

# SVMs: Duality and Kernel Trick

Machine Learning - 10601

Geoff Gordon, Miroslav Dudík

[partly based on slides of Ziv-Bar Joseph]

<http://www.cs.cmu.edu/~ggordon/10601/>

November 18, 2009

## SVMs as quadratic programs

Two optimization problems: For the separable and non separable cases

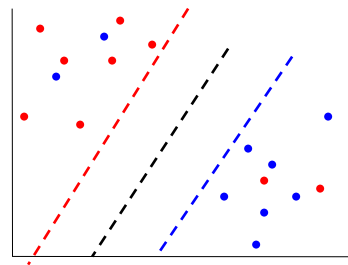
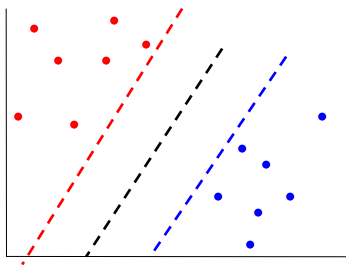
$$\min_{\mathbf{w}, b} \frac{\mathbf{w}^T \mathbf{w}}{2}$$

$$(\mathbf{w}^T \mathbf{x}_i + b) y_i \geq 1$$

$$\min_{\mathbf{w}, b, \varepsilon_i} \frac{\mathbf{w}^T \mathbf{w}}{2} + \sum_{i=1}^n C \varepsilon_i$$

$$(\mathbf{w}^T \mathbf{x}_i + b) y_i \geq 1 - \varepsilon_i$$

$$\varepsilon_i \geq 0$$



## Dual for separable case

$$\min_{\mathbf{w}, b} \frac{\mathbf{w}^T \mathbf{w}}{2}$$
$$(w^T x_i + b) y_i \geq 1$$

## Dual for separable case

$$\min_{\mathbf{w}, b} \frac{\mathbf{w}^T \mathbf{w}}{2}$$
$$(w^T x_i + b) y_i - 1 \geq 0 \quad (\alpha_i)$$

## Dual for separable case

$$\min_{\mathbf{w}, b} \frac{\mathbf{w}^T \mathbf{w}}{2}$$

$$(\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \geq 0 \quad (\alpha_i)$$

Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{\mathbf{w}^T \mathbf{w}}{2} - \sum_i \alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \right)$$

## Dual for separable case

$$\min_{\mathbf{w}, b} \frac{\mathbf{w}^T \mathbf{w}}{2}$$

$$(\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \geq 0 \quad (\alpha_i)$$

Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{\mathbf{w}^T \mathbf{w}}{2} - \sum_i \alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \right)$$

$$\min_{\mathbf{w}, b} \max_{\boldsymbol{\alpha} \geq 0} L(\mathbf{w}, b, \boldsymbol{\alpha})$$

$$\max_{\boldsymbol{\alpha} \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha})$$

## Dual for separable case

$$\min_{\mathbf{w}, b} \frac{\mathbf{w}^T \mathbf{w}}{2}$$

$$(\mathbf{w}^T \mathbf{x}_i + b)y_i - 1 \geq 0 \quad (\alpha_i)$$

Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{\mathbf{w}^T \mathbf{w}}{2} - \sum_i \alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b)y_i - 1 \right)$$

$$\min_{\mathbf{w}, b} \max_{\boldsymbol{\alpha} \geq 0} L(\mathbf{w}, b, \boldsymbol{\alpha})$$

$$\max_{\boldsymbol{\alpha} \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha})$$

**optimality of  $\alpha_i$**

**optimality of  $\mathbf{w}$  and  $b$**

for all  $i$ :

$$\alpha_i \geq 0$$

$$\alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b)y_i - 1 \right) = 0$$

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i y_i$$

$$\sum_i \alpha_i y_i = 0$$

## Dual for separable case

Dual formulation

$$\max_{\boldsymbol{\alpha} \geq 0} \min_{\mathbf{w}, b} \left( \frac{\mathbf{w}^T \mathbf{w}}{2} - \sum_i \alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b)y_i - 1 \right) \right)$$

Optimality conditions  
(KKT conditions)

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i y_i$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b)y_i - 1 \right) = 0$$

## Dual for separable case

### Dual formulation

$$\max_{\mathbf{a} \geq 0} \min_{\mathbf{w}, b} \left( \frac{\mathbf{w}^T \mathbf{w}}{2} - \sum_i \alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \right) \right)$$

$$\max_{\mathbf{a}} \left( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \right)$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

### Optimality conditions (KKT conditions)

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i y_i$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \right) = 0$$

## Dual for separable case

### Dual formulation

$$\max_{\mathbf{a}} \left( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \right)$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

### Optimality conditions (KKT conditions)

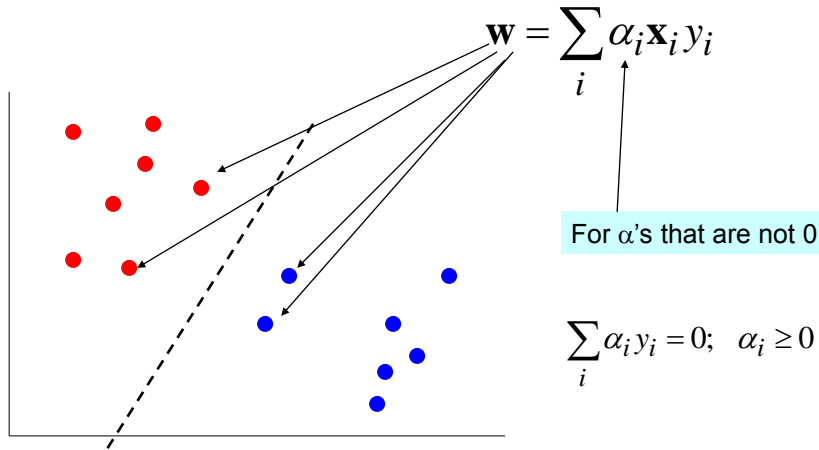
$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i y_i$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i \left( (\mathbf{w}^T \mathbf{x}_i + b) y_i - 1 \right) = 0$$

## Dual SVM - interpretation



## Dual SVM for linearly separable case

Our dual target function:  $\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

Dot product across all pairs of training samples

Dot product with all training samples

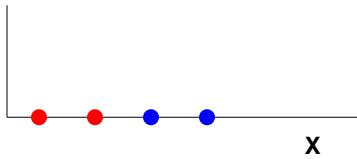
To evaluate a new sample  $\mathbf{x}$   
we need to compute:

$$\mathbf{w}^T \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

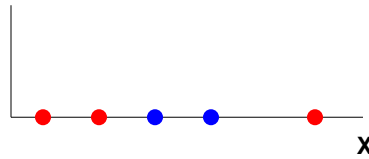
This might be too much work!  
(e.g. when lifting  $\mathbf{x}$  into high dimensions)

## Classifying in 1-d

Can an SVM correctly classify this data?

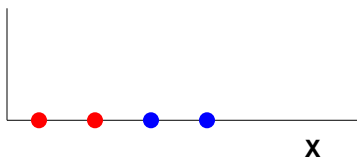


What about this?

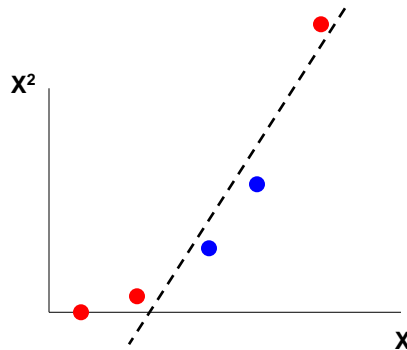


## Classifying in 1-d

Can an SVM correctly classify this data?

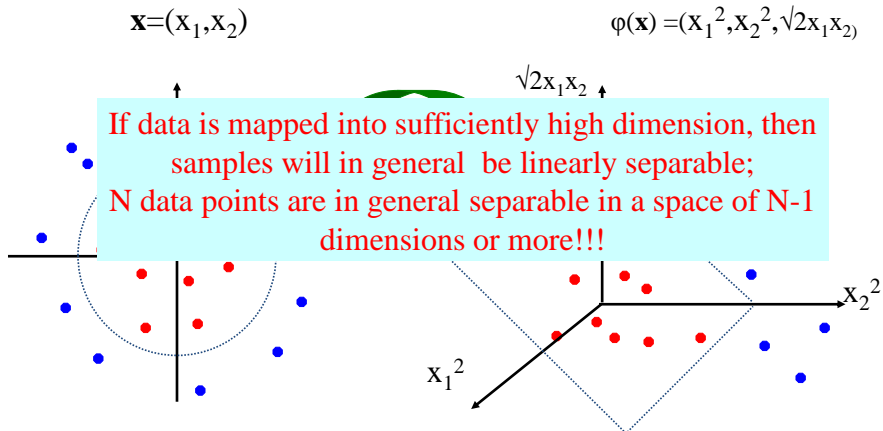


And now?



## Non-linear SVDs in 2-d

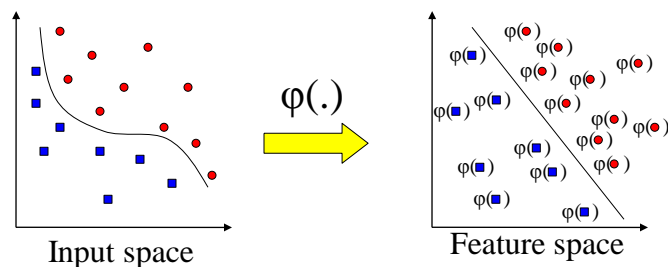
- The original input space ( $\mathbf{x}$ ) can be mapped to some higher-dimensional feature space ( $\varphi(\mathbf{x})$ ) where the training set is separable:



This slide is courtesy of [www.iro.umontreal.ca/~pift6080/documents/papers/svm\\_tutorial.ppt](http://www.iro.umontreal.ca/~pift6080/documents/papers/svm_tutorial.ppt)

## Transformation of Inputs

- Possible problems
  - High computation burden due to high-dimensionality
  - Many more parameters
- SVM solves these two issues simultaneously
  - "Kernel tricks" for efficient computation
  - Dual formulation only assigns parameters to samples, not features



# Polynomials of degree two

- While working in higher dimensions is beneficial, it also increases our running time because of the dot product computation
- However, there is a neat trick we can use
- consider all quadratic terms for  $x_1, x_2 \dots x_m$

$$\max_{\alpha} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i) \phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

m is the number of features in each vector

The  $\sqrt{2}$  term will become clear in the next slide

$$\phi(x) = \begin{matrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{matrix}$$

← m+1 linear terms

← m quadratic terms

← m(m-1)/2 pairwise terms

# Dot product for polynomials of degree two

How many operations do we need for the dot product?

$$\phi(x)\phi(z) = \begin{matrix} 1 & 1 \\ \sqrt{2}x_1 & \sqrt{2}z_1 \\ \vdots & \vdots \\ \sqrt{2}x_m & \sqrt{2}z_m \\ x_1^2 & z_1^2 \\ \vdots & \vdots \\ x_m^2 & z_m^2 \\ \sqrt{2}x_1x_2 & \sqrt{2}z_1z_2 \\ \vdots & \vdots \\ \sqrt{2}x_{m-1}x_m & \sqrt{2}z_{m-1}z_m \end{matrix} \cdot \begin{matrix} 1 & 1 \\ \sqrt{2}z_1 & \sqrt{2}z_1 \\ \vdots & \vdots \\ \sqrt{2}z_m & \sqrt{2}z_m \\ z_1^2 & z_1^2 \\ \vdots & \vdots \\ z_m^2 & z_m^2 \\ \sqrt{2}z_1z_2 & \sqrt{2}z_1z_2 \\ \vdots & \vdots \\ \sqrt{2}z_{m-1}z_m & \sqrt{2}z_{m-1}z_m \end{matrix}$$

$$= \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_{i,j=i+1} 2x_i x_j z_i z_j + 1$$

m                      m                      m(m-1)/2                       **$\approx m^2$**

## Polynomials of degree **d** in **m** variables

## Polynomials of degree **d** in **m** variables

### Original formulation

$$\text{Min } (w^T w)/2$$

$$(w^T \phi(x_i) + b) y_i \geq 1$$

### Dual formulation

$$\max_{\alpha} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i) \phi(x_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

## The kernel trick

How many operations do we need for the dot product?

$$\varphi(x)\varphi(z) = \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

m
m
m(m-1)/2
≈ m<sup>2</sup>

There's **structure** to this dot product... we can do this faster!

$$(xz+1)^2 = (xz)^2 + 2(xz) + 1$$

We only need m operations!

Note that to evaluate a new sample we are also using dot products so we save there as well

## Where we are

Our dual target function:

$$\max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x}_i) \varphi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i$$

mn<sup>2</sup> operations to evaluate all coefficients

To evaluate a new sample  $\mathbf{x}$  we need to compute:

$$\mathbf{w}^T \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \varphi(\mathbf{x}) + b$$

mr operations where r is the number of support vectors ( $\alpha_i > 0$ )

## Other kernels

• Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right kernel function

- Radial-Basis Function: 
$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{(\mathbf{x} - \mathbf{z})^2}{2\sigma^2}\right)$$

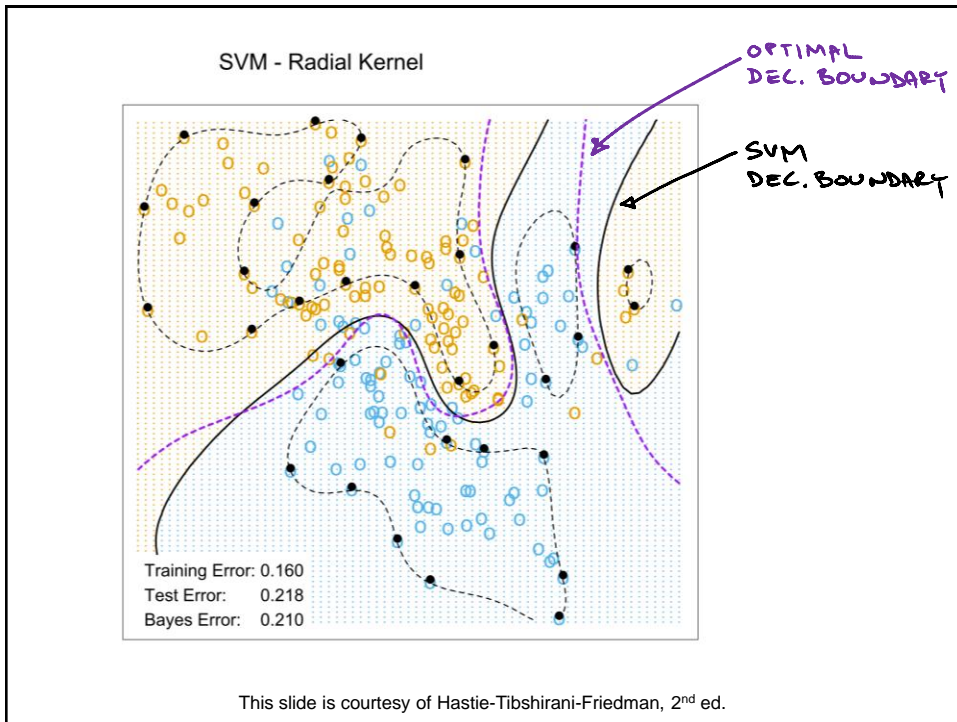
- kernel functions for discrete objects (graphs, strings, etc.)

## Kernels measure similarity

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{(\mathbf{x} - \mathbf{z})^2}{2\sigma^2}\right)$$

Decision rule for a new sample  $\mathbf{x}$ :

$$\mathbf{w}^T \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$



## Dual formulation for non-separable case

Dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0 \quad \forall i$$

To evaluate a new sample  $\mathbf{x}$   
we need to compute:

$$\mathbf{w}^T \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \mathbf{x} + b$$

The only difference is  
that the  $\alpha_i$ 's are now  
bounded

## Why do SVMs work?

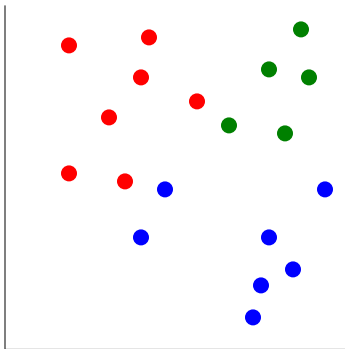
- If we are using huge features spaces (with kernels) how come we are not overfitting the data?
  - We maximize margin!
  - We minimize loss + regularization

## Software

- A list of SVM implementation can be found at <http://www.kernel-machines.org/software.html>
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

## Multi-class classification with SVMs

What if we have data from more than two classes?



- Most common solution: One vs. all
  - create a classifier for each class against all other data
  - for a new point use all classifiers and compare the margin for all selected classes

Note that this is not necessarily valid since this is not what we trained the SVM for, but often works well in practice

## Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

→Lots of very successful applications!!!

## Handwritten digit recognition



3-nearest-neighbor = 2.4% error

400–300–10 unit MLP = 1.6% error

LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms)  $\approx$  0.6% error

## Important points

- Difference between regression classifiers and SVMs'
- Maximum margin principle
- Target function for SVMs
- Linearly separable and non separable cases
- Dual formulation of SVMs
- Kernel trick and computational complexity