



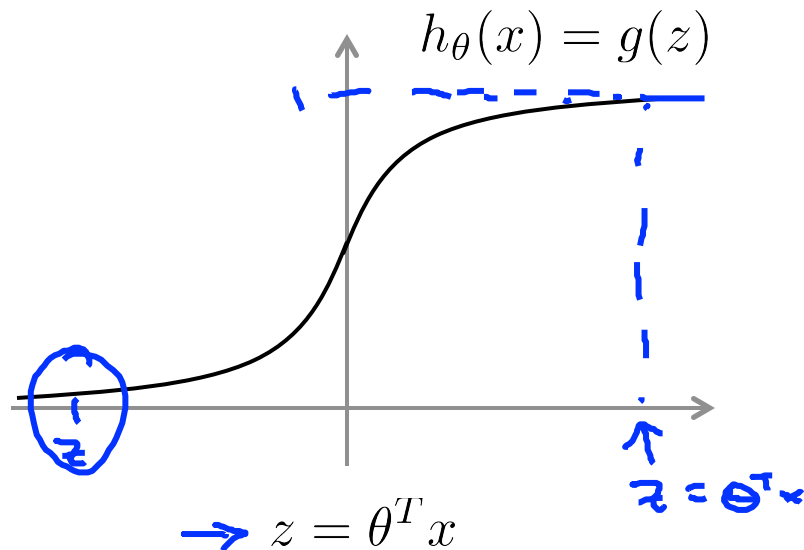
Machine Learning

Support Vector Machines

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

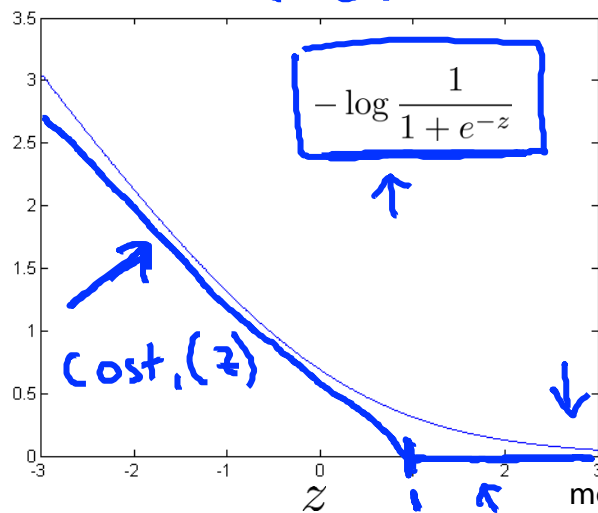
Alternative view of logistic regression

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

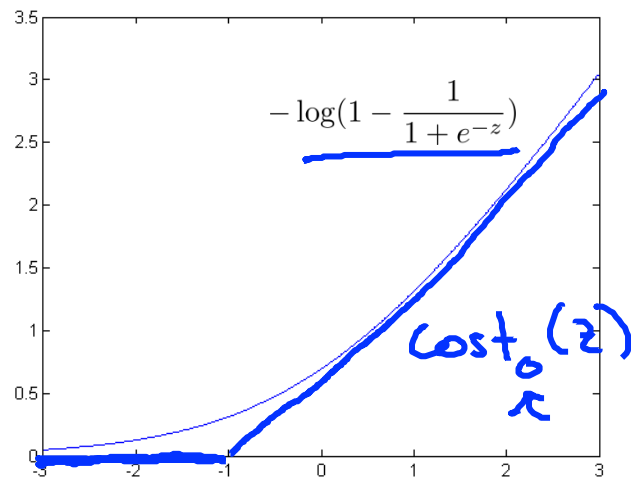
$$= -y \log \frac{1}{1 + e^{-\theta^T x}} - (1 - y) \log(1 - \frac{1}{1 + e^{-\theta^T x}})$$

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

$$z = \theta^T x$$



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



modify logistic regression to get support vector machine

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:

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

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:

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



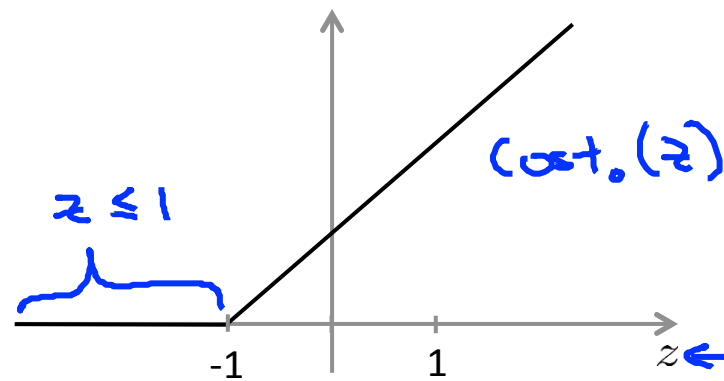
Machine Learning

Support Vector Machines

Large Margin Intuition

Support Vector Machine

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



\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)

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

$$C = 100,000$$

SVM Decision Boundary

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

$= 0$

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

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

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

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

$$\begin{aligned} \min_{\theta} & C \sum_{i=1}^m \theta_j + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \\ \text{s.t. } & \theta^T x^{(i)} \geq 1 \quad \text{if } y^{(i)} = 1 \\ & \theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0 \end{aligned}$$

SVM Decision Boundary: Linearly separable case



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



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

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

$$\|u\| = \text{length of vector } u \\ = \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$$

$p =$ length of projection of v onto u .

$$\begin{aligned} u^T v &= \underline{p} \cdot \underline{\|u\|} \leftarrow = v^T u \\ \text{Signed} \quad &= u_1 v_1 + u_2 v_2 \leftarrow p \in \mathbb{R} \end{aligned}$$

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

$$p < 0$$

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

SVM Decision Boundary

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

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

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

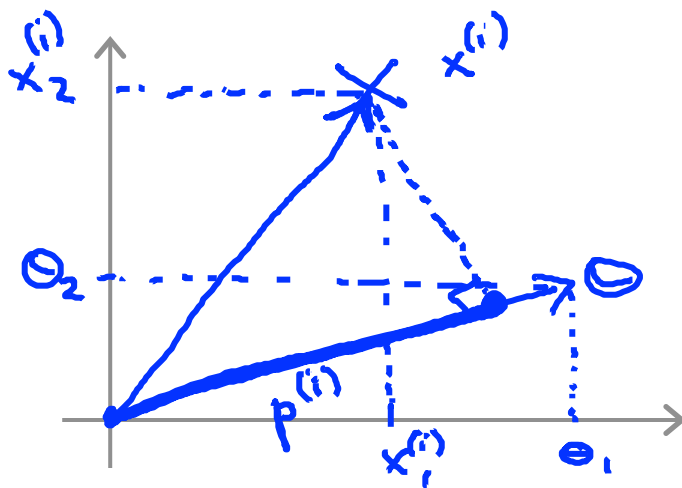
Simplification: $\theta_0 = 0$ $n=2$

$$= \|\theta\|$$

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

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

↑ ↑
u^T v



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

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

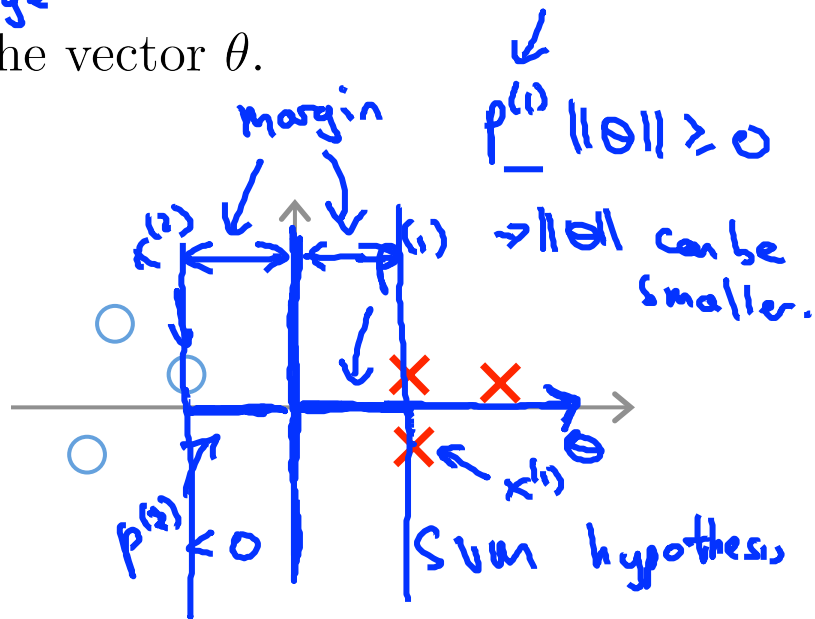
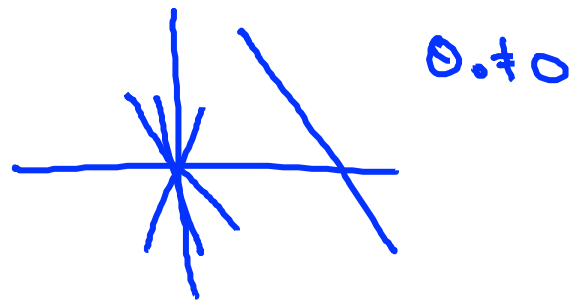
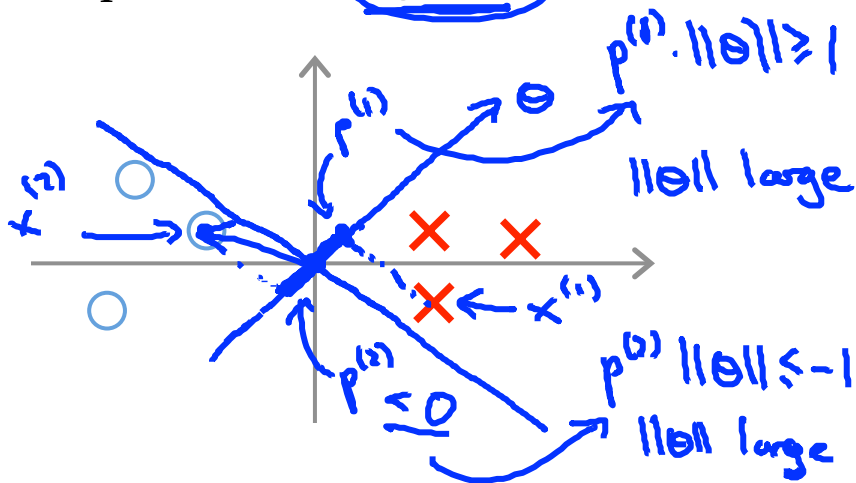
SVM Decision Boundary

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

$$\text{s.t. } \left. \begin{array}{ll} p^{(i)} \cdot \|\theta\| \geq 1 & \text{if } y^{(i)} = 1 \\ p^{(i)} \cdot \|\theta\| \leq -1 & \text{if } y^{(i)} = -1 \end{array} \right\} C \text{ very large}$$

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

Simplification: $\theta_0 = 0$





Machine Learning

Support Vector Machines

Kernels I

Non-linear Decision Boundary



Predict $y = 1$ if

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

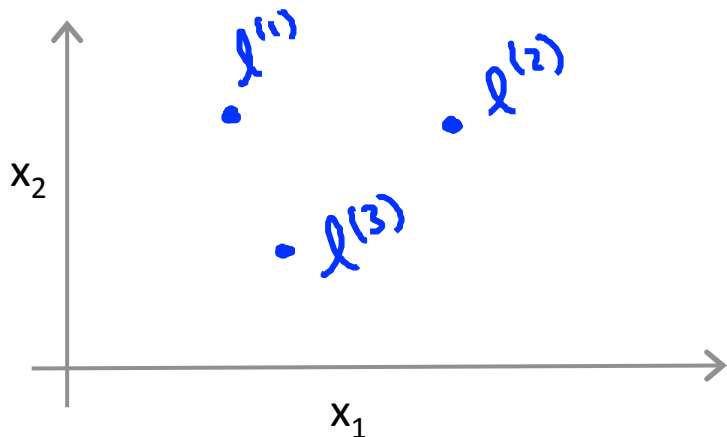
$$h_0(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \dots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

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

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

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 :

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

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

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

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

Kernels and Similarity

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

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

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

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

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

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

\uparrow \uparrow
X

Example:

$$\rightarrow l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

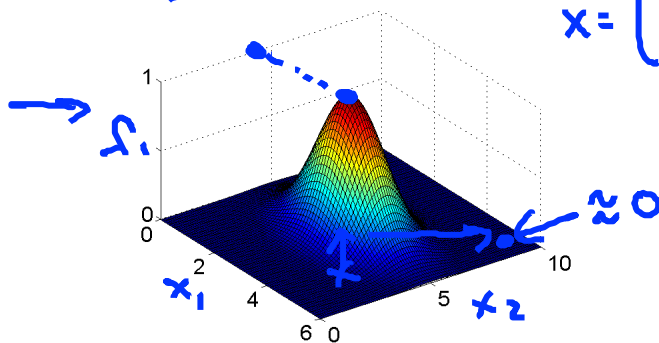
$$f_1 = \exp \left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2} \right)$$

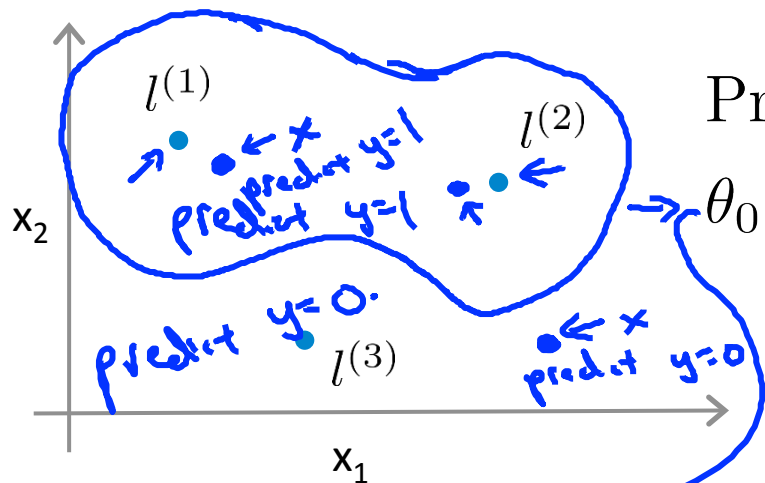
$$\rightarrow \sigma^2 = 1$$

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

$$\sigma^2 = 0.5$$

$$\sigma^2 = 3$$





Predict "1" when

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



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

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

$$\begin{aligned} \rightarrow \theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0 \\ = -0.5 + 1 = 0.5 \geq 0 \end{aligned}$$

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

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



Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks



Given x :

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

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

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

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



SVM with Kernels

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

Given example \underline{x} :

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

$\swarrow x^{(i)}$

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

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

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

$\swarrow x^{(i)}$

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

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

SVM with Kernels

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

→ Predict "y=1" if $\theta^T f \geq 0$

$$\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$$

$$\theta \in \mathbb{R}^{n+1}$$

Training:

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

Handwritten notes: $\theta^T f^{(i)}$, θ_0 , $n=m$, θ_j

$$\sum_{j=1}^m \theta_j^2 = \theta^T \theta$$

Handwritten notes: $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$, $\|\theta\|^2$, $\theta^T M \theta$, $(\text{ignore } \theta_0)$, $m = 10,000$

SVM parameters:

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

(small λ)

(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.





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 $\theta^T x \geq 0$

$$\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0 \quad \rightarrow \quad \underline{n} \text{ large, } \underline{m} \text{ small} \quad \underline{x \in \mathbb{R}^{n+1}}$$

→ 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 n large



Kernel (similarity) functions:

function $f = \text{kernel}(\underline{x1}, \underline{x2})$

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

return

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

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

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

$$v = x - l$$

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

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

1000 feet² 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:

$$k(x, l) = (x^T l)^3, \quad (x^T l)^2 + 1, \quad (x^T l + 5)^4$$

Handwritten annotations:
 - Above $(x^T l)^3$: $(x^T l)^2 + 0$ with an arrow pointing to the exponent 3.
 - Above $(x^T l + 1)^3$: $(x^T l + \text{constant})$ with an arrow pointing to the constant 1, and a circled 3 with an arrow pointing to the exponent.
 - Above $(x^T l + 5)^4$: $(x^T l + \text{constant})$ with an arrow pointing to the constant 5, and a circled 4 with an arrow pointing to the exponent. The word "degree" is written above the circled 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 $(\theta^{(i)})^T x$

↑
 $y=1$ ↑
 $y=2$... ↑
 $\theta = K$

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

→ Use logistic regression, or SVM without a kernel ("linear kernel")

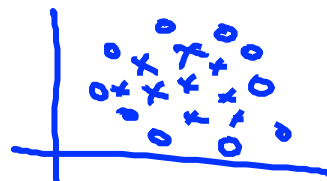
→ If n is small, m is intermediate:

($n = 1 - 1000$, $m = 10 - 10,000$) ←

→ 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.