Today

- Naïve Bayes models
  - Smoothing
  - Real world issues

- Perceptrons
  - Mistake-driven learning
  - Data separation, margins, and convergence
General Naïve Bayes

- This is an example of a *naive Bayes* model:

\[
P(C, \text{Effect}_1 \ldots \text{Effect}_n) = P(C) \prod_i P(\text{Effect}_i|C)
\]

- Total number of parameters is *linear* in \( n \)!

Example: Spam Filtering

- Model: \( P(C, W_1 \ldots W_n) = P(C) \prod_i P(W_i|C) \)

- Parameters:

| \( P(C) \)          | \( P(W|\text{spam}) \)   | \( P(W|\text{ham}) \)   |
|---------------------|-------------------------|-------------------------|
| ham: 0.66           | the: 0.016              | the: 0.021              |
| spam: 0.33          | to: 0.015               | to: 0.013               |
|                     | and: 0.012              | and: 0.011              |
|                     | ...                     | ...                     |
|                     | free: 0.001             | free: 0.005             |
|                     | click: 0.001            | click: 0.004            |
|                     | ...                     | ...                     |
|                     | morally: 0.001          | screens: 0.000          |
|                     | nicely: 0.001           | minute: 0.000           |
|                     | ...                     | ...                     |
Estimation: Laplace Smoothing

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

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

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

- Can derive this as a maximum a posteriori estimate using Dirichlet priors (see cs281a)

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:

\[
P_{LAP,k}(x|y) = \frac{c(x, y) + k}{c(y) + k|X|}
\]
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
  - Get unconditional $P(X)$ 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?

- For even better ways to estimate parameters, as well as details of the math see cs281a, cs294-5

Real NB: Smoothing

- For real classification problems, smoothing is critical
  - ... and usually done badly, even in big commercial systems

- New odds ratios:

|                | $P(W|\text{ham})$ | $P(W|\text{spam})$ |
|----------------|-------------------|---------------------|
| $P(W|\text{ham})$ | $P(W|\text{spam})$ |
| helvetica      | 11.4              | 28.8                |
| seems          | 10.8              | 28.4                |
| group          | 10.2              | 27.2                |
| ago            | 8.4               | 26.9                |
| areas          | 8.3               | 26.5                |
| ...            |                   |                     |

$Do\ these\ make\ more\ sense?$
Tuning on Held-Out Data

- Now we’ve got two kinds of unknowns
  - Parameters: the probabilities $P(Y|X), P(Y)$
  - Hyper-parameters, like the amount of smoothing to do: $k, \alpha$

- Where to learn?
  - Learn parameters from training data
  - Must tune hyper-parameters on different data
    - Why?
  - For each value of the hyper-parameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data

### Spam Example

| Word  | $P(w|\text{spam})$ | $P(w|\text{ham})$ | Tot Spam | Tot Ham |
|-------|--------------------|-------------------|----------|---------|
| (prior) | 0.33333 | 0.66666 | -1.1 | -0.4 |

$P(\text{spam} | w) = 0.989$
Confidences from a Classifier

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

- Calibration
  - Weak calibration: higher confidences mean higher accuracy
  - Strong calibration: confidence predicts accuracy rate
  - What’s the value of calibration?

Precision vs. Recall

- 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 \( r \):
    \[
    F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}
    \]

Errors, and What to Do

- **Examples of errors**

  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?

- 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 which signals a property of the input

- Examples:
  - ALL_CAPS: value is 1 iff email in all caps
  - HAS_URL: value is 1 iff email has a URL
  - NUM_URLS: number of URLs in email
  - VERY_LONG: 1 iff email is longer than 1K
  - SUSPICIOUS_SENDER: 1 iff reply-to domain doesn’t match originating server

- Features are anything you can think of code to evaluate on an input
  - Some cheap, some very very expensive to calculate
  - Can even be the output of another classifier
  - Domain knowledge goes here!

- In naïve Bayes, how did we encode features?
Feature Extractors

- 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, ...
  - W=dear : 1
    W=sir : 1
    W=this : 2
    ...
    W=wish : 0
    ...
    MISSPELLED : 2
    NAMELESS : 1
    ALL_CAPS : 0
    NUM_URLS : 0
    ...

- 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
  - We build a causal model of the variables
  - We then query that model for causes, given evidence

- Discriminative classifiers:
  - E.g. perceptron (next)
  - No causal model, no Bayes rule, often no probabilities
  - Try to predict output directly
  - Loosely: mistake driven rather than model driven
Some (Vague) Biology

- Very loose inspiration: human neurons

The Binary Perceptron

- Inputs are features
- Each feature has a weight
- Sum is the activation

\[ \text{activation}_w(x) = \sum_i w_i \cdot f_i(x) \]

- If the activation is:
  - Positive, output 1
  - Negative, output 0
Example: Spam

- Imagine 4 features:
  - Free (number of occurrences of “free”)
  - Money (occurrences of “money”)
  - BIAS (always has value 1)

<table>
<thead>
<tr>
<th>x</th>
<th>( f(x) )</th>
<th>( w )</th>
<th>( \sum_i w_i \cdot f_i(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>free</strong> : 1</td>
<td><strong>free</strong> : 4</td>
<td>( 1(-3) + )</td>
<td></td>
</tr>
<tr>
<td><strong>money</strong> : 1</td>
<td><strong>money</strong> : 2</td>
<td>( 1(4) + )</td>
<td></td>
</tr>
<tr>
<td><strong>the</strong> : 0</td>
<td><strong>the</strong> : 0</td>
<td>( 1(2) + )</td>
<td></td>
</tr>
<tr>
<td><strong>BIAS</strong> : -3</td>
<td><strong>BIAS</strong> : 1</td>
<td>( 0(0) + )</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( 0(0) + )</td>
<td></td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( 0(0) + )</td>
<td></td>
</tr>
</tbody>
</table>

\( = 3 \)

Binary Decision Rule

- In the space of feature vectors
  - Any weight vector is a hyperplane
  - One side will be class 1
  - Other will be class 0
The Multiclass Perceptron

- If we have more than two classes:
  - Have a weight vector for each class
  - Calculate an activation for each class

\[
\text{activation}_w(x, c) = \sum_i w_{c,i} \cdot f_i(x)
\]

- Highest activation wins

\[
c = \arg \max_c (\text{activation}_w(x, c))
\]

Example

“win the vote”

<table>
<thead>
<tr>
<th>wSPORTS</th>
<th>wPOLITICS</th>
<th>wTECH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS : -2</td>
<td>BIAS : 1</td>
<td>BIAS : 2</td>
</tr>
<tr>
<td>win : 4</td>
<td>win : 2</td>
<td>win : 0</td>
</tr>
<tr>
<td>game : 4</td>
<td>game : 0</td>
<td>game : 2</td>
</tr>
<tr>
<td>vote : 0</td>
<td>vote : 4</td>
<td>vote : 0</td>
</tr>
<tr>
<td>the : 0</td>
<td>the : 0</td>
<td>the : 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
The Perceptron Update Rule

- Start with zero weights
- Pick up training instances one by one
- Try to classify
  \[ c = \arg \max_c \ w_c \cdot f(x) \]
  \[ = \arg \max_c \ \sum_i w_{c,i} \cdot f_i(x) \]
- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer
  \[ w_c = w_c - f(x) \]
  \[ w_{c^*} = w_{c^*} + f(x) \]

Example

“win the vote”
“win the election”
“win the game”

<table>
<thead>
<tr>
<th>wSPORTS</th>
<th>wPOLITICS</th>
<th>wTECH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS :</td>
<td>BIAS :</td>
<td>BIAS :</td>
</tr>
<tr>
<td>win :</td>
<td>win :</td>
<td>win :</td>
</tr>
<tr>
<td>game :</td>
<td>game :</td>
<td>game :</td>
</tr>
<tr>
<td>vote :</td>
<td>vote :</td>
<td>vote :</td>
</tr>
<tr>
<td>the :</td>
<td>the :</td>
<td>the :</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Mistake-Driven Classification

- In naïve Bayes, parameters:
  - From data statistics
  - Have a causal interpretation
  - One pass through the data

- For the perceptron parameters:
  - From reactions to mistakes
  - Have a discriminative interpretation
  - Go through the data until held-out accuracy maxes out

Properties of Perceptrons

- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the margin or degree of separability
  \[ \text{mistakes} < \frac{1}{\delta^2} \]
Issues with Perceptrons

- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining isn’t quite as bad as overfitting, but is similar

- Regularization: if the data isn’t separable, weights might thrash around
  - Averaging weight vectors over time can help (averaged perceptron)

- Mediocre generalization: finds a “barely” separating solution

Summary

- Naïve Bayes
  - Build classifiers using model of training data
  - Smoothing estimates is important in real systems
  - Classifier confidences are useful, when you can get them

- Perceptrons:
  - Make less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data