Support Vector Machines & Kernels

Doing *really* well with linear decision surfaces

Adapted from slides by Tim Oates

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

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

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

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:

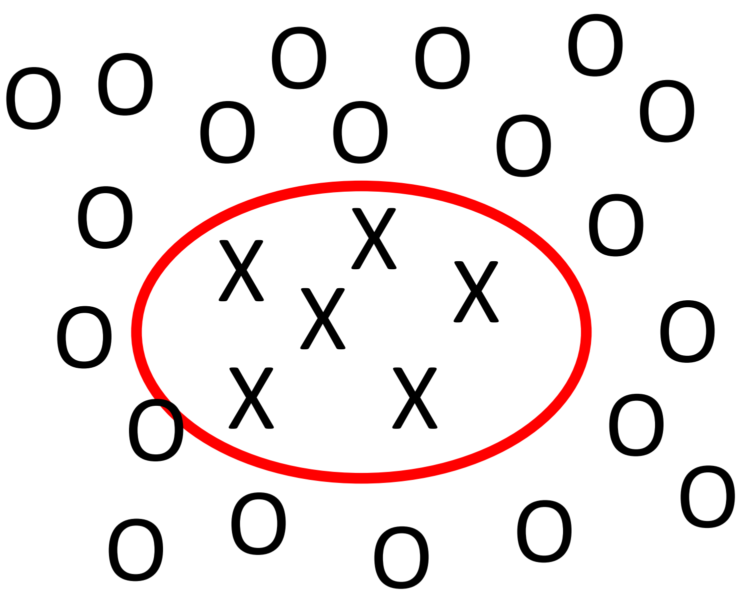
$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 + C \sum_i \xi_i$$

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

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?



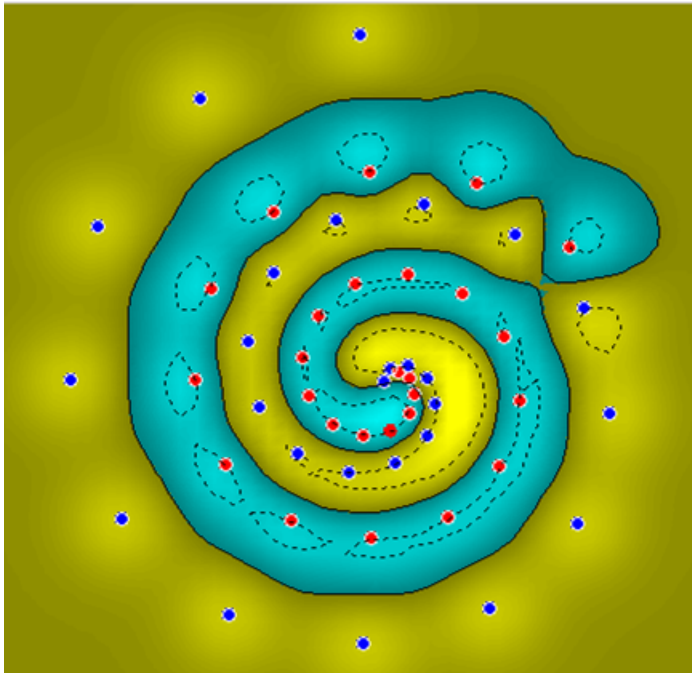
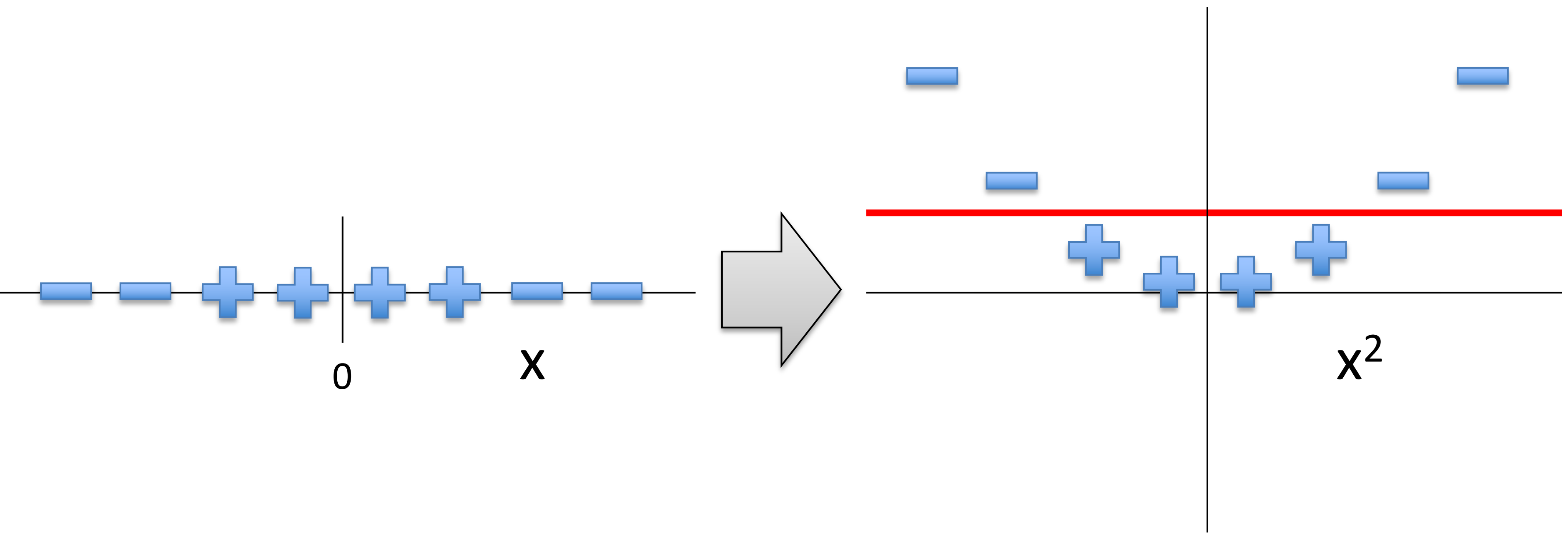


Image from http://www.atrandomresearch.com/iclass/

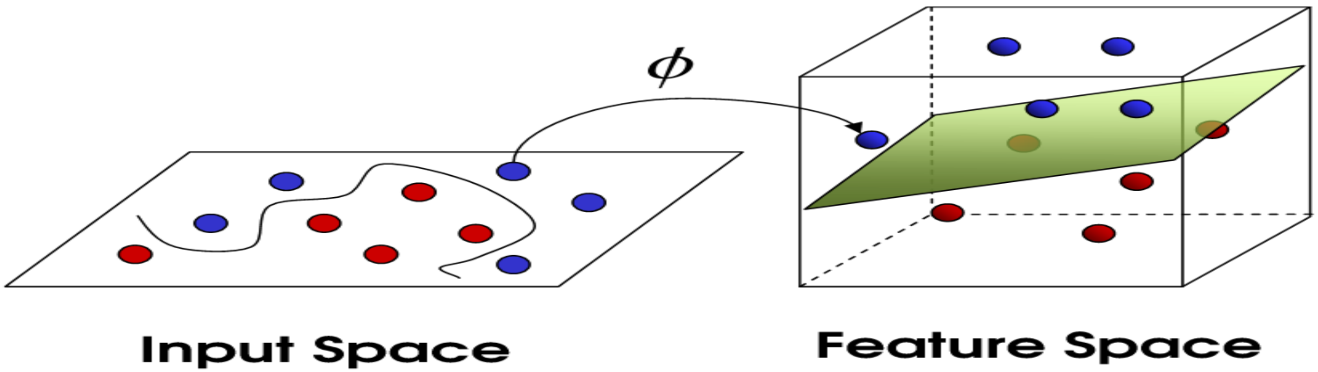
Kernel Methods

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 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

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

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

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

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

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

$$\Phi(\mathbf{x}_j) = [x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}]$$

The Polynomial Kernel

- Given by $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d$
  - Φ(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 Φ(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)!

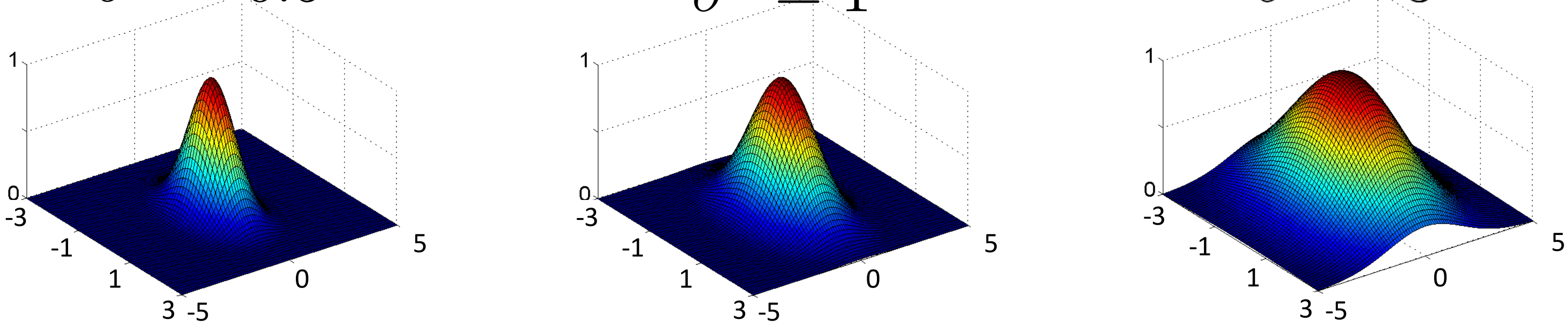
The Gaussian Kernel

- 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 $x_i = x_j$
  - 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 ⟷ higher bias, lower variance

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

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

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

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

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

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

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

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

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)