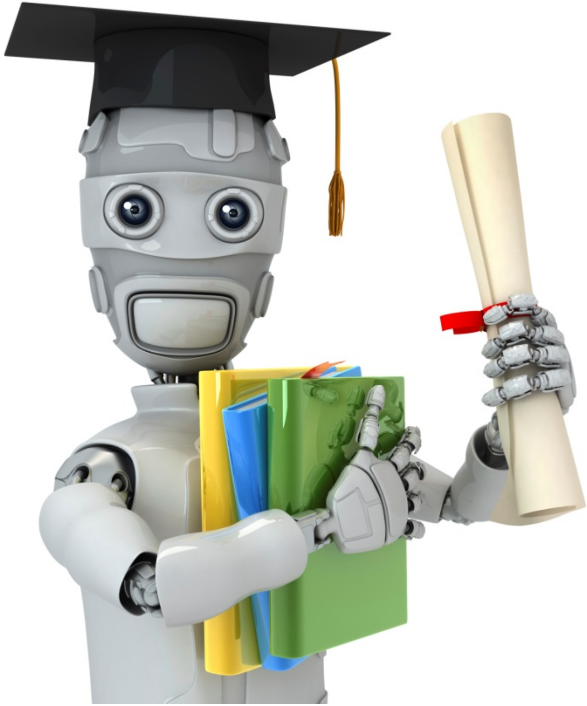


Machine Learning

Support Vector Machines

sometimes gives a cleaner and more powerful way of learning complex non-linear functions.

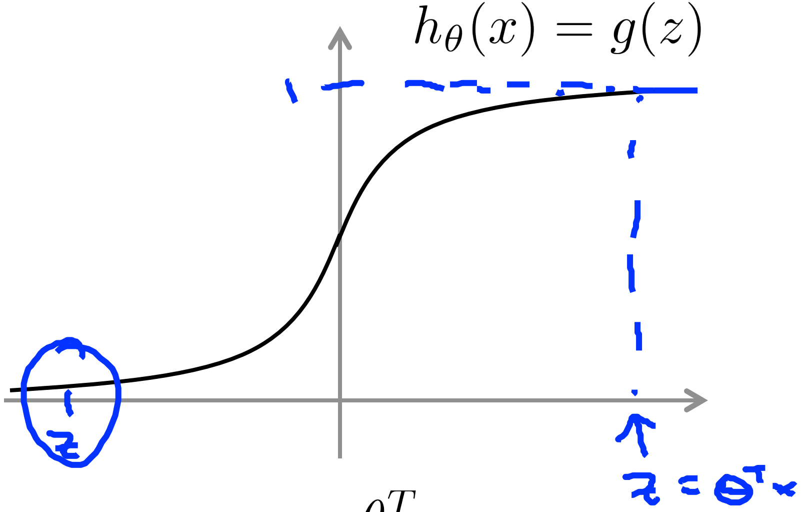
Optimization objective

Alternative view of logistic regression

sigmoid activation function

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

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



$z = \theta^T x$

much much greater than

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

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

Alternative view of logistic regression

(x, y)

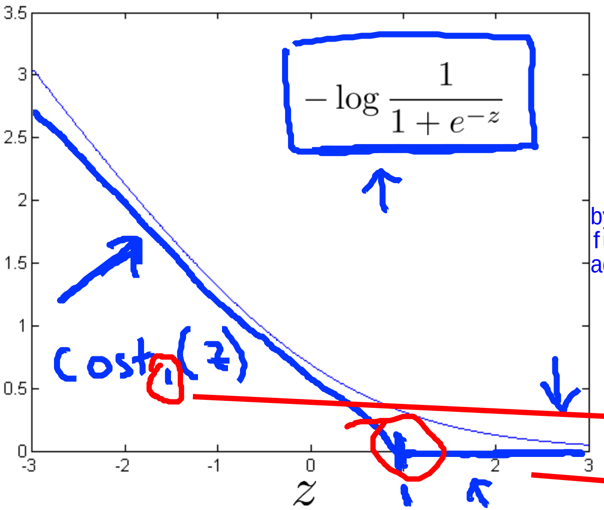
Cost of example: $-(y \log \underline{h_\theta(x)} + (1-y)\log(1-\underline{h_\theta(x)}))$ ←

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

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

$z = \theta^T x$

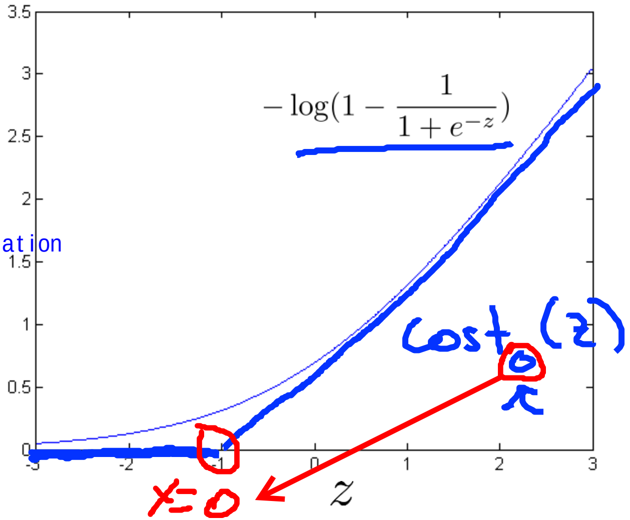
If $y = 0$ (want $\theta^T x \ll 0$):



by defining this new cost function, it gives us computation advantage

$x=1$

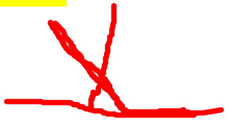
new cost function

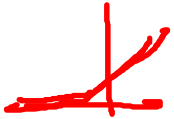


Andrew Ng

Support vector machine

Logistic regression:





$$\min_\theta \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \underbrace{\left(-\log h_\theta(x^{(i)})\right)}_{\text{replace with } Cost_1(\theta^T x^{(i)})} + (1-y^{(i)}) \underbrace{\left((-\log(1-h_\theta(x^{(i)}))\right)}_{Cost_0(\theta^T x^{(i)})}\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Support vector machine:

C by convention, just a diff way of controlling the tradeoff in the regularization term!

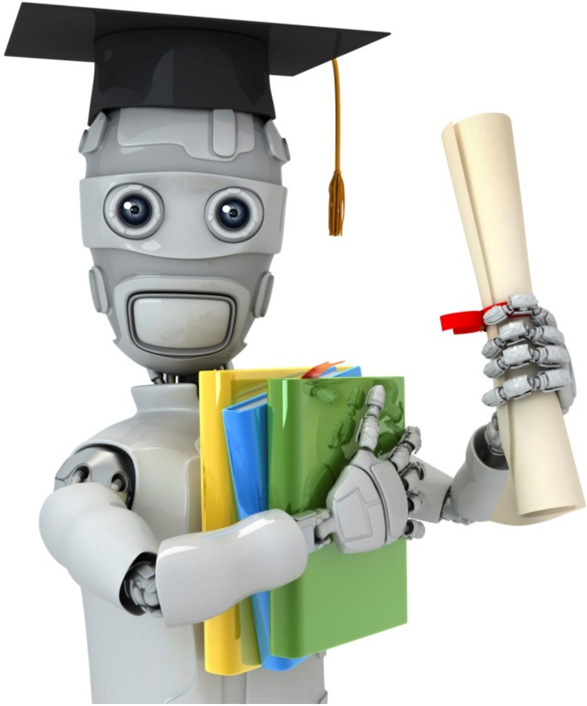
$$\boxed{\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2}$$

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:

$$h_\theta(x) \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{o.w} \end{cases}$$



Machine Learning

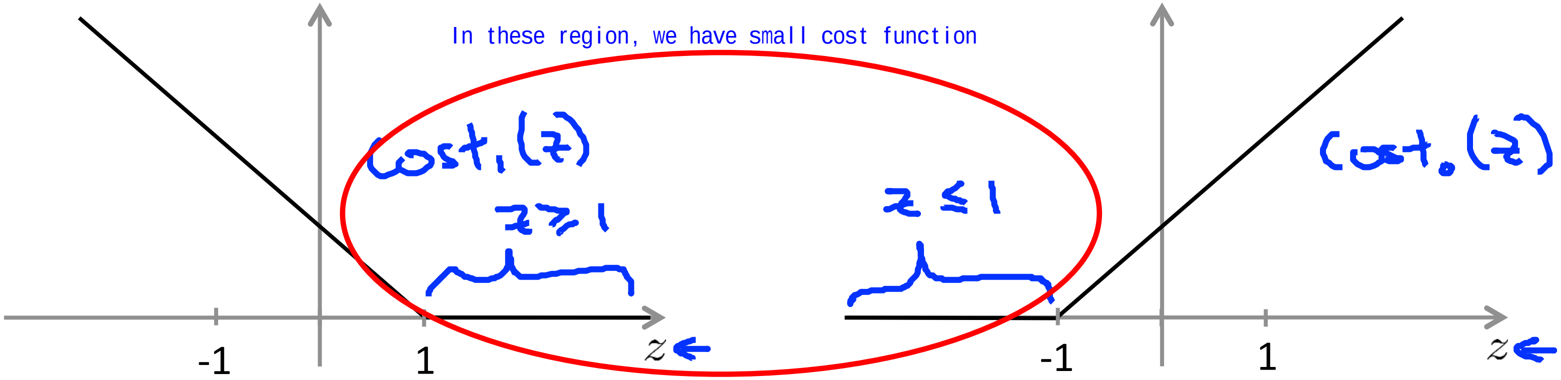
= large margin classifier

Support Vector Machines

Large Margin Intuition

Support Vector Machine

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



In these region, we have small cost function

so that cost function small!

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

$C = 100{,}000$

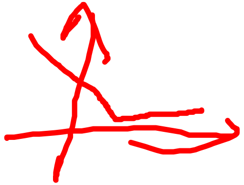
Andrew Ng

SVM Decision Boundary

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

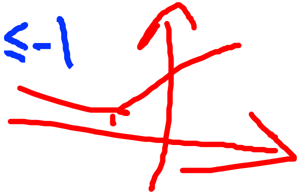
Whenever $y^{(i)} = 1$:

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



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

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



New optimization problem: much efficient!

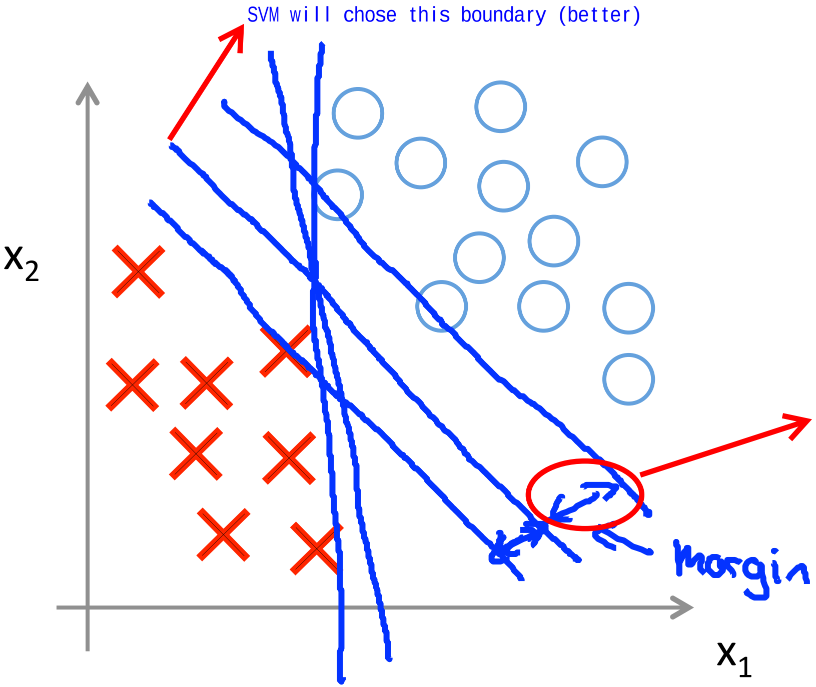
$$\min_{\theta} \cancel{C \times 0} + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

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

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

subject to these constraints

SVM Decision Boundary: Linearly separable case



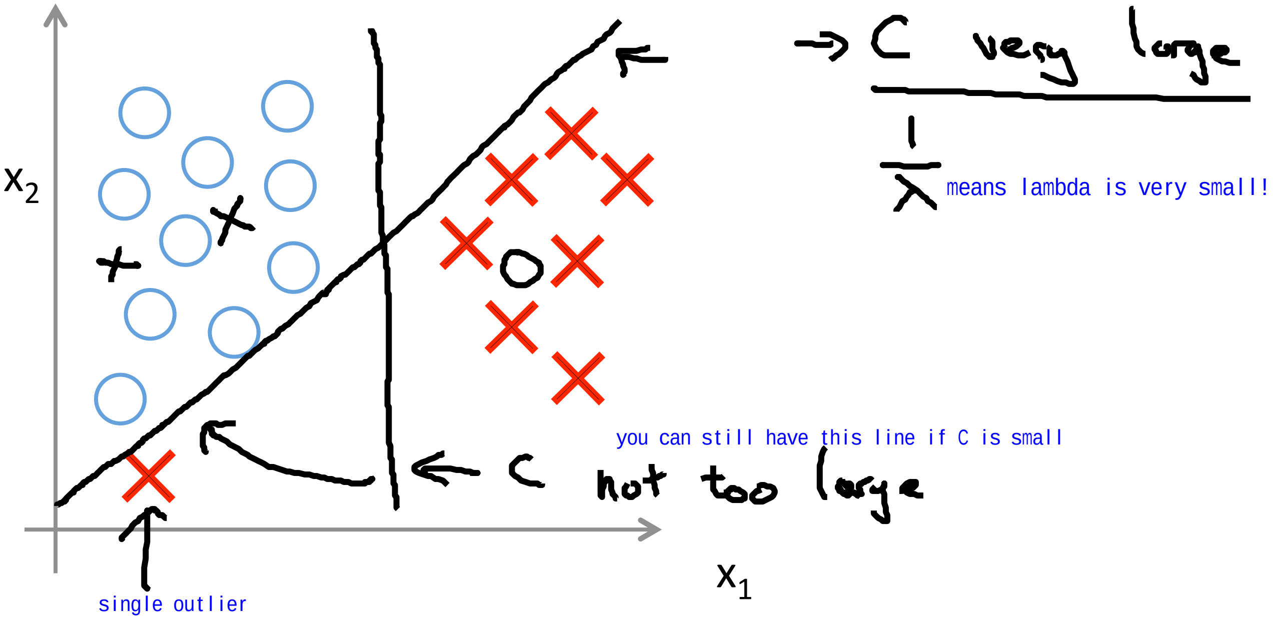
SVM will chose this boundary (better)

this distance is called the margin of SVC
this gives us certain robustness!
we want to separate data with the margin as large as possible!

Large margin classifier = SVM

Andrew Ng

Large margin classifier in presence of outliers



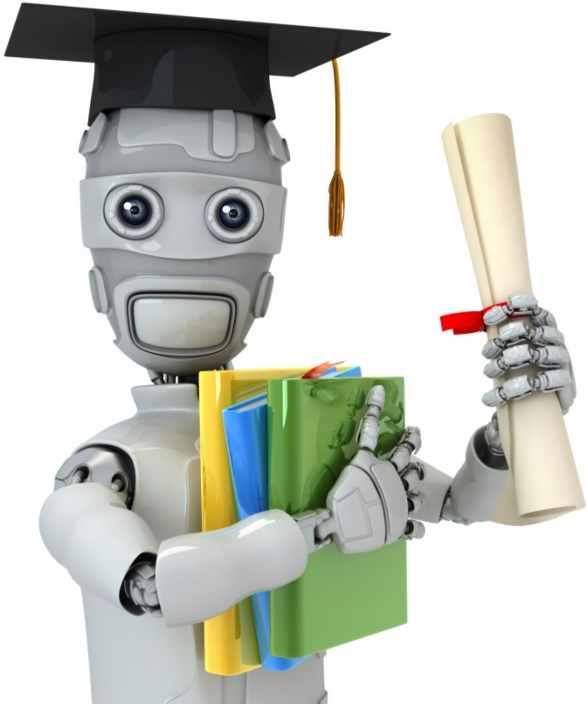
→ C very large

$\frac{1}{\lambda}$ means lambda is very small!

you can still have this line if C is small

C not too large

single outlier



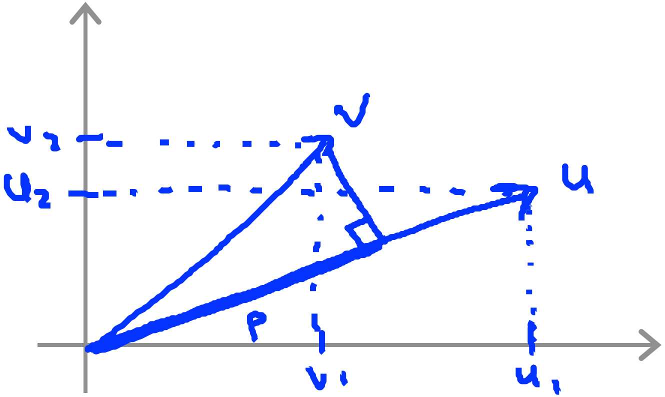
Machine Learning

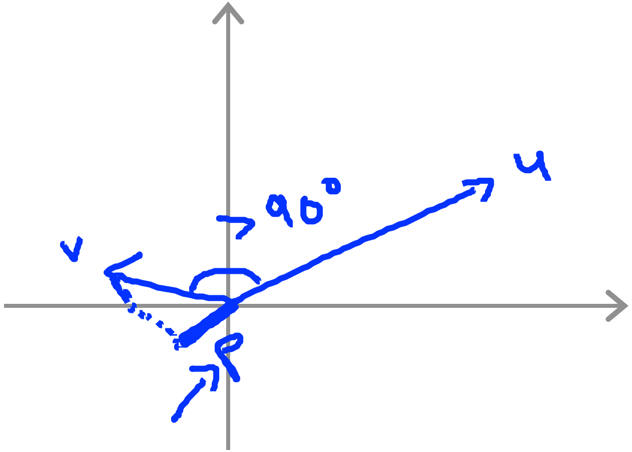
Gives some intuition about how the new optimization problem can result in a large margin classification problem.

Support Vector Machines

The mathematics behind large margin classification (optional)

Vector Inner Product





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

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

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

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

p = length of projection of v onto u.

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

Signed

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

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

$$p < 0$$

SVM Decision Boundary

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

optimization objective

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

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

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

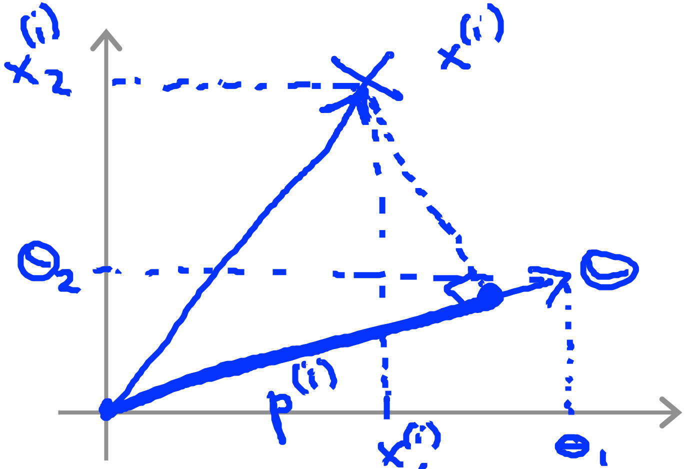
$$\begin{bmatrix} \cancel{\theta_0} \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad \theta_0 = 0$$

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

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

$$\uparrow \quad \uparrow$$

$$u^T v$$



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

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

A geometric view/derivation/intuition/proof of large margin classifier

SVM Decision Boundary

equivalently, we have the following optimization objective!

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

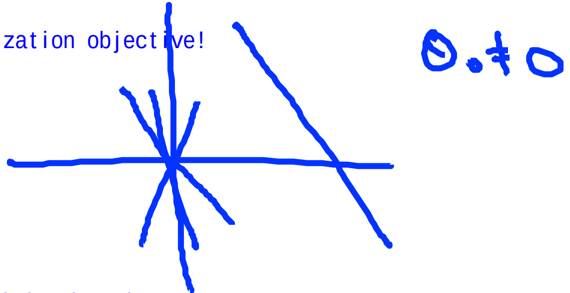
s.t. $\boxed{p^{(i)} \cdot \|\theta\| \geq 1}$ if $y^{(i)} = 1$

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

C very large

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

Simplification: $\theta_0 = 0$



decision boundary

corresponding theta

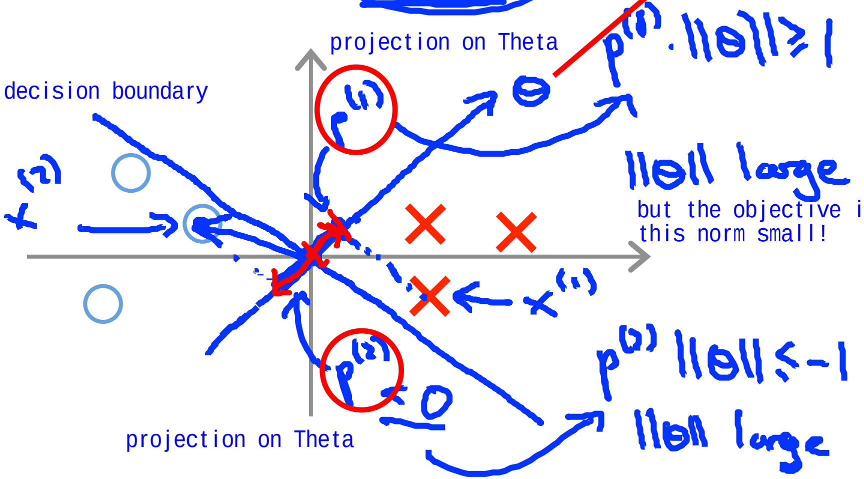
$\theta^T x = 0$

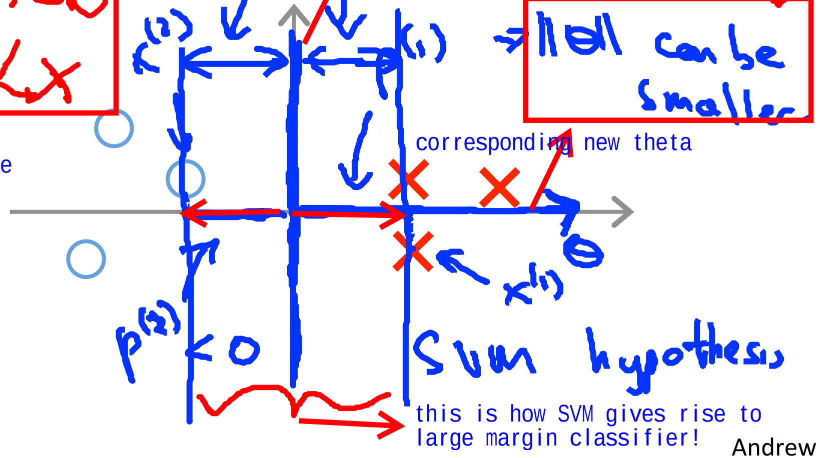
$\theta \perp x$

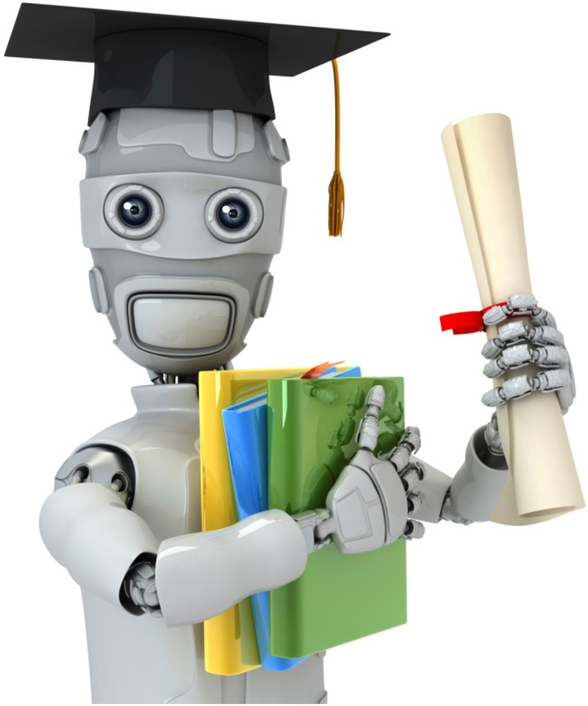
different decision boundary

margin

$p^{(1)} \|\theta\| \geq 0$







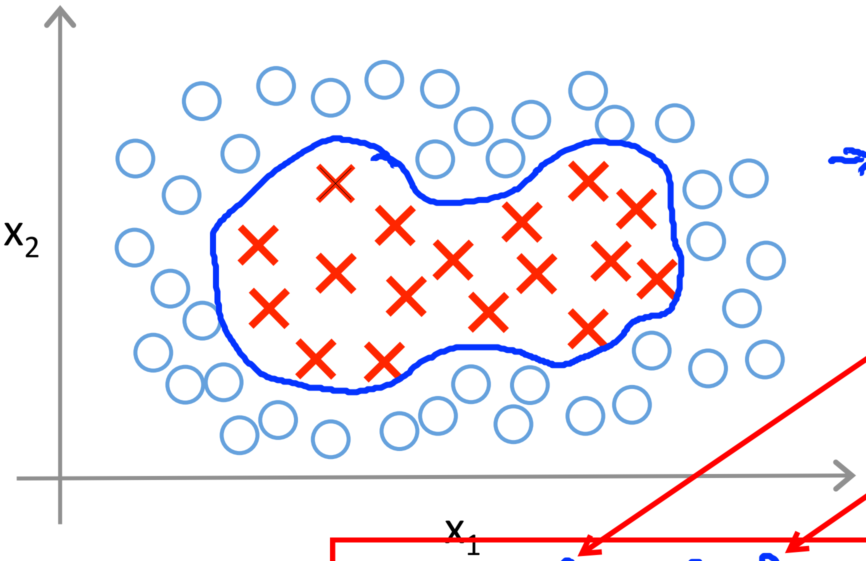
Machine Learning

Support Vector Machines

Kernels I

Main technique for adapting support vector machines in order to develop complex nonlinear classifiers.

Non-linear Decision Boundary



One way of doing this is to use polynomial fit

Predict $y = 1$ if

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

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

$$\rightarrow \boxed{\begin{array}{l} \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots \\ f_1 = x_1, \quad f_2 = x_2, \quad f_3 = x_1 x_2, \quad f_4 = x_1^2, \quad f_5 = x_2^2, \ldots \end{array}}$$

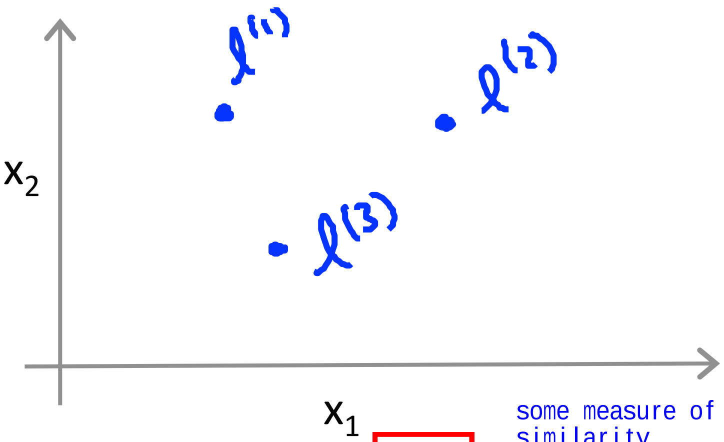
linear regression

high order polynomials

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

idea for developing new features

Kernel



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

measures similarity b/c x and l:
1. same, returns 1;
2. different, returns 0-1;

$\|w\|$

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

Given x:

some measure of similarity

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

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

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

new features

also denoted in this way

$\boxed{k(x, l^{(i)})}$

$\boxed{\text{kernel}}$ (Gaussian kernels)

The similarity function is called the kernel function

The specific kernel here is called the Gaussian Kernels ignoring x0, which is always 1

Kernels and Similarity

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

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

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

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

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

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

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

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

x

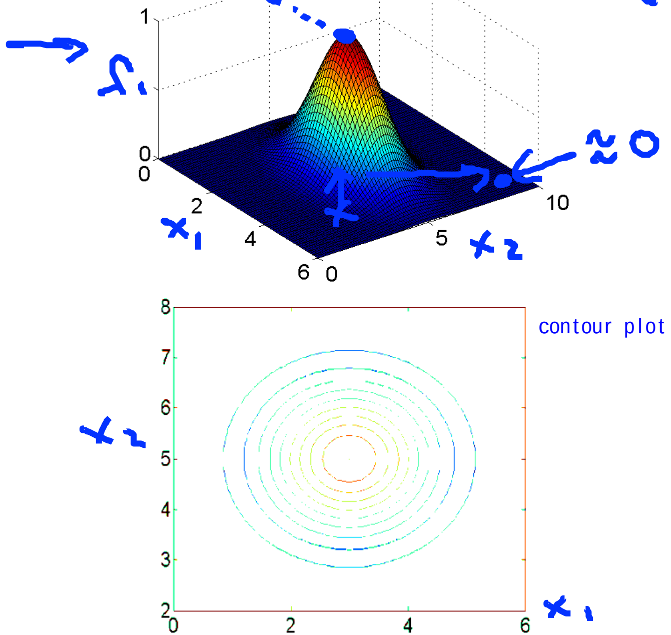
give a x, we can define three new features using ls

Example:

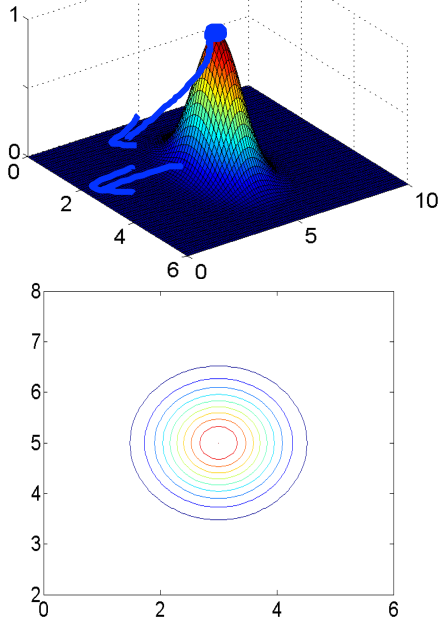
$$\rightarrow l^{(1)} = \boxed{\begin{bmatrix}3\\5\end{bmatrix}}, \quad \underline{f_1 = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\boxed{\sigma^2}}\right)}$$

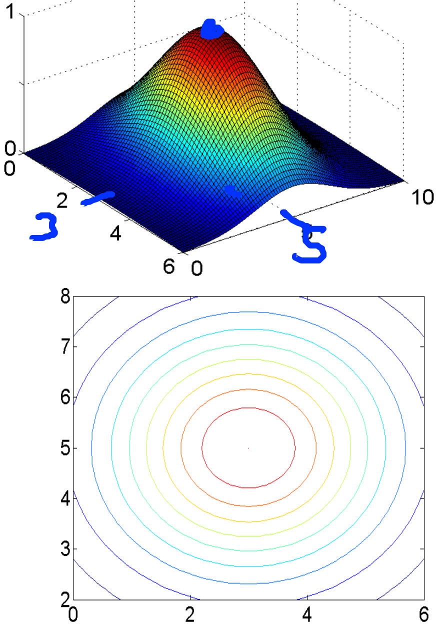
$\rightarrow \boxed{\sigma^2 = 1}$ $\qquad x = \begin{bmatrix}3\\5\end{bmatrix}$ $\qquad \boxed{\sigma^2 = 0.5}$ narrower $\qquad \underline{\boxed{\sigma^2 = 3}}$ wider

this measures how close the x is to the landmark l_1

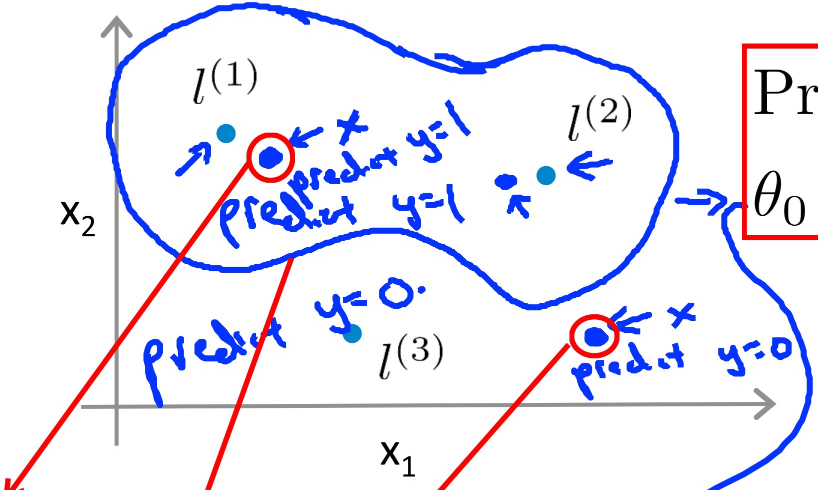


contour plot





Andrew Ng



a training example

another training example

for points close to l1 and l2, we predict positive
for points far away from l1 and l2, we predict negative
therefore, we will have a decision boundary shown above.
By using this, we can learn more complex classifiers!

Predict "1" when

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

x

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

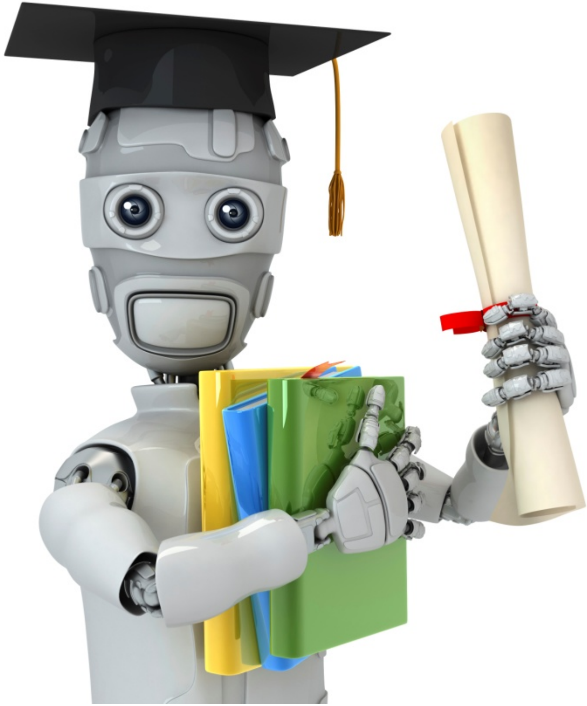
$f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0.$

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

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

$f_1, f_2, f_3 \approx 0$

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

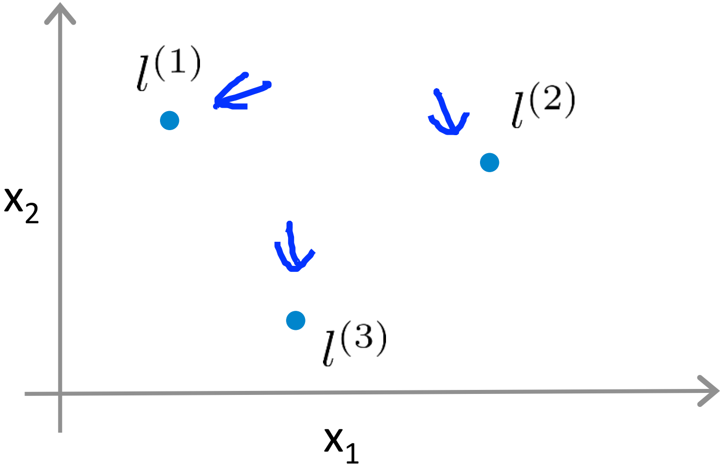


Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks



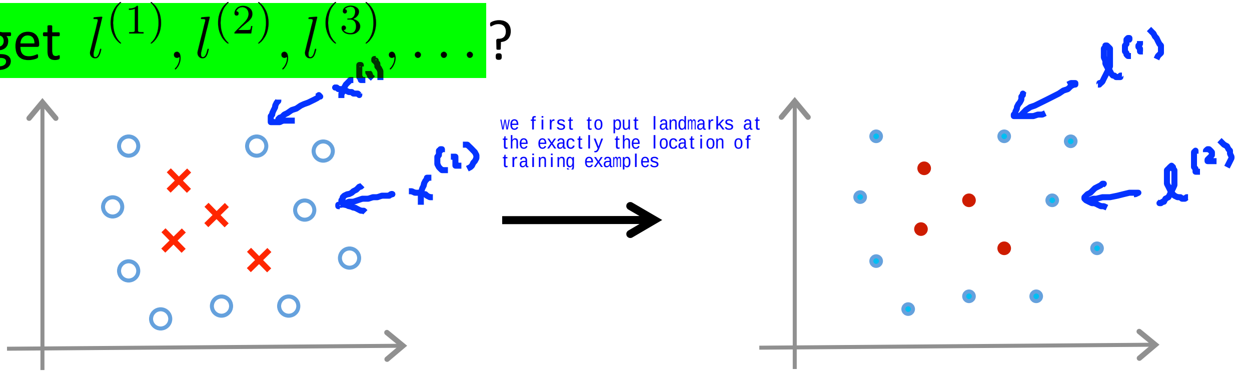
Given x:

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

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

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

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



we first to put landmarks at the exactly the location of training examples

with 1 landmark per location for each of my training examples

$$\boxed{\begin{matrix} l^{(1)} \\ l^{(2)} \\ \vdots \\ l^{(m)} \end{matrix}}$$

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: $x^{(i)}$

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

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

$\ldots$

feature vector

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

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

$x^{(i)} \rightarrow$

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

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

$\vdots$

$x^{(i)}$

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

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

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

we get a new feature vector that represent my training example xi

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

SVM with Kernels

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

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

Training: we use the support vector machine algorithm to train the theta parameter

$n = m$

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

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

→ θ_0

we do not regularize the theta_0 term

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

$\|\theta\|^2$

$m = 10{,}000$

$\boxed{\theta^T M \theta}$

we usually minimize this term, which allow the optimization software to run more efficiently.

SVM parameters:

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

more prone to overfitting

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

underfitting

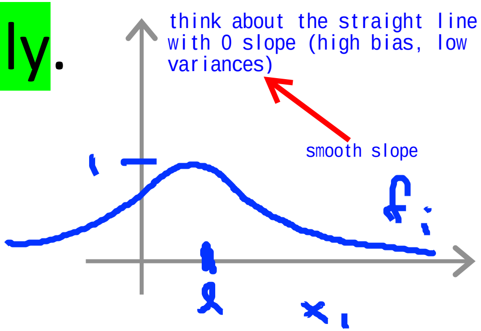
you will need to choose this Parameter in SVM

σ^2

Large σ^2: Features f_i vary more smoothly.

→ Higher bias, lower variance.

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

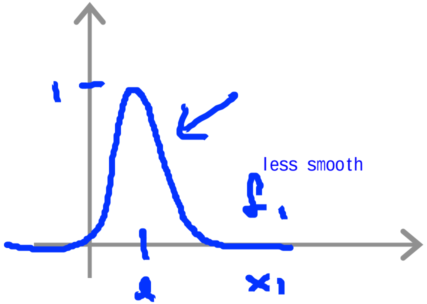


think about the straight line with 0 slope (high bias, low variances)

smooth slope

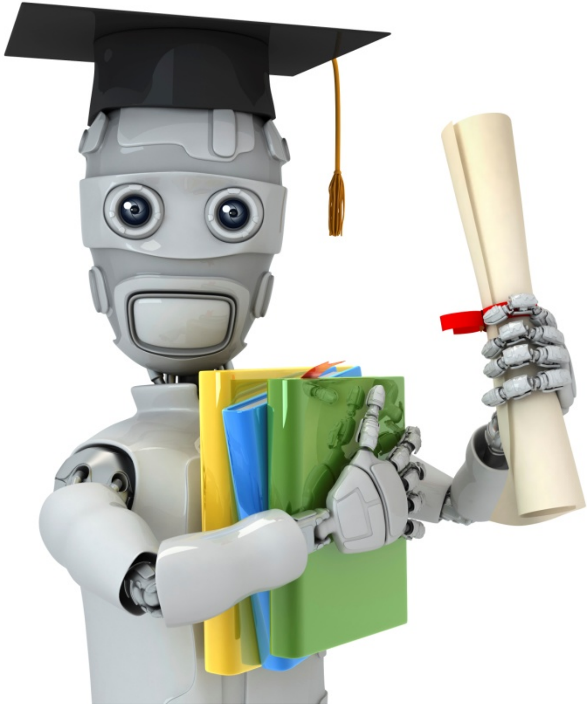
Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.



less smooth

Andrew Ng



Machine Learning

Support Vector Machines

Using an SVM

Poses a new optimization Problem but we do not recommend to write a new software to solve you problem. Just use the available one.

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

2 good optimization solver

Need to specify:

- Choice of parameter C.
- Choice of kernel (similarity function):

is also called the linear kernel

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

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

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

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

training samples

features

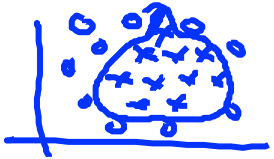
Gaussian kernel:

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

Need to choose $\underline{\sigma^2}$.

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

and/or m large



steps for implementing kernels

Kernel (similarity) functions:

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

landmarks

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

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

```
return
```

generate all the features

$$x \rightarrow \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{matrix}$$

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

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

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

$$v = x - l$$

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

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

1000 feet^2

1-5 bedrooms

perform feature scaling!!!

Andrew Ng

Other choices of kernel

linear and Guassian kernels are two most commonly used kernels

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).

main form

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

the idea is that
if x and l are very close
the product tends to be large

Many off-the-shelf kernels available:

- Polynomial kernel: $k(x, l) = (x^T l)^2 + 0$

  $(x^T l)^3$, $(x^T l + 1)^{3}$, $(x^T l + 5)^{4}$

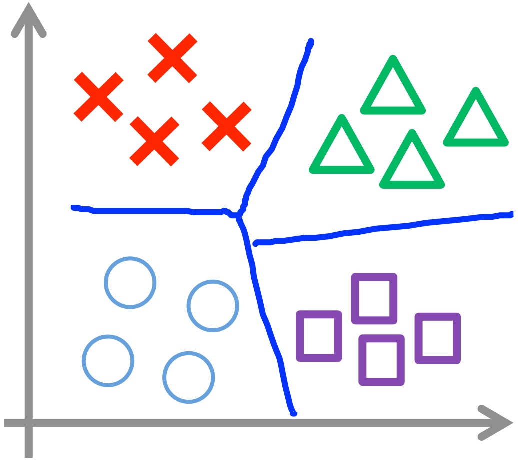
  people do not use it too much

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

  string

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

Multi-class classification



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

K Classes

as in logistic classification

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

y=1 y=2 .... θ=K

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

Andrew Ng

Logistic regression vs. SVMs

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

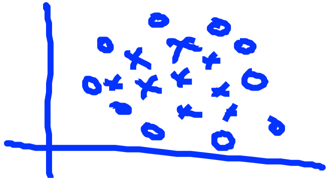
If n is large (relative to m): (E.g. $n \geq m$, $n = 10{,}000$, $m = 10 \cdots 1000$)

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

Gaussian kernel is slow!



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.