SVMs, Duality and the Kernel Trick (cont.)

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University
October 26th, 2009

Finally: the "kernel trick"!

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

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

$$\sum_i \alpha_i y_i = 0$$

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

$$\boxed{\begin{array}{c} \mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \\ b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k) \\ \text{for any } k \text{ where } C > \alpha_k > 0 \end{array}}$$

- 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

Common kernels

- Polynomials of degree d

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

- Polynomials of degree up to d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Gaussian kernels

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

- 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 Φ(**x**), we are in trouble!
- Recall classifier: sign(**w**.Φ(**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 $C > \alpha_k > 0$

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 $C > \alpha_k > 0$

Classify as

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

Remember kernel regression

Remember kernel regression???

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 = $\Sigma w_i y_i / \Sigma w_i$

SVMs v. Kernel Regression

SVMs

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

or

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

Kernel Regression

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

SVMs v. Kernel Regression

SVMs

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

or

$sign$

Kernel Regression

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

Differences:

- SVMs:
  - Learn weights α_i (and bandwidth)
  - Often sparse solution
- KR:
  - 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		
High dimensional features with kernels		

Kernels in logistic regression

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

- Define weights in terms of support vectors:

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

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

- Derive simple gradient descent rule on α_i

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

	SVMs	**Logistic Regression**
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

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

PAC-learning, VC Dimension

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University
October 26th, 2009

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

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

- Hypothesis *h* that is **consistent** with training data $\rightarrow$ got *m* i.i.d. points right
  - h "bad" if it gets all this data right, but has high true error
- Prob. *h* with $\text{error}_{true}(h) \geq \varepsilon$ gets one data point right
- Prob. *h* with $\text{error}_{true}(h) \geq \varepsilon$ gets *m* data points right

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

Union bound

- P(A or B or C or D or …)

How likely is learner to pick a bad hypothesis

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

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

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:

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

Using a PAC bound

- Typically, 2 use cases: $P(\text{error}_{true}(h) > \epsilon) \leq |H|e^{-m\epsilon}$
  - 1: Pick ε and δ, give you *m*
  - 2: Pick m and δ, give you ε

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

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

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

Even if *h* makes zero errors in training data, may make errors in test

Limitations of Haussler '88 bound

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

- Consistent classifier

- Size of hypothesis space

What if our classifier does not have zero error on the training data?

- A learner with zero training errors may make mistakes in test set
- What about a learner with $error_{train}(h)$ in training set?

Simpler question: What's the expected error of a hypothesis?

- The error of a hypothesis is like estimating the parameter of a coin!
- Chernoff bound: for *m* i.i.d. coin flips, $x_1,\ldots,x_m$, where $x_i \in \{0,1\}$. For $0<\varepsilon<1$:

$$\boxed{P\left(\theta - \frac{1}{m}\sum_i x_i > \epsilon\right) \leq e^{-2m\epsilon^2}}$$

Using Chernoff bound to estimate error of a single hypothesis

$$\boxed{P\left(\theta - \frac{1}{m}\sum_i x_i > \epsilon\right) \leq e^{-2m\epsilon^2}}$$

But we are comparing many hypothesis: **Union bound**

For each hypothesis h_i:

$$P\left(\text{error}_{true}(h_i) - \text{error}_{train}(h_i) > \epsilon\right) \leq e^{-2m\epsilon^2}$$

What if I am comparing two hypothesis, h_1 and h_2?

Generalization bound for |H| hypothesis

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

$$P\left(\text{error}_{true}(h) - \text{error}_{train}(h) > \epsilon\right) \leq |H| e^{-2m\epsilon^2}$$

PAC bound and Bias-Variance tradeoff

$$P\left(\text{error}_{true}(h) - \text{error}_{train}(h) > \epsilon\right) \leq |H|e^{-2m\epsilon^2}$$

or, after moving some terms around, with probability at least 1-δ:

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

- **Important: PAC bound holds for all *h*, but doesn't guarantee that algorithm finds best *h*!!!**