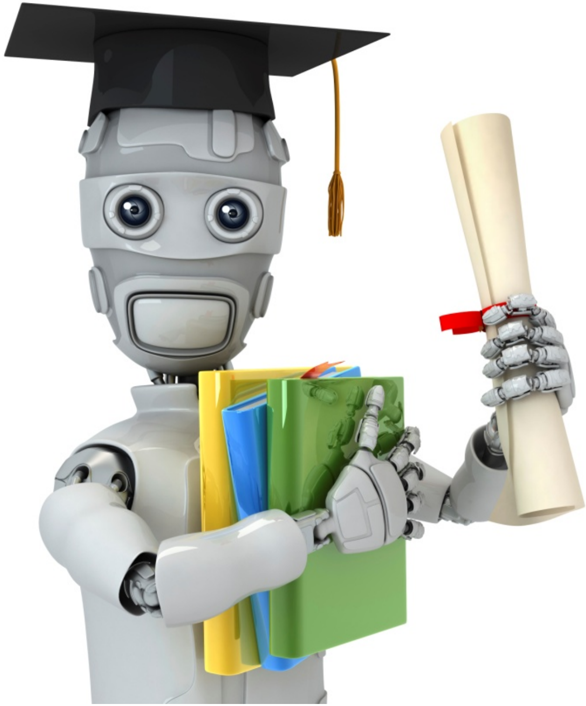


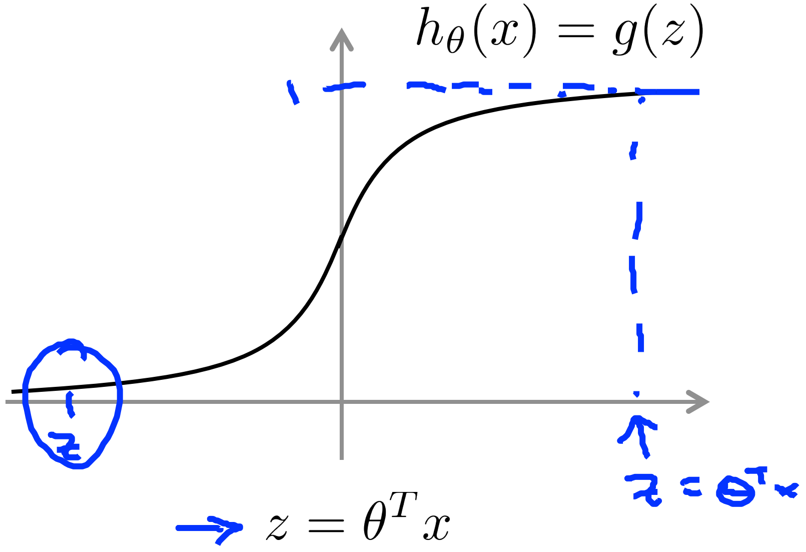
Machine Learning

Support Vector Machines

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 $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

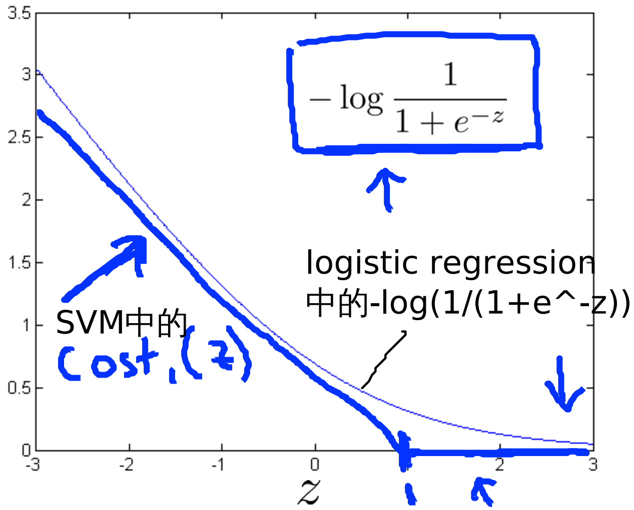
Alternative view of logistic regression

(x, y)

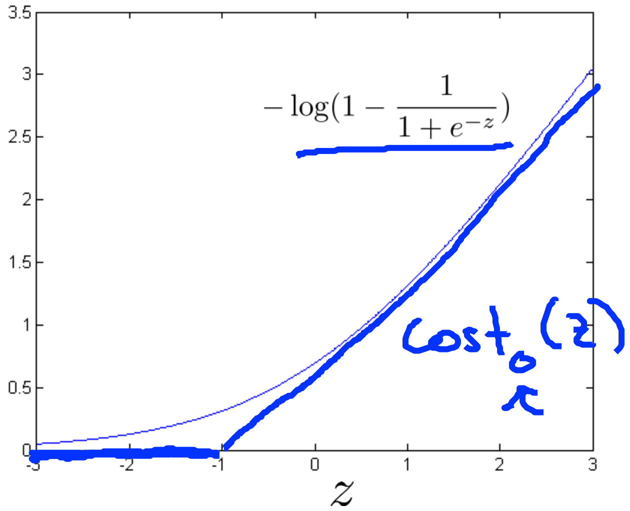
Cost of example: $-(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$

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

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



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



z為θ^T x

Andrew Ng

在之后的的优化问题中 这会变得更坚定 并且为支持向量机 带来计算上的优势 例如 更容易计算股票交易的问题 等等

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:

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

在这个最小化问题中 无论前面是否有 1/m 这一项 最终我所得到的 最优值θ都是一样的

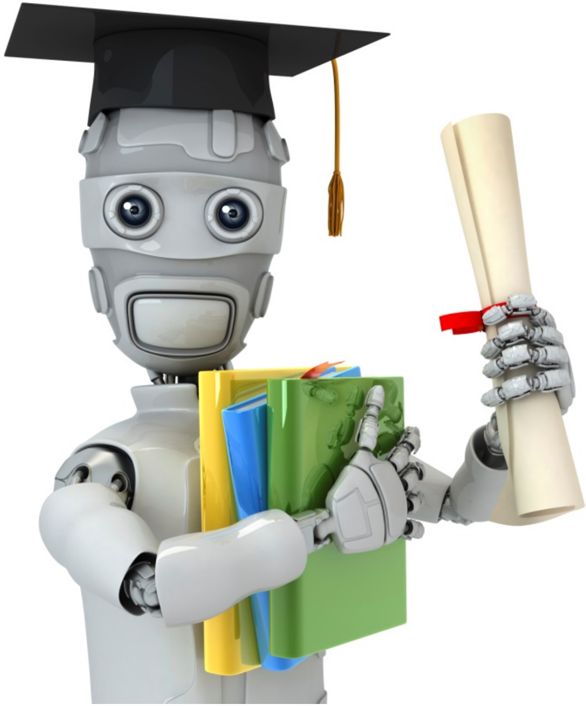
SVM hypothesis

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

你也可以 把这里的参数C 考虑成 1/λ(regularizaton中的) 同 1/λ 所扮演的 角色相同

Hypothesis:

有别于逻辑回归 输出的概率 在这里 我们的代价函数 当最小化代价函数 获得参数θ时 支持向量机所做的是
它来直接预测 y的值等于1 还是等于0



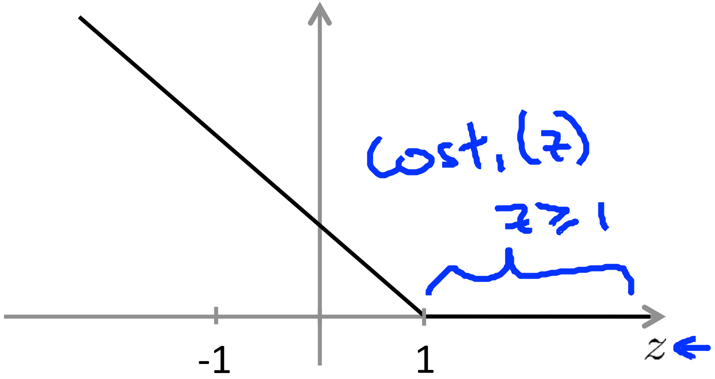
Support Vector Machines

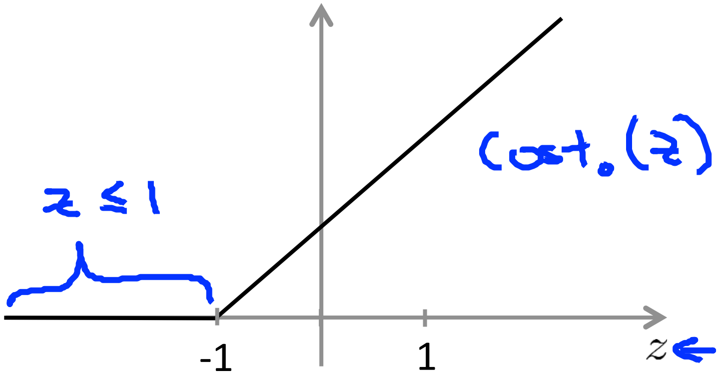
Large Margin Intuition

Machine Learning

Support Vector Machine

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





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

$C = 100{,}000$

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

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

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

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

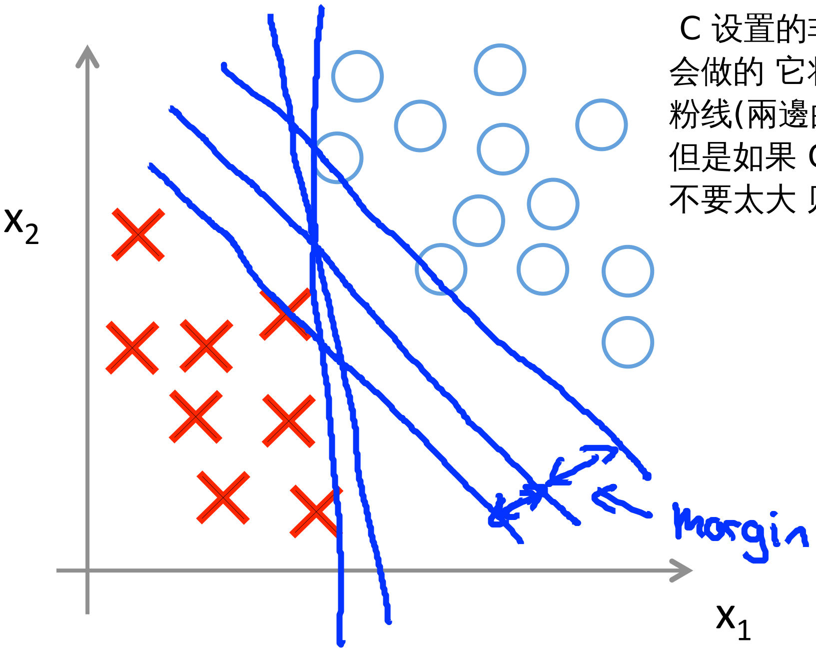
$$\min_\theta \cancel{C \times 0} + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

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

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

SVM Decision Boundary: Linearly separable case

我知道你也许 想知道 求解上一页
幻灯片中的优化问题 为什么会产生
这个结果 它是如何产生这个大间距
分类器的呢
我知道我还没有解释这一点
在下一节视频中 我将会从
直观上 略述

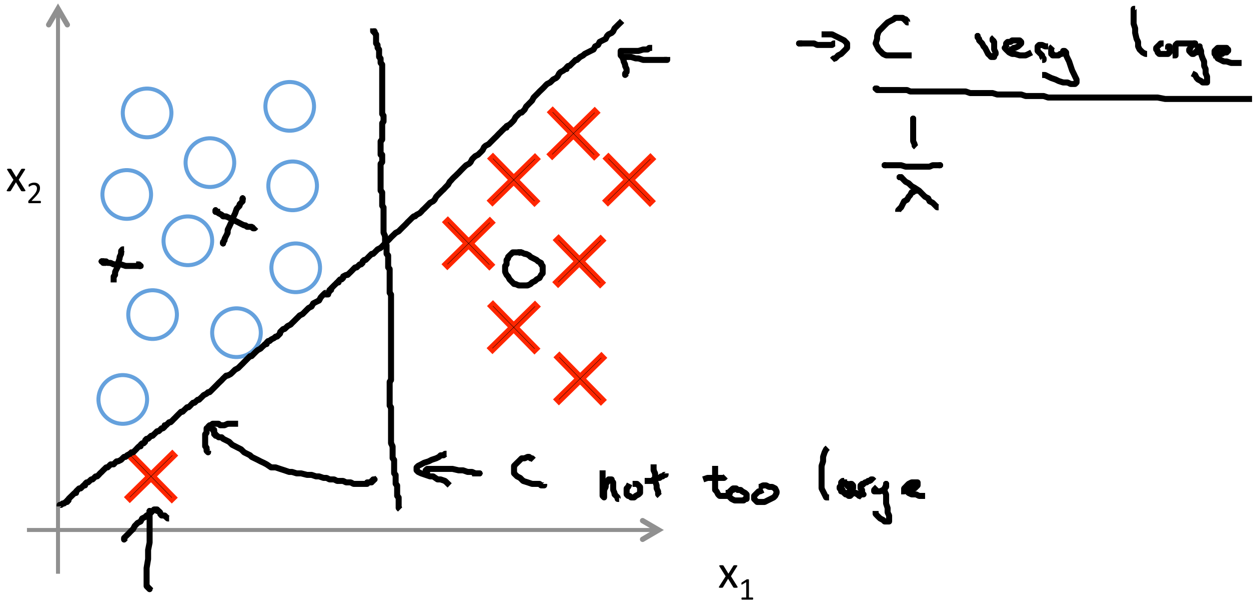


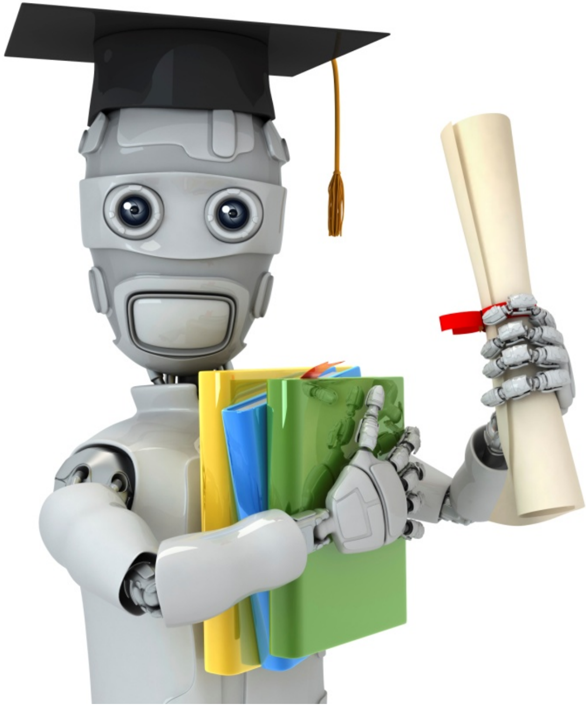
C 设置的非常大 这事实上正是 支持向量机将
会做的 它将决策界 从黑线(中間的) 变到了
粉线(兩邊的)
但是如果 C 设置的小一点 如果你将 C 设置的
不要太大 则你最终会得到 这条黑线

Large margin classifier

Large margin classifier in presence of outliers

当 C 不是 非常非常大的时候
它可以忽略掉一些异常点的影响
得到更好的决策界



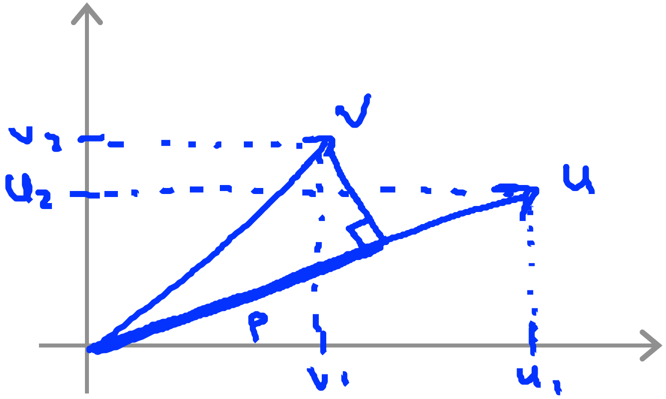


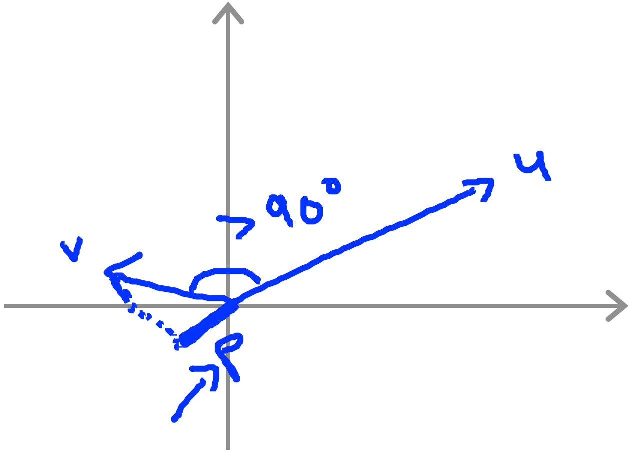
Machine Learning

Support Vector Machines

The mathematics behind large margin classification (optional)

Vector Inner Product





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

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

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

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

p = length of projection of v onto u.

signed

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

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

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

$p < 0$

SVM Decision Boundary

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

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

$$= \|\theta\|$$

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

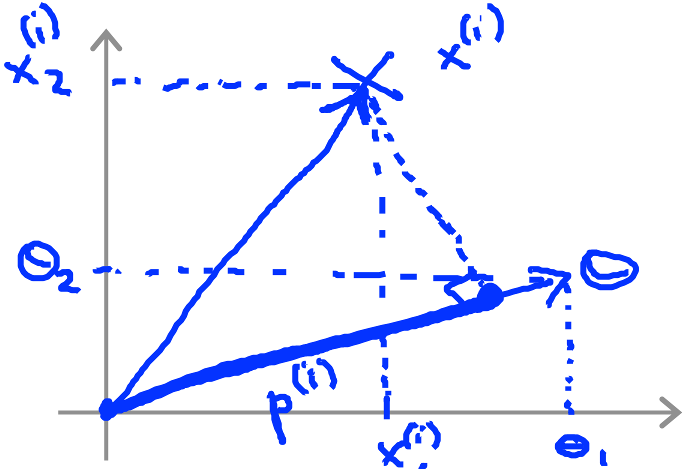
s.t. $\boxed{\theta^T x^{(i)} \geq 1}$ if $y^{(i)} = 1$

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

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

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

$$u^T v$$



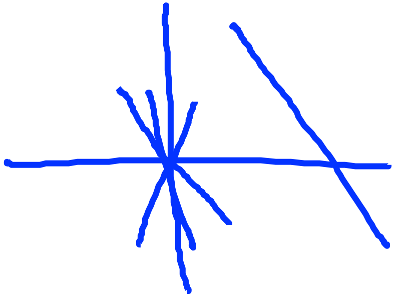
$$\theta^T x^{(i)} = \boxed{p^{(i)} \cdot \|\theta\|}$$

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

Andrew Ng

SVM Decision Boundary

θ₀ ≠ 0



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

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

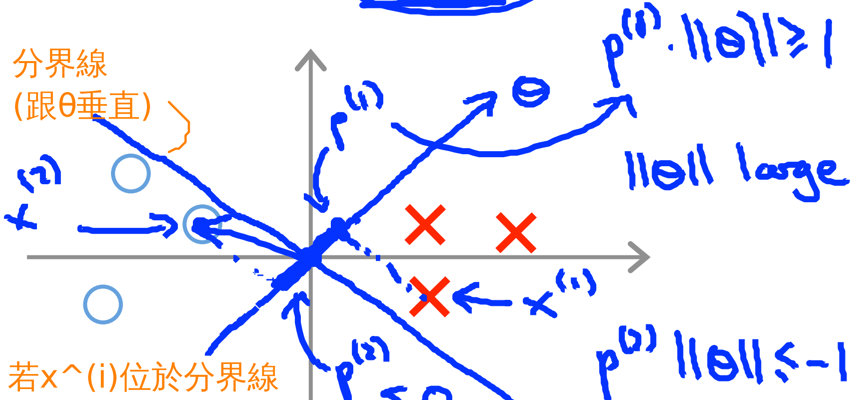
$p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)} = 1$

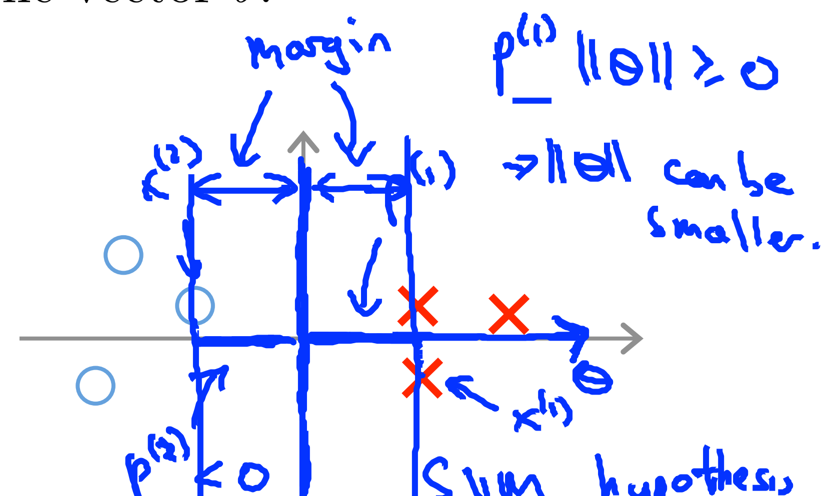
C very large

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

Simplification: $\theta_0 = 0$

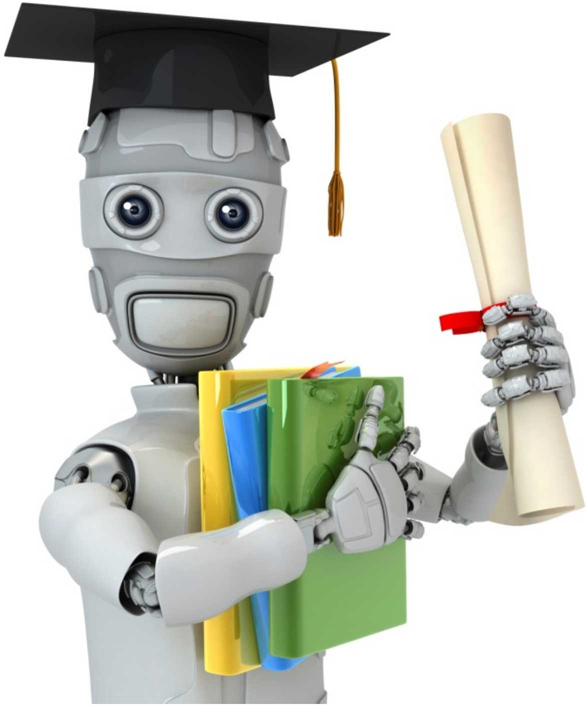
分界線
(跟θ垂直)





易知: 若x^(i)位於分界線右上邊, 則p^(i) || θ || > 0;
若在左下邊, 則p^(i) || θ || < 0.

即便 θ0 不等于0
产生大间距分类器的结论 会被证明同样成立

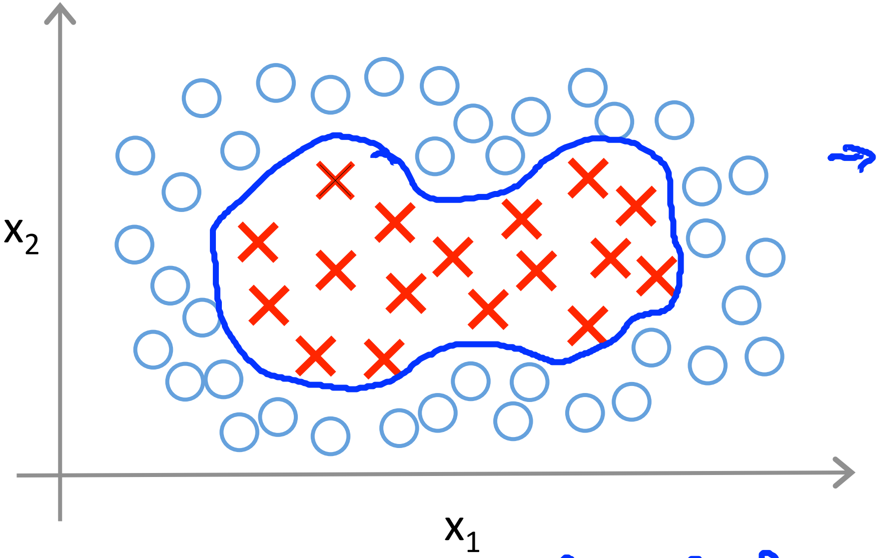


Support Vector Machines

Kernels I

Machine Learning

Non-linear Decision Boundary



Predict $y = 1$ if

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

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

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

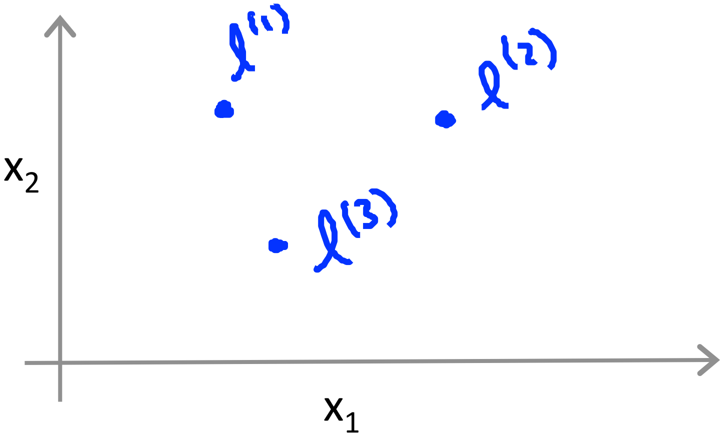
$$f_1 = x_1, \quad f_2 = x_2, \quad f_3 = x_1 x_2, \quad f_4 = x_1^2, \quad f_5 = x_2^2, \ldots$$

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

Andrew Ng

我打算手动选取一些点 然后将这些点定义为l(1) 再选一个 不同的点 把它定义为l(2) 再选第三个点 定为l(3)

Kernel



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

Given x:

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

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

$\|w\|$

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

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

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

Kernels and Similarity

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

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

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

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

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

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

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

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

x

Andrew Ng

Example:

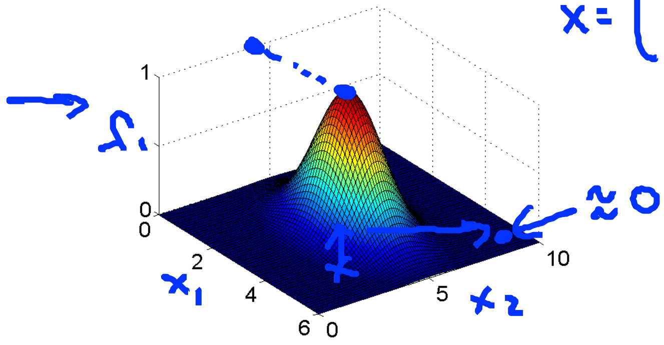
$$l^{(1)} = \boxed{\begin{bmatrix} 3 \\ 5 \end{bmatrix}}, \quad \underline{f_1 = \exp\left(- \frac{\| x - l^{(1)} \|^2}{2\boxed{\sigma^2}} \right)}$$

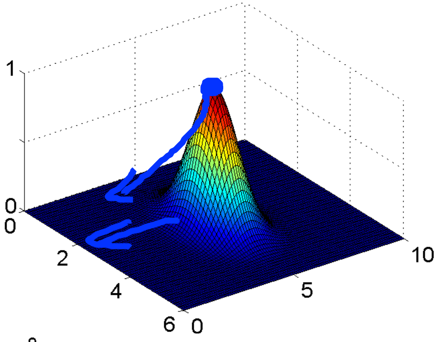
$\sigma^2 = 1$

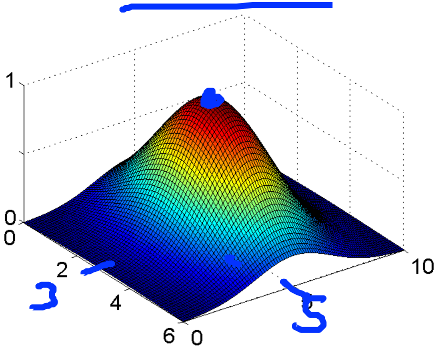
$x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

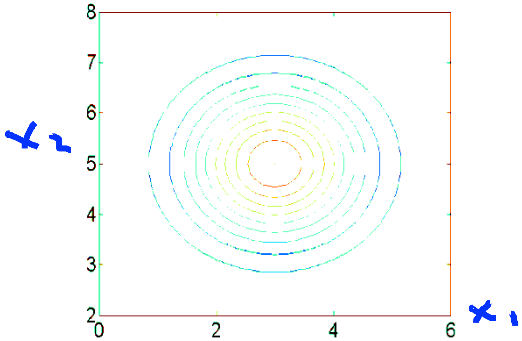
$\sigma^2 = 0.5$

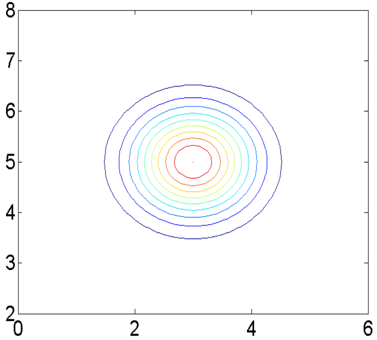
$\underline{\sigma^2 = 3}$

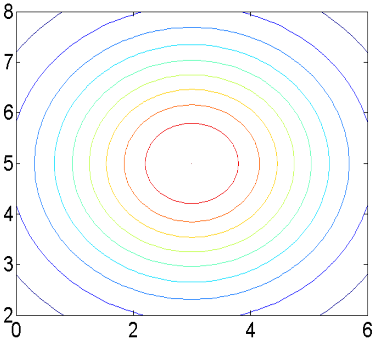






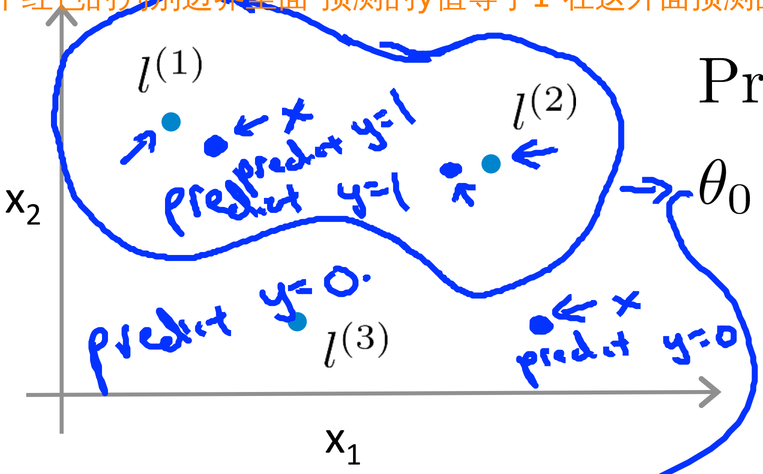






Andrew Ng

对于接近l(1)和l(2)的点 我们的预测值是1. 对于远离 l(1)和l(2)的点 我们最后预测的结果 是等于0的
我们最后会得到 这个预测函数的 判别边界 会像这样
在这个红色的判别边界里面 预测的y值等于1 在这外面预测的y值 等于0



Predict "1" when

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

x

假设 我已经得到了 这些参数的值

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

$f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0.$ 因为x 接近于l(1) 那么f1 就接近于1

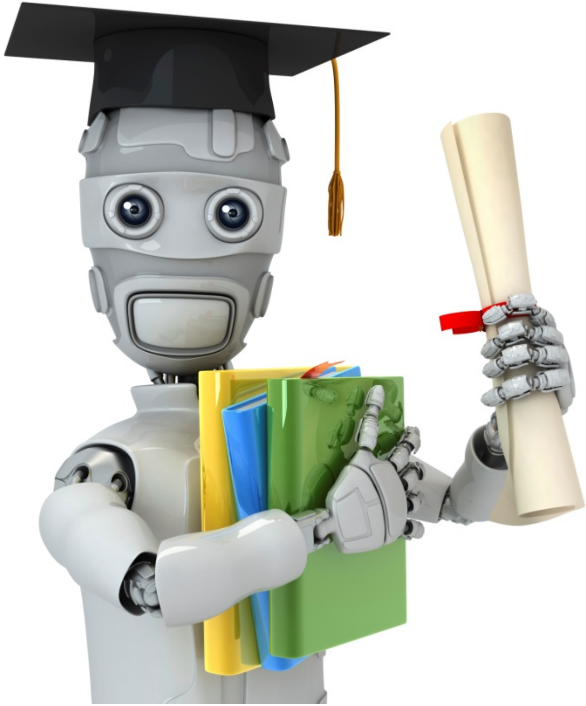
$\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$

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

$f_1, f_2, f_3 \approx 0$

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

Andrew Ng



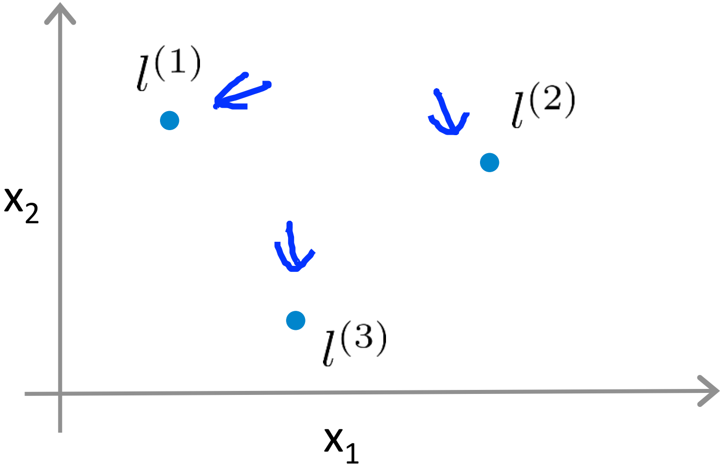
Machine Learning

Support Vector Machines

Kernels II

我们从哪里得到l(1) l(2) l(3) ？我们直接 将训练样本 作为标记点. 这说明特征函数基本上 是在描述 每一个样本距离 样本集中其他样本的距离

Choosing the landmarks



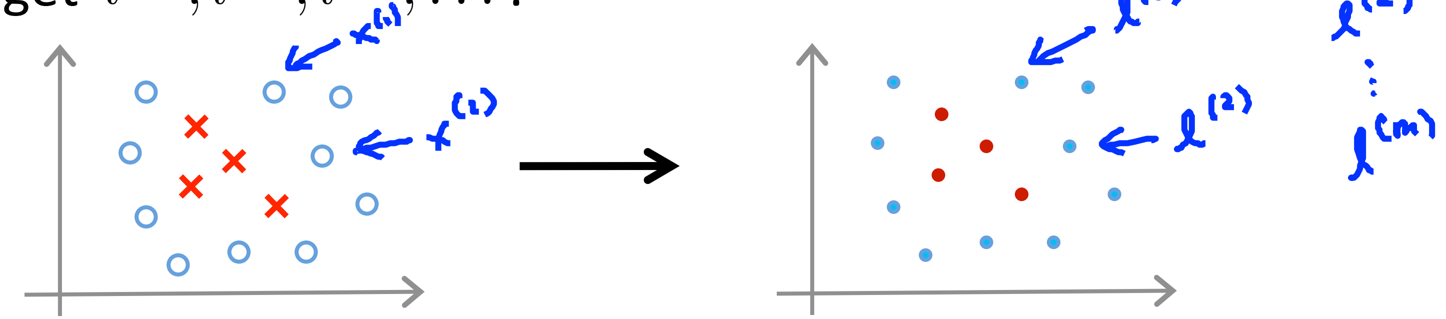
Given x:

$$f_i = \text{similarity}(x, l^{(i)})$$

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

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

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



我们具体的列出 这个过程的大纲

SVM with Kernels

给定m个训练样本

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

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

如果我们需要的话 可以添加额外的特征f0 f0的值始终为1

Given example x: 即x^(i)

$x^{(i)}$

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

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

$\ldots$

由後面知，一個f_i就是一個feature.

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

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

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

$x^{(i)}$

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

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

Andrew Ng

SVM with Kernels

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

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

Training:

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

$n = m$

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

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

$\|\theta\|^2$ $\quad m = 10{,}000$

$\theta^T M \theta$

不直接用 θ 的模的平方进行最小化 而是最小化了另一种类似的度量(它主要是为了计算效率, 沒細講)

SVM parameters:

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

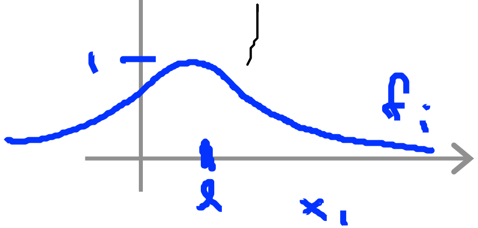
→ Small C: Higher bias, low variance. (large λ)

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

→ Higher bias, lower variance.

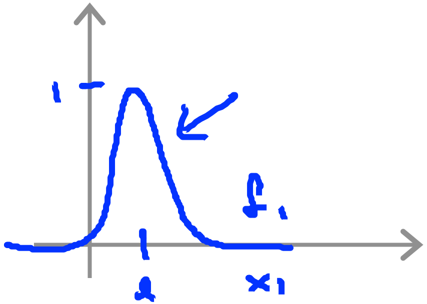
對不同的x_i值, 對應的f_i值
差别不大, 所以higher bias $\exp\left(-\frac{\|x-l^{(i)}\|^2}{2\sigma^2}\right)$

這是{f_1, f_2, ...}這一些
數的分佈. f_1是一個數.

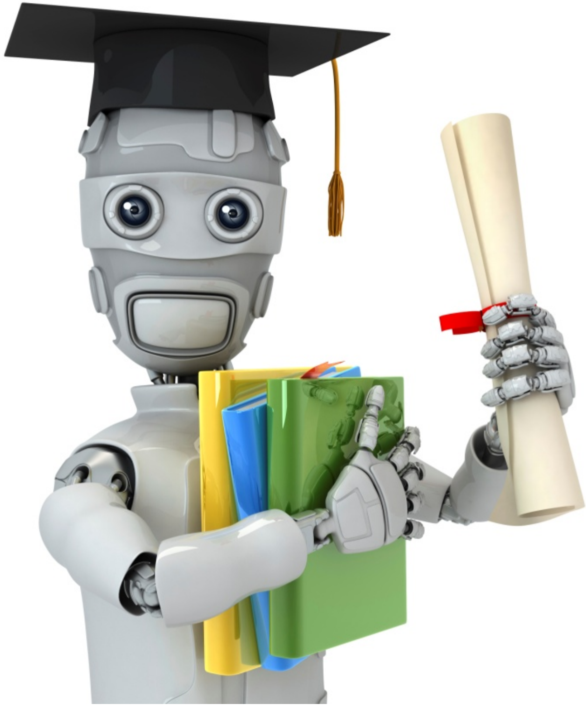


Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.



将核函数用于 逻辑回归时 会变得非常的慢



Support Vector Machines

Using an SVM

Machine Learning

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):

如果你有大量的特征变量 如果 n 很大 而训练集的样本数 m 很小，也许你应该拟合 一个线性的判定边界 不要拟合非常复杂的非线性函数 因为没有足够的数据

不用核函数这个作法 也叫线性核函数

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

什么时候选择高斯核函数呢？

目前看来 高斯核函数和线性核函数确实是最普遍的核函数

Kernel (similarity) functions:

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

f_i

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

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

```
return
```

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

如果你原来的特征变量 x 是 n 维的 如果 n 很小 并且 理想情况下 如果 m 很大
那么可能你需要用 一个核函数去拟合一个 更复杂的非线性判定边界
那么高斯核函数会是不错的选择

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

归一化

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

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

$v = x - l$

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

$= (x_1 - l_1)^2 + (x_2 - l_2)^2 + \cdots + (x_n - l_n)^2$

1000 feet^2

1-5 bedrooms

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:

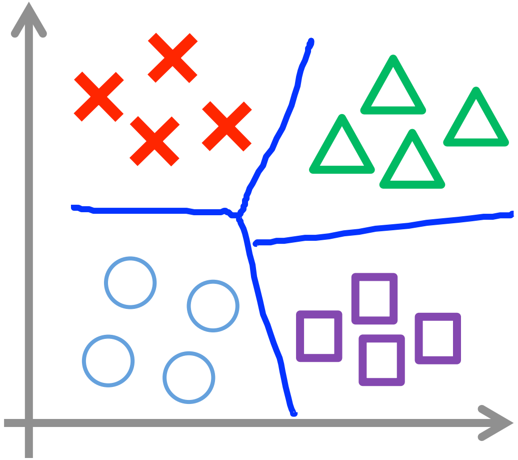
- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$

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

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

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

Multi-class classification



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

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

$y=1$ $y=2$ .... θ K

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

(good!) 对这两个算法 你什么时候应该用哪个呢？

Logistic regression vs. SVMs

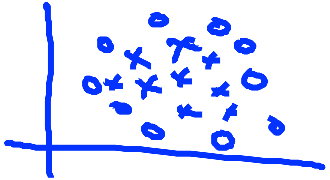
n = number of features ($x \in \mathbb{R}^{n+1}$), m = number of training examples

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

此時高斯核函数 运行起来就会很慢

Neural network likely to work well for most of these settings, but may be slower to train.

实际上 SVM 的优化问题 是一种凸优化问题
因此好的 SVM 优化软件包 总是会找到 全局最小值 或者接近它的值
对于SVM 你不需要担心局部最优
在实际应用中 局部最优 对神经网络来说不是非常大 但是也不小