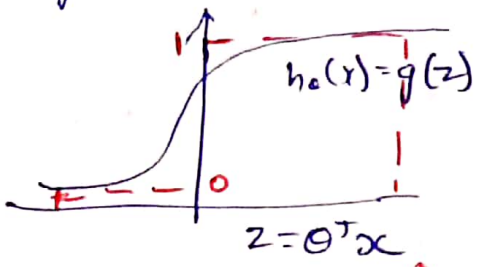
Week 7 – Support Vector Machine

Derivation

Consider logistic regression

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



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

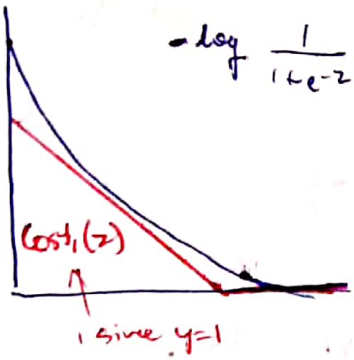
If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

Cost of logistic regression:

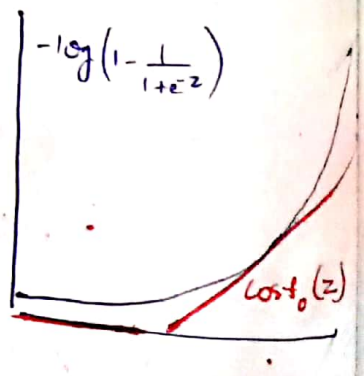
$$-(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$$

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

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



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



∴ Logistic regression:

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

(A) (B)

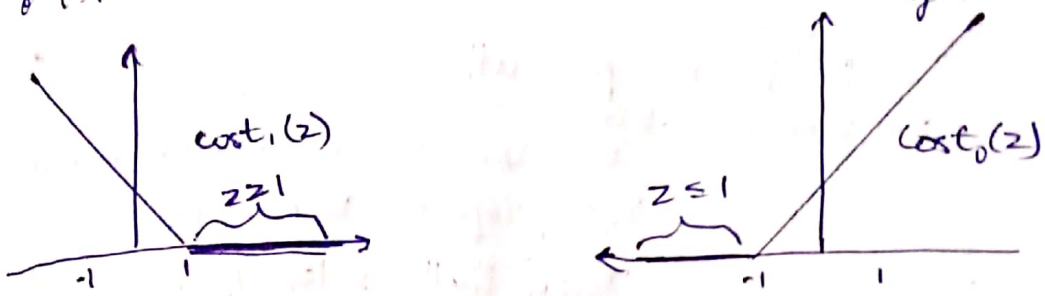
- Cancel m terms
- Before $A + \lambda B \Rightarrow$ Now $CA + B$ ← C is like $\frac{1}{\lambda}$

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

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

Support Vector machine

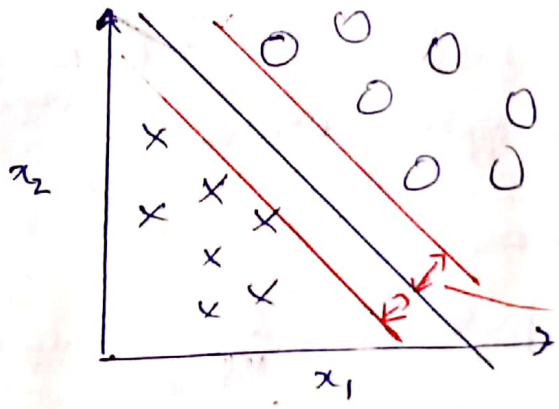
$$\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$



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)

SVM decision boundary:



This margin comes because we are using $\theta^T x \geq 1$ & $\theta^T x \leq -1$ instead of $\theta^T x \geq 0$ & $\theta^T x \leq 0$

∴ It is also called a large margin classifier

How does value of C affect it?

Lets say we introduce a new point x. Then if C is large, the line shifts

If C is small, it remains same

We may want to keep the line same since we don't want to change it so much just for one data value

Mathematics behind Large Margin Classification

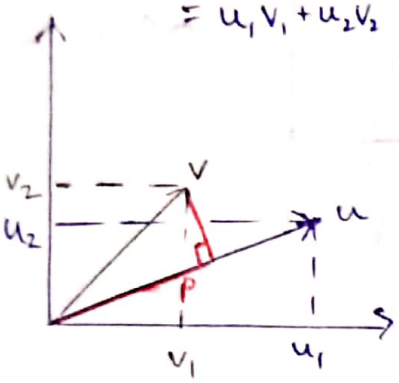
Vector inner Product:

$u^T v$ is called vector inner product

i.e $[u_1 \; u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$.

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

$$= u_1 v_1 + u_2 v_2 = p \cdot \|u\|$$



Consider the 2 vectors v and u

We project v on u.

Then $\|u\|$ is length of vector u

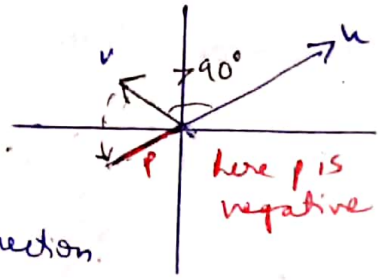
$\|u\| = \sqrt{u_1^2 + u_2^2}$ - From Pythagoras Theorem

↑ norm of vector u

p = length of projection v onto u,

Here, p can also be negative if the projection angle > 90°

So we project v in opposite direction.



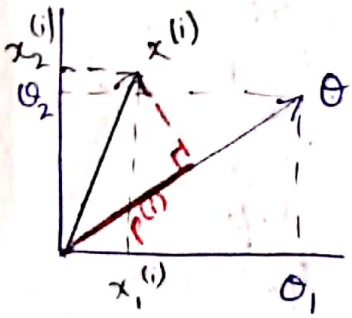
SVM decision boundary:

$\min_\theta \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$ Suppose $\theta_0 = 0$ & $n = 2$ (only θ_1 & θ_2)

$\therefore \; \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \frac{1}{2}\|\theta\|^2$ ($\|\theta\|$)

This holds true since $\theta_0 = 0$ $\begin{bmatrix} 0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$

$\therefore$ We are basically trying to minimize length of vector θ



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

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

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

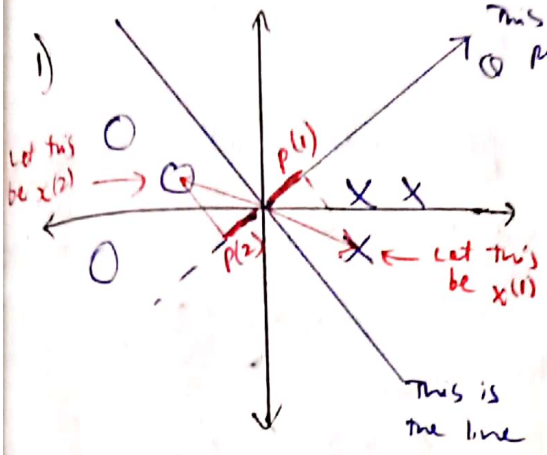
$$\therefore \min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\theta\|^2$$

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

$p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)} = 0$

consider 2 cases to see how SVM finds the decision boundary: Let $\theta_0 = 0$. This means line needs to pass through origin

1)



We can see here that $p(1)$ and $p(2)$ are small

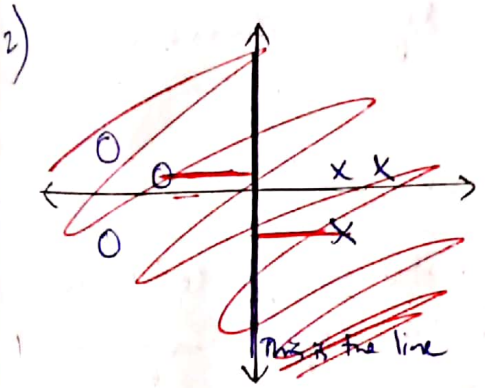
$\therefore$ For $p^{(1)} \cdot \|\theta\| \geq 1$

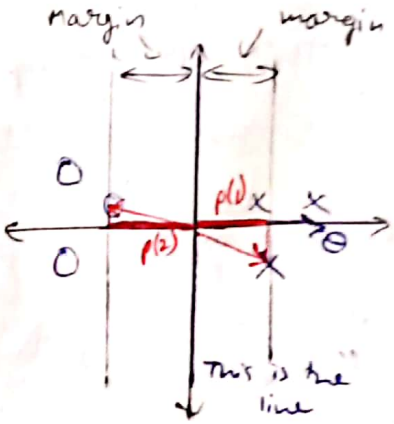
& $p^{(2)} \cdot \|\theta\| \leq -1$

$\therefore \|\theta\|$ needs to be large

But we are minimizing $\|\theta\|$ since $\frac{1}{2}\|\theta\|^2$, so this line isn't good

2)





Notice here that $p(1)$ and $p(2)$ are large

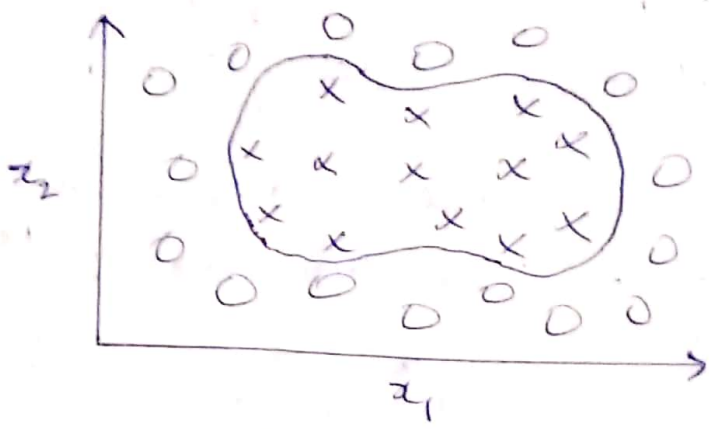
$\therefore$ For $p(1) \cdot \|\theta\| \geq 1$, $\|\theta\|$ can be small

$p(2) \cdot \|\theta\| \leq -1$, $\therefore$ This is what SVM will choose

$p(1), p(2), p(3), \ldots$ is basically the margin

Here we have set $\theta_0 = 0$ and confined it to the origin, but its the same logic even if $\theta_0 \neq 0$ and the line doesn't pass through origin.

Non-linear Boundaries



Usually we use polynomial terms

Predict $y=1$ if,

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots$

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

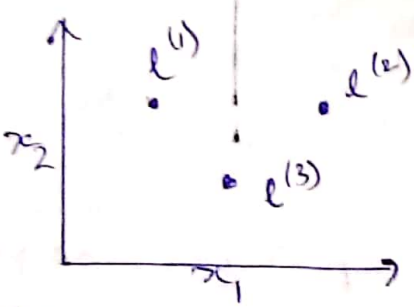
We will write this as $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$

where $f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, $f_4 = x_1^2$, $f_5 = x_2^2$

But this is computationally intensive.

So we use **Kernels**.

Kernel



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

Given x,

norm of $x - l^{(1)}$

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

This is sig not s

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

$$f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(-\frac{\|x - l^{(3)}\|^2}{2\sigma^2}\right)$$

↑ Kernel

↗ Gaussian kernel

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

Lets take f_1,

f_1 = 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 $l^{(1)}$:

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

So each landmark ~~gives~~ defines a new feature f

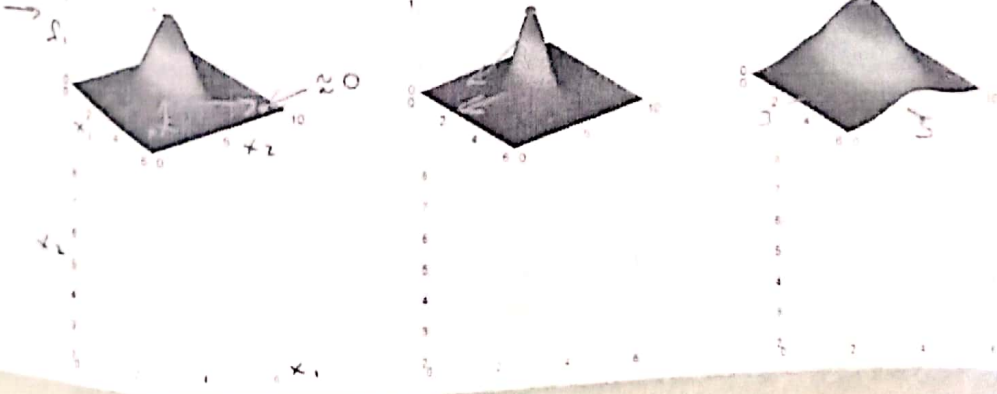
$l^{(1)} \rightarrow f_1$; $l^{(2)} \rightarrow f_2$, $l^{(3)} \rightarrow f_3$

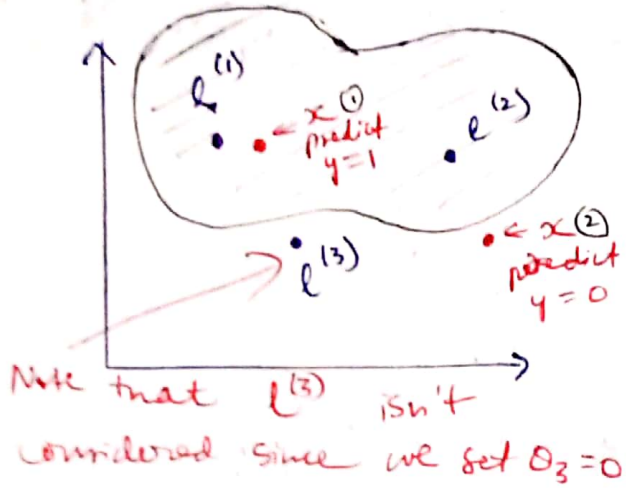
SICK PRINTOUT

Example:

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

$\rightarrow \sigma^2 = 1$ $\quad x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$, $\sigma^2 = 0.5$ $\quad \sigma^2 = 3$





Note that $l^{(3)}$ isn't considered since we set $\theta_3 = 0$

Predict 1 when

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

Lets set

$\theta_0 = -0.5$

$\theta_1 = 1$

$\theta_3 = 0$ as an example

Consider 2 cases:

① Here x is close to $l^{(1)}$

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

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

$= -0.5 + 1 = 0.5 \geq 0 \quad \therefore y = 1$

② Here x is far from $l^{(1)}$, $l^{(2)}$ and $l^{(3)}$

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

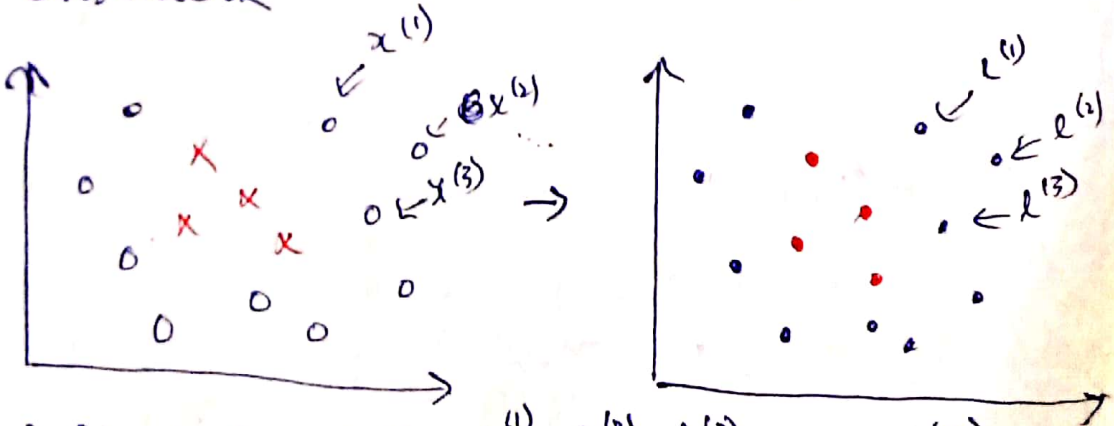
$= \theta_0 + \theta_1 \times 0 + \theta_2 \times 0 + \theta_3 \times 0$

$= -0.5 \leq 0 \quad \therefore y = 0$

Choosing landmarks:

Before we chose $l^{(1)}, l^{(2)}, l^{(3)}$, but how do we choose it in problem?

We take all the training points as landmark



$\therefore$ We end up with $l^{(1)}, l^{(2)}, l^{(3)} \ldots l^{(m)}$

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,

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

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

...

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

where $f_0 = 1$ (we set this like $\theta_0 = 1$)

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

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

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

$\vdots$ ← somewhere in the middle $f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(\frac{-0}{2\sigma^2}\right) = 1$

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

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

SVM with kernels

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

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

Training:

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

(n is equal to m)

Notice this, we are using this instead of $\theta^T x^{(i)}$

We an already built software to minimise J. It has included in it,

Mathematically this terms is different. They use $\theta^T M \theta$, but treats only to decrease computational intensity.

SVM parameters

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

SVM in Practice

- Use SVM software package (eg. liblinear, libsvm, ...) to solve for parameters θ.

Need to specify:
- Choice of parameter C
- Choice of kernel (similarity function):

E.g. ① No kernel (linear kernel):
Predict $y = 1$ if $\theta^T x \geq 0$
We use this when n (no. of features) is large but m (no. of training examples) is less.

② Gaussian kernel:
$f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$, where $l^{(i)} = x^{(i)}$
- Need to choose σ^2
We use this when n is small and/or m is large.

If we choose Gaussian kernel:

- The SVM software package will ask you to implement the kernel (similarity) function

function f_i = kernel($x1$, $x2$) — $x^{(i)}$, $l^{(j)} = x^{(j)}$

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

return

Note: Perform feature scaling before using the Gaussian kernel.

- The SVM software package will then use this function to generate all the features $f_1, f_2, f_3 \ldots f_m$ and train the SVM.

- Some SVM software packages will have inbuilt kernel functions and will allow you to choose

Other choices of kernels:

- All available kernels need to satisfy "Mercer's Theorem" to make sure SVM packages' optimizations run correctly

① Polynomial kernel: $K(x, l) = (x^T l + \text{constant})^{\text{degree}}$
$= (x^T l)^2, (x^T l)^3, (x^T l + 1)^3 \ldots$

② More esoteric kernels: 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 (Train K SVMs, one to distinguish $y = i$ from the rest, for $i = 1, 2, \ldots, K$), get $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$. Pick class i with largest $(\theta^{(i)})^T x$

($\theta^{(1)}$: $y=1$, $\theta^{(2)}$: $y=2$, $\theta^{(K)}$: $y=K$)

Logistic regression vs. SVMs.

n = no. of features, m = no. of training examples

→ If n is large (relative to m):
- Use logistic regression or SVM without a kernel ("linear kernel")

Eg. $n = 10{,}000$ $m = 10$ or 1000

→ If n is small, m is intermediate:
- Use SVM with Gaussian kernel

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

→ If n is small, m is large:
- Then SVM will be slow. So create/add more features, then use logistic regression or SVM without a kernel

They both perform almost similarly

Eg. $n = 1 - 1000$ $m = 50{,}000+$

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