

# SVMs, Duality and the Kernel Trick, *Cont.*

Machine Learning – 10701/15781  
 Carlos Guestrin  
 Carnegie Mellon University

October 22<sup>nd</sup>, 2007

©2005-2007 Carlos Guestrin

1

## Lagrange multipliers – Dual variables

**Solving:**  $\min_x \max_{\alpha} x^2 - \alpha(x - b)$   
 s.t.  $\alpha \geq 0$

$\frac{\partial L}{\partial x} = 2x - \alpha = 0 \Rightarrow \alpha = 2x$

$\frac{\partial L}{\partial \alpha} = -(x - b)$

$x=1 \Rightarrow \frac{\partial L}{\partial \alpha} = 0$   
 $\alpha = 2 > 0$   
 constraint relevant

if I set  $x = -1$   
 $\frac{\partial L}{\partial \alpha} = 0$   
 but  $\alpha = -2 < 0$   
 no way I can set  $\frac{\partial L}{\partial \alpha} = 0$

$x=0 \Rightarrow$  ignore  $\alpha=0$   
 constraint irrelevant

©2005-2007 Carlos Guestrin

2

## Dual SVM formulation – the non-separable case

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

$\alpha_i$  can't be too big  
 $\alpha_i$  very large "when you really constraint problem"  
 + + +  
 \* target  $k_j$

$$\sum_i \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0$$

only difference

$\frac{dL}{d\alpha}$

derivation of dual for problem with slack penalty similar

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

$$b = y_k - \mathbf{w} \cdot \mathbf{x}_k$$

for any  $k$  where  $C > \alpha_k > 0$

## Why did we learn about the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal
- But, more importantly, the "kernel trick"!!!
  - Another little detour...

Reminder from last time: What if the data is not linearly separable?

*high dim space*

*w/  $x_1, x_2$*

**Use features of features of features...**

$$\Phi(x) : R^m \mapsto F$$

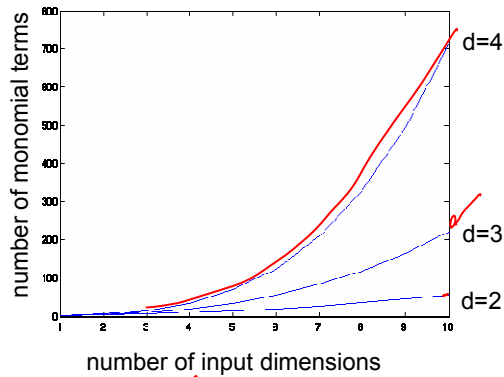
$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$\Phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ \sin x_1 \cdot e^{x_2} \\ \vdots \end{pmatrix}$

**Feature space can get really large really quickly!**

## Higher order polynomials

$$\text{num. terms} = \binom{d+m-1}{d} = \frac{(d+m-1)!}{d!(m-1)!}$$



m – input features  
d – degree of polynomial

*can take a looong time!!*

grows fast!  
d = 6, m = 100  
about 1.6 billion terms

## Dual formulation only depends on dot-products, not on $w$ !

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$x_i \rightarrow \mathbb{R}^m$   
 $x_j \rightarrow \mathbb{R}^m$

Scale-  
 $x_i \cdot x_j = \langle x_i, x_j \rangle$   
 $= \sum_{v=1}^m x_i^{(v)} \cdot x_j^{(v)}$

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \quad \text{dot product}$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i > 0$$

©2005-2007 Carlos Guestrin

7

## Dot-product of polynomials

$\Phi(u) \cdot \Phi(v) =$  polynomials of degree  $d$

$d=1 \quad \Phi(u) \cdot \Phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = \underline{u \cdot v}$

$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \leftarrow 2\text{-dimensional}$

$d=2 \quad \Phi(u) \cdot \Phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2^2 \\ \mu_1 \mu_1 \\ \mu_2 \mu_1 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2^2 \\ \nu_2 \nu_1 \\ \nu_2 \nu_1 \end{pmatrix} = u_1^2 v_1^2 + \mu_1 \mu_2 v_1 v_2 + \mu_2^2 v_2^2$

$+ \mu_2 \mu_1 v_2 v_1$

$d=m,$

$$\Phi(u) \cdot \Phi(v) = (u \cdot v)^m$$

$= (\mu_1 \nu_1)^2 + (\mu_2 \nu_2)^2$

$+ 2 \mu_1 \nu_1 \cdot \mu_2 \nu_2 = (\mu_1 \nu_1 + \mu_2 \nu_2)^2$

$= (\mu \nu)^2$

©2005-2007 Carlos Guestrin

8

# Finally: the “kernel trick”!

$$\text{maximize}_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

*closed form  
e.g., poly degree  
exactly d*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

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

$$b = y_k - \mathbf{w} \cdot \Phi(\mathbf{x}_k)$$

for any  $k$  where  $C > \alpha_k > 0$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features
- Very interesting theory – Reproducing Kernel Hilbert Spaces
  - Not covered in detail in 10701/15781, more in 10702

©2005-2007 Carlos Guestrin

9

# Polynomial kernels

- All monomials of degree  $d$  in  $O(d)$  operations:

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d = \text{polynomials of degree } d$$

- How about all monomials of degree up to  $d$ ?

- Solution 0:  $\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \sum_{i=0}^d (\mathbf{u} \cdot \mathbf{v})^i$

- Better solution:

$$(\mathbf{u} \cdot \mathbf{v})^0 + (\mathbf{u} \cdot \mathbf{v})^1 + (\mathbf{u} \cdot \mathbf{v})^2 + \dots + (\mathbf{u} \cdot \mathbf{v})^d = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

*poly degree up to including  $d$  :  $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$*

©2005-2007 Carlos Guestrin

10

## Common kernels

- Polynomials of degree  $d$   $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$
- Polynomials of degree up to  $d$   $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$
- Squared-exponential Gaussian kernels  $K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right)$   
 *$\phi(u) \rightarrow 1.6$  billion dimensions*  
*bandwidth*
- Sigmoid  $K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$   
 *$\phi(u) =$  infinite dimensional*

©2005-2007 Carlos Guestrin

11

## Overfitting?

- Huge feature space with kernels, what about overfitting???
- Maximizing margin leads to sparse set of support vectors
- Some interesting theory says that SVMs search for simple hypothesis with large margin
- Often robust to overfitting

©2005-2007 Carlos Guestrin

12

## What about at classification time

- For a new input  $\mathbf{x}$ , if we need to represent  $\Phi(\mathbf{x})$ , we are in trouble!
- Recall classifier:  $\text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)$
- Using kernels we are cool!

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$$

$$\begin{aligned} & \text{w. } \phi(\mathbf{x}) \quad \leftarrow \text{test point} \\ & = \left( \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \right) \cdot \phi(\mathbf{x}) \end{aligned}$$

$$= \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) \quad \leftarrow \text{training data}$$

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

$$b = y_k - \mathbf{w} \cdot \Phi(\mathbf{x}_k)$$

for any  $k$  where  $C > \alpha_k > 0$

©2005-2007 Carlos Guestrin

13

## SVMs with kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors  $\alpha_i$
- At classification time, compute:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

for any  $k$  where  $C > \alpha_k > 0$

Classify as

$$\text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)$$

test case  $\mathbf{x}$

©2005-2007 Carlos Guestrin

14

# Remember kernel regression

Remember kernel regression???

1.  $w_i = \exp(-D(x_p, \text{query})^2 / K_w^2)$
2. How to fit with the local points?

Predict the weighted average of the outputs:  
 predict =  $\frac{\sum w_i y_i}{\sum w_i}$

$$\hat{y} = \frac{\sum_{i=1}^{\text{train}} w_i y_i}{\sum_i w_i}$$

# SVMs v. Kernel Regression

## SVMs

$$\text{sign}(w \cdot \Phi(x) + b)$$

or

$$\text{sign}\left(\sum_i \alpha_i y_i K(x, x_i) + b\right)$$

"learn" from data

## Kernel Regression

$$\text{sign}\left(\frac{\sum_i y_i K(x, x_i)}{\sum_j K(x, x_j)}\right)$$

same "type" of classifier

# SVMs v. Kernel Regression

## SVMs

$$\text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)$$

or

$\text{sign}$



## Kernel Regression

$$\text{sign}\left(\frac{\sum_i y_i K(\mathbf{x}, \mathbf{x}_i)}{\sum_i K(\mathbf{x}, \mathbf{x}_i)}\right)$$

### Differences:

- SVMs:
  - Learn weights  $\alpha_i$  (and bandwidth)
  - Often sparse solution
- KR:
  - Fixed "weights", learn bandwidth <sup>1 estimate, guess</sup>
  - Solution may not be sparse
  - Much simpler to implement

# What's the difference between SVMs and Logistic Regression?

	SVMs	Logistic Regression
Loss function	Hinge loss 	Log-loss 
High dimensional features with kernels	Yes!	No <i>actually, yes...</i>

# Kernels in logistic regression

$$P(Y = 1 | x, \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)}}$$

*features in high d.*

- Define weights in terms of support vectors:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

$$P(Y = 1 | x, \mathbf{w}) = \frac{1}{1 + e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b)}}$$

$$= \frac{1}{1 + e^{-(\sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b)}}$$

*same idea*

- Derive simple gradient descent rule on  $\alpha_i$

©2005-2007 Carlos Guestrin

19

# What's the difference between SVMs and Logistic Regression? (Revisited)

*Optimization*

*QP, (specialized method)*

*conjugate gradient*

	<b>SVMs</b>	<b>Logistic Regression</b>
<b>Loss function</b>	Hinge loss	Log-loss
<b>High dimensional features with kernels</b>	Yes!	Yes!
<b>Solution sparse</b> <i>many <math>\alpha_i = 0</math></i>	Often yes! <i># of support vectors</i>	Almost always <u>no!</u> <i>every point is a support vector</i> <i>because of loss fun</i>
<b>Semantics of output</b> <i><math>w \cdot x + b</math></i>	"Margin" <i>"confidence"</i>	<u>Real probabilities</u> <i><math>P(Y=1 X) = 0.62</math></i>

©2005-2007 Carlos Guestrin

20

# What you need to know

- Dual SVM formulation
  - How it's derived
- The kernel trick
- Derive polynomial kernel
- Common kernels
- Kernelized logistic regression
- Differences between SVMs and logistic regression

©2005-2007 Carlos Guestrin

21

# Announcements

*HW2 solutions*

## ■ Midterm:

- Thursday Oct. 25th, Thursday 5-6:30pm, MM A14

- All content up to, and including SVMs and Kernels

- Not learning theory

*bring a calculator, no laptops, open book, notes, any on the website, no phones, no pds, etc.*

## ■ Midterm review:

- Tuesday, 5-6:30pm, location ~~100~~ *Wen 5409*

- You should read midterms for Spring 2006 and 2007 before the review session

- Then, you can ask about some of the questions in these midterms

©2005-2007 Carlos Guestrin

22

# PAC-learning, VC Dimension ~~and~~ ~~Margin-based Bounds~~

Machine Learning – 10701/15781  
Carlos Guestrin  
Carnegie Mellon University

October 22<sup>nd</sup>, 2007

©2005-2007 Carlos Guestrin

23

## What now...

- We have explored many ways of learning from data
- But...
  - How good is our classifier, really?
  - How much data do I need to make it "good enough"?

©2005-2007 Carlos Guestrin

24

## A simple setting...

- Classification
  - $m$  data points
  - Finite number of possible hypothesis (e.g., dec. trees of depth  $d$ )
- A learner finds a hypothesis  $h$  that is consistent with training data
  - Gets zero error in training –  $\text{error}_{\text{train}}(h) = 0$
- What is the probability that  $h$  has more than  $\epsilon$  true error?
  - $\text{error}_{\text{true}}(h) \geq \epsilon$

©2005-2007 Carlos Guestrin

25

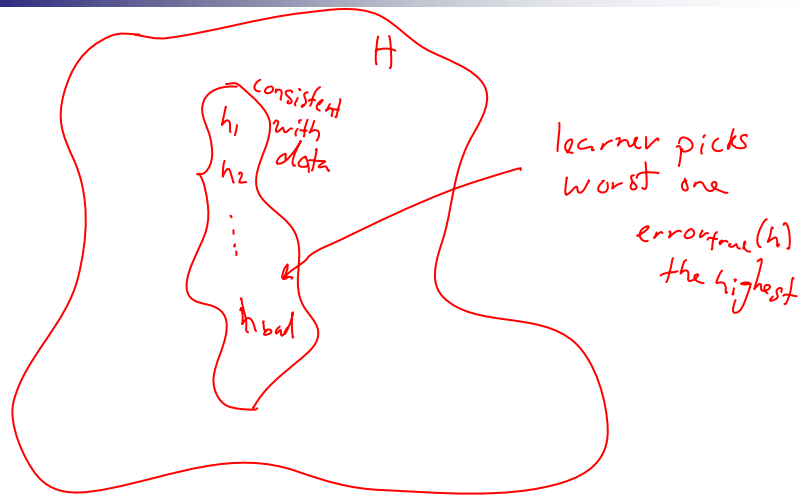
## How likely is a bad hypothesis to get $m$ data points right?

- Hypothesis  $h$  that is consistent with training data → got  $m$  i.i.d. points right
  - $h$  “bad” if it gets all this data right, but has high true error
- Prob.  $h$  with  $\text{error}_{\text{true}}(h) \geq \epsilon$  gets one data point right  
 $P(\text{hypothesis gets one point right}) \leq 1 - \epsilon$
- Prob.  $h$  with  $\text{error}_{\text{true}}(h) \geq \epsilon$  gets  $m$  data points right  
 $P(\text{hypothesis gets } m \text{ iid points right}) \leq (1 - \epsilon)^m$   
*exponentially small (as  $m$  increases)*

©2005-2007 Carlos Guestrin

26

But there are many possible hypothesis that are consistent with training data



©2005-2007 Carlos Guestrin

27

How likely is learner to pick a bad hypothesis

- Prob.  $h$  with  $\text{error}_{\text{true}}(h) \geq \epsilon$  gets  $m$  data points right

$$P(h_{\text{bad}} \text{ consistent with data}) \leq (1-\epsilon)^m$$

- There are  $k$  hypothesis consistent with data

- How likely is learner to pick a bad one?

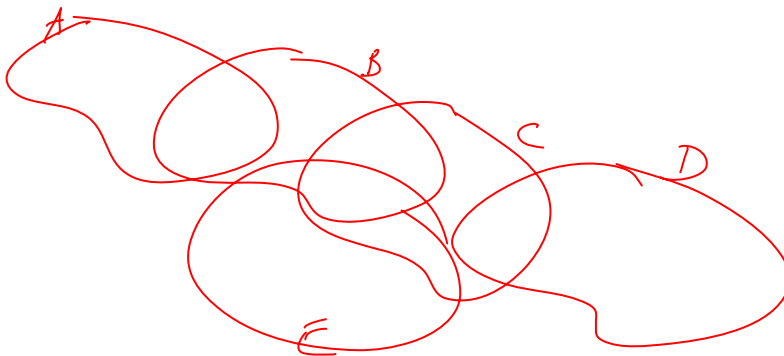
$$P(\exists h \text{ that is bad and consistent with data}) \\ = P(h_1 \text{ bad consistent} \vee h_2 \text{ bad consistent} \vee \dots \vee h_k \text{ bad consistent})$$

©2005-2007 Carlos Guestrin

28

## Union bound

- $P(A \text{ or } B \text{ or } C \text{ or } D \text{ or } \dots) \leq P(A) + P(B) + P(C) + \dots$



©2005-2007 Carlos Guestrin

29

## How likely is learner to pick a bad hypothesis

- Prob.  $h$  with  $\text{error}_{\text{true}}(h) \geq \epsilon$  gets  $m$  data points right

$$P(h \text{ bad, consistent}) \leq (1-\epsilon)^m$$

- There are  $k$  hypothesis consistent with data

- How likely is learner to pick a bad one?

$$P(\exists \text{ bad } h \text{ consistent with data}) \leq k (1-\epsilon)^m$$

$$\begin{aligned} &\leq |H| (1-\epsilon)^m \\ &\leq |H| e^{-m\epsilon} \end{aligned}$$

what's  $k$ ?  
 $k \leq |H|$   
 $\uparrow$  # of hypotheses

$(1-\epsilon)^m \leq e^{-m\epsilon}$

©2005-2007 Carlos Guestrin

30

# Review: Generalization error in finite hypothesis spaces [Hausler '88]

■ **Theorem:** Hypothesis space  $H$  finite, dataset  $D$  with  $m$  i.i.d. samples,  $0 < \epsilon < 1$ : for any learned hypothesis  $h$  that is consistent on the training data:

$$P(\text{error}_{\text{true}}(h) \geq \epsilon) \leq |H|e^{-m\epsilon}$$

