[Summary]SVMs 1

Optimization Objective

The **Support Vector Machine** (SVM) is yet another type of supervised machine learning algorithm. It is sometimes cleaner and more powerful.

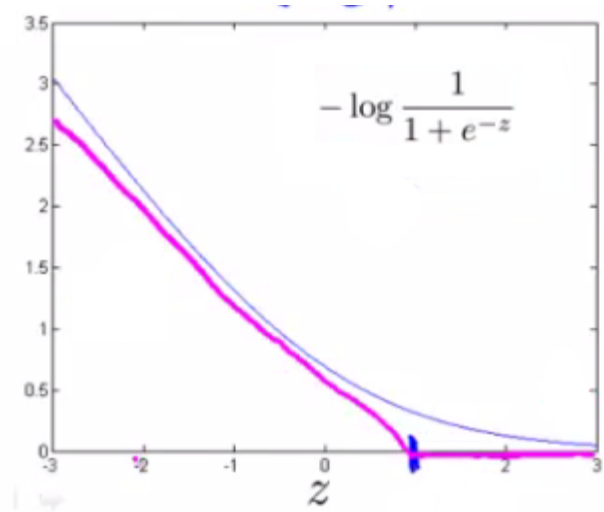
Recall that in logistic regression, we use the following rules:

$$if\ y = 1,\ then\ \mathrm{h}_\Theta(x) \approx 1\ and\ \Theta^T x \gg 0$$
$$if\ y = 0,\ then\ \mathrm{h}_\Theta(x) \approx 0\ and\ \Theta^T x \ll 0$$

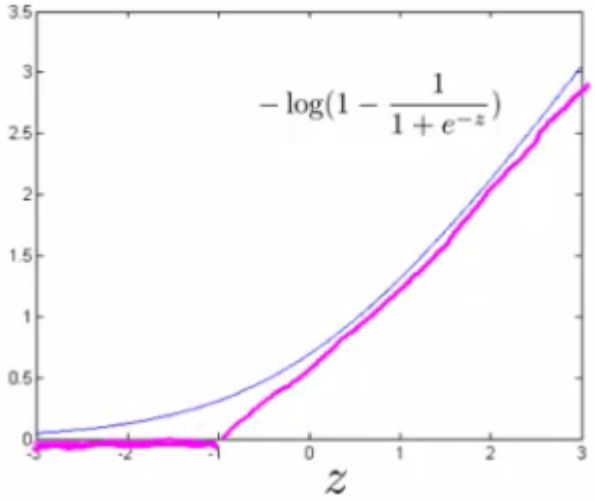
Recall the cost function for (unregularized) logistic regression:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log\left(\mathrm{h}_\theta\left(x^i\right)\right) + \left(1-y^i\right)\log\left(1-\mathrm{h}_\theta\left(x^i\right)\right)\right]$$
$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log\left(\frac{1}{1+e^{-\theta^T x^i}}\right) + \left(1-y^i\right)\log\left(1-\frac{1}{1+e^{-\theta^T x^i}}\right)\right]$$

To make a support vector machine, we will modify the first term of the cost function $-\log\left(\mathrm{h}_\theta(x)\right) = -\log\left(\frac{1}{1+e^{-\theta^T x}}\right)$ so that when $\theta^T x$ (from now on, we shall refer to this as z) is **greater than** 1, it outputs 0. Furthermore, for values of z less than 1, we shall use a straight decreasing line instead of the sigmoid curve.



Similarly, we modify the second term of the cost function $-\log(1-\mathrm{h}_\theta(x)) = -\log\left(1-\frac{1}{1+e^{-\theta^T x}}\right)$ so that when z is **less than** –1, it outputs 0. We also modify it so that for values of z greater than –1, we use a straight increasing line instead of the sigmoid curve.



We shall denote these as $cost_1(z)$ and $cost_0(z)$ (respectively, note that $cost_1(z)$ is the cost for classifying when y=1, and $cost_0(z)$ is the cost for classifying when y=0), and we may define them as follows (where k is an arbitrary constant defining the magnitude of the slope of the line):

$$z = \theta^T x$$

$$cost_0(z) = \max(0, k(1+z))$$
$$cost_1(z) = \max(0, k(1-z))$$

Recall the full cost function from (regularized) logistic regression:

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

Note that the negative sign has been distributed into the sum in the above equation.

We may transform this into the cost function for support vector machines by substituting $cost_0(z)$ and $cost_1(z)$:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[y^i cost_1(\theta^T x^i) + \left(1-y^i\right)(cost_0(\theta^T x^i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

We can optimize this a bit by multiplying this by m (thus removing the m factor in the denominators). Note that this does not affect our optimization, since we're simply multiplying our cost function by a positive constant (for example, minimizing $(u-5)^2+1$ gives us 5; multiplying it by 10 to make it $10(u-5)^2+10$ still gives us 5 when minimized).

$$J(\theta) = \sum_{i=1}^{m}\left[y^i cost_1(\theta^T x^i) + \left(1-y^i\right)(cost_0(\theta^T x^i))\right] + \frac{\lambda}{2}\sum_{j=1}^{n}\theta_j^2$$

Furthermore, convention dictates that we regularize using a factor C, instead of λ, like so:

$$J(\theta) = C\sum_{i=1}^{m}\left[y^i cost_1(\theta^T x^i) + \left(1-y^i\right)(cost_0(\theta^T x^i))\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

This is equivalent to multiplying the equation by $C = \frac{1}{\lambda}$, and thus results in the same values when optimized. Now, when we wish to regularize more (that is, reduce overfitting), we decrease C, and when we wish to regularize less (that is, reduce underfitting), we increase C.

Finally, note that the hypothesis of the Support Vector Machine is *not* interpreted as the probability of y being 1 or 0 (as it is for the hypothesis of logistic regression). Instead, it outputs either 1 or 0. (In technical terms, it is a discriminant function.)

$$\mathrm{h}_\theta(x) = \begin{cases} 1\ if\ \Theta^T x \geq 0 \\ 0\ otherwise \end{cases}$$

Large Margin Intuition

A useful way to think about Support Vector Machines is to think of them as **Large Margin Classifiers**.

If y=1, we want $\Theta^T x \geq 1$ (not just ⩾0)
If y=0, we want $\Theta^T x \leq -1$ (not just <0)

Now when we set our constant C to a very **large** value (e.g. 100,000), our optimizing function will constrain Θ such that the equation A (the summation of the cost of each example) equals 0. We impose the following constraints on Θ:

$\Theta^T x \geq 1$ if y=1 and $\Theta^T x \leq -1$ if y=0.

If C is very large, we must choose Θ parameters such that:

$$\sum_{i=1}^{m}\left[y^i cost_1(\theta^T x^i) + \left(1-y^i\right)(cost_0(\theta^T x^i))\right] = 0$$

This reduces our cost function to:

$$J(\theta) = C \cdot 0 + \frac{1}{2}\sum_{j=1}^{n} \Theta_j^2$$
$$= \frac{1}{2}\sum_{j=1}^{n} \Theta_j^2$$

Recall the decision boundary from logistic regression (the line separating the positive and negative examples). In SVMs, the decision boundary has the special property that it is **as far away as possible** from both the positive and the negative examples.

The distance of the decision boundary to the nearest example is called the **margin**. Since SVMs maximize this margin, it is often called a *Large Margin Classifier*.

The SVM will separate the negative and positive examples by a **large margin**.

This large margin is only achieved when **C is very large**.

Data is **linearly separable** when a **straight line** can separate the positive and negative examples.

If we have **outlier** examples that we don't want to affect the decision boundary, then we can **reduce** C.

Increasing and decreasing C is similar to respectively decreasing and increasing λ, and can simplify our decision boundary.