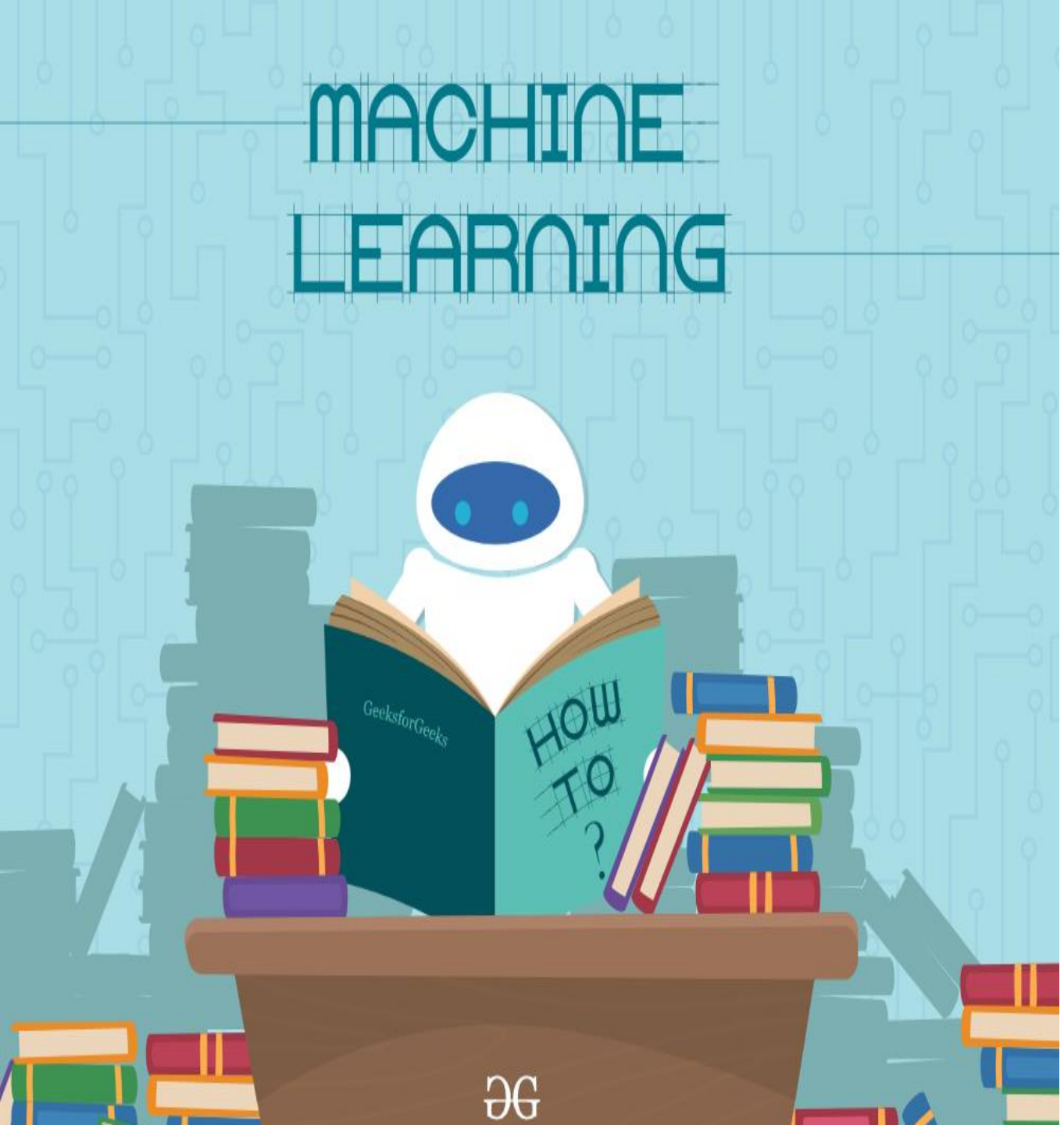



>TCC
THE CODERS' CLUB

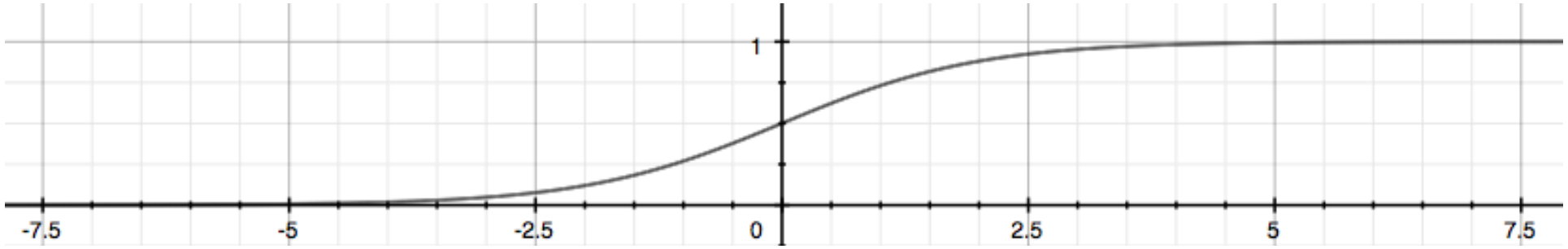
# Support Vector Machines (SVM)

WEEK 7

# 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 $h_\theta(x) \approx 1$, $\theta^T x >> 0$
- If y=0, we want $h_\theta(x) \approx 0$, $\theta^T x << 0$

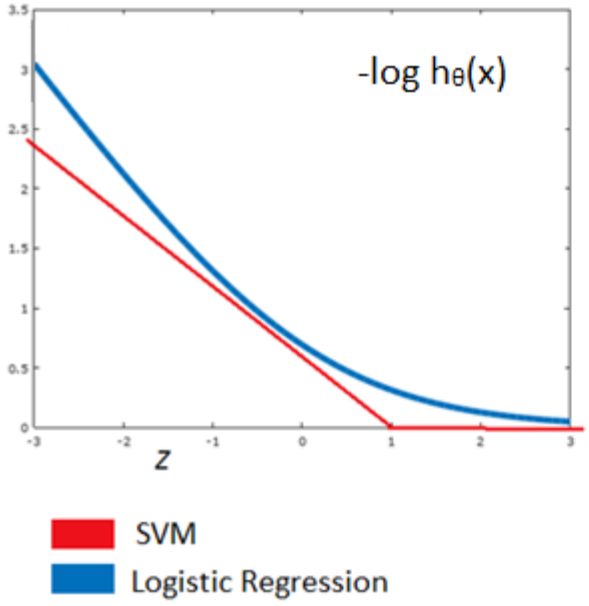
Cost of an example (x, y) is given by,

$$\boxed{\text{Cost} = -\left(y \log h_\theta(x) + (1 - y) \log(1 - h_\theta(x))\right)}$$

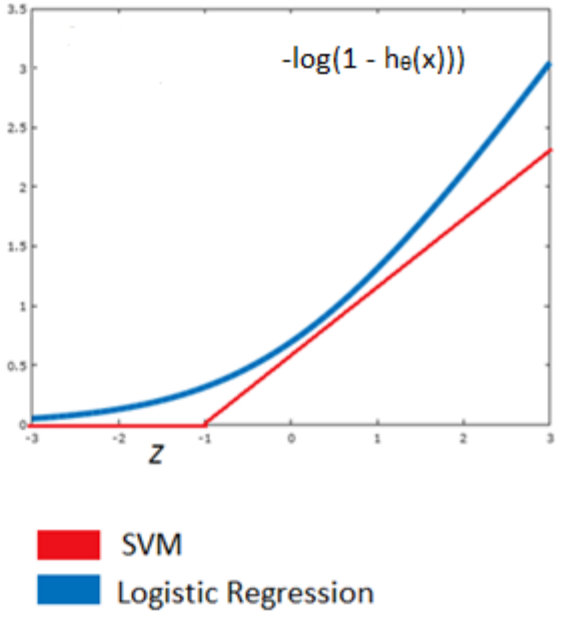
This can also be written as,

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

If y=1, we want $\theta^T x >> 0$




If y=0, we want $\theta^T x << 0$



Cost function of logistic regression with regularization is given by,

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

On modifying this equation, we obtain the cost function of SVM as follows:

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

This is of the form **A + λB**

On multiplying the equation by **C = 1/λ,** we get,

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

Thus, the cost function of Support Vector Machine is of the form **CA + B** where **C = 1/λ**

**Summary:**

Finally, cost function of SVM is given by,

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

**Hypothesis:**

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

Consider the following minimization problems:

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

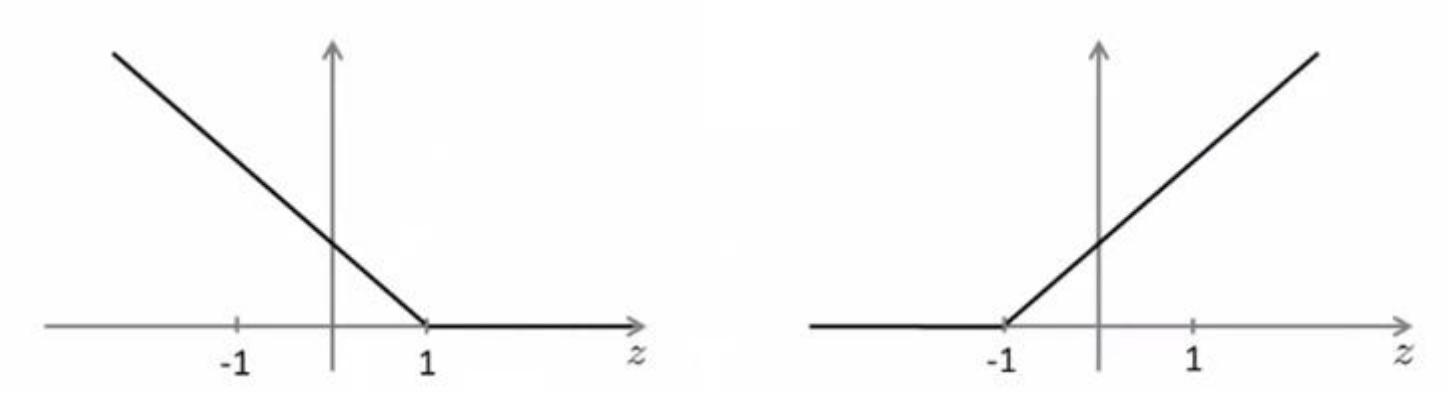
2. $$\min_{\theta} C\left[\sum_{i=1}^{m} y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

These two optimization problems will give the same value of θ (i.e. the same value of θ gives the optimal solution to both problems) if:

- C = λ
- C = -λ
- C = 1/λ
- C = 2/λ

# Support Vector Machines: Large Margin Intuition

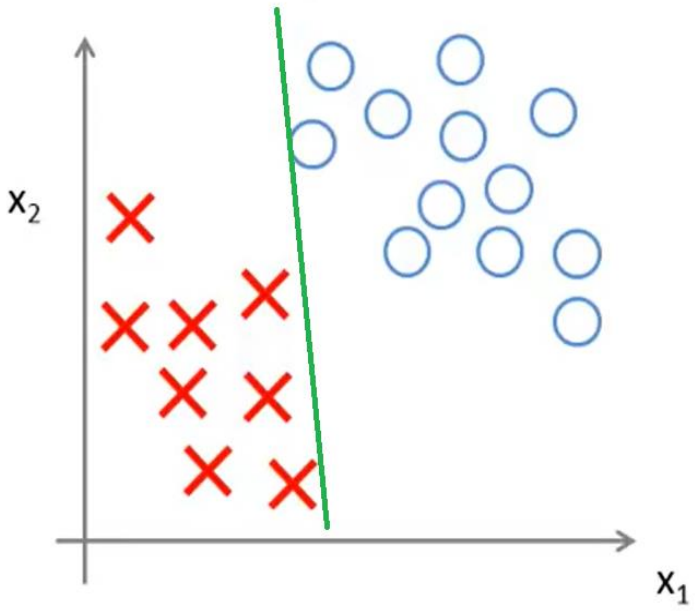
**How is SVM different from logistic regression?**



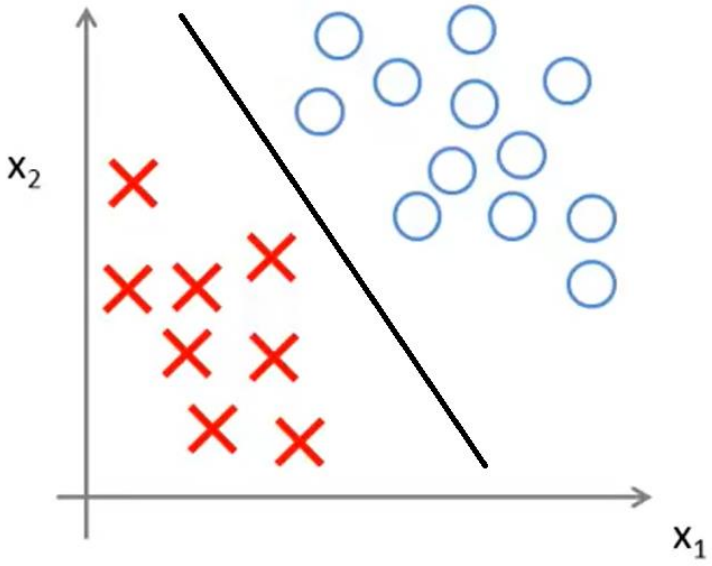

- If y=1, we want $\theta^T x \geq 1$ and not just $\geq 0$ as in logistic regression
- If y=0, we want $\theta^T x \leq 1$ and not just $< 0$ as in logistic regression

This is an interesting property of SVM as it builds an extra "safety margin" factor.
In other words, the positive examples and the negative examples are separated by the decision boundary by a larger margin.

## SVM Decision Boundary: Linearly separable case



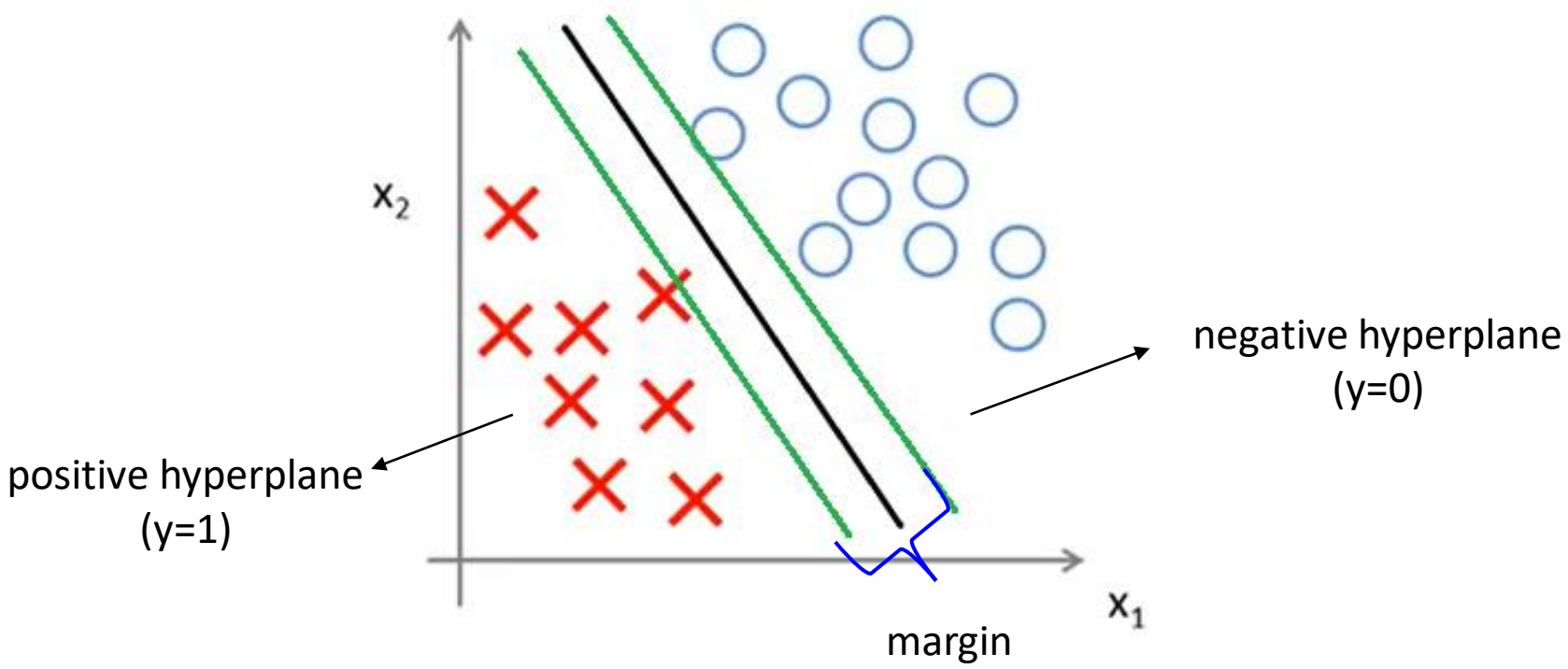

Logistic Regression
"Small margin classifier"




SVM
"Large margin classifier"

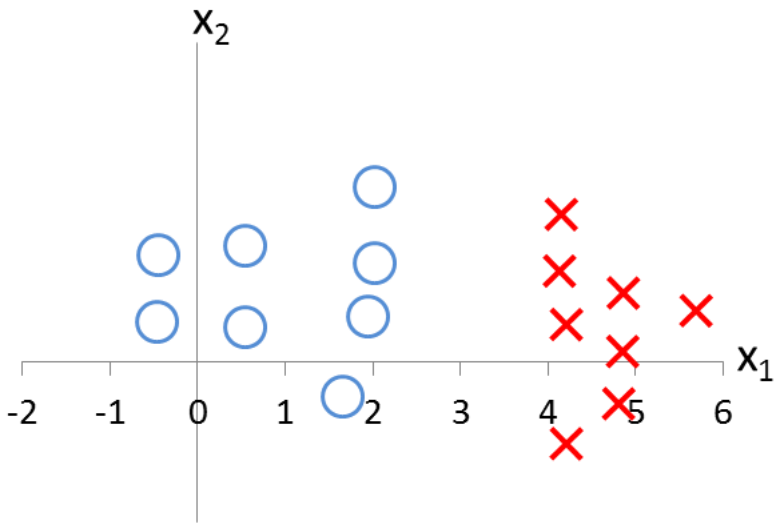
The positive and negative examples are very closely separated by the decision boundary in logistic regression whereas in SVM, they are separated by a comparatively larger margin.

## Understanding linearly separable case:




- The black line is called decision boundary.
- The green lines are called support vectors.
- Distance between the support vectors is called margin.

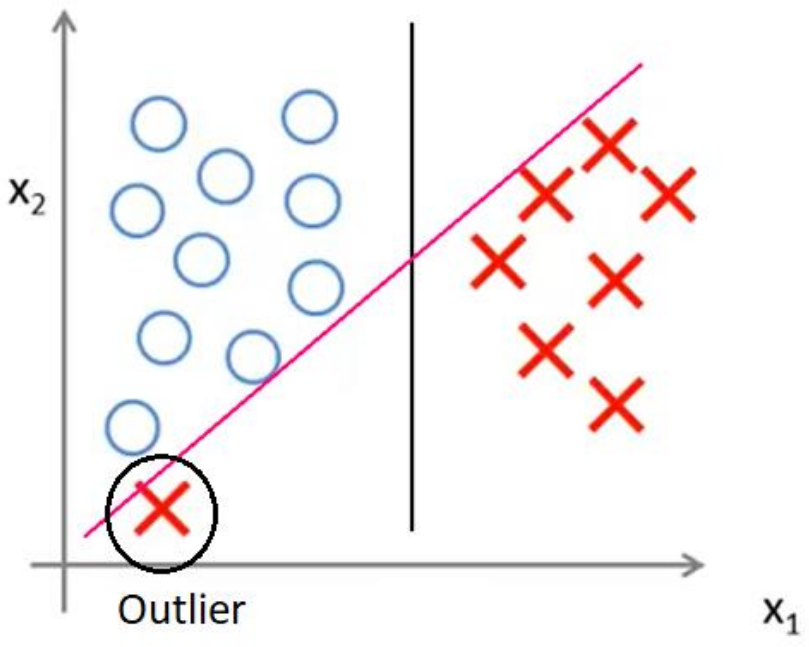
Consider the training set to the right, where "x" denotes positive examples (y=1) and "o" denotes negative examples (y=0). Suppose you train an SVM (which will predict 1 when $\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0$). What values might the SVM give for $\theta_0$, $\theta_1$ and $\theta_2$?




- $\theta_0 = 3, \theta_1 = 1, \theta_2 = 0$
- $\theta_0 = -3, \theta_1 = 1, \theta_2 = 0$
- $\theta_0 = 3, \theta_1 = 0, \theta_2 = 1$
- $\theta_0 = -3, \theta_1 = 0, \theta_2 = 1$

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

We know that the cost function of SVM is represented by CA + B.



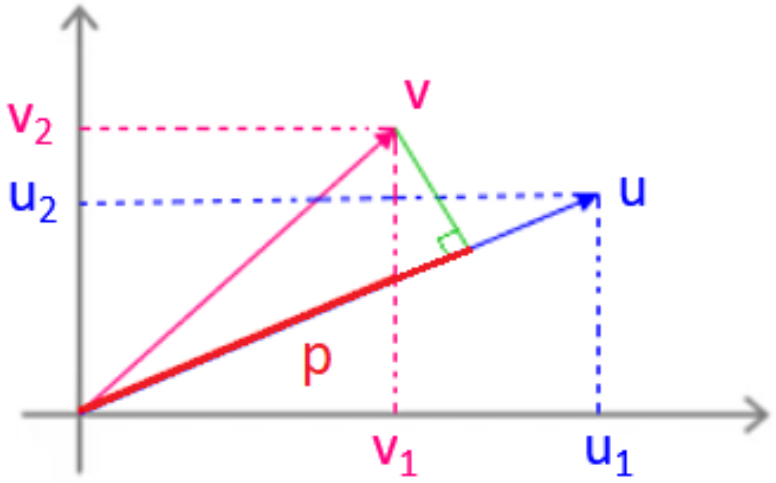

If the regularization parameter C is very large, then the SVM will change the **black** decision boundary to the **pink** decision boundary to try to fit the outlier at the cost of margin between the positive and negative examples which is not good.

# Mathematics behind large margin classifier:

**Vector Inner Product:**

Consider two vectors u and v

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




**How do we represent $u^Tv$?**

Vector v is orthogonally projected on vector u, and $u^Tv = p \,.\, \|u\|$ where,
p = length of projection of v onto u.

- $u^Tv = p \,.\, \|u\|$ where, $\|u\| = \sqrt{u_1^2 + u_2^2}$ is the length of vector u
- $u^Tv = u_1v_1 + u_2v_2$

**Note:**
p is signed (+/-). If p > 0, then both p and u point in same direction and if p < 0, they point in opposite direction.

## SVM Decision Boundary:

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

$$\text{s.t.} \quad \left.\begin{array}{ll} \theta^T x^{(i)} \geq 1 & \text{if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 & \text{if } y^{(i)} = 0 \end{array}\right\} \text{when C is very large}$$

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

To understand how the SVM chooses the decision boundary in order to meet the above given conditions, let us take two examples.

Let's try to understand this by two cases:

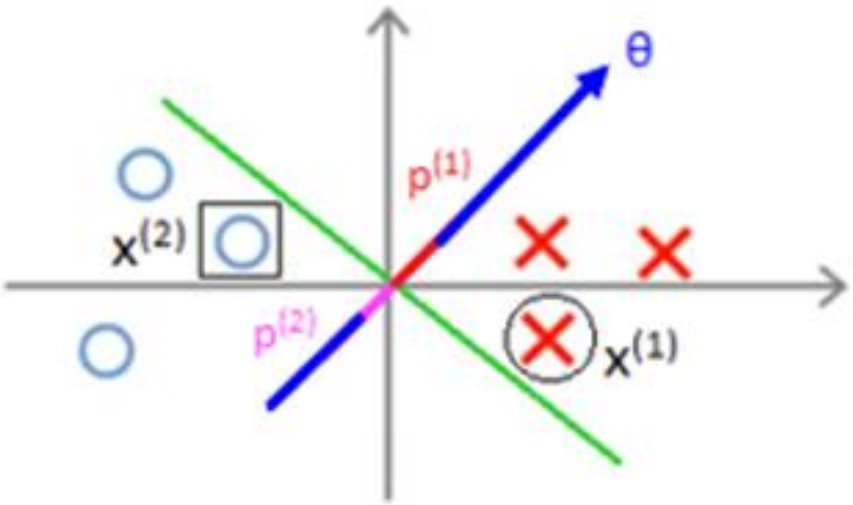
For simplification purpose, let $\theta_0 = 0$ (decision boundary passes through origin)

**Conditions of SVM for large margin are:**

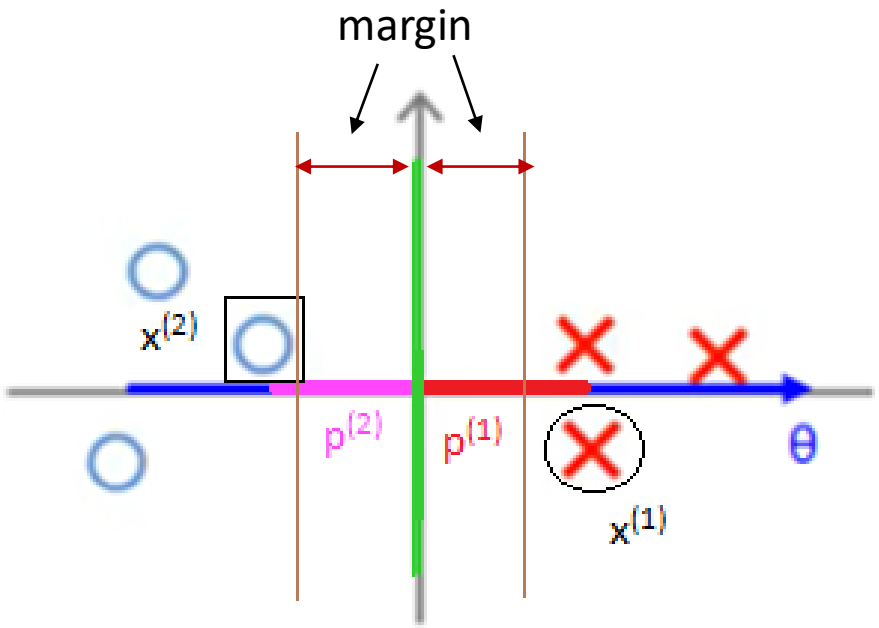
- $p^{(i)} . \|\theta\| \geq 1$ if $y^{(i)} = 1$
- $p^{(i)} . \|\theta\| \leq -1$ if $y^{(i)} = 0$

**Case 1:**

- The **decision boundary** narrowly separates the positive and negative examples.
- **Theta vector** is orthogonal to the decision boundary.
- The projections $\mathbf{p^{(1)}}$ and $\mathbf{p^{(2)}}$ are very small.
  In order to meet the conditions given, $\|\theta\|$ has to be large so that $p^{(1)} . \|\theta\| \geq 1$ and $p^{(2)} . \|\theta\| \leq -1$
- As a result, the SVM prefers not to choose this hypothesis where the decision boundary narrowly separates the positive and negative examples.



**Case 2:**



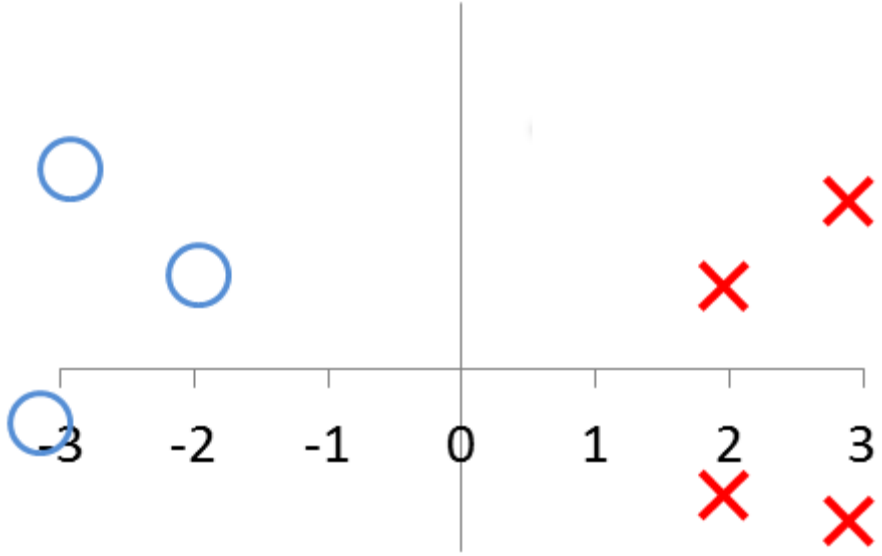

- The **decision boundary** separates the positive and negative examples by a larger margin.
- **Theta vector** is orthogonal to the decision boundary.
- The projections $p^{(1)}$ and $p^{(2)}$ are larger, thus $\|\theta\|$ can be small for the conditions to be met.
- Hence, SVM prefers to choose this hypothesis over that in Case 1 where the positive and negative examples are separated by a larger margin.
- As a result, the concept of "large margin classifier" was introduced.

The SVM optimization problem we used is:

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

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

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

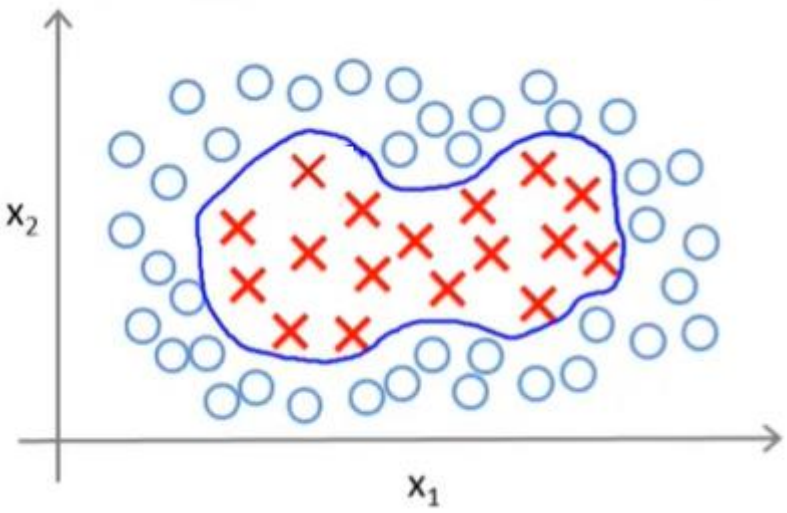



where $p^{(i)}$ is the (signed - positive or negative) projection of $x^{(i)}$ onto θ. Consider the training set above. At the optimal value of θ, what is ‖θ‖?

- 1/4
- 1/2
- 1
- 2

# Kernels I

Consider a non-linear decision boundary as shown:




Let's say we come up with a hypothesis as follows (for example) that distinguishes the positive and negative examples.

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \ldots$$

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

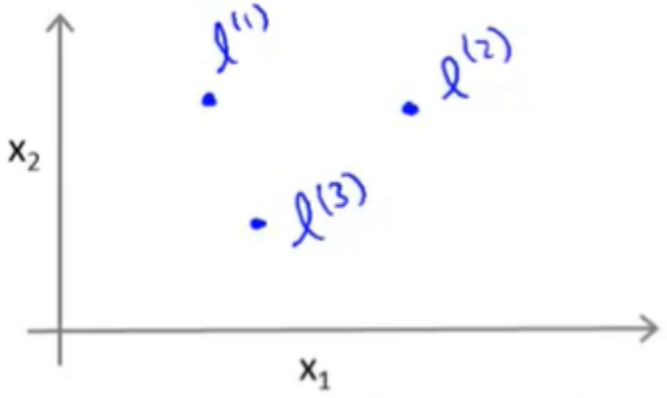
This can also be written as:

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \theta_5 f_5 + \ldots$ where, $f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, $f_4 = x_1^2$, $f_5 = x_2^2$

**The question is, is there a better choice of the features f1, f2, f3, … ?**

**Kernel:**

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




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

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

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

The similarity( ) function (LHS) is called kernel function while the exp( ) function (RHS) is called Gaussian Kernel.

**Kernels and Similarity:**

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

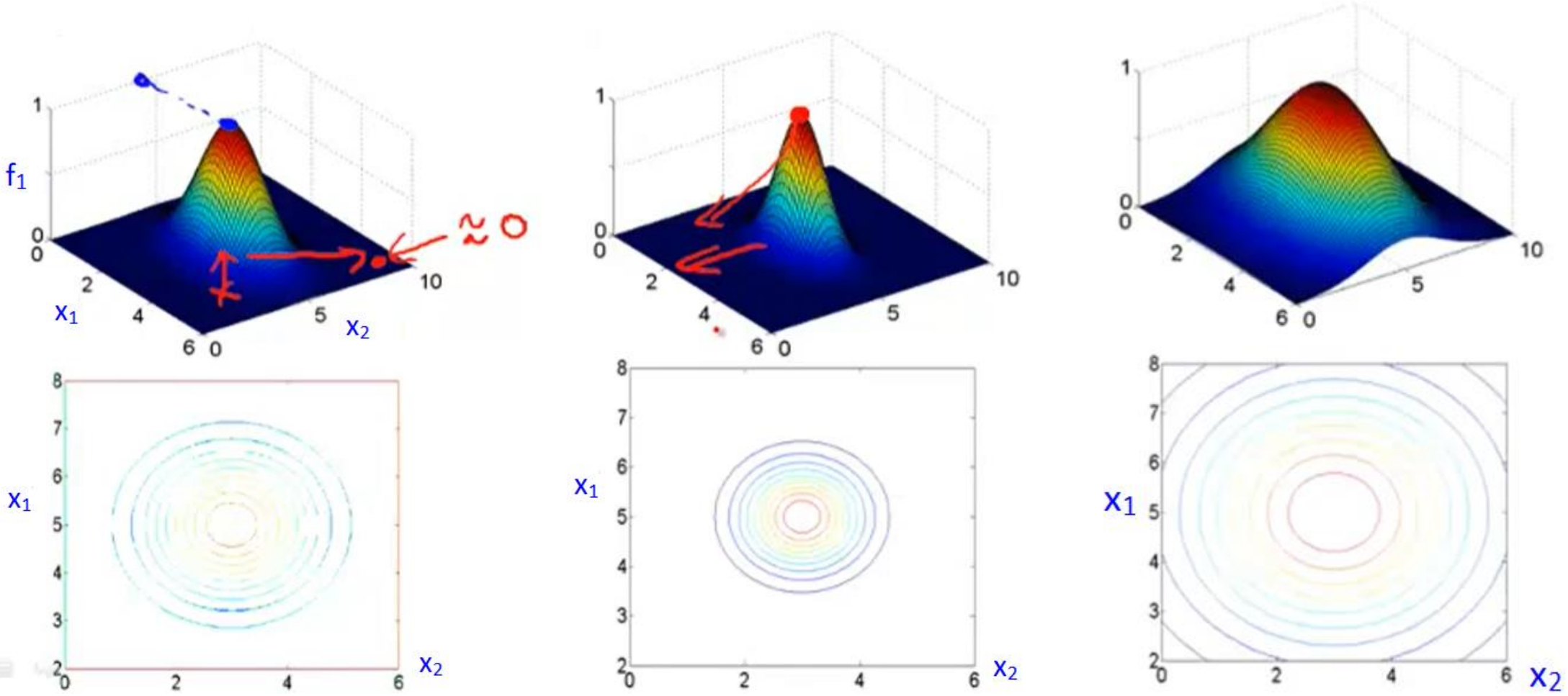
- If $x \approx l^{(1)}$ i.e. x is close to landmark $l^{(1)}$, then $f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$
- If x is far from $l^{(1)}$, then $f_1 \approx \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0$

Similarly, we can compute features $f_1$, $f_2$, $f_3$ for the corresponding landmarks $l^{(1)}$, $l^{(2)}$, $l^{(3)}$

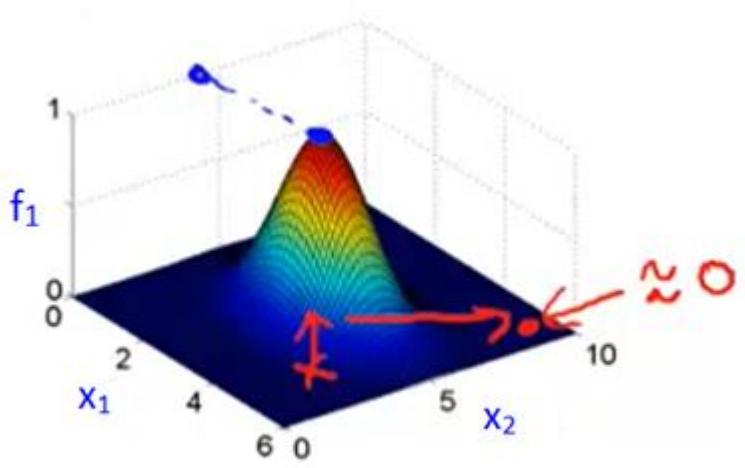
**Example:**

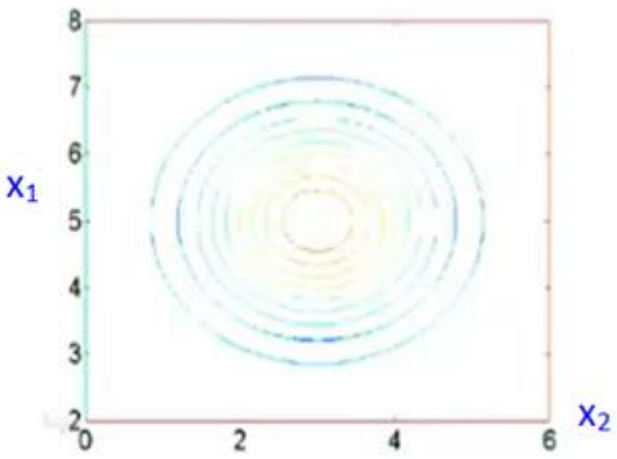
$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \; f_1 = \exp\left(-\frac{\| x - l^{(1)} \|^2}{2\sigma^2}\right)$$

**$\sigma^2 = 1$** **$\sigma^2 = 0.5$** **$\sigma^2 = 3$**



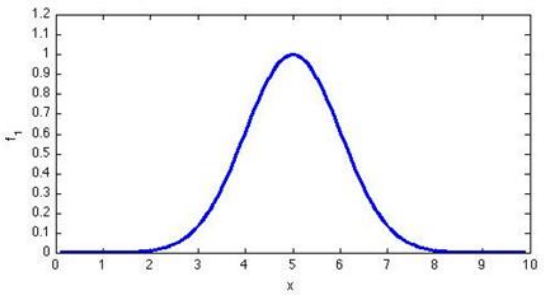
**Points to note:**



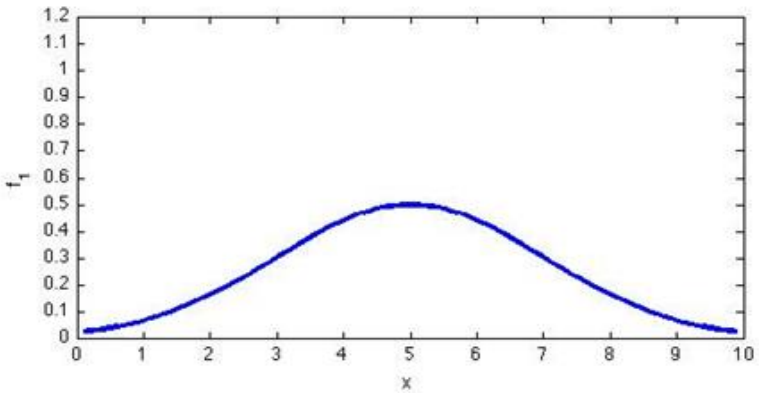



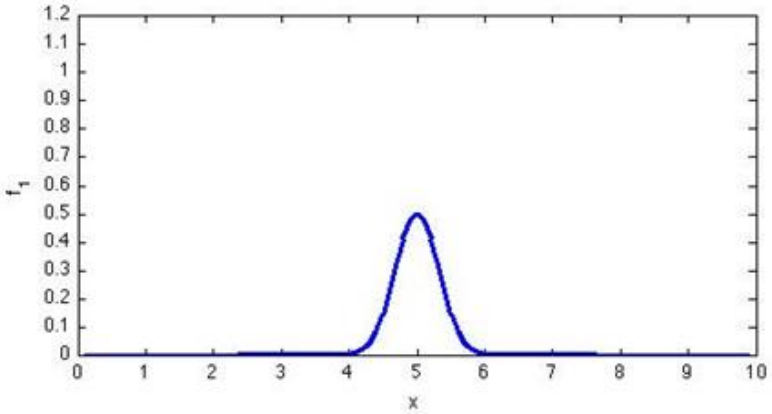

- If $x = l^{(1)}$, then $f_1$ is at the top of the peek i.e. $f_1 = 1$
- As x moves towards $l^{(1)}$, the value of $f_1$ increases and eventually tends to 1.
- As x moves away from $l^{(1)}$, the value of $f_1$ decreases and eventually tends to 0.
- If the value of $\sigma^2$ is decreased, the bump becomes narrower and the contour shrinks.
- If the value of $\sigma^2$ is increased, the bump becomes broader and the contour expands.

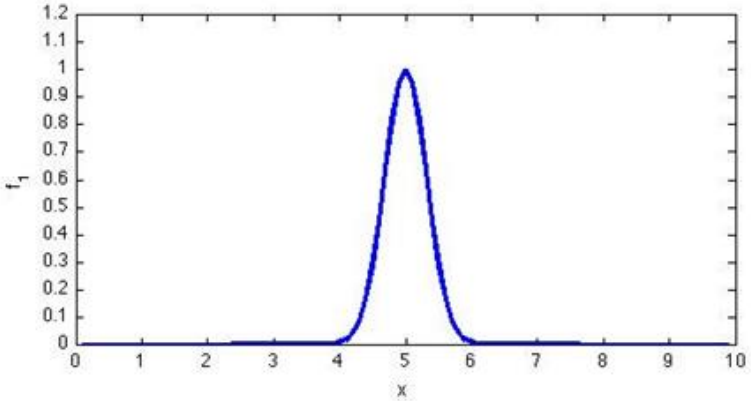
Consider a 1-D example with one feature $x_1$. Suppose $l^{(1)}=5$. To the right is a plot of $f_1 = \exp(-\frac{||x_1 - l^{(1)}||^2}{2\sigma^2})$ when $\sigma^2 = 1$. Suppose now we change $\sigma^2 = 4$.

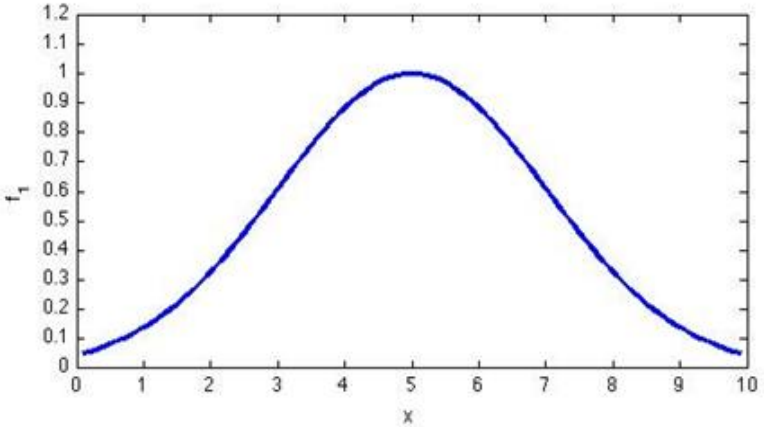



Which of the following is a plot of $f_1$ with the new value of $\sigma^2$ ?



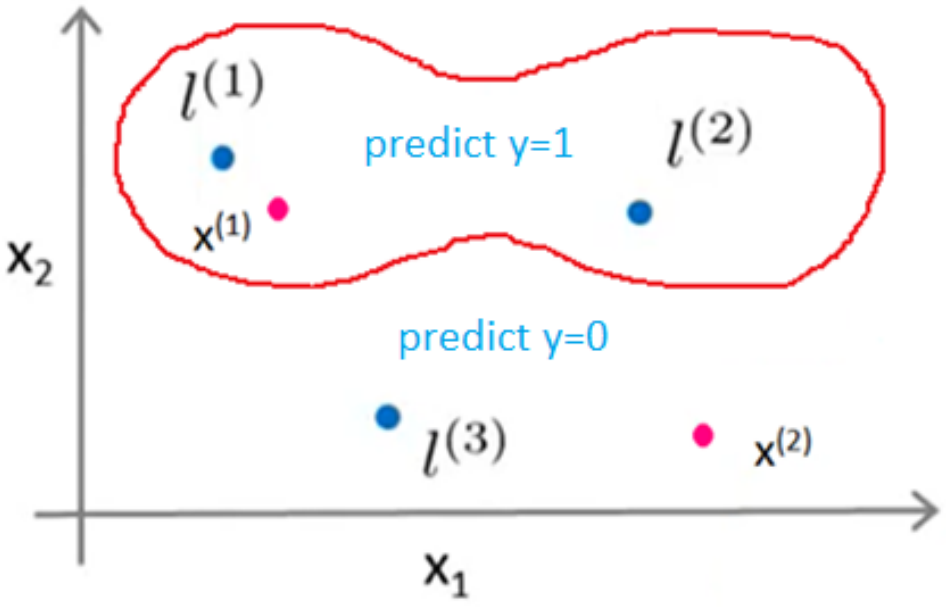



**Example:**

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

We predict y=1 if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$ and 0 otherwise.

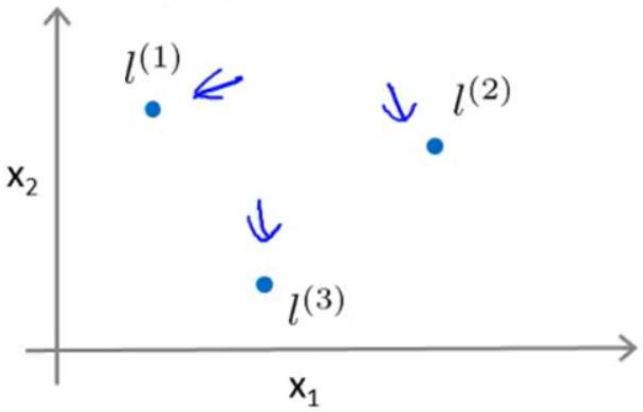



**Solution:**

- $x^{(1)}$ is closer to landmark $l^{(1)}$. Hence, $f_1 \approx 1$, $f_2 \approx 0$, $f_3 \approx 0$.
- This means, $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 = (-0.5) + (1)(1) + (1)(0) + (0)(0) = 0.5 \geq 0 \Rightarrow$ predict y=1
- $x^{(2)}$ is far from proximity to any of the landmarks. Hence, $f_1 \approx 0$, $f_2 \approx 0$, $f_3 \approx 0$.
- This means, $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 = (-0.5) + (1)(0) + (1)(0) + (0)(0) = -0.5 < 0 \Rightarrow$ predict y=0
- We keep predicting the value of y for different proximities of $x^{(i)}$ corresponding to $l^{(i)}$ until we are able to find a decision boundary of the hypothesis that separates the positive and negative examples.

# Kernels II

**Choosing the landmarks:**




For a given example x, we find the feature $f_i$ as:

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

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

**The question is, how do we choose the landmarks?**


We take the training examples as the landmarks themselves.

**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:**

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

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

... and so on

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

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

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

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

...

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

...

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

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

**SVM with Kernels:**

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

Predict "y=1" if $\theta^T f \geq 0$ i.e. $\theta_0 f_0 + \theta_1 f_1 + \theta_2 f_2 + \ldots + \theta_m f_m \geq 0$

**Training:**

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

This can be written as $\theta^T M \theta$

where

- $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \ldots \\ \theta_m \end{bmatrix}$
- M is some matrix

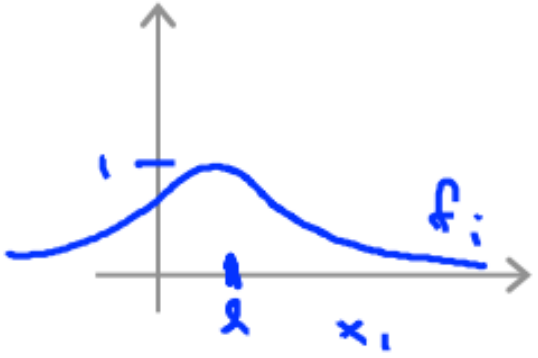
**SVM Parameters:**

**$C = 1/\lambda$**

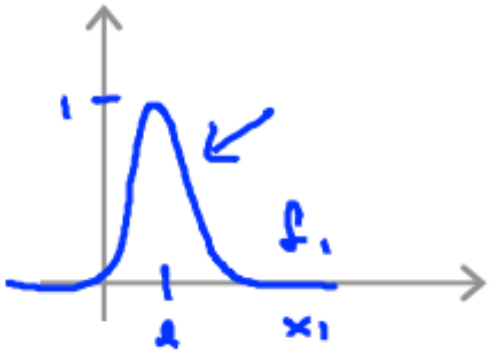
- Large C: Lower bias, high variance
- Small C: Higher bias, low variance

**$\sigma^2$**

- Large $\sigma^2$ : Features $f_i$ vary more smoothly
  Higher bias, lower variance




- Small $\sigma^2$ : Features $f_i$ vary less smoothly
  Lower bias, higher variance



Suppose you train an SVM and find it overfits your training data. Which of these would be a reasonable next step? Check all that apply.

- Increase C
- Decrease C
- Increase $\sigma^2$
- Decrease $\sigma^2$

# Using an SVM

To implement SVM, we may use SVM software package (e.g. liblinear, libsvm, …) to solve for parameters θ.

For this, we need to specify:

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

**Example:**

- No kernel ("linear kernel") : Predict $y = 1$ if $\theta^T x \geq 0$
  This is used when
- Gaussian kernel:
  $f_i = \exp\left(-\frac{\| x - l^{(i)} \|^2}{2\sigma^2}\right)$ , where $l^{(i)} = x^{(i)}$

  Here we need to choose $\sigma^2$
  This is used when no. of features (n) is small and no. of training examples (m) is large.

If you decide to use Gaussian kernel, here's what you need to do:

**Kernel (similarity) functions:**

function f = kernel(x1, x2)

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

return

For vectorized implementation:

$f = f_i$

$x1 = x^{(i)}$

$x2 = l^{(i)} = x^{(j)}$

**Note:**

Do perform feature scaling before using Gaussian kernel, otherwise the error would be large.

**Example:**

Consider an example of housing price prediction.

$||x - l||^2$ can be written as $||v||^2$ where $v = x - l$

Thus, $||v||^2 = v_1^2 + v_2^2 + \ldots + v_n^2$

$$= \underbrace{(x_1 - l_1)^2}_{\text{sq. feet (in 1000s)}} + \underbrace{(x_2 - l_2)^2}_{\text{\# of bedrooms (1-5)}} + \ldots + (x_n - l_n)^2$$

As shown in this example,
**range of sq. feet >> range of # of bedrooms**
Hence, # of bedrooms may get ignored due to its very small value if feature scaling is not done.
This will result in high error.

**Other choices of kernel:**

- Not all similarity functions similarity(x, l) make valid kernels.
- They need to satisfy a technical condition called "Mercer's Theorem" to make sure SVM packages' optimizations run correctly, and not diverge.

- For this, there are many off-the-shelf kernels available:

  - Polynomial kernel: It is of the form $(X^T l + \text{constant})^{\text{degree}}$

  **Example:** $(X^T l + 5)^4$, $(X^T l + 1)^3$, etc.

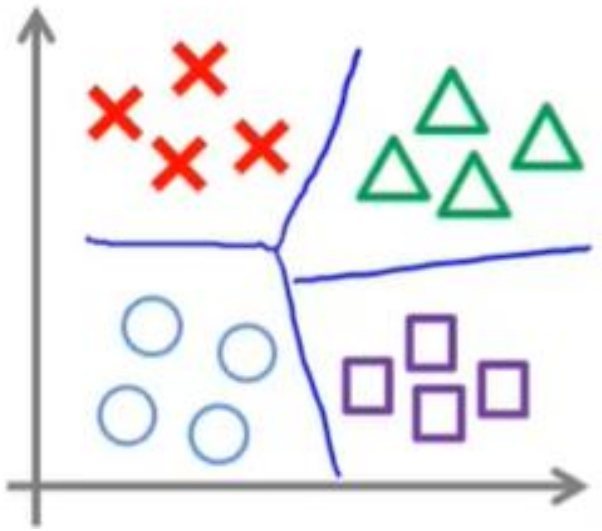
  - More esoteric: String kernel, chi-square kernel, histogram intersection kernel, etc.

**Some References:**
https://en.wikipedia.org/wiki/String_kernel
https://pdfs.semanticscholar.org/bcff/9506398bc3d069288d23e3d044318916c89e.pdf
http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#chisquare

Suppose you are trying to decide among a few different choices of kernel and are also choosing parameters such as C, $\sigma^2$ , etc. How should you make the choice?

- Choose whatever performs best on the training data.
- Choose whatever performs best on the cross-validation data.
- Choose whatever performs best on the test data.
- Choose whatever gives the largest SVM margin.

**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 $(\theta^{(i)})^T x$

## Logistic Regression vs. SVMs

### When to use logistic regression and SVM?

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

- If n is large and m is small [n= 10,000 and m = 10-10,000]
  then use logistic regression, or SVM without a kernel ("linear kernel")

- If n is small and m is intermediate [n = 1-1000 and m = 10-10,000]
  then use SVM with Gaussian kernel

- If n is small and m is large [n = 1-1000, m ≥ 50,000]
  then use logistic regression, or SVM without a kernel

- Neural Network is likely to work well for most of these conditions, but may be slower to train.