Machine Learning

Bayesian Learning

Dr. Joschka Boedecker
AG Maschinelles Lernen und Natürlichsprachliche Systeme
Institut für Informatik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

jboedeck@informatik.uni-freiburg.de

Acknowledgement
Slides courtesy of Martin Riedmiller
Bayesian Learning

[Read Ch. 6]
[Suggested exercises: 6.1, 6.2, 6.6]

• Bayes Theorem
• MAP, ML hypotheses
• MAP learners
• Minimum description length principle
• Bayes optimal classifier
• Naive Bayes learner
• Example: Learning over text data
Two Roles for Bayesian Methods

Provides practical learning algorithms

• Naive Bayes learning
• Bayesian belief network learning
• Combine prior knowledge (prior probabilities) with observed data
• Requires prior probabilities

Provides useful conceptual framework

• Provides “gold standard” for evaluating other learning algorithms
• Additional insight into Occam’s razor
Remark on Conditional Probabilities and Priors

- $P((d_1, \ldots, d_m)|h)$: probability that a hypothesis $h$ generated a certain classification for a fixed input data set $(x_1, \ldots, x_m)$
- $P((x_1, \ldots, x_m)|\mu, \sigma^2)$ probability that input data set was generated by a Gaussian distribution with specific parameter values $\mu, \sigma$
- $= \text{Likelihood}$ of these values
- For a hypothesis $h$ (e.g., a decision tree) $P(h)$ should be seen as prior knowledge about hypothesis:
  - For instance: smaller trees are more probable than more complex trees
  - Or: uniform distribution, if no prior knowledge
- $\rightarrow \text{subjective probability} \approx$ probability as belief
Bayes Theorem

• In the following: fixed training set $x_1, \ldots, x_m$

• Classifications $D = (d_1, \ldots, d_m)$

• This allows to determine the most probable hypothesis given the data using Bayes theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

• $P(h)$ = prior probability of hypothesis $h$

• $P(D)$ = prior probability of $D$

• $P(h|D) = $ probability of $h$ given $D$

• $P(D|h)$ = probability of $D$ given $h$
Choosing Hypotheses

\[ P(h|D) = \frac{P(D|h)P(h)}{P(D)} \]

Generally want the most probable hypothesis given the training data

**Maximum a posteriori** hypothesis \( h_{MAP} \):

\[
    h_{MAP} = \arg \max_{h \in H} P(h|D) \\
    = \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\
    = \arg \max_{h \in H} P(D|h)P(h)
\]

If assume \( P(h_i) = P(h_j) \) then can further simplify, and choose the

**Maximum likelihood** (ML) hypothesis

\[
    h_{ML} = \arg \max_{h_i \in H} P(D|h_i)
\]
Basic Formulas for Probabilities

• **Product Rule**: probability $P(A \land B)$ of a conjunction of two events A and B:

$$P(A \land B) = P(A|B)P(B) = P(B|A)P(A)$$

• **Sum Rule**: probability of a disjunction of two events A and B:

$$P(A \lor B) = P(A) + P(B) - P(A \land B)$$

• **Theorem of total probability**: if events $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$
Brute Force MAP Hypothesis Learner

1. For each hypothesis $h$ in $H$, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Output the hypothesis $h_{MAP}$ with the highest posterior probability

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$
Relation to Concept Learning

Consider our usual concept learning task

- instance space $X$, hypothesis space $H$, training examples $D$

- consider the FINDS learning algorithm (outputs most specific hypothesis from the version space $VS_{H,D}$)

What would Bayes rule produce as the MAP hypothesis?
Relation to Concept Learning

Assume fixed set of instances \( \langle x_1, \ldots, x_m \rangle \)

Assume \( D \) is the set of classifications
\[
D = \langle c(x_1), \ldots, c(x_m) \rangle = \langle d_1, \ldots, d_m \rangle
\]

Choose \( P(D|h) \):
Relation to Concept Learning

Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$
Assume $D$ is the set of classifications $D = \langle c(x_1), \ldots, c(x_m) \rangle$
Choose $P(D|h)$

- $P(D|h) = 1$ if $h$ consistent with $D$
- $P(D|h) = 0$ otherwise

Choose $P(h)$ to be uniform distribution

- $P(h) = \frac{1}{|H|}$ for all $h$ in $H$

Then,

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$
Evolution of Posterior Probabilities

\[ P(h) \]

\[ P(h|D_1) \]

\[ P(h|D_1, D_2) \]
Characterizing Learning Algorithms by Equivalent MAP Learners

Inductive system

Training examples D → Candidate Elimination Algorithm → Output hypotheses

Hypothesis space H

Equivalent Bayesian inference system

Training examples D → Brute force MAP learner → Output hypotheses

Hypothesis space H

Prior assumptions made explicit

P(h) uniform
P(D|h) = 0 if inconsistent,
= 1 if consistent

Does $FindS$ output a MAP hypothesis? Yes, if $P(h)$ is chosen such it prefers more specific over more general hypothesis.
Consider any real-valued target function $f$
Training examples $\langle x_i, d_i \rangle$, where $d_i$ is noisy training value:
$$d_i = f(x_i) + e_i$$ and
$e_i$ is random variable (noise) drawn independently for each $x_i$
according to some Gaussian distribution with mean=0

Then, the maximum likelihood hypothesis $h_{ML}$ is the one that
minimizes the sum of squared errors:

$$h_{ML} = \arg \min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$$
Learning A Real Valued Function (cont’d)

Proof:

\[ h_{ML} = \arg\max_{h \in H} p(D|h) \]

\[ = \arg\max_{h \in H} \prod_{i=1}^{m} p(d_i|h) \]

\[ = \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}\left(\frac{d_i-h(x_i)}{\sigma}\right)^2} \]

Maximize logarithm of this instead...

\[ h_{ML} = \arg\max_{h \in H} \ln(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}\left(\frac{d_i-h(x_i)}{\sigma}\right)^2}) \]
\[ h_{ML} = \arg\max_{h \in H} \ln \left( \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left( \frac{d_i - h(x_i)}{\sigma} \right)^2} \right) \]

\[ = \arg\max_{h \in H} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left( \frac{d_i - h(x_i)}{\sigma} \right)^2 \]

\[ = \arg\max_{h \in H} \sum_{i=1}^{m} -\frac{1}{2} \left( \frac{d_i - h(x_i)}{\sigma} \right)^2 \]

\[ = \arg\max_{h \in H} \sum_{i=1}^{m} -(d_i - h(x_i))^2 \]

\[ = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2 \]
Learning to Predict Probabilities

- Training examples \( \langle x_i, d_i \rangle \), where \( d_i \) is 1 or 0
- Want to train neural network to output a probability given \( x_i \) (not only a 0 or 1)
- Example: predicting probability that (insert your favourite soccer team here) wins.
- How to do this?
  1. Building relative frequencies from training examples, train regression model
  2. Use different error function (shown here)

In this case we can show that

\[
h_{ML} = \arg \max_{h \in H} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))
\]
Situation: given p training examples \( \{(x_i, d_i)\}_{i=1}^{p} \). \( d_i \) are class labels, i.e. \( d_i \in \{0, 1\} \).

Idea: \( h(x) \overset{!}{=} P(c(x) = 1) \) (output value should equal to class probability of correct class \( c(x) \) given \( x \)).

ML-approach: maximize \( P(D|h) \).
\[
P(D|h) = P(x_1, d_1|h) \cdots P(x_p, d_p|h)
\]
\( x_i \) is independent from \( h \). Therefore (with product rule):
\[
P(x_i, d_i|h) = P(d_i|x_i, h) P(x_i|h) = P(d_i|x_i, h) P(x_i)
\]

What is \( P(d_i|x_i, h) \)? Recall: \( h(i) \) should compute probability for \( d_i \) being 1. Therefore

\[
P(d_i|x_i, h) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ 1 - h(x_i) & \text{if } d_i = 0 \end{cases}
\]

in short notation:
\[
P(d_i|x_i, h) = h(x_i)^{d_i} (1 - h(x_i))^{(1-d_i)}
\]

Therefore:
\[
P(D|h) = \prod_{i=1}^{p} P(x_i, d_i|h) = \prod_{i=1}^{p} P(d_i|x_i, h) P(x_i) = \prod_{i=1}^{p} h(x_i)^{d_i} (1 - h(x_i))^{(1-d_i)} P(x_i)
\]
Maximum-likelihood:
\[ h_{ML} = \arg\max_h P(D|h) = \arg\max_h \prod_{i=1}^p h(x_i)^{d_i} (1 - h(x_i))^{(1-d_i)} P(x_i) \]

taking logarithm finally yields:
\[ h_{ML} = \arg\max_h \sum_i^n d_i \ln(h(x_i)) + (1 - d_i) \ln(1 - h(x_i)) = \arg\min_h -\sum_i^n d_i \ln(h(x_i)) + (1 - d_i) \ln(1 - h(x_i)) \]

This expression is often termed the 'cross-entropy'-error function
$$h_{ML} = \arg \min_h - \sum_{i}^{n} d_i \ln(h(x_i)) + (1 - d_i) \ln(1 - h(x_i))$$

What does this mean for a machine learning setup? E.g. multilayer-perceptrons?

Use the 'cross-entropy' error function (instead of the usual mean-square error function) to learn probabilities of classification. Remark: fits particularly well to sigmoid activation function, since some terms cancel out then.
Minimum Description Length Principle

Occam’s razor: prefer the shortest hypothesis

MDL: prefer the hypothesis \( h \) that minimizes

\[
h_{MDL} = \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)
\]

where \( L_C(x) \) is the description length of \( x \) under encoding \( C \)
Example: $H = \text{decision trees}, \ D = \text{training data labels}$

- $L_{C_1}(h)$ is \# bits to describe tree $h$

- $L_{C_2}(D|h)$ is \# bits to describe $D$ given $h$
  
  - Note $L_{C_2}(D|h) = 0$ if examples classified perfectly by $h$. Need only describe exceptions

- Hence $h_{MDL}$ trades off tree size for training errors
Minimum Description Length Principle

\[ h_{MAP} = \arg \max_{h \in H} P(D|h)P(h) \]

\[ = \arg \max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \]

\[ = \arg \min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \quad (2) \]

Interesting fact from information theory:

The optimal (shortest expected coding length) code for an event with probability \( p \) is \( -\log_2 p \) bits.

So interpret (1):

- \( -\log_2 P(h) \) is length of \( h \) under optimal code
- \( -\log_2 P(D|h) \) is length of \( D \) given \( h \) under optimal code

→ prefer the hypothesis that minimizes

\[ \text{length}(h) + \text{length}(\text{misclassifications}) \]
Most Probable Classification of New Instances

So far we’ve sought the most probable hypothesis given the data $D$ (i.e., $h_{\text{MAP}}$)

Given new instance $x$, what is its most probable classification?

• $h_{\text{MAP}}(x)$ is not the most probable classification! Why?

Consider:

• Three possible hypotheses:

$$P(h_1|D) = 0.4, \quad P(h_2|D) = 0.3, \quad P(h_3|D) = 0.3$$

• Given new instance $x$,

$$h_1(x) = +, \quad h_2(x) = -, \quad h_3(x) = -$$

• What’s most probable classification of $x$?
Bayes Optimal Classifier

Bayes optimal classification:

\[
\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)
\]

'Optimal': No other classification method using the same hypothesis space and the same prior knowledge can outperform this method in average.
Bayes Optimal Classifier

\[ \text{arg max}_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) \]

Example:

\[ P(h_1|D) = .4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1 \]
\[ P(h_2|D) = .3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0 \]
\[ P(h_3|D) = .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0 \]

\[ \sum_{h_i \in H} P(+'|h_i)P(h_i|D) = .4 \quad \text{and} \quad \sum_{h_i \in H} P(-'|h_i)P(h_i|D) = .6 \]

Thus,

\[ \text{arg max}_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = '−' \]
Gibbs Classifier

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.

Gibbs algorithm:

1. Choose one hypothesis at random, according to $P(h|D)$

2. Use this to classify new instance

**Surprising fact:** Assume target concepts are drawn at random from $H$ according to priors on $H$. Then (Haussler et al, 1994):

$$E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$$

Suppose correct, uniform prior distribution over $H$, then

- Pick any hypothesis from VS, with uniform probability
- Its expected error no worse than twice Bayes optimal
Naive Bayes Classifier

Along with decision trees, neural networks, nearest nbr, one of the most practical learning methods.

When to use

• Moderate or large training set available

• Attributes that describe instances are conditionally independent given classification

Successful applications:

• Diagnosis

• Classifying text documents
Naive Bayes Classifier

Assume target function $f : X \to V$, where each instance $x$ described by attributes $\langle a_1, a_2 \ldots a_n \rangle$. Most probable value of $f(x)$ is:

$$
v_{MAP} = \arg\max_{v_j \in V} P(v_j|a_1, a_2 \ldots a_n)
$$

$$
v_{MAP} = \arg\max_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n|v_j)P(v_j)}{P(a_1, a_2 \ldots a_n)}
$$

$$
= \arg\max_{v_j \in V} P(a_1, a_2 \ldots a_n|v_j)P(v_j)
$$
Naive Bayes assumption:

\[ P(a_1, a_2 \ldots a_n|v_j) = P(a_1|v_j) P(a_2|v_j) \ldots P(a_n|v_j) = \prod_i P(a_i|v_j) \]

**cond. independence assumption:** individual features are independent given the class

('correct computation' example: 
\[ P(a_1, a_2, a_3|v_j) = P(a_1, a_2|a_3, v_j) P(a_3|v_j) = P(a_1|a_2, a_3, v_j) P(a_2|a_3, v_j) P(a_3|v_j) \]

using conditional independence assumption:
\[ = P(a_1|v_j) P(a_2|v_j) P(a_3|v_j) \]

**Naive Bayes classifier:** 
\[ v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j) \]
Naive Bayes Algorithm

Naive\_Bayes\_Learn(\textit{examples})

For each target value $v_j$

\[ \hat{P}(v_j) \leftarrow \text{estimate } P(v_j) \]

For each attribute value $a_i$ of each attribute $a$

\[ \hat{P}(a_i|v_j) \leftarrow \text{estimate } P(a_i|v_j) \]

Classify\_New\_Instance(\textit{x})

\[ v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j) \]
Naive Bayes: Example

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

How to derive $\hat{P}(v_j)$, $\hat{P}(a_i|v_j)$?

simply by counting, e.g.

$\hat{P}('yes') = \frac{9}{14}$

$\hat{P}(strong|'yes') = \frac{3}{9}$

Why is this easier than computing $\hat{P}(strong, rain, mild, normal, sunny|'yes')$?

Much less training examples of exactly that combination in the latter case.
Naive Bayes: Example

Consider *PlayTennis* again, and new instance

\[ \langle \text{Outlk} = \text{sun}, \text{Temp} = \text{cool}, \text{Humid} = \text{high}, \text{Wind} = \text{strong} \rangle \]

Want to compute:

\[
\begin{align*}
\nu_{NB} &= \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) \\
\end{align*}
\]

\[
\begin{align*}
P(y) \ P(\text{sun}|y) \ P(\text{cool}|y) \ P(\text{high}|y) \ P(\text{strong}|y) &= .005 \\
P(n) \ P(\text{sun}|n) \ P(\text{cool}|n) \ P(\text{high}|n) \ P(\text{strong}|n) &= .021 \\
\rightarrow \nu_{NB} &= n
\end{align*}
\]
Naive Bayes: Subtleties

1. Conditional independence assumption

\[ P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j) \]

is often violated (e.g: \( P(\text{word}_t = 'machine'|\text{word}_{t+1} = 'learning', \text{Author} = 'TomMitchell') \neq P(\text{word}_t = 'machine'|\text{Author} = 'TomMitchell') \))

- ...but it works surprisingly well anyway. Note don’t need estimated posteriors \( \hat{P}(v_j|x) \) to be correct; need only that

\[ \arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \arg\max_{v_j \in V} P(v_j)P(a_1 \ldots, a_n | v_j) \]

- see [Domingos & Pazzani, 1996] for analysis
- Naive Bayes posteriors often unrealistically close to 1 or 0
2. what if none of the training instances with target value $v_j$ have attribute value $a_i$? Then

$$
\hat{P}(a_i|v_j) = 0, \text{ and therefore... } \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0
$$

Typical solution is Bayesian estimate for $\hat{P}(a_i|v_j)$

$$
\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}
$$

where

- $n$ is number of training examples for which $v = v_j$,
- $n_c$ number of examples for which $v = v_j$ and $a = a_i$
- $p$ is prior estimate for $\hat{P}(a_i|v_j)$
- $m$ is weight given to prior (i.e. number of “virtual” examples)
Learning to Classify Text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text documents??
Learning to Classify Text

Target concept \textit{Interesting?} : \textit{Document} $\rightarrow \{+, -, \}$

1. Represent each document by vector of words
   - one attribute per word position in document

2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(\text{doc}|+)$
   - $P(\text{doc}|-)$

Naive Bayes conditional independence assumption

$$P(\text{doc}|v_j) = \prod_{i=1}^{\text{length(doc)}} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position $i$ is $w_k$, given $v_j$

one more assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$
Learn_naive_Bayes_text(Examples, V)

1. collect all words and other tokens that occur in Examples
   - Vocabulary ← all distinct words and other tokens in Examples

2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms
   - For each target value $v_j$ in $V$ do
     - $docs_j ←$ subset of Examples for which the target value is $v_j$
     - $P(v_j) ← \frac{|docs_j|}{|Examples|}$
     - $Text_j ←$ a single document created by concatenating all members of $docs_j$
     - $n ←$ total number of words in $Text_j$ (counting duplicate words multiple times)
     - for each word $w_k$ in Vocabulary
       * $n_k ←$ number of times word $w_k$ occurs in $Text_j$
       * $P(w_k|v_j) ← \frac{n_k+1}{n+|Vocabulary|}$
\textsc{classify\_naive\_bayes\_text}(\textit{Doc})

- \textit{positions} $\leftarrow$ all word positions in \textit{Doc} that contain tokens found in \textit{Vocabulary}

- Return $v_{NB}$, where

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in \text{positions}} P(a_i|v_j)$$
Twenty NewsGroups

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
Random guessing: 5%
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 decided
Learning Curve for 20 Newsgroups

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

- Probability theory offers a powerful framework to design and analyse learning methods
- Probabilistic analysis offers insight in learning algorithms
- Even if not directly manipulating probabilities, algorithms might be seen fruitfully in a probabilistic perspective