

Support Vector Machines and Kernels

Doing *Really* Well with
Linear Decision Surfaces

Adapted from slides by Tim Oates
Cognition, Robotics, and Learning (CORAL) Lab
University of Maryland Baltimore County

Outline

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

Supervised ML = Prediction

- Given training instances (x, y)
- Learn a model f
- Such that $f(x) = y$
- Use f to predict y for new x
- Many variations on this basic theme

Why might predictions be wrong?

- True Non-Determinism
 - Flip a biased coin
 - $p(\text{heads}) = \theta$
 - 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

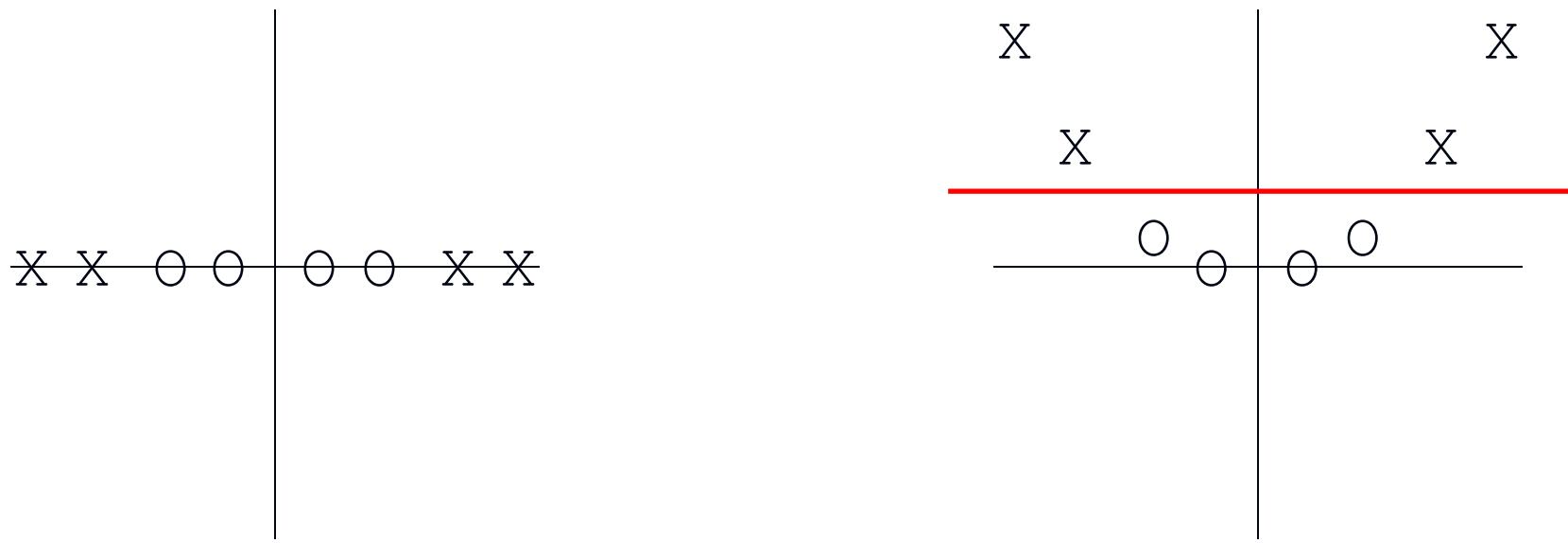
- Something needed to predict y is missing from observation x
- N-bit parity problem
 - x contains $N-1$ bits (hard PO)
 - x contains N bits but learner ignores some of them (soft PO)

Why might predictions be wrong?

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

Representational Bias

- Having the right features (x) is crucial



Support Vector Machines

Doing *Really* Well with Linear
Decision Surfaces

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

Linear Separators

■ Training instances

- $x \in \Re^n$
- $y \in \{-1, 1\}$

■ $w \in \Re^n$

■ $b \in \Re$

■ Hyperplane

- $\langle w, x \rangle + b = 0$
- $w_1x_1 + w_2x_2 \dots + w_nx_n + b = 0$

■ Decision function

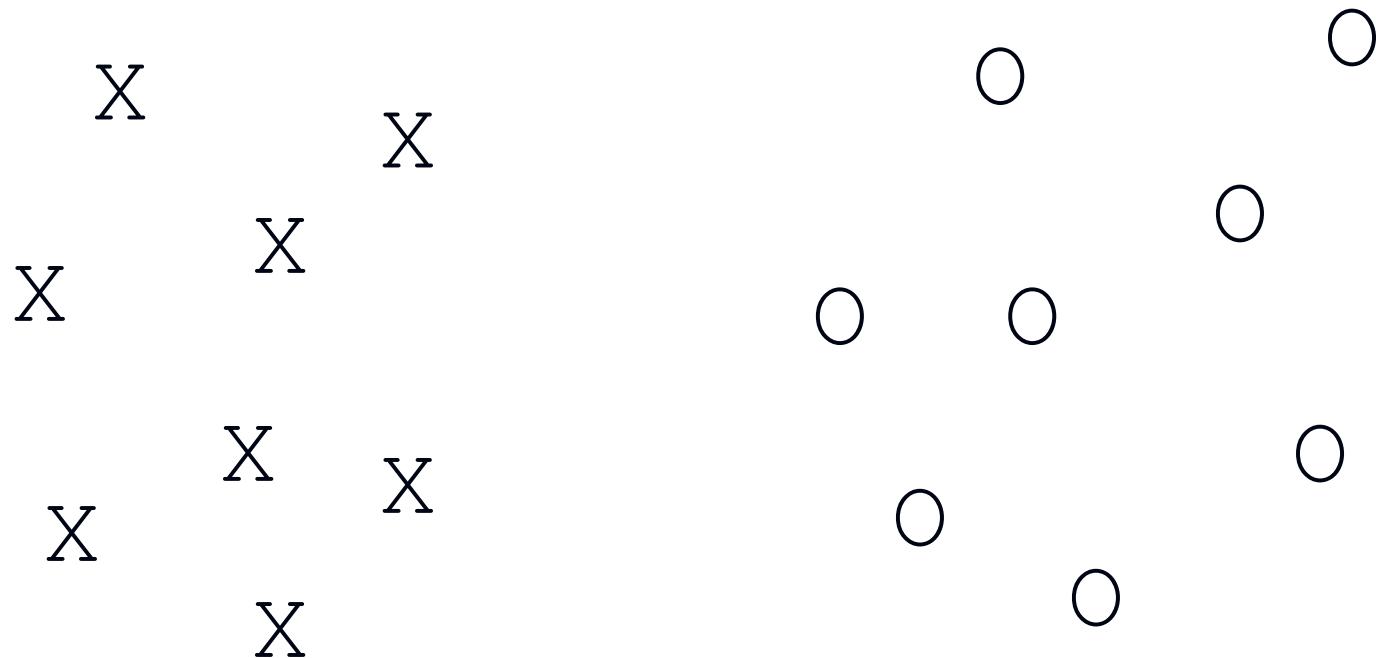
- $f(x) = \text{sign}(\langle w, x \rangle + b)$

Math Review

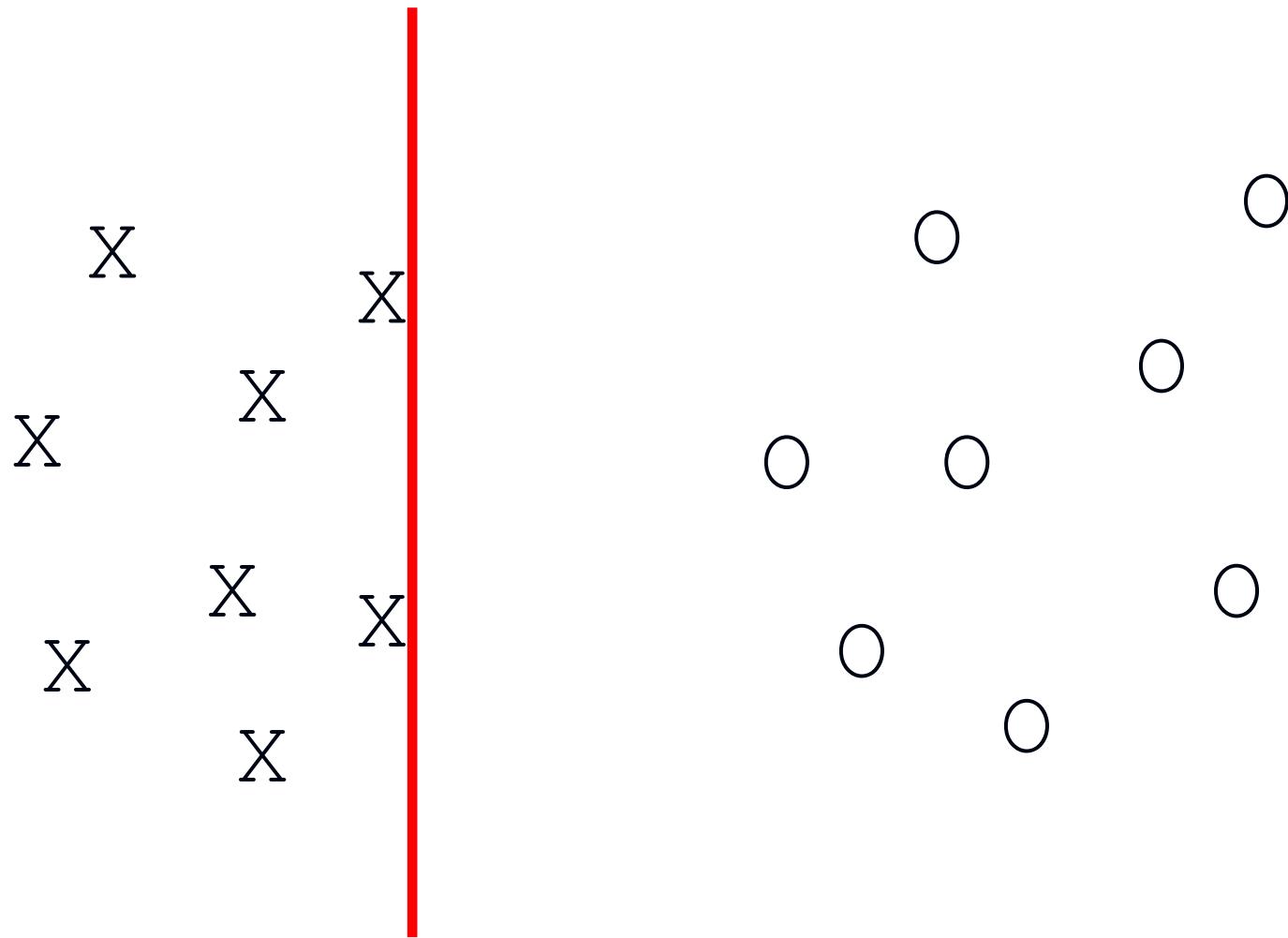
Inner (dot) product:

$$\begin{aligned}\langle a, b \rangle &= a \cdot b = \sum a_i * b_i \\ &= a_1b_1 + a_2b_2 + \dots + a_nb_n\end{aligned}$$

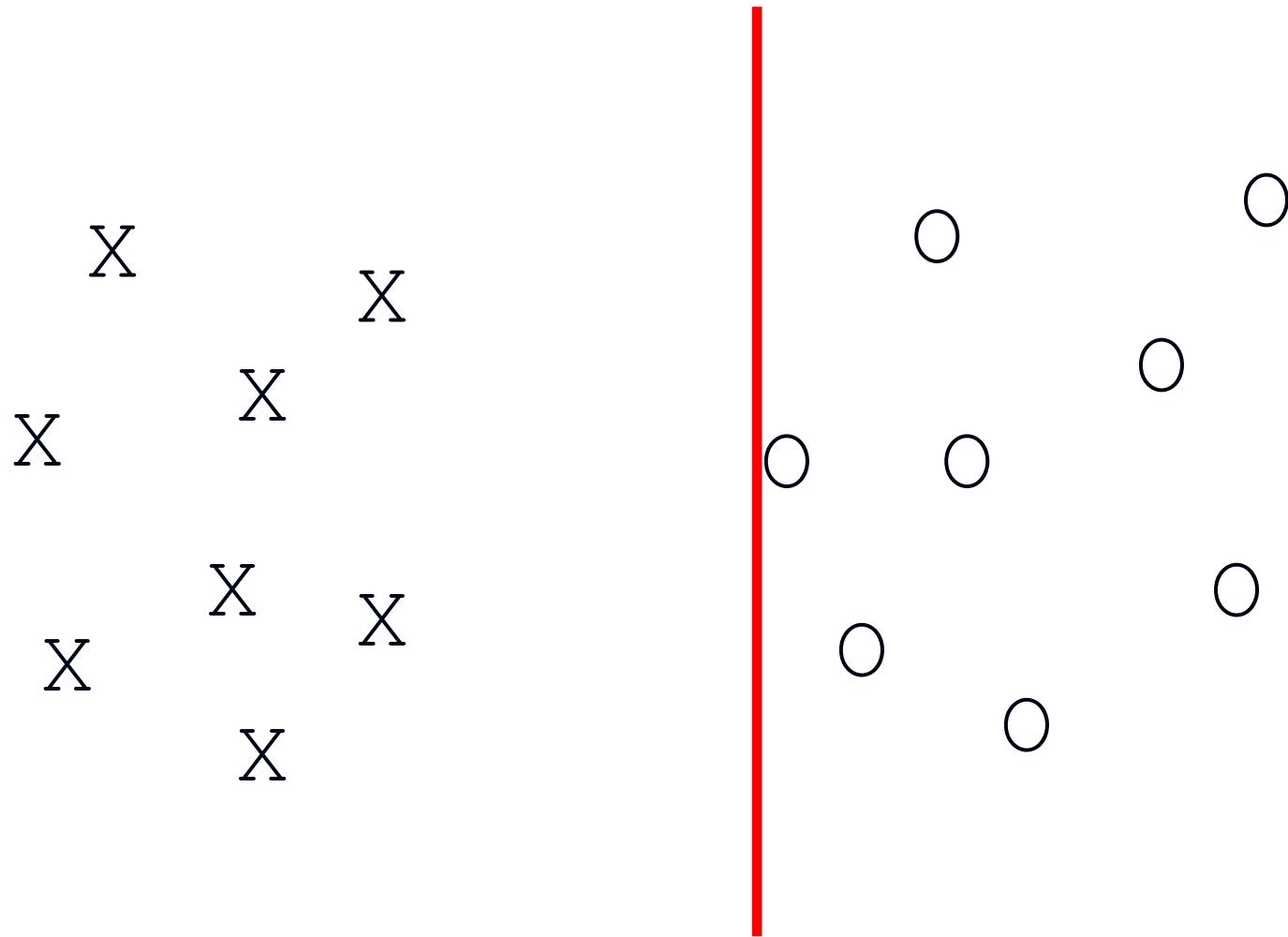
Intuitions



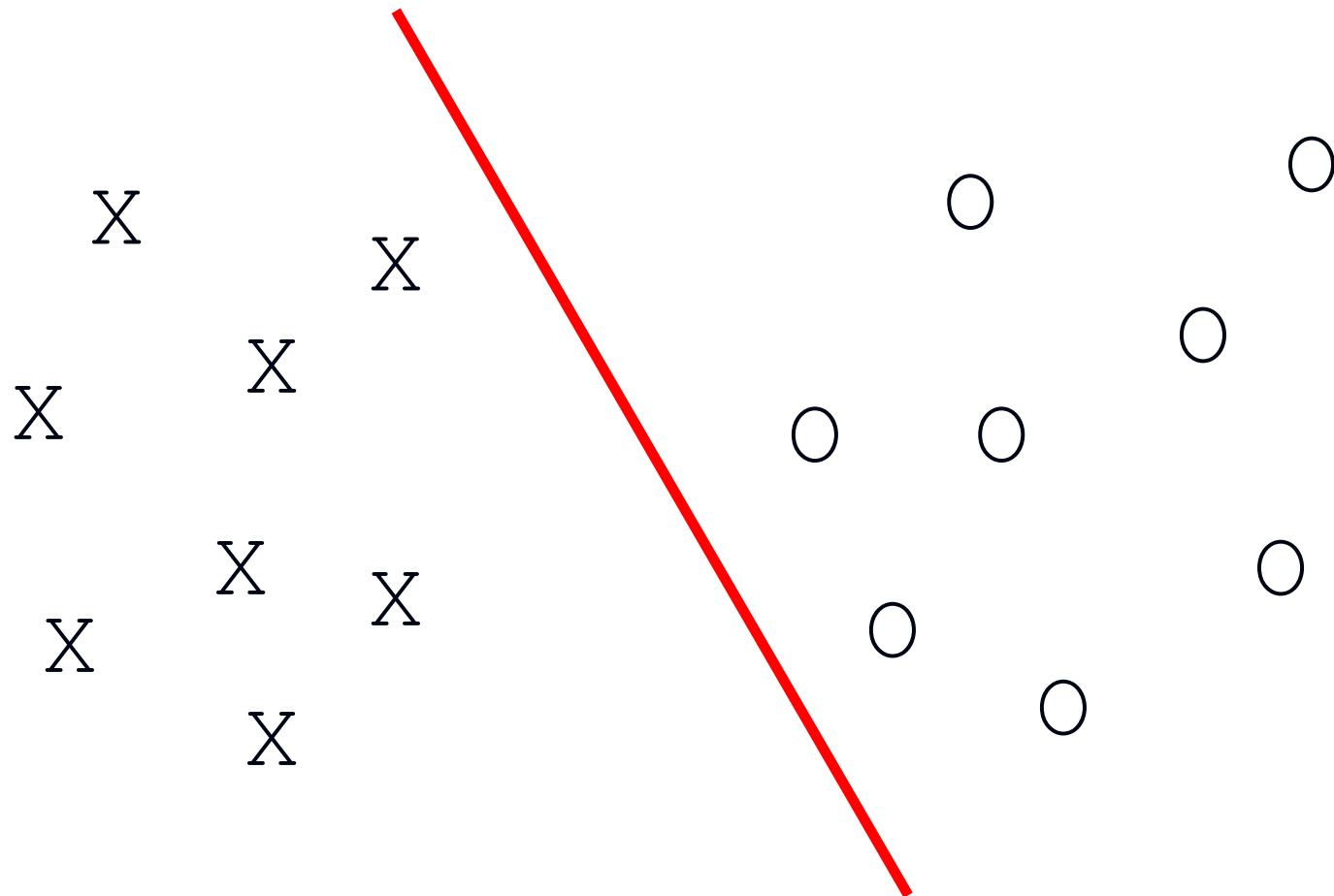
Intuitions



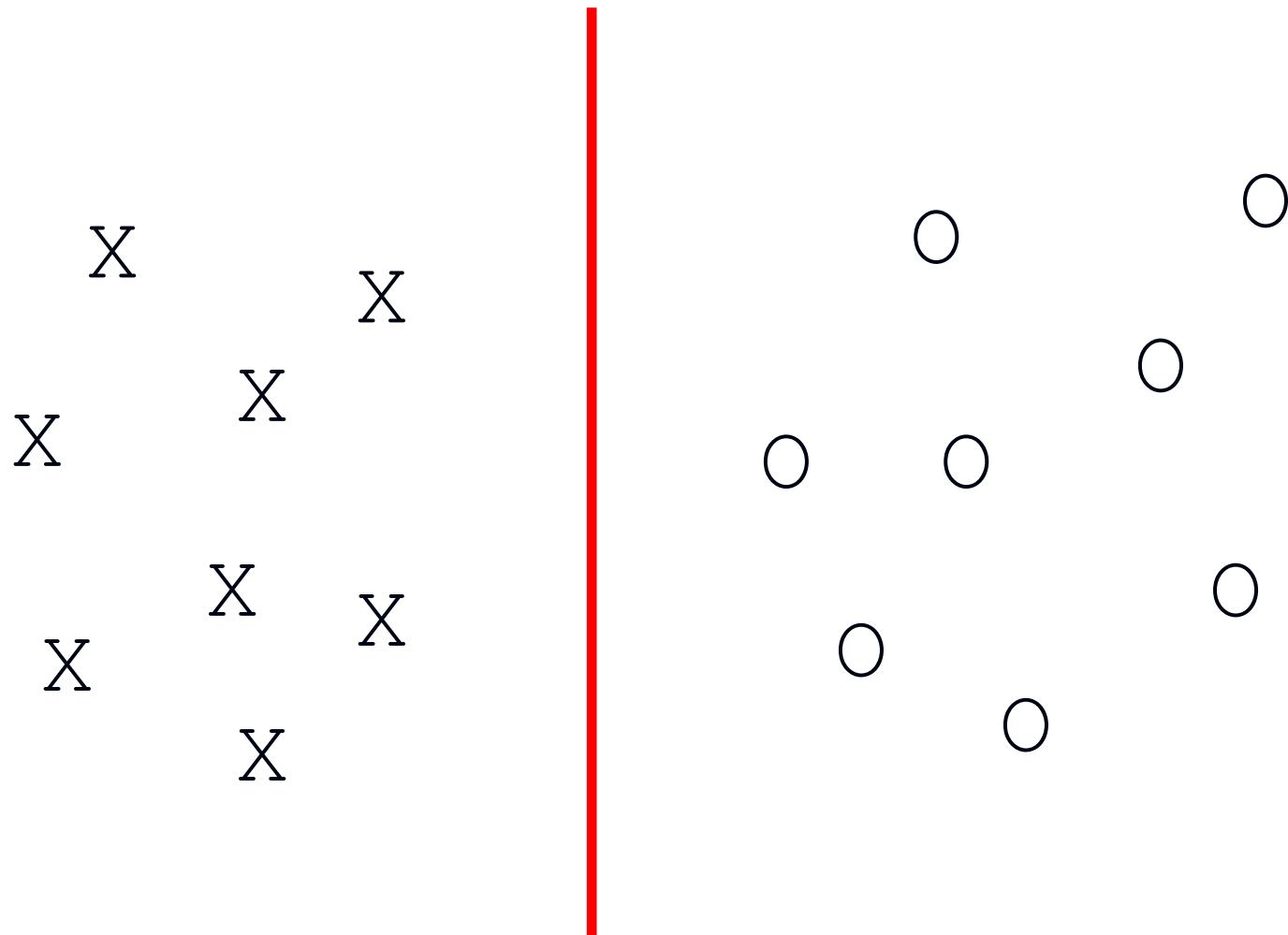
Intuitions



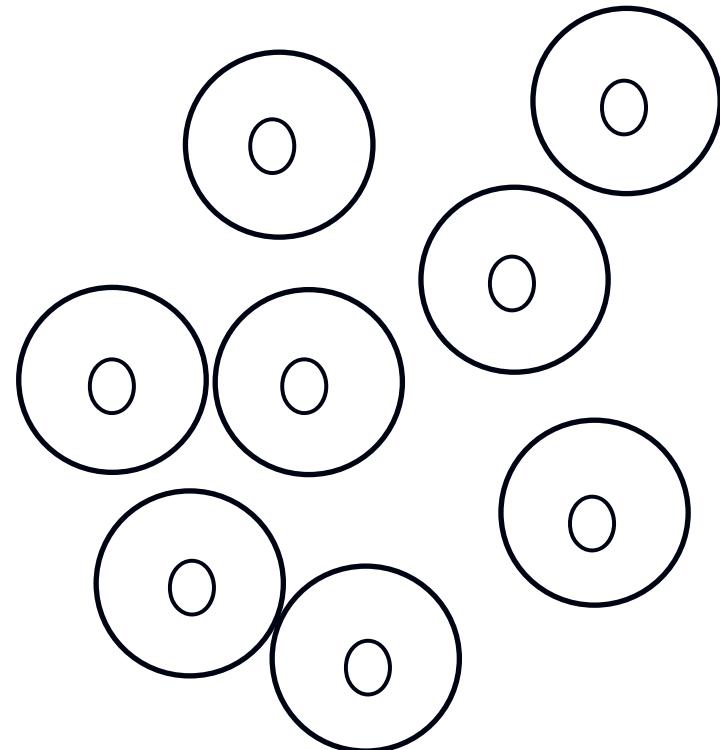
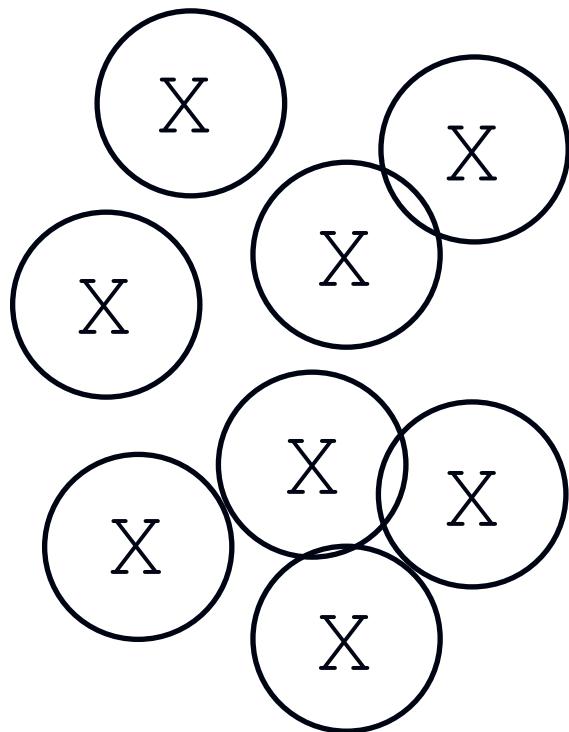
Intuitions



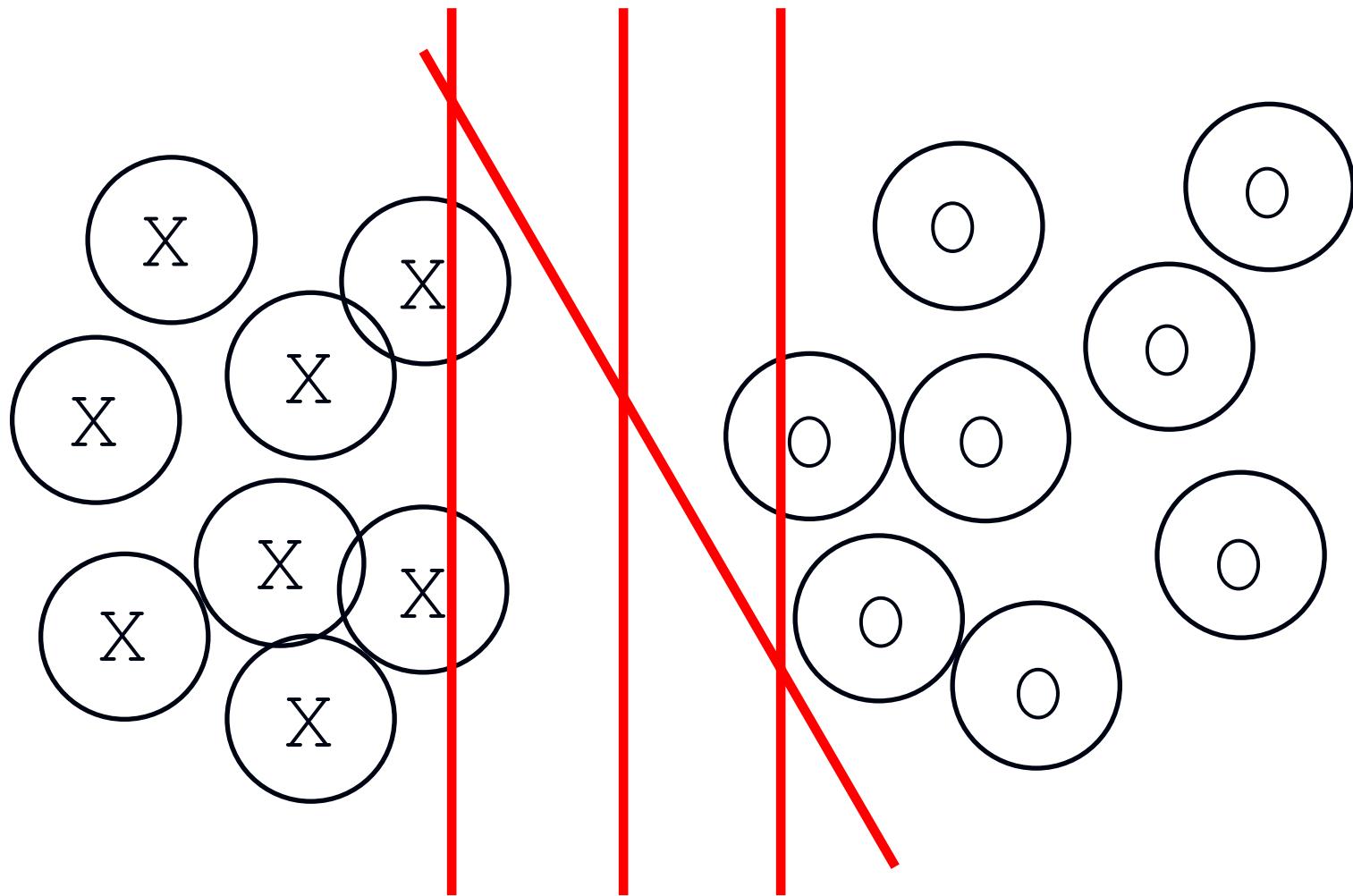
A “Good” Separator



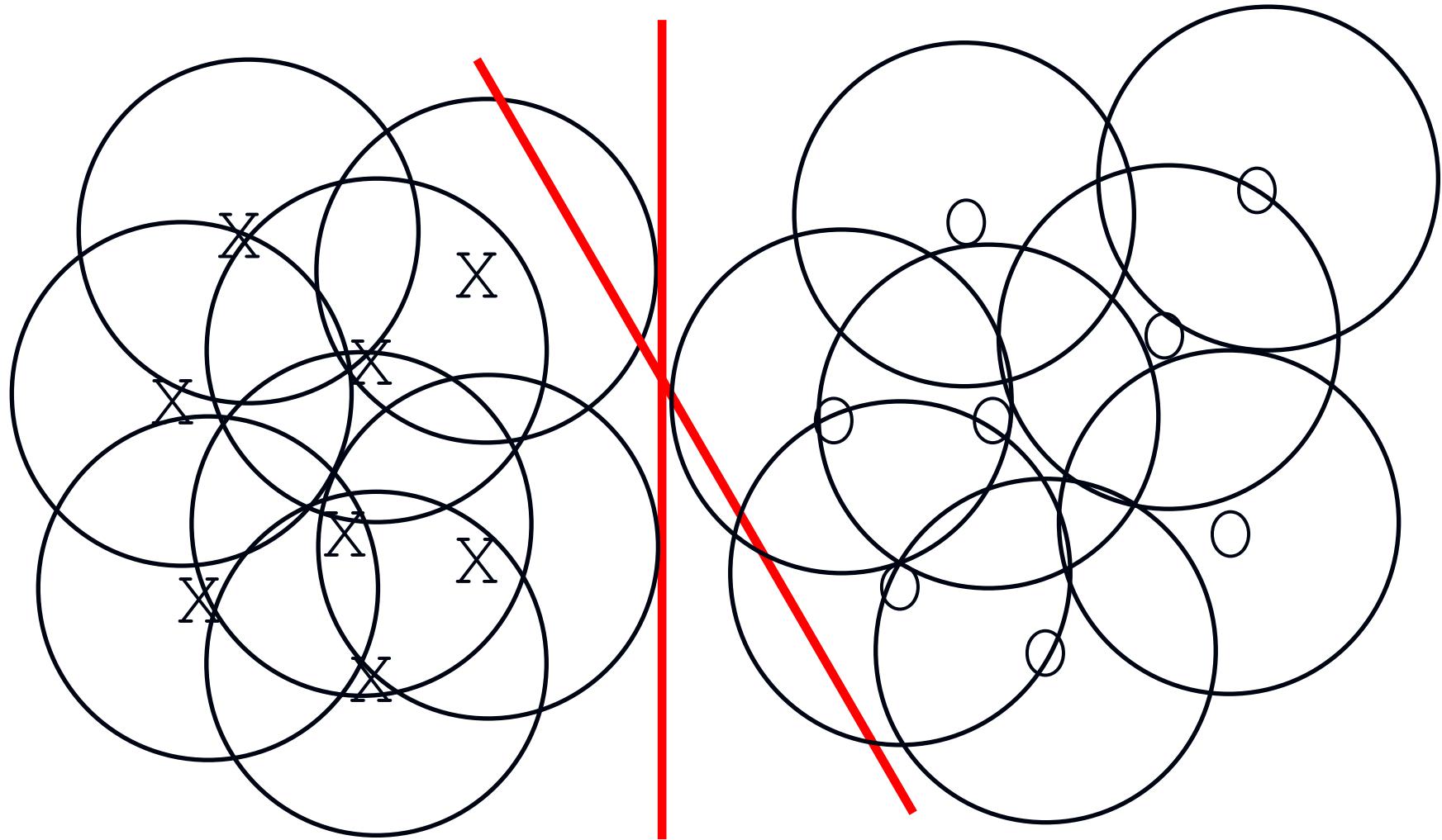
Noise in the Observations



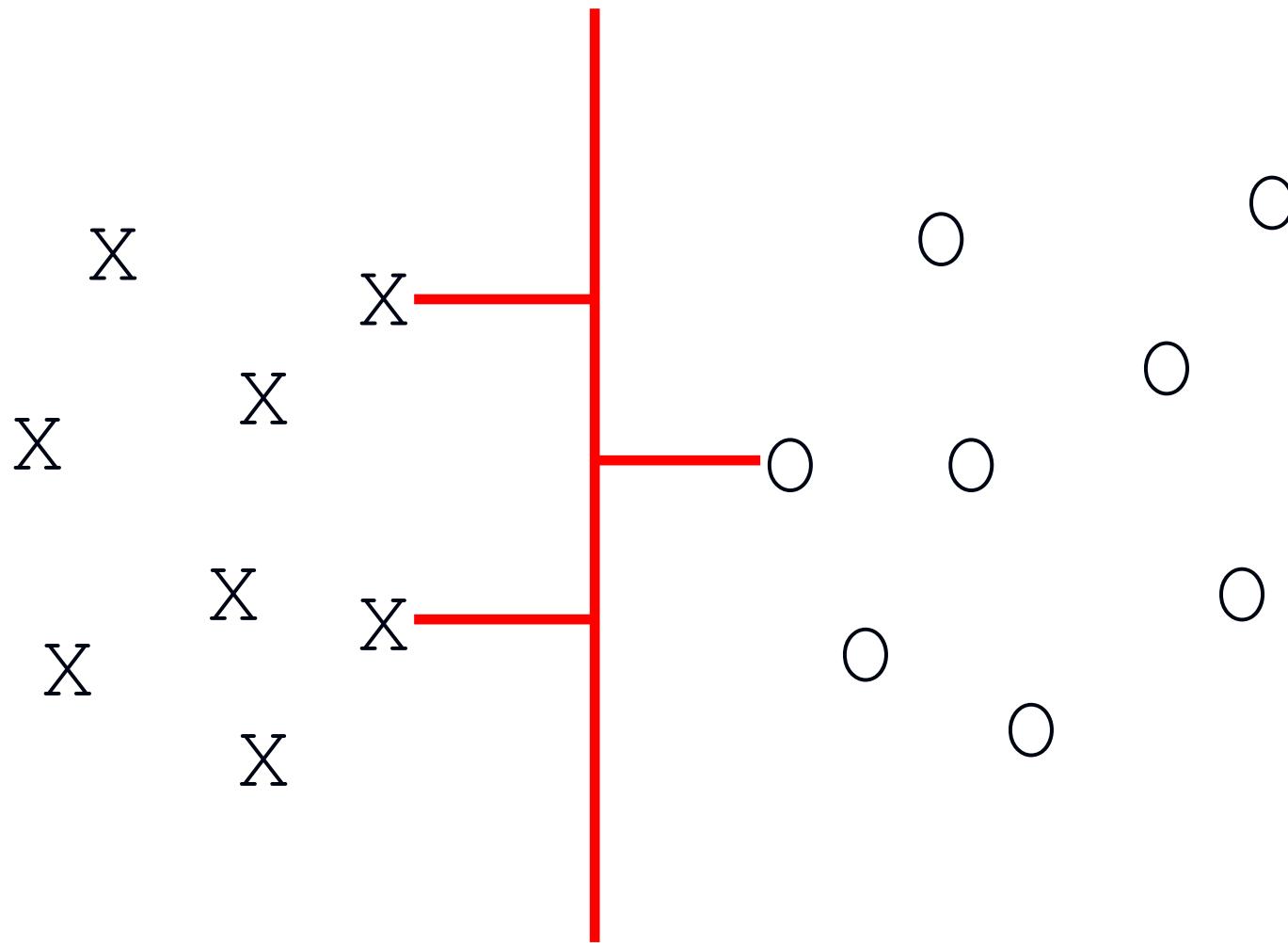
Ruling Out Some Separators



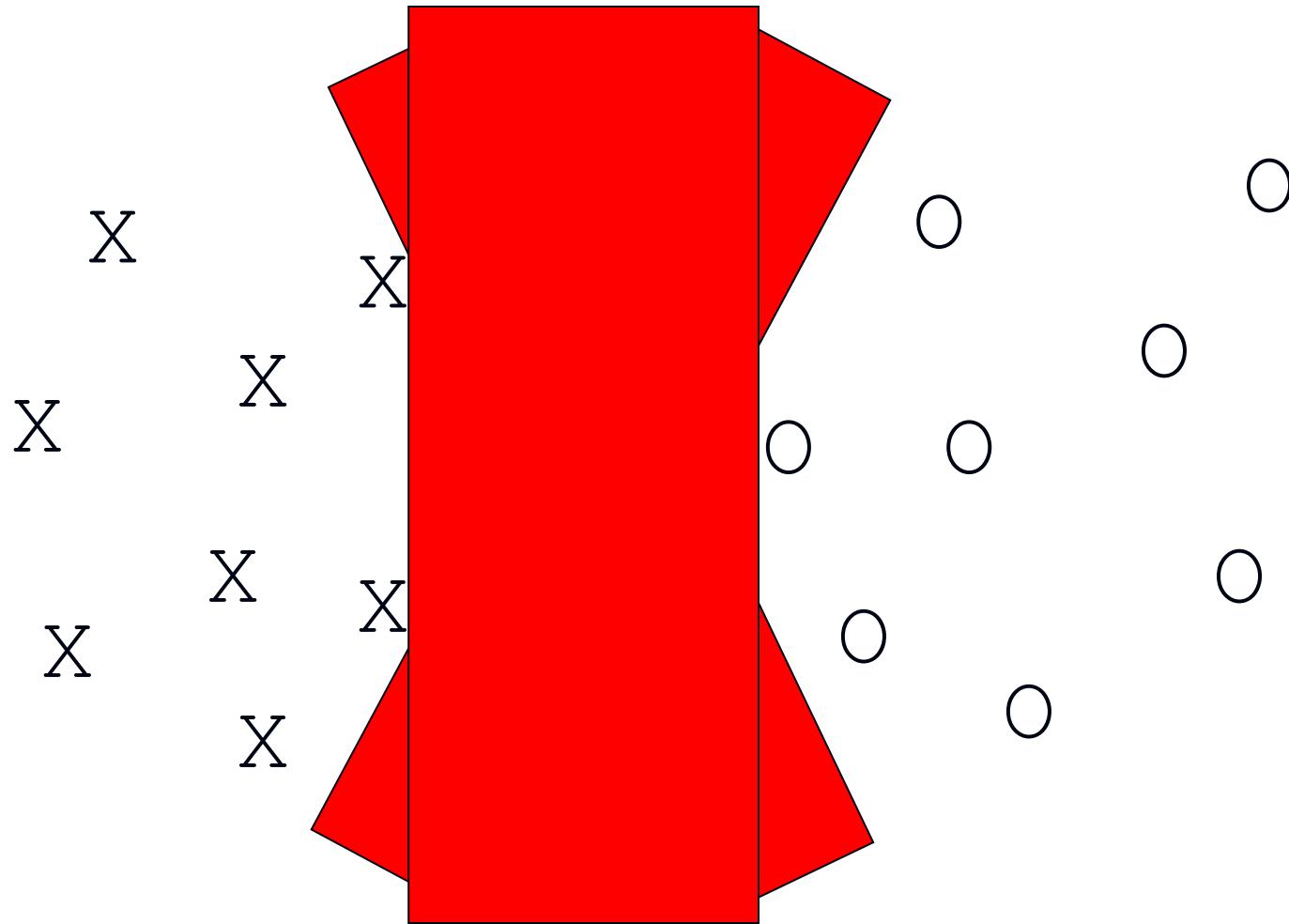
Lots of Noise



Maximizing the Margin



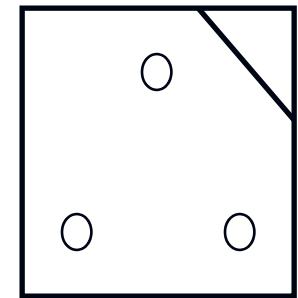
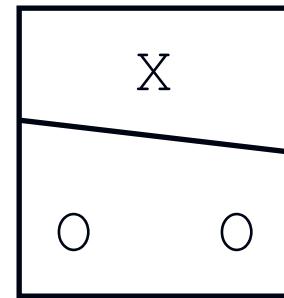
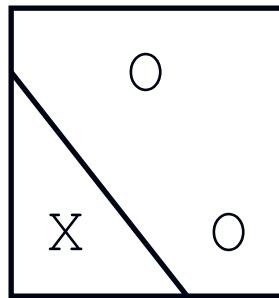
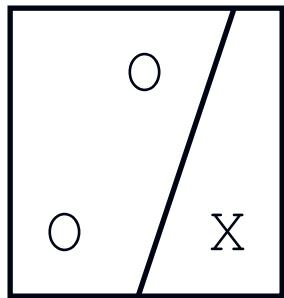
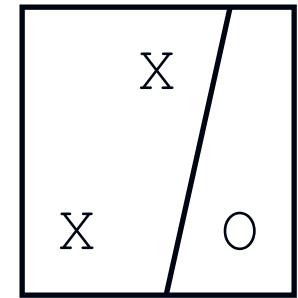
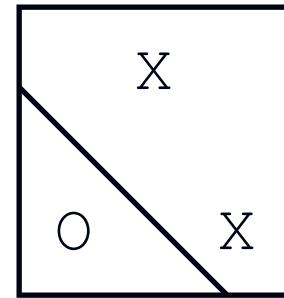
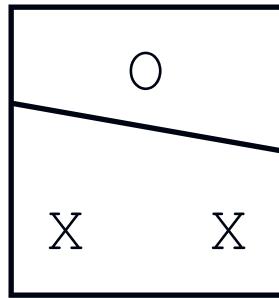
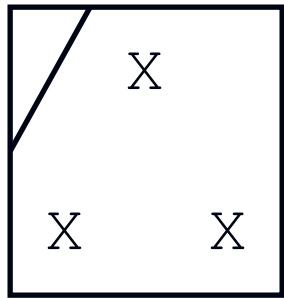
“Fat” Separators



Why Maximize Margin?

- Increasing margin reduces *capacity*
- Must restrict capacity to generalize
 - m training instances
 - 2^m ways to label them
 - What if function class that can separate them all?
 - *Shatters* the training instances
- VC Dimension is largest m such that function class can shatter some set of m points

VC Dimension Example

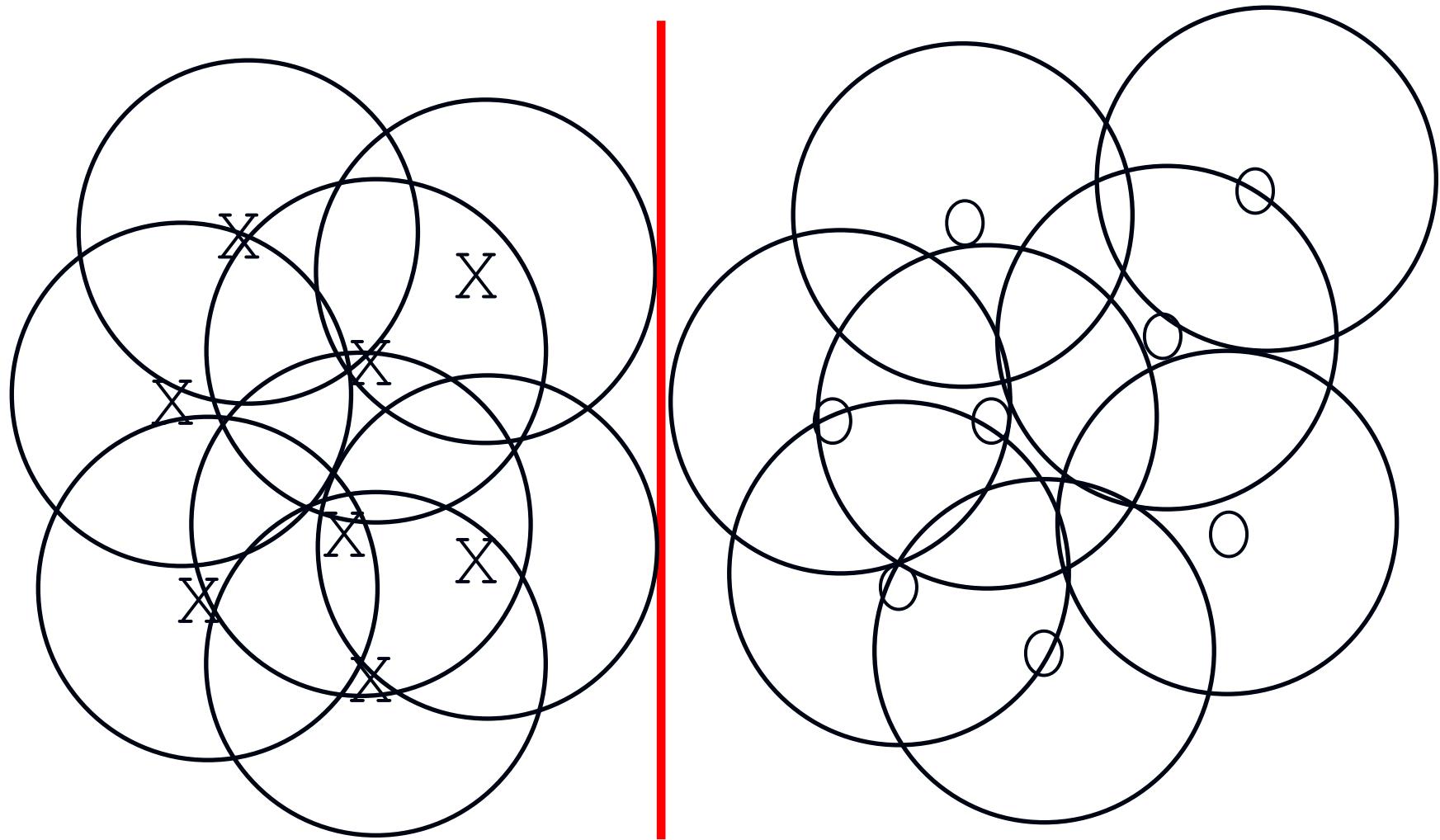


Bounding Generalization Error

- $R[f]$ = risk, test error
- $R_{\text{emp}}[f]$ = empirical risk, train error
- h = VC dimension
- m = number of training instances
- δ = probability that bound does not hold

$$R[f] \leq R_{\text{emp}}[f] + \sqrt{\frac{1}{m} \left[h \left[\ln \frac{2m}{h} + 1 \right] + \ln \frac{4}{\delta} \right]}$$

Support Vectors



The Math

■ Training instances

- $x \in \Re^n$
- $y \in \{-1, 1\}$

■ Decision function

- $f(x) = \text{sign}(\langle w, x \rangle + b)$
- $w \in \Re^n$
- $b \in \Re$

■ Find w and b that

- Perfectly classify training instances
 - Assuming linear separability
- Maximize margin

The Math

- For perfect classification, we want
 - $y_i (\langle w, x_i \rangle + b) \geq 0$ for all i
 - Why?
- To maximize the margin, we want
 - w that minimizes $|w|^2$

Dual Optimization Problem

- Maximize over α

- $W(\alpha) = \sum_i \alpha_i - 1/2 \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$

- Subject to

- $\alpha_i \geq 0$
 - $\sum_i \alpha_i y_i = 0$

- Decision function

- $f(x) = \text{sign}(\sum_i \alpha_i y_i \langle x, x_i \rangle + b)$

What if Data Are Not Perfectly Linearly Separable?

- Cannot find w and b that satisfy
 - $y_i (\langle w, x_i \rangle + b) \geq 1$ for all i
- Introduce slack variables ξ_i
 - $y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i$ for all i
- Minimize
 - $|w|^2 + C \sum \xi_i$

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?

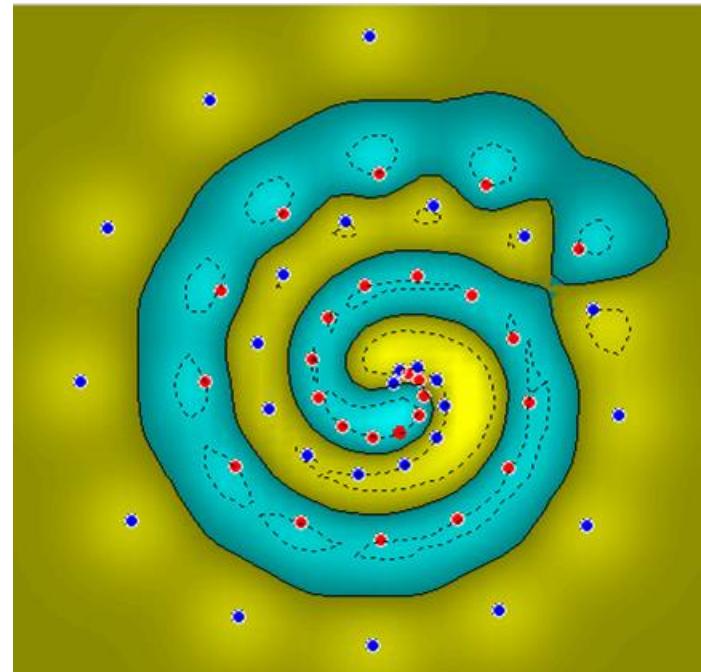
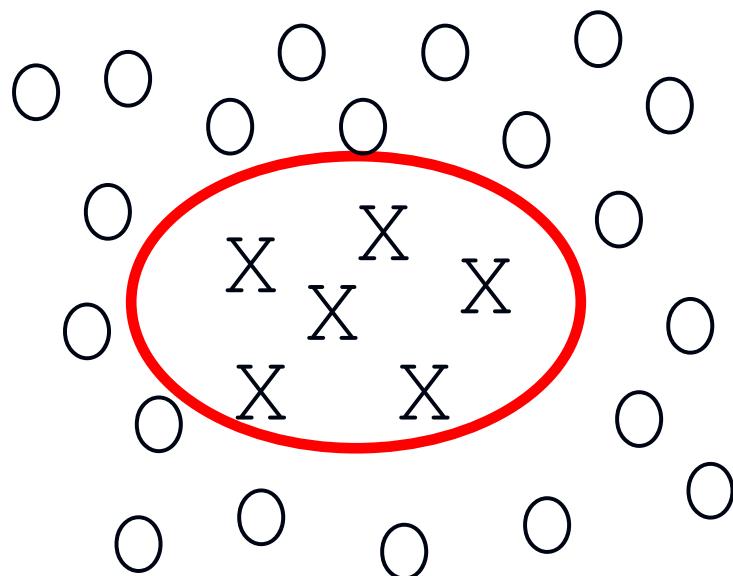
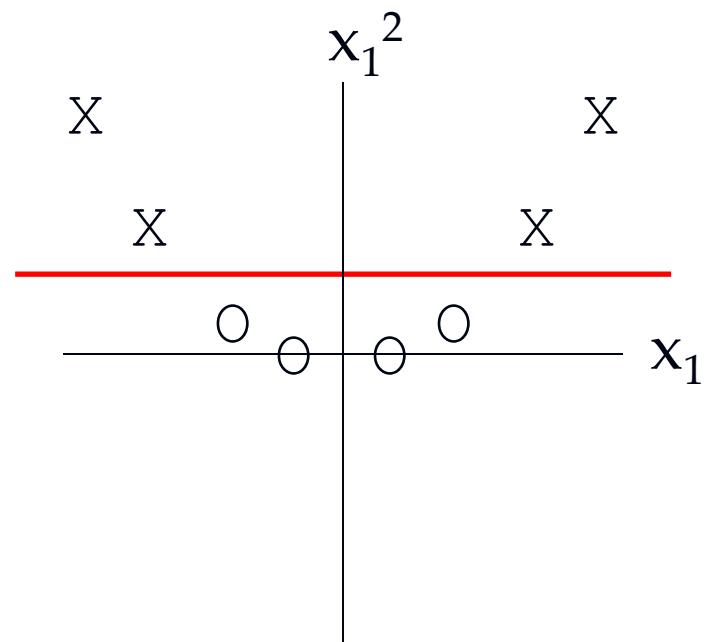
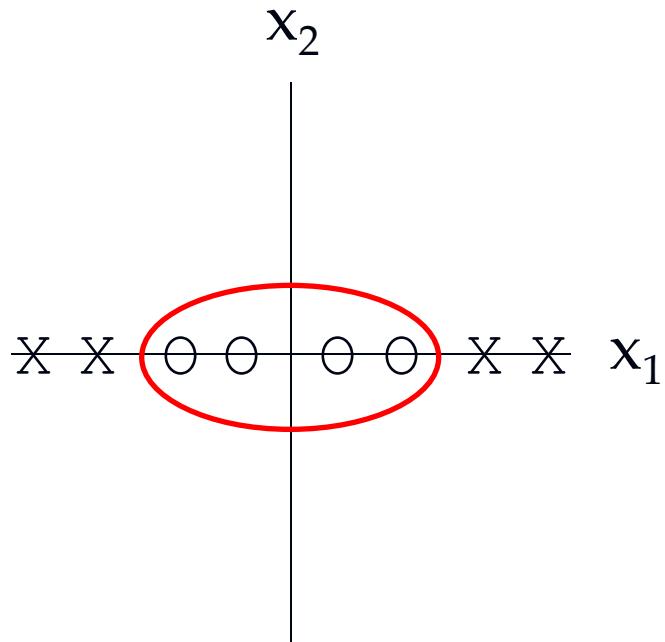


Image from <http://www.atrandomresearch.com/iclass/>

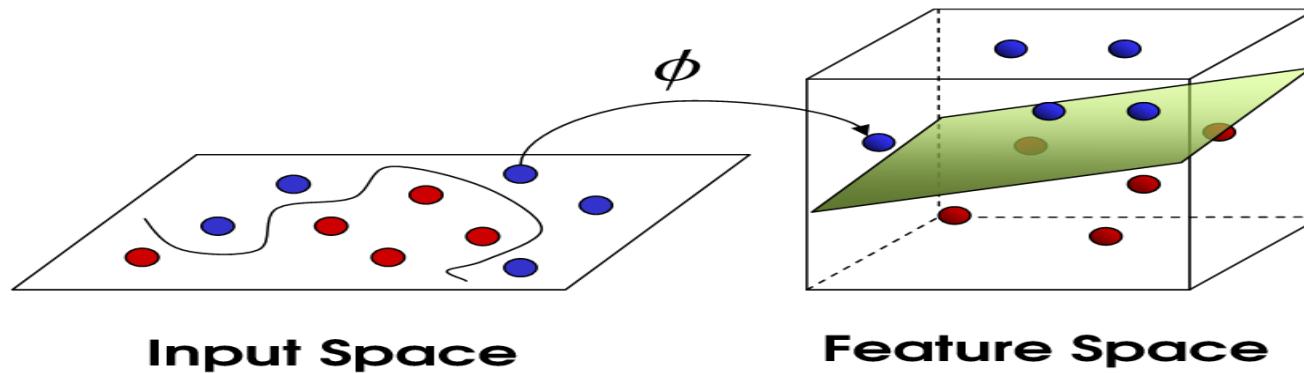
Kernel Methods

Making the Non-Linear Linear

When Linear Separators Fail



Mapping into a New Feature Space



$$\Phi : x \rightarrow X = \Phi(x)$$

$$\Phi(x_1, x_2) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

- Rather than run SVM on x_i , run it on $\Phi(x_i)$
- Find non-linear separator in input space
- What if $\Phi(x_i)$ 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
 - $K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$
- Computing $K(x_1, x_2)$ should be efficient, much more so than computing $\Phi(x_1)$ and $\Phi(x_2)$
- Use $K(x_1, x_2)$ in SVM algorithm rather than $\langle x_1, x_2 \rangle$
- Remarkably, this is possible

The Polynomial Kernel

- $K(x_1, x_2) = \langle x_1, x_2 \rangle^2$
 - $x_1 = (x_{11}, x_{12})$
 - $x_2 = (x_{21}, x_{22})$
- $\langle x_1, x_2 \rangle = (x_{11}x_{21} + x_{12}x_{22})$
- $\langle x_1, x_2 \rangle^2 = (x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11} x_{12} x_{21} x_{22})$
- $\Phi(x_1) = (x_{11}^2, x_{12}^2, \sqrt{2}x_{11} x_{12})$
- $\Phi(x_2) = (x_{21}^2, x_{22}^2, \sqrt{2}x_{21} x_{22})$
- $K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$

The Polynomial Kernel

- $\Phi(x)$ contains all monomials of degree d
- Useful in visual pattern recognition
- Number of monomials
 - 16x16 pixel image
 - 10^{10} monomials of degree 5
- Never explicitly compute $\Phi(x)$!
- Variation - $K(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^2$

A Few Good Kernels

- Dot product kernel
 - $K(x_1, x_2) = \langle x_1, x_2 \rangle$
- Polynomial kernel
 - $K(x_1, x_2) = \langle x_1, x_2 \rangle^d$ (Monomials of degree d)
 - $K(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^d$ (All monomials of degree 1,2,...,d)
- Gaussian kernel
 - $K(x_1, x_2) = \exp(-\|x_1 - x_2\|^2 / 2\sigma^2)$
 - Radial basis functions
- Sigmoid kernel
 - $K(x_1, x_2) = \tanh(\langle x_1, x_2 \rangle + \theta)$
 - Neural networks
- Establishing “kernel-hood” from first principles is non-trivial

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

Using a Different Kernel in the Dual Optimization Problem

- For example, using the polynomial kernel with $d = 4$ (including lower-order terms).

- Maximize over α

- $W(\alpha) = \sum_i \alpha_i - 1/2 \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle^4$

- Subject to

- $\alpha_i \geq 0$
- $\sum_i \alpha_i y_i = 0$

- Decision function

- $f(x) = \text{sign}(\sum_i \alpha_i y_i \langle x, x_i \rangle + b)$

$$(\langle x_i, x_j \rangle + 1)^4$$

$$\cancel{\langle x_i, x_j \rangle}$$

So by the kernel trick,
These are kernels!
we just replace them!

$$(\langle x_i, x_j \rangle + 1)^4$$

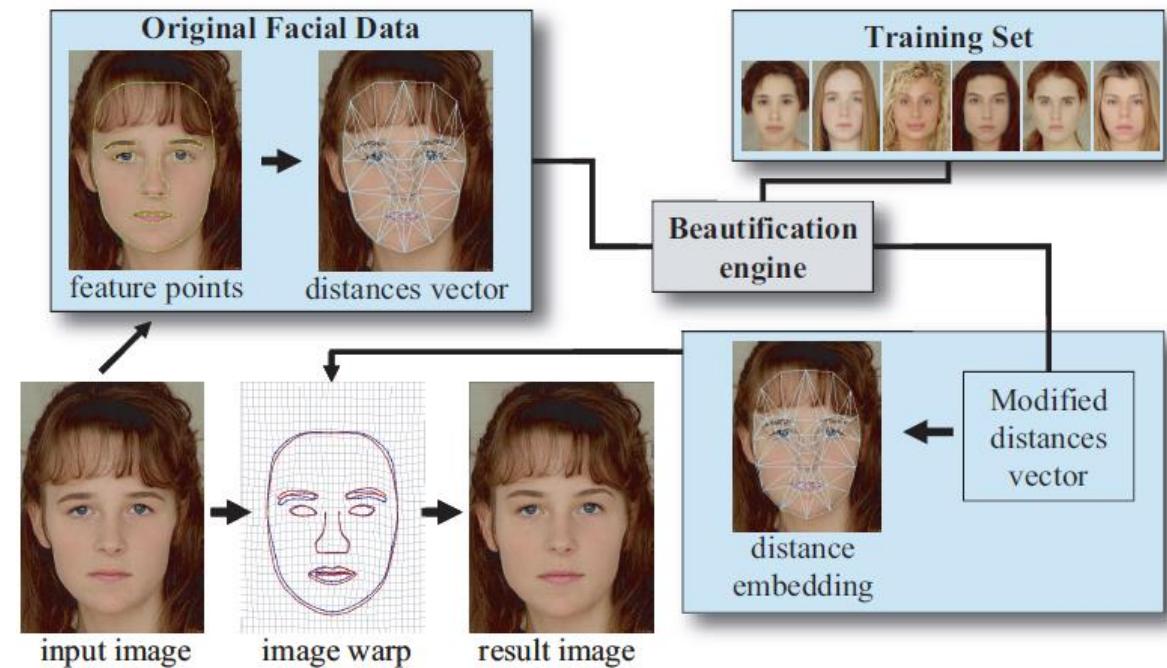
$$\cancel{\langle x, x_i \rangle}$$

Exotic Kernels

- Strings
- Trees
- Graphs
- The hard part is establishing kernel-hood

Application: “Beautification Engine”

(Leyvand et al., 2008)



Conclusion

- SVMs find optimal linear separator
- The kernel trick makes SVMs non-linear learning algorithms