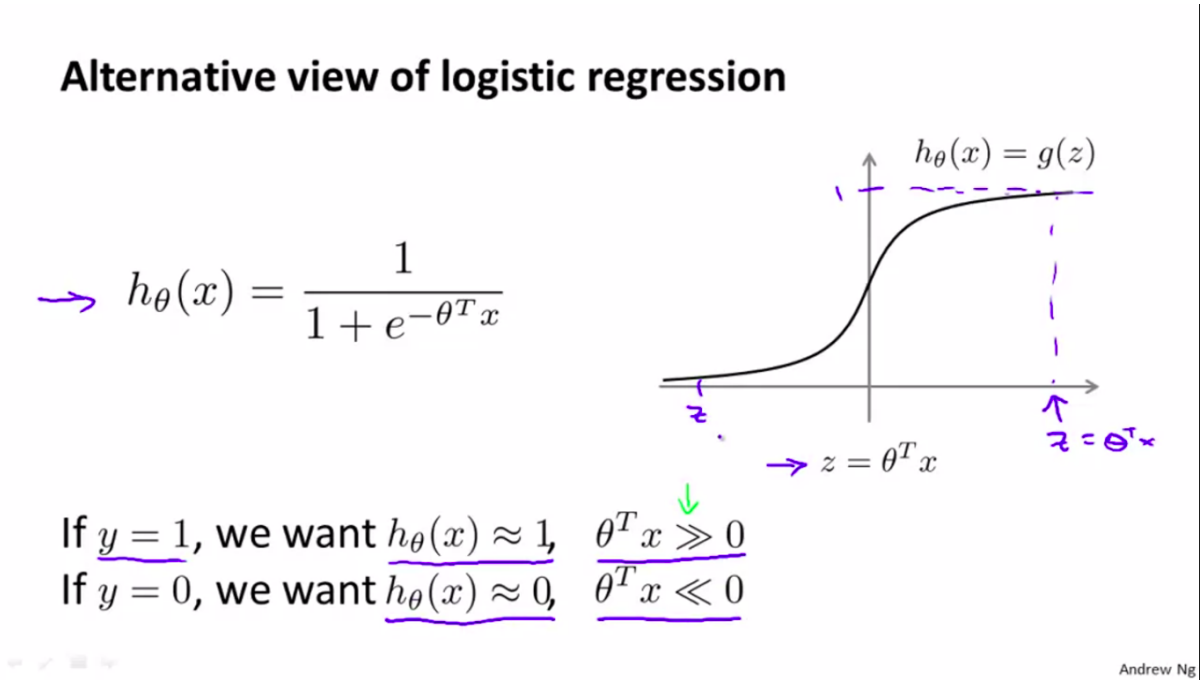
Week 7 - Lecture 1

Optimization Objective

Let's start considering a Logistic Regression and how we can transform it into a Support Vector Machine (SVM).

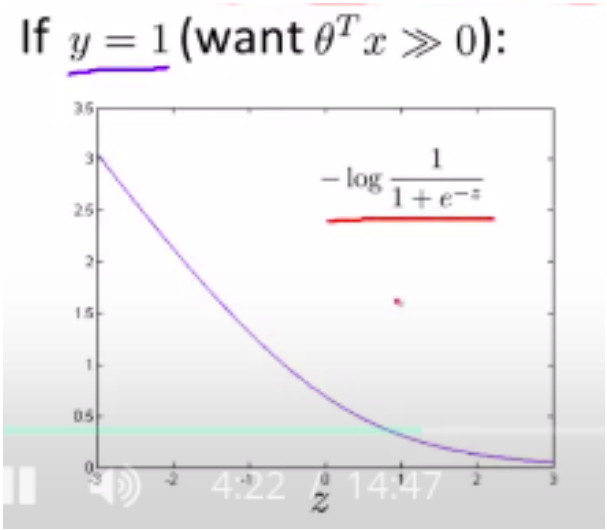


Considering the cost function, we can say that **each example** increases the cost by this factor:

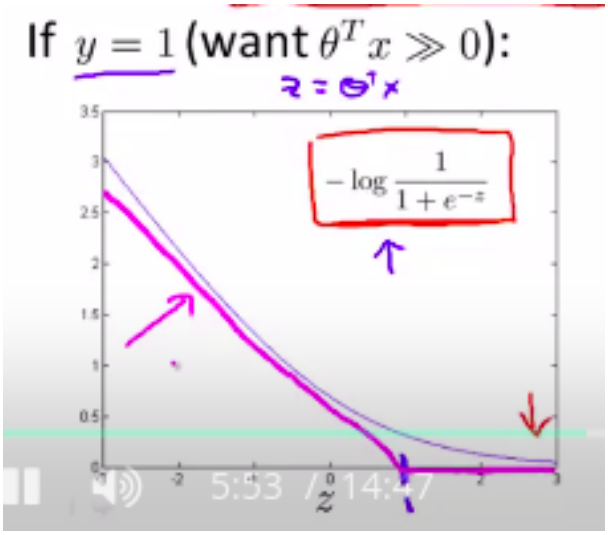
Cost of example:

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

If $y = 1$, we want that $\theta^T x \gg 0$, i.e., z much greater than 0.

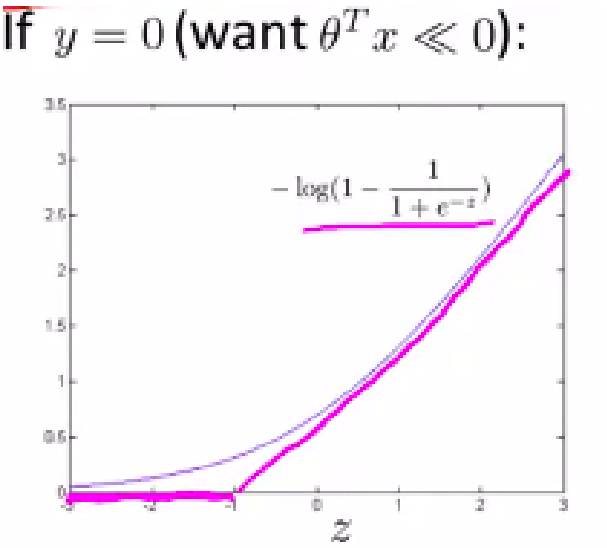


For a Support Vector Machine, we're going to modify this cost function a little bit to be like the magenta line in the following chart:



From this moment on, we're going to denote this function by: $cost_1(z)$

This will give us some computational advantages when working with SVM's. Analogously, for the case $y = 0$, we have something similar to:



And we're denoting this by: $cost_0(z)$

> **Cost function for Logistic Regression:**
> $\min_\theta \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log(h_\theta(x^{(i)})\right) + (1-y^{(i)})\left(-\log(1-h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

For the Support Vector Machine, the cost function will be:

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

For convention and practical reasons, we're going to modify it a little bit...

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

> **Note:** getting rid of the $\frac{1}{m}$ constant won't change de optimal parameters θ, i.e., we should get the same minimization solution.

Aditionally, we're going to parameterize it a bit different compared to logistic regression.

> **Parameterization of the logistic regression cost function:** $A + \lambda B$, where the expressions for A and B can be deducted easily just by comparing this simplified expression with the cost function.

For SVM's, we're going to parameterize like: $cA + B$ and, in this case, c plays a role similar to $c = \frac{1}{\lambda}$ (it still means how well we want to fit our model to the sample data). Furthermore, setting $c = \frac{1}{\lambda}$ guarantees that the solution to the minimization problem will be the same as in the Logistic Regression case.

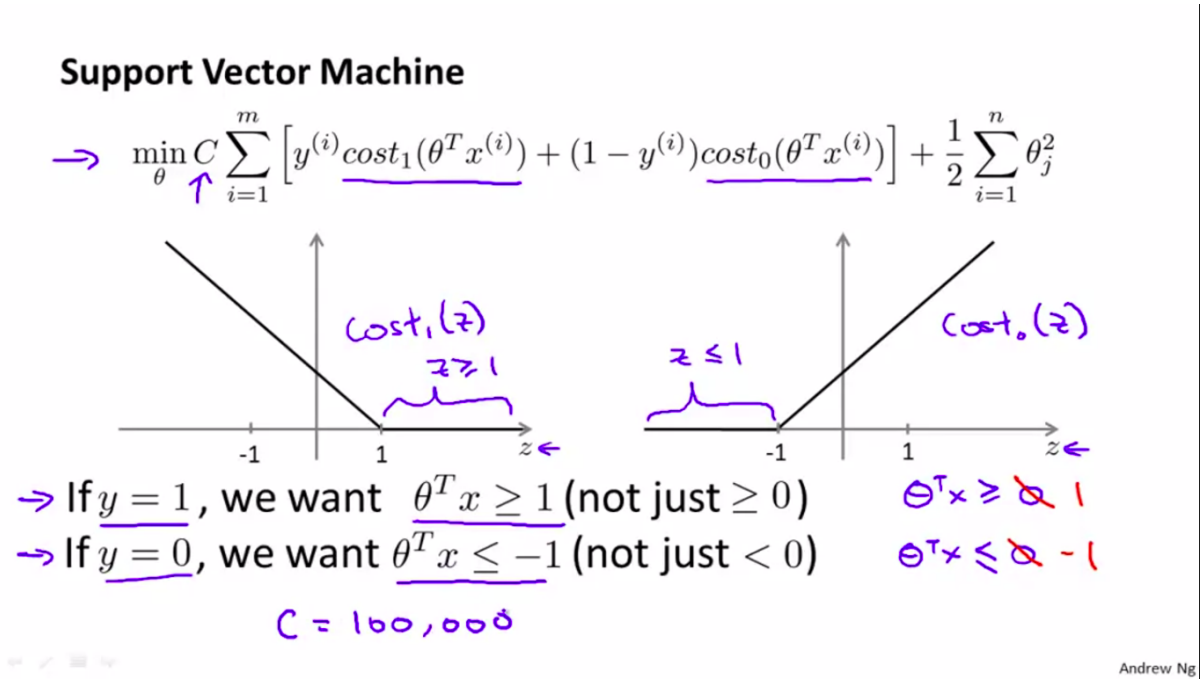
Overall, our cost function will be:

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1 \left(\theta^T x^{(i)} \right) + (1 - y^{(i)}) cost_0 \left(\theta^T x^{(i)} \right) \right] + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2 \tag{3}$$

> **Note:** SVM's doesn't output probabilities. Instead, the hypothesis is
>
> $$h_\theta(x) = \begin{cases} 1 \text{ if } \theta^T x \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Large Margin Intuition

> **Note:** Support Vector Machines (SVMs) are also known as Large Margin Classifiers

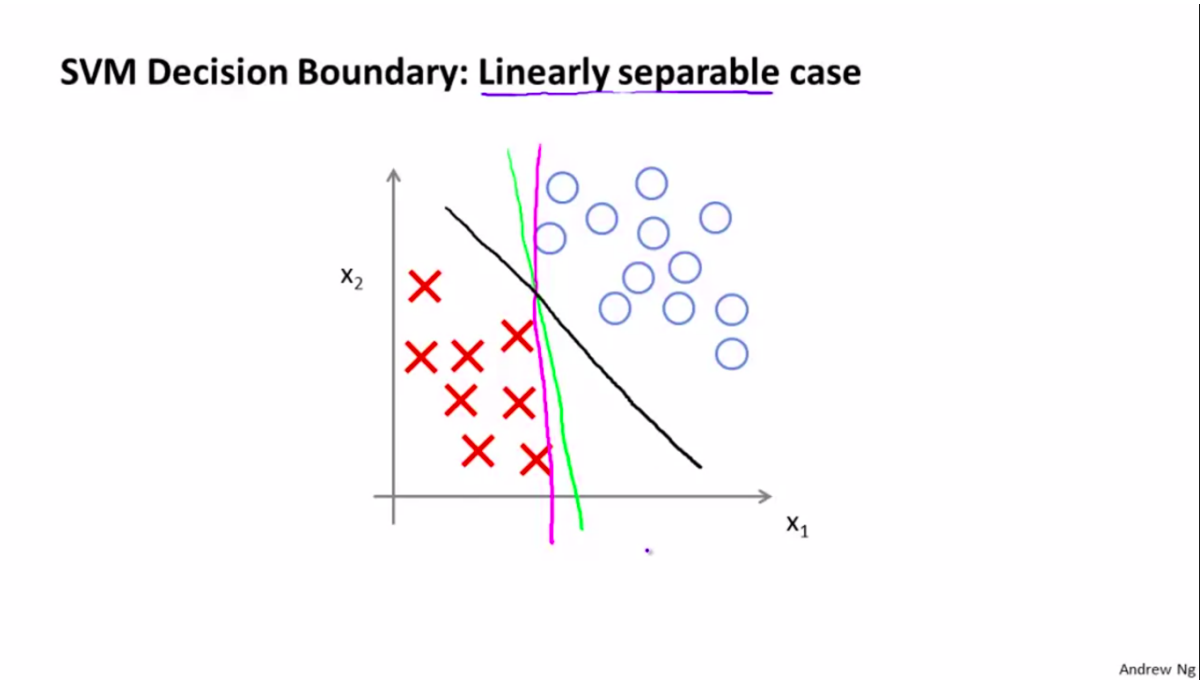


By looking at this problem, we can easily see that:

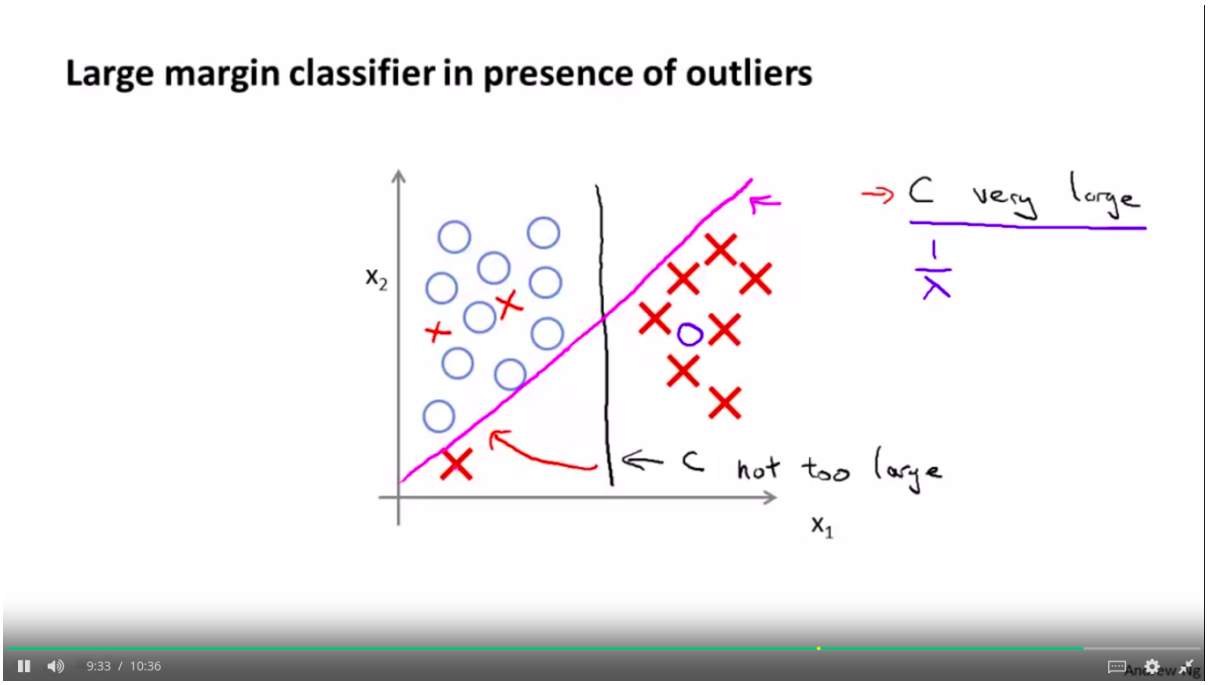
- Whenever $y^{(i)} = 1$, to minimize the cost function, we need that $cost_1(z) = 0$. This will happen when $\theta^T x^{(i)} \geq 1$.
- Whenever $y^{(i)} = 0$, to minimize the cost function, we need that $cost_0(z) = 0$. This will happen when $\theta^T x^{(i)} \leq -1$.

So, in this situation, we're minimizing:

$$\begin{aligned} &\min_\theta C \cdot 0 + \tfrac{1}{2} \textstyle\sum_{j=1}^{n} \theta_j^2 \\ &s.t.\ \theta^T x^{(i)} \geq 1, \text{ if } y^{(i)} = 1 \\ &\quad \theta^T x^{(i)} \leq -1 \text{ if } y^{(i)} = 0 \end{aligned} \tag{4}$$



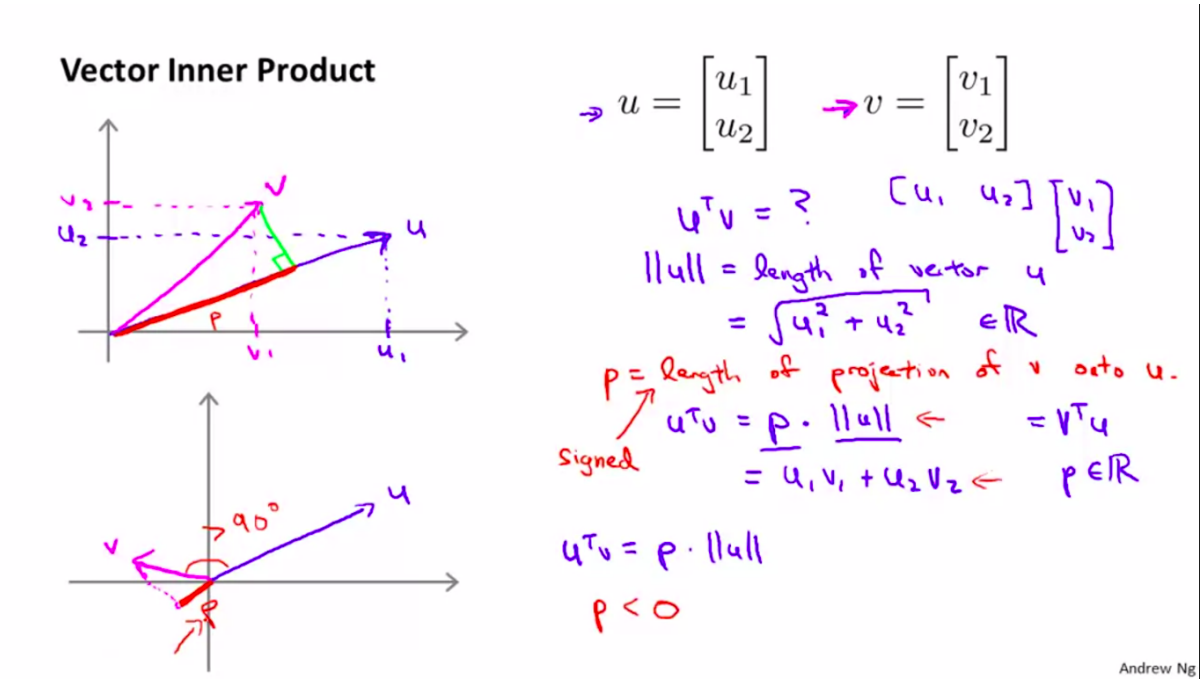
Note: The SVM will tend to choose the decision boundary drawn in black (the one that has the higher margin or distance from any example point)



With a large value of C, our SVM still do a pretty decent job at separating these data points, even in the presence of outliers (i.e., even if the data is not linearly separable).

Mathematics Behind Large Margin Classification

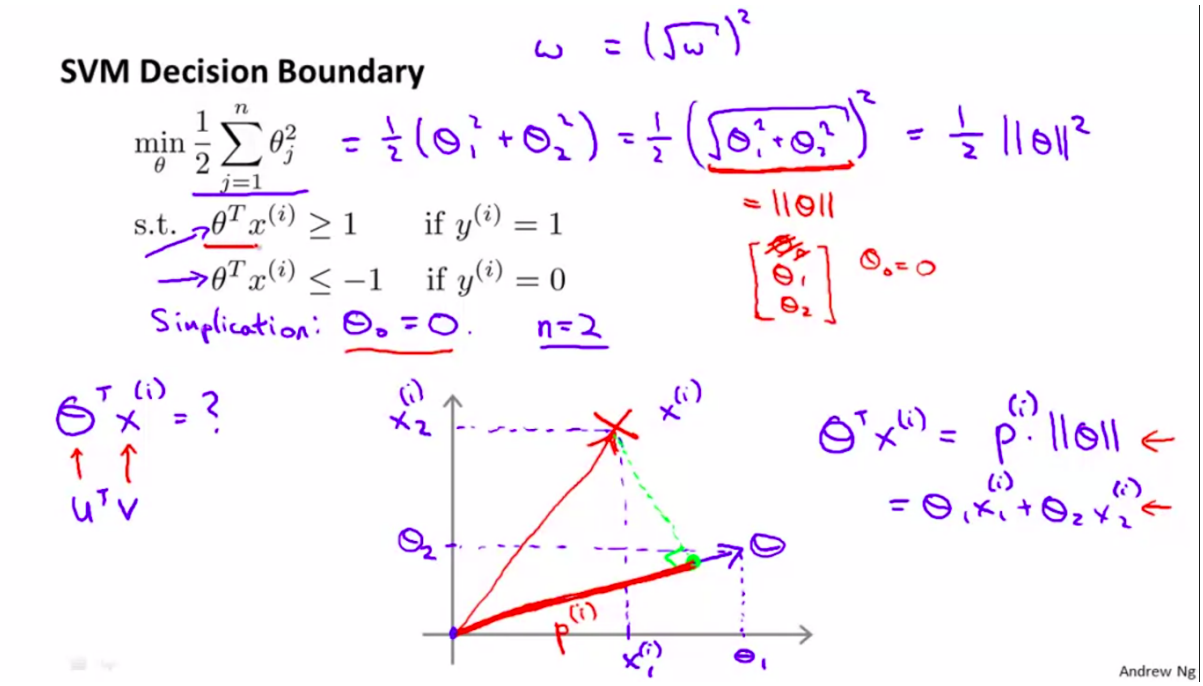
Note from the author: Opposing to what professor Ng did in this lecture, I'm going to just point out some topics that were covered in this section and the avid reader can just look it up and review these concepts on a good Linear Algebra Book.



Topics:

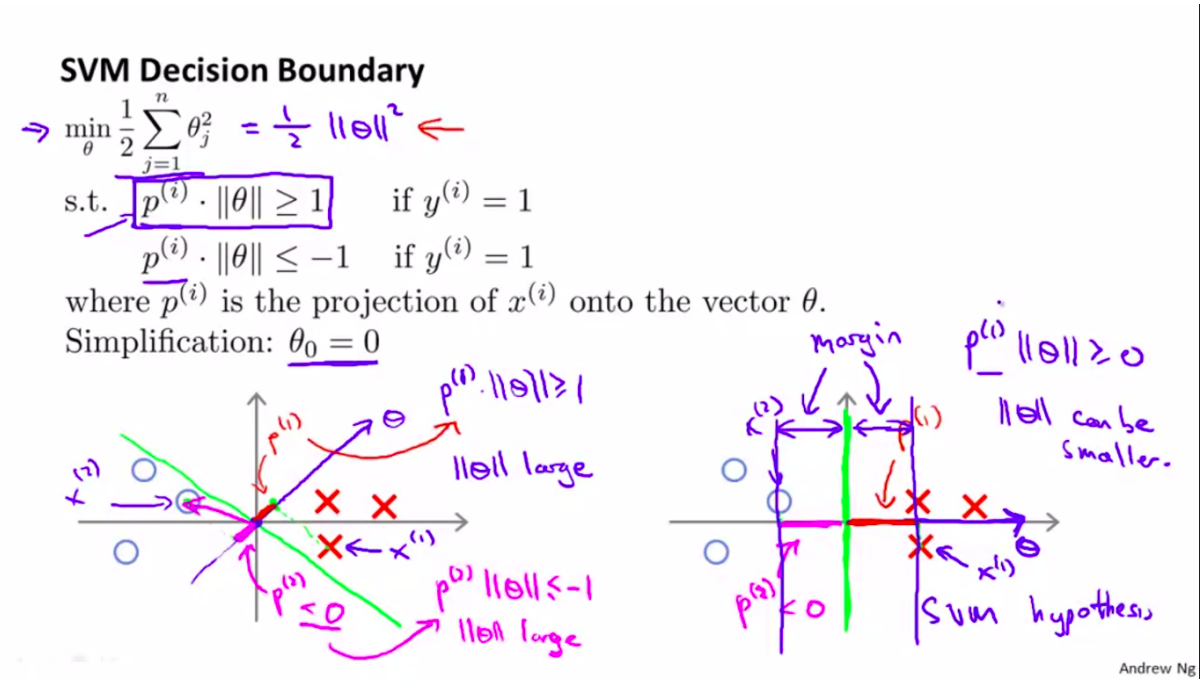
- Inner product (vectors)
- Norm of vectors

Lecture slides:



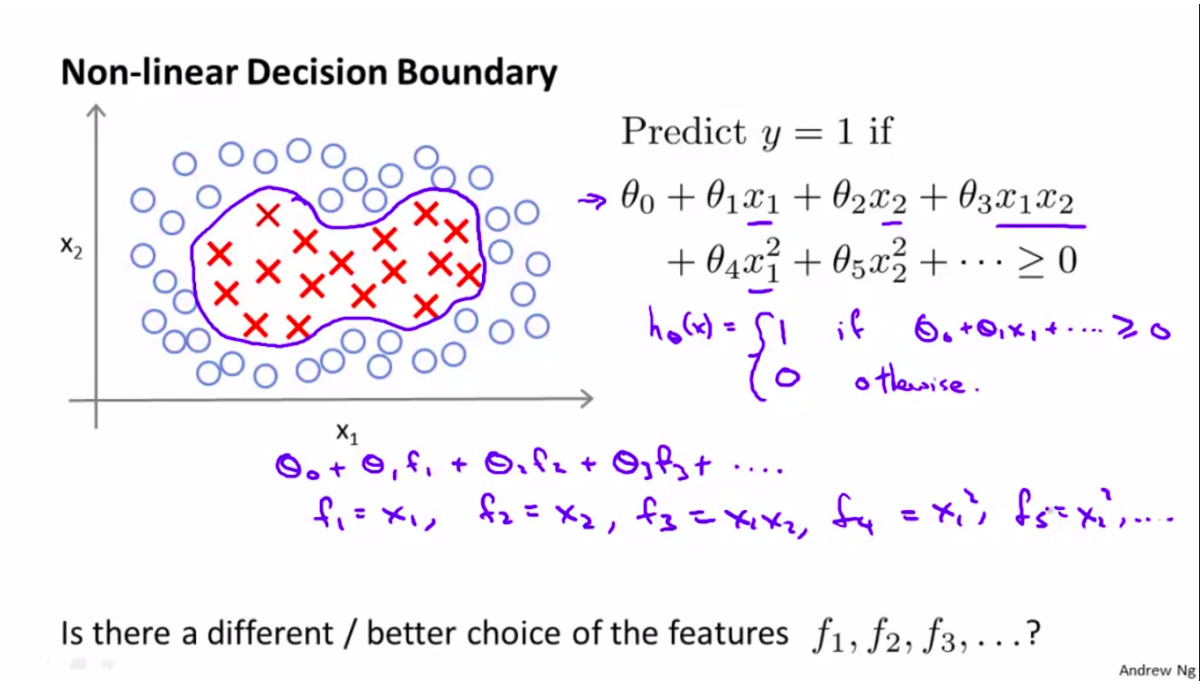
Intuition:

- If $p^{(i)}$ is small, to satisfy the constraint that $p^{(i)} \cdot \|\theta\| \geq 1$, we need $\|\theta\|$ to be large and this increases the value of our cost function which, for instance, we want to minimize. The same rationale applies to the other constraint $p^{(i)} \cdot \|\theta\| \leq -1$. If $p^{(i)}$ is a small negative number, we still need $\|\theta\|$ to be large.
- In contrast, $p^{(i)}$ is large, $\|\theta\|$ can be small and we can minimize the value of the cost function $\frac{1}{2}\|\theta\|^2$



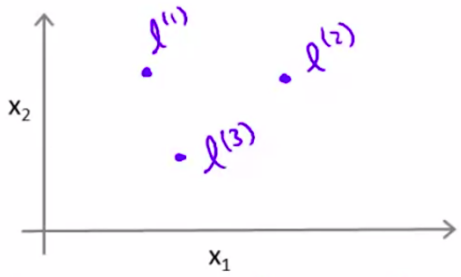
Kernels I

In this section we're going to expand SVMs a little bit to construct more complicated non-linear classifiers.



Now we're going to see how to define new features...

Kernel



Given x, compute new feature depending on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

Given x:

$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

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

$f_3 = \text{similarity}(x, l^{(3)}) = \exp(\dots)$

Kernel (Gaussian kernels) $k(x, l^{(i)})$

Andrew Ng

Note: the landmarks $l^{(1)}, l^{(2)}, l^{(3)}, \dots$ are manually chosen.

Intuition:

Kernels and Similarity

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

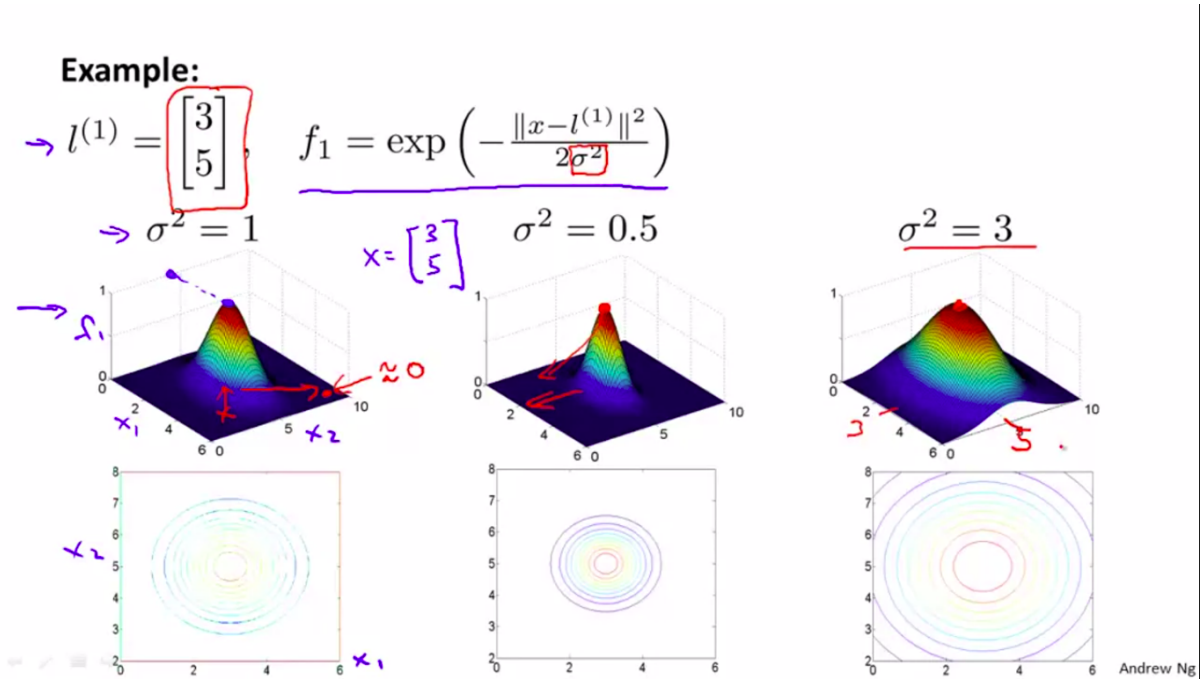
If $x \approx l^{(1)}$:

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

If x if far from $l^{(1)}$:

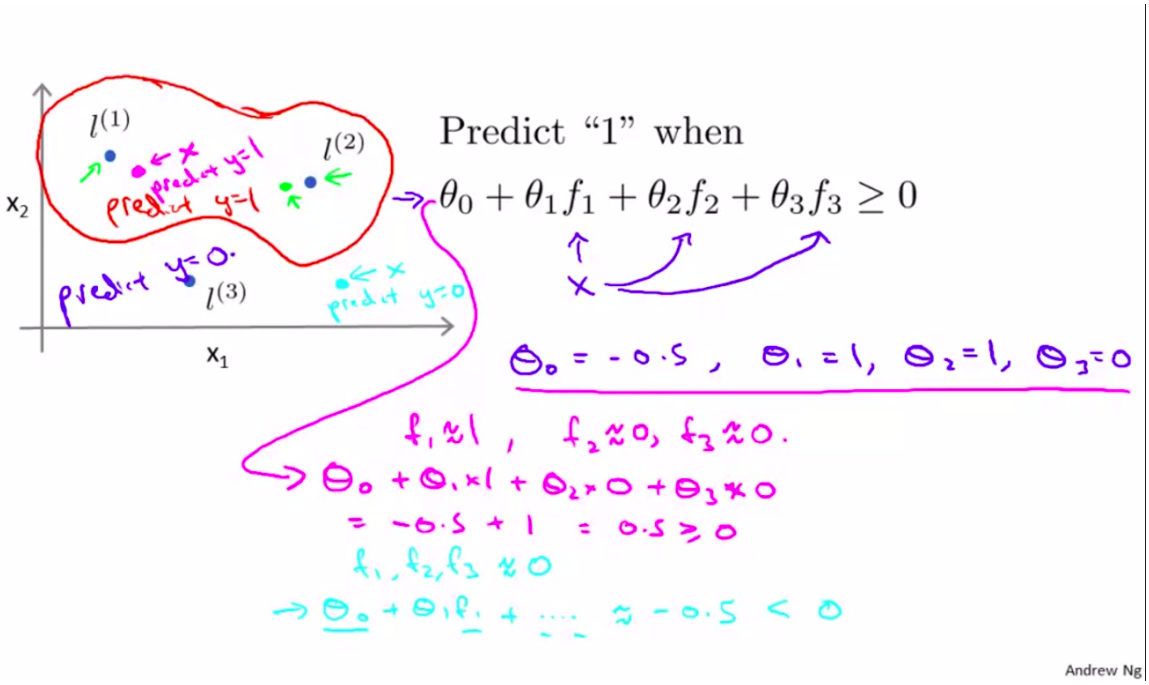
$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$

Andrew Ng



Basically the graphs show how far x is from the landmark $l^{(i)}$. The parameter σ affects the speed in which the value of the function (kernel) increases/decreases as x moves closer/away from the landmark.

Now lets see what sorts of hipothesys we can learn with these new features...



Kernels II

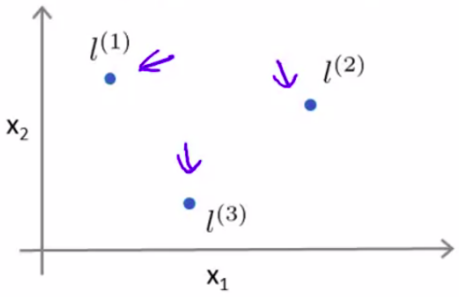
From the previous section, there are still some questions that need to be answered:

- How do we choose these landmark points?
- What kernels can we use other than the Gaussian one?

Answers:

Given some training examples, what we're going to do is to put landmarks exactly where our examples are.

Choosing the landmarks

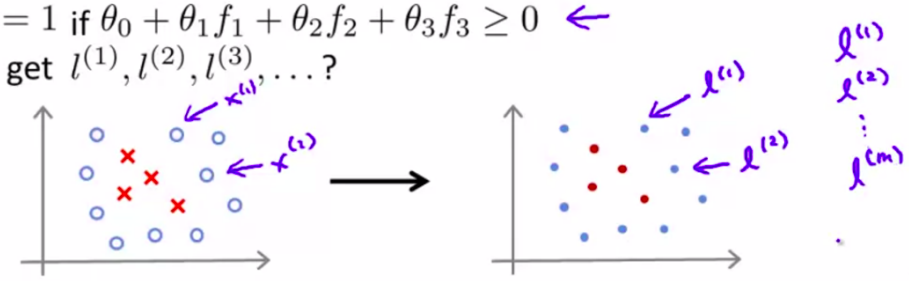


Given x:

$$f_i = \text{similarity}(x, l^{(i)}) = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$$

Predict $y = 1$ if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \ldots$?



Andrew Ng

Formally...

SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$,

choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$.

Given example x:

$f_1 = \text{similarity}(x, l^{(1)})$

$f_2 = \text{similarity}(x, l^{(2)})$

$\ldots$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

For training example $(x^{(i)}, y^{(i)})$:

$x^{(i)} \in \mathbb{R}^{n+1}$ (or $\mathbb{R}^n$)

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \quad f_0^{(i)} = 1$$

$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)})$

$f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$

$f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp(-\frac{0}{2\sigma^2}) = 1$

$\text{sim}(x^{(i)}, l^{(m)})$

Andrew Ng

SVM with Kernels

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$

→ Predict "y=1" if $\theta^T f \geq 0$

Training:

$$\min_\theta C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

$\theta^T f^{(i)}$

$\sum_{j=1}^{m} \theta_j^2 = \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$ (ignore θ_0)

$\theta^T M \theta$

Andrew Ng

SVM parameters:

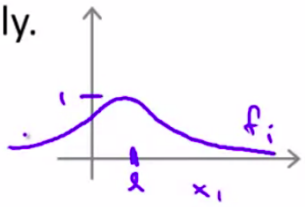
C ($= \frac{1}{\lambda}$). → Large C: Lower bias, high variance. (small λ)

→ Small C: Higher bias, low variance. (large λ)

σ^2 Large σ^2: Features f_i vary more smoothly.

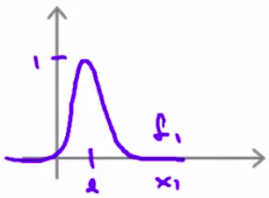
→ Higher bias, lower variance.

$\exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$



Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.



Andrew Ng

SVMs in practice

This section brings some comments on the implementation of SVMs.

Use SVM software package (e.g. liblinear, libsvm, …) to solve for parameters θ.

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

E.g. No kernel ("linear kernel")

Predict "y = 1" if $\theta^T x \geq 0$

$\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \geq 0$

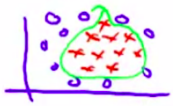
→ n large, m small $x \in \mathbb{R}^{n+1}$

Gaussian kernel:

$f_i = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$, where $l^{(i)} = x^{(i)}$.

Need to choose σ^2.

$x \in \mathbb{R}^n$, n small and/or m large



Kernel (similarity) functions:

$x^{(i)}$ $l^{(j)} = x^{(j)}$

```
function f = kernel(x1,x2)
    f = exp( -|| x1 - x2 ||^2 / (2*sigma^2) )
return
```

Note: Do perform feature scaling before using the Gaussian kernel.

Other choices of kernel

Note: Not all similarity functions $\text{similarity}(x, l)$ make valid kernels.

→ (Need to satisfy technical condition called "Mercer's Theorem" to make sure SVM packages' optimizations run correctly, and do not diverge).

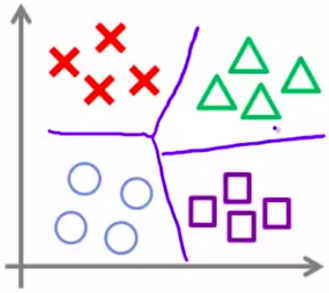
Many off-the-shelf kernels available:

- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$, $(x^T l + \text{constant})^{\text{degree}}$

- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, …

$sim(x, l)$

Multi-class classification



$$y \in \{1, 2, 3, \ldots, K\}$$

Many SVM packages already have built-in multi-class classification functionality.

Otherwise, use one-vs.-all method. (Train K SVMs, one to distinguish $y = i$ from the rest, for $i = 1, 2, \ldots, K$), get $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$

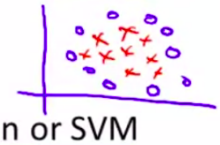
Pick class i with largest $(\theta^{(i)})^T x$

Andrew Ng

Logistic regression vs. SVMs

n = number of features ($x \in \mathbb{R}^{n+1}$), m = number of training examples

- If n is large (relative to m): (e.g. $n \geq m$, $n = 10{,}000$, $m = 10 \cdots 1000$)
  - Use logistic regression, or SVM without a kernel ("linear kernel")
- If n is small, m is intermediate: ($n = 1$-1000, $m = 10$-$10{,}000$)
  - Use SVM with Gaussian kernel
- If n is small, m is large: ($n = 1$-1000, $m = 50{,}000+$)
  - Create/add more features, then use logistic regression or SVM without a kernel
- Neural network likely to work well for most of these settings, but may be slower to train.



Andrew Ng

References ¶

[1] Machine Learning - Stanford University (https://www.coursera.org/learn/machine-learning)