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:

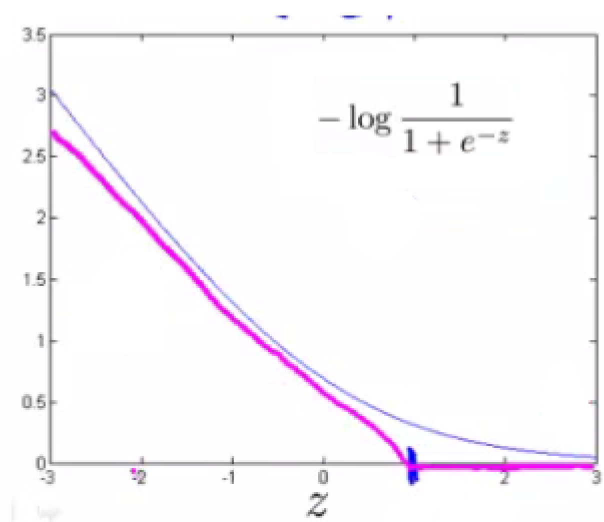
if y=1, then $h_\theta(x) \approx 1$ and $\Theta^T x \gg 0$

if y=0, then $h_\theta(x) \approx 0$ and $\Theta^T x \ll 0$

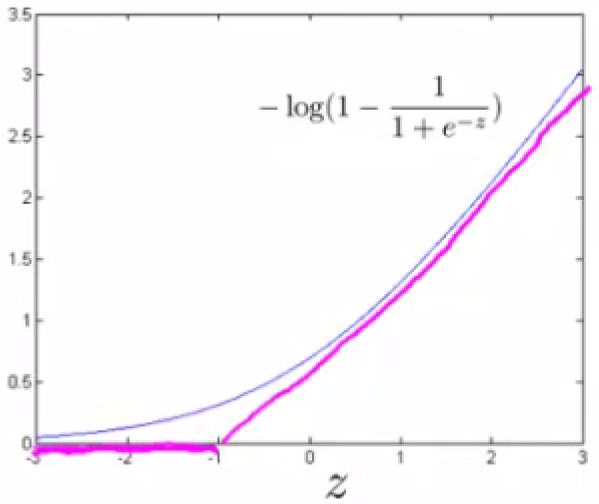
Recall the cost function for (unregularized) logistic regression:

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \\ &= \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} \log\left(\frac{1}{1 + e^{-\theta^T x^{(i)}}}\right) - (1 - y^{(i)}) \log\left(1 - \frac{1}{1 + e^{-\theta^T x^{(i)}}}\right) \end{aligned}$$

To make a support vector machine, we will modify the first term of the cost function $-\log(h_\theta(x)) = -\log\left(\frac{1}{1 + e^{-\theta^T x}}\right)$ 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.(In the literature, this is called a hinge loss (https://en.wikipedia.org/wiki/Hinge_loss) function.)



Similarly, we modify the second term of the cost function $-\log(1 - h_{\theta(x)}) = -\log\left(1 - \frac{1}{1 + e^{-\theta^T x}}\right)$ 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 $\text{cost}_1(z)$ and $\text{cost}_0(z)$ (respectively, note that $\text{cost}_1(z)$ is the cost for classifying when y=1, and $\text{cost}_0(z)$ 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):

$z = \theta^T x$

$\text{cost}_0(z) = \max(0, k(1+z))$

$\text{cost}_1(z) = \max(0, k(1-z))$

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

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}(-\log(h_\theta(x^{(i)}))) + (1-y^{(i)})(-\log(1-h_\theta(x^{(i)}))) + \frac{\lambda}{2m}\sum_{j=1}^{n}\Theta_j^2$$

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 $\text{cost}_0(z)$ and $\text{cost}_1(z)$:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}\ \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\ \text{cost}_0(\theta^T x^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n}\Theta_j^2$$

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).

$$J(\theta) = \sum_{i=1}^{m} y^{(i)}\ \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\ \text{cost}_0(\theta^T x^{(i)}) + \frac{\lambda}{2}\sum_{j=1}^{n}\Theta_j^2$$

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

$$J(\theta) = C \sum_{i=1}^{m} y^{(i)} \text{ cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{ cost}_0(\theta^T x^{(i)}) + \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2$$

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.)

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

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:

$\sum_{i=1}^{m} y^{(i)} \text{cost}_1(\Theta^T x) + (1 - y^{(i)}) \text{cost}_0(\Theta^T x) = 0$

This reduces our cost function to:

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

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.

Mathematics Behind Large Margin Classification (Optional)

Vector Inner Product

Say we have two vectors, u and v:

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

The **length of vector v** is denoted $||v||$, and it describes the line on a graph from origin (0,0) to (v_1, v_2).

The length of vector v can be calculated with $\sqrt{v_1^2 + v_2^2}$ by the Pythagorean theorem.

The **projection** of vector v onto vector u is found by taking a right angle from u to the end of v, creating a right triangle.

- p= length of projection of v onto the vector u.
- $u^T v = p \cdot ||u||$

Note that $u^T v = ||u|| \cdot ||v|| \cos\theta$ where θ is the angle between u and v. Also, $p = ||v|| \cos\theta$. If you substitute p for $||v|| \cos\theta$, you get $u^T v = p \cdot ||u||$.

So the product $u^T v$ is equal to the length of the projection times the length of vector u.

In our example, since u and v are vectors of the same length, $u^T v = v^T u$.

$u^T v = v^T u = p \cdot ||u|| = u_1 v_1 + u_2 v_2$

If the **angle** between the lines for v and u is **greater than 90 degrees**, then the projection p will be **negative**.

$$
\begin{aligned}
&\min_{\Theta} \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2 \\
&= \frac{1}{2} (\Theta_1^2 + \Theta_2^2 + \cdots + \Theta_n^2) \\
&= \frac{1}{2} \left(\sqrt{\Theta_1^2 + \Theta_2^2 + \cdots + \Theta_n^2}\right)^2 \\
&= \frac{1}{2} ||\Theta||^2
\end{aligned}
$$

We can use the same rules to rewrite $\Theta^T x^{(i)}$:

$\Theta^T x^{(i)} = p^{(i)} \cdot ||\Theta|| = \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)}$

So we now have a new **optimization objective** by substituting $p^{(i)} \cdot ||\Theta||$ in for $\Theta^T x^{(i)}$:

If y=1, we want $p^{(i)} \cdot ||\Theta|| \geq 1$

If y=0, we want $p^{(i)} \cdot ||\Theta|| \leq -1$

The reason this causes a "large margin" is because: the vector for Θ is perpendicular to the decision boundary. In order for our optimization objective (above) to hold true, we need the absolute value of our projections $p^{(i)}$ to be as large as possible.

If $\Theta_0 = 0$, then all our decision boundaries will intersect (0,0). If $\Theta_0 \neq 0$, the support vector machine will still find a large margin for the decision boundary.