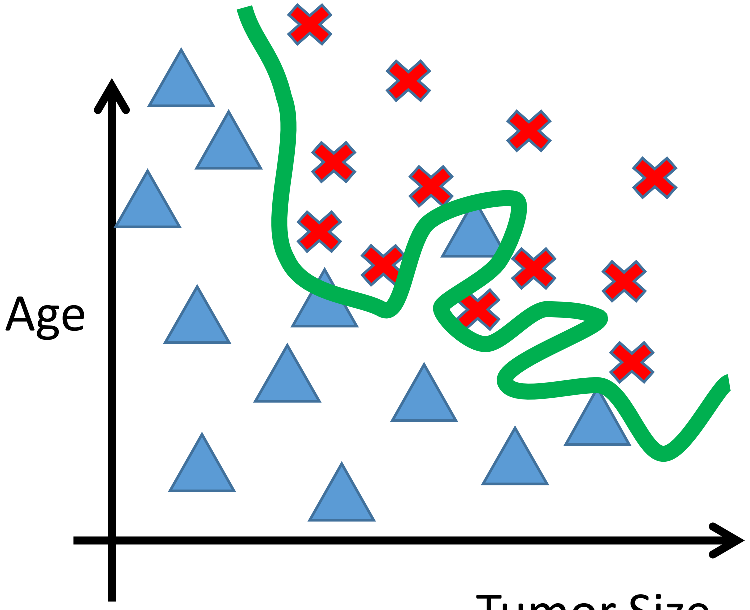
Support Vector Machines

Regularized logistic regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \theta_6 x_1^3 x_2 + \theta_7 x_1 x_2^3 + \cdots)$$

- Cost function:

$$J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right) + \frac{\lambda}{2}\sum_{j=1}^{n}\theta_j^2\right]$$

Gradient descent (Regularized)

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

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

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \lambda\theta_j\right]$$

}

$$\frac{\partial}{\partial\theta_j}J(\theta)$$

Terminology

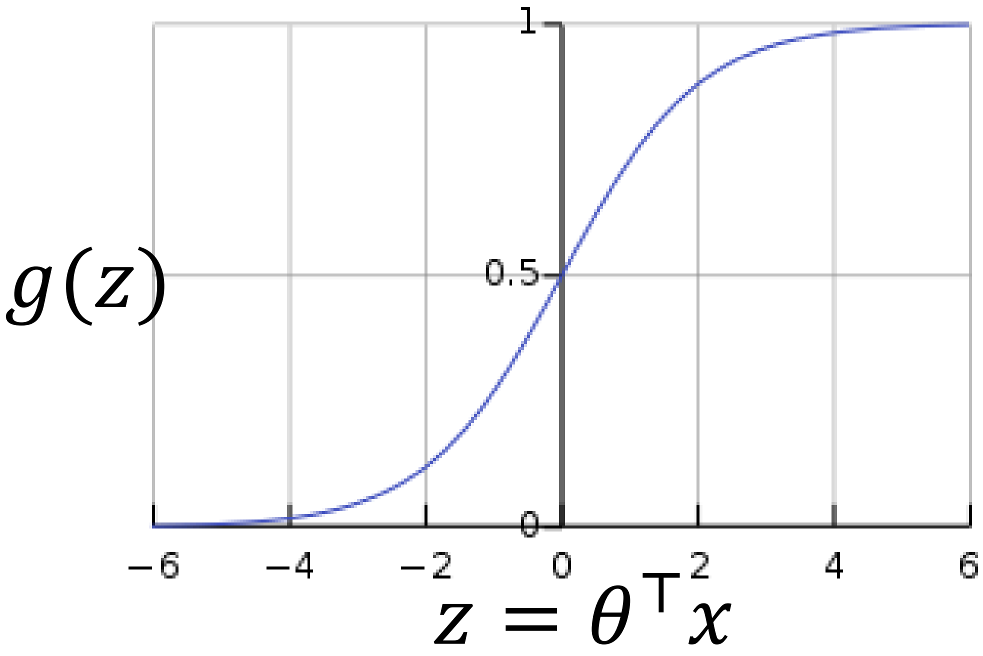
Regularization function	Name	Solver
$\|\theta\|_2^2 = \sum_{j=1}^{n} \theta_j^2$	Tikhonov regularization Ridge regression	Close form
$\left\|\|\theta\|\right\|_1 = \sum_{j=1}^{n} \|\theta_j\|$	LASSO regression	Proximal gradient descent, least angle regression
$\alpha\left\|\|\theta\|\right\|_1 + (1-\alpha)\ \|\theta\|_2^2$	Elastic net regularization	Proximal gradient descent

Support Vector Machine

- **Cost function**
- Large margin classification
- Kernels
- Using an SVM

Logistic regression

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



Suppose predict "y = 1" if $h_\theta(x) \geq 0.5$

$$z = \theta^\top x \geq 0$$

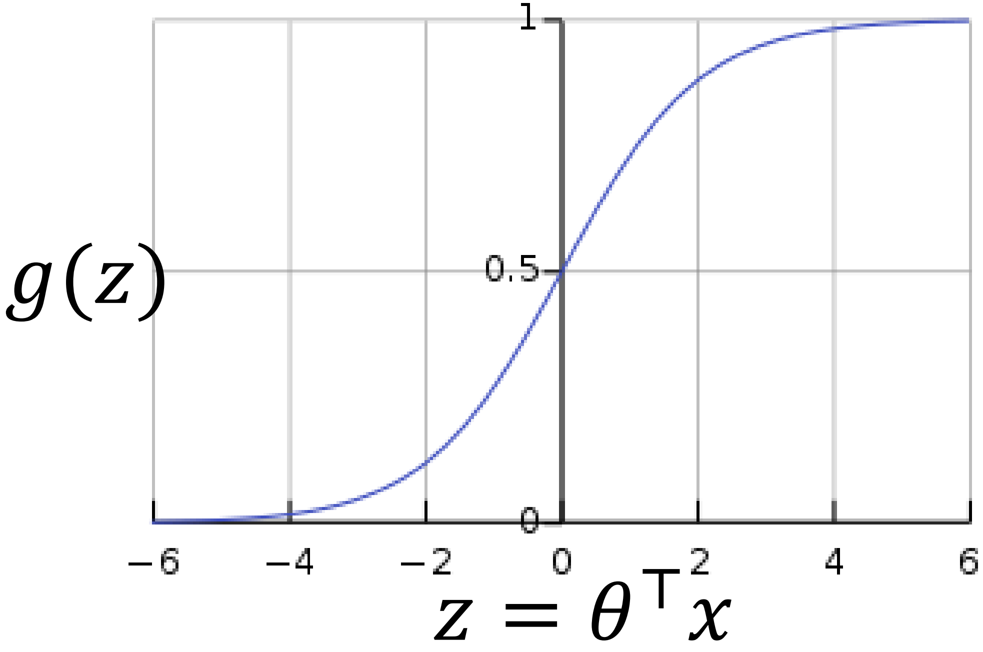
predict "y = 0" if $h_\theta(x) < 0.5$

$$z = \theta^\top x < 0$$

Alternative view

$$h_\theta(x) = g(\theta^\top x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



If "y = 1", we want $h_\theta(x) \approx 1$

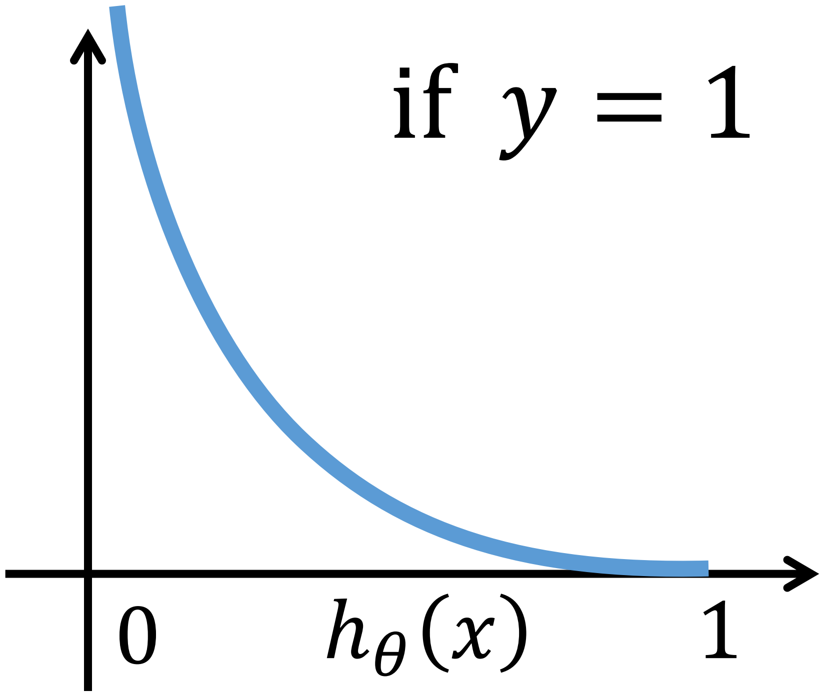
$$z = \theta^\top x \gg 0$$

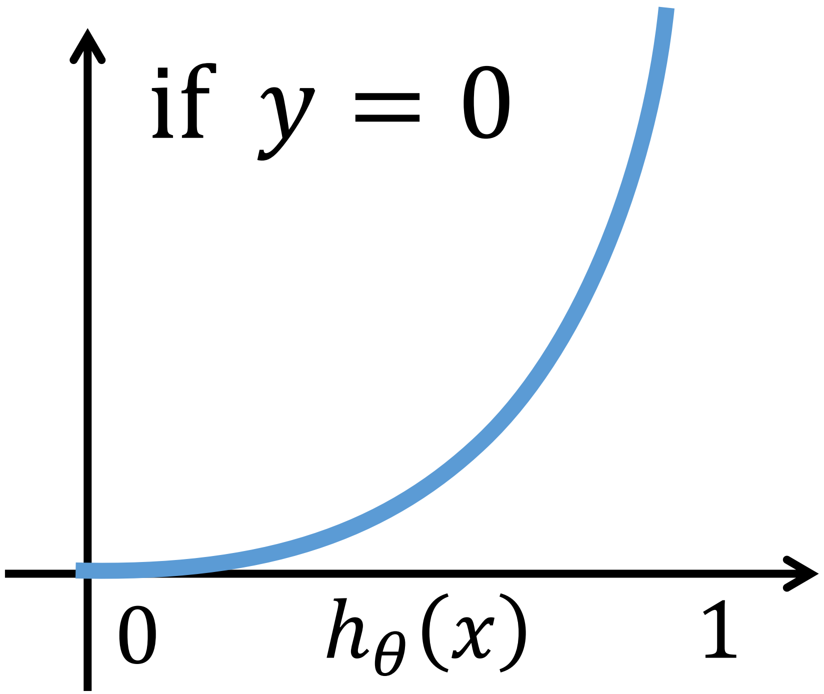
If "y = 0", we want $h_\theta(x) \approx 0$

$$z = \theta^\top x \ll 0$$

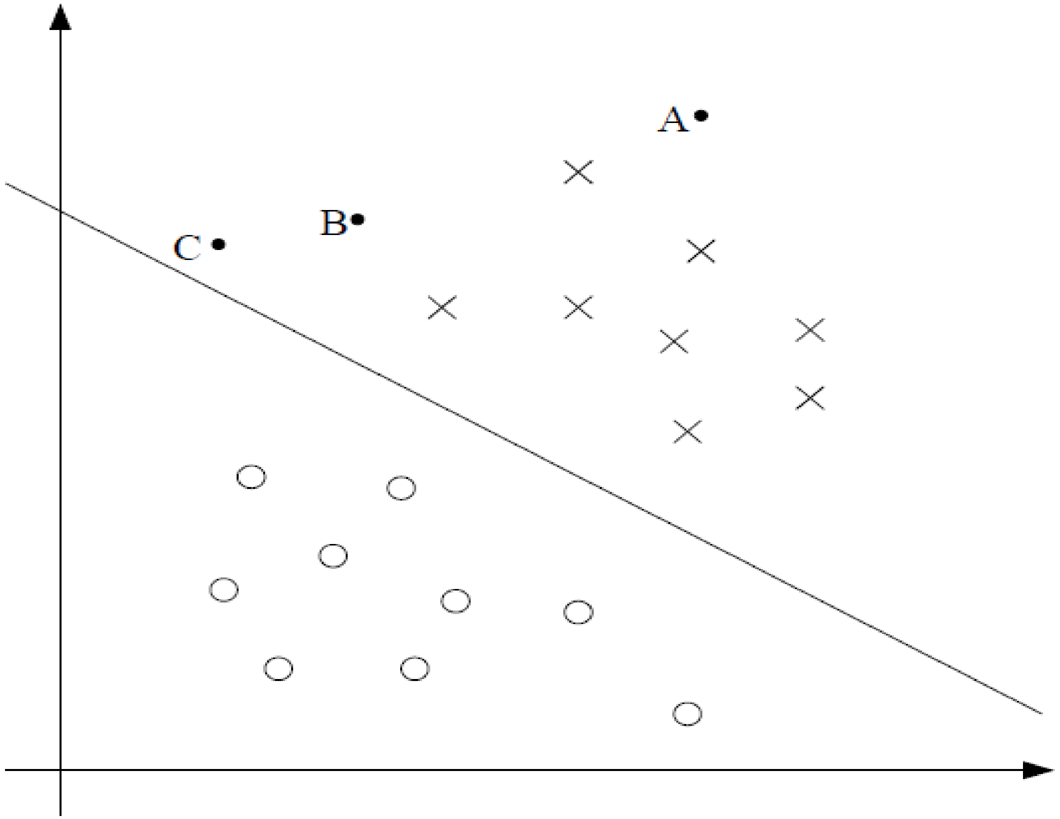
Cost function for **Logistic Regression**

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

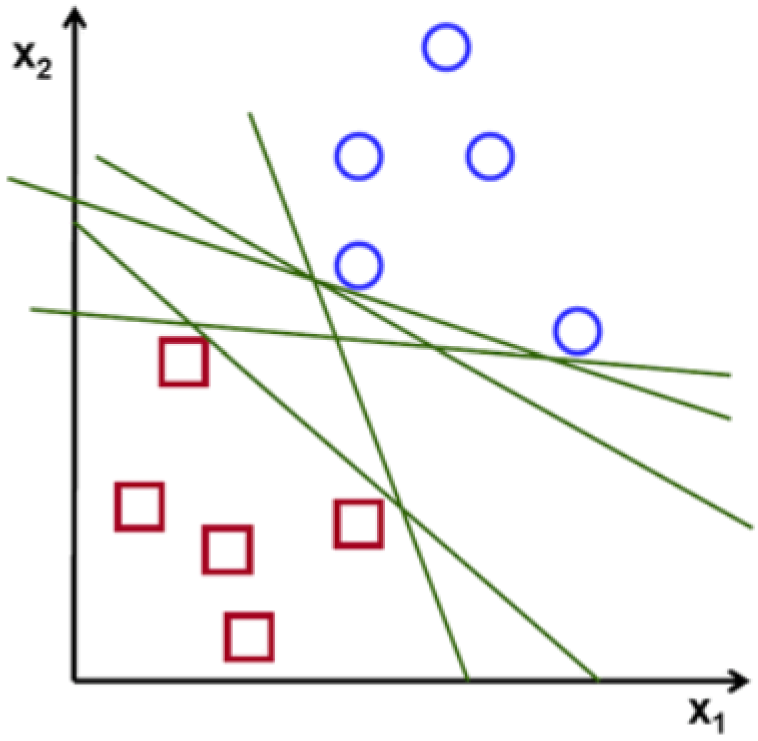


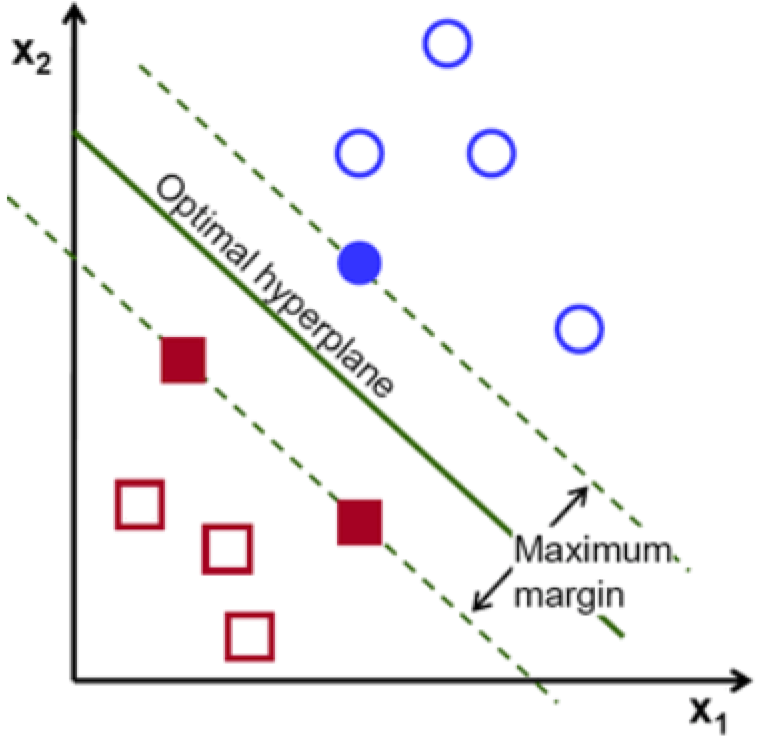


Margins: Intuition



Consider the following figure, in which x's represent positive training examples, o's denote negative training examples, a decision boundary (this is the line given by the equation $\theta^\top x = 0$, and is also called the separating hyperplane) is also shown, and three points have also been labelled A, B and C.


x2
x1


x2
Optimal hyperplane
Maximum
margin
x1

- Notice that the point A is very far from the decision boundary. If we are asked to make a prediction for the value of y at A, it seems we should be quite confident that y = 1 there.

- Conversely, the point C is very close to the decision boundary, and while it’s on the side of the decision boundary on which we would predict y = 1, it seems likely that just a small change to the decision boundary could easily have caused out prediction to be y = 0.

- Hence, we’re much more confident about our prediction at A than at C.

- The point B lies in-between these two cases, and more broadly, we see that if a point is far from the separating hyperplane, then we may be significantly more confident in our predictions.

Notation

- For SVM we will be considering a linear classifier for a binary classification problem with labels y and features x.
- We'll use $y \in \{-1, 1\}$ (instead of $\{0, 1\}$) to denote the class labels.
- Also, rather than parameterizing our linear classifier with the vector θ, we will use parameters w, b, and write our classifier as

$$h_{w,b}(x) = g(w^T x + b).$$

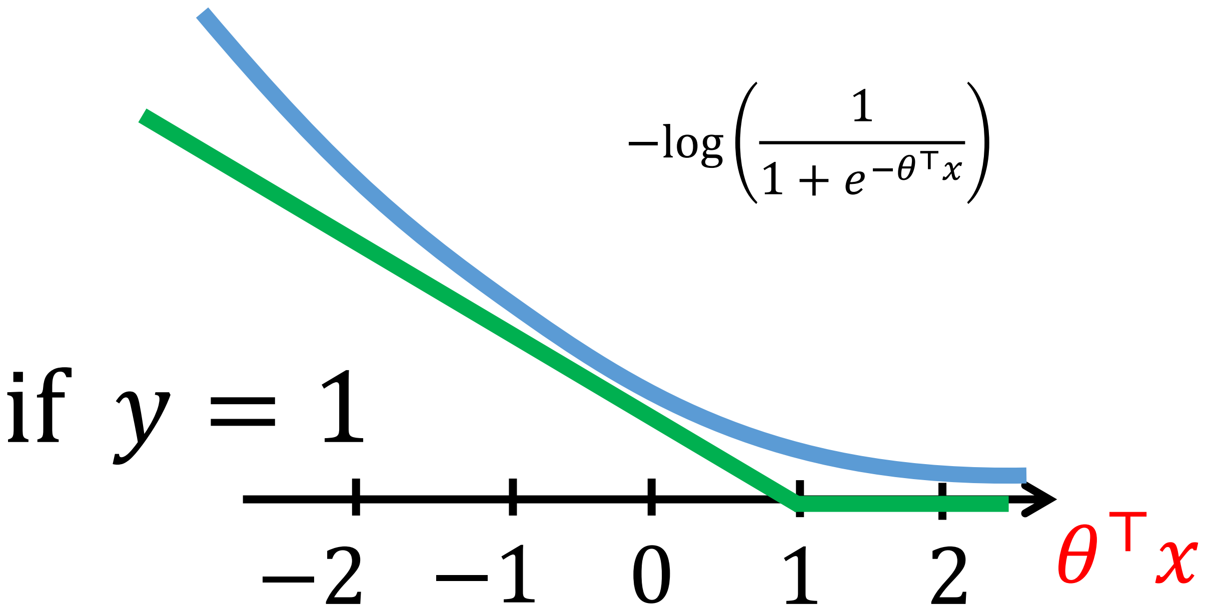
Here, $g(z) = 1$ if $z \geq 0$,
and $g(z) = -1$ otherwise.

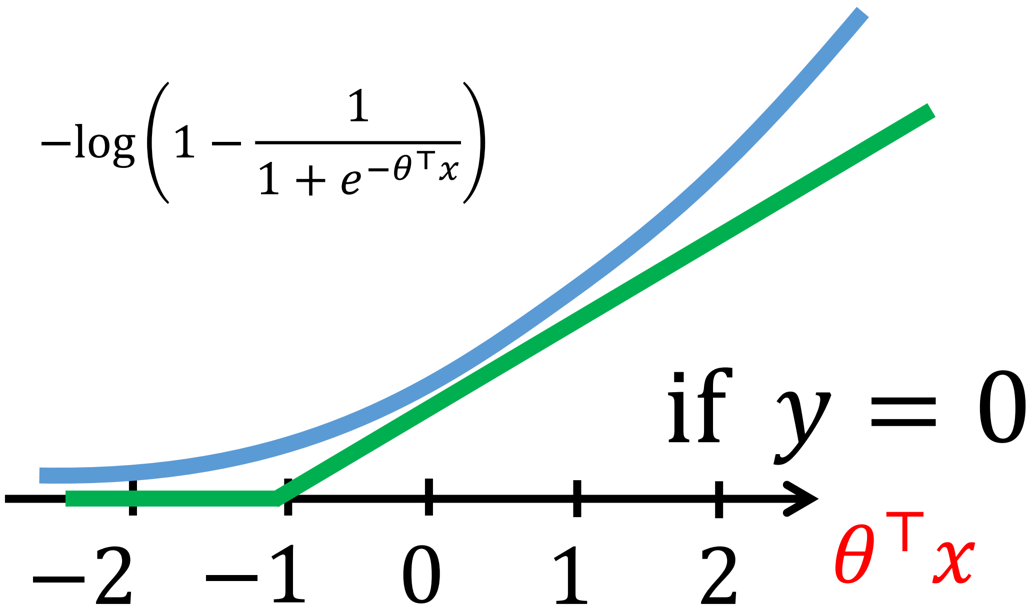
- This "w, b" notation allows us to explicitly treat the intercept term b separately from the other
- parameters. (We also drop the convention we had previously of letting $x_0 = 1$ be an extra coordinate in the input feature vector.)
- Thus, b takes the role of what was previously θ_0, and w takes the role of $[\theta_1 \ldots \theta_n]^T$
- From our definition of g above, our classifier will directly predict either 1 or -1.

Alternative view of logistic regression

- $\mathrm{Cost}(h_\theta(x), y) = -\, y\ \log\big(\mathrm{h}_\theta(\mathrm{x})\big) - (1-\mathrm{y})\log\big(1 - h_\theta(x)\big)$

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



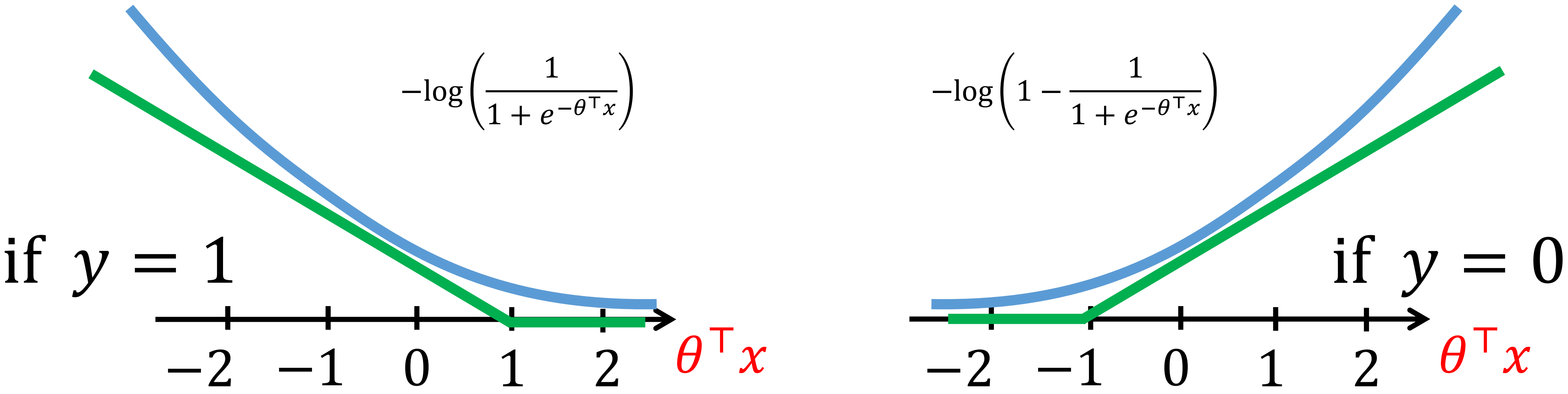


Logistic regression (logistic loss)

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

Support vector machine (hinge loss)

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



Optimization objective for SVM

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

1) Remove $\frac{1}{m}$

2) Multiply $\mathrm{C} = \frac{1}{\lambda}$

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

Hypothesis of SVM

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

- Hypothesis

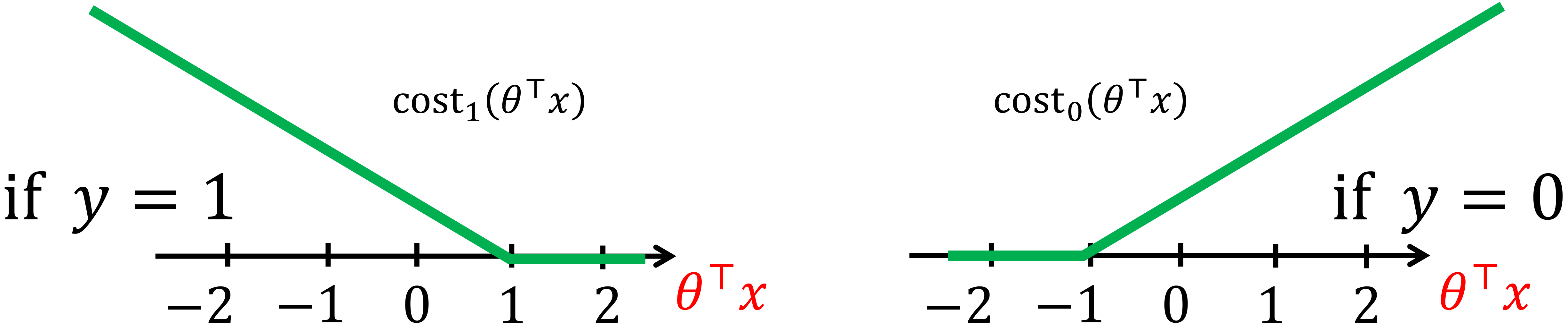
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^\top x \geq 0 \\ 0 & \text{if } \theta^\top x < 0 \end{cases}$$

Support Vector Machine

- Cost function
- **Large margin classification**
- Kernels
- Using an SVM

Support vector machine

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



If "y = 1", we want $\theta^\top x \geq \ \ 1$ (not just ≥ 0)

If "y = 0", we want $\theta^\top x \leq -1$ (not just < 0)

SVM decision boundary

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

- Let’s say we have a very large C …

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

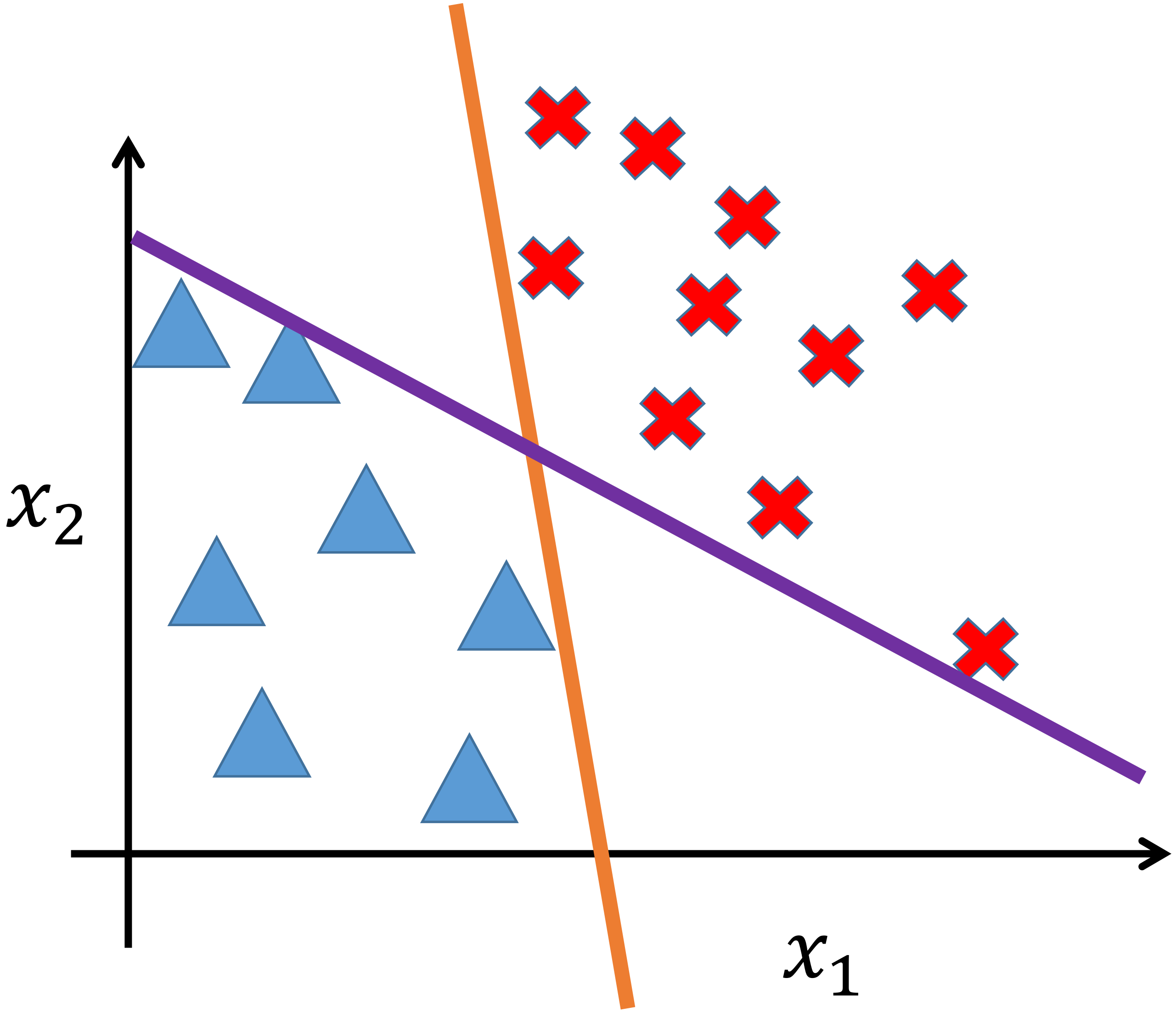
  $\theta^\top x^{(i)} \geq 1$

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

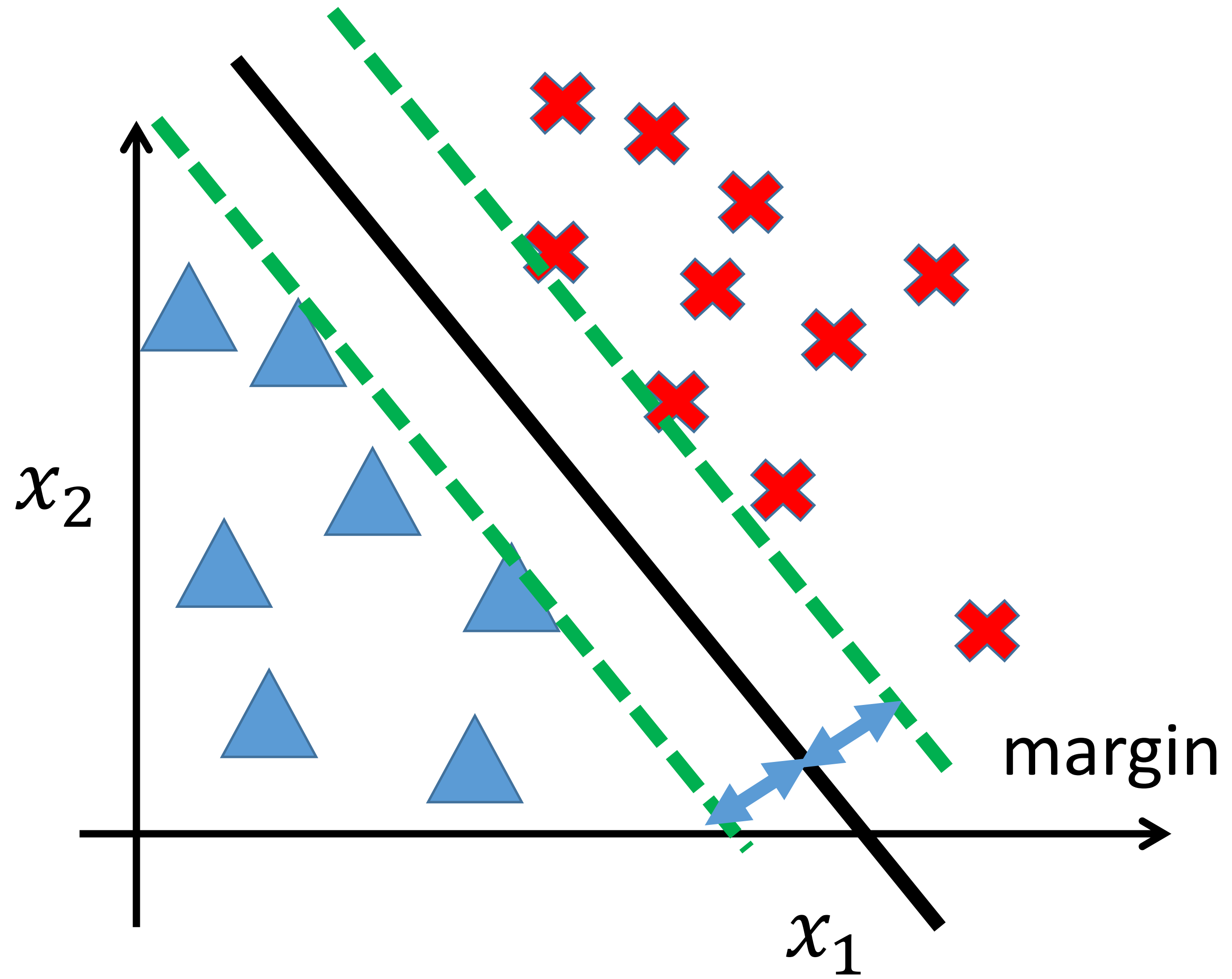
  $\theta^\top x^{(i)} \leq -1$

$$\boxed{\begin{aligned}&\min_{\theta} \tfrac{1}{2}\textstyle\sum_{j=1}^{n}\theta_j^2\\ &\mathrm{s.t.}\quad \theta^\top x^{(i)} \geq 1 \quad \text{if} \quad \mathrm{y}^{(i)} = 1\\ &\phantom{\mathrm{s.t.}}\quad \theta^\top x^{(i)} \leq -1 \ \ \text{if} \quad \mathrm{y}^{(i)} = 0\end{aligned}}$$

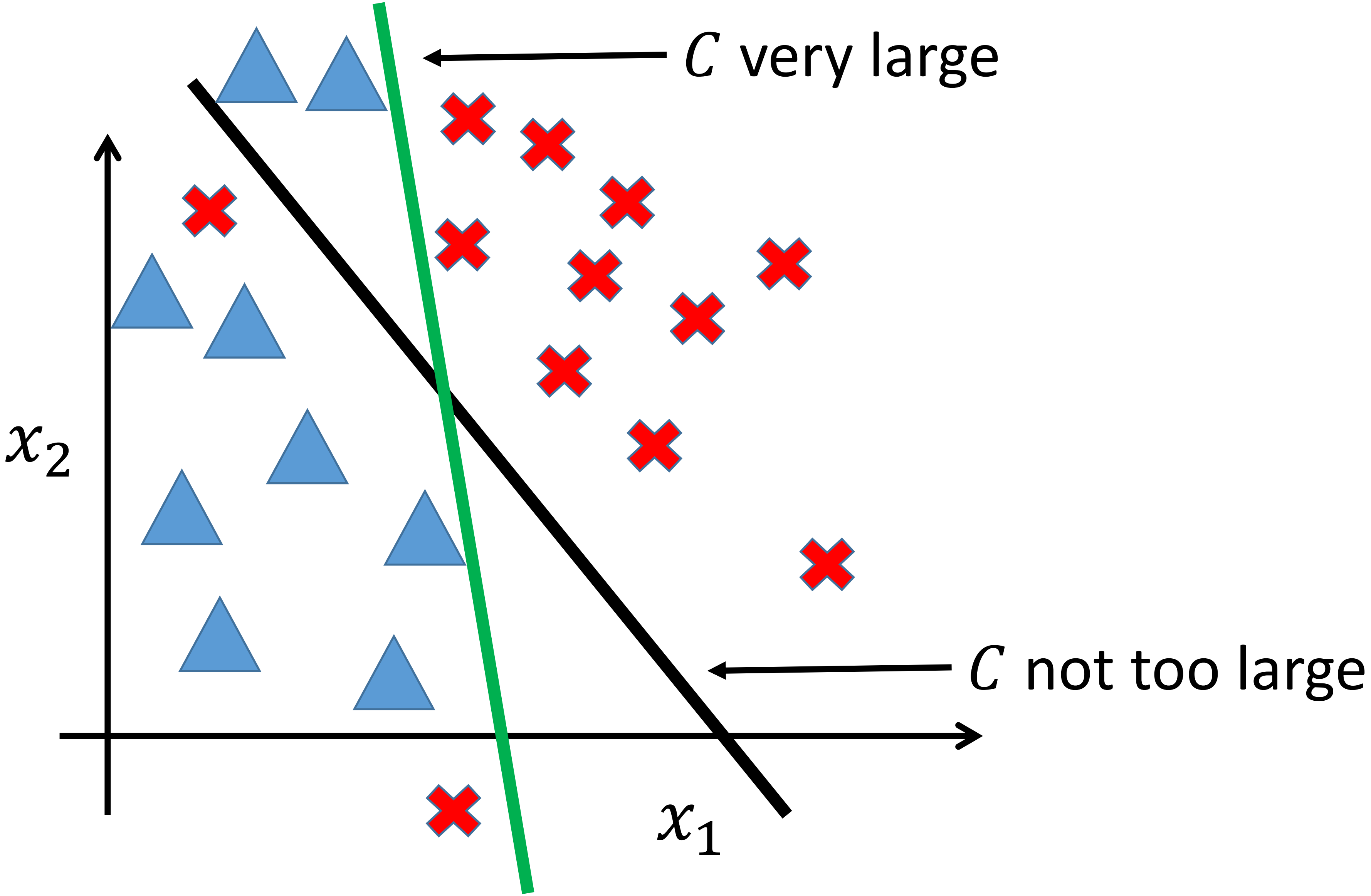
SVM decision boundary: Linearly separable case



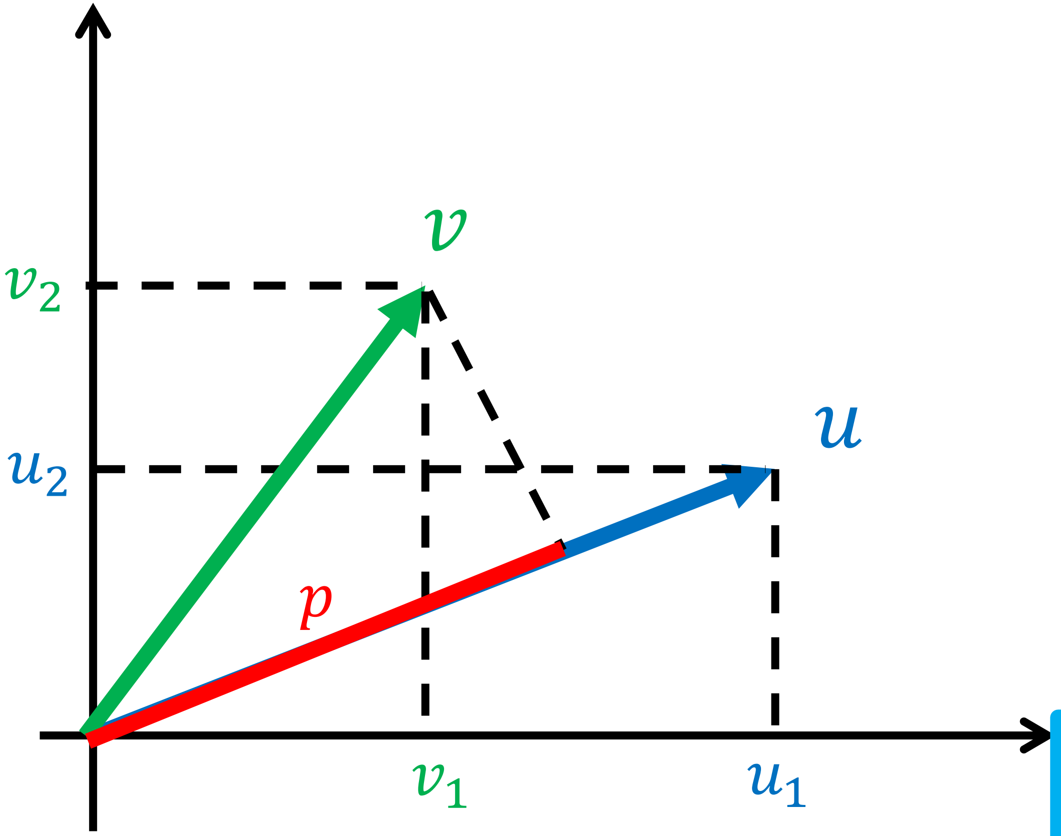
SVM decision boundary: Linearly separable case



Large margin classifier in the presence of outlier



Vector inner product



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

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

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

p = length of projection of v onto u

$$\boxed{\begin{aligned} u^\top v &= p \cdot \|u\| \\ &= \mathrm{u}_1\mathrm{v}_1 + \mathrm{u}_2\mathrm{v}_2 \end{aligned}}$$

SVM decision boundary

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

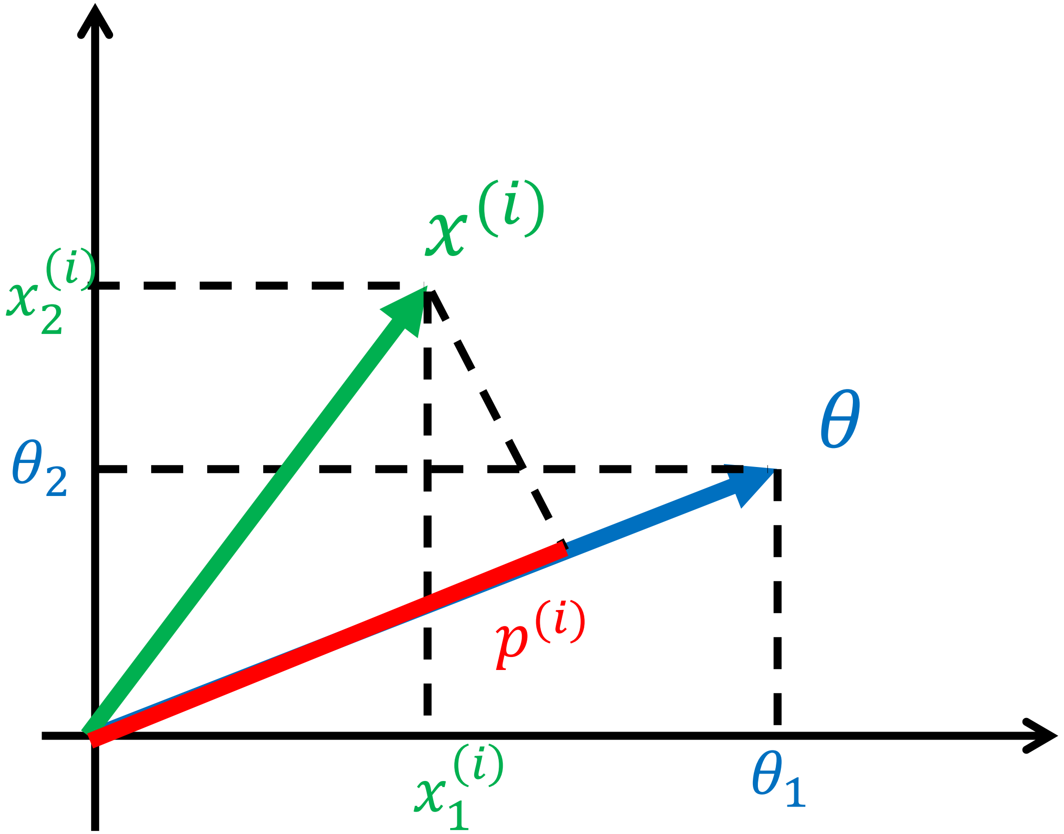
$$\text{s.t.} \quad \theta^\top x^{(i)} \geq 1 \quad \text{if} \quad \mathrm{y}^{(i)} = 1$$

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

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

Simplication: $\theta_0 = 0, n = 2$

What's $\theta^\top x^{(i)}$?



$$\theta^\top x^{(i)} = p^{(i)} \|\theta\|^2$$

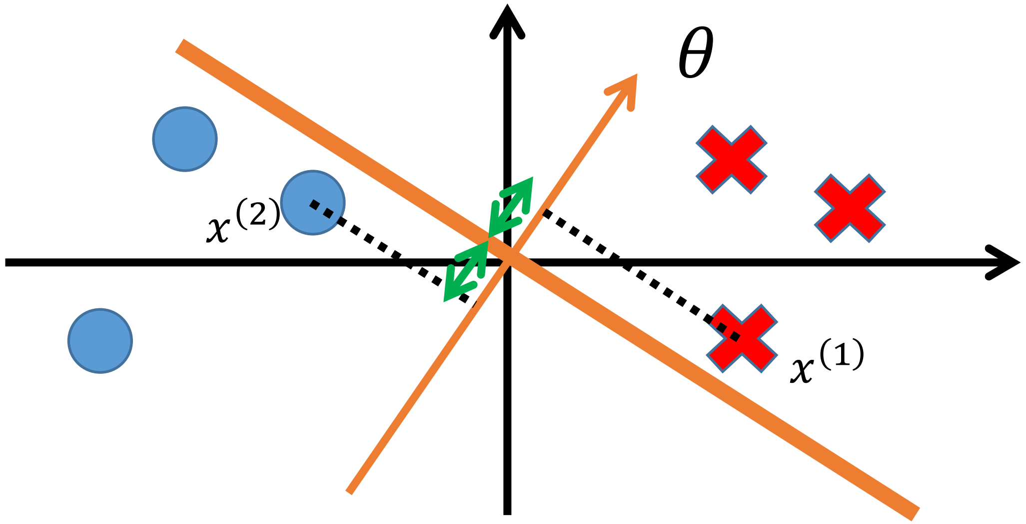
SVM decision boundary

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

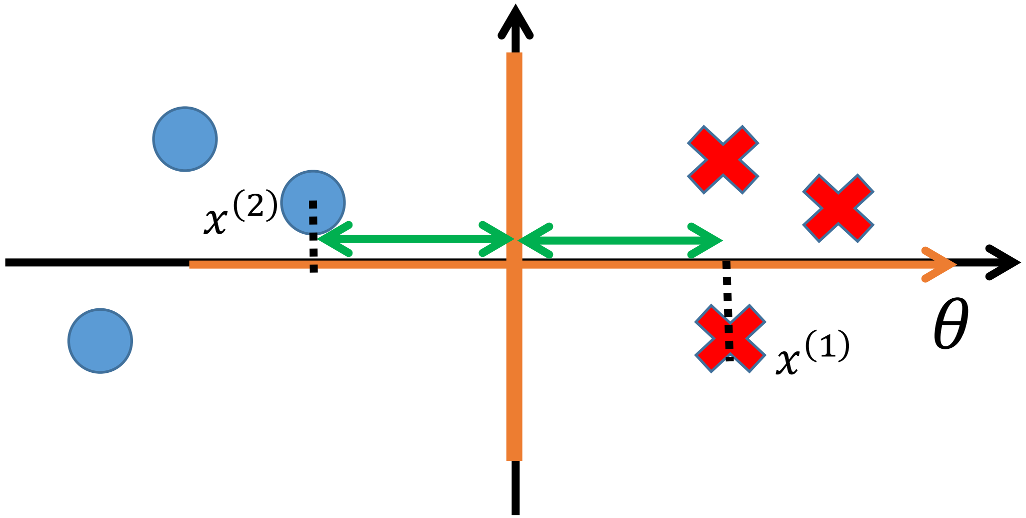
$$\text{s.t.} \quad p^{(i)} \|\theta\|^2 \geq 1 \quad \text{if} \quad y^{(i)} = 1$$
$$p^{(i)} \|\theta\|^2 \leq -1 \quad \text{if} \quad y^{(i)} = 0$$

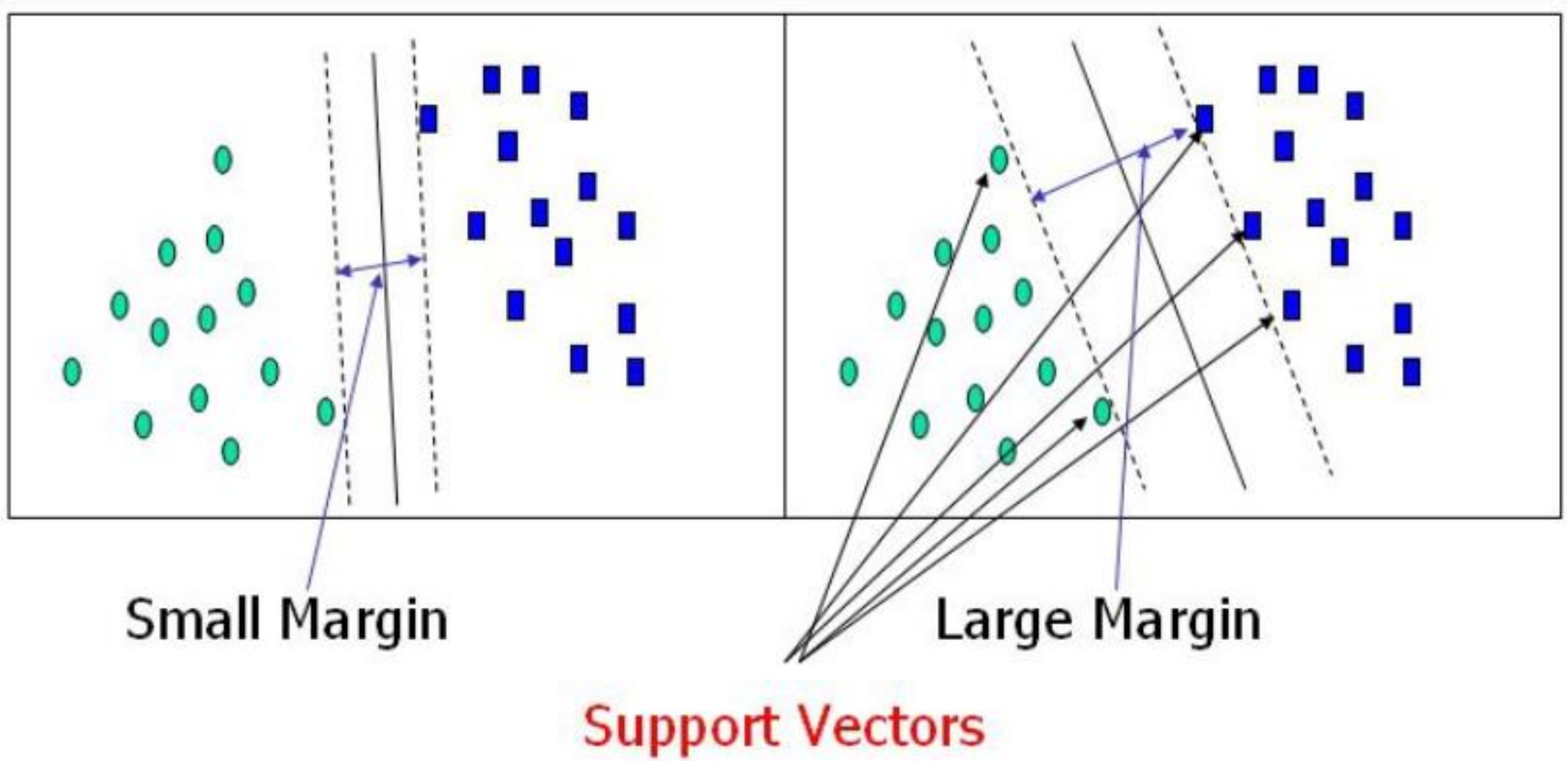
Simplication: $\theta_0 = 0, n = 2$

$p^{(1)}, p^{(2)}$ small $\rightarrow \|\theta\|^2$ large



$p^{(1)}, p^{(2)}$ large $\rightarrow \|\theta\|^2$ can be small

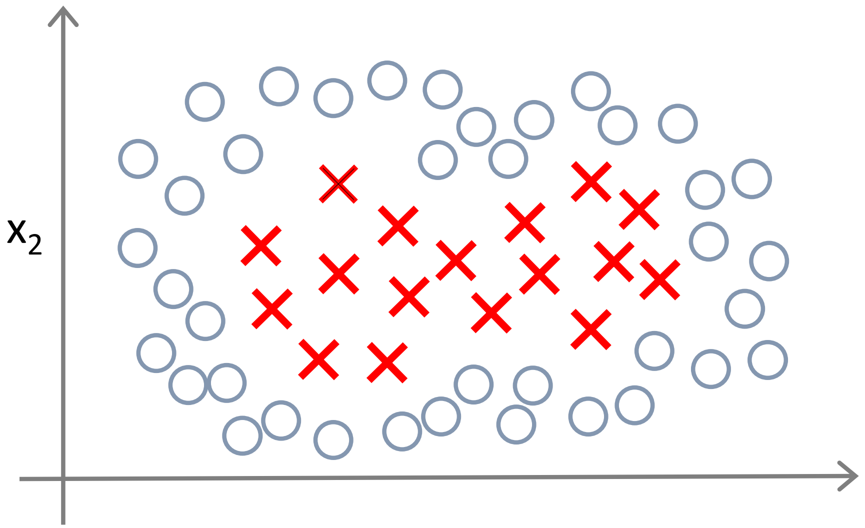



Small Margin
Large Margin
Support Vectors

Support Vector Machine

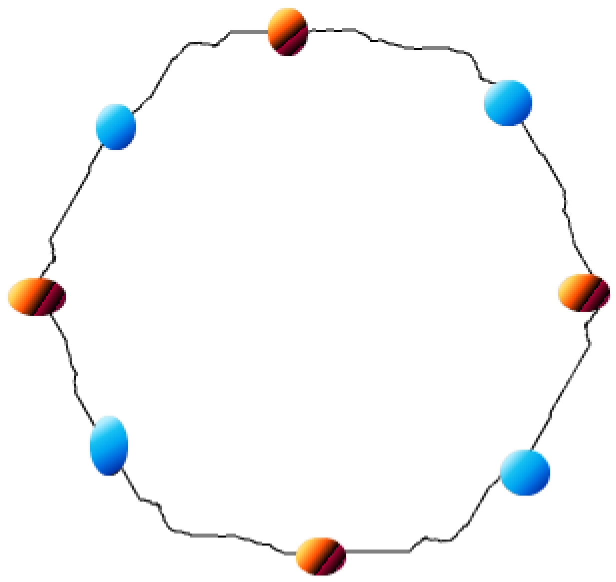
- Cost function
- Large margin classification
- **Kernels**
- Using an SVM

SVM: Introducing Nonlinearity



Need to generate new features!

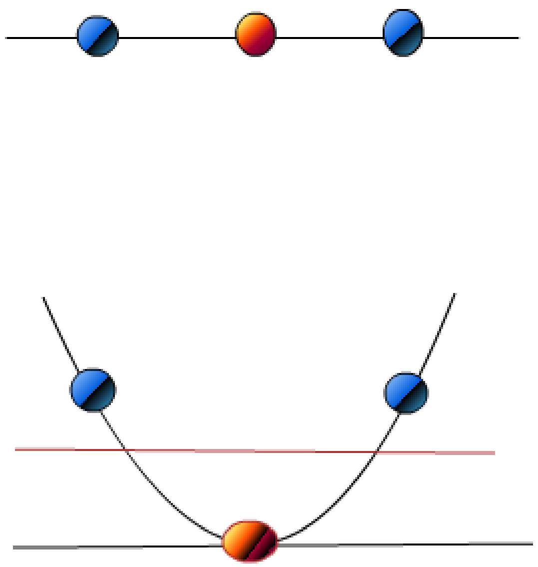
Kernel trick ?



Inseparable data! T________T
What to do? ☹

- There is no straight line (hyper plane in 2 dimensions) which can separate red and blue dots.
- Generate more features!

Kernel trick? Simple as it is



- Inseparable data

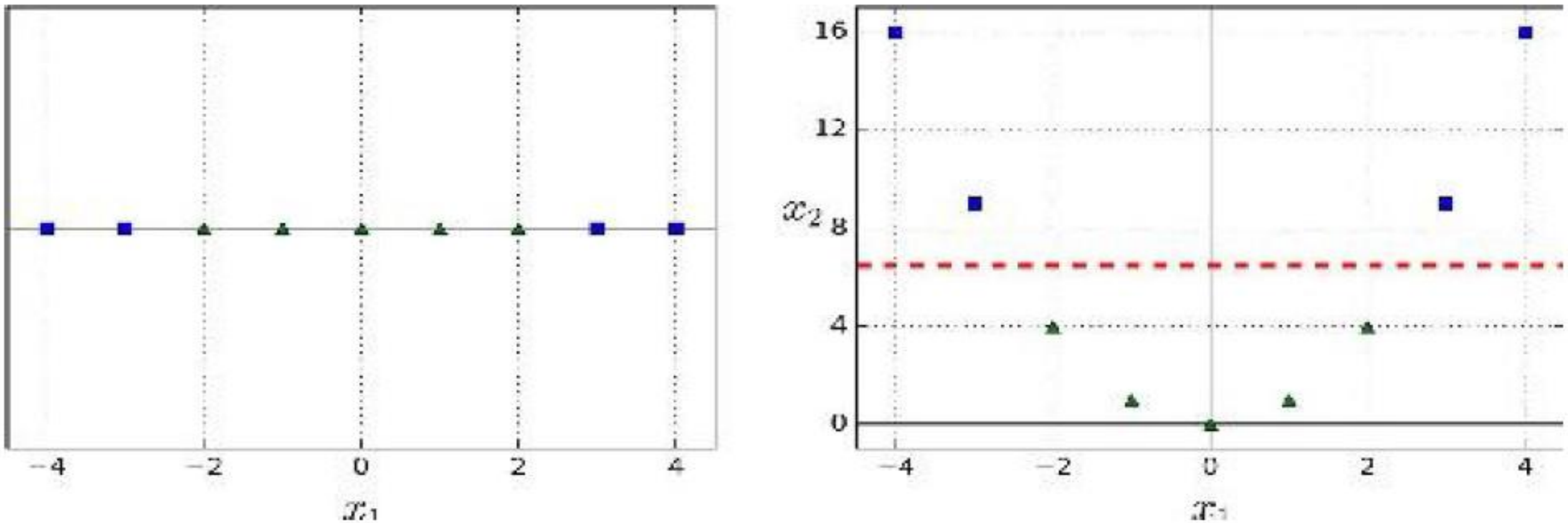
$$x \to (x, x^2)$$

- project all points up to a two dimensional space using the mapping
- We can indeed find a hyper plane to separate data with SVM.

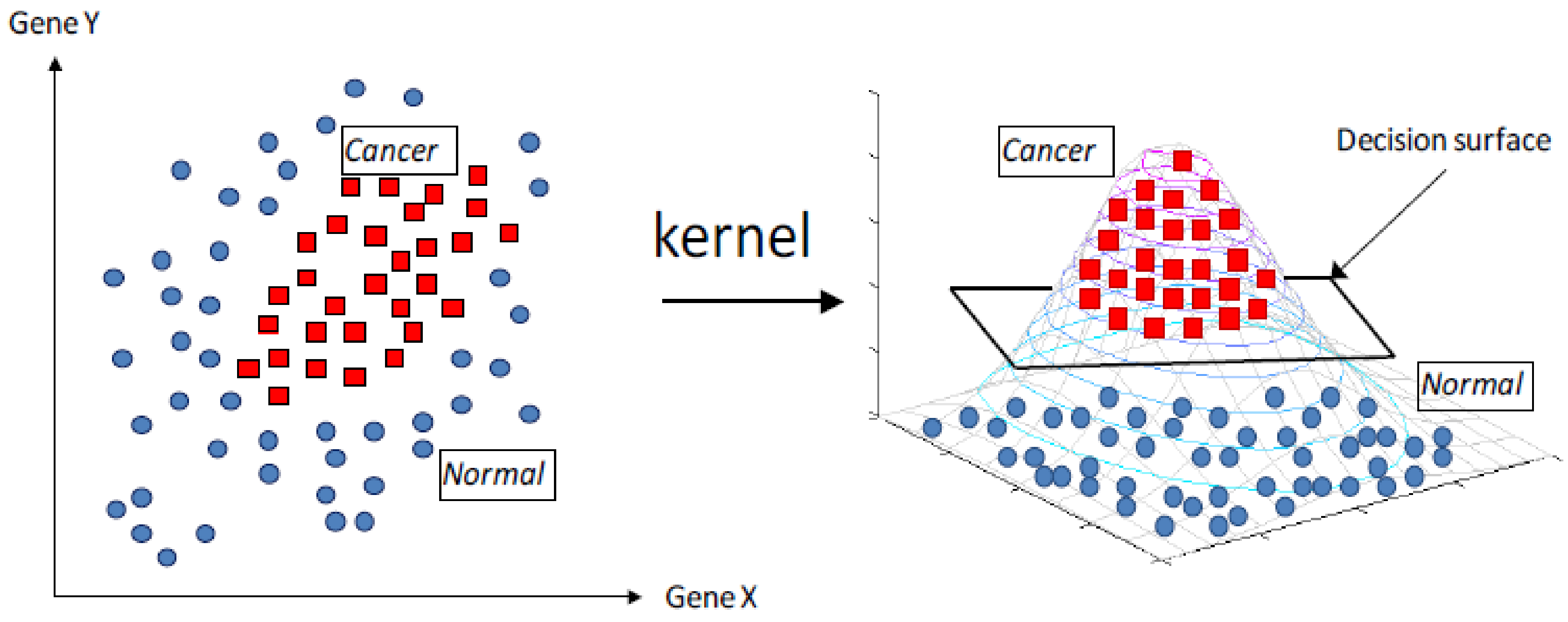
The mapping $x \to (x, x^2)$ in this case is called **KERNEL FUNCTION**

Nonlinear SVM classification

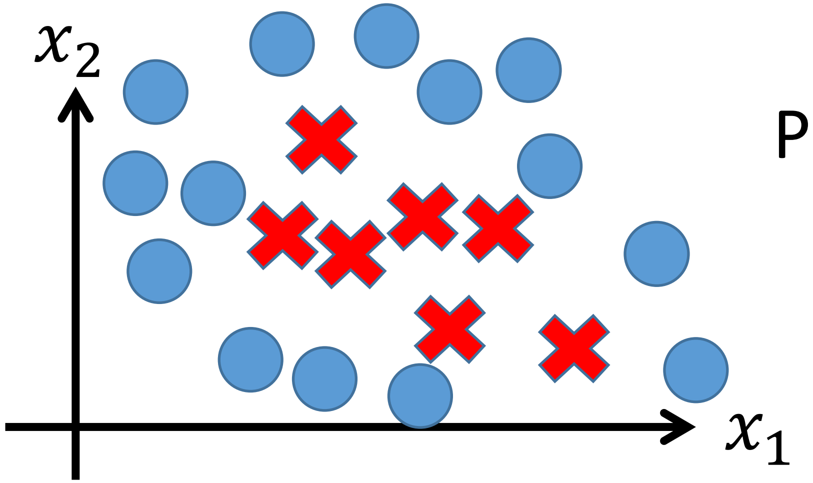
- Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable. One approach to handling nonlinear datasets is to add more features, such as polynomial features, in some cases this can result in a linearly separable dataset.
- Consider the left plot in figure: it represents a simple dataset with just one feature x1.
- This dataset is not linearly separable, as you can see. But if you add a second feature **$x2 = (x1)^2$**, the resulting 2D dataset is perfectly linearly separable.



Example of kernel trick



Non-linear decision boundary



Predict $y = 1$ if

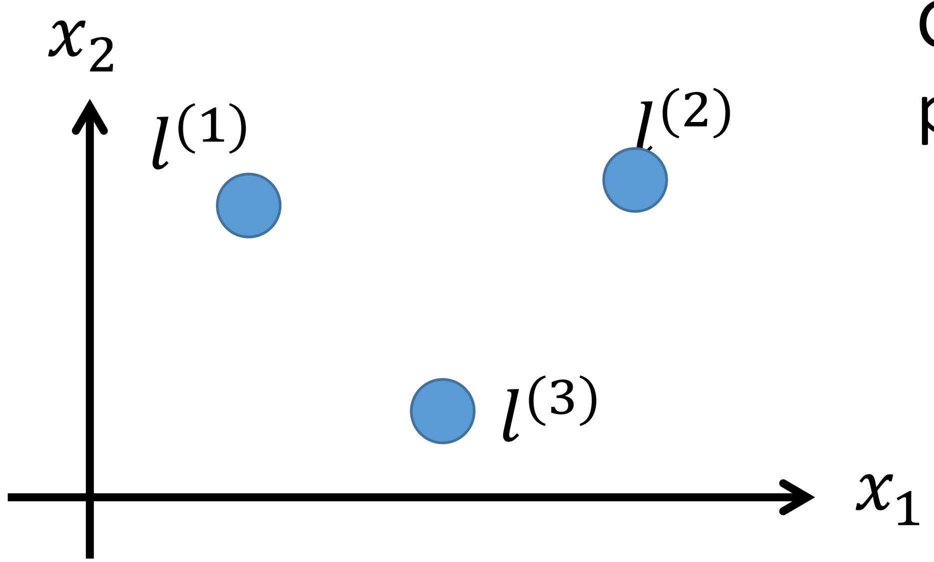
$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots$$

$$f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, \cdots$$

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

Kernel



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

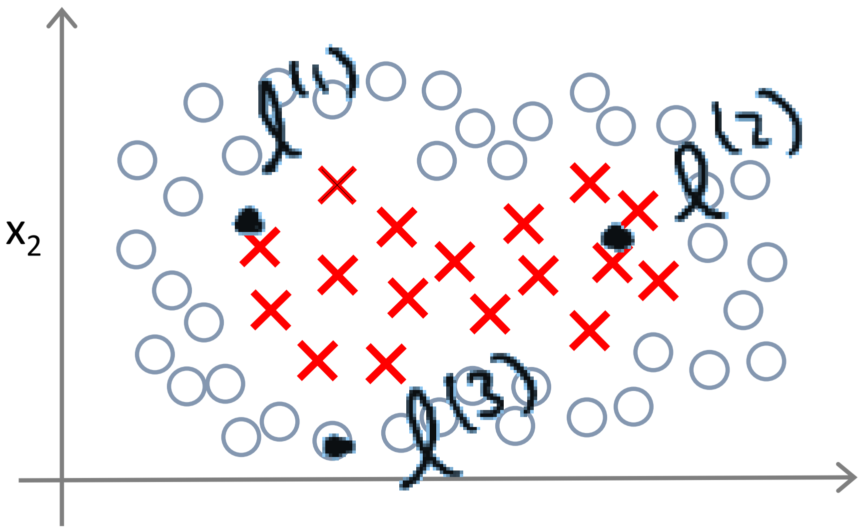
$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$f_3 = \text{similarity}(x, l^{(3)})$$

Gaussian kernel

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

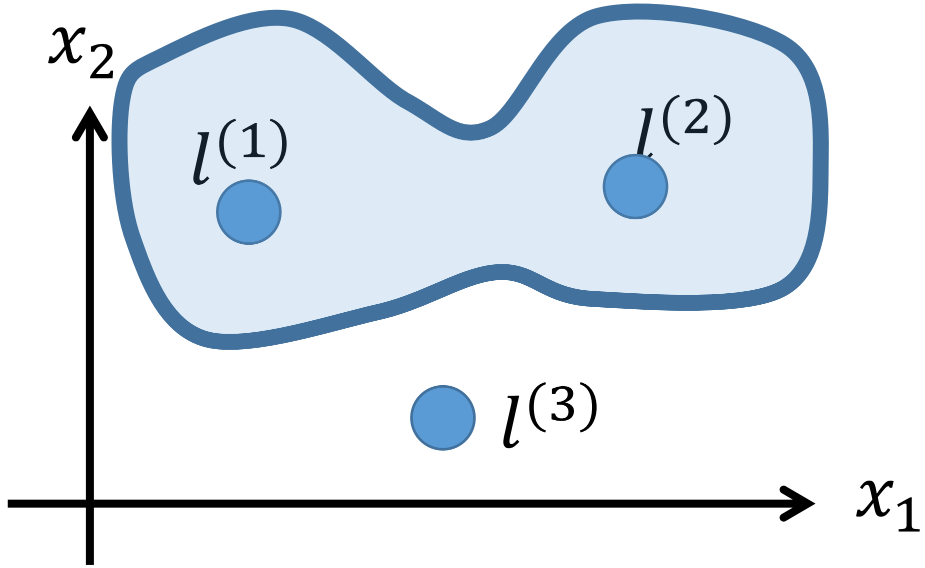
Kernel trick in practice

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



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

If x if far from $l^{(1)}$: f is approx 0



Predict $y = 1$ if

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

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

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$f_3 = \text{similarity}(x, l^{(3)})$$

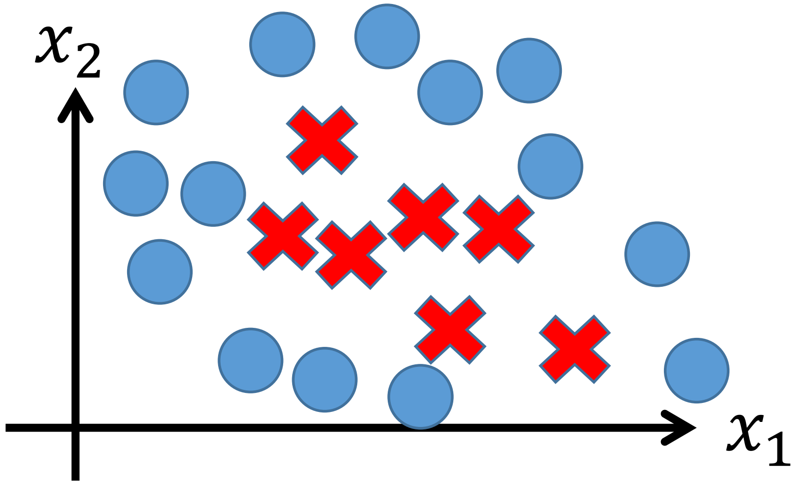
Choosing the landmarks

- Given x

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

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

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



$l^{(1)}$
$l^{(2)}$
$l^{(3)}$
$\vdots$
$l^{(m)}$

SVM with kernels

- Given $\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(m)}, y^{(m)}\right)$
- Choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, l^{(3)} = x^{(3)}, \cdots, l^{(m)} = x^{(m)}$
- Given example x:
  - $f_1 = \text{similarity}\left(x, l^{(1)}\right)$
  - $f_2 = \text{similarity}\left(x, l^{(2)}\right)$
  - $\cdots$

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

- For training example $\left(x^{(i)}, y^{(i)}\right)$:
  - $x^{(i)} \rightarrow f^{(i)}$
  - $f_1^{(i)} = \text{similarity}(x^{(i)}, l^1)$
  - $f_2^{(i)} = \text{similarity}(x^{(i)}, l^2)$ ……. $f_m^{(i)} = \text{similarity}(x^{(i)}, l^m)$

SVM with kernels

- Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$
  - Predict $y = 1$ if $\theta^\top f \geq 0$

- **Training (original)**

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

- **Training (with kernel)**

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

SVM parameters

- $C \left(= \frac{1}{\lambda}\right)$
  Large C: Lower bias, high variance - may lead to overfitting
  Small C: Higher bias, low variance - may lead to underfitting

- 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

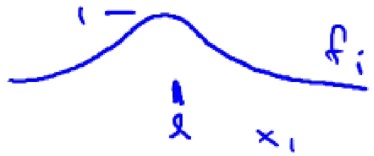
SVM Parameters

C ($= \frac{1}{\lambda}$). Large C: Lower bias, high variance (may lead to overfitting) Small λ

Small C: Higher bias, low variance (may lead to underfitting) Large λ

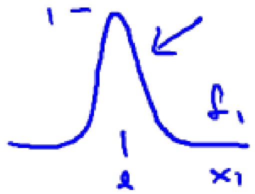
σ^2 Large σ^2: Features f_i vary more smoothly.

Higher bias, lower variance.



Small σ^2: Features f_i vary less smoothly.

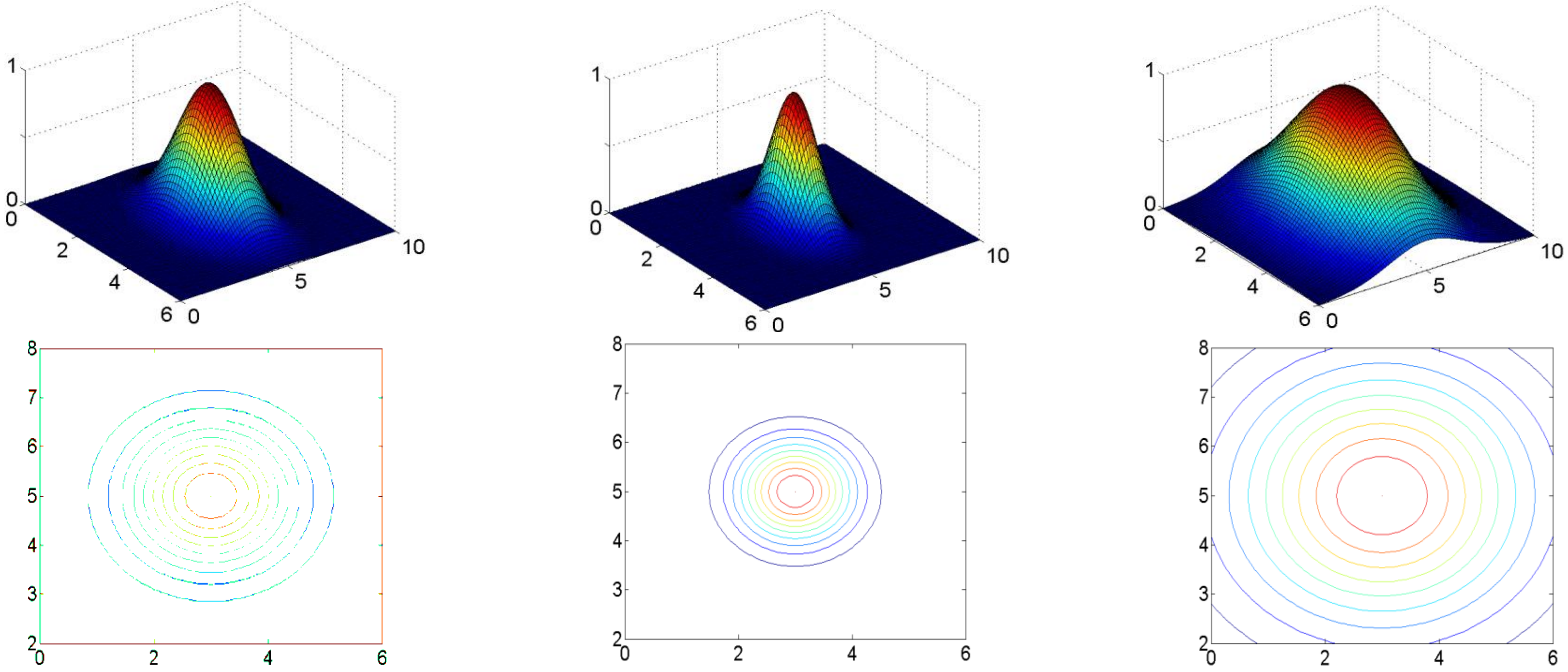
Lower bias, higher variance.



SVM with radial basis kernels: sigma

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

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



Support Vector Machine

- Cost function
- Large margin classification
- Kernels
- **Using an SVM**

Using SVM

- SVM software package (e.g., liblinear, libsvm) to solve for θ

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

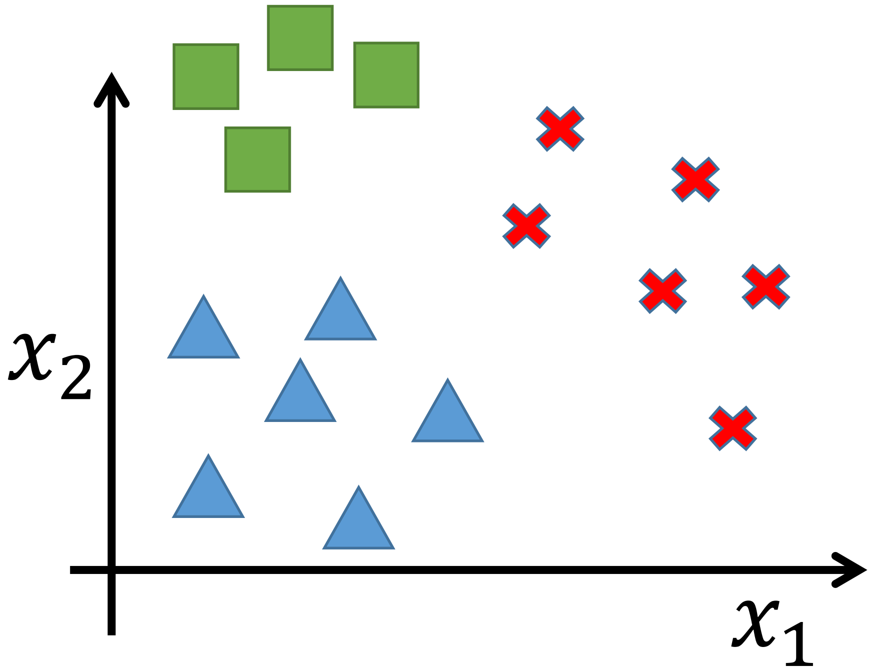
- Linear kernel: Predict $y = 1 \quad \text{if} \quad \theta^\top x \geq 0$
- Gaussian kernel:
  - $f_i = \exp(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2})$, where $l^{(i)} = x^{(i)}$
  - Need to choose σ^2. Need proper feature scaling

Kernel (similarity) functions

- Note: not all similarity functions make valid kernels.

- Many off-the-shelf kernels available:
  - Polynomial kernel
  - String kernel
  - Chi-square kernel
  - Histogram intersection kernel

Multi-class classification

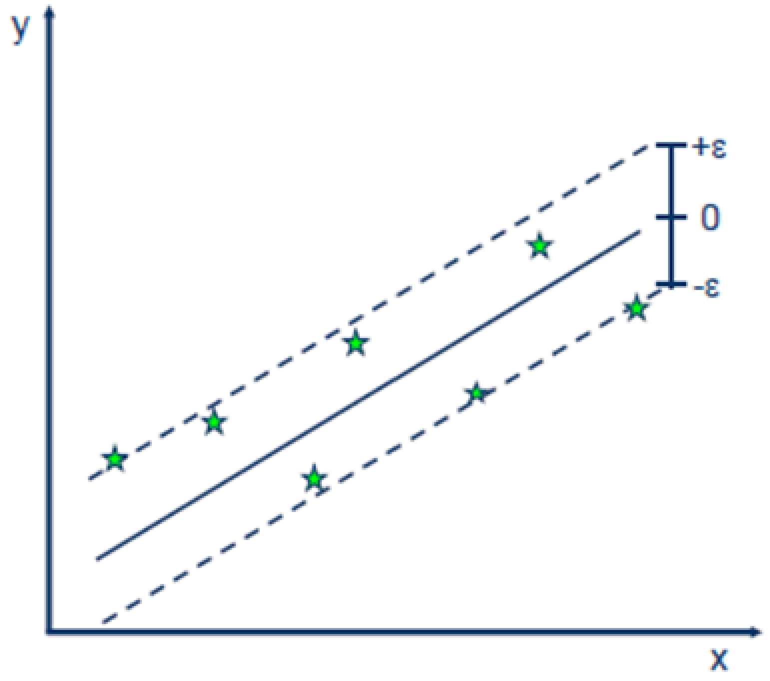


- Use one-vs.-all method. Train K SVMs, one to distinguish $y = i$ from the rest, get $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(K)}$

- Pick class i with the largest ${\theta^{(i)}}^\top x$

Support Vector Regression

- Do kernel trick
- Build linear model using parameter **ε - precision**



Support Vector Regression

- Find a function, f(x), with at most ε-deviation from the target y

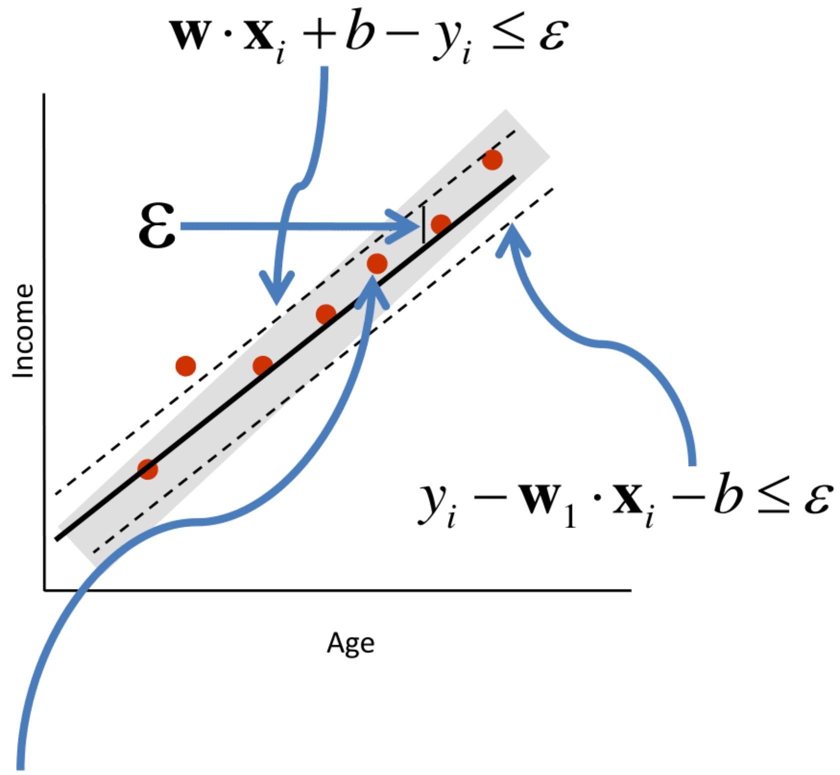
The problem can be written as a convex optimization problem

$$\min \frac{1}{2} \| \mathbf{w} \|^2$$
$$s.t.\ y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon;$$
$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \le \varepsilon;$$

C: trade off the complexity

What if the problem is not feasible?

We can introduce slack variables (similar to soft margin loss function).



We do not care about errors as long as they are less than ε

Logistic regression vs. SVMs

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

1. **If $\boldsymbol{n}$ is large (relative to $\boldsymbol{m}$)**: $(n = 10{,}000, \mathrm{m} = 10 - 1000)$
   $\rightarrow$ Use logistic regression or SVM without a kernel ("linear kernel")
2. **If $\boldsymbol{n}$ is small, $\boldsymbol{m}$ is intermediate**: $(n = 1 - 1000, \mathrm{m} = 10 - 10{,}000)$
   $\rightarrow$ Use SVM with Gaussian kernel
3. **If $\boldsymbol{n}$ is small, $\boldsymbol{m}$ is large**: $(n = 1 - 1000, \mathrm{m} = 50{,}000+)$
   $\rightarrow$ Create/add more features, then use logistic regression of linear SVM

Neural network likely to work well for most of these case,
but slower to train

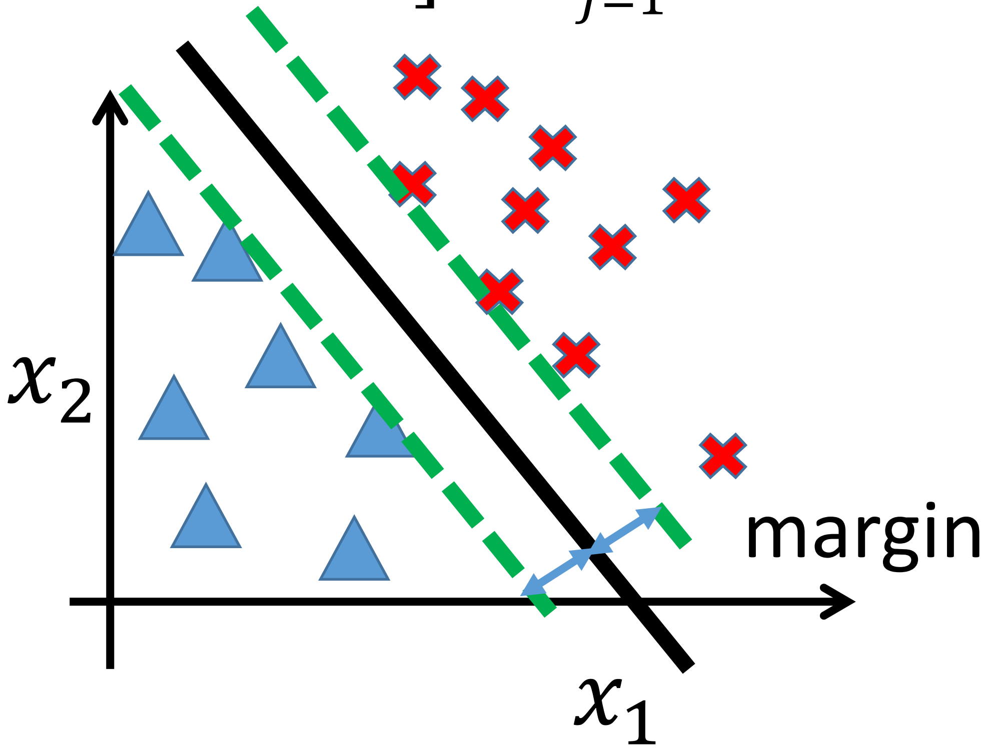
Slide credit: Andrew Ng

Things to remember

- **Cost function**

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

- **Large margin classification**



- **Kernels**
- **Using an SVM**