Machine Learning

- Up until now: how to reason in a model and how to make optimal decisions

- Machine learning: how to acquire a model on the basis of data / experience
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering)

This slide deck courtesy of Dan Klein at UC Berkeley
Example: Overfitting

\[ P(\text{features}, C = 2) \]

\[ P(C = 2) = 0.1 \]

\[ P(\text{on}|C = 2) = 0.8 \]

\[ P(\text{on}|C = 2) = 0.1 \]

\[ P(\text{off}|C = 2) = 0.1 \]

\[ P(\text{on}|C = 2) = 0.01 \]

\[ P(\text{features}, C = 3) \]

\[ P(C = 3) = 0.1 \]

\[ P(\text{on}|C = 3) = 0.8 \]

\[ P(\text{on}|C = 3) = 0.9 \]

\[ P(\text{off}|C = 3) = 0.7 \]

\[ P(\text{on}|C = 3) = 0.0 \]

\[ 2 \text{ wins!!} \]
Example: Overfitting

- Posteriors determined by *relative* probabilities (odds ratios):

\[
\frac{P(W|\text{ham})}{P(W|\text{spam})} \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
\]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>south-west</td>
<td>inf</td>
<td>screens</td>
<td>inf</td>
</tr>
<tr>
<td>nation</td>
<td>inf</td>
<td>minute</td>
<td>inf</td>
</tr>
<tr>
<td>morally</td>
<td>inf</td>
<td>guaranteed</td>
<td>inf</td>
</tr>
<tr>
<td>nicely</td>
<td>inf</td>
<td>$205.00</td>
<td>inf</td>
</tr>
<tr>
<td>extent</td>
<td>inf</td>
<td>delivery</td>
<td>inf</td>
</tr>
<tr>
<td>seriously</td>
<td>inf</td>
<td>signature</td>
<td>inf</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

What went wrong here?
Generalization and Overfitting

- Relative frequency parameters will **overfit** the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn’t mean we won’t see it at test time
  - Unlikely that every occurrence of “minute” is 100% spam
  - Unlikely that every occurrence of “seriously” is 100% ham
  - What about all the words that don’t occur in the training set at all?
  - In general, we can’t go around giving unseen events zero probability

- As an extreme case, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn’t **generalize** at all
  - Just making the bag-of-words assumption gives us some generalization, but isn’t enough

- To generalize better: we need to **smooth** or **regularize** the estimates
Estimation: Smoothing

- Maximum likelihood estimates:
  \[ P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}} \]
  \[ P_{ML}(r) = 1/3 \]

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it’s heads, what’s the estimate for \( P(\text{heads}) \)?
  - What if I flip 10 times with 8 heads?
  - What if I flip 10M times with 8M heads?

- Basic idea:
  - We have some prior expectation about parameters (here, the probability of heads)
  - Given little evidence, we should skew towards our prior
  - Given a lot of evidence, we should listen to the data
Estimation: Smoothing

- Relative frequencies are the maximum likelihood estimates

\[
\theta_{ML} = \arg \max_{\theta} P(X|\theta) = \arg \max_{\theta} \prod_{i} P_{\theta}(X_i)
\]

\[
P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}
\]

- In Bayesian statistics, we think of the parameters as just another random variable, with its own distribution

\[
\theta_{MAP} = \arg \max_{\theta} P(\theta|X) = \arg \max_{\theta} P(X|\theta)P(\theta)/P(X)
\]

\[
= \arg \max_{\theta} P(X|\theta)P(\theta)
\]
Estimation: Laplace Smoothing

- **Laplace’s estimate:**
  - Pretend you saw every outcome once more than you actually did

\[
P_{LAP}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}
\]

\[
= \frac{c(x) + 1}{N + |X|}
\]

- Can derive this as a MAP estimate with *Dirichlet priors*
Estimation: Laplace Smoothing

- Laplace's estimate (extended):
  - Pretend you saw every outcome \( k \) extra times
    \[
    P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}
    \]
  - What's Laplace with \( k = 0 \)?
  - \( k \) is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:
    \[
    P_{LAP,k}(x|y) = \frac{c(x, y) + k}{c(y) + k|X|}
    \]
    \[
    P_{LAP,0}(X) =
    \]
    \[
    P_{LAP,1}(X) =
    \]
    \[
    P_{LAP,100}(X) =
    \]
Estimation: Linear Interpolation

- In practice, Laplace often performs poorly for $P(X|Y)$:
  - When $|X|$ is very large
  - When $|Y|$ is very large

- Another option: linear interpolation
  - Also get $P(X)$ from the data
  - Make sure the estimate of $P(X|Y)$ isn’t too different from $P(X)$

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha) \hat{P}(x)$$

- What if $\alpha$ is 0? 1?
Better: Linear Interpolation

- Linear interpolation for conditional likelihoods
  - **Idea**: the conditional probability of a feature $x$ given a label $y$ should be close to the marginal probability of $x$
  - **Example**: A rare word like “interpolation” should be similarly rare in both ham and spam (a priori)
  - **Procedure**: Collect relative frequency estimates of both conditional and marginal, then average

\[
P_{ML}(x|y) = \frac{\text{count}(x, y)}{\text{count}(\cdot, y)} \quad P_{ML}(x) = \frac{\text{count}(x)}{\text{count}(\cdot)}
\]

\[
P_{LIN}(x|y) = (1 - \alpha)P_{ML}(x|y) + (\alpha)P_{ML}(x)
\]

- **Effect**: Features have odds ratios closer to 1
Real NB: Smoothing

- Odds ratios without smoothing:

\[
\frac{P(W|\text{ham})}{P(W|\text{spam})} \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
\]

- south-west : inf
- nation : inf
- morally : inf
- nicely : inf
- extent : inf
- ...

- screens : inf
- minute : inf
- guaranteed : inf
- $205.00 : inf
- delivery : inf
- ...

Real NB: Smoothing

- Odds ratios after smoothing:

\[
\frac{P(W|\text{ham})}{P(W|\text{spam})} \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
\]

<table>
<thead>
<tr>
<th>Word</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>helvetica</td>
<td>11.4</td>
</tr>
<tr>
<td>seems</td>
<td>10.8</td>
</tr>
<tr>
<td>group</td>
<td>10.2</td>
</tr>
<tr>
<td>ago</td>
<td>8.4</td>
</tr>
<tr>
<td>areas</td>
<td>8.3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>verdana</td>
<td>28.8</td>
</tr>
<tr>
<td>Credit</td>
<td>28.4</td>
</tr>
<tr>
<td>ORDER</td>
<td>27.2</td>
</tr>
<tr>
<td>&lt;FONT&gt;</td>
<td>26.9</td>
</tr>
<tr>
<td>money</td>
<td>26.5</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Do these make more sense?
Now we’ve got two kinds of unknowns
- Parameters: \( P(F_i|Y) \) and \( P(Y) \)
- Hyperparameters, like the amount of smoothing to do: \( k, \alpha \)

Where to learn which unknowns
- Learn parameters from training set
- Can’t tune hyperparameters on training data (why?)
- For each possible value of the hyperparameters, train and test on the held-out data
- Choose the best value and do a final test on the test data
Baselines

- First task when classifying: get a baseline
  - Baselines are very simple “straw man” procedures
  - Help determine how hard the task is
  - Help know what a “good” accuracy is

- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g. for spam filtering, might label everything as spam
  - Accuracy might be very high if the problem is skewed

- When conducting real research, we usually use previous work as a (strong) baseline
Confidences from a Classifier

- The confidence of a classifier:
  - Posterior of the most likely label
    \[
    \text{confidence}(x) = \max_y P(y|x)
    \]
  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee confidence is correct

- Calibration
  - Strong calibration: confidence predicts accuracy rate
  - Weak calibration: higher confidences mean higher accuracy
  - What’s the value of calibration?
Let’s say we want to classify web pages as homepages or not
- In a test set of 1K pages, there are 3 homepages
- Our classifier says they are all non-homepages
- 99.7 accuracy!
- Need new measures for rare positive events

Precision: fraction of guessed positives which were actually positive

Recall: fraction of actual positives which were guessed as positive

Say we guess 5 homepages, of which 2 were actually homepages
- Precision: 2 correct / 5 guessed = 0.4
- Recall: 2 correct / 3 true = 0.67

Which is more important in customer support email automation?
Which is more important in airport face recognition?
Precision vs. Recall

- Precision/recall tradeoff
  - Often, you can trade off precision and recall
  - Only works well with weakly calibrated classifiers

- To summarize the tradeoff:
  - Break-even point: precision value when $p = r$
  - F-measure: harmonic mean of $p$ and $r$:
    $$F_1 = \frac{2}{1/p + 1/r}$$
Naïve Bayes Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Confidences are useful when the classifier is calibrated
Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just $99.99* – the regular list price is $499! The most common question we've received about this offer is – Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

. . . To receive your $30 Amazon.com promotional certificate, click through to

   http://www.amazon.com/apparel

and see the prominent link for the $30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .
What to Do About Errors

- Problem: there’s still spam in your inbox

- Need more features – words aren’t enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?

- Naïve Bayes models can incorporate a variety of features, but tend to do best in homogeneous cases (e.g. all features are word occurrences)
Features

- **A feature** is a function that signals a property of the input
  - **Naïve Bayes**: features are random variables & each value has conditional probabilities given the label.
  - **Most classifiers**: features are real-valued functions
  - **Common special cases**:  
    - Indicator features take values 0 and 1 (or -1 and 1)
    - Count features return non-negative integers

- **Features are anything you can think of for which you can write code to evaluate on an input**
  - Many are cheap, but some are expensive to compute
  - Can even be the output of another classifier or model
  - Domain knowledge goes here!
Feature Extractors

- Features: anything you can compute about the input
- A feature extractor maps inputs to feature vectors

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidencial and top secret. ...

<table>
<thead>
<tr>
<th>W</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dear</td>
<td>1</td>
</tr>
<tr>
<td>sir</td>
<td>1</td>
</tr>
<tr>
<td>this</td>
<td>2</td>
</tr>
<tr>
<td>wish</td>
<td>0</td>
</tr>
<tr>
<td>MISSPELLED</td>
<td>2</td>
</tr>
<tr>
<td>YOUR_NAME</td>
<td>1</td>
</tr>
<tr>
<td>ALL_CAPS</td>
<td>0</td>
</tr>
<tr>
<td>NUM_URLS</td>
<td>0</td>
</tr>
</tbody>
</table>

- Many classifiers take feature vectors as inputs
- Feature vectors usually very sparse, use sparse encodings (i.e. only represent non-zero keys)
Generative vs. Discriminative

- **Generative classifiers:**
  - E.g. naïve Bayes
  - A causal model with evidence variables
  - Query model for causes given evidence

- **Discriminative classifiers:**
  - No causal model, no Bayes rule, often no probabilities at all!
  - Try to predict the label Y directly from X
  - Robust, accurate with varied features
  - Loosely: mistake driven rather than model driven
Nearest-Neighbor Classification

- Nearest neighbor for digits:
  - Take new image
  - Compare to all training images
  - Assign based on closest example

- Encoding: image is vector of intensities:
  \[ \mathbf{1} = \langle 0.0 \ 0.0 \ 0.3 \ 0.8 \ 0.7 \ 0.1 \ldots 0.0 \rangle \]

- What’s the similarity function?
  - Dot product of two images vectors?
    \[ \text{sim}(x, y) = x \cdot y = \sum_i x_i y_i \]
    - Usually normalize vectors so \( ||x|| = 1 \)
    - \( \min = 0 \) (when?), \( \max = 1 \) (when?)