Support Vector Machines (SVM)

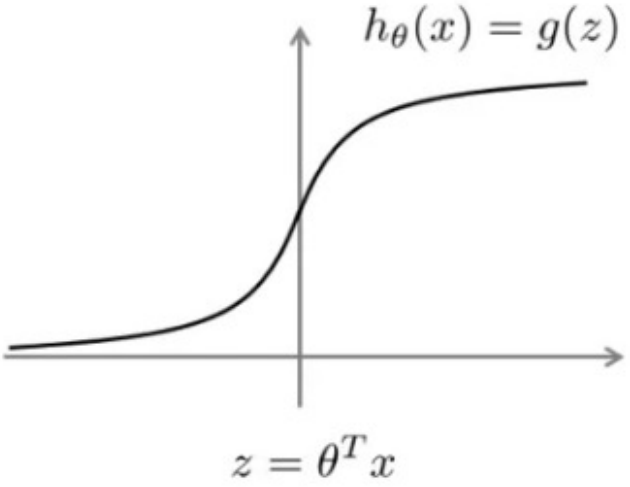
With supervised learning, the performance of many supervised learning algorithms will be pretty similar, and what matters less often will be whether we use learning algorithm A or learning algorithm B, but what matters more will often be things like the amount of data we create these algorithms on, as well as our skill in applying these algorithms.

One final supervised learning algorithm that is widely used - Support Vector Machines (SVM). Compared to both logistic regression and neural networks, a SVM sometimes gives a cleaner way of learning non-linear functions.

Alternative view of logistic regression

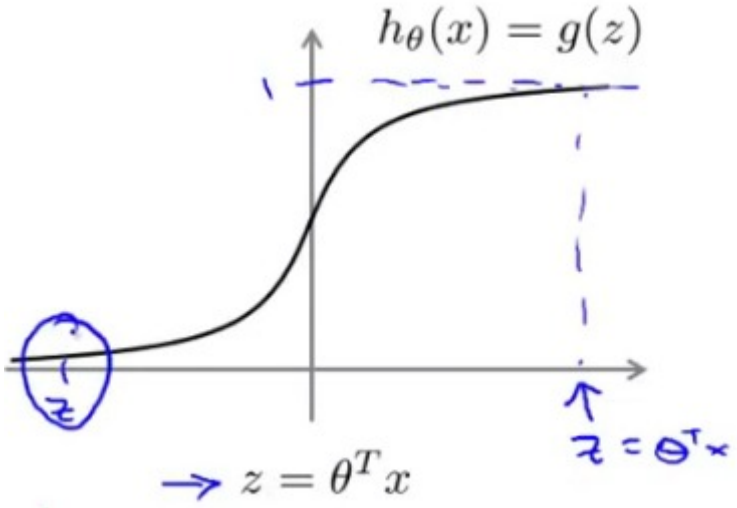
In order to describe the support vector machine, we're going to start with logistic regression, and show how we can modify it a bit, and get what is essentially the support vector machine.

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



Now let's think about what we would like logistic regression to do:

- If $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x >> 0$ (When we have an example where $y = 1$ then we hope $h_\theta(x)$ is close to 1).
- If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x << 0$ (When we have an example where $y = 0$ then we hope $h_\theta(x)$ is close to 0).



If we look at cost function, each example contributes a term like the one below to the overall cost function.

$$\text{Cost of example: } -(y \log h_\theta(x) + (1 - y) \log(1 - h_\theta(x)))$$

If we then plug in the hypothesis definition ($h_\theta(x)$), we get an expanded cost function equation;

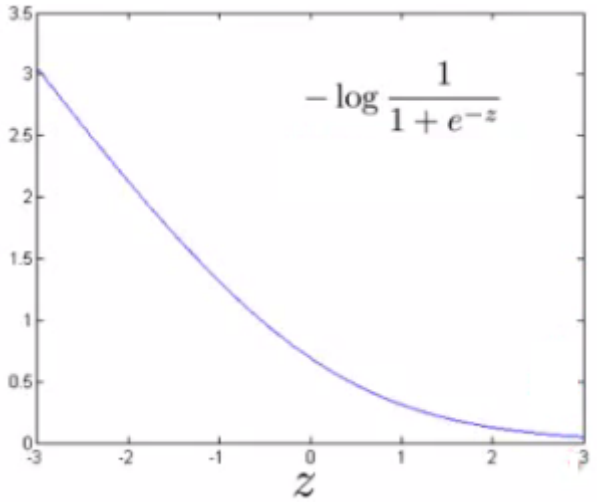
$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log(1 - \frac{1}{1+e^{-\theta^T x}})$$

So each training example contributes that term to the cost function for logistic regression

Now let's consider two cases of when $y = 1$ and when $y = 0$. In the first case, let's suppose that $y = 1$. In that case, only the first term in the objective matters (because in the second term (1 - y) would be 0).

- If $y = 1$ (want $\theta^T x >> 0$):

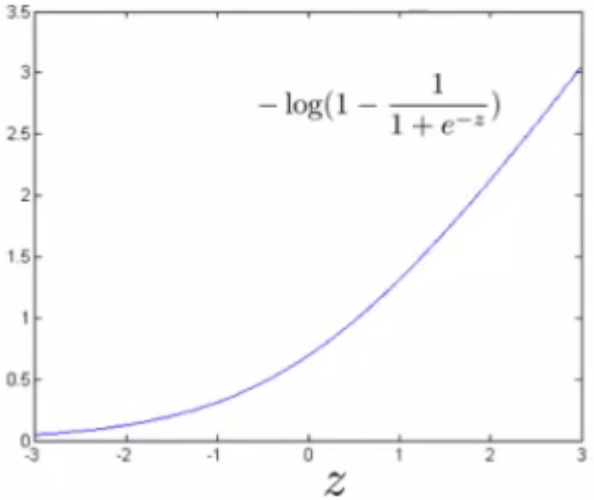
If we plot the function $-\log \frac{1}{1+e^{-z}}$ as a function of z:



What we find is that we get the before curve. And thus, we also see that when z is too large, that is, when $\theta^T X$ is large, that corresponds to a value of z that gives us a fairly small value, very small contribution to the cost function and this kinda explains why, when logistic regression sees a positive example, with $y = 1$, it tries to set $\theta^T X$ to be very large term. Concretely if z is big, **the cost is low**. But if z is 0 or negative **the cost contribution is high.**

The other case when $y = 0$. In that case, only the second term in the objective matters (because in the first term -y would be 0).

- If $y = 0$ (want $\theta^T x << 0$) We can again plot it and get a similar graph:

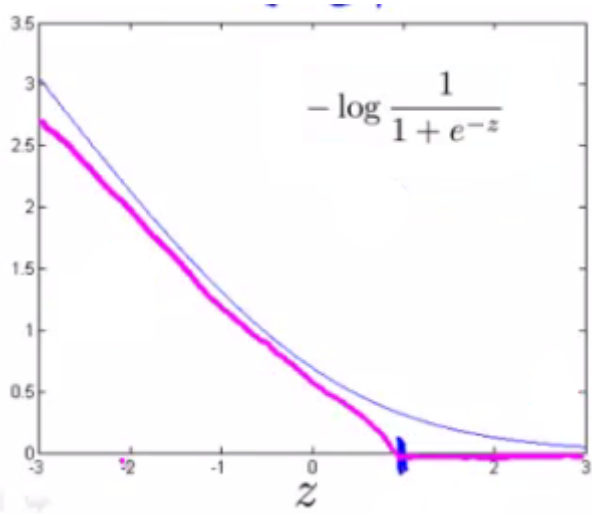


Same deal, if z is small then **the cost is low**. But if z is large then **the cost is massive.**

SVM cost functions plots (from logistic regression cost functions)

To build a SVM we must redefine our cost functions.

In the first case when $y = 1$ (we want $h_\theta(x) \approx 1$, $\theta^T x >> 0$), we take the $y = 1$ function and create a new cost function, instead of a curved line create two straight lines (magenta) which acts as an approximation to the logistic regression $y = 1$ function:



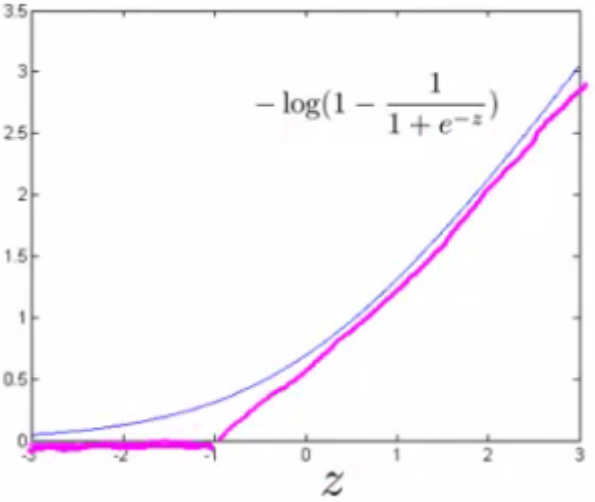
Take point 1 on the z axis, flat from 1 onwards, grows when we reach 1 or a lower number. This means we have two straight lines

- Flat when cost is 0
- Straight growing line after 1

So this is the new $y = 1$ **cost function**

- Gives the SVM a computational advantage and an easier optimization problem (that would be easier for software to solve)
- We call this function $\text{cost}_1(z)$

In the second case when $y = 0$ (we want $h_\theta(x) \approx 0$, $\theta^T x << 0$), do the equivalent with the $y = 0$ function plot



We call this function $\text{cost}_0(z)$, so here we define the two cost function terms for our SVM graphically.

Support Vector Machine

Logistic regression:

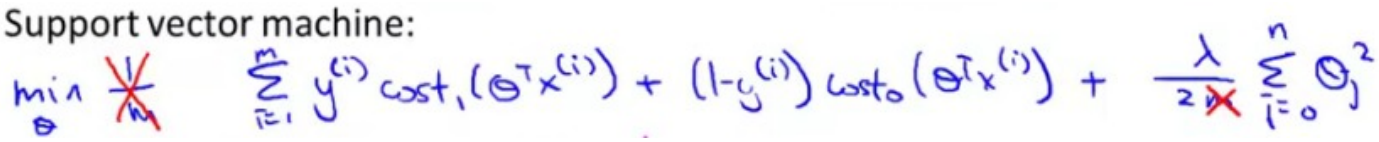
$$\min_\theta \frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \left(-\log h_\theta(x^{(i)}) \right) + (1 - y^{(i)}) \left((-\log(1 - h_\theta(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Support Vector Machine:

For the support vector machine what we're going to do is essentially take the first term where $y = 1$ and replace it with $\text{cost}_1(\theta^T X)$ and take the second term where $y = 0$ and replace it with $\text{cost}_0(\theta^T X)$.

By convention, for the support of vector machine, we're actually write things slightly different. First, we're going to remove the 1 / m term, and this just this happens to be a slightly different convention that people use for support vector machines compared to logistic regression.

By removing 1 / m we should get the same optimal values (1 / m is a constant, so should get same optimization).



For logistic regression we had two terms;

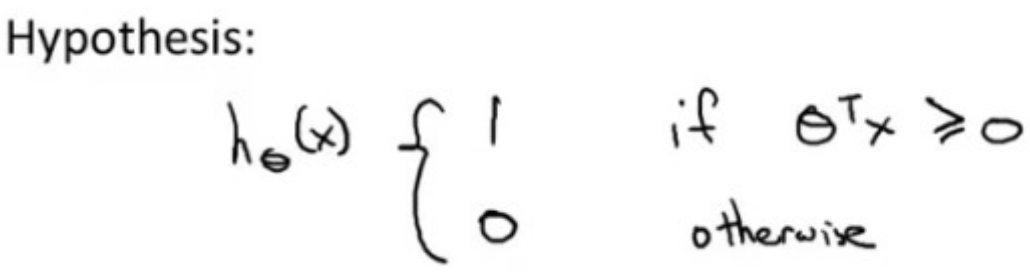
- Training data set term (i.e. that we sum over m) = A
- Regularization term (i.e. that we sum over n) = B
  - So we could describe it as $A + \lambda B$
  - Need some way to deal with the trade-off between regularization and data set terms
  - Set different values for λ to parametrize this trade-off
- Instead of parameterization this as $A + \lambda B$
  - For SVMs the convention is to use a different parameter called C
  - So do $CA + B$
  - If C were equal to 1 / λ then the two functions ($CA + B$ and $A + \lambda B$) would give the same value.
  - These two optimization objectives ($CA + B$ and $A + \lambda B$) should give us the same value, the same optimal value for theta

So this is just a different way of controlling the trade off, it's just a different way of prioritizing how much we care about optimizing **the first term**, versus how much we care about optimizing **the second term.**

So that gives us our overall optimization objective function for the support vector machine. And if we minimize that function, then what we have is the parameters learned by the SVM.

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

Unlike logistic, $h_\theta(x)$ doesn't give us a probability, but instead we get a direct prediction of 1 or 0



Video Question: Consider the following minimization problems:

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

These two optimal problems will give the same value of θ (i.e., the same value of θ gives the optimal solution to both problems) if:

- $C = \lambda$
- $C = -\lambda$

> $C = \frac{1}{\lambda}$

- $C = \frac{2}{\lambda}$