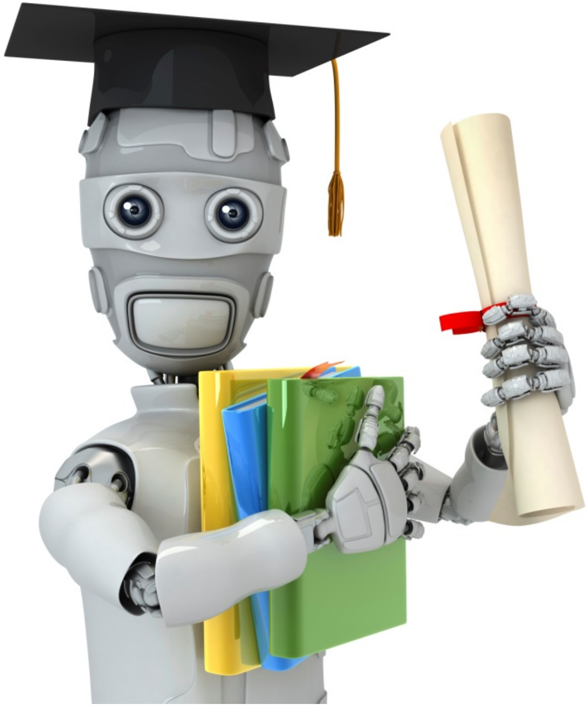


Machine Learning

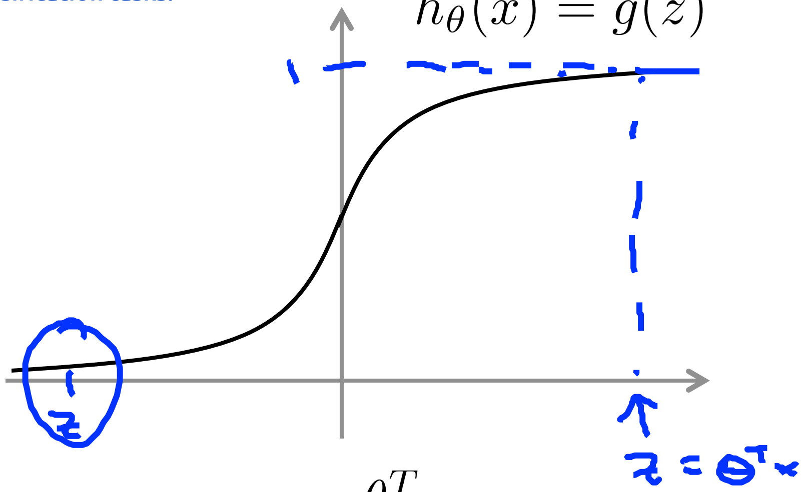
Support Vector Machines

Optimization objective

Alternative view of logistic regression

Support vector machine is highly preferred by many as it produces significant accuracy with less computation power.
Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks.
But, it is widely used in classification objectives.

$h_\theta(x) = g(z)$



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

$z = \theta^T x$

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

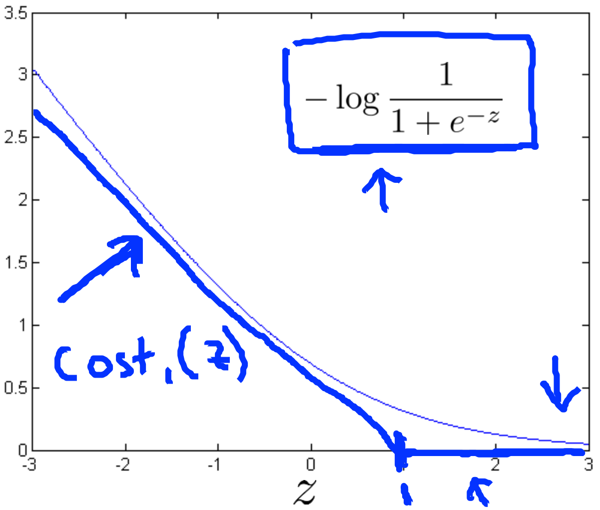
(x, y)

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

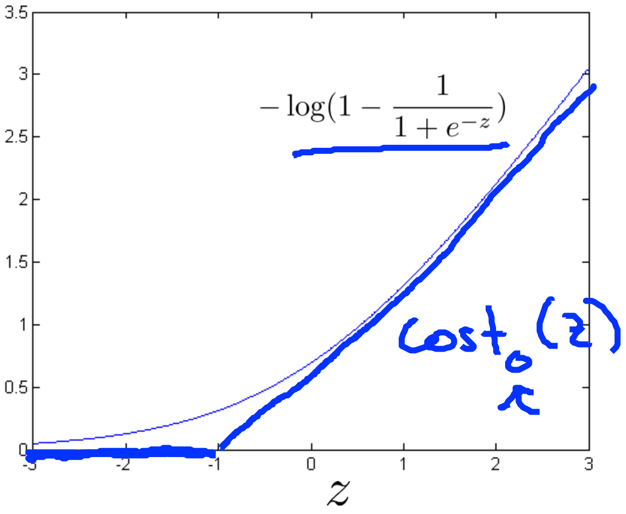
$$= \boxed{-y \log \frac{1}{1+e^{-\theta^T x}}} - \boxed{(1-y)\log\left(1-\frac{1}{1+e^{-\theta^T x}}\right)}$$

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

$z = \theta^T x$



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



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: C = 1/lambda

Removed 1/m factor as it'll not affect achieving the minimum **θ**
Using C instead of lambda is the same exact thing as if we minimize C ==> regularization term will be large and vice versa (A + lambda.B) ==> (C.A + B)

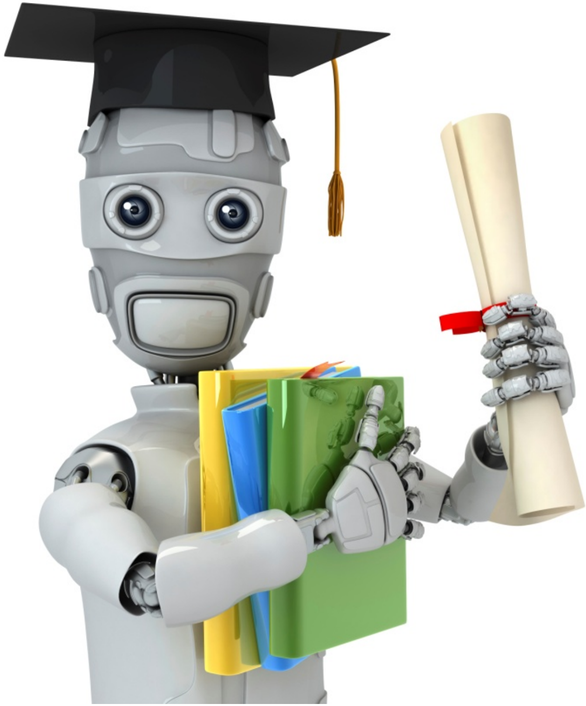
$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

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:

hθ (x) = { 1 if θT*X >= 0
0 otherwise }



Machine Learning

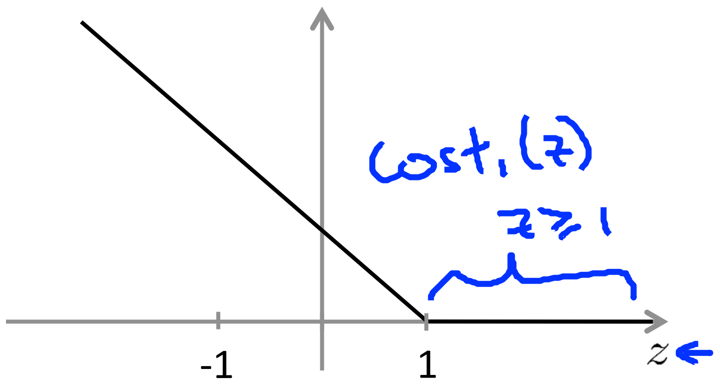
Support Vector Machines

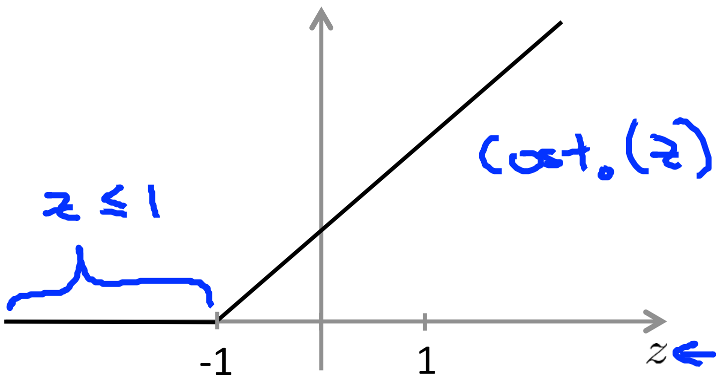
Large Margin Intuition

Support Vector Machine

In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

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





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

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

$\theta^T x \geq \cancel{0}\ 1$

$\theta^T x \leq \cancel{0}\ -1$

$C = 100{,}000$

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

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

$$\min_\theta \cancel{C * 0} + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

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

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

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

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

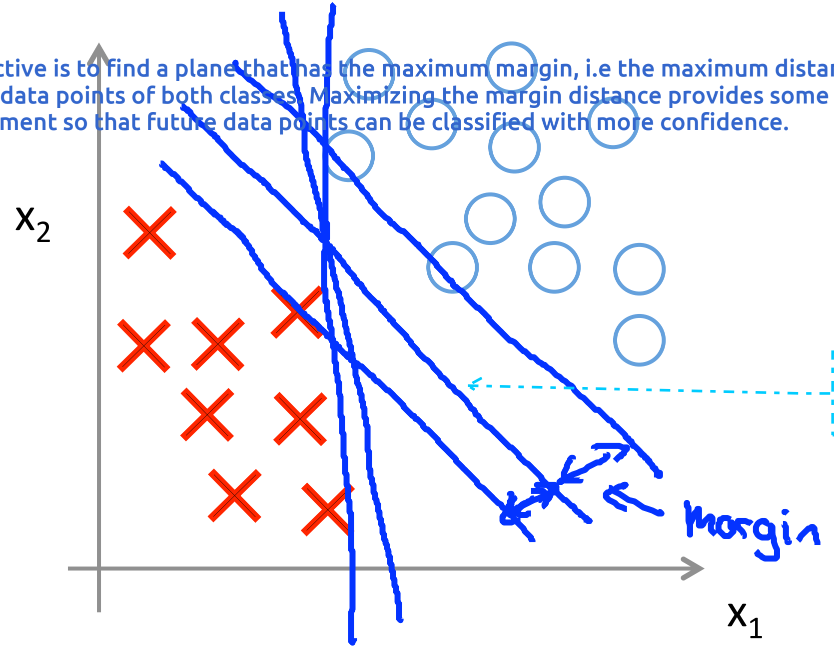
Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

SVM Decision Boundary: Linearly separable case

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.

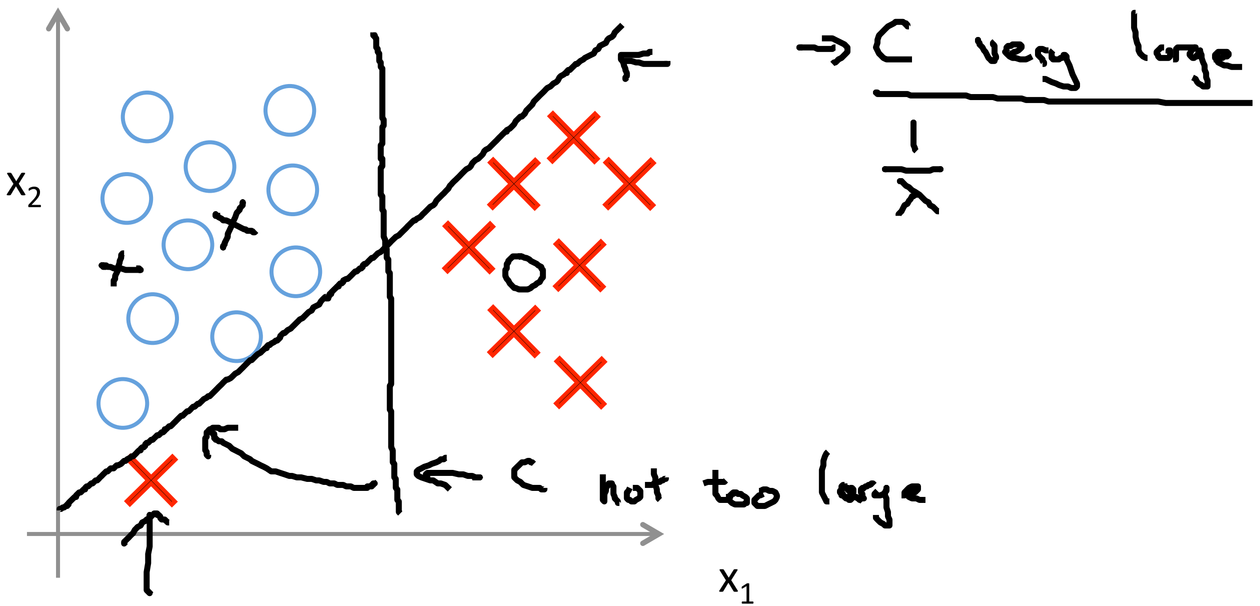
Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

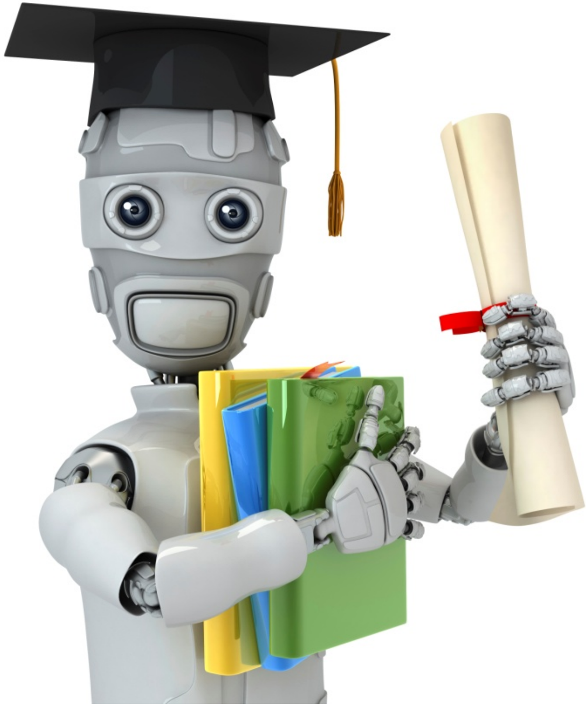


SVM will choose this line as a decision boundary

SVM sometimes is called **Large margin classifier**

Large margin classifier in presence of outliers



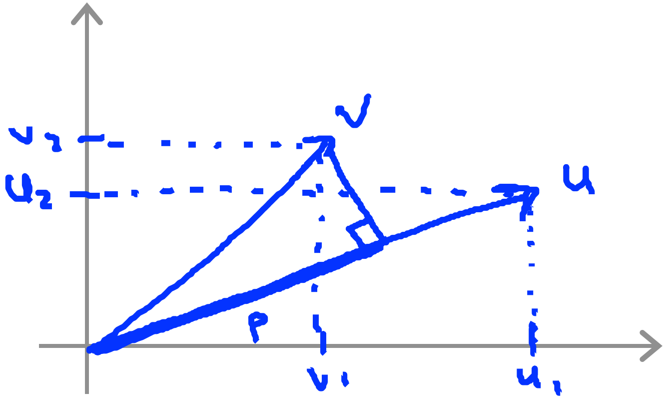


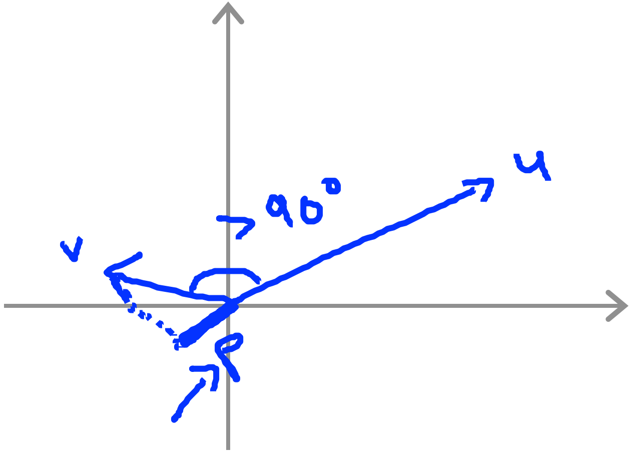
Machine Learning

Support Vector Machines

The mathematics behind large margin classification (optional)

Vector Inner Product





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

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

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

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

p = length of projection of v onto u.

Signed

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

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

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

$p < 0$

SVM Decision Boundary

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

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

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

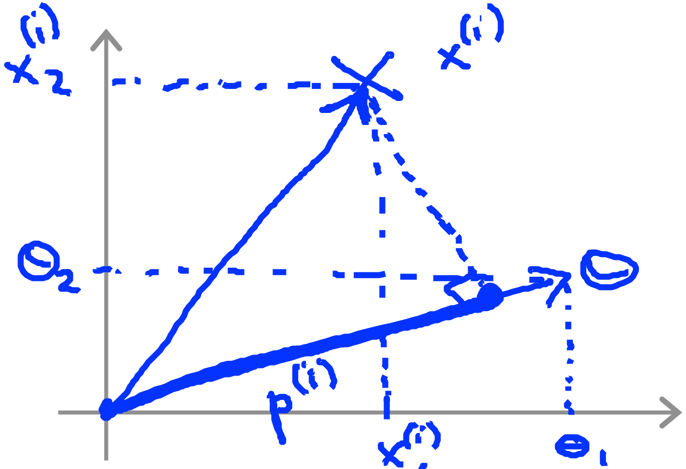
$$\theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0$$

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

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

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

$u^T v$



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

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

SVM Decision Boundary

Our target is minimizing norm of theta

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

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

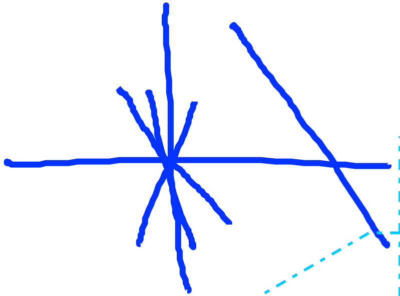
$p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)} = 1$

C very large

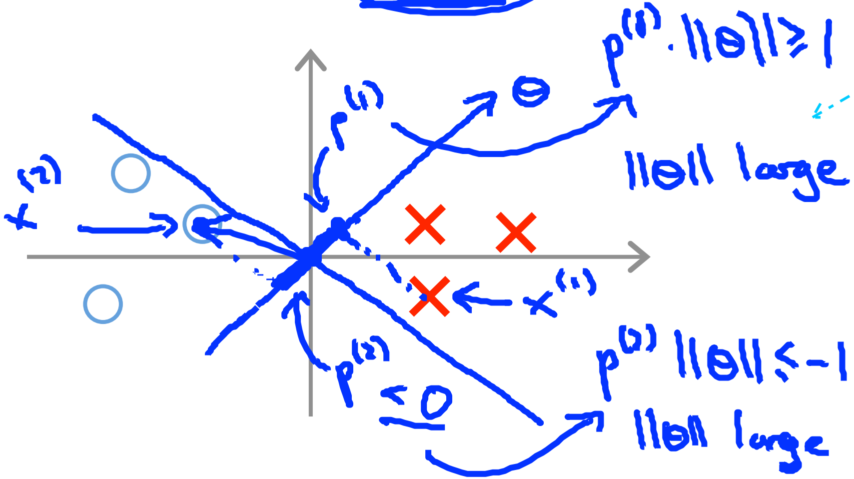
where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

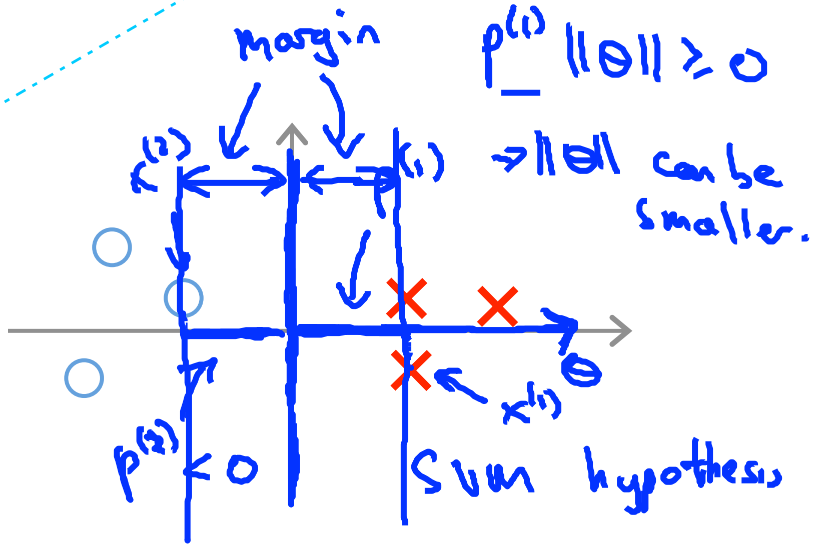
Simplification: $\theta_0 = 0$

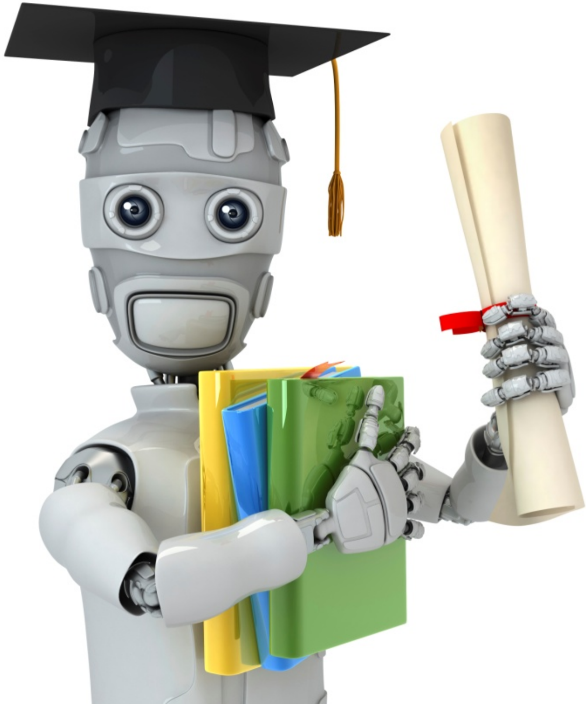
$\theta_0 \neq 0$



Only way to get this term >> 1 to have better margin is to get large norm theta and that is against our objective of minimizing norm of theta





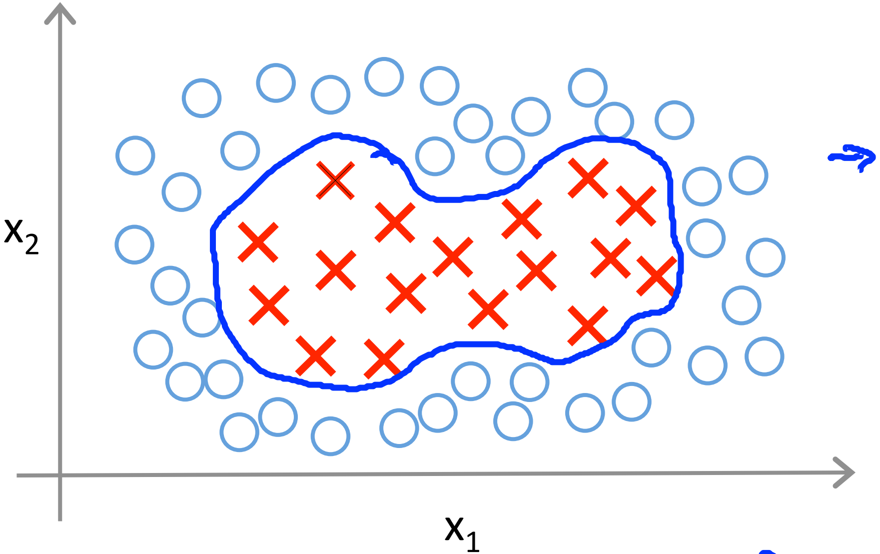


Machine Learning

Support Vector Machines

Kernels I

Non-linear Decision Boundary



Predict $y = 1$ if

$$\rightarrow \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$$

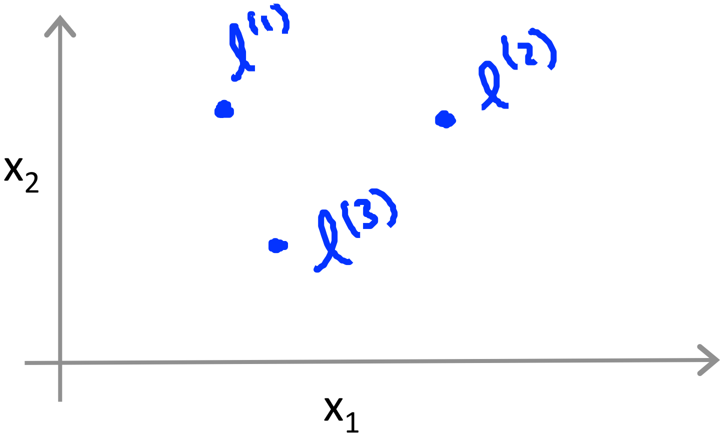
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

Kernel



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

$\|w\|$

$\|x - l^{(1)}\|^2$

Given x:

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

$$f_2 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$$

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

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

Kernels and Similarity

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

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

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

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

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

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

x

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

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

Example:

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

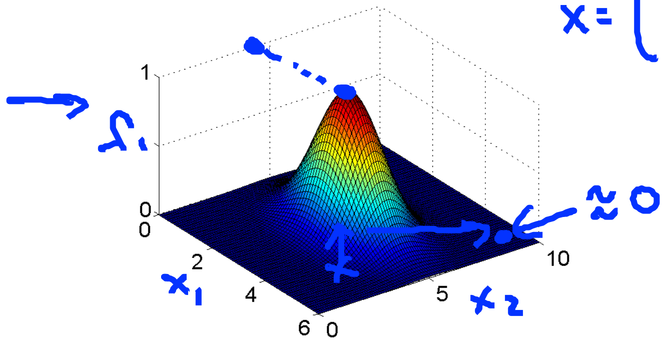
Gaussian Kernel

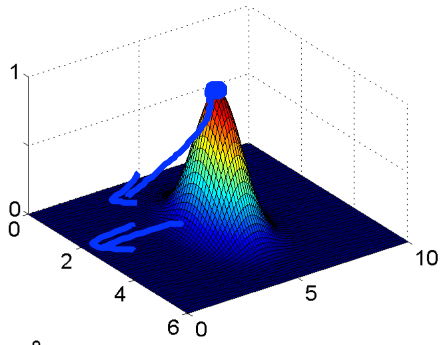
$\sigma^2 = 1$

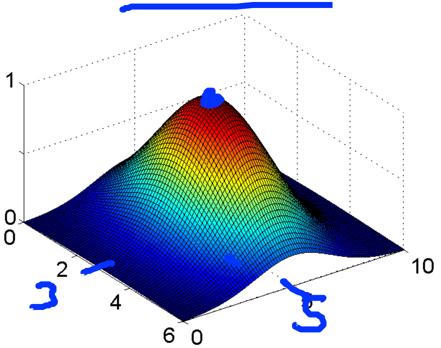
$x = \begin{bmatrix}3\\5\end{bmatrix}$

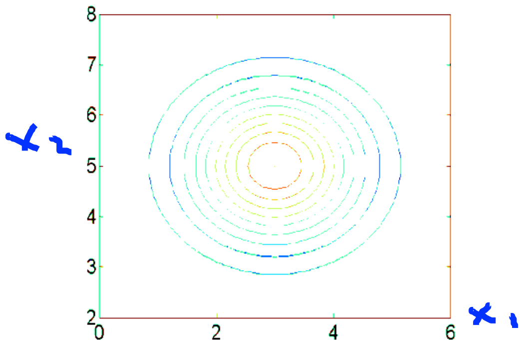
$\sigma^2 = 0.5$

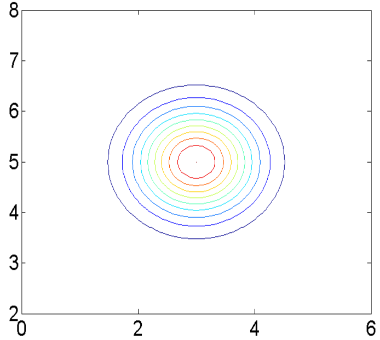
$\sigma^2 = 3$

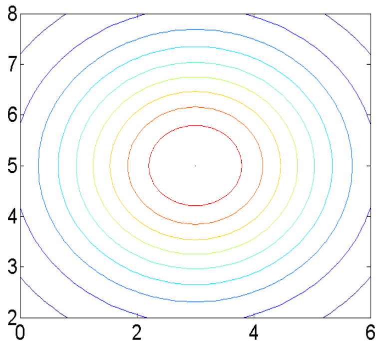




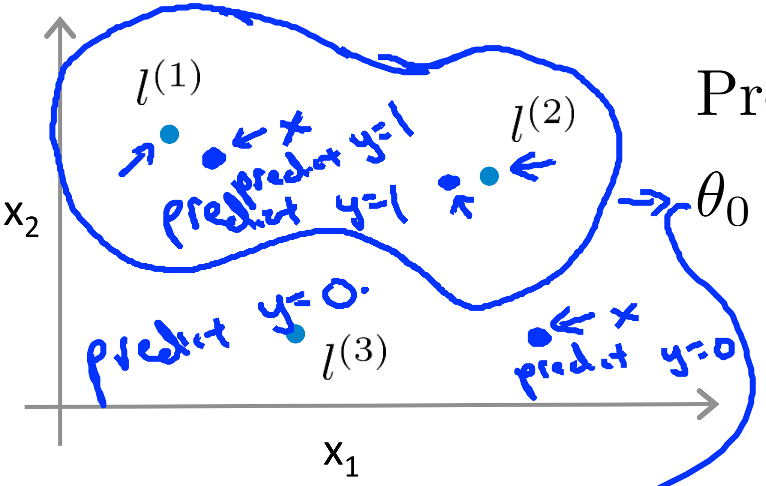








Andrew Ng



Predict "1" when

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

x

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

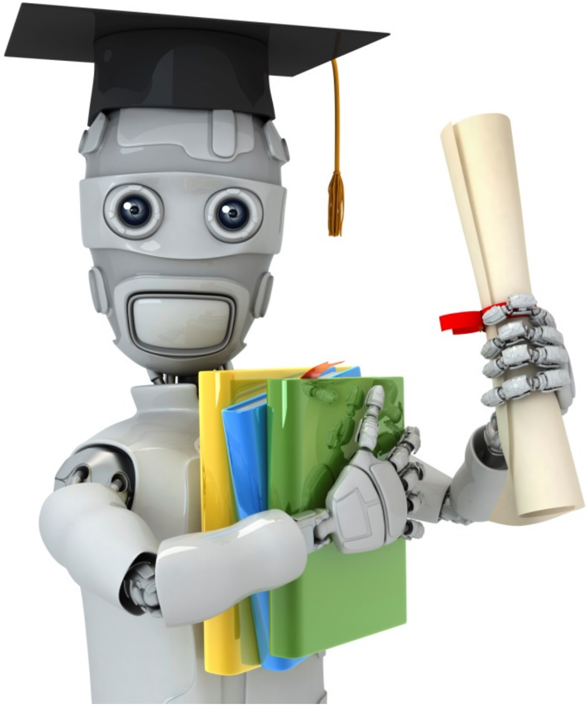
$f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0.$

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

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

$f_1, f_2, f_3 \approx 0$

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

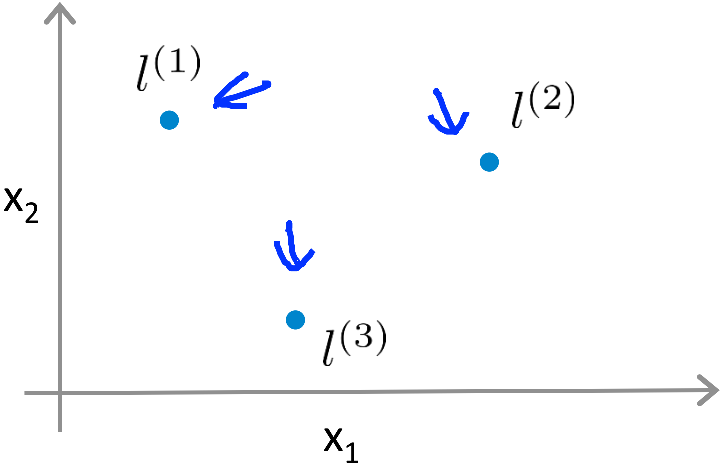


Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks

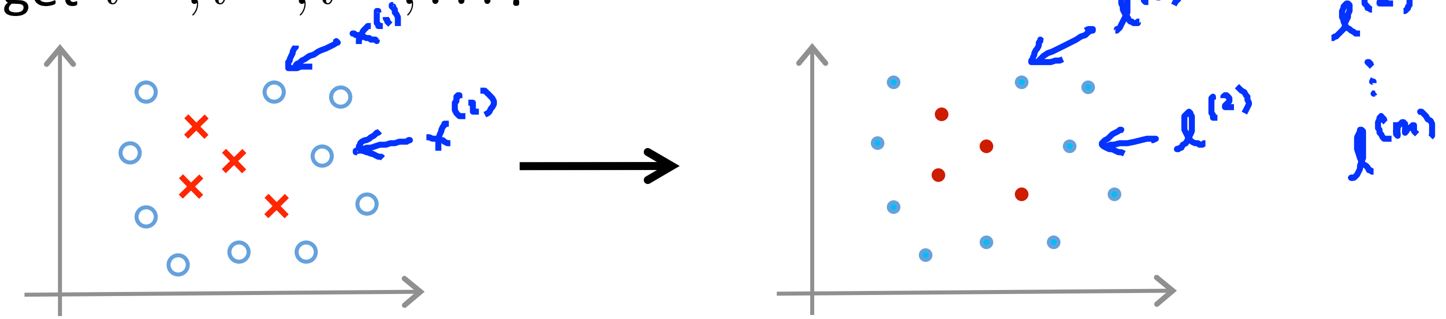


Given x:

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

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

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



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:

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$\ldots$$

$x^{(1)}$

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

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

$x^{(i)} \rightarrow$

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

$x^{(i)}$

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

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

$f_0^{(i)} = 1$

SVM with Kernels

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

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

Training: $n = m$

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

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

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

(ignore θ_0)

$\|\theta\|^2$

$m = 10{,}000$

$\theta^T M \theta$

SVM parameters:

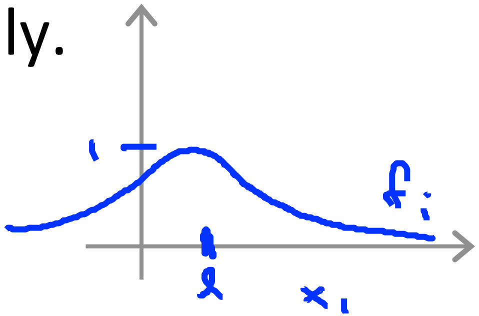
C ($= \frac{1}{\lambda}$). → Large C: Lower bias, high variance. (small λ)

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

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

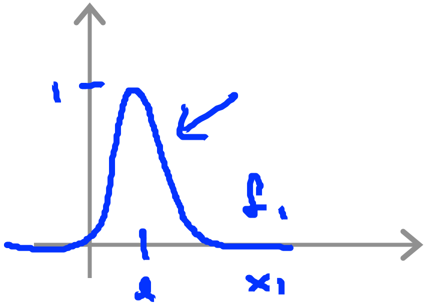
→ Higher bias, lower variance.

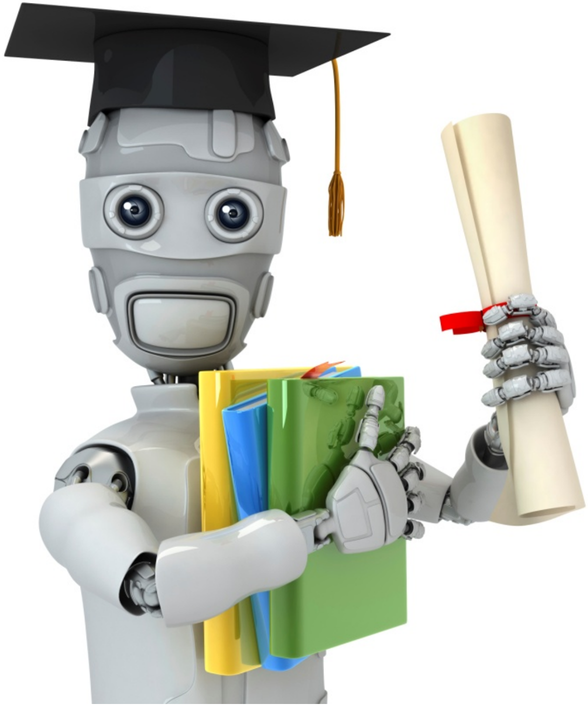
$\exp\left(-\frac{\|x-l^{(i)}\|^2}{2\sigma^2}\right)$



Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.





Machine Learning

Support Vector Machines

Using an SVM

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:

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

Need to choose σ^2.

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

and/or m large

Kernel (similarity) functions:

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

f_i

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

$$f = \exp\left(-\frac{\| \mathbf{x1} - \mathbf{x2} \|^2}{2\sigma^2}\right)$$

```
return
```

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

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

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

$\boxed{\|x - l\|^2}$

$v = x - l$

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

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

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:

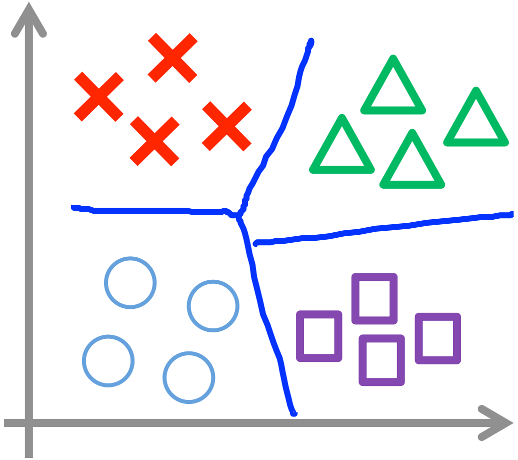
- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$

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

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

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

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

$y=1$ $y=2$ .... $\theta = K$

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

Logistic regression vs. SVMs

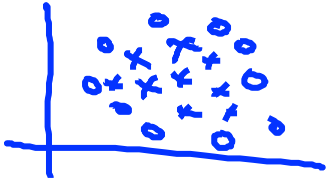
$n =$ number of features ($x \in \mathbb{R}^{n+1}$), $m =$ number of training examples

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