What's learning, revisited
Overfitting
Bayes optimal classifier
Naïve Bayes

Machine Learning – 10701/15781

Carlos Guestrin

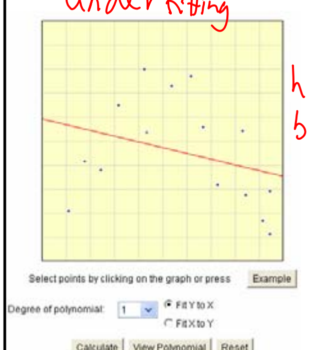
Carnegie Mellon University

January 24th, 2007

Bias-Variance Tradeoff

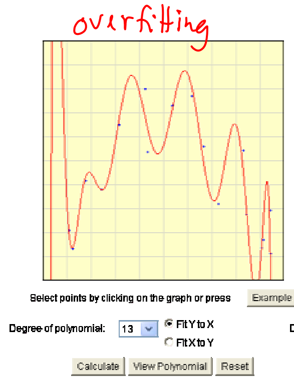
- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance

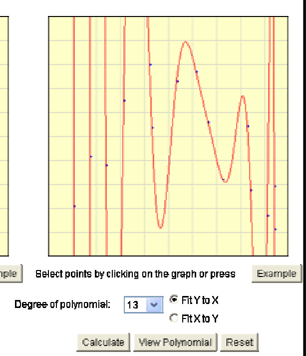
under fitting



high bias

overfitting





Training set error

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

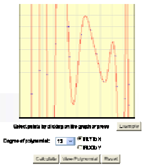
- Given a dataset (Training data)
- Choose a loss function
  - e.g., squared error (L_2) for regression
- **Training set error:** For a particular set of parameters, loss function on training data:

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

Training set error as a function of model complexity

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

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:

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

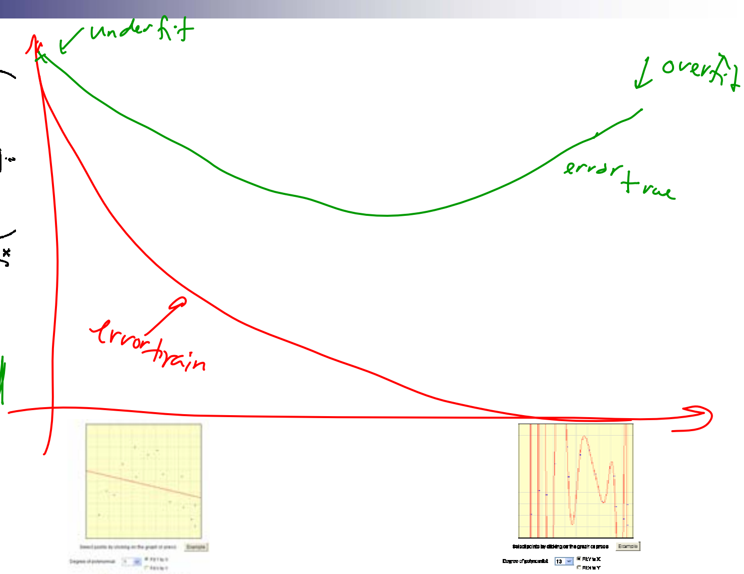
$$= \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2 p(\mathbf{x})\, d\mathbf{x}$$

don't have $t(x)$, $\forall x$,

Prediction error as a function of model complexity

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

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



Computing prediction error

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

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

- Monte Carlo integration (sampling approximation)
  - Sample a set of i.i.d. points $\{\mathbf{x}_1,\ldots,\mathbf{x}_M\}$ from $p(\mathbf{x})$
  - Approximate integral with sample average

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

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???

Why training set error doesn't approximate prediction error?

Because you cheated!!!

Training error good estimate for a single **w,**
But you optimized **w** with respect to the training error, and found **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???

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 – $\{\mathbf{x}_1, \ldots, \mathbf{x}_{Ntrain}\}$
  - Test data – $\{\mathbf{x}_1, \ldots, \mathbf{x}_{Ntest}\}$
- Use training data to optimize parameters **w**
- **Test set error:** For the ***final solution*** **w***, evaluate the error using:

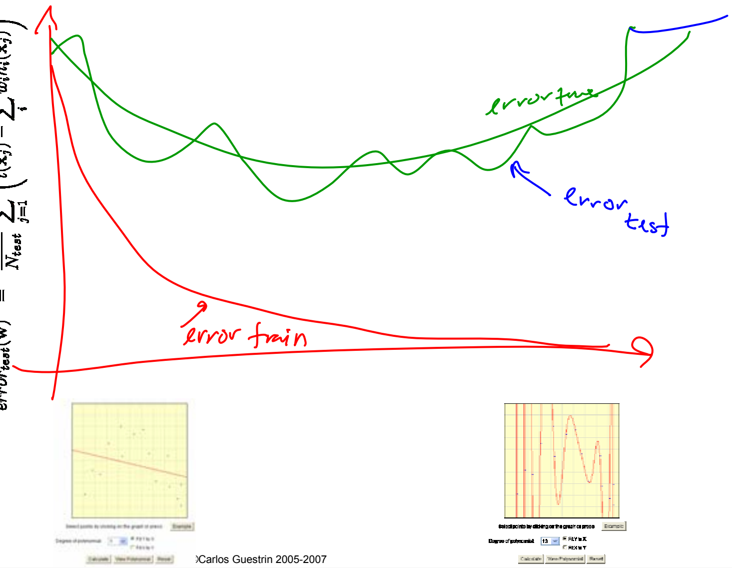
$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

Test set error as a function of model complexity

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

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

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



How many points to I use for training/testing?

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

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

- 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

Error estimators

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

gold standard
unbiased

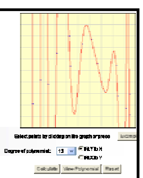
$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$

use to learn
optimistic

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

approx. truth
unbiased (if you are careful)

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

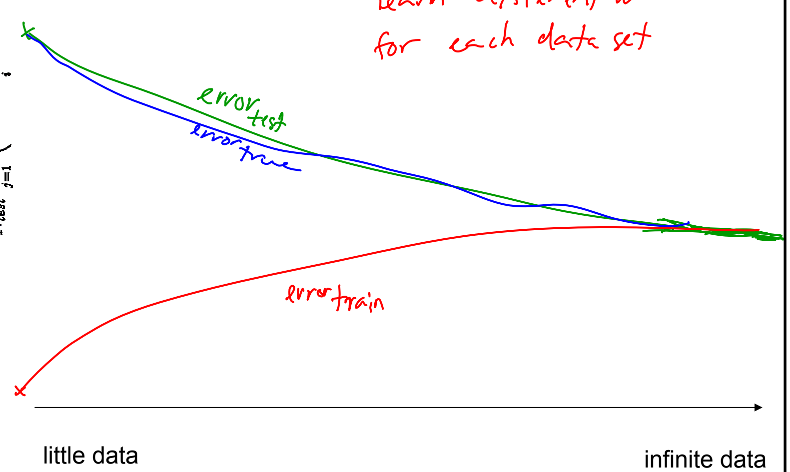


= 13 basis functions

learn different w for each data set

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

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



Error estimators

er

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)

erro

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

Overfitting

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

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

w great in train but bad in test
w' worst in train but better in test

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.~~ Wed. Feb 7th **beginning of class**
  - Start early!

What's (supervised) learning, more formally

- Given:
  - **Dataset**: Instances $\{\langle \mathbf{x}_1;t(\mathbf{x}_1)\rangle,\ldots, \langle \mathbf{x}_N;t(\mathbf{x}_N)\rangle\}$
    - e.g., $\langle \mathbf{x}_i;t(\mathbf{x}_i)\rangle = \langle \underbrace{\text{(GPA=3.9,IQ=120,MLscore=99)}}_{x_i};\underbrace{\text{150K}}_{t(x_i)}\rangle$
  - **Hypothesis space**: *H*
    - e.g., polynomials of degree 8
  - **Loss function**: measures quality of hypothesis $h \in H$
    - e.g., squared error for linear 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
      t-min

Types of (supervised) learning problems, revisited

- **Regression**, e.g., $f(x)$
  - **dataset**: ⟨position; temperature⟩ $\langle (1,3), 27 \rangle$
  - **hypothesis space**: Polynomials degree 8
  - **Loss function**: Squared error

$\langle 99 \rangle$ 98 79 98.5

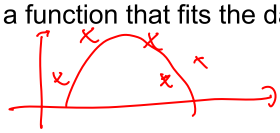
- **Density estimation**, e.g., P(Grade)
  - **dataset**: ⟨grades⟩
  - **hypothesis space**: Gaussians
  - **Loss function**: $-P(D \mid \mu, \sigma)$ — -likelihood

- **Classification**, e.g., discrete
  - **dataset**: ⟨brain image; {verb v. noun}⟩
  - **hypothesis space**: Naive Bayes classifiers
  - **Loss function**: (# mistakes) in train data
    error func optimize

Learning is (simply) function approximation!

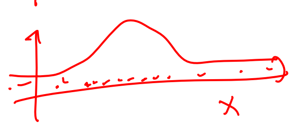
- The general (supervised) learning problem:
  - Given some data (including features), hypothesis space, loss function
  - Learning is no magic!
  - Simply trying to find a function that fits the data

- **Regression**



$f: X \mapsto \mathbb{R}$

- **Density estimation**



$f: X \mapsto [0,1]$

$\int_X f(x)\,dx = 1$

- **Classification**

$f: X \mapsto \{1, \dots, k\}$

- (Not surprisingly) Seemly different problem, very similar solutions…

Classification

- **Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes

X: < picture of brain>
Y = {verb, noun}
given brain image

- Suppose you know P(Y|**X**) exactly, how should you classify?
  - Bayes classifier: $y^*(x) \in \arg\max_y P(Y=y \mid X=x)$

- **Why?**

Optimal classification

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

if you know P(Y|X) exactly

$h_{Bayes}(x) = y^*(x) \in \arg\max_y P(Y=y \mid X=x)$

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

- **Proof**: $p(error_h) = \int_x p(error_h|x)p(x)dx$

$\min_h p(error_h|x) = \begin{cases} P(Y=f|x), & h(x)=t \\ P(Y=t|x), & h(x)=f \end{cases}$

$\equiv$ max prob get right
$\equiv$ guess answer with highest prob
$\equiv \arg\max_y P(Y=y \mid X=x)$

Bayes Rule

want

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

Which is shorthand for:

estimate by

likelihood prior

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

posterior

$\arg\max_y P(Y=y \mid X=x) = \arg\max_y P(X=x \mid Y=y) \cdot P(Y=y)$

How hard is it to learn the optimal classifier?

X Y

- Data =

Sky	Temp	Humid	Wind	Water	Forecst	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): $k-1$ ← probably ok
    - Suppose Y is composed of *k* classes
  - Likelihood, P(**X**|Y): $k(2^n-1)$ ← get you into trouble
    - Suppose **X** is composed of *n* binary features

$\#\text{param}(P(X \mid Y=y)) = 2^n - 1$ for each y

- Complex model → High variance with limited data!!!

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., $P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$

$T \perp R \mid L$ = Thunder independent of rain given lightning

- Equivalent to: $X \perp Y \mid Z$

$$P(X, Y \mid Z) = P(X \mid Z) P(Y \mid Z)$$

What if features are independent?

- Predict 10701Grade $= G$ ← class you want to predict
- From two **conditionally Independent** features $A \perp H \mid G$ ← Naïve Bayes
  - HomeworkGrade $= H$
  - ClassAttendance $= A$

$$P(G \mid H, A) \propto \underbrace{P(G)}_{\text{prior}} \; \underbrace{P(H, A \mid G)}_{\text{likehood}}$$

$P(H, A \mid G)$ ← # params = 6 = 2.(4-1) (binary)

NB assumption $= P(G) \; \underbrace{P(H \mid G) \cdot P(A \mid G)}_{\text{\# params} = 4 = 2.2.(2-1)}$

The Naïve Bayes assumption

- Naïve Bayes assumption: $X_1 \perp X_2 \mid Y$
  - Features are independent given class:

likelihood

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$
$$= P(X_1|Y)P(X_2|Y)$$

  - More generally: $\forall i \; X_i \perp \{X_1 \ldots X_{i-1}, X_{i+1}, \ldots X_n\} \mid Y$

$$\boxed{P(X_1 \ldots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)}$$

likelihood ↑ k values

binary

- How many parameters now?
  - Suppose **X** is composed of *n* binary features

$k.(2^n - 1)$ parameters

$n \cdot k \cdot (2-1) = nk$

$Z \equiv m$ values

$P(Z)$	z	$P(z)$
	1	0.1
	2	0.2
	⋮	⋮
	m	0.05

$\sum_j P(Z=j) = 1$

saves one parameter

The Naïve Bayes Classifier

- Given:
  - Prior P(Y)
  - *n* conditionally independent features **X** given the class Y
  - For each X_i, we have likelihood $P(X_i|Y)$
- Decision rule:

$$y^* = h_{NB}(\mathbf{x}) = \arg\max_y P(y)P(x_1, \ldots, x_n \mid y)$$
$$= \arg\max_y P(y) \prod_i P(x_i|y)$$

- If assumption holds, NB is optimal classifier!

MLE for the parameters of NB

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

Subtleties of NB classifier 1 – Violating the NB assumption

- Usually, features are not conditionally independent:

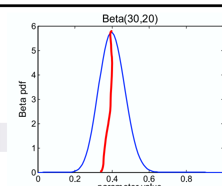
$$\boxed{P(X_1 \ldots X_n|Y) \neq \prod_i P(X_i|Y)}$$

- Actual probabilities P(Y|**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

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={SpamEmail}, X_1={'Enlargement'}
  - $P(X_1=a \mid Y=b) = 0$
- Thus, no matter what the values $X_2,\ldots,X_n$ take:
  - $P(Y=b \mid X_1=a, X_2,\ldots,X_n) = 0$
- What now???

MAP for Beta distribution



multinomial-like

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

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

- MAP: use most likely parameter:

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

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

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

Text classification

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

Features **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 (opinic
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 Hrudey 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 decidec
```

NB for Text classification

- P(**X**|Y) is huge!!!
  - Article at least 1000 words, **X**=$\{X_1,\ldots,X_{1000}\}$
  - 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.

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

aardvark	0
about	2
all	2
Africa	1
apple	0
anxious	0
...	
gas	1
...	
oil	1
…	
Zaire	0

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

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

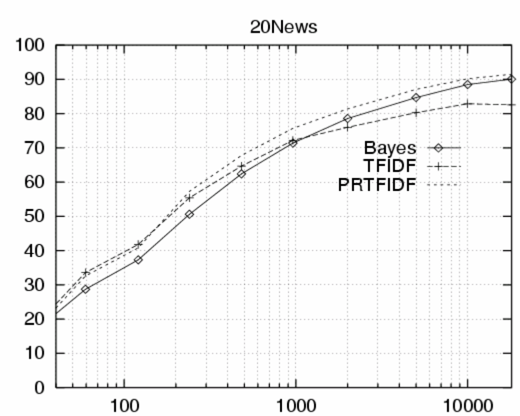
Twenty News Groups results

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

Learning curve for Twenty News Groups



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

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