

What's learning, revisited

Overfitting

Bayes optimal classifier

Naïve Bayes

Machine Learning – 10701/15781

Carlos Guestrin

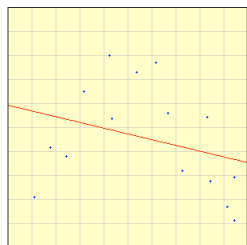
Carnegie Mellon University

January 24th, 2007

©Carlos Guestrin 2005-2007

Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
 - More complex class → less bias
 - More complex class → more variance



Select points by clicking on the graph or press [Example](#)

Degree of polynomial: 1 ☒ Fit Y to X
☐ Fit X to Y

[Calculate](#) [View Polynomial](#) [Reset](#)



Select points by clicking on the graph or press [Example](#)

Degree of polynomial: 13 ☒ Fit Y to X
☐ Fit X to Y

[Calculate](#) [View Polynomial](#) [Reset](#)



Select points by clicking on the graph or press [Example](#)

Degree of polynomial: 13 ☒ Fit Y to X
☐ Fit X to Y

[Calculate](#) [View Polynomial](#) [Reset](#)

©Carlos Guestrin 2005-2007

Training set error

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

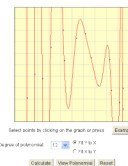
- Given a dataset (Training data)
- Choose a loss function
 - e.g., squared error (L_2) for regression
- **Training set error:** For a particular set of parameters, loss function on training data:

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

©Carlos Guestrin 2005-2007

Training set error as a function of model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$



©Carlos Guestrin 2005-2007

Prediction error

- Training set error can be poor measure of “quality” of solution
- **Prediction error:** We really care about error over all possible input points, not just training data:

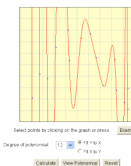
$$\begin{aligned} error_{true}(\mathbf{w}) &= E_{\mathbf{x}} \left[\left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 \right] \\ &= \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 d\mathbf{x} \end{aligned}$$

©Carlos Guestrin 2005-2007



Prediction error as a function of model complexity

$$\begin{aligned} error_{train}(\mathbf{w}) &= \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 \\ error_{true}(\mathbf{w}) &= \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 d\mathbf{x} \end{aligned}$$



©Carlos Guestrin 2005-2007

Computing prediction error

- Computing prediction

- hard integral
- May not know $t(\mathbf{x})$ for every \mathbf{x}

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 d\mathbf{x}$$

- Monte Carlo integration (sampling approximation)

- Sample a set of i.i.d. points $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ from $p(\mathbf{x})$
- Approximate integral with sample average

$$error_{true}(\mathbf{w}) \approx \frac{1}{M} \sum_{j=1}^M \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

©Carlos Guestrin 2005-2007

Why training set error doesn't approximate prediction error?

- Sampling approximation of prediction error:

$$error_{true}(\mathbf{w}) \approx \frac{1}{M} \sum_{j=1}^M \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

- Training error :

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

- Very similar equations!!!

- Why is training set a bad measure of prediction error???

©Carlos Guestrin 2005-2007

Why training set error doesn't approximate prediction error?

Because you cheated!!!

Training error good estimate for a single \mathbf{w} ,
But you optimized \mathbf{w} with respect to the training error,
and found \mathbf{w} that is good for this set of samples

**Training error is a (optimistically) biased
estimate of prediction error**

- Very similar equations!!!
 - Why is training set a bad measure of prediction error???

©Carlos Guestrin 2005-2007

Test set error

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

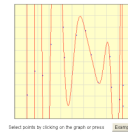
- Given a dataset, **randomly** split it into two parts:
 - Training data – $\{\mathbf{x}_1, \dots, \mathbf{x}_{N_{\text{train}}}\}$
 - Test data – $\{\mathbf{x}_1, \dots, \mathbf{x}_{N_{\text{test}}}\}$
- Use training data to optimize parameters \mathbf{w}
- **Test set error:** For the *final solution* \mathbf{w}^* , evaluate the error using:

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

©Carlos Guestrin 2005-2007

Test set error as a function of model complexity

$$\begin{aligned} error_{train}(\mathbf{w}) &= \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 \\ error_{true}(\mathbf{w}) &= \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 d\mathbf{x} \\ error_{test}(\mathbf{w}) &= \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 \end{aligned}$$



©Carlos Guestrin 2005-2007

How many points to I use for training/testing?

- Very hard question to answer!
 - Too few training points, learned \mathbf{w} is bad
 - Too few test points, you never know if you reached a good solution
- Bounds, such as Hoeffding's inequality can help:

$$P(|\hat{\theta} - \theta^*| \geq \epsilon) \leq 2e^{-2N\epsilon^2}$$

- More on this later this semester, but still hard to answer
- Typically:
 - if you have a reasonable amount of data, pick test set "large enough" for a "reasonable" estimate of error, and use the rest for learning
 - if you have little data, then you need to pull out the big guns...
 - e.g., bootstrapping

©Carlos Guestrin 2005-2007

Error estimators

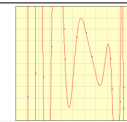
$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 d\mathbf{x}$$

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

©Carlos Guestrin 2005-2007

Error as a function of number of training examples for a fixed model complexity



$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

little data

infinite data

©Carlos Guestrin 2005-2007

Error estimators

Be careful!!!

Test set only unbiased if you never never never never do any any any any learning on the test data

For example, if you use the test set to select the degree of the polynomial... no longer unbiased!!!
(We will address this problem later in the semester)

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

©Carlos Guestrin 2005-2007

Overfitting

- **Overfitting:** a learning algorithm overfits the training data if it outputs a solution \mathbf{w} when there exists another solution \mathbf{w}' such that:

$$[error_{train}(\mathbf{w}) < error_{train}(\mathbf{w}')] \wedge [error_{true}(\mathbf{w}') < error_{true}(\mathbf{w})]$$

©Carlos Guestrin 2005-2007

Announcements

- First homework is out today:
 - Programming part and Analytic part
 - Remember collaboration policy: can discuss questions, but need to write your own solutions and code
 - Remember you are not allowed to look at previous years' solutions, search the web for solutions, use someone else's solutions, etc.
 - Due Mon. Feb 7th **beginning of class**
 - Start early!

©Carlos Guestrin 2005-2007

What's (supervised) learning, more formally

- Given:
 - **Dataset:** Instances $\{\langle \mathbf{x}_1; t(\mathbf{x}_1) \rangle, \dots, \langle \mathbf{x}_N; t(\mathbf{x}_N) \rangle\}$
 - e.g., $\langle \mathbf{x}_i; t(\mathbf{x}_i) \rangle = \langle (\text{GPA}=3.9, \text{IQ}=120, \text{MLscore}=99); 150K \rangle$
 - **Hypothesis space:** H
 - e.g., polynomials of degree 8
 - **Loss function:** measures quality of hypothesis $h \in H$
 - e.g., squared error for regression
- Obtain:
 - **Learning algorithm:** obtain $h \in H$ that minimizes loss function
 - e.g., using matrix operations for regression
 - Want to minimize prediction error, but can only minimize error in dataset

©Carlos Guestrin 2005-2007

Types of (supervised) learning problems, revisited

- **Regression**, e.g.,
 - **dataset**: $\langle \text{position}; \text{temperature} \rangle$
 - **hypothesis space**:
 - **Loss function**:
- **Density estimation**, e.g.,
 - **dataset**: $\langle \text{grades} \rangle$
 - **hypothesis space**:
 - **Loss function**:
- **Classification**, e.g.,
 - **dataset**: $\langle \text{brain image}; \{\text{verb v. noun}\} \rangle$
 - **hypothesis space**:
 - **Loss function**:

©Carlos Guestrin 2005-2007

Learning is (simply) function approximation!

- The general (supervised) learning problem:
 - Given some data (including features), hypothesis space, loss function
 - Learning is no magic!
 - Simply trying to find a function that fits the data
- **Regression**
- **Density estimation**
- **Classification**
- (Not surprisingly) Seemly different problem, very similar solutions...

©Carlos Guestrin 2005-2007

Classification

- **Learn:** $h: \mathbf{X} \mapsto Y$

- \mathbf{X} – features
- Y – target classes

- Suppose you know $P(Y|\mathbf{X})$ exactly, how should you classify?

- Bayes classifier:

- **Why?**

©Carlos Guestrin 2005-2007

Optimal classification

- **Theorem:** Bayes classifier h_{Bayes} is optimal!

- That is $\text{error}_{\text{true}}(h_{\text{Bayes}}) \leq \text{error}_{\text{true}}(h), \forall h(\mathbf{x})$

- **Proof:** $p(\text{error}) = \int_{\mathbf{x}} p(\text{error}|\mathbf{x})p(\mathbf{x})d\mathbf{x}$

©Carlos Guestrin 2005-2007

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Which is shorthand for:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i) P(Y = y_i)}{P(X = x_j)}$$

©Carlos Guestrin 2005-2007

How hard is it to learn the optimal classifier?

■ Data =

Sky	Temp	Humid	Wind	Water	Forecast	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

■ How do we represent these? How many parameters?

□ Prior, $P(Y)$:

■ Suppose Y is composed of k classes

□ Likelihood, $P(\mathbf{X}|Y)$:

■ Suppose \mathbf{X} is composed of n binary features

■ **Complex model → High variance with limited data!!!**

©Carlos Guestrin 2005-2007

Conditional Independence

- X is **conditionally independent** of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z
($\forall i, j, k$) $P(X = i | Y = j, Z = k) = P(X = i | Z = k)$

- e.g., $P(\text{Thunder} | \text{Rain}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})$

- Equivalent to:

$$P(X, Y | Z) = P(X | Z)P(Y | Z)$$

©Carlos Guestrin 2005-2007

What if features are independent?

- Predict 10701Grade
- From two **conditionally Independent** features
 - ☐ HomeworkGrade
 - ☐ ClassAttendance

©Carlos Guestrin 2005-2007

The Naïve Bayes assumption

- Naïve Bayes assumption:

- Features are independent given class:

$$\begin{aligned}P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\ &= P(X_1|Y)P(X_2|Y)\end{aligned}$$

- More generally:

$$P(X_1 \dots X_n|Y) = \prod_i P(X_i|Y)$$

- How many parameters now?

- Suppose \mathbf{X} is composed of n binary features

©Carlos Guestrin 2005-2007

The Naïve Bayes Classifier

- Given:

- Prior $P(Y)$
 - n conditionally independent features \mathbf{X} given the class Y
 - For each X_i , we have likelihood $P(X_i|Y)$

- Decision rule:

$$\begin{aligned}y^* = h_{NB}(\mathbf{x}) &= \arg \max_y P(y)P(x_1, \dots, x_n | y) \\ &= \arg \max_y P(y) \prod_i P(x_i|y)\end{aligned}$$

- If assumption holds, NB is optimal classifier!

©Carlos Guestrin 2005-2007

MLE for the parameters of NB

- Given dataset
 - $\text{Count}(A=a, B=b) \leftarrow$ number of examples where $A=a$ and $B=b$
- MLE for NB, simply:
 - Prior: $P(Y=y) =$
 - Likelihood: $P(X_i=x_i | Y_i=y_i) =$

©Carlos Guestrin 2005-2007

Subtleties of NB classifier 1 – Violating the NB assumption

- Usually, features are not conditionally independent:

$$P(X_1 \dots X_n | Y) \neq \prod_i P(X_i | Y)$$

- Actual probabilities $P(Y|\mathbf{X})$ often biased towards 0 or 1
- Nonetheless, NB is the single most used classifier out there
 - NB often performs well, even when assumption is violated
 - [Domingos & Pazzani '96] discuss some conditions for good performance

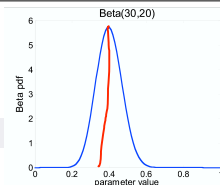
©Carlos Guestrin 2005-2007

Subtleties of NB classifier 2 – Insufficient training data

- What if you never see a training instance where $X_1=a$ when $Y=b$?
 - e.g., $Y=\{\text{SpamEmail}\}$, $X_1=\{\text{'Enlargement'}\}$
 - $P(X_1=a \mid Y=b) = 0$
- Thus, no matter what the values X_2, \dots, X_n take:
 - $P(Y=b \mid X_1=a, X_2, \dots, X_n) = 0$
- What now???

©Carlos Guestrin 2005-2007

MAP for Beta distribution



$$P(\theta \mid \mathcal{D}) = \frac{\theta^{\beta_H + \alpha_H - 1} (1 - \theta)^{\beta_T + \alpha_T - 1}}{B(\beta_H + \alpha_H, \beta_T + \alpha_T)} \sim \text{Beta}(\beta_H + \alpha_H, \beta_T + \alpha_T)$$

- MAP: use most likely parameter:

$$\hat{\theta} = \arg \max_{\theta} P(\theta \mid \mathcal{D}) = \frac{\beta_H + \alpha_H - 1}{\beta_H + \alpha_H + \beta_T + \alpha_T - 2}$$

$\alpha_H = 3$
 $\alpha_T = 2$
 β_H, β_T extra data

- Beta prior equivalent to extra thumbtack flips
- As $N \rightarrow \infty$, prior is “forgotten”
- **But, for small sample size, prior is important!**

©Carlos Guestrin 2005-2007

Bayesian learning for NB parameters – a.k.a. smoothing

- Dataset of N examples
- Prior
 - “distribution” $Q(X_i, Y)$, $Q(Y)$
 - m “virtual” examples
- MAP estimate
 - $P(X_i|Y)$
- Now, even if you never observe a feature/class, posterior probability never zero

©Carlos Guestrin 2005-2007

Text classification

- Classify e-mails
 - $Y = \{\text{Spam}, \text{NotSpam}\}$
- Classify news articles
 - $Y = \{\text{what is the topic of the article?}\}$
- Classify webpages
 - $Y = \{\text{Student, professor, project, ...}\}$
- What about the features \mathbf{X} ?
 - The text!

©Carlos Guestrin 2005-2007

Features \mathbf{X} are entire document – X_i for i^{th} word in article

Article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.e
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he's clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrdey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he's only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided

NB for Text classification

- $P(\mathbf{X}|Y)$ is huge!!!
 - Article at least 1000 words, $\mathbf{X}=\{X_1, \dots, X_{1000}\}$
 - X_i represents i^{th} word in document, i.e., the domain of X_i is entire vocabulary, e.g., Webster Dictionary (or more), 10,000 words, etc.
- NB assumption helps a lot!!!
 - $P(X_i=x_i|Y=y)$ is just the probability of observing word x_i in a document on topic y

$$h_{NB}(\mathbf{x}) = \arg \max_y P(y) \prod_{i=1}^{LengthDoc} P(x_i|y)$$

Bag of words model

- Typical additional assumption – **Position in document doesn't matter**: $P(X_i=x_i|Y=y) = P(X_k=x_i|Y=y)$

- “Bag of words” model – order of words on the page ignored
- Sounds really silly, but often works very well!

$$P(y) \prod_{i=1}^{LengthDoc} P(x_i|y)$$

When the lecture is over, remember to wake up the person sitting next to you in the lecture room.

©Carlos Guestrin 2005-2007

Bag of words model

- Typical additional assumption – **Position in document doesn't matter**: $P(X_i=x_i|Y=y) = P(X_k=x_i|Y=y)$

- “Bag of words” model – order of words on the page ignored
- Sounds really silly, but often works very well!

$$P(y) \prod_{i=1}^{LengthDoc} P(x_i|y)$$

in is lecture lecture next over person remember room
sitting the the the to to up wake when you

©Carlos Guestrin 2005-2007

Bag of Words Approach



©Carlos Guestrin 2005-2007

NB with Bag of Words for text classification

■ Learning phase:

- Prior $P(Y)$
 - Count how many documents you have from each topic (+ prior)
- $P(X_i|Y)$
 - For each topic, count how many times you saw word in documents of this topic (+ prior)

■ Test phase:

- For each document
 - Use naïve Bayes decision rule

$$h_{NB}(\mathbf{x}) = \arg \max_y P(y) \prod_{i=1}^{LengthDoc} P(x_i|y)$$

©Carlos Guestrin 2005-2007

Twenty News Groups results

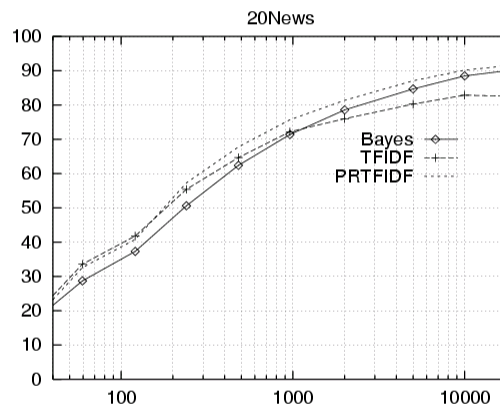
Given 1000 training documents from each group
Learn to classify new documents according to
which newsgroup it came from

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	
talk.politics.guns	

Naive Bayes: 89% classification accuracy

©Carlos Guestrin 2005-2007

Learning curve for Twenty News Groups



Accuracy vs. Training set size (1/3 withheld for test)

©Carlos Guestrin 2005-2007

What you need to know

- Training/test/true errors
 - Biased v. unbiased error estimate
 - Never train on the test data!!! (Even if you think you are not doing it)
- Types of learning problems
 - Learning is (just) function approximation!
- Optimal decision using Bayes Classifier
- Naïve Bayes classifier
 - What's the assumption
 - Why we use it
 - How do we learn it
 - Why is Bayesian estimation important
- Text classification
 - Bag of words model

©Carlos Guestrin 2005-2007