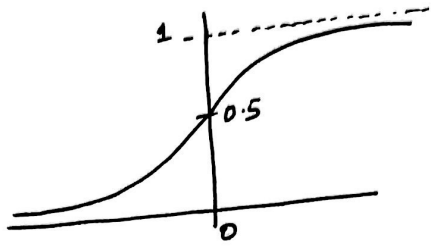
WEEK-7

Support Vector Machines

Large Margin Classification

→ Alternate view of logistic regression

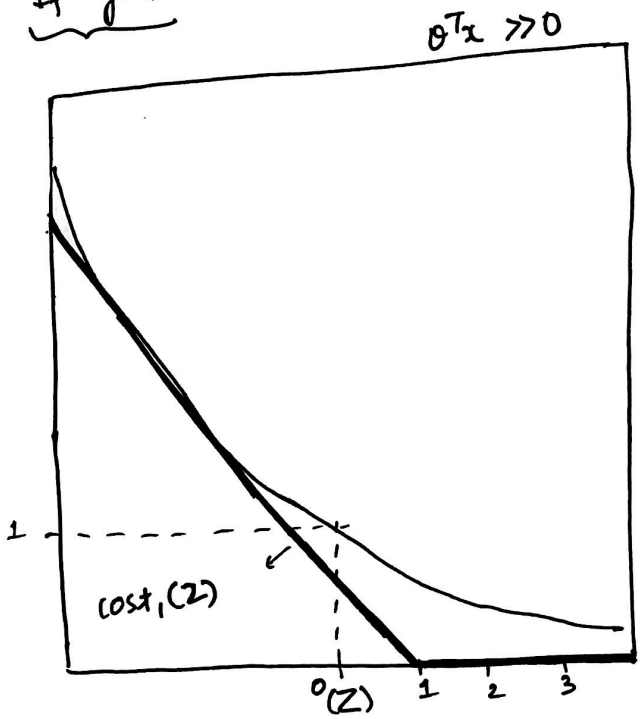
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



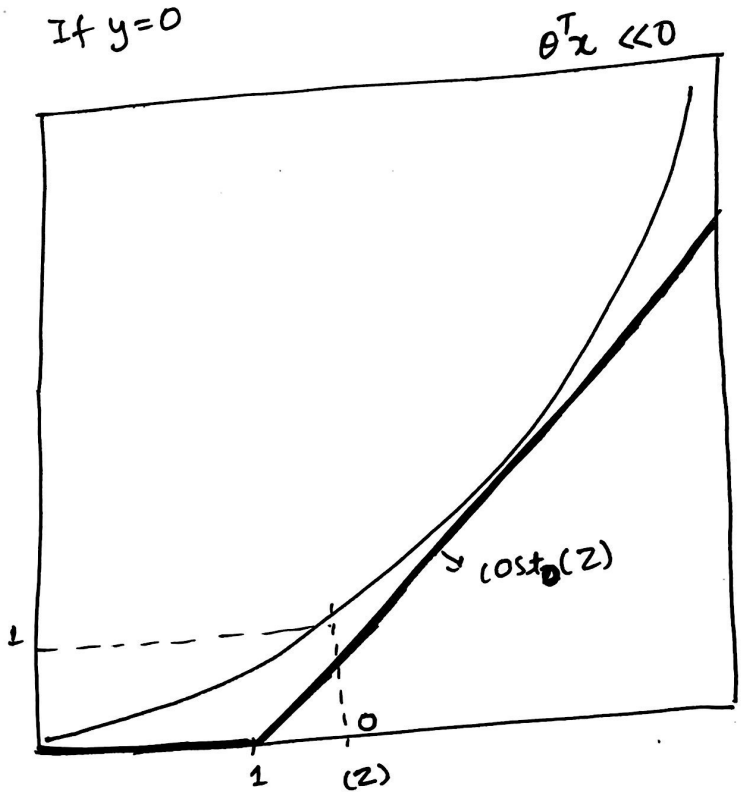
[if $y=1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$
if $y=0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$]

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

If $y=1$



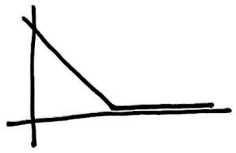
If $y=0$

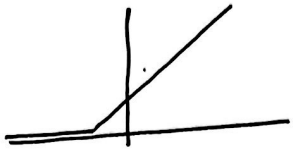


Support Vector Machine (SVM)

Logistic Regression:

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





SVM:

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

$+\ \lambda$ (B) — controlling the trade off b/w A & B.

$\Downarrow$

C (A) $+$ B — controlling the trade off. same as λ $\Rightarrow$ $\boxed{C \sim \frac{1}{\lambda}}$

cost fn. of SVM.

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

SVM hypothesis:

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

SVM Intuition:

if $y=1$, we want $\theta^T x \geq 1$ (not just ≥ 0)
if $y=0$, we want $\theta^T x \leq -1$ (not just < 0)
} extra Safety Margin

SVM Decision Boundary:

Case: C is very large, then $y^i \text{cost}_1(\theta^T x) + (1-y^i)\text{cost}_0(\theta^T x) \approx 0$

and

whenever $y^i = 1$

$\theta^T x \geq 1$

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

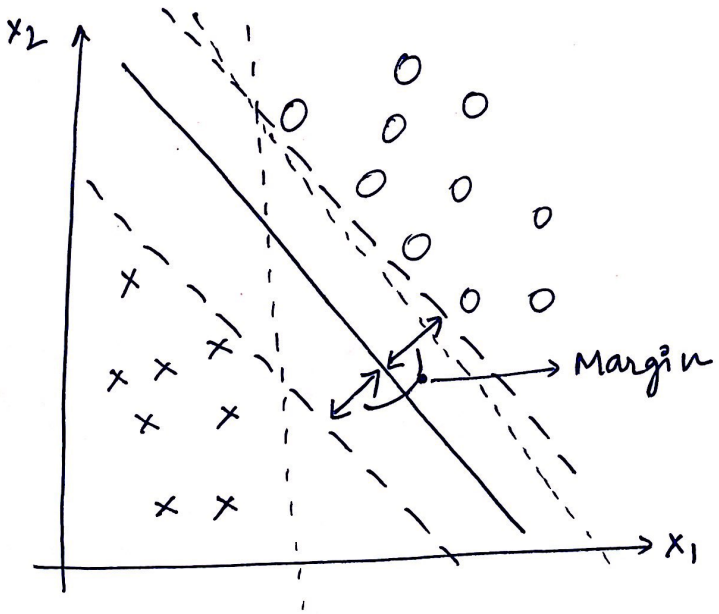
s.t. $\theta^T x \geq 1$

whenever $y^i = 0$

$\theta^T x \leq -1$

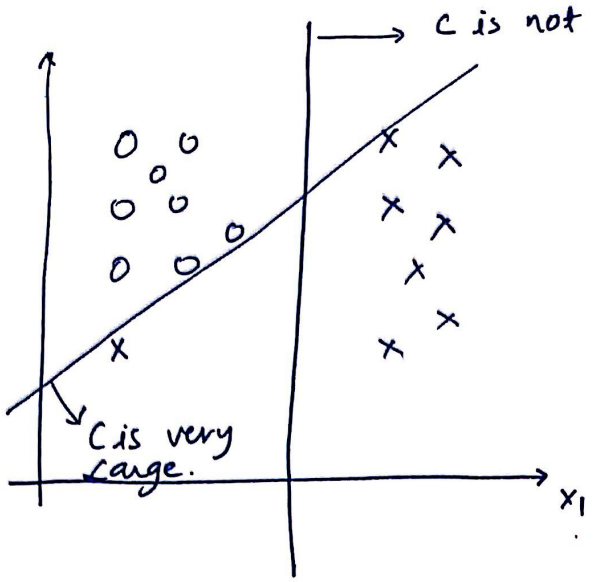
$$\min_\theta \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

s.t $\theta^T x \leq -1$



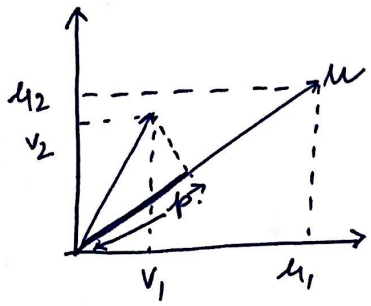
Therefore, SVM's are also called LMC (Large margin classifiers)

- **Large margin classifier in presence of outliers:**



Mathematics behind SVM

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



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

$= \sqrt{u_1^2 + u_2^2}$

p = length of projection of v onto u

signed.

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

$= u_1 v_1 + u_2 v_2$

→ SVM decision Boundary

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

s.t $\theta^T x \geq 1$ if $y^i = 1$

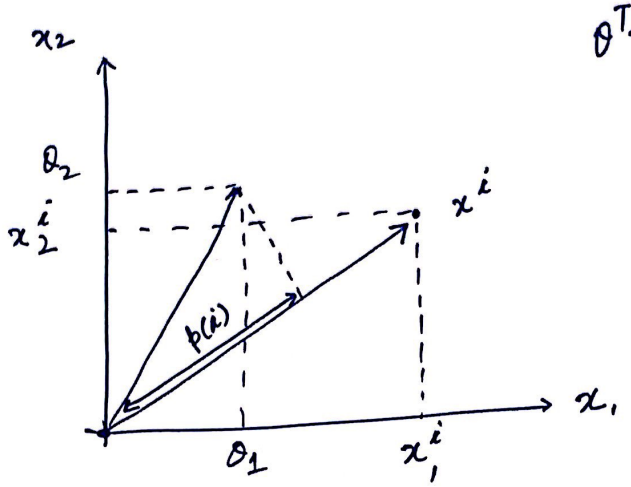
$\theta^T x \leq -1$ if $y^i = 0$

simplification, $\theta_0 = 0$

$n = 2 \Rightarrow \theta_1, \theta_2$

$\therefore$ obj. fn $= \frac{1}{2}\left(\sqrt{\theta_1 + \theta_2}\right)^2$

$= \frac{1}{2} \|\theta\|^2$



$\theta^T x^i = p^i \cdot \|\theta\|$

$= \theta_1 x_1^i + \theta_2 x_2^i$

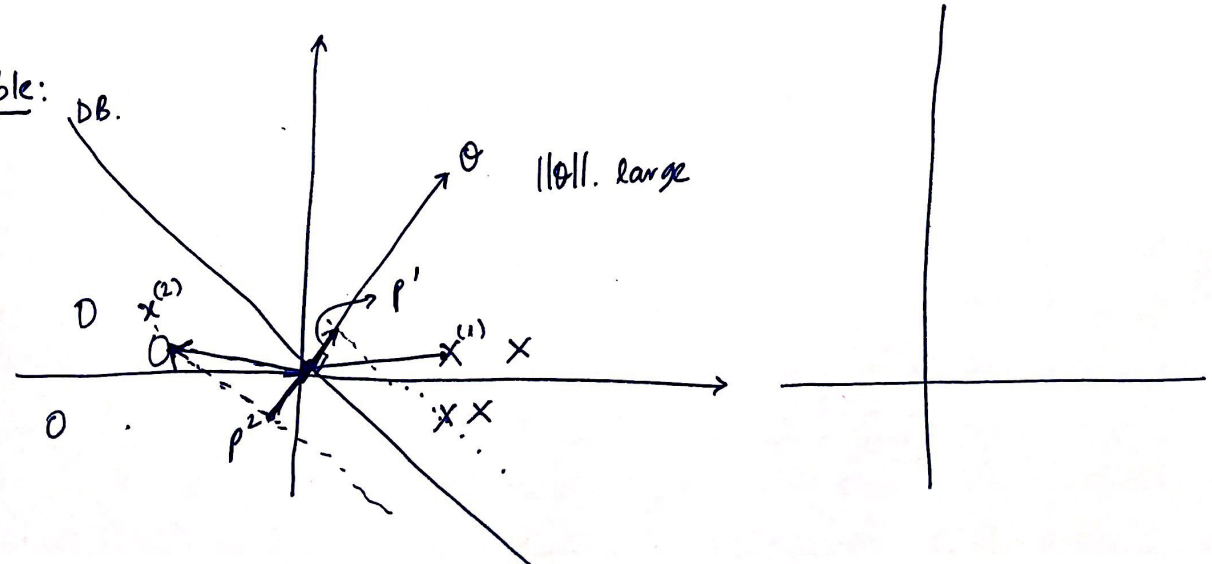
$\therefore$ ~~Σ~~ New objective:

$$\min_{\theta} \frac{1}{2} \cdot \|\theta\|$$

s.t. $p^i \cdot \|\theta\| \geq 1$ $\forall$ $y^i = 1$

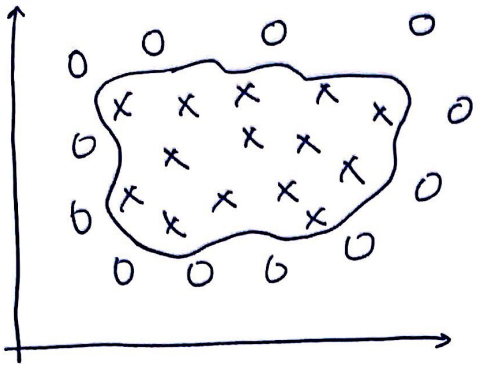
$p^i \cdot \|\theta\| \leq -1$ $\forall$ $y^i = 0$

example:



KERNELS

Non-linear Decision Boundary

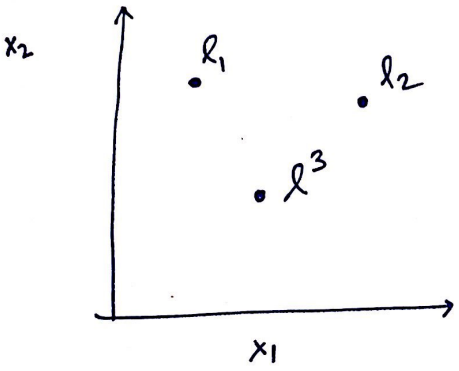


predict $y=1$ if

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \dots \theta_{27} x_1^6 x_2^6 \dots \geq 0$

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

is there a different / better choice of the features?
(higher order polynomials)



Given x, compute new feature depending on proximity to landmarks l^1, l^2, l^3

Given x:

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

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

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

↑

Kernal (Gaussian Kernals)

$k(x, l^i)$

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

If $x \approx l^1$

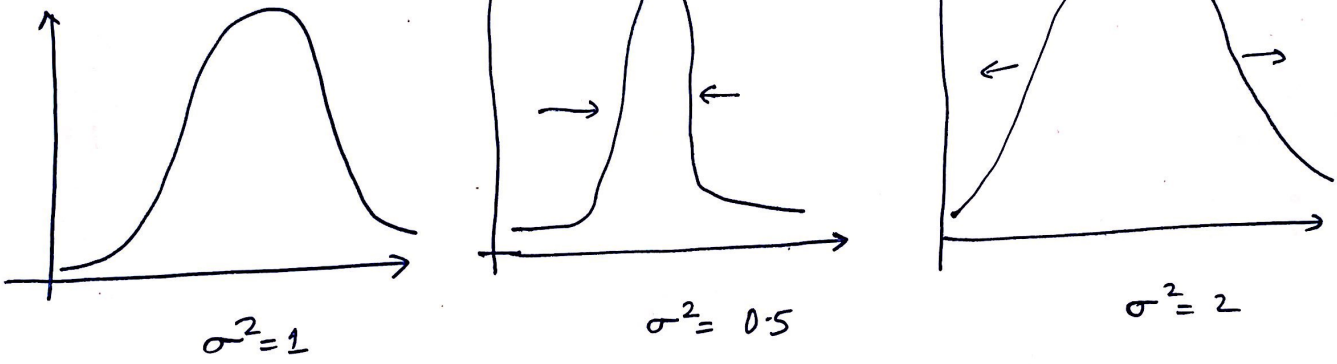
$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$$

If x is far from l^2

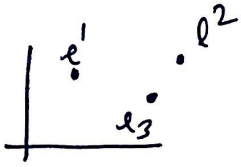
$$f_1 \approx \exp\left(-\frac{\text{large number}^2}{2\sigma^2}\right) \approx 0$$

$l_1 \rightarrow f_1$
$l_2 \rightarrow f_2$
$l^3 \rightarrow f_3$
$\sim$
x^i

{ new set of features based on landmarks }



New Hypothesis:

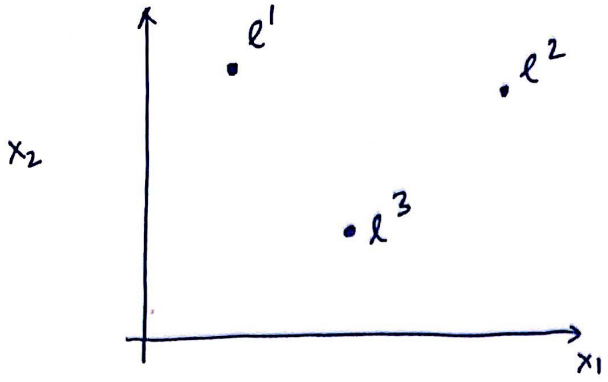


Predict (1) when

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$ } $\Rightarrow$ $\theta_0 = -0.5$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 0$

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

choosing the landmarks:-



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

[for every training example, choose landmark at exactly same position!] ** **

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

choose: $l^1 = x^1, l^2 = x^2, \ldots l^m = x^m$

$$\therefore f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1 \quad \leftarrow \quad f \in \mathbb{R}^{m+1}.$$ where

for training example (x^i, y^i)

$$\therefore x^i \rightarrow \begin{cases} f_1^i = sim(x^i, l^1) \\ f_2^i = sim(x^i, l^2) \\ \vdots \\ f_m^i = sim(x^i, l^m) \end{cases} \leftarrow f_i^i = sim(x^i, l^i) \times 1$$

$x^i \in \mathbb{R}^{n+1} \Rightarrow$ $\boxed{f^i \in \mathbb{R}^{m+1}}$ vector $\quad f_0^i = 1$

feature vector.

SVM with Kernals (Algorithm)

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

predict "$y=1$" if $\theta^T f \geq 0$ $\quad \theta \in \mathbb{R}^{m+1}$

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \dots + \theta_m f_m$

Training:

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

$$\sum_{j} \theta^T \theta$$

$$\downarrow$$

$$\theta^T M \theta$$

SVM optimization.
* computational eff.

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}$$

SVM Parameters:

$C = \frac{1}{\lambda}$

- Large C: Lower bias, high variance
- Small C: Higher bias, lower variance

σ^2

- Large σ^2: features f_i vary more smoothly. Higher bias, lower variance.
- Small σ^2: features f_i vary less smoothly. Lower bias, higher variance (overfit)

USING AN SVM

→ Use SVM Software (eg. liblinear, libsvm ...)

→ Need to specify :
- ↳ choice of parameter C
- ↳ choice of kernel.

eg. No. Kernel. ("linear kernel.")

predict "$y = 1$" if $\theta^T x \geq 0$

(n large, m small)

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

Guassian Kernel.

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

Need to choose σ^2

$x \in \mathbb{R}^{n+1}$, (n small. and m is large)

* Note: Do perform feature scaling before using the Guassian Kernel.

function f = Kernel (x_1, x_2)

$$f = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\cdot\sigma^2}\right)$$

return.

$x_1 \in x^i$

$x_2 \in l^i$

- Other choices of Kernel

Not all the similarity functions similarity (x, ℓ) make valid kernels. (Need to satisfy technical condition called "Mercer Theorem" to make sure SVM packages' optimizations run correctly and do not diverge).

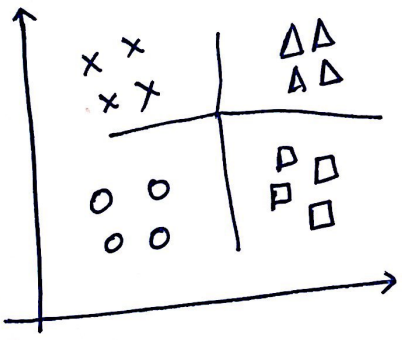
Many off-the-shelf kernels available

→ Polynomial Kernel: $k(x, \ell) = (x^T \ell)^2$

↳ $(x^T \ell + \text{constant})^{\text{degree}}$.

→ Esoteric: string kernel, chi-square kernel, histogram intersection kernel, ...

MULTI-CLASS CLASSIFICATION



Many SVM packages already have built-in multi-class classification functionality

Otherwise, use one-vs-all method

① Train K SVM's, one to distinguish $y = i$ from the rest, for $i = 1, 2, \ldots K$

② get $\theta^1, \theta^2, \ldots \theta^K$

③ Pick class i with largest $(\theta^i)^T x$

Logistic Regression vs. SVMs

① if $n \geq m$ $\quad n = 10{,}000$, $m = 10\ldots$ or 1000

use logistic regression or SVM without a kernel (Linear kernel)

② if n is small, m is intermediate.

$n = 1 - 1000$
$m = 10 - 10{,}000$

use SVM with Gaussian kernel.

③ If n is small, m is large

$n = 1 - 1000$
$m = 50000+$

Create/add more features, then use Logisitic Regression or SVM without a kernel.

Similar Algo's.

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

Octave code : for CV test.

```
for C_test = [0.01 0.03 0.1 0.3 1 3 10 30]
  for sigma_test = [0.01 0.03 0.1 0.3 1 3 10 30]
      i = i+1
      model = svmTrain(X, y, C_test, @(x1, x2) gaussianKernel
              (x1, x2, sigma_test));
      % svm package to minimize
      predictions = svmPredict(model, Xval)
      prediction_error = mean(double(predictions ~= yval));

      results(i, :) = [C_test sigma_test prediction_error]
  endfor
end for.

sorted_results = sortedrows(results, 3)

C = sorted_results(1, 1);
sigma = sorted_results(1, 2);
```