

Week Seven

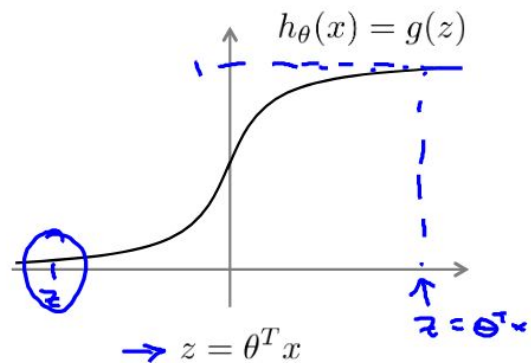
Support Vector Machines

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

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



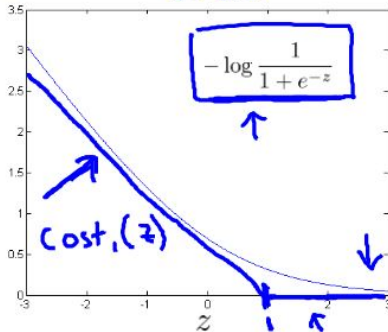
If $y = 1$, we want $h_{\theta}(x) \approx 1$, $\theta^T x \gg 0$
If $y = 0$, we want $h_{\theta}(x) \approx 0$, $\theta^T x \ll 0$

Alternative view of logistic regression

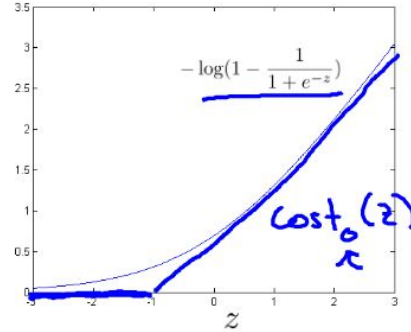
Cost of example: $-(y \log h_{\theta}(x) + (1 - y) \log(1 - h_{\theta}(x)))$ ←

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

If $y = 1$ (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:

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

Support vector machine:

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

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

$A + \lambda B \leftarrow$
 $C A + B \leftarrow$
 $C = \frac{1}{\lambda}$

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

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

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

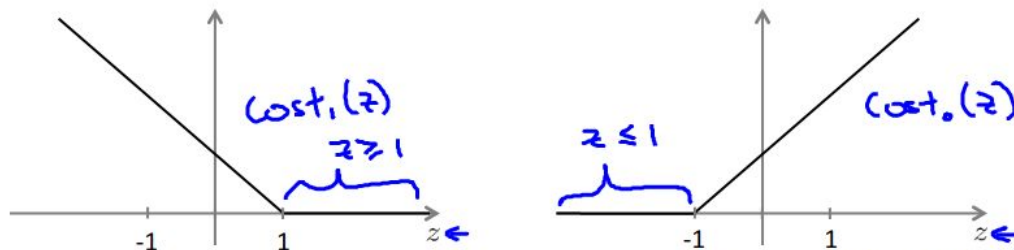
Hypothesis:

Large Margin Intuition :

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

Support Vector Machine

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



\rightarrow If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

$$\theta^T x \geq 1$$

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

$$\theta^T x \leq -1$$

$$C = 100,000$$

Andrew Ng

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

SVM Decision Boundary

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

$\leftarrow = 0$

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

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

$$\min_{\theta} C + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

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

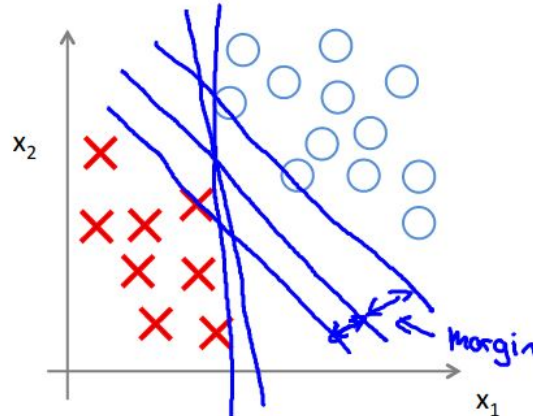
$$\theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0$$

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

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

Andrew Ng

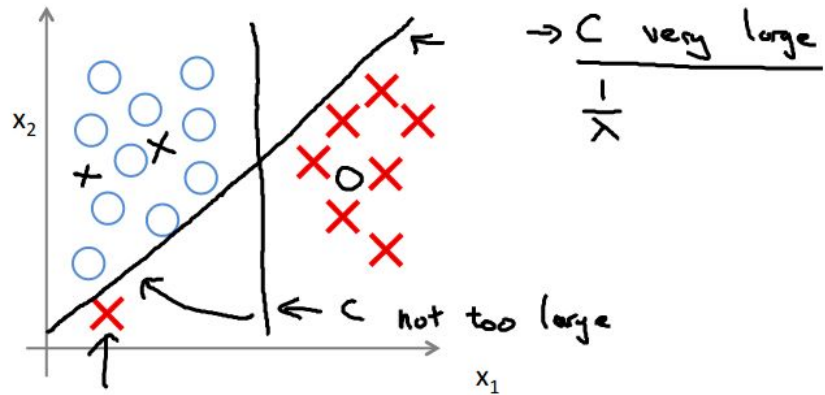
SVM Decision Boundary: Linearly separable case



Large margin classifier

Andrew Ng

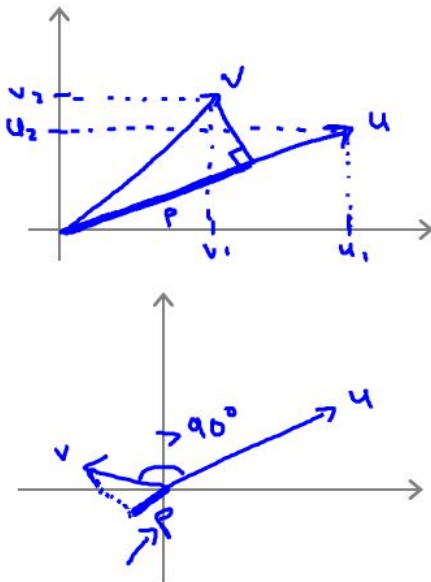
Large margin classifier in presence of outliers



Andrew Ng

The mathematics behind large margin :

Vector Inner Product



$$\begin{aligned} \rightarrow u &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \rightarrow v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ u^T v &= ? \quad [u_1 \ u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ \|u\| &= \text{length of vector } u \\ &= \sqrt{u_1^2 + u_2^2} \in \mathbb{R} \\ p &= \text{length of projection of } v \text{ onto } u. \\ \text{Signed } u^T v &= \underline{p} \cdot \|u\| \leftarrow = v^T u \\ &= u_1 v_1 + u_2 v_2 \leftarrow p \in \mathbb{R} \\ u^T v &= p \cdot \|u\| \\ p &< 0 \end{aligned}$$

Andrew Ng

SVM Decision Boundary

$$\omega = (\sqrt{\omega})^2$$

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} (\sqrt{\theta_1^2 + \theta_2^2})^2 = \frac{1}{2} \|\theta\|^2$$

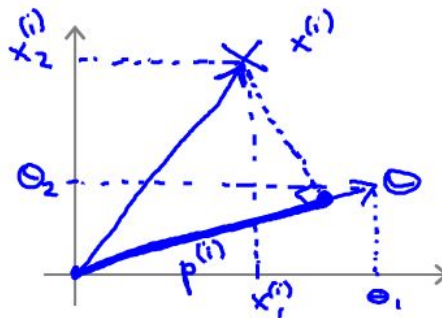
s.t. $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$
 $\rightarrow \theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

Simplification: $\theta_0 = 0$ $n=2$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \theta_0 = 0$$

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

$\uparrow \quad \uparrow$
 $u^T v$



$$\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\| \leftarrow$$

$$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \leftarrow$$

Andrew Ng

SVM Decision Boundary

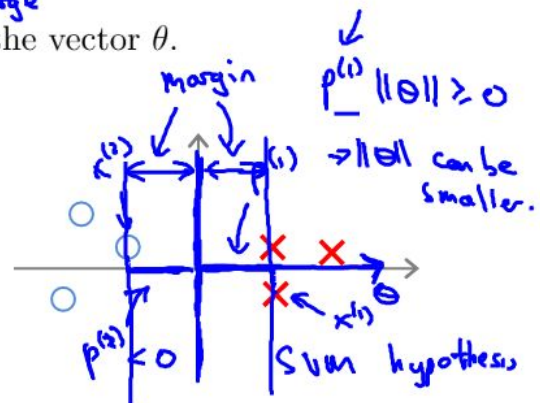
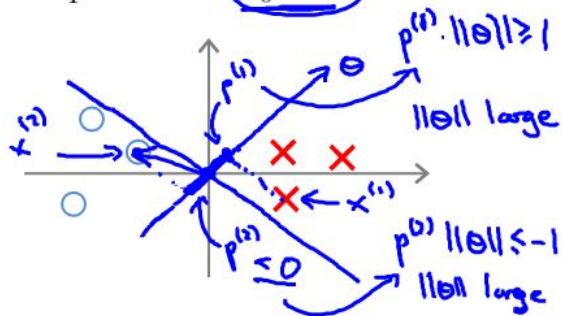
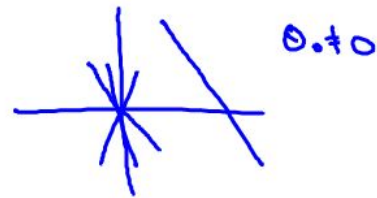
$$\Rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

s.t. $p^{(i)} \cdot \|\theta\| \geq 1$ if $y^{(i)} = 1$
 $p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)} = -1$

$\left. \begin{array}{l} \text{if } y^{(i)} = 1 \\ \text{if } y^{(i)} = -1 \end{array} \right\} C \text{ vary large}$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ .

Simplification: $\theta_0 = 0$



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.

The SVM optimization problem we used is:

$$\min_{\theta} \frac{1}{2} \sum_{i=1}^n \theta_i^2$$

s.t. $\|\theta\| \cdot p^{(i)} \geq 1$ if $y^{(i)} = 1$
 $\|\theta\| \cdot p^{(i)} \leq -1$ if $y^{(i)} = 0$

where $p^{(i)}$ is the (signed - positive or negative) projection of $x^{(i)}$ onto θ . Consider the training set above. At the optimal value of θ , what is $\|\theta\|$?

☐ 1/4
☒ 1/2
☐ 1
☐ 2

Skip Submit

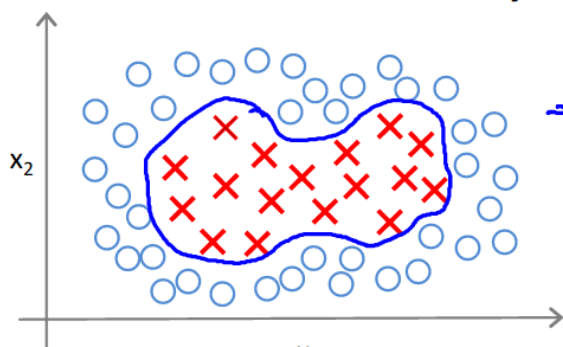
examples. So that

Andrew Ng

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

Kernels I :

Non-linear Decision Boundary



Predict $y = 1$ if

$$\rightarrow \theta_0 + \theta_1 \underline{x_1} + \theta_2 \underline{x_2} + \theta_3 \underline{x_1 x_2} + \theta_4 \underline{x_1^2} + \theta_5 \underline{x_2^2} + \dots \geq 0$$

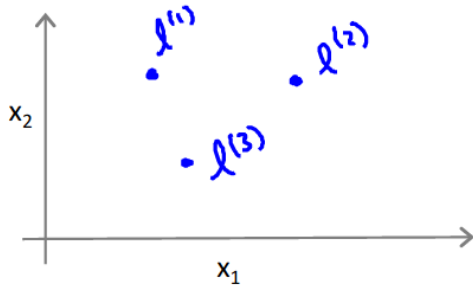
$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \dots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$$

$$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_4 = x_1^2, f_5 = x_2^2, \dots$$

Is there a different / better choice of the features f_1, f_2, f_3, \dots ?

Kernel



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

Given x :

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

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

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

\nwarrow kernel (Gaussian kernels) \nearrow $k(x, l^{(i)})$

Handwritten notes: $\|w\|$ points to $\|x - l^{(1)}\|^2$. $\|x - l^{(1)}\|^2$ is boxed in the original image.

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

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

$$\begin{array}{lcl} l^{(1)} & \rightarrow & f_1 \\ l^{(2)} & \rightarrow & f_2 \\ l^{(3)} & \rightarrow & f_3 \end{array}$$

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

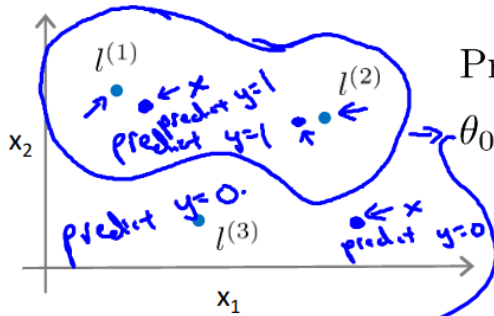
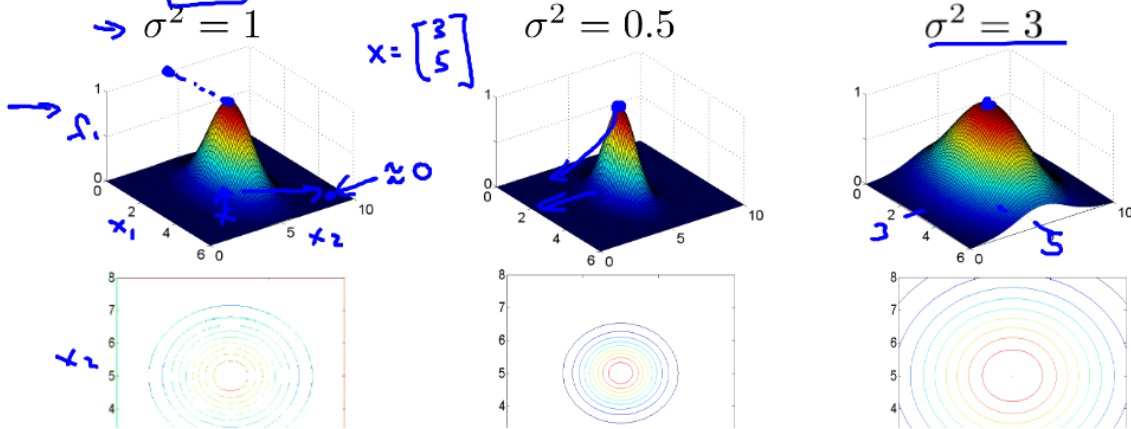
$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$$

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:

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



Predict "1" when

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

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

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

$\rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3$

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

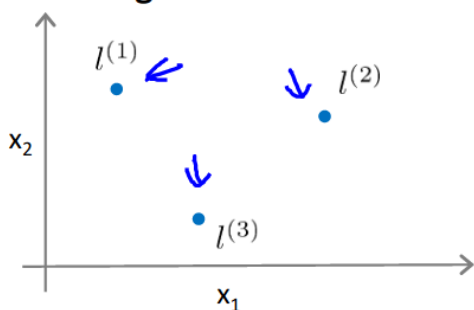
$f_1, f_2, f_3 \approx 0$

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

Kernel II :

We will choose landmarks as much as our training sets.

Choosing the landmarks

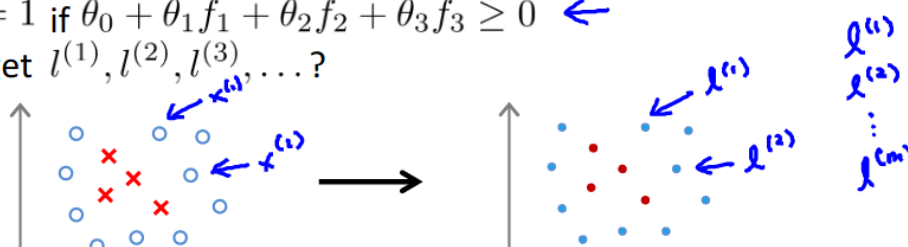


Given x :

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

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

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



SVM with Kernels

- \rightarrow Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$,
- \rightarrow choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$.

Given example x :

$$\begin{aligned} \rightarrow f_1 &= \text{similarity}(x, l^{(1)}) \\ \rightarrow f_2 &= \text{similarity}(x, l^{(2)}) \\ &\vdots \end{aligned}$$

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

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

$$\begin{aligned} x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} &= \begin{bmatrix} \text{sim}(x^{(i)}, l^{(1)}) \\ \text{sim}(x^{(i)}, l^{(2)}) \\ \vdots \\ \text{sim}(x^{(i)}, l^{(m)}) \end{bmatrix} \\ f_i^{(i)} &= \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = 1 \end{aligned}$$

$$\begin{aligned} x^{(i)} \in \mathbb{R}^{n+1} \quad (\text{or } \mathbb{R}^n) \\ \rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \\ f_0^{(i)} = 1 \end{aligned}$$

SVM with Kernels

Hypothesis: Given x , compute features $f \in \mathbb{R}^{m+1}$ $\theta \in \mathbb{R}^{n+1}$
 \rightarrow Predict "y=1" if $\theta^T f \geq 0$ $\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$

Training:

$$\rightarrow \min_{\theta} C \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^m \theta_j^2$$

$n=m$
 $\rightarrow \theta_0$

$$\rightarrow \begin{bmatrix} - \sum_j \theta_j^2 \\ - \end{bmatrix} = \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_m \end{bmatrix} \quad (\text{ignore } \theta_0)$$

$M = 10,000$

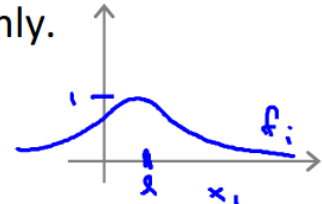
$\rightarrow \theta^T M \theta \leftarrow \|\theta\|^2$

SVM parameters:

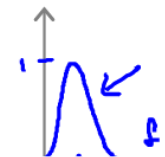
$C (= \frac{1}{\lambda})$. \rightarrow Large C: Lower bias, high variance. (small λ)
 \rightarrow Small C: Higher bias, low variance. (large λ)

σ^2 Large σ^2 : Features f_i vary more smoothly.
 \rightarrow Higher bias, lower variance.

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



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 $\theta^T x \geq 0$

$$\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0 \quad x \in \mathbb{R}^{n+1}$$

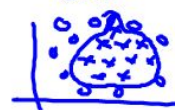
→ n large, n small

→ Gaussian kernel:

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

Need to choose $\frac{\sigma^2}{\uparrow}$

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



Andrew Ng

Kernel (similarity) functions:

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

Handwritten annotations: $x^{(i)}$ points to $x1$, $l^{(j)} = x^{(j)}$ points to $x2$. f_i points to the function name. $x \rightarrow \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{matrix}$ points to the output f .

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

→ $\|x - l\|^2$

$$v = x - l$$

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

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

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

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 $\text{similarity}(x, l)$ 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:

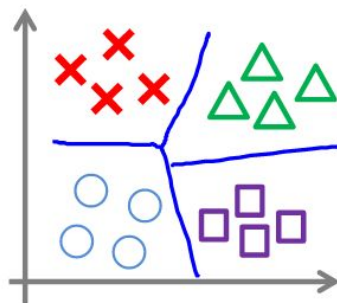
$$k(x, l) = (x^T l + \text{constant})^{\text{degree}}$$

Handwritten notes for polynomial kernel: $(x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$

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

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

Multi-class classification



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

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 $i = 1, 2, \dots, K$), get $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}$

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

$$\begin{matrix} \uparrow & \uparrow & \dots & \uparrow \\ y=1 & y=2 & & \theta=K \end{matrix}$$

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. $n \geq m$, $n = 10,000$, $m = 10 \dots 1000$)
- 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.

