Support Vector Machines & Kernels

Doing *really* well with linear decision surfaces

Adapted from slides by Tim Oates

Outline

- Prediction
  - Why might predictions be wrong?
- Support vector machines
  - Doing really well with linear models
- Kernels
  - Making the non-linear linear

Why Might Predictions be Wrong?

- True non-determinism
  - Flip a biased coin
  - p(heads) = θ
  - Estimate θ
  - If $\theta > 0.5$ predict 'heads', else 'tails'

Lots of ML research on problems like this:

- Learn a model
- Do the best you can in expectation

Why Might Predictions be Wrong?

- Partial observability
  - Something needed to predict y is missing from observation $\mathbf{x}$
  - N-bit parity problem
    - $\mathbf{x}$ contains N-1 bits (hard PO)
    - $\mathbf{x}$ contains N bits but learner ignores some of them (soft PO)

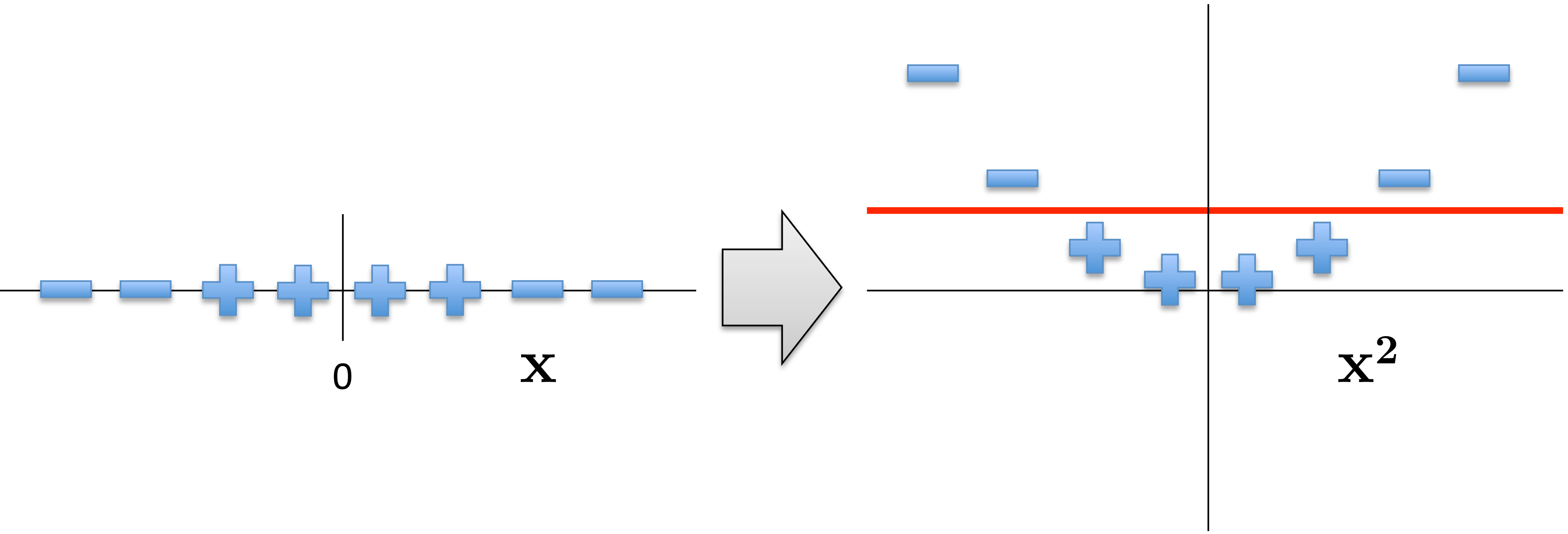
- Noise in the observation $\mathbf{x}$
  - Measurement error
  - Instrument limitations

Why Might Predictions be Wrong?

- True non-determinism
- Partial observability
  - hard, soft
- Representational bias
- Algorithmic bias
- Bounded resources

Representational Bias

- Having the right features ($\mathbf{x}$) is crucial



Support Vector Machines

Doing ***Really*** Well with Linear Decision Surfaces

Strengths of SVMs

- Good generalization
  - in theory
  - in practice
- Works well with few training instances
- Find globally best model
- Efficient algorithms
- Amenable to the kernel trick

Minor Notation Change

To better match notation used in SVMs

...and to make matrix formulas simpler

We will drop using superscripts for the i^{th} instance

i^{th} instance	$\boldsymbol{x}^{(i)} \longrightarrow \mathbf{x}_i$	**Bold** denotes vector
i^{th} instance label	$y^{(i)} \longrightarrow y_i$	**Non-bold** denotes scalar
j^{th} feature of i^{th} instance	$x_j^{(i)} \longrightarrow x_{ij}$	

Linear Separators

- Training instances

  $$\mathbf{x} \in \mathbb{R}^{d+1}, x_0 = 1$$

  $$y \in \{-1, 1\}$$

- Model parameters

  $$\boldsymbol{\theta} \in \mathbb{R}^{d+1}$$

- Hyperplane

  $$\boldsymbol{\theta}^\intercal \mathbf{x} = \langle \boldsymbol{\theta}, \mathbf{x} \rangle = 0$$

- Decision function

  $$h(\mathbf{x}) = \operatorname{sign}(\boldsymbol{\theta}^\intercal \mathbf{x}) = \operatorname{sign}(\langle \boldsymbol{\theta}, \mathbf{x} \rangle)$$

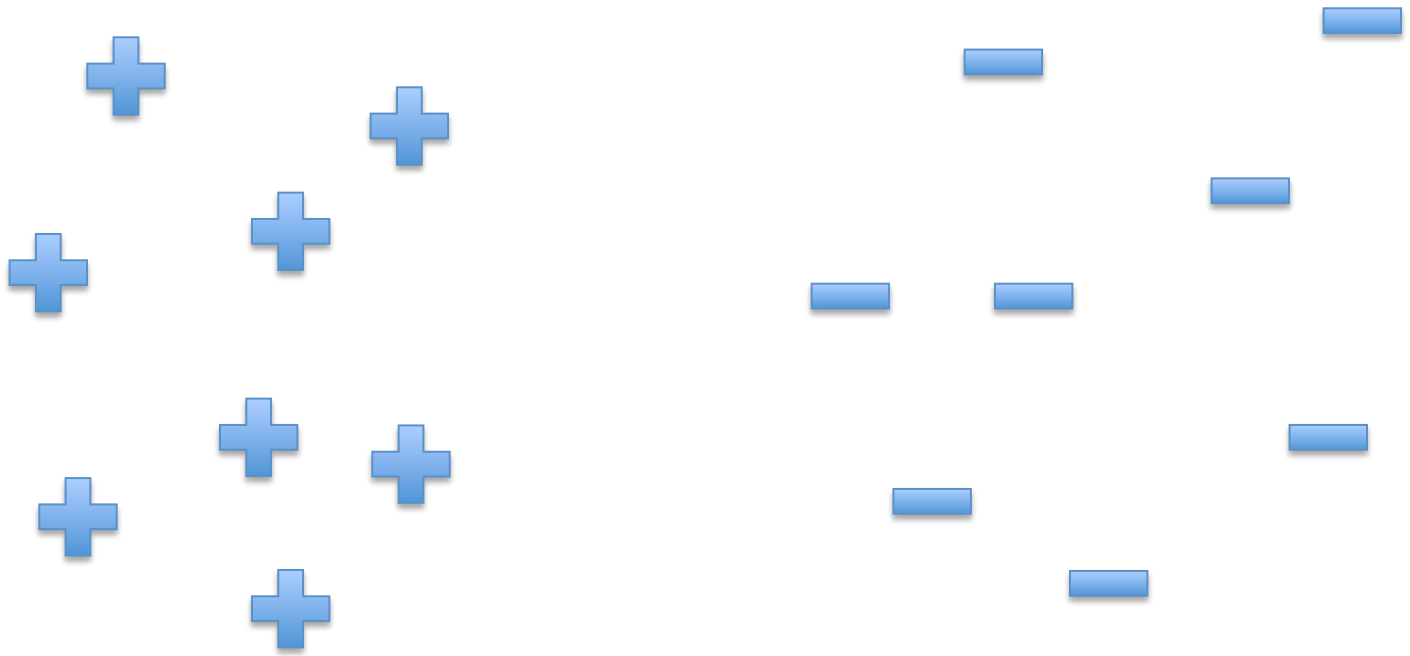
<u>Recall:</u>

Inner (dot) product:

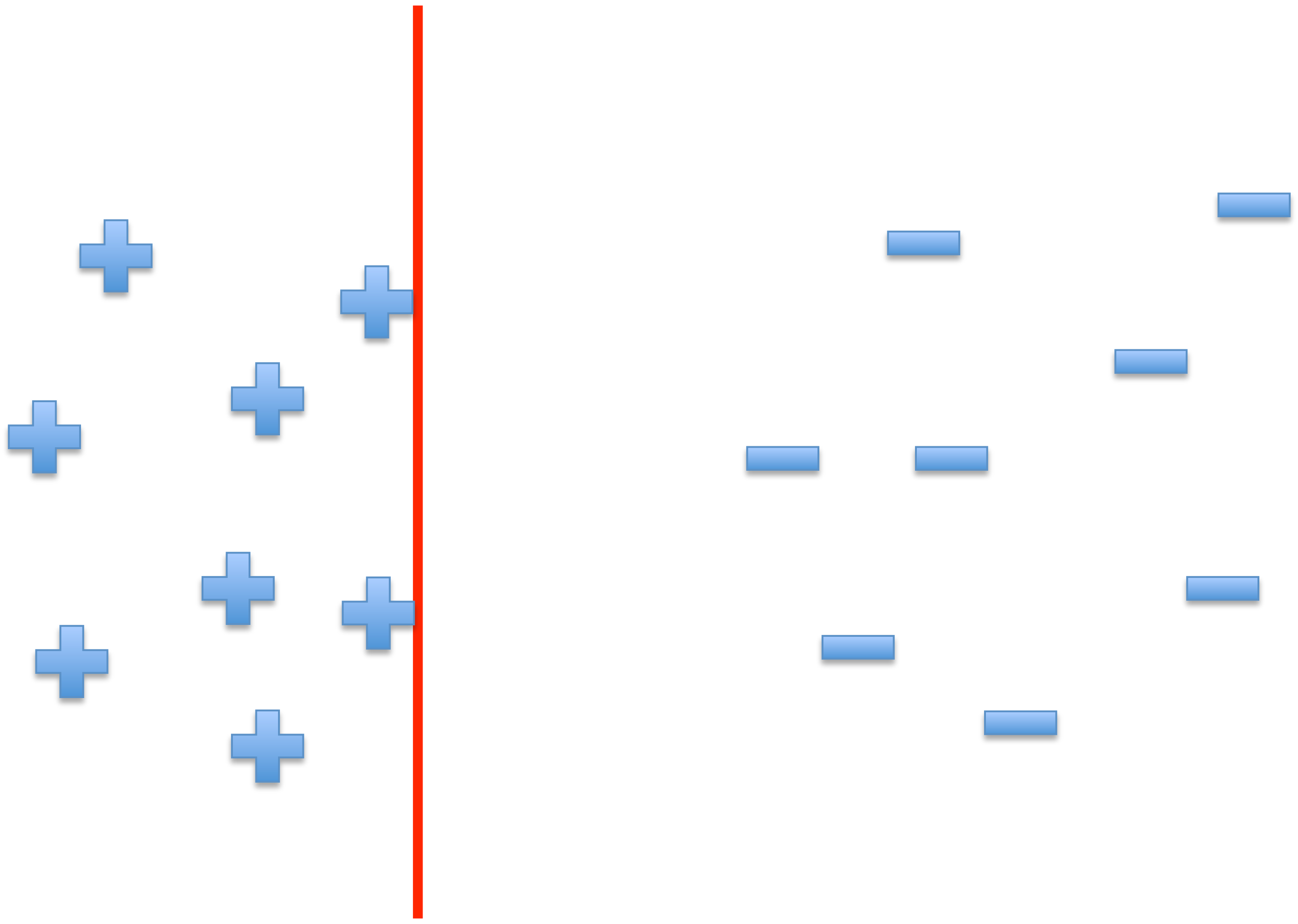
$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^\intercal \mathbf{v}$$

$$= \sum_i u_i v_i$$

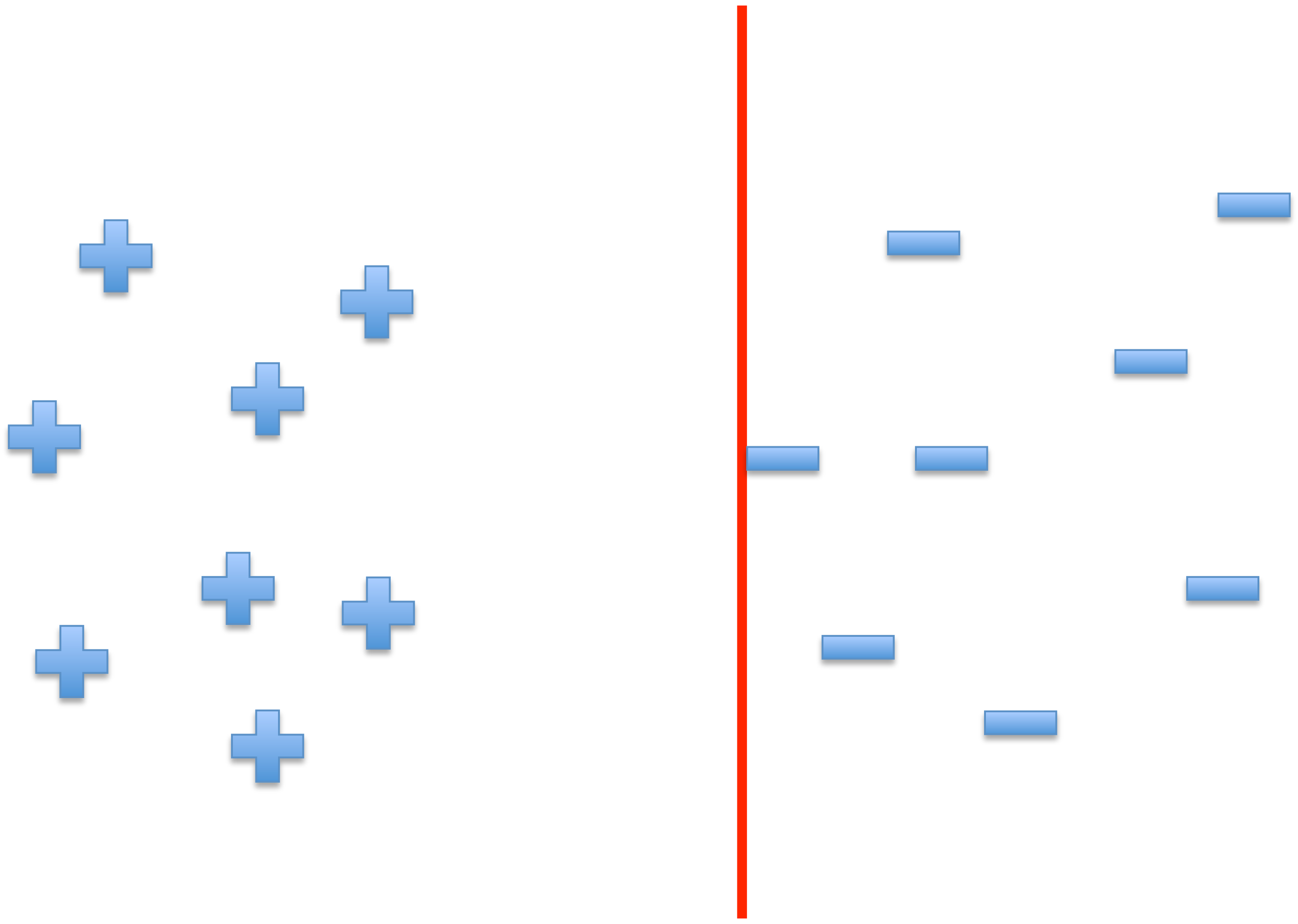
Intuitions



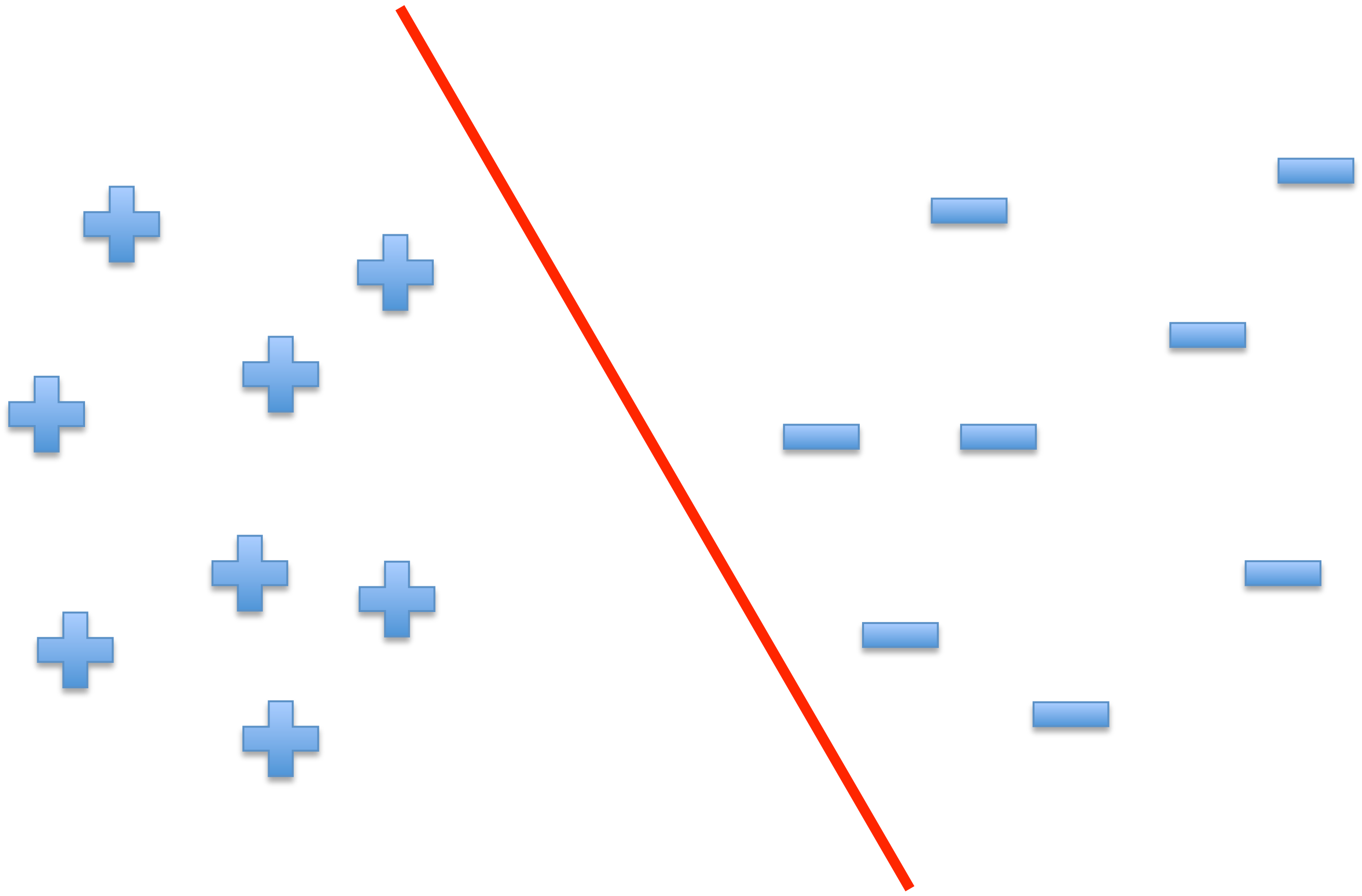
Intuitions



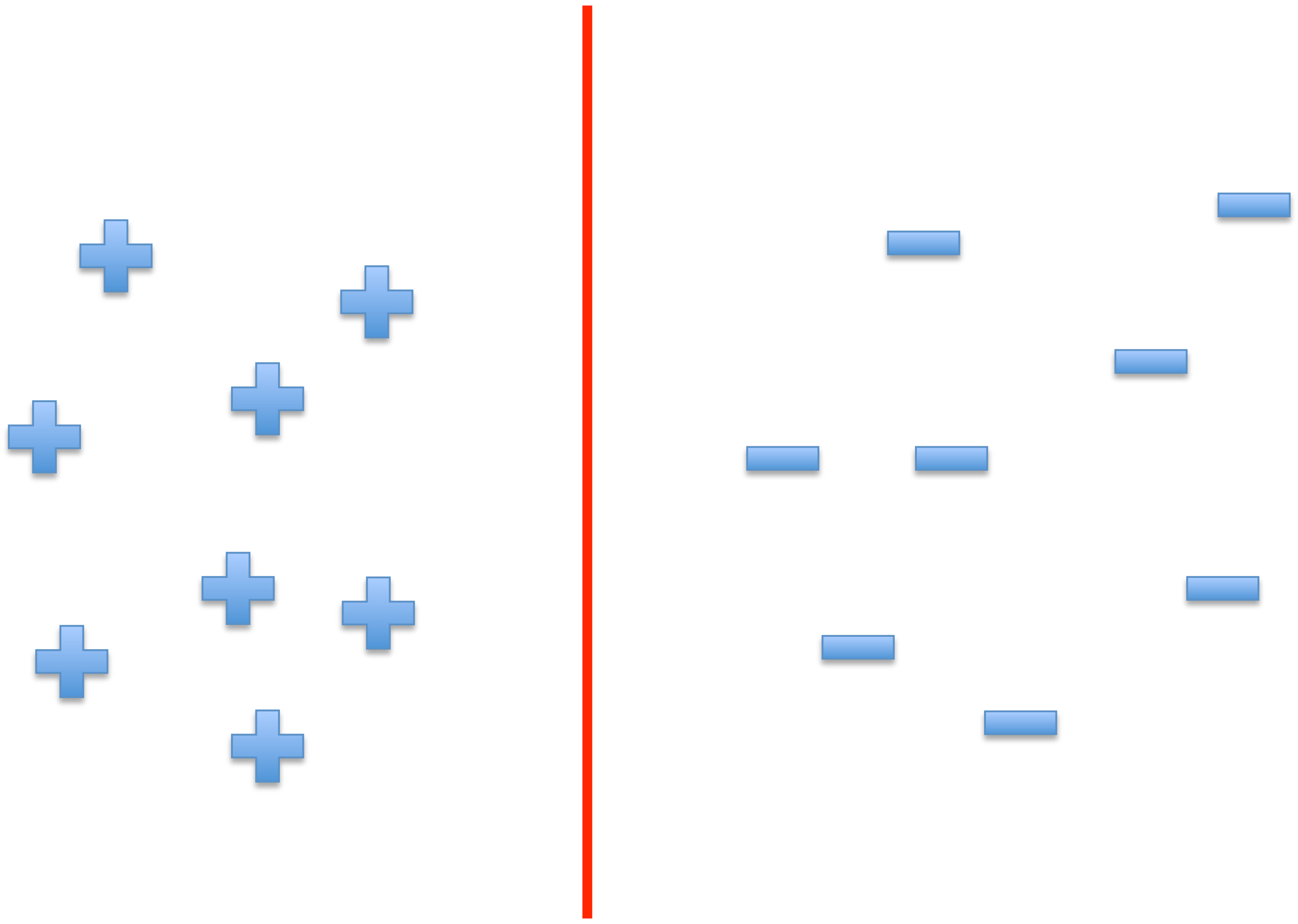
Intuitions



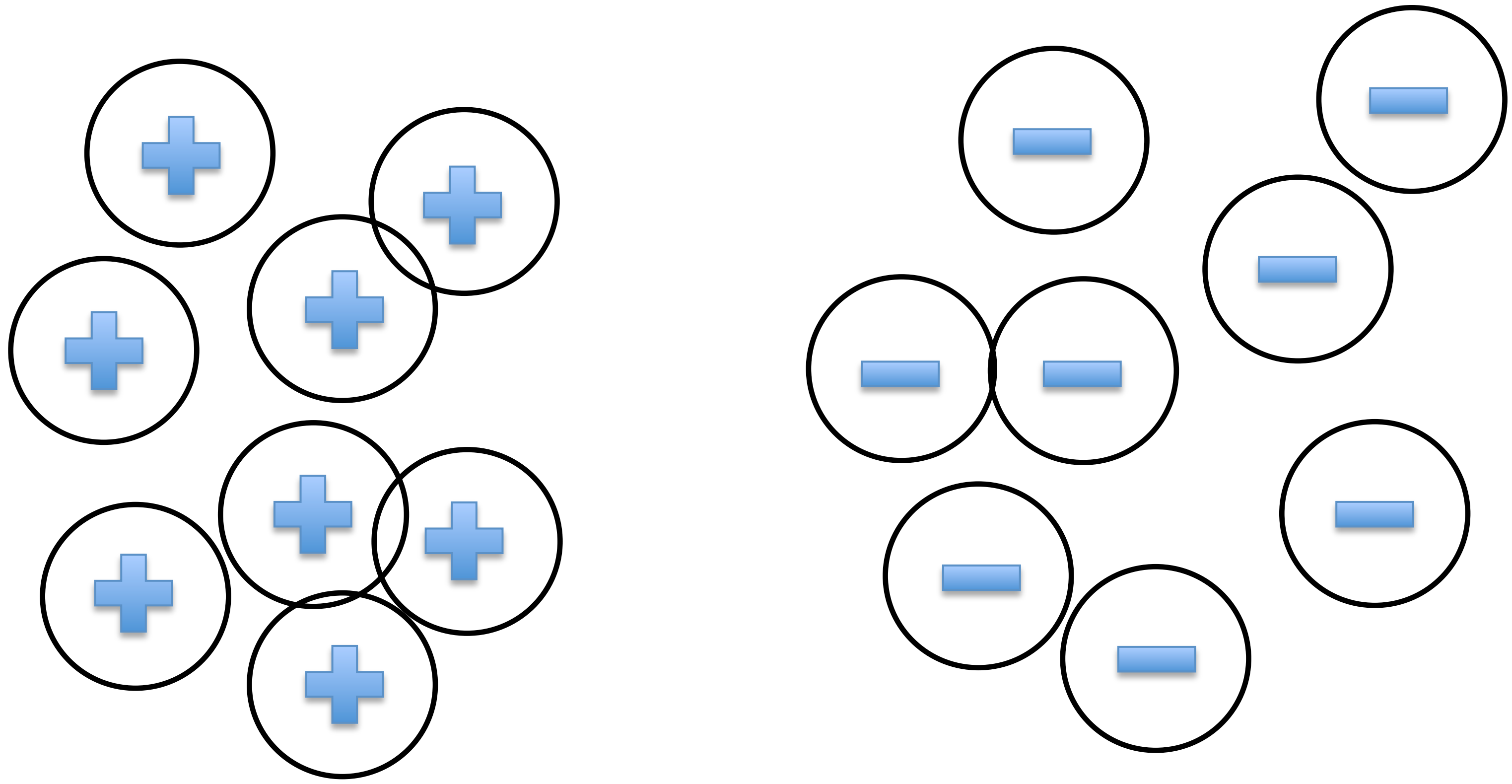
Intuitions



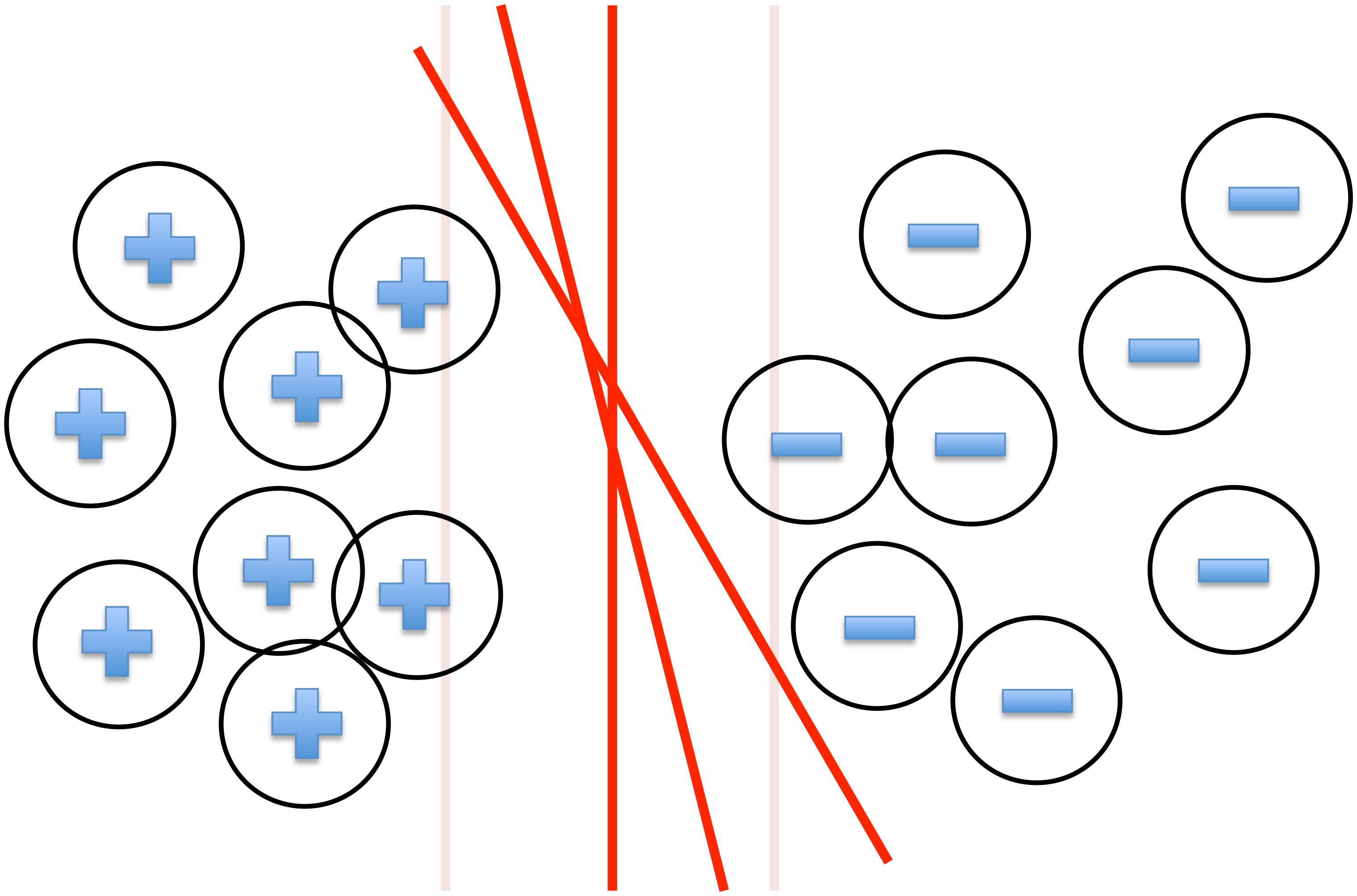
A “Good” Separator



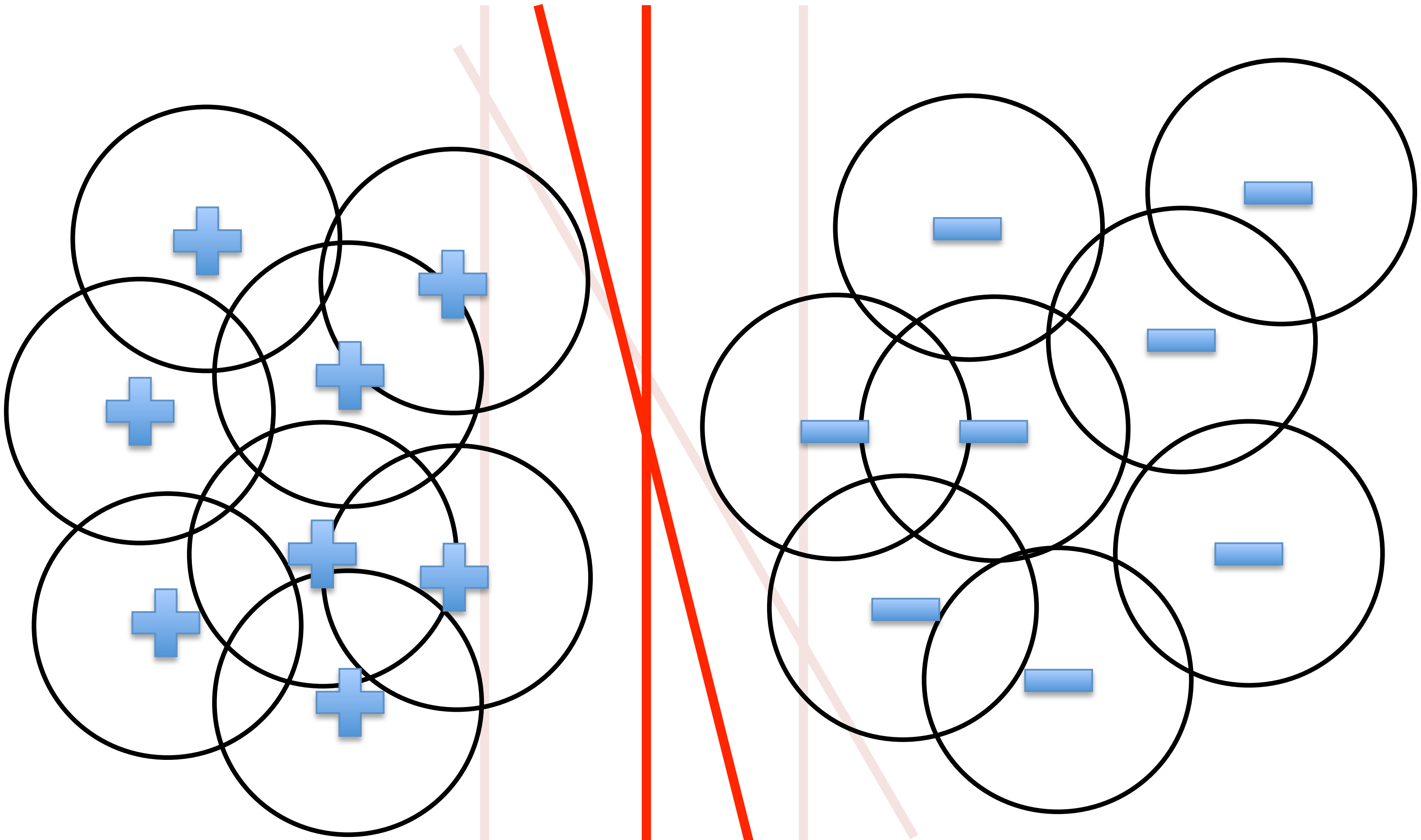
Noise in the Observations



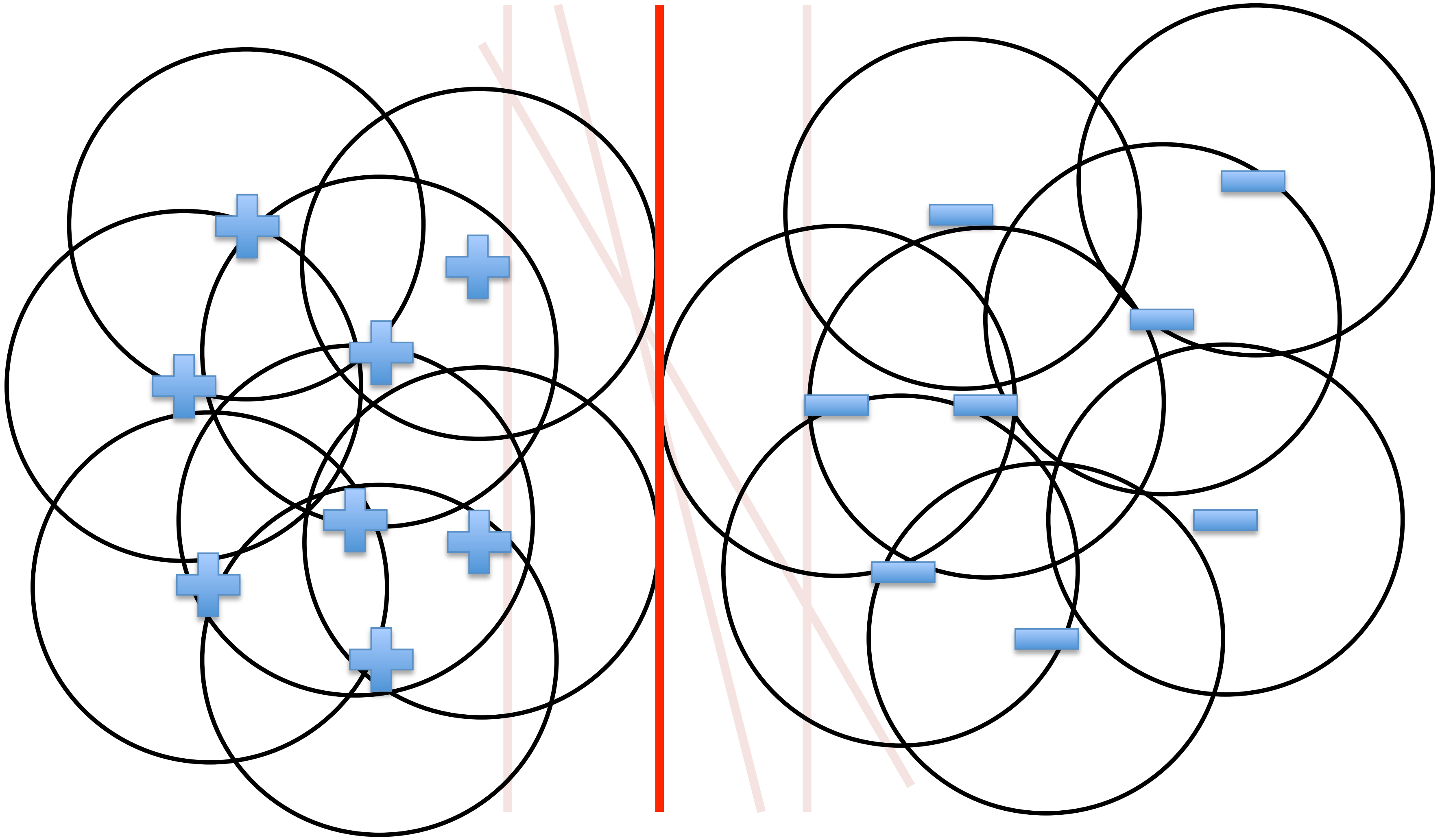
Ruling Out Some Separators



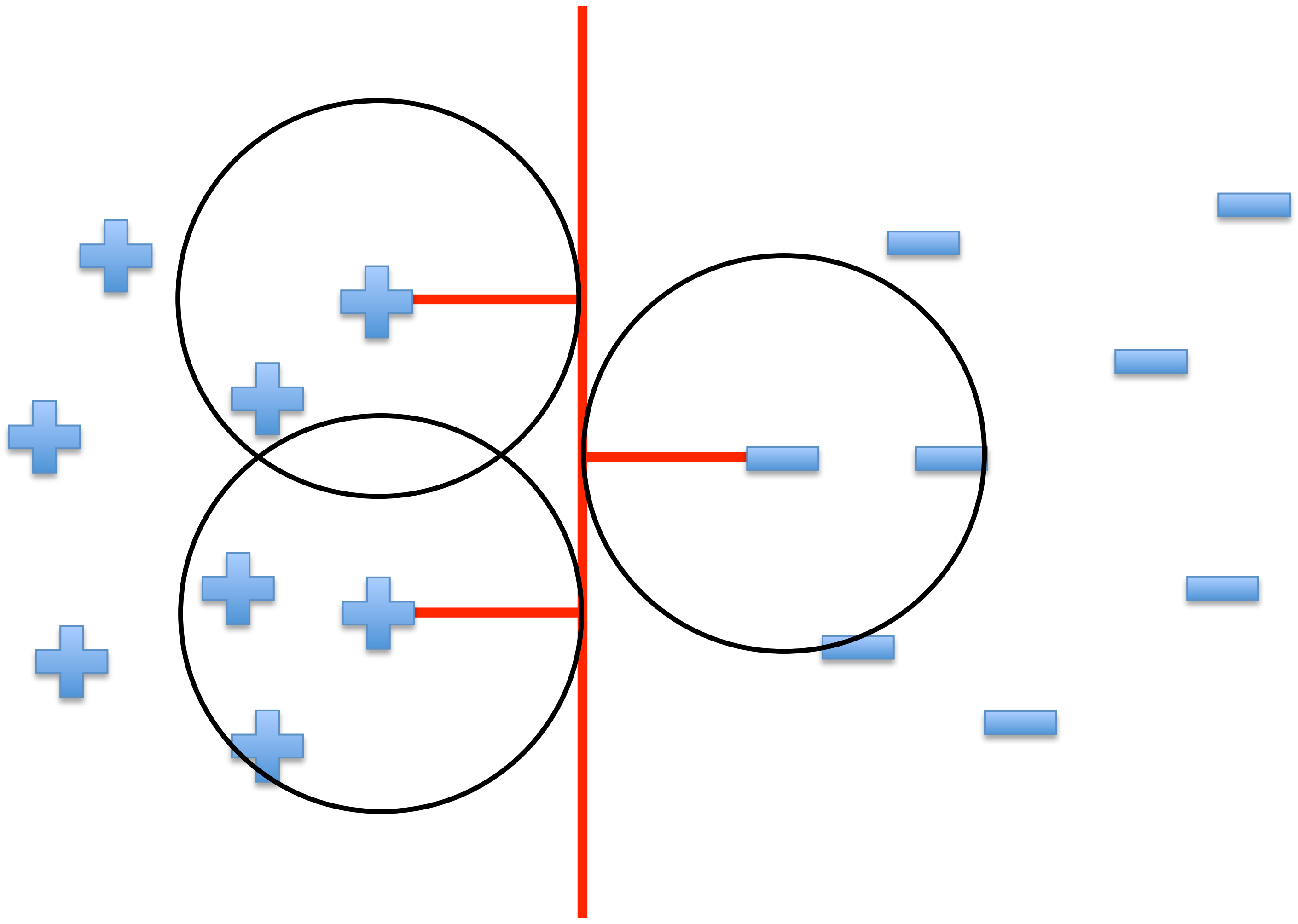
Lots of Noise



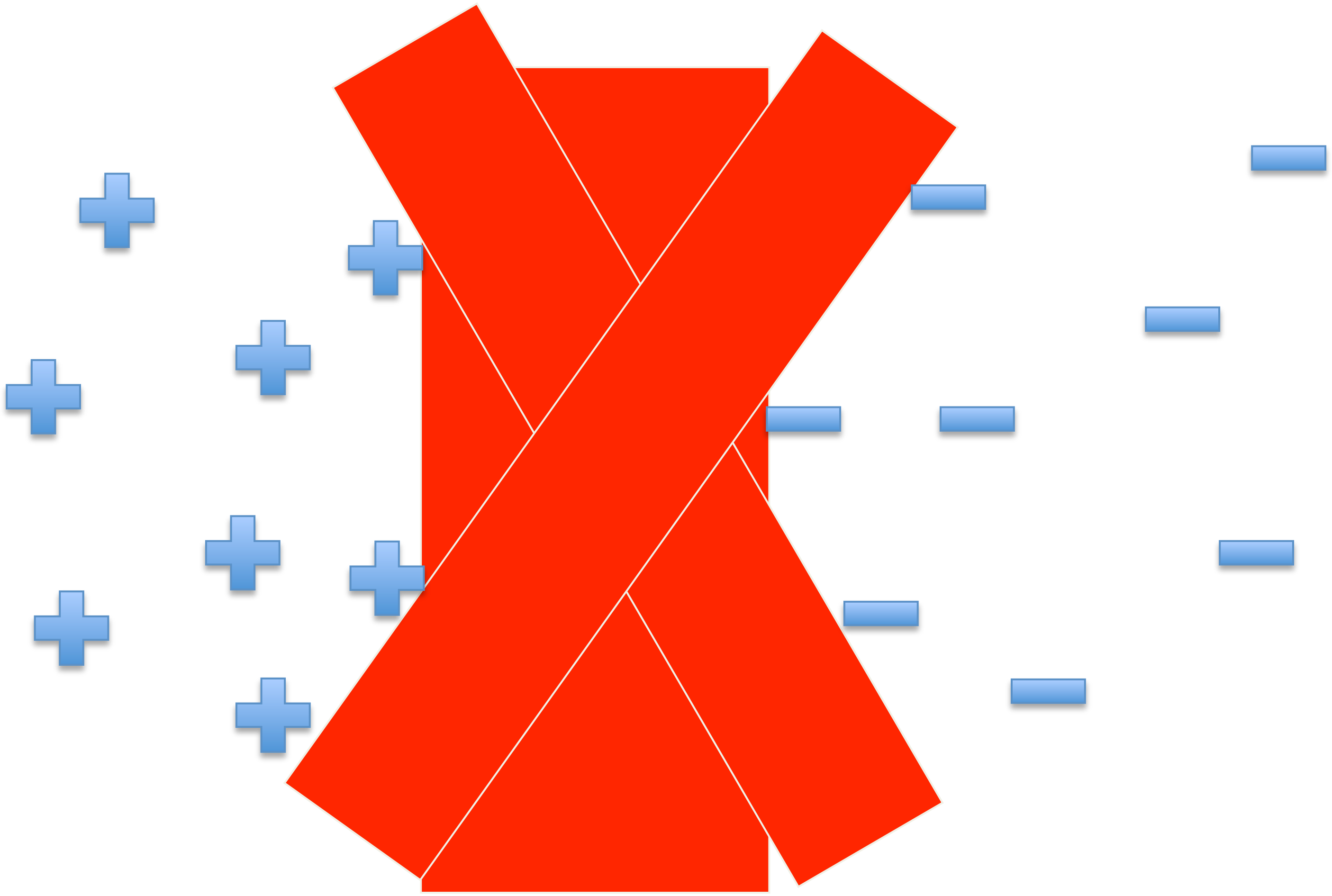
Only One Separator Remains



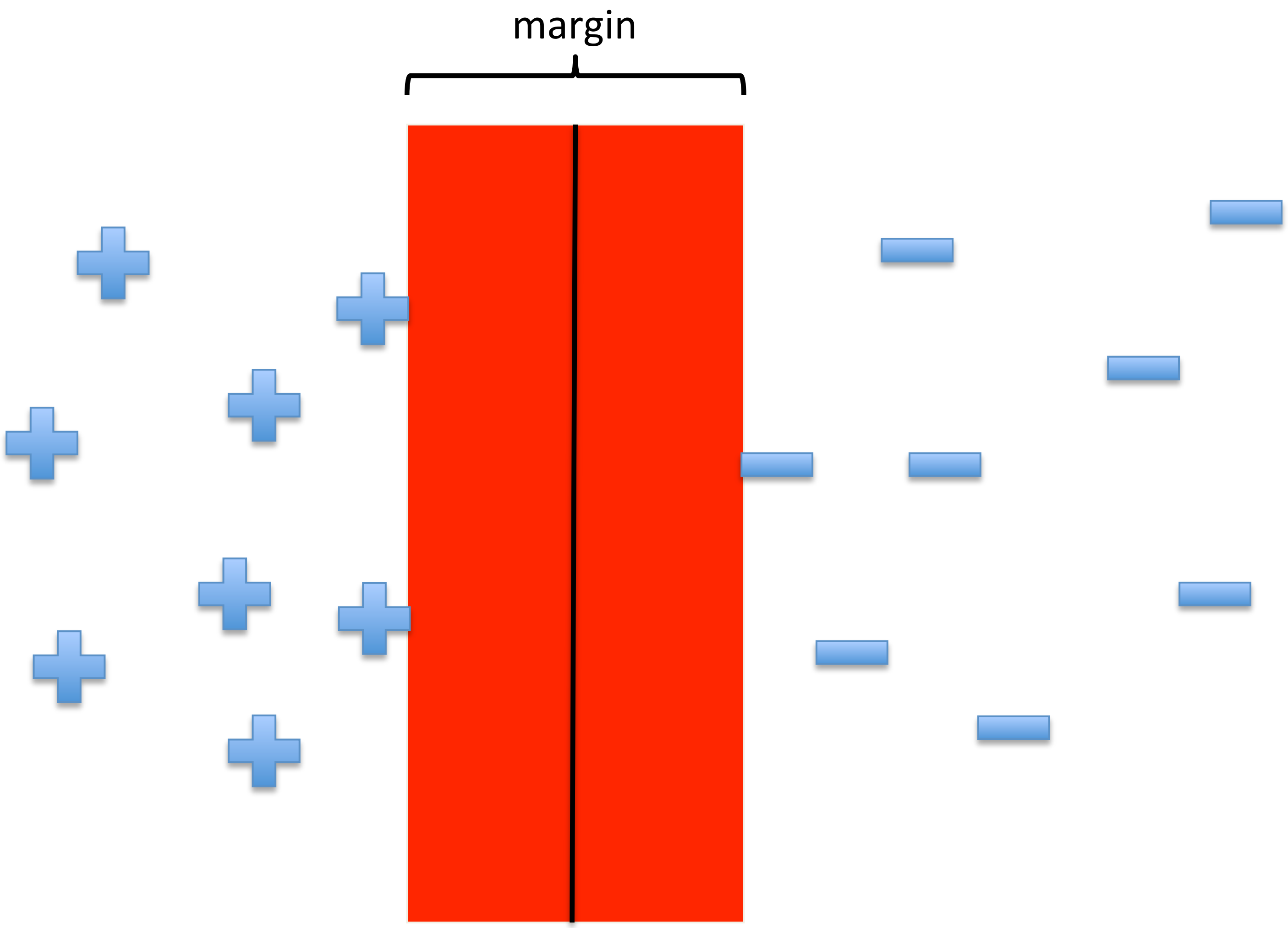
Maximizing the Margin



“Fat” Separators



"Fat" Separators



Why Maximize Margin

Increasing margin reduces *capacity*

- i.e., fewer possible models

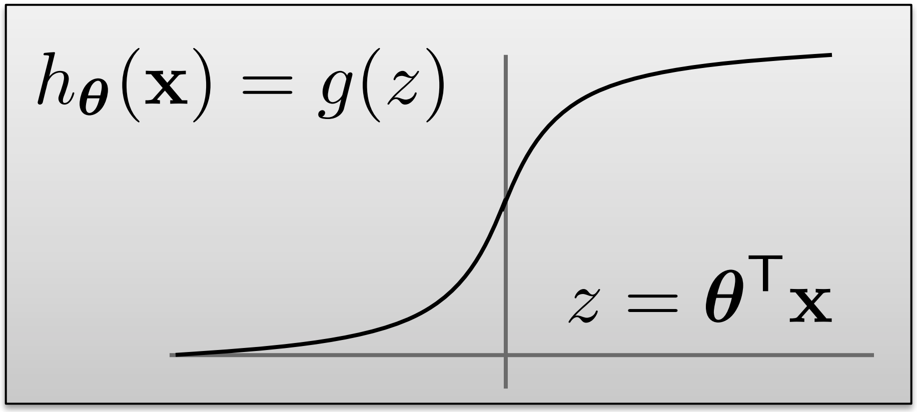
Lesson from Learning Theory:

- If the following holds:
  - H is sufficiently constrained in size
  - and/or the size of the training data set n is large,

  then low training error is likely to be evidence of low generalization error

Alternative View of Logistic Regression

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



If $y = 1$, we want $h_{\boldsymbol{\theta}}(\mathbf{x}) \approx 1, \; \boldsymbol{\theta}^\intercal \mathbf{x} \gg 0$

If $y = 0$, we want $h_{\boldsymbol{\theta}}(\mathbf{x}) \approx 0, \; \boldsymbol{\theta}^\intercal \mathbf{x} \ll 0$

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} [y_i \underbrace{\log h_{\boldsymbol{\theta}}(\mathbf{x}_i)}_{\mathrm{cost}_1(\boldsymbol{\theta}^\intercal \mathbf{x}_i)} + (1-y_i) \underbrace{\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}_i))}_{\mathrm{cost}_0(\boldsymbol{\theta}^\intercal \mathbf{x}_i)}]$$

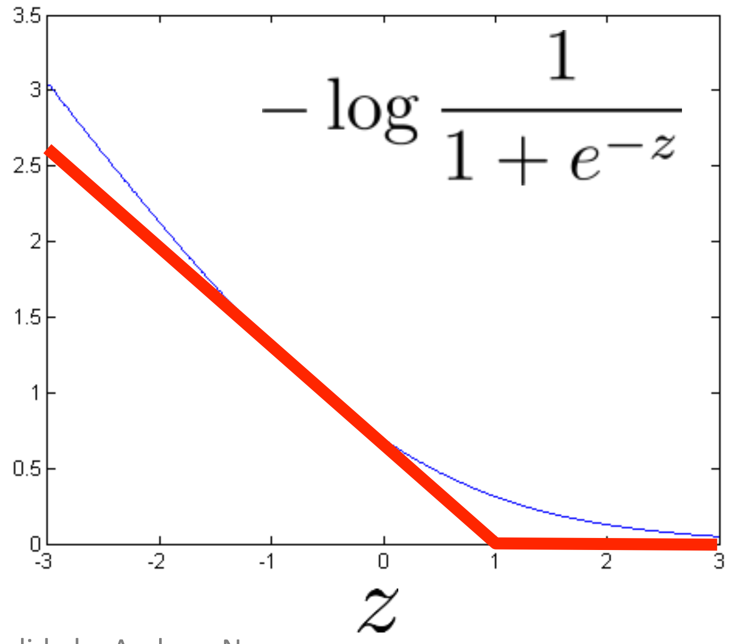
$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Alternate View of Logistic Regression

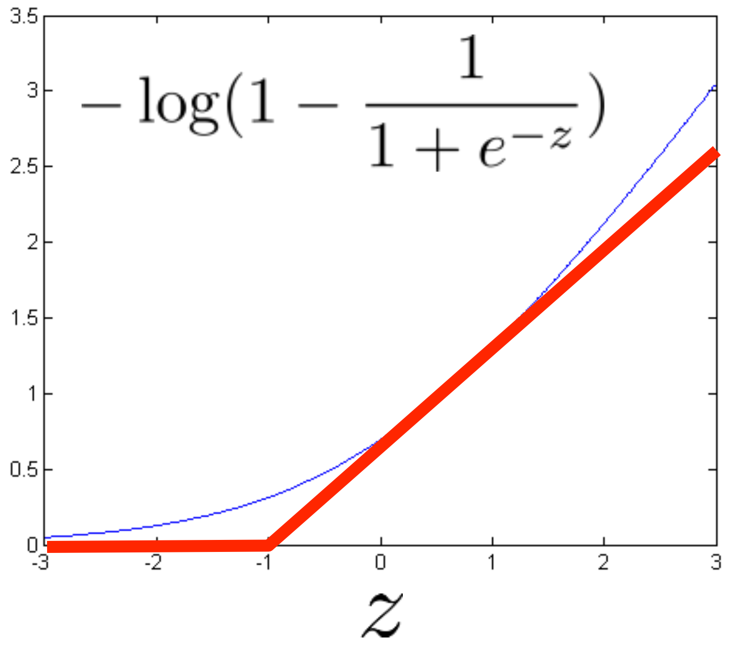
Cost of example: $-y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) - (1 - y_i) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}_i))$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\intercal}\mathbf{x}}} \qquad z = \boldsymbol{\theta}^{\intercal}\mathbf{x}$$

If $y = 1$ (want $\boldsymbol{\theta}^{\intercal}\mathbf{x} \gg 0$):



If $y = 0$ (want $\boldsymbol{\theta}^{\intercal}\mathbf{x} \ll 0$):



Logistic Regression to SVMs

Logistic Regression:

$$\min_{\boldsymbol{\theta}} \; -\sum_{i=1}^{n}[y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) + (1-y_i)\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}_i))] + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

Support Vector Machines:

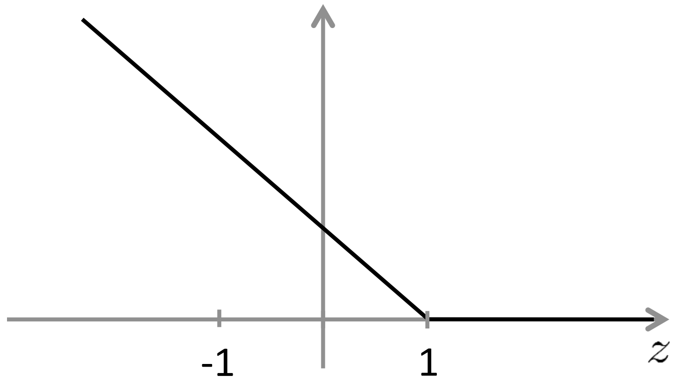
$$\min_{\boldsymbol{\theta}} \; C\sum_{i=1}^{n}[y_i \text{cost}_1(\boldsymbol{\theta}^\intercal \mathbf{x}_i) + (1-y_i)\,\text{cost}_0(\boldsymbol{\theta}^\intercal \mathbf{x}_i)] + \frac{1}{2}\sum_{j=1}^{d}\theta_j^2$$

You can think of C as similar to $\frac{1}{\lambda}$

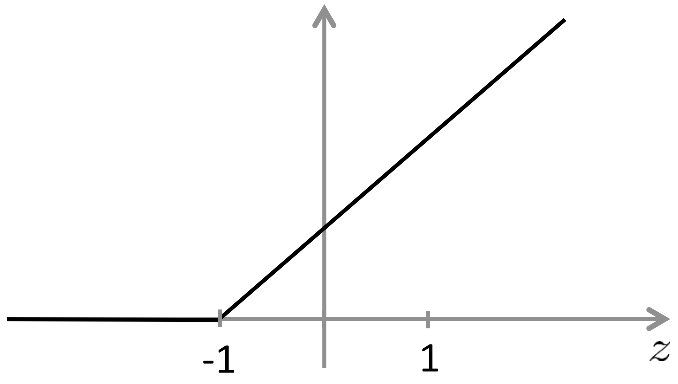
Support Vector Machine

$$\min_{\boldsymbol{\theta}} \; C \sum_{i=1}^{n} \left[y_i \text{cost}_1(\boldsymbol{\theta}^\intercal \mathbf{x}_i) + (1 - y_i) \, \text{cost}_0(\boldsymbol{\theta}^\intercal \mathbf{x}_i) \right] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2$$

If $y = 1$ (want $\boldsymbol{\theta}^\intercal \mathbf{x} \geq 1$):



If $y = 0$ (want $\boldsymbol{\theta}^\intercal \mathbf{x} \leq -1$):



$$\ell_{\text{hinge}}(h(\mathbf{x})) = \max(0, 1 - y \cdot h(\mathbf{x}))$$

Based on slide by Andrew Ng

Support Vector Machine

$$\min_{\boldsymbol{\theta}} \; C \sum_{i=1}^{n} [y_i \text{cost}_1(\boldsymbol{\theta}^\intercal \mathbf{x}_i) + (1 - y_i) \, \text{cost}_0(\boldsymbol{\theta}^\intercal \mathbf{x}_i)] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2$$

y = 1 / 0

y = +1 / -1

with C = 1

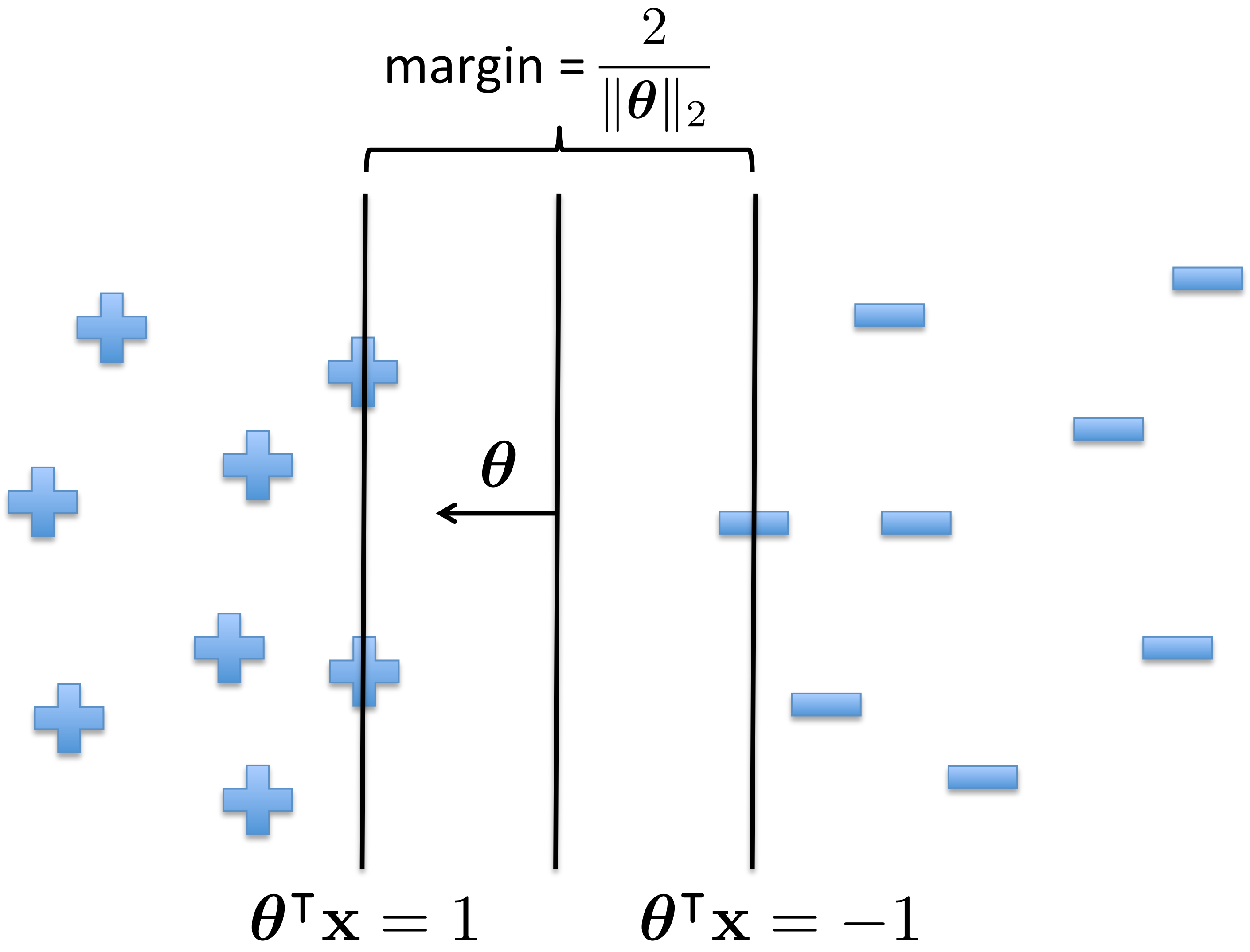
$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2$$

$$\begin{aligned} \text{s.t.} \quad & \boldsymbol{\theta}^\intercal \mathbf{x}_i \geq 1 && \text{if } y_i = 1 \\ & \boldsymbol{\theta}^\intercal \mathbf{x}_i \leq -1 && \text{if } y_i = -1 \end{aligned}$$

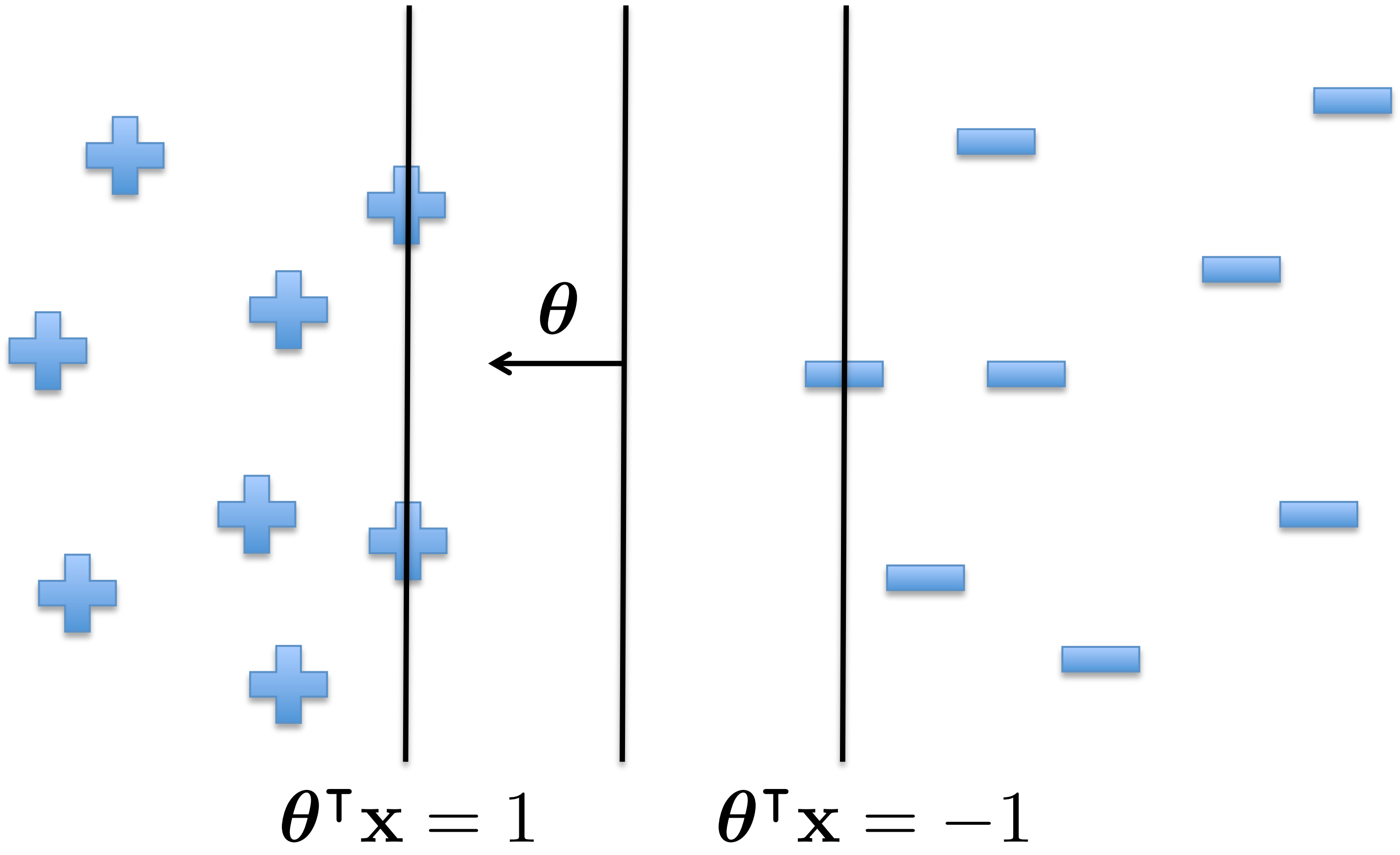
$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2$$

$$\text{s.t.} \quad y_i(\boldsymbol{\theta}^\intercal \mathbf{x}_i) \geq 1$$

Maximum Margin Hyperplane

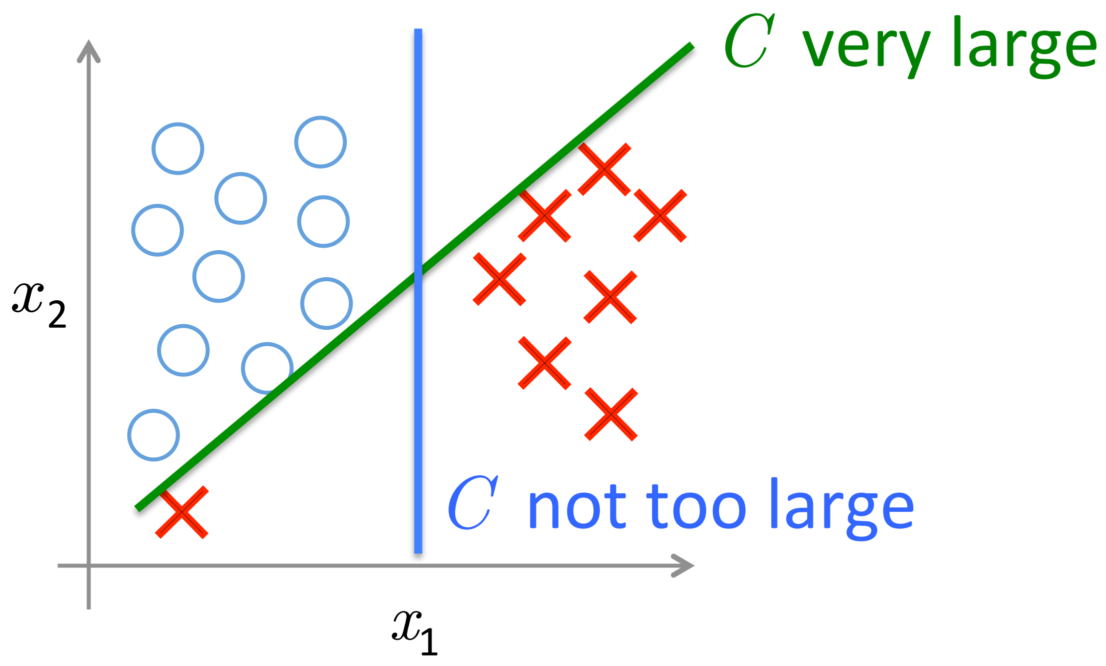

margin = $\frac{2}{\|\boldsymbol{\theta}\|_2}$
$\boldsymbol{\theta}$
$\boldsymbol{\theta}^\intercal \mathbf{x} = 1$
$\boldsymbol{\theta}^\intercal \mathbf{x} = -1$

Support Vectors

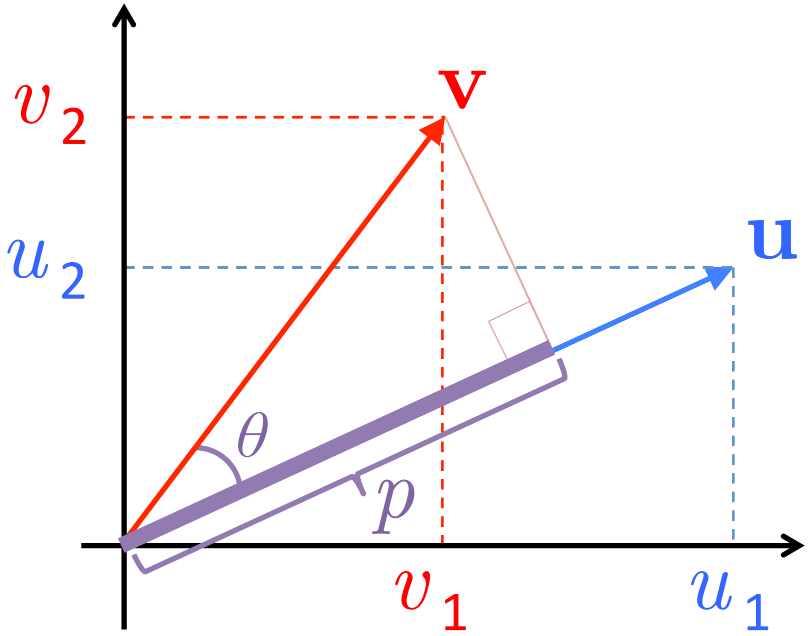


θ
$\theta^\intercal \mathbf{x} = 1$
$\theta^\intercal \mathbf{x} = -1$

Large Margin Classifier in Presence of Outliers



Vector Inner Product



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

$$\begin{aligned} \|\mathbf{u}\|_2 &= \text{length}(\mathbf{u}) \in \mathbb{R} \\ &= \sqrt{u_1^2 + u_2^2} \end{aligned}$$

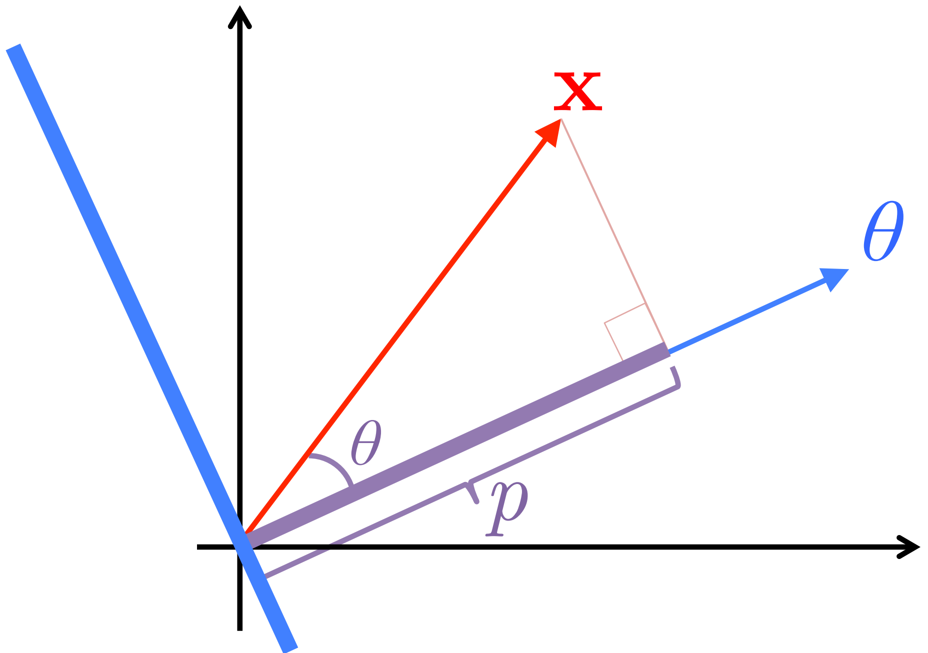
$$\begin{aligned} \mathbf{u}^\intercal \mathbf{v} &= \mathbf{v}^\intercal \mathbf{u} \\ &= u_1 v_1 + u_2 v_2 \\ &= \|\mathbf{u}\|_2 \; \|\mathbf{v}\|_2 \cos\theta \\ &= p\|\mathbf{u}\|_2 \quad \text{where } p = \|\mathbf{v}\|_2 \cos\theta \end{aligned}$$

Understanding the Hyperplane

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

$$\begin{aligned} \text{s.t. } \boldsymbol{\theta}^\mathsf{T}\mathbf{x}_i &\geq 1 && \text{if } y_i = 1 \\ \boldsymbol{\theta}^\mathsf{T}\mathbf{x}_i &\leq -1 && \text{if } y_i = -1 \end{aligned}$$

Assume $\theta_0 = 0$ so that the hyperplane is centered at the origin, and that $d = 2$



$$\boldsymbol{\theta}^\mathsf{T}\mathbf{x} = \|\boldsymbol{\theta}\|_2 \underbrace{\|\mathbf{x}\|_2 \cos\theta}_{p}$$

$$= p\|\boldsymbol{\theta}\|_2$$

Based on example by Andrew Ng

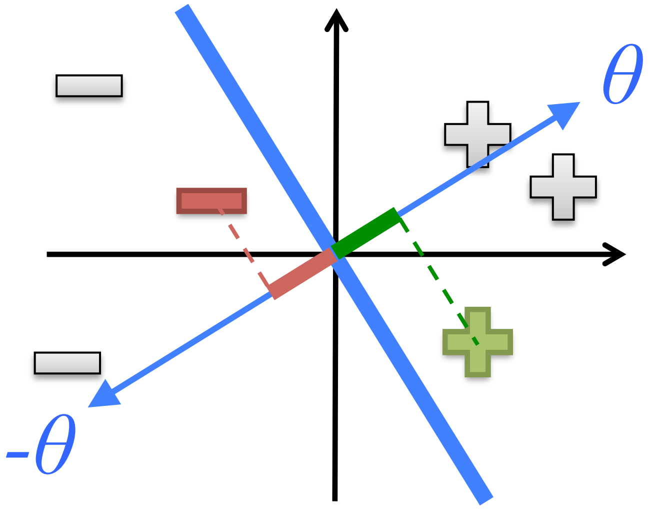
Maximizing the Margin

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

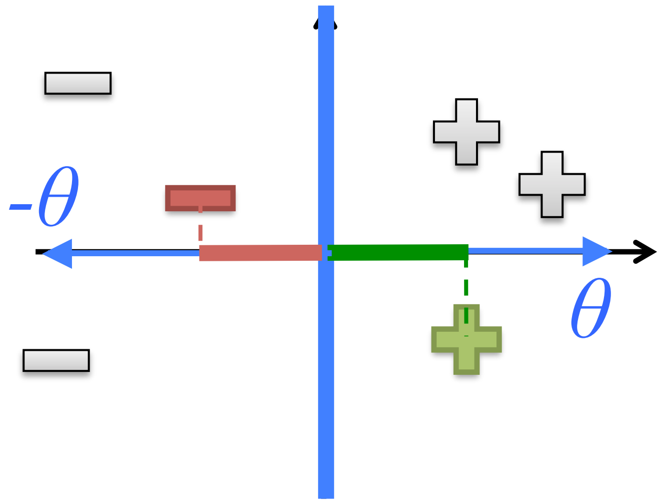
$$\text{s.t. } \boldsymbol{\theta}^\intercal \mathbf{x}_i \geq 1 \quad \text{if } y_i = 1$$
$$\boldsymbol{\theta}^\intercal \mathbf{x}_i \leq -1 \quad \text{if } y_i = -1$$

Assume $\theta_0 = 0$ so that the hyperplane is centered at the origin, and that $d = 2$

Let p_i be the projection of $\mathbf{x}_i$ onto the vector $\boldsymbol{\theta}$



Since p is small, therefore $\|\boldsymbol{\theta}\|_2$ must be large to have $p\|\boldsymbol{\theta}\|_2 \geq 1$ (or ≤ -1)

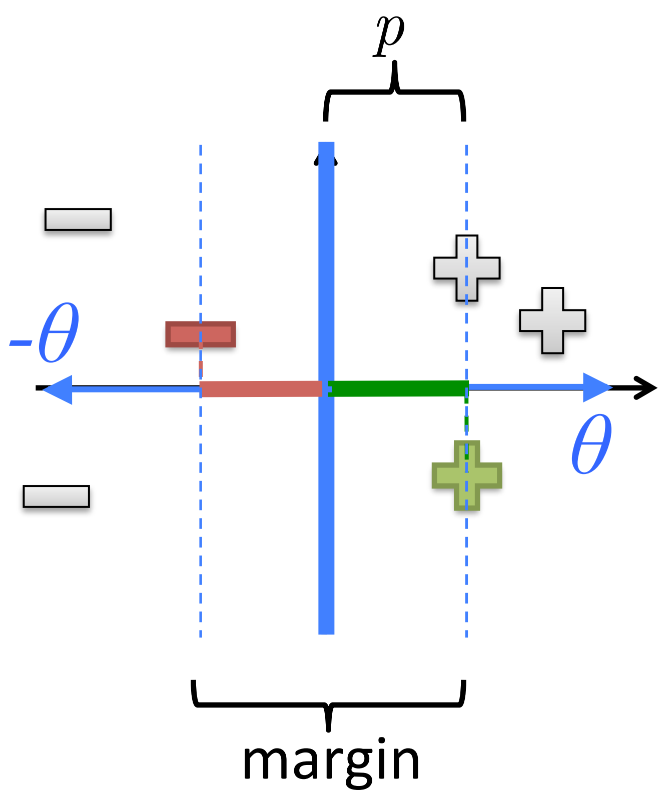


Since p is larger, $\|\boldsymbol{\theta}\|_2$ can be smaller in order to have $p\|\boldsymbol{\theta}\|_2 \geq 1$ (or ≤ -1)

Size of the Margin

For the support vectors, we have $p\|\boldsymbol{\theta}\|_2 = \pm 1$

- p is the length of the projection of the SVs onto $\boldsymbol{\theta}$



Therefore,

$$p = \frac{1}{\|\boldsymbol{\theta}\|_2}$$

$$\text{margin} = 2p = \frac{2}{\|\boldsymbol{\theta}\|_2}$$

The SVM Dual Problem

The primal SVM problem was given as

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

$$\text{s.t. } y_i(\boldsymbol{\theta}^\intercal \mathbf{x}_i) \geq 1 \quad \forall i$$

Can solve it more efficiently by taking the Lagrangian dual

- Duality is a common idea in optimization
- It transforms a difficult optimization problem into a simpler one
- Key idea: introduce slack variables α_i for each constraint
  - α_i indicates how important a particular constraint is to the solution

The SVM Dual Problem

- The Lagrangian is given by

$$L(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \frac{1}{2}\sum_{j=1}^{d}\theta_j^2 - \sum_{i=1}^{n}\alpha_i(y_i\boldsymbol{\theta}^\intercal\mathbf{x} - 1)$$

$$\text{s.t. } \alpha_i \geq 0 \;\; \forall i$$

- We must minimize over $\boldsymbol{\theta}$ and maximize over **α**
- At optimal solution, partials w.r.t $\boldsymbol{\theta}$'s are 0

Solve by a bunch of algebra and calculus ...

and we obtain ...

SVM Dual Representation

Maximize $J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

$$\text{s.t. } \alpha_i \geq 0 \quad \forall i$$

$$\sum_i \alpha_i y_i = 0$$

The decision function is given by

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i \in \mathcal{SV}} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b \right)$$

$$\text{where} \quad b = \frac{1}{|\mathcal{SV}|} \sum_{i \in \mathcal{SV}} \left(y_i - \sum_{j \in \mathcal{SV}} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

Understanding the Dual

Maximize $J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

s.t. $\boxed{\alpha_i \geq 0 \quad \forall i}$

$\boxed{\sum_i \alpha_i y_i = 0}$

Balances between the weight of constraints for different classes

Constraint weights (α_i's) cannot be negative

Understanding the Dual

Maximize $J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}$

$$\text{s.t. } \alpha_i \geq 0 \quad \forall i$$

$$\sum \alpha_i y_i = 0$$

Points with different labels increase the sum

Points with same label decrease the sum

Measures the similarity between points

Intuitively, we should be more careful around points near the margin

Understanding the Dual

Maximize $$J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\text{s.t. } \alpha_i \geq 0 \quad \forall i$$

$$\sum_i \alpha_i y_i = 0$$

In the solution, either:

- $\alpha_i > 0$ and the constraint is tight $(y_i(\boldsymbol{\theta}^\intercal \mathbf{x}_i) = 1)$
  - ➢ point is a support vector
- $\alpha_i = 0$
  - ➢ point is not a support vector

Employing the Solution

- Given the optimal solution **α***, optimal weights are

$$\boldsymbol{\theta}^\star = \sum_{i \in SVs} \alpha_i^\star y_i \mathbf{x}_i$$

  - In this formulation, have *not* added $x_0 = 1$

- Therefore, we can solve one of the SV constraints

$$y_i(\boldsymbol{\theta}^\star \cdot \mathbf{x}_i + \theta_0) = 1$$

  to obtain θ_0

  - Or, more commonly, take the average solution over all support vectors

What if Data Are Not Linearly Separable?

- Cannot find $\boldsymbol{\theta}$ that satisfies $y_i(\boldsymbol{\theta}^\intercal \mathbf{x}_i) \geq 1 \;\; \forall i$

- Introduce slack variables ξ_i

$$y_i(\boldsymbol{\theta}^\intercal \mathbf{x}_i) \geq 1 - \xi_i \;\; \forall i$$

- New problem:

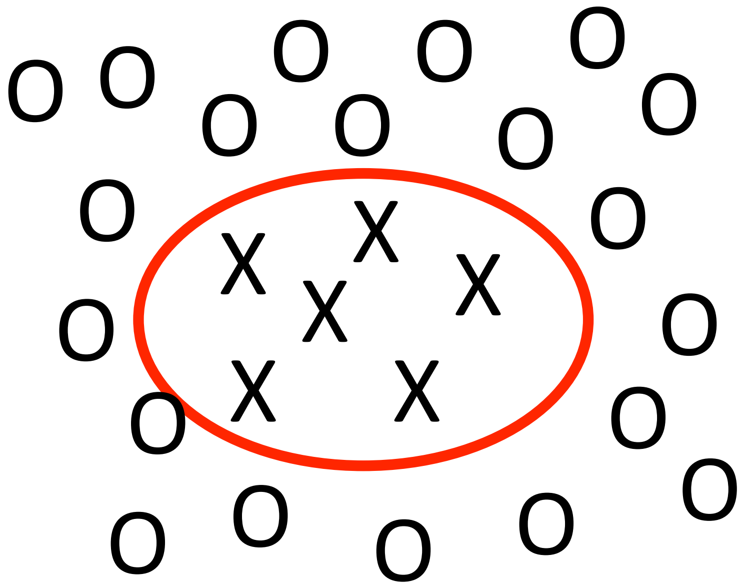
$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 + C \sum_i \xi_i$$

$$\text{s.t. } y_i(\boldsymbol{\theta}^\intercal \mathbf{x}_i) \geq 1 - \xi_i \;\; \forall i$$

Strengths of SVMs

- Good generalization in theory
- Good generalization in practice
- Work well with few training instances
- Find globally best model
- Efficient algorithms
- Amenable to the kernel trick …

What if Surface is Non-Linear?



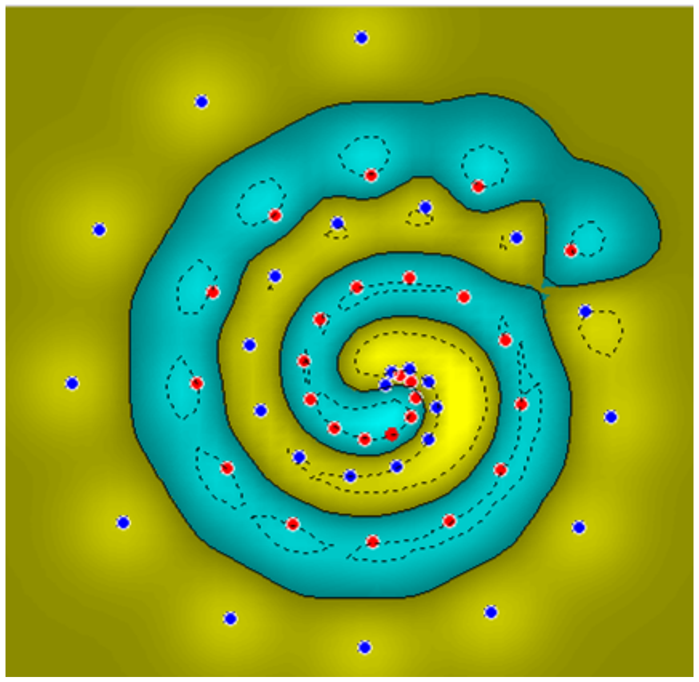
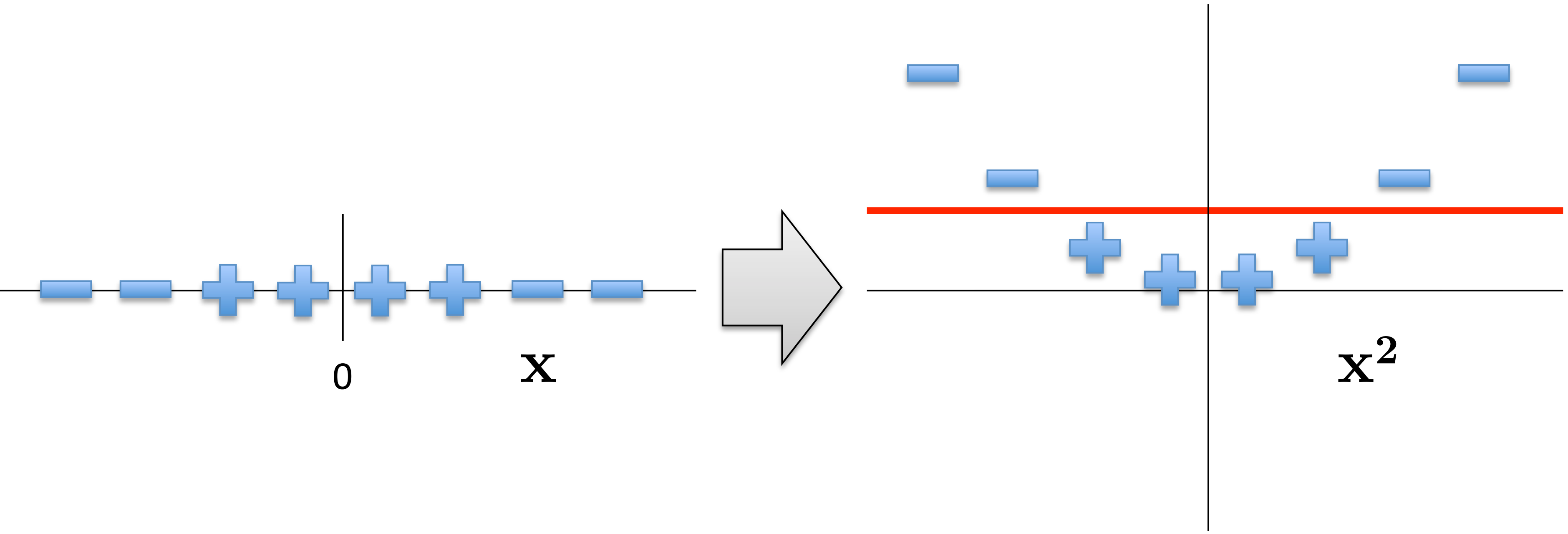


Image from http://www.atrandomresearch.com/iclass/

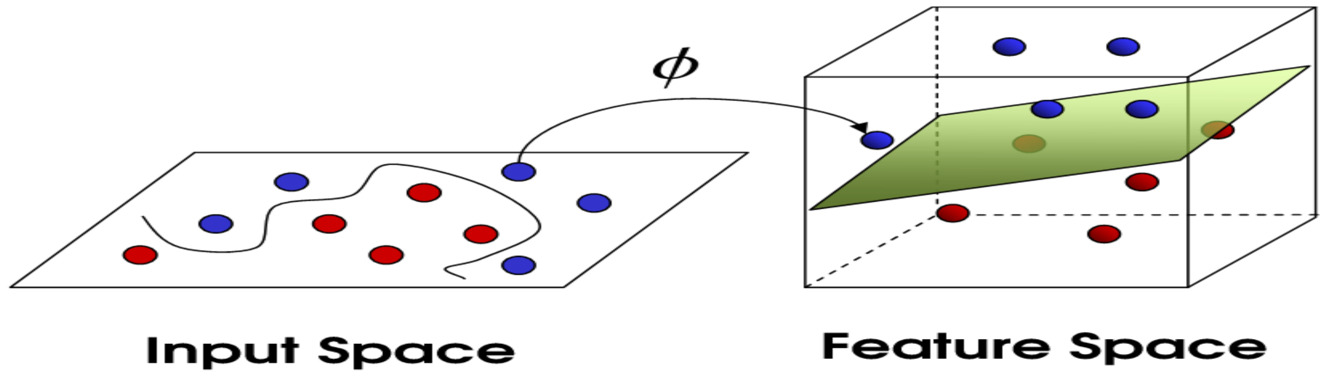
Kernel Methods

Making the Non-Linear Linear

When Linear Separators Fail



Mapping into a New Feature Space



$$\Phi : \mathcal{X} \mapsto \hat{\mathcal{X}} = \Phi(\mathbf{x})$$

- For example, with $\mathbf{x}_i \in \mathbb{R}^2$

$$\Phi([x_{i1}, x_{i2}]) = [x_{i1}, x_{i2}, x_{i1}x_{i2}, x_{i1}^2, x_{i2}^2]$$

- Rather than run SVM on $\mathbf{x}_i$, run it on $\Phi(\mathbf{x}_i)$
  - Find non-linear separator in input space

- What if $\Phi(\mathbf{x}_i)$ is really big?
- Use kernels to compute it implicitly!

Image from http://web.engr.oregonstate.edu/ ~afern/classes/cs534/

Kernels

- Find kernel K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

- Computing $K(\mathbf{x}_i, \mathbf{x}_j)$ should be efficient, much more so than computing $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$

- Use $K(\mathbf{x}_i, \mathbf{x}_j)$ in SVM algorithm rather than $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- Remarkably, this is possible!

The Polynomial Kernel

Let $\mathbf{x}_i = [x_{i1}, x_{i2}]$ and $\mathbf{x}_j = [x_{j1}, x_{j2}]$

Consider the following function:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2 \\ &= (x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= (x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}) \\ &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \end{aligned}$$

where

$$\begin{aligned} \Phi(\mathbf{x}_i) &= [x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}] \\ \Phi(\mathbf{x}_j) &= [x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}] \end{aligned}$$

The Polynomial Kernel

- Given by $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d$
  - $\Phi(\mathbf{x})$ contains all monomials of degree d

- Useful in visual pattern recognition
  - Example:
    - 16x16 pixel image
    - 10^{10} monomials of degree 5
    - Never explicitly compute $\Phi(\mathbf{x})$!

- Variation: $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d$
  - Adds all lower-order monomials (degrees 1,...,d)!

The Kernel Trick

"Given an algorithm which is formulated in terms of a positive definite kernel K_1, one can construct an alternative algorithm by replacing K_1 with another positive definite kernel K_2"

- SVMs can use the kernel trick

Incorporating Kernels into SVM

$$J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \cancel{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}$$

$$\Downarrow$$

$$J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$\text{s.t.} \;\; a_i \geq 0 \;\; \forall i$$

$$\sum_i \alpha_i y_i = 0$$

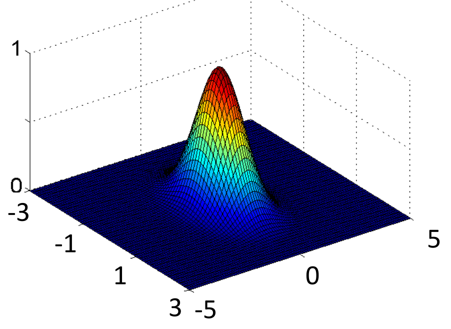
The Gaussian Kernel

- Also called Radial Basis Function (RBF) kernel

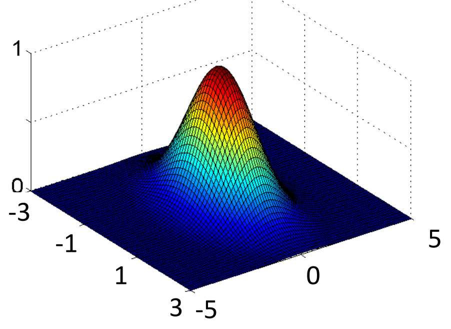
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

  - Has value 1 when $\mathbf{x}_i = \mathbf{x}_j$
  - Value falls off to 0 with increasing distance
  - Note: Need to do feature scaling <u>before</u> using Gaussian Kernel

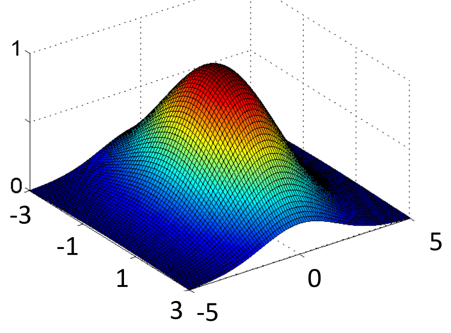
$\sigma^2 = 0.5$



$\sigma^2 = 1$



$\sigma^2 = 3$

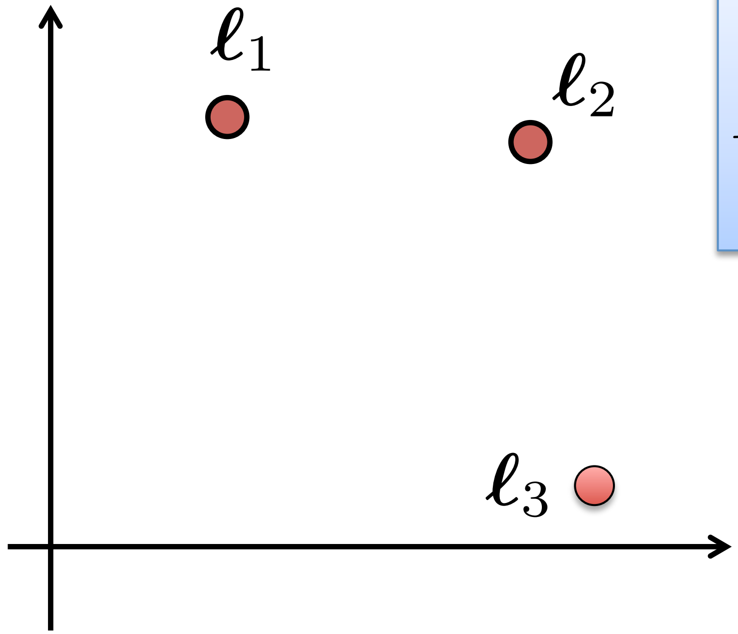


lower bias,
higher variance

$\longleftrightarrow$

higher bias,
lower variance

Gaussian Kernel Example



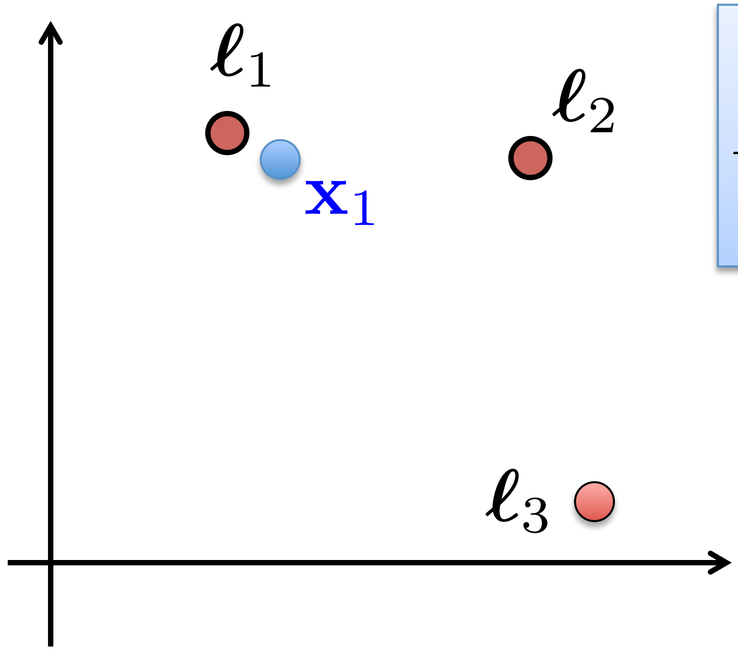
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Imagine we've learned that:

$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

Gaussian Kernel Example



$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Imagine we've learned that:

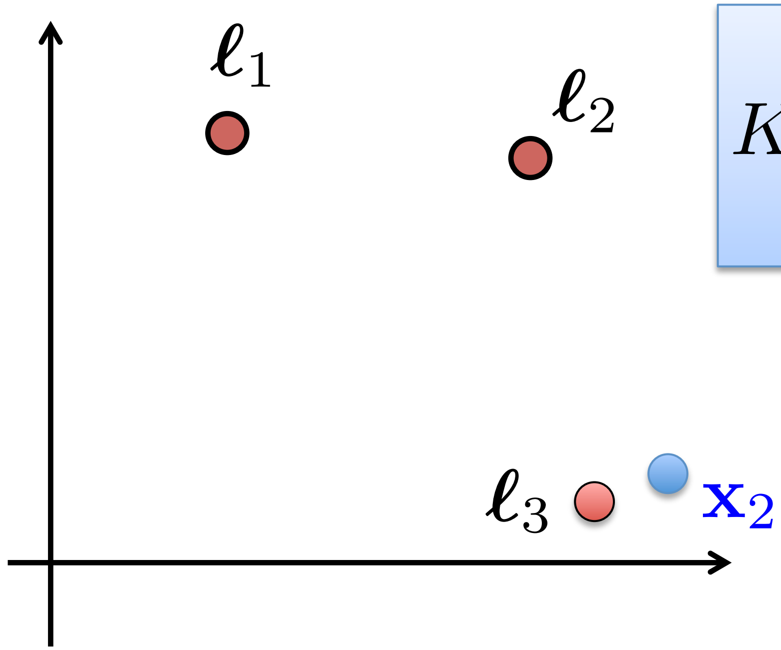
$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

- For $\mathbf{x}_1$, we have $K(\mathbf{x}_1, \ell_1) \approx 1$, other similarities ≈ 0

$$\begin{aligned} &\theta_0 + \theta_1(1) + \theta_2(0) + \theta_3(0) \\ &= -0.5 + 1(1) + 1(0) + 0(0) \\ &= 0.5 \geq 0 \end{aligned}$$, so predict +1

Gaussian Kernel Example



$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2} \right)$$

Imagine we've learned that:

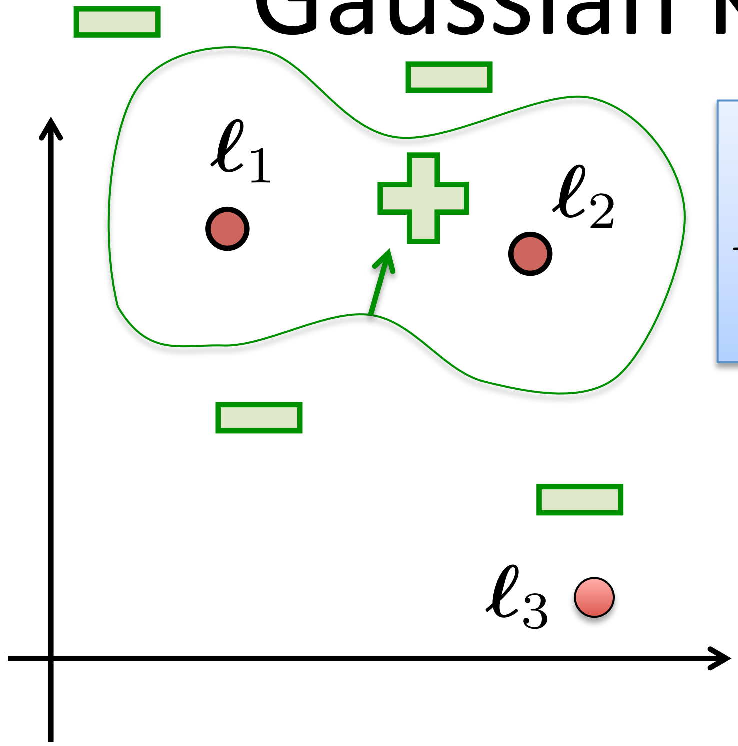
$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

- For $\mathbf{x}_2$, we have $K(\mathbf{x}_2, \ell_3) \approx 1$, other similarities ≈ 0

$$\theta_0 + \theta_1(0) + \theta_2(0) + \theta_3(1)$$
$$= -0.5 + 1(0) + 1(0) + 0(1)$$
$$= -0.5 < 0$$, so predict -1

Gaussian Kernel Example



$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Imagine we've learned that:

$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

Rough sketch of decision surface

Other Kernels

- Sigmoid Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\alpha \mathbf{x}_i^\mathsf{T} \mathbf{x}_j + c\right)$$

  – Neural networks use sigmoid as activation function
  – SVM with a sigmoid kernel is equivalent to 2-layer perceptron

- Cosine Similarity Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^\mathsf{T} \mathbf{x}_j}{\|\mathbf{x}_i\| \; \|\mathbf{x}_j\|}$$

  – Popular choice for measuring similarity of text documents
  – L_2 norm projects vectors onto the unit sphere; their dot product is the cosine of the angle between the vectors

Other Kernels

- Chi-squared Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \sum_k \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}}\right)$$

  - Widely used in computer vision applications
  - Chi-squared measures distance between probability distributions
  - Data is assumed to be non-negative, often with L_1 norm of 1

- String kernels
- Tree kernels
- Graph kernels

An Aside: The Math Behind Kernels

What does it *mean* to be a kernel?

- $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ for some Φ

What does it *take* to be a kernel?

- The Gram matrix $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$
  - Symmetric matrix
  - Positive semi-definite matrix:

    $\mathbf{z}^{\mathsf{T}}\mathrm{G}\mathbf{z} \geq 0$ for every non-zero vector $\mathbf{z} \in \mathbb{R}^n$

Establishing “kernel-hood” from first principles is non-trivial

A Few Good Kernels...

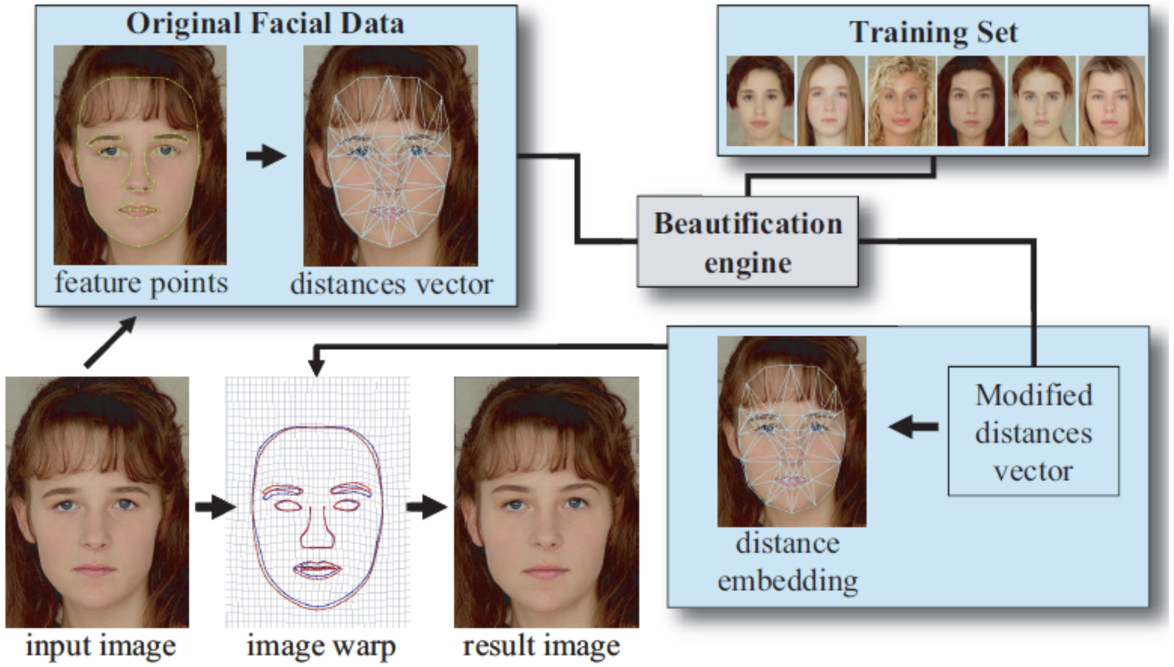
- Linear Kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- Polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$
  - c ≥ 0 trades off influence of lower order terms
- Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$
- Sigmoid kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\alpha \mathbf{x}_i^\intercal \mathbf{x}_j + c\right)$

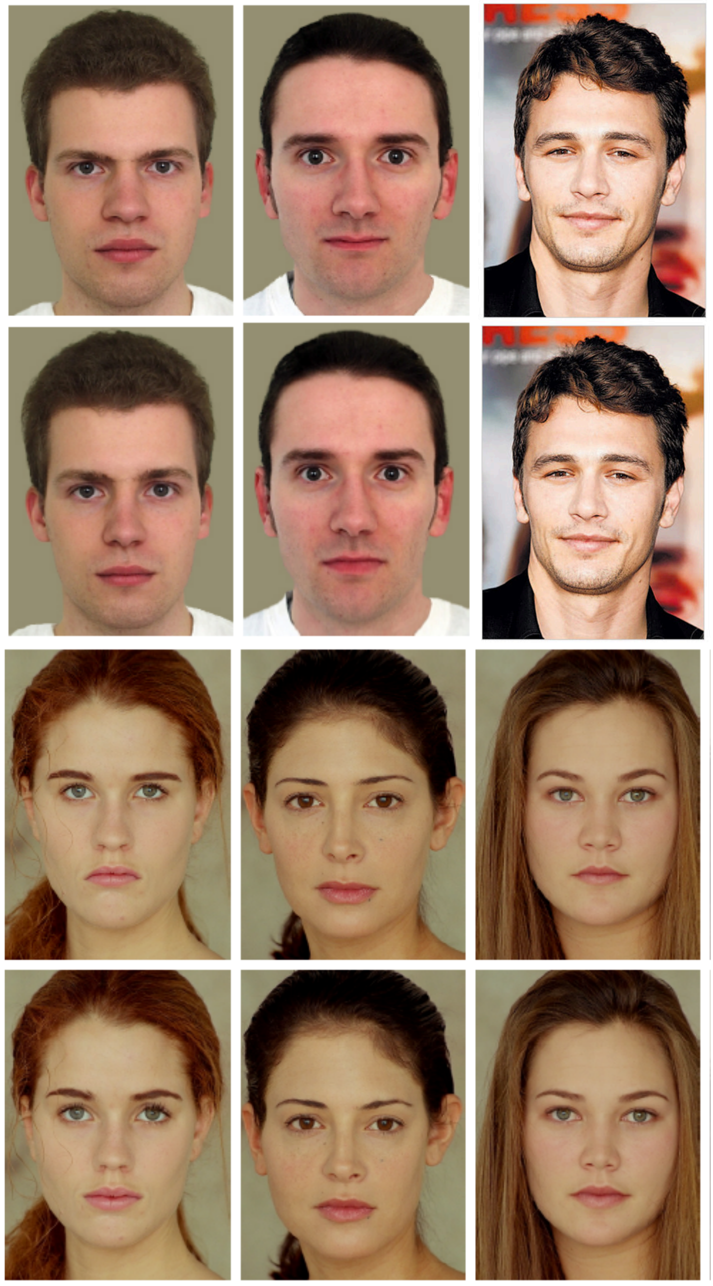
Many more...

- Cosine similarity kernel
- Chi-squared kernel
- String/tree/graph/wavelet/etc kernels

Application: Automatic Photo Retouching

(Leyvand et al., 2008)





Practical Advice for Applying SVMs

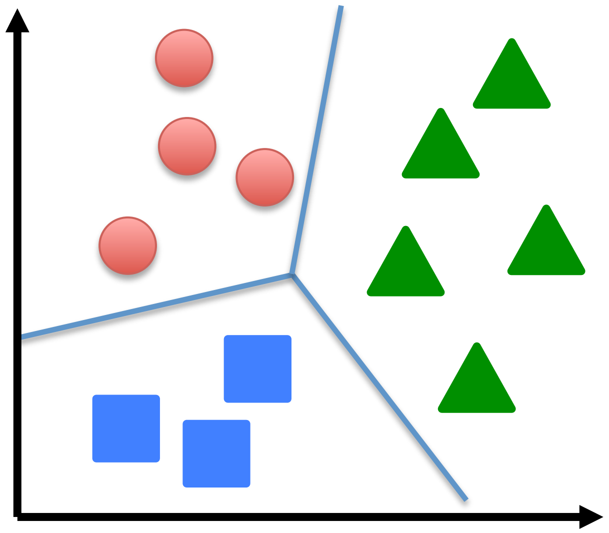
- Use SVM software package to solve for parameters
  - e.g., SVMlight, libsvm, cvx (fast!), etc.

- Need to specify:
  - Choice of parameter C
  - Choice of kernel function
    - Associated kernel parameters

e.g., $$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2} \right)$$

Multi-Class Classification with SVMs



$y \in \{1, \ldots, K\}$

- Many SVM packages already have multi-class classification built in
- Otherwise, use one-vs-rest
  - Train K SVMs, each picks out one class from rest, yielding $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)}$
  - Predict class i with largest $(\boldsymbol{\theta}^{(i)})^\intercal \mathbf{x}$

SVMs vs Logistic Regression
(Advice from Andrew Ng)

n = # training examples d = # features

If d is large (relative to n) (e.g., $d > n$ with d = 10,000, n = 10-1,000)

- Use logistic regression or SVM with a linear kernel

If d is small (up to 1,000), n is intermediate (up to 10,000)

- Use SVM with Gaussian kernel

If d is small (up to 1,000), n is large (50,000+)

- Create/add more features, then use logistic regression or SVM without a kernel

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

Other SVM Variations

- nu SVM
  - nu parameter controls:
    - Fraction of support vectors (lower bound) and misclassification rate (upper bound)
    - E.g., $\nu = 0.05$ guarantees that ≥ 5% of training points are SVs and training error rate is ≤ 5%
  - Harder to optimize than C-SVM and not as scalable
- SVMs for regression
- One-class SVMs
- SVMs for clustering

  ...

Conclusion

- SVMs find optimal linear separator
- The kernel trick makes SVMs learn non-linear decision surfaces

- Strength of SVMs:
  - Good theoretical and empirical performance
  - Supports many types of kernels

- Disadvantages of SVMs:
  - "Slow" to train/predict for huge data sets (but relatively fast!)
  - Need to choose the kernel (and tune its parameters)