the data points, say a setosa, clearly into the versicolor group.

When you try to solve for the dividing hyperplane you will find that there is 'no feasible solution.'

Soft Margins

What if our data is not linearly separable? I'm going to explain a possible way to address this situation.

We simply try to do the *best* we can while accepting that some data points are in places you'd think they shouldn't be. It's all about minimizing a loss function.

Let me introduce the function

$$J = \frac{1}{N}\sum_{n=1}^{N} \max\left(1 - y^{(n)}\left(\boldsymbol{\theta}^T \mathbf{x}^{(n)} + \theta_0\right), 0\right) + \lambda|\boldsymbol{\theta}|^2. \qquad (7)$$

This we want to minimize by choosing $\boldsymbol{\theta}$ and θ_0.

- The parameter λ is a weighting between the margin size and whether the data lies on the 'correct' side of the margin
- In the language of optimization the first term is the loss function and the second term is the regularization
- The maximum function in the first term has the effect of totally ignoring how far into the 'correct' region a point is and only penalizes distance into the 'wrong' region, measured from the 'correct' margin.

Gradient descent

Because this function is convex in the θs we can easily apply a gradient descent method to find the minimum.

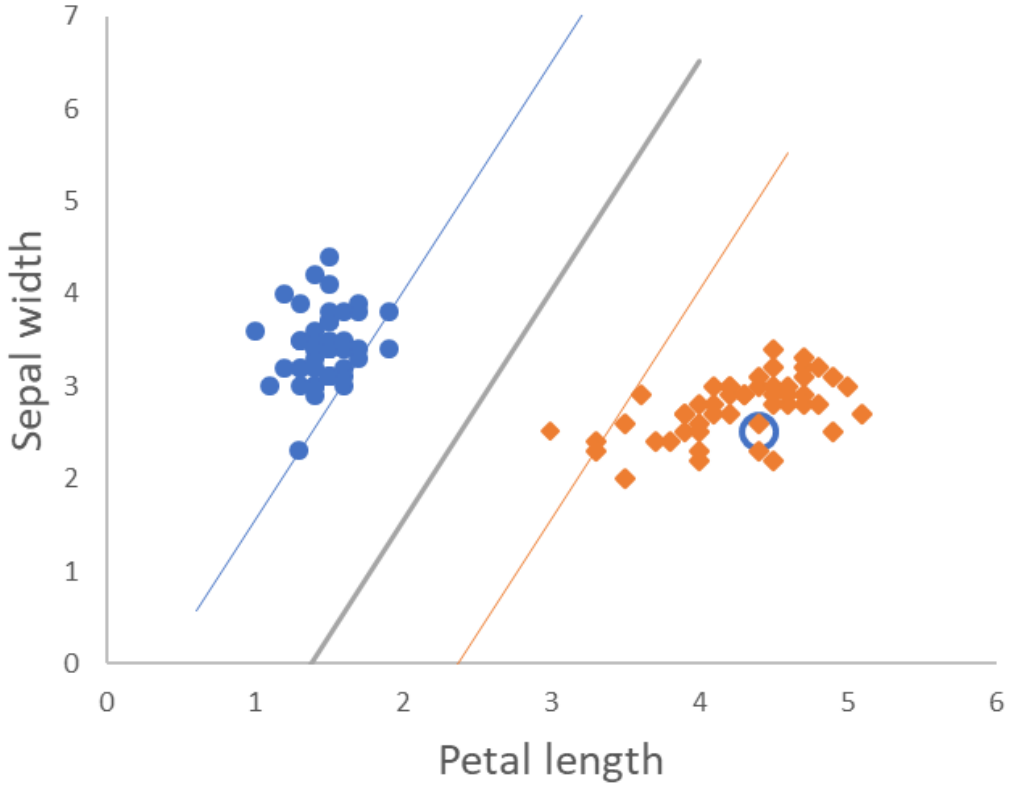
For example, with stochastic gradient descent pick an n at random and update according to

$$\text{New } \theta_0 = \text{Old } \theta_0 - \beta \begin{cases} -y^{(n)} & \text{if } 1 - y^{(n)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(n)} + \theta_0 \right) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\text{New } \boldsymbol{\theta} = \text{Old } \boldsymbol{\theta} - \beta \begin{cases} 2\lambda\boldsymbol{\theta} - \frac{1}{N} y^{(n)} \mathbf{x}^{(n)} & \text{if } 1 - y^{(n)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(n)} + \theta_0 \right) > 0 \\ 2\lambda\boldsymbol{\theta} & \text{otherwise} \end{cases} .$$

In the figure below I've taken the earlier iris data and relabeled one versicolor as setosa. The resulting margins have become wider, and are softer since a few data points are within those margins.



Summary

Please take away the following important ideas

- SVM is a supervised-learning classification method that divides data across hyperplanes in feature space
- If data is linearly separable you have hard margins, otherwise the margins are soft
- With soft margins there is a cost function to be minimized