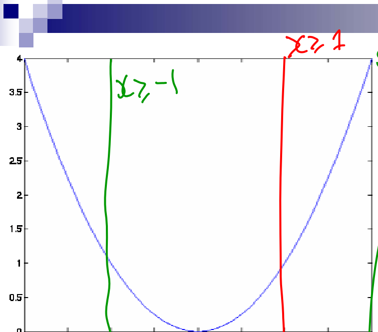
SVMs, Duality and the Kernel Trick, cont.

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

October 22nd, 2007

Lagrange multipliers – Dual variables



Solving: $\min_x \max_\alpha \; \underbrace{x^2 - \alpha(x-b)}_{L=}$

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 $\alpha = -2 < 0$

$x = 0$

no way I can set $\frac{\partial L}{\partial \alpha} = 0$

$\Rightarrow$ ignore $\alpha = 0$

constraint irrelevant

Dual SVM formulation – the non-separable case

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

$$\sum_i \alpha_i y_i = 0$$
$$\underbrace{C >}_{\text{only difference}} \alpha_i \geq 0$$

$\frac{\partial L}{\partial w}$

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

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

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

α_j can't be too big

α_j very large "when you really constraint problem"

"large α_j"

deriviation of dual for problem with slack penalty similar

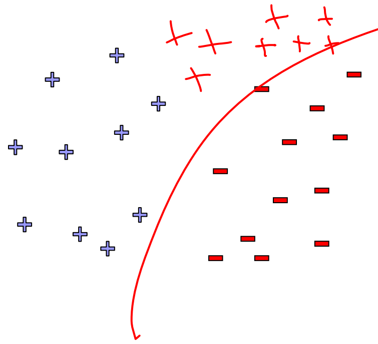
Why did we learn about the dual SVM?

- 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

$w^T \phi(x)$



Use features of features of features of features….

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

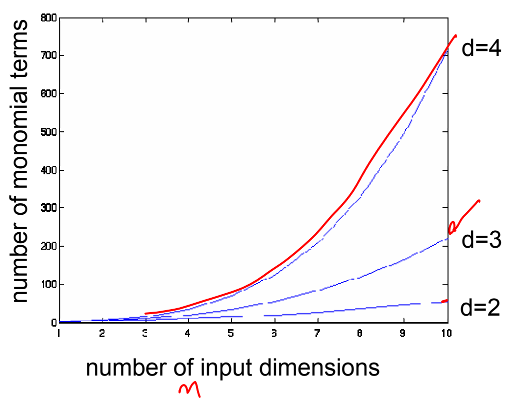
$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^{13} \\ x_2^{12} \\ \sin x_1 \cdot e^{x_2} \\ \vdots \end{pmatrix}$$

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**!

$x_i \to \mathbb{R}^m$

$x_j \to \mathbb{R}^m$

1x1 scalar

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

$$\sum_i \alpha_i y_i = 0$$

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

$$\underbrace{x_i x_j}_{\text{scalar}} = \langle x_i, x_j \rangle = \sum_{v=1}^{m} x_i^{(v)} \cdot x_j^{(v)}$$

kernel

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$ dot product

$$\sum_i \alpha_i y_i = 0$$

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

Dot-product of polynomials

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

$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ ← 2dimensional

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

$d=2 \quad \phi(u) \cdot \phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2^2 \\ u_2 u_1 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2^2 \\ v_2 v_1 \end{pmatrix} = u_1^2 v_1^2 + u_1 u_2 v_1 v_2 + u_2^2 v_2^2 + u_2 u_1 v_2 v_1$

$= (u_1 v_1)^2 + (u_2 v_2)^2 + 2\, u_1 v_1 \cdot u_2 v_2 = (u_1 v_1 + u_2 v_2)^2$

$= (uv)^2$

$d = m,$

$\phi(u) \cdot \phi(v)$

$= (u \cdot v)^m$

$\vdots$

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

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

$$\sum_i \alpha_i y_i = 0$$

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

- 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

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

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

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

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: $\phi(u) \cdot \phi(v) = \sum_{i=0}^{d} (u \cdot v)^i$
  - Better solution:

$$(uv)^1 + (uv)^2 + (uv)^0 + (v.u)^1 = (uv+1)^2$$

poly degree up to including d : $K(u,v) = (u.v+1)^d$

Common kernels

- Polynomials of degree d $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$
- Polynomials of degree up to d $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$

$\phi(u) \rightarrow$ 1.6 billion dimensions

squared-exponential

- Gaussian kernels $K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right)$ bandwidth

$\phi(u)$ = infinite dimensional

- Sigmoid $K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$

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

What about at classification time

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

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

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

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

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

need

test point

$$w.\phi(x) = \left(\sum_i \alpha_i y_i \phi(x_i)\right) \cdot \phi(x)$$

training data

$$= \sum_i \alpha_i y_i \phi(x_i) \cdot \phi(x) = \sum_{i=1} \alpha_i y_i K(x, x_i)$$

SVMs with kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors α_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 $C > \alpha_k > 0$

test case x

Classify as $sign\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right)$

Remember kernel regression

Remember kernel regression??? Gaussian kernel

1. ***$w_i = exp(-D(x_i, query)^2 / K_w^2)$***
2. *How to fit with the local points?*
   Predict the weighted average of the outputs:
   predict = $\Sigma w_i y_i / \Sigma w_i$

$K(X, x_i)$

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

©2005-2007 Carlos Guestrin 15

SVMs v. Kernel Regression

SVMs

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

or

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

"learn" from data

Kernel Regression

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

same "type" of classifier

©2005-2007 Carlos Guestrin 16

SVMs v. Kernel Regression

SVMs

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

or

$sign$

Kernel Regression

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

Differences:

- SVMs:
  - Learn weights α_i (and bandwidth)
  - Often sparse solution
- KR:
  - 1   estimate, guess
  - Fixed "weights", learn bandwidth
  - 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 actually, yes...

Kernels in logistic regression

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

features in high d.

- Define weights in terms of support vectors:

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

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

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

Same idea

- Derive simple gradient descent rule on α_i

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

Optimization | QP, (specialized method) | Conjugate gradient

	SVMs	Logistic Regression
Loss function	Hinge loss 0	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse many $\alpha_i = 0$	Often yes! # of support vectors	$\alpha_i > 0\ \forall i$ every point is a support vector Almost always no! because of loss fun never = 0
Semantics of output $w.x+b$	"Margin" "confidence"	Real probabilities $P(Y=1 \mid X) = 0.62$

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

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
    - open book, notes, any on the website, no laptops, no phones, no pdas, etc.
- Midterm review:
  - Tuesday, 5-6:30pm, location ~~TBD~~ Wean 5409
    - You should read midterms for Spring 2006 and Spring 2007 **before** the review session
    - Then, you can ask about some of the questions in these midterms

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

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

October 22nd, 2007

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"?

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 – $error_{train}(h) = 0$
- What is the probability that *h* has more than ε true error?
  - $error_{true}(h) \geq \varepsilon$

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 $error_{true}(h) \geq \varepsilon$ gets one data point right

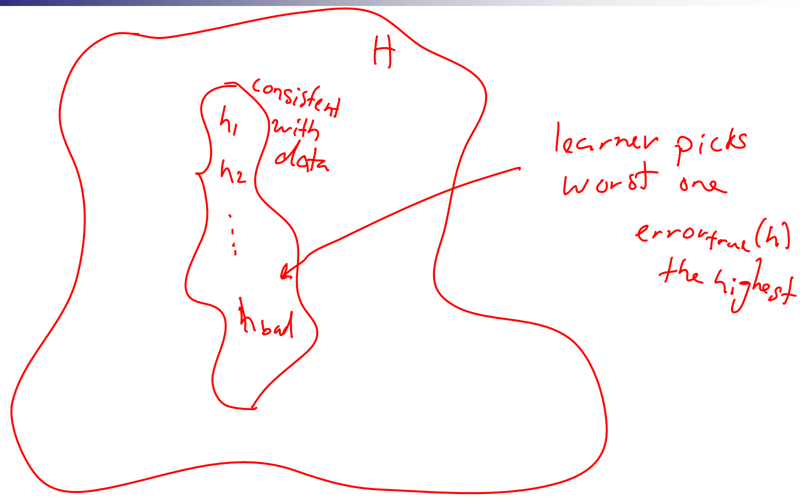
$P(h_{bad}\text{ gets one point right}) \leq 1-\varepsilon$

- Prob. *h* with $error_{true}(h) \geq \varepsilon$ gets *m* data points right

$P(h_{bad}\text{ gets } m \text{ iid points right}) \leq (1-\varepsilon)^m$

exponentially small (as m increases)

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



How likely is learner to pick a bad hypothesis

- Prob. *h* with $error_{true}(h) \geq \varepsilon$ gets *m* data points right

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

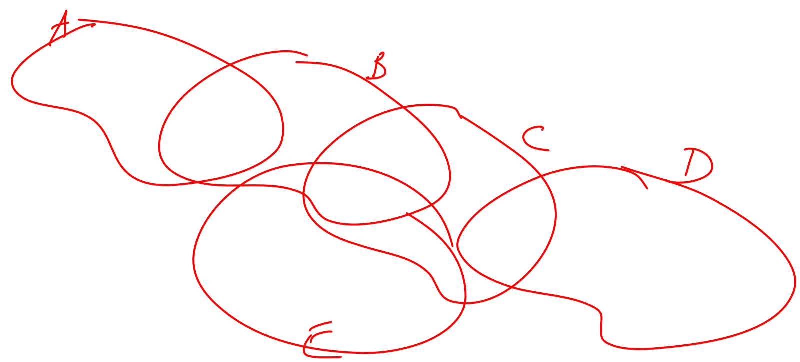
- There are *k* hypothesis consistent with data
  - How likely is learner to pick a bad one?

$P(\exists h \text{ that is bad and consistent with data})$

$= P(h_1 \text{ bad, consistent} \vee h_2 \text{ bad consistent} \vee \ldots \vee h_k \text{ bad \& consistent})$

Union bound

- P(A or B or C or D or …) $\leq P(A) + P(B) + P(C) + \ldots$



How likely is learner to pick a bad hypothesis

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

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

- There are k hypothesis consistent with data
  - How likely is learner to pick a bad one?

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

$\leq |H|\,(1-\varepsilon)^m$

$\leq |H|\, e^{-m\varepsilon}$

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

what's k?

$k \leq |H|$

\# of hypothesis

Review: Generalization error in finite hypothesis spaces [Haussler ’88]

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

you give me ε

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