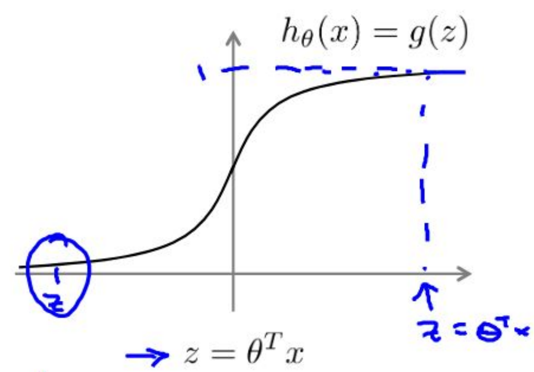
Week Seven
Support Vector Machines

We have learned logistic regression and neural network on supervised learning. Alongside these we have another algorithm named Support Vector Machines (SVM). Which gives a cleaner, and sometimes more powerful way of learning complex non-linear functions.

Optimization objective:

Alternative view of logistic regression

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



If $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$

If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

Andrew Ng

Alternative view of logistic regression

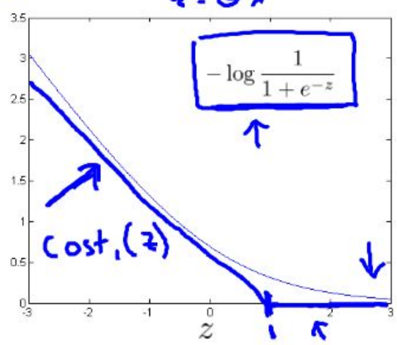
(x, y)

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

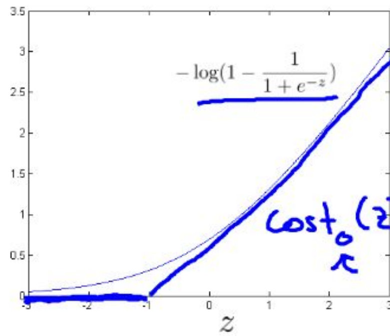
$$= \boxed{-y \log \frac{1}{1+e^{-\theta^T x}}} \boxed{-(1-y)\log\left(1-\frac{1}{1+e^{-\theta^T x}}\right)}$$

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

$z = \theta^T x$



If $y = 0$ (want $\theta^T x \ll 0$):



Andrew Ng

Support vector machine

Logistic regression:

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

Support vector machine:

$$\min_\theta \cancel{\frac{1}{m}}\, C \sum_{i=1}^m y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)}) + \frac{1\cancel{\lambda}}{2\cancel{m}}\sum_{i=0}^n \theta_j^2$$

$\min_u \left((u-5)^2+1\right) \times 10 \rightarrow u=5$

$\min_u 10(u-5)^2+10 \rightarrow u=5$

$A + \lambda B$

$C A + B$

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

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

Andrew Ng

As m is a constant, so there will be no contribution on min theta. And if C=1/lambda, then the two equations above will give the same output (same min theta).

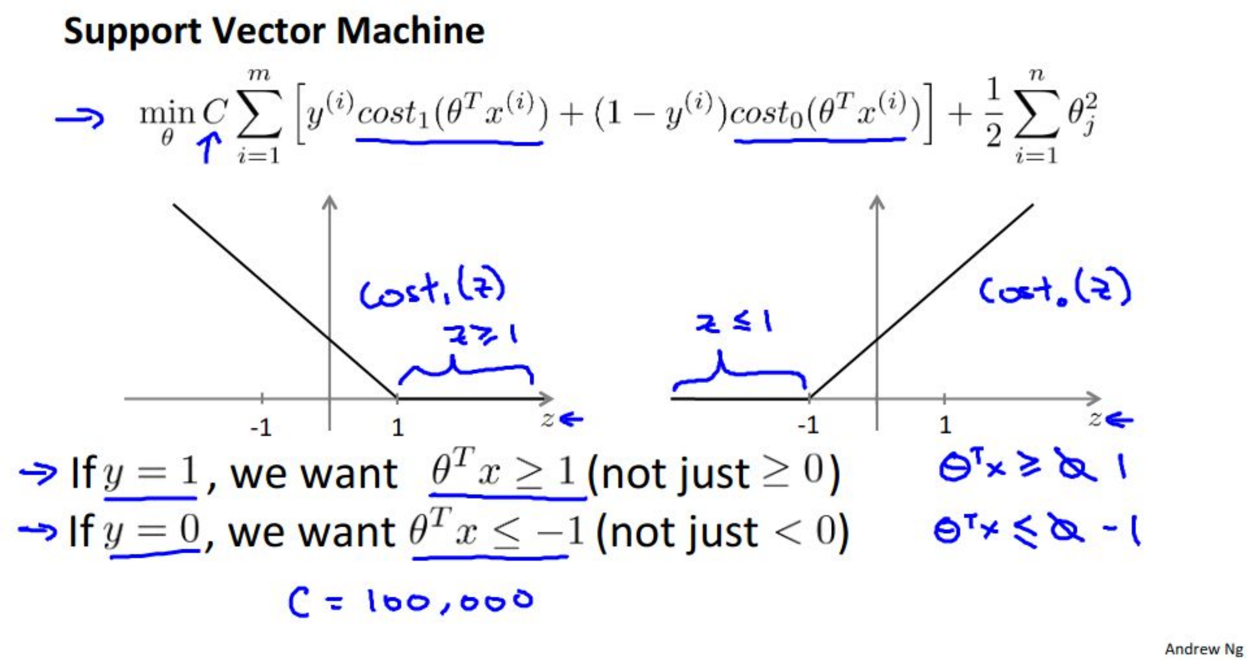
SVM hypothesis

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

Hypothesis:

Large Margin Intuition :

In support vector machine we wanna be more confident than logistic regression. So we choose a high threshold value than before.



In the picture below, if we wanna set the boxed part of the equation to zero we have two way:

SVM Decision Boundary

$$\min_\theta C \underbrace{\boxed{\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right]}}_{=0} + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

Whenever $y^{(i)} = 1$:

$\Theta^T x^{(i)} \geq 1$

$$\min_\Theta \cancel{C \times 0} + \frac{1}{2}\sum_{i=1}^{n}\Theta_j^2$$

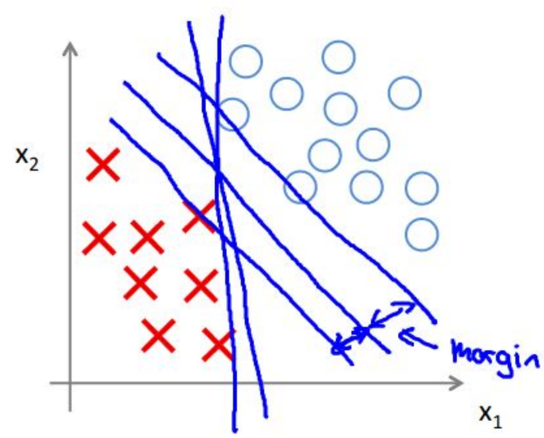
s.t. $\Theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

$\Theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$.

Whenever $y^{(i)} = 0$:

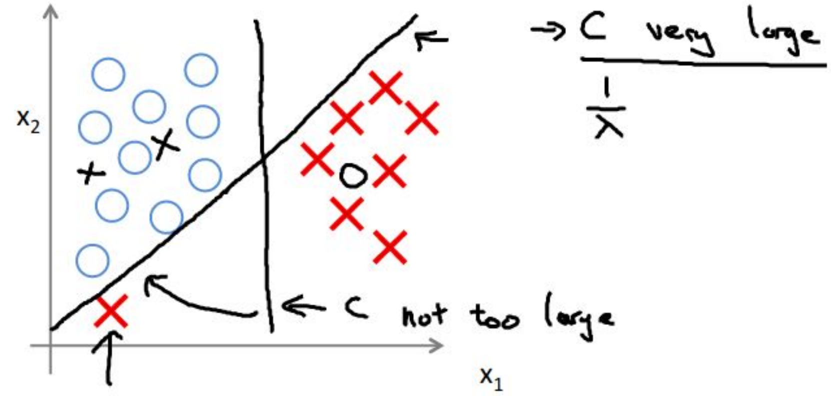
$\Theta^T x^{(i)} \leq -1$

SVM Decision Boundary: Linearly separable case



Large margin classifier

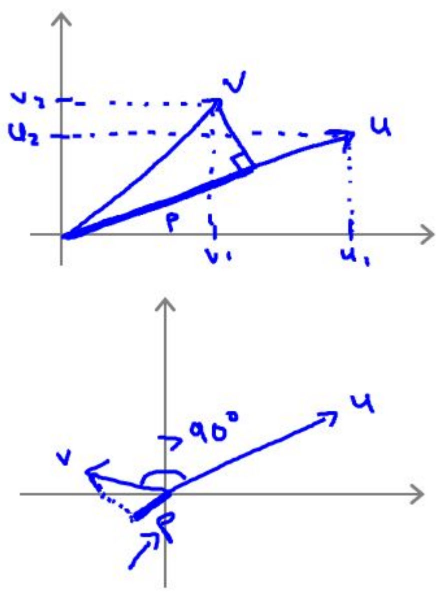
Large margin classifier in presence of outliers



Andrew Ng

The mathematics behind large margin :

Vector Inner Product



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

$u^T v = ?$ $\quad [u_1 \;\; u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

$\|u\|$ = length of vector u

$= \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

p = length of projection of v onto u.

signed

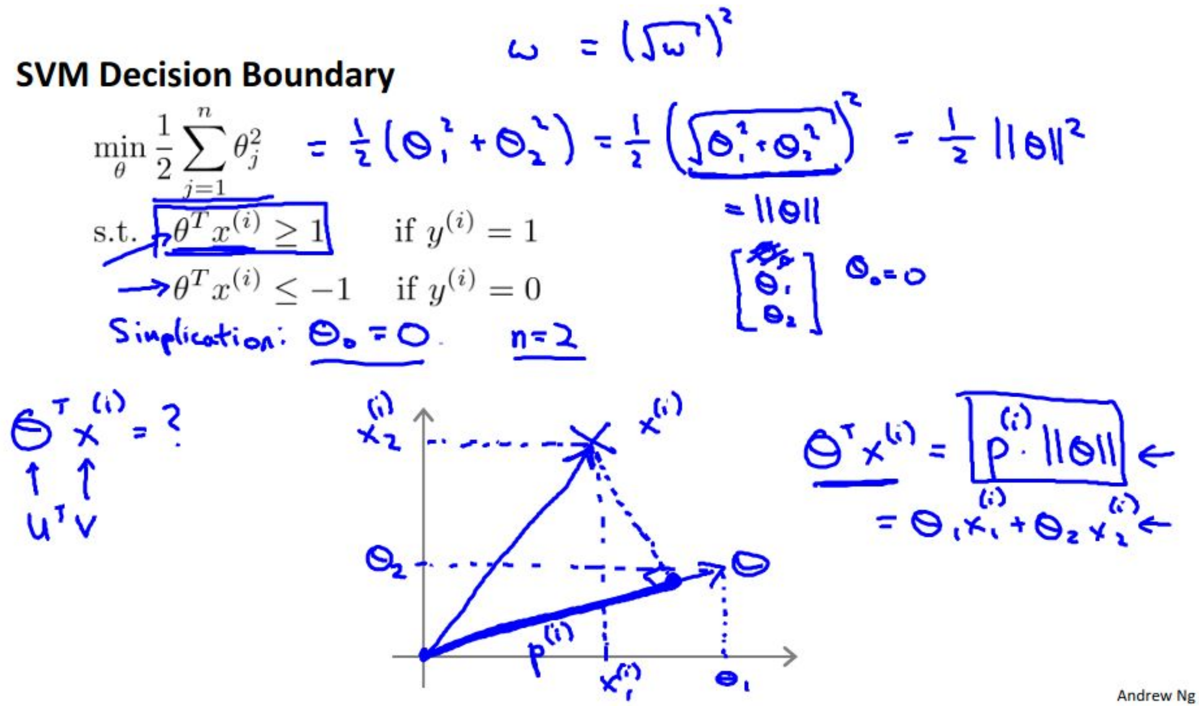
$u^T v = p \cdot \|u\| \leftarrow \qquad = v^T u$

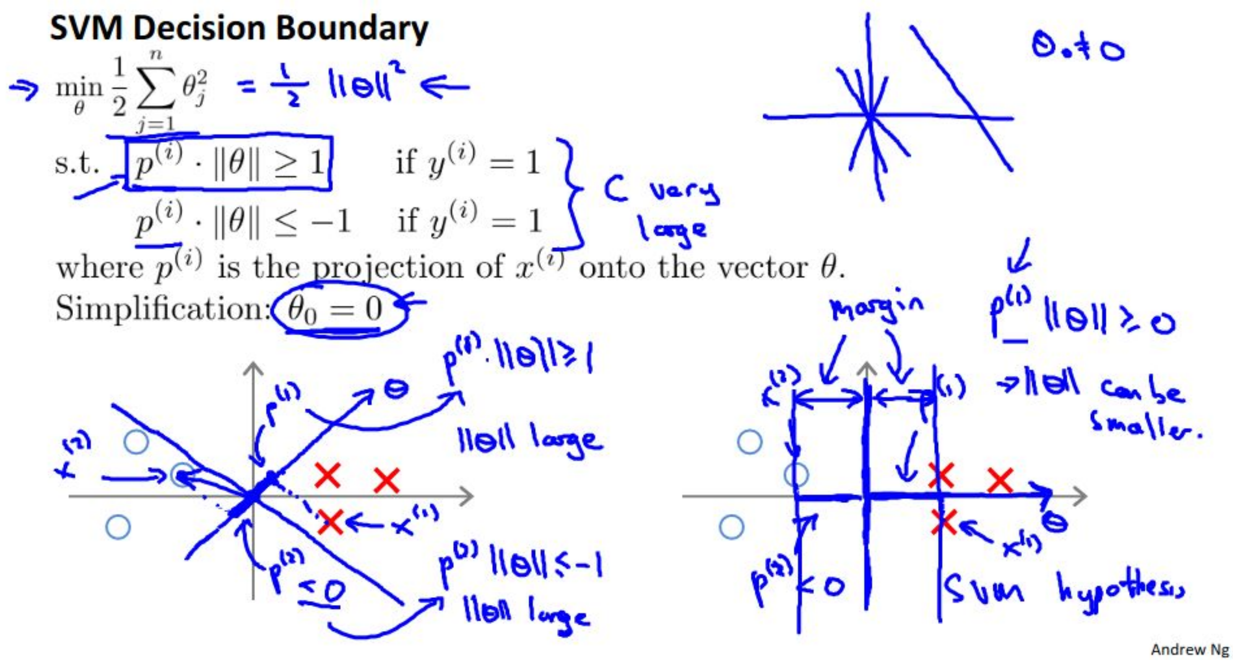
$= u_1 v_1 + u_2 v_2 \leftarrow \qquad p \in \mathbb{R}$

$u^T v = p \cdot \|u\|$

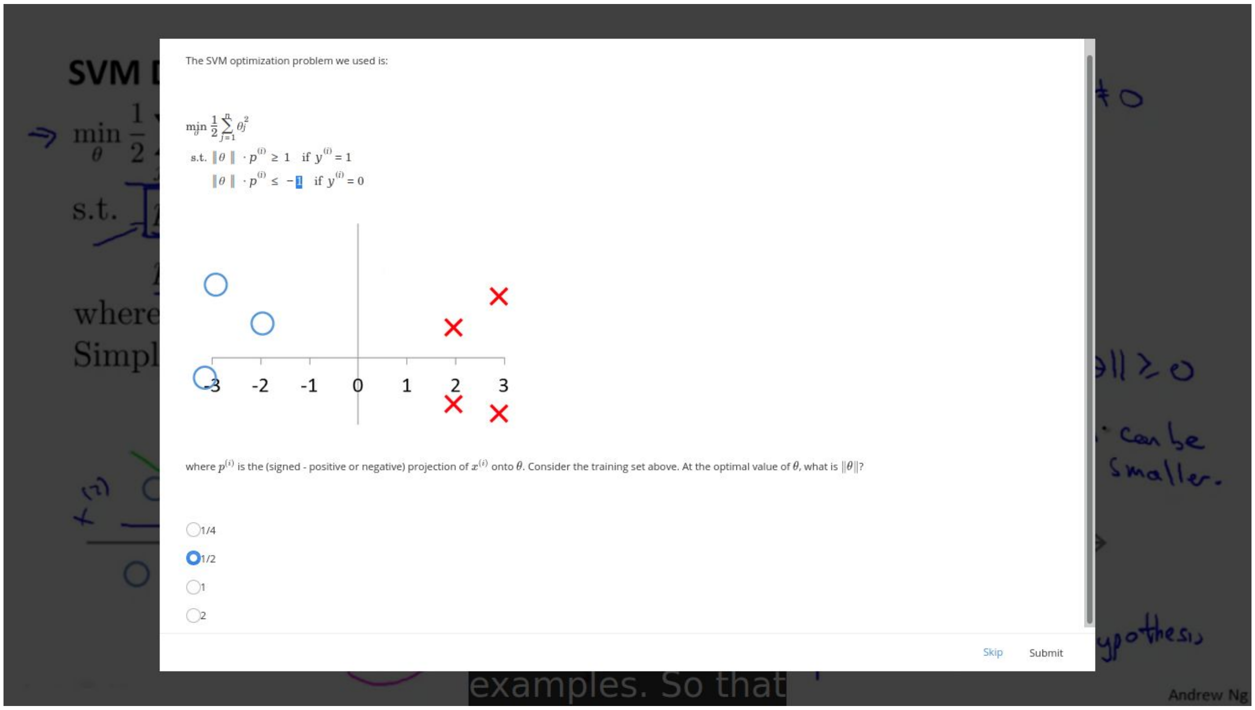
$p < 0$

Andrew Ng



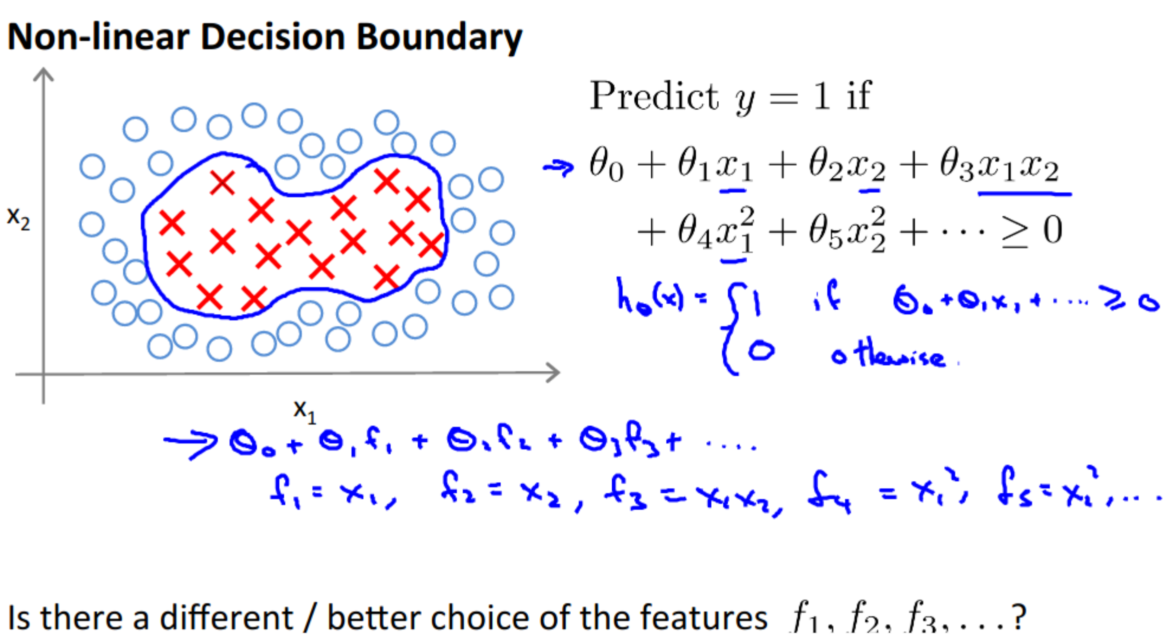


If you have any confusion how decision boundary is perpendicular to theta then you can go here and see Q3 to find the answer.

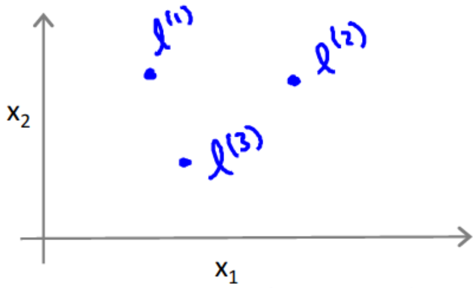


If you can get the clue to solve this quiz then go to [here]() and see Q4b.

Kernels I :



Kernel



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

$\|w\|$

Given x:

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

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

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

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

Kernels and Similarity

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

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

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

$l^{(1)} \rightarrow f_1$

$l^{(2)} \rightarrow f_2$

$l^{(3)} \rightarrow f_3$.

x

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

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

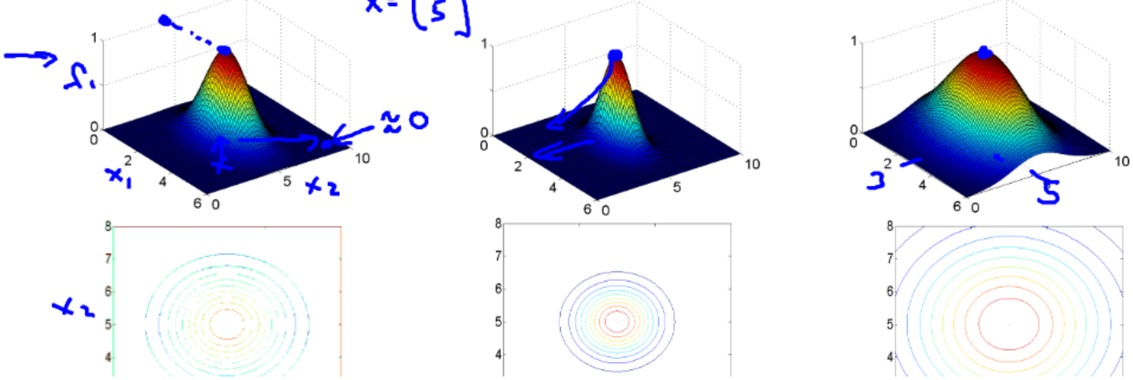
Andrew Ng

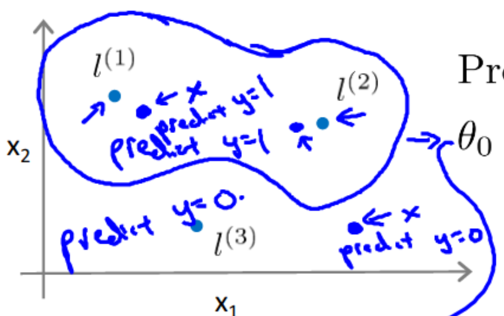
In the equation sigma squared is the parameter of the Gaussian kernel and as you vary it, you get slightly different effects. As we see with sigma squared the width of the graph is increasing.

Example:

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

$\sigma^2 = 1$ $\qquad x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$ $\qquad \sigma^2 = 0.5$ $\qquad \sigma^2 = 3$





Predict "1" when

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

x

$\theta_0 = -0.5, \quad \theta_1 = 1, \quad \theta_2 = 1, \quad \theta_3 = 0$

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

$\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$

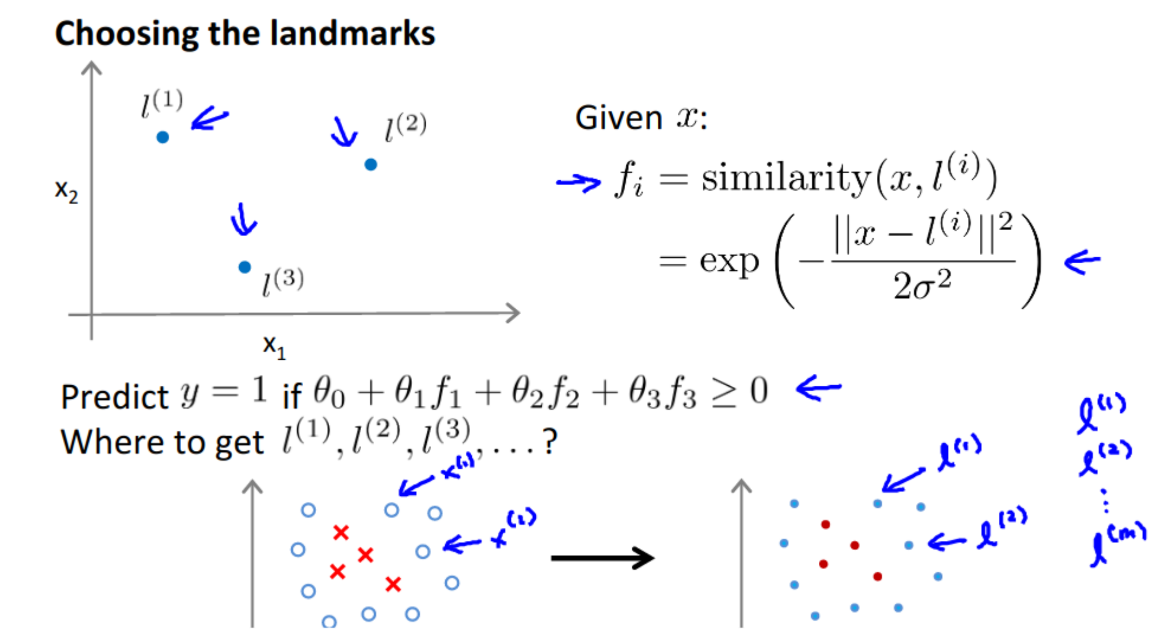
$= -0.5 + 1 = 0.5 \geq 0$

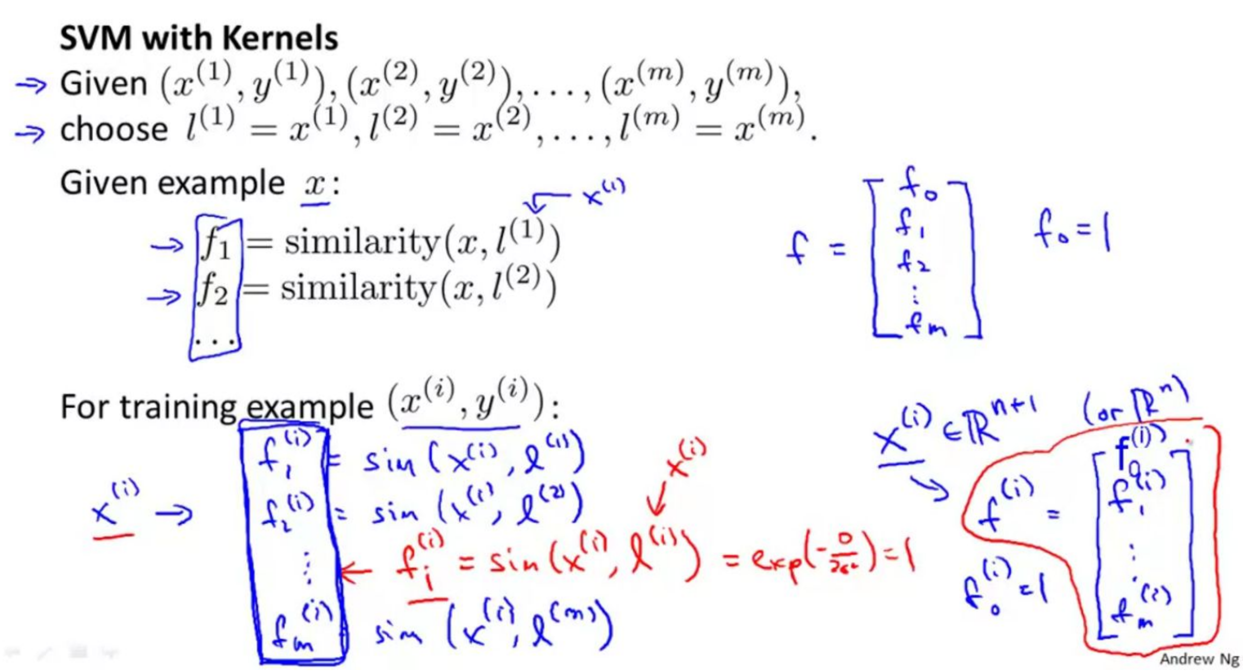
$f_1, f_2, f_3 \approx 0$

$\theta_0 + \theta_1 f_1 + \dots \approx -0.5 < 0$

Kernel II :

We will choose landmarks as much as our training sets.





SVM with Kernels

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

→ Predict "y=1" if $\theta^T f \geq 0$ $\qquad \theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$

Training: $\qquad n = m$

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

$\cancel{\theta^T x^{(i)}} \qquad \theta^T f^{(i)} \qquad \rightarrow \theta_0$

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

$\|\theta\|^2$

$\theta^T M \theta$ $\qquad M = 10{,}000$

SVM parameters:

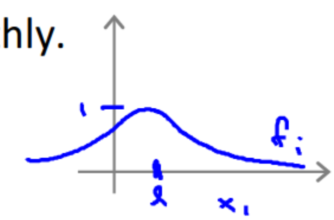
C ($= \frac{1}{\lambda}$). → Large C: Lower bias, high variance. (small λ)

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

σ^2 $\qquad$ Large σ^2: Features f_i vary more smoothly.

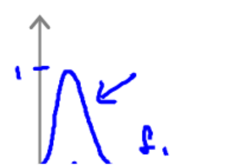
→ Higher bias, lower variance.

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



Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.



Using an SVM :

We will not write SVM algorithm from scratch. Rather than we will use a built-in library from many of them, like : liblinear, libsvm etc.

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 $\quad x \in \mathbb{R}^{n+1}$

Gaussian kernel:

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

Need to choose σ^2.

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

Andrew Ng

Kernel (similarity) functions:

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

```
function f = kernel(x1,x2)

    f = exp(-|| x1 - x2 ||^2 / (2σ^2))

return
```

$x \rightarrow f_1, f_2, \ldots, f_m$

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

$x \in \mathbb{R}^n$

$\boxed{||x - l||^2}$ $\quad v = x - l$

$||v||^2 = v_1^2 + v_2^2 + \cdots + v_n^2$

$= \underbrace{(x_1 - l_1)^2}_{1000\ \text{feet}^2} + \underbrace{(x_2 - l_2)^2}_{1\text{-}5\ \text{bedrooms}} + \cdots + (x_n - l_n)^2$

We need to use feature scaling technique if our features may vary in a big range. For example we have a feature which describes flat size another is number of rooms then we need to scale them.

Besides these linear, or Gaussian kernel we have some other kernel like Polynomial kernel, string kernel and so on.

Other choices of kernel

Note: Not all similarity functions $\mathrm{similarity}(x, l)$ 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:

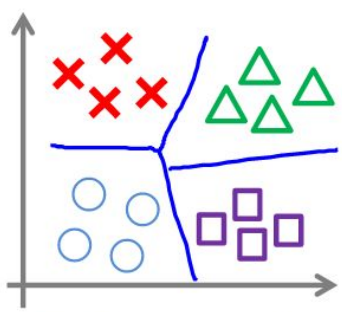
$(x^T l + \text{constant})^{\text{degree}}$

- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$

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

$\mathrm{sim}(x, l)$

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

$y=1$ $\quad y=2$ $\quad \cdots \quad$ $\theta = K$

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