Stanford Machine Learning - Week VII

Eric N Johnson

August 27, 2016

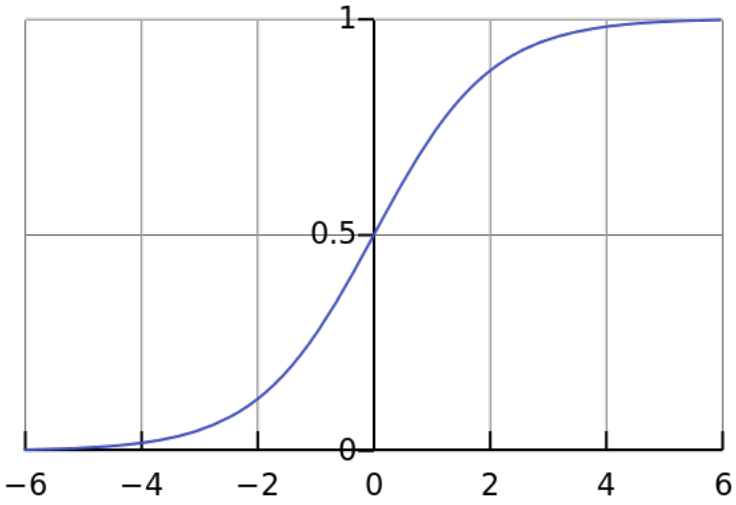
1 Support Vector Machines

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

1.1 Optimization Objective

We start with an alternative view of logistic regression. In logistic regression we have the familiar sigmoid function:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$



If an example has $y = 1$, we want $h_\theta(x) \approx 1$, or $z = \theta^T x >> 0$. Here, we would correctly classify x such that $y = 1$. Conversely, if we have $y = 0$, we want $h_\theta(x) \approx 0$, or $z = \theta^T x << 0$.

We have our cost function:

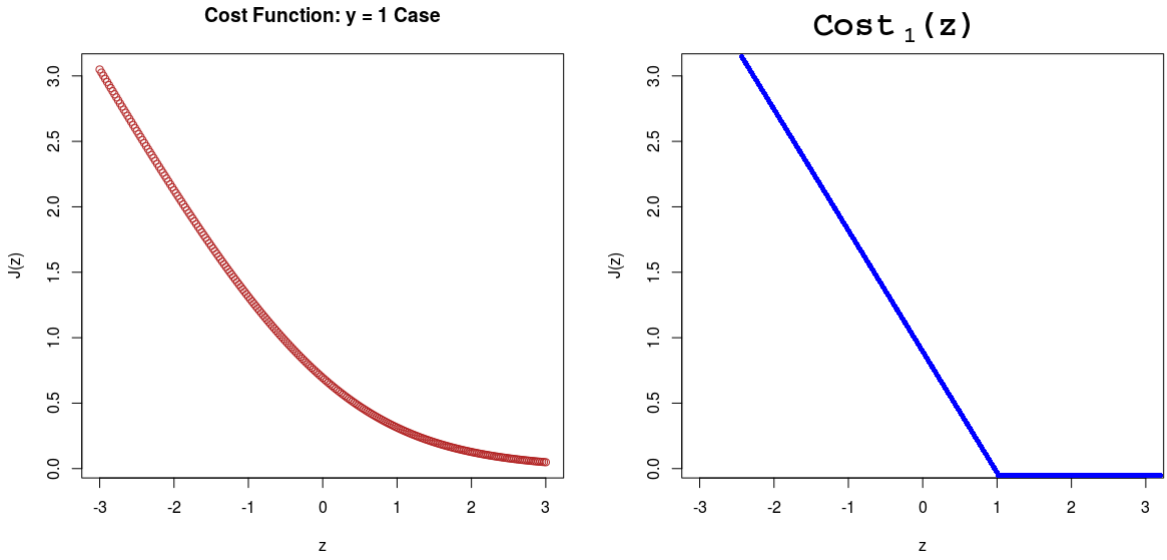
$$-(y \log(h_\theta(x) + (1 - y) \log(1 - h_\theta(x))) = -y \log\left(\frac{1}{1 + e^{-\theta^T x}}\right) - (1 - y) \log\left(1 - \frac{1}{1 + e^{-\theta^T x}}\right)$$

There are two cases to consider. When $y = 1$ and when $y = 0$.

- The $y = 1$ case... We have the function

$$-\log\left(\frac{1}{1+\mathrm{e}^{-\theta^T x}}\right)$$

which looks like this:

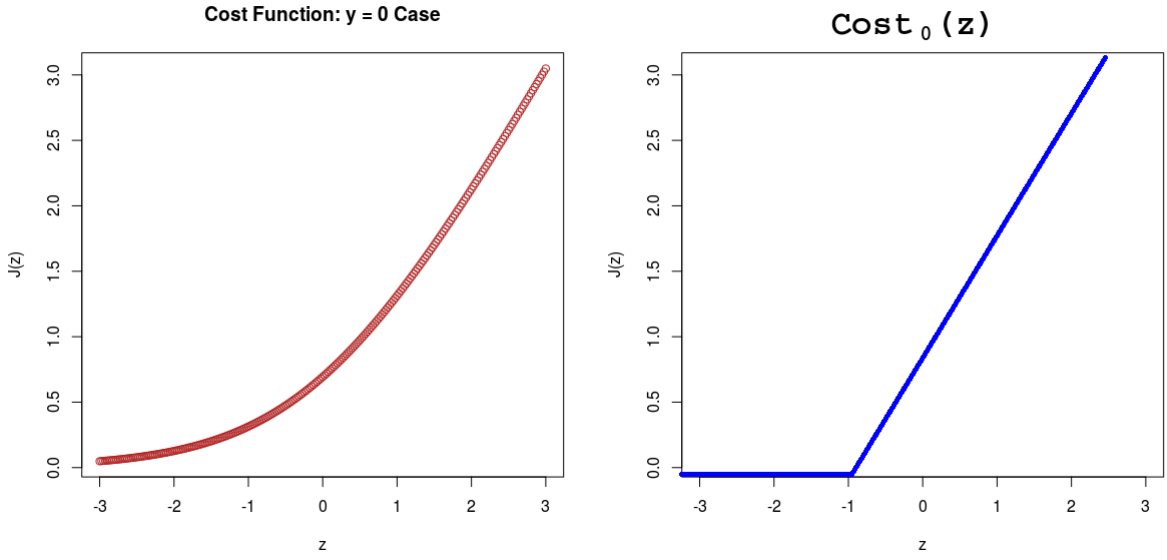


We will call this $\mathbf{cost}_1(z)$.

- The $y = 0$ case... We have the function

$$-\log\left(1-\frac{1}{1+\mathrm{e}^{-\theta^T x}}\right)$$

which looks like this:



We will call this $\mathbf{cost}_0(z)$.

In both cases, we use the part of these curves where the slope is approximately ± 1, and set the rest to zero.

In logistic regression, we minimize the function:

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log\left(h_\theta(x^{(i)})\right)\right) + (1-y^{(i)})\left(-\log\left(1-h^{\theta}(x^{(i)})\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$$= \min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\text{cost}_1\left(\theta^T x^{(i)}\right) + (1-y^{(i)})\text{cost}_0\left(\theta^T x^{(i)}\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

But for a SVM, we minimize:

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\text{cost}_1\left(\theta^T x^{(i)}\right) + (1-y^{(i)})\text{cost}_0\left(\theta^T x^{(i)}\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$
$$= min_\theta(A + \lambda B)$$

but re-parameterize the function by convention by dropping the $1/m$ terms. Obviously this does not change the values in θ that minimizes the function. Second, we replace the regularization parameter λ, and rewrite.

$$\min_{\theta}(cA + B)$$

This matrix equation is just a different way of parameterizing. Previously, λ was used to change the weight of B - but now we change the weight of A using the parameter $c \in \mathbb{R}$. These expressions will give the save value if $c = 1/\lambda$.

Thus we minimize the following for a Support Vector Machine:

$$\min_{\theta} c\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1\left(\theta^T x^{(i)}\right) + (1-y^{(i)})\text{cost}_0\left(\theta^T x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

The hypothesis then outputs:

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

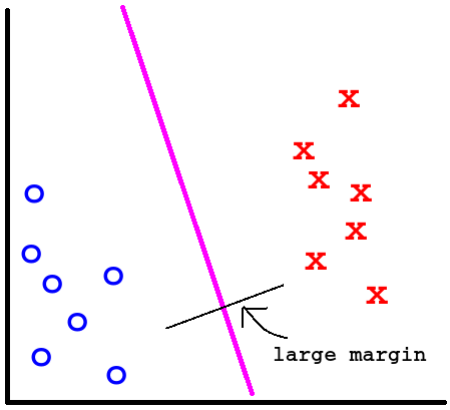
2 Large Margin Classification

Sometimes SVMs are referred to as "Large Margin Classifiers." Again, we are finding the values of θ such that we minimize

$$\min_\theta c\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1\left(\theta^T x^{(i)}\right) + (1-y^{(i)})\text{cost}_0\left(\theta^T x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

- Whenever $y^{(i)} = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0).
- Whenever $y^{(i)} = 0$, we want $\theta^T x \leq -1$ (not just < 0).

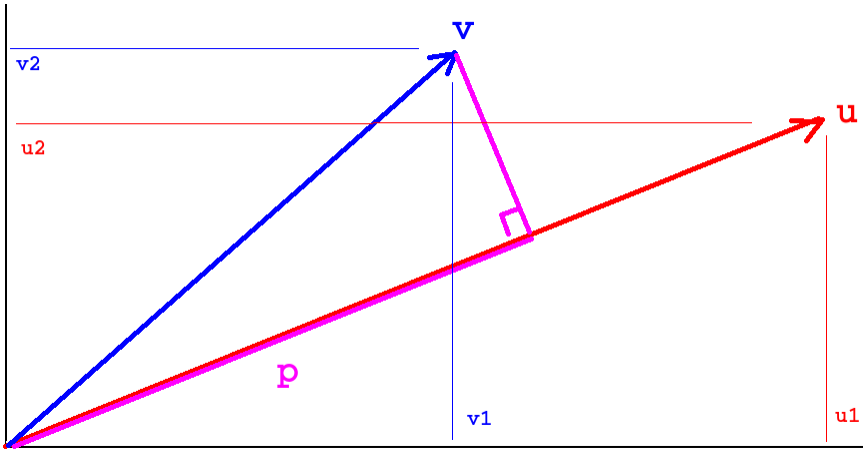
When we do this, we get an interesting decision boundary because it is *linearly separable*. That means the examples are further away from our boundary, or that the margin that separates classes is maximized. Hence the name 'large margin classifier.'



If we use large margin classification by itself, however, the algorithm will be sensitive to outliers. The regularization parameter c effects how sensitive the decision boundary is to outliers. If c is very large, the SVM will be very sensitive - and will be less sensitive if c is small. Remember, $c \approx \frac{1}{\lambda}$. The SVM will be looking for a hyperplane rather than a line in most cases. If the categories aren't linearly separable, changes in c will help the SVM to perform well anyway.

2.1 The Mathematics Behind Large Margin Classification

First, the vector inner product $u^T v$... Let $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$, and $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$. Then let p be the length of the projection of v onto u.



The inner product is

$$\begin{aligned} u^T v &= v^T u \\ &= p \cdot |u|_2 \\ &= u_1 v_1 + u_2 v_2 \end{aligned}$$

Note that $p, |u|_2, u^T v \in \mathbb{R}$. p is signed - but the others are nonnegative.

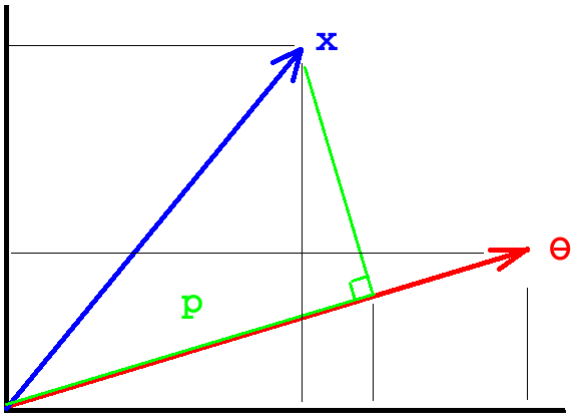
SVM Decision Boundary for an example:

$$\begin{aligned} \min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 & \\ \text{s.t. } \theta^T x^{(i)} &\geq 1 \qquad \text{if } y^{(i)} = 1 \\ \theta^T x^{(i)} &\leq -1 \qquad \text{if } y^{(i)} = 0 \end{aligned}$$

For simplification, we set $\theta_0 = 0$, and $n = 2$. We set

$$\begin{aligned} \min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2) &= \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 \\ &= \frac{1}{2}|\theta|_2^2 \end{aligned}$$

We plot $\theta^T x^{(i)}$ using the earlier method where we took the projection of x onto θ.



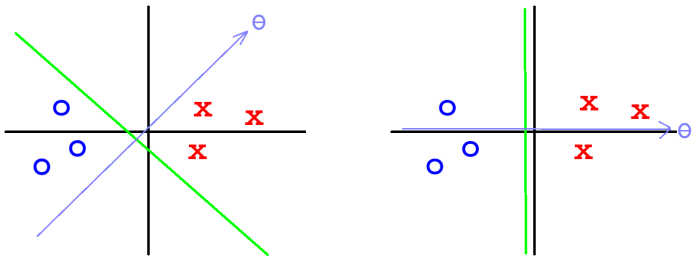
and note that

$$\begin{aligned} \theta^T x^{(i)} &= p^{(i)} |\theta|_2^2 \\ &= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \end{aligned}$$

to restate the problem as

$$\begin{aligned} \min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2}|\theta|_2^2 & \\ \text{s.t. } p^{(i)} \cdot |\theta|_2^2 &\geq 1 \qquad \text{if } y^{(i)} = 1 \\ p^{(i)} \cdot |\theta|_2^2 &\leq -1 \qquad \text{if } y^{(i)} = 0 \end{aligned}$$

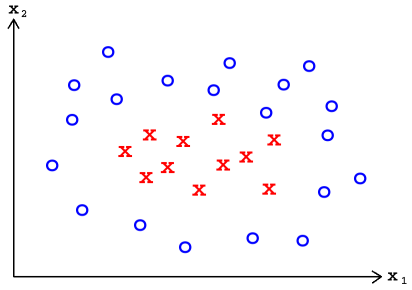
where $\theta_0 = 0$, and $p^{(i)}$ is the projection of $x^{(i)}$ onto $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$. In this case, why would the decision boundary (green) be like the right plot and not like the left plot?



The right plot allows the norm of θ to be smaller - which optimizes the problem.

3 Kernels I

Suppose we have a nonlinear decision boundary - something, perhaps, that looks like this:



One way to create a decision boundary is to have a large polynomial that predicts $y = 1$ if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \ldots \geq 0$$

$$\leftrightarrow$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \ldots \geq 0$$

with hypothesis

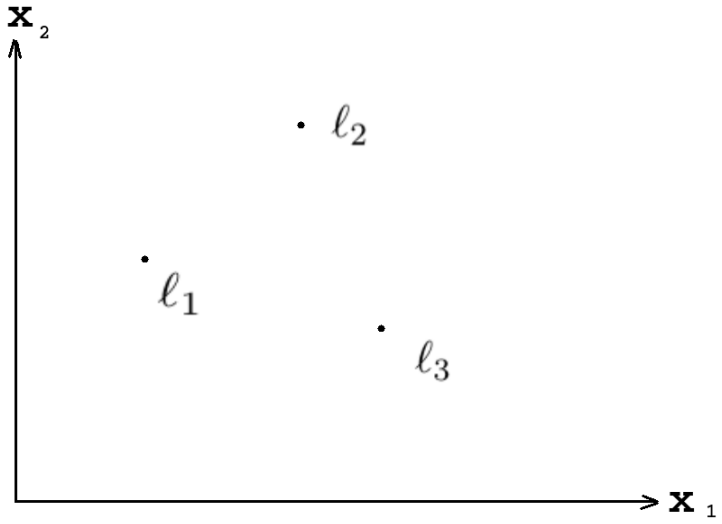
$$h_\theta(x) = \begin{cases} 1 & \text{if} \quad \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \ldots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

But is there a better way to pick the features $f_1, f_2, \ldots$?

Given x, we can compute new features depending upon proximity to landmarks ℓ_1, ℓ_2, ℓ_3. Note, I used slightly different notation than the instructor. Let

$$\ell_{i,j} = \ell_i^{(j)}$$

in these drawings.



Given x, we define

$$f_1 = \text{similarity}(x, \ell_1) = \exp\left(-\frac{|x - \ell_1|_2^2}{2\sigma^2}\right)$$

$$f_2 = \text{similarity}(x, \ell_2) = \exp\left(-\frac{|x - \ell_2|_2^2}{2\sigma^2}\right)$$

$$f_3 = \text{similarity}(x, \ell_3) = \exp\left(-\frac{|x - \ell_3|_2^2}{2\sigma^2}\right)$$

This particular choice is called a *Gaussian kernel*.

3.1 Kernels and Similarity

What do kernels do? They provide some metric for how close something is... For instance, let

$$f_1 = \text{similarity}(x, \ell^{(1)}) = \exp\left(-\frac{|x - \ell^{(1)}|_2^2}{2\sigma^2}\right)$$
$$= \exp\left(-\frac{\sum_{j=1}^{n}\left(x_j - \ell_j^{(1)}\right)^2}{2\sigma^2}\right).$$

- If $x \approx \ell^{(1)}$:

$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$$

- If x is far from $\ell^{(1)}$:

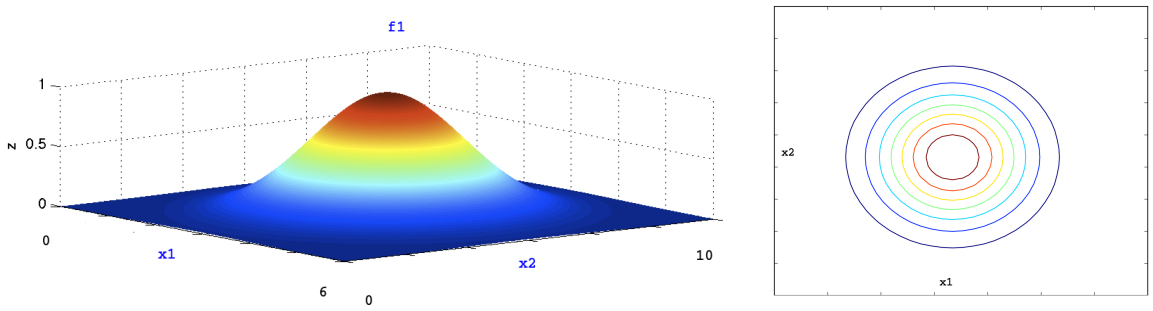
$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0$$

We then use $\ell^{(1)}, \ell^{(2)}, \ldots$ to come up with features $f_1, f_2, \ldots$

Here is an example. Let

$$\ell^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, f_1 = \exp\left(-\frac{|x - \ell^{(1)}|_2^2}{2\sigma^2}\right)$$

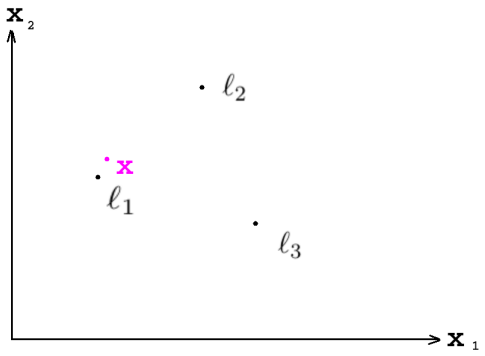
where $\sigma^2 = 1$.



These show the density given how close x is to $\ell^{(1)}$. Sigma serves the function of determining how wide the curve is.

3.2 Kernel example

Suppose we have a training example that is close to $\ell^{(1)}$.



and we predict 1 when

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0,$$

where

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1 \\ 1 \\ 0 \end{bmatrix},$$

$$f_1 \approx 1$$
$$f_2 \approx 0$$
$$f_3 \approx 0.$$

Then

$$\begin{aligned} \theta_0 + \theta_1(1) + \theta_2(0) + \theta_3(0) &= 0.5 \\ &\geq 0 \end{aligned}$$

and we predict $y = 1$. However, if we pick a different point that makes $f_1, f_2, f_3 \approx 0$, we predict $y = 0$. When points are near $\ell^{(i)}$, for $i = 1, 2, 3$, we predict $y = 1$. Otherwise we predict $y = 0$.

4 Kernels II

Where do we get our landmarks in practice?

Pick landmarks that are exactly the training examples. Let $\ell^{(1)} = x_1$, $\ell^{(2)} = x_2$, etc. This will measure in a direct way how close new examples are to training examples:

$$\begin{aligned}&\text{Given } (x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)}),\\ &\text{choose } \ell^{(1)} = x^{(1)}, ..., \ell^{(m)} = x^{(m)}\end{aligned}$$

and pick

$$\begin{aligned}f_1 &= \text{similarity}(x, \ell^{(1)})\\ f_2 &= \text{similarity}(x, \ell^{(2)})\\ &\vdots\end{aligned}$$

given x.

This gives us a feature vector

$$f = \begin{bmatrix} f_0 = 1 \\ f_1 \\ f_2 \\ \vdots \\ f_i \\ \vdots \\ f_m \end{bmatrix} \in \mathbb{R}^{m+1}$$

So for a training example $(x^{(i)}, y^{(i)})$, we obtain

$$\begin{aligned}f_1^{(i)} &= \text{similarity}(x^{(i)}, \ell^{(1)})\\ f_2^{(i)} &= \text{similarity}(x^{(i)}, \ell^{(2)})\\ f_3^{(i)} &= \text{similarity}(x^{(i)}, \ell^{(3)})\\ &\vdots\\ f_i^{(i)} &= \text{similarity}(x^{(i)}, \ell^{(i)})\\ &\vdots\\ f_m^{(i)} &= \text{similarity}(x^{(i)}, \ell^{(m)})\end{aligned}$$

and allows us represent the examples as $(f^{(i)}, y^{(i)})$.

4.1 Support Vector Machines with Kernels

- Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$
  - Predict $y = 1$ if $\theta^T f \geq 0$.
  - Predict $y = 0$ otherwise.
- Train the SVM using

$$\min_\theta c \sum_{i=1}^{m} y^{(i)} \text{cost}_1\left(\theta^T f^{(i)}\right) + (1 - y^{(i)}) \text{cost}_0\left(\theta^T f^{(i)}\right) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

  This gives us the parameters found in θ. Just like before, we skip regularizing θ_0, which is why $j = 0$ is missing from the second sum. Also note that $n = m$, so we take that second sum to m. In practice, the $\theta^2 = \theta^T \theta$ term is replaced by $\theta^T M \theta$, where the choice of matrix M depends upon the kernel used.

Note that it is better to use external packages to perform optimization on functions like this. Writing one is very difficult - and will not likely work as well as an 'off the shelf' optimization package.

4.2 Some notes about SVM parameters

- $C = \frac{1}{\lambda}$... A regularization term.
  - Large C (small λ): Lower bias, high variance
  - Small C (large λ): Higher bias, low variance
- σ^2: A measure, obviously, of the spread of the Gaussian curve.
  - Large σ^2: Features f_i vary more smoothly. Higher bias, lower variance.
  - Small σ^2: Features f_i vary less smoothly. Lower bias, higher variance.

5 Support Vector Machines in Practice

- First, use SVM packages to solve for the parameters θ and perform optimization. See *liblinear, libsvm*, etc.
- Regardless of what package is used, we will have to pick C and which kernel is used.
  - We can use a 'linear kernel' (which is like not using one at all), which predicts $y = 1$ if $\theta^T x \geq 0$. This may be good if n (the number of features) is very large, but the number of training examples m is small.
  - Gaussian kernel. We will need to pick σ^2.

$$f_i = \exp\left(-\frac{|x - \ell^{(i)}|_2^2}{2\sigma^2}\right)$$
$$\ell^{(i)} = x^{(i)}$$

    This may be a good choice if the number of features n is small and / or the number of training examples m is large.

- Regardless of what package we use, the kernel will probably have to be written...

```
function f = kernel(x1, x2)
        f = exp(-1/(2*sigma^2) * (x1^2 + x2^2)^.5)
return
```

  It is good practice to perform feature scaling before using a Gaussian kernel. Gaussian and linear kernels are often implemented in SVM packages since they are so common.

- Other choices of kernel...
  - Whatever kernel is used has to satisfy "Mercer's Theorem."
  - Not all similarity functions make valid kernels.
  - Polynomial kernels: $k(x, \ell) = (x^T \ell)2$, which have a general form

$$k(x, \ell) = (x^T \ell + \alpha)^p$$

    where $\alpha \in \mathbb{R}$ and p is the polynomial degree.
  - String kernels
  - chi-squared kernels
  - histogram intersection kernels
  - Etc.

5.1 Multi-class classification

Many SVM packages already have built-in multi-class classification functionality. We can, otherwise, use a one-verses-all method. This method is the same as for logistic regression.

5.2 SVMs verses Logistic Regression

If the number of features n, where $x \in \mathbb{R}^{n+1}$, and the number of training examples m is...

- If n is large relative to m, use Logistic Regression or SVM with a linear kernel.
- If n is small and m is fairly large (less than 10000), use SVM with a Gaussian Kernel.
- If n is small but m is very large, use Logistic Regression or SVM with a linear kernel.
- Neural networks are likely to work well for most of these cases but may be slower to train.

SVMs have gained a reputation of being highly capable machine learning methods.