Example: Spam Filter

Input: email
Output: spam/ham
Setup:
- Get a large collection of example emails, each labeled "spam" or "ham"
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future emails

Features: The attributes used to make the ham/spam decision
- Words: FREE!
- Text Patterns: $dd, CAPS
- Non-text: SenderInContacts
- ...

Dear Sir,
First, I must solicit your confidence in this transaction, this is by virtue of it's nature as being utterly confidential and top secret.

TO BE REMOVED FROM FUTURE MAILINGS: SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.
99 MILLION EMAIL ADDRESSES FOR ONLY $99

Oh, leave this is blatant OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use. I know it was working pre being stuck in the corner but when I plugged it in, hit the power nothing happened.

Other Classification Tasks

- In classification, we predict labels $y$ (classes) for inputs $x$
- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - ... many more
- Classification is an important commercial technology!

Example: Digit Recognition

Input: images/pixel grids
Output: a digit 0-9
Setup:
- Get a large collection of example images, each labeled with a digit
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future digit images

Features: The attributes used to make the digit decision
- Pixels: (6,8)=ON
- Shape Patterns: NumComponents, AspectRatio, NumLoops
- ...

Naïve Bayes for Digits

Simple version:
- One feature $F_{ij}$ for each grid position $<i,j>$
- Possible feature values are on/off, based on whether intensity is more or less than 0.5 in underlying image
- Each input maps to a feature vector, e.g.
  $1 \leftarrow (F_{0,0} = 0 \ F_{1,1} = 0 \ F_{0,2} = 1 \ F_{0,3} = 1 \ F_{3,8} = 0 \ ... \ F_{15,15} = 0)$
- Here: lots of features, each is binary valued

- Naïve Bayes model:
  $P(Y|F_{0,0}, F_{15,15}) P(Y) \prod_{i,j} P(F_{i,j}|Y)$
- What do we need to learn?

General Naïve Bayes

A general naive Bayes model:

$$P(Y,F_1\ldots F_n) = P(Y) \prod_{i=1}^n P(F_i|Y)$$

$|Y|$ parameters $n \times |F|$ parameters

- We only specify how each feature depends on the class
- Total number of parameters is linear in $n$
Inference for Naïve Bayes

- **Goal:** compute posterior over causes
  - **Step 1:** get joint probability of causes and evidence
    \[
    P(Y, f_1 \ldots f_n) = \frac{P(f_1 \ldots f_n | Y)P(Y)}{P(f_1 \ldots f_n)}
    \]
  - **Step 2:** get probability of evidence
  - **Step 3:** renormalize

General Naïve Bayes

- **What do we need in order to use naïve Bayes?**
  - Inference (you know this part)
  - Start with a bunch of conditionals, P(Y) and the P(F_i|Y) tables
  - Use standard inference to compute P(Y|F_1 \ldots F_n)
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - P(F_i|Y) for each feature (evidence variable)
    - These probabilities are collectively called the parameters of the model and denoted by \( \theta \)
    - Up until now, we assumed these appeared by magic, but...
    - ...they typically come from training data: we’ll look at this now

Examples: CPTs

- **P(Y)**
- **P(F_{3,1} = 0|Y)**, **P(F_{3,5} = 0|Y)**

Important Concepts

- **Data:** labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held-out set
  - Test set
- **Features:** attribute-value pairs which characterize each x
- **Experimentation cycle**
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
- **Evaluation**
  - Accuracy: fraction of instances predicted correctly
  - Overfitting and generalization
    - Want a classifier which does well on test data
    - Overfitting: fitting the training data very closely, but not generalizing well
    - We’ll investigate overfitting and generalization formally in a few lectures

A Spam Filter

- Naïve Bayes spam filter
- **Data:** Collection of emails, labeled spam or ham
  - Note: someone has to hand label all this data!
  - Split into training, held-out, test sets
- **Classifiers**
  - Learn on the training set
  - (Tune it on a held-out set)
  - Test it on new emails

Naïve Bayes for Text

- **Bag-of-Words Naïve Bayes:**
  - Predict unknown class label (spam vs. ham)
  - Assume evidence features (e.g. the words) are independent
  - Warning: subtly different assumptions than before!
- **Generative model**
  \[
  P(C, W_1 \ldots W_n) = P(C) \prod P(W_i|C)
  \]
- **Tied distributions and bag-of-words**
  - Usually, each variable gets its own conditional probability distribution P(F|Y)
  - In a bag-of-words model
    - Each position is identically distributed
    - All positions share the same conditional probs P(W|C)
  - Why make this assumption?
Example: Spam Filtering

- Model: \[ P(C, W_1, \ldots W_n) = P(C) \prod P(W_i | C) \]

- What are the parameters?

| \( P(C) \) | \( P(W | \text{spam}) \) | \( P(W | \text{ham}) \) |
|--|--|--|
| the : 0.0156 | the : 0.0210 |
| to : 0.0153 | to : 0.0133 |
| and : 0.0115 | of : 0.0115 |
| of : 0.0095 | 2002 : 0.0110 |
| you : 0.0093 | with : 0.0108 |
| a : 0.0086 | from : 0.0107 |
| with : 0.0080 | and : 0.0105 |
| from : 0.0075 | a : 0.0050 |
| ... | ...

- Where do these tables come from?

Spam Example

| Word | P(w|spam) | P(w|ham) | Tot Spam | Tot Ham |
|------|----------|----------|----------|---------|
| ham | 0.66666 | 0.33333 | -1.1 | -0.4 |
| spam | 0.33333 | 0.66666 | -1.1 | -0.4 |

\[ P(\text{spam} | w) = 98.9 \]

Example: Overfitting

<table>
<thead>
<tr>
<th>( P(\text{features}, C = 2) )</th>
<th>( P(\text{features}, C = 3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(C = 2) = 0.1 )</td>
<td>( P(C = 3) = 0.1 )</td>
</tr>
<tr>
<td>( P(\text{on}</td>
<td>C = 2) = 0.8 )</td>
</tr>
<tr>
<td>( P(\text{off}</td>
<td>C = 2) = 0.1 )</td>
</tr>
<tr>
<td>( P(\text{on}</td>
<td>C = 2) = 0.1 )</td>
</tr>
<tr>
<td>( P(\text{on}</td>
<td>C = 2) = 0.01 )</td>
</tr>
</tbody>
</table>

2 wins!!

Example: Overfitting

- Posterior determined by relative probabilities (odds ratios):

| Word | \( P(W | \text{ham}) \) | \( P(W | \text{spam}) \) |
|------|----------------|----------------|
| south-west | inf | inf |
| nation | inf | inf |
| minute | inf | inf |
| morally | inf | inf |
| guaranteed | inf | $205.00 |
| delivery | inf | inf |
| signature | inf | inf |
| ... | ... | ... |

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
  - Unlike that every occurrence of “minute” is 100% spam
  - 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:
  \[ p_{\text{ML}}(\epsilon) = \frac{\text{count}(\epsilon)}{\text{total samples}} \]
  \[ p_{\text{ML}}(\epsilon) = 1/3 \]

- 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: Laplace Smoothing

- Laplace’s estimate (extended):
  - Pretend you saw every outcome k extra times
    \[
    P_{\text{LAP1}}(x) = \frac{c(x) + k}{N + kK}
    \]
  - What’s Laplace with k = 0?
    - k is the strength of the prior
  - Laplace for conditionals:
    - Smooth each condition independently:
      \[
      P_{\text{LAP1}}(x|y) = \frac{c(x,y) + k}{c(y) + kK}
      \]

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)
    \[
    P_{\text{LIN}}(x|y) = \alpha P(x|y) + (1.0 - \alpha)P(x)
    \]
  - What if \(\alpha\) is 0? 1?

For even better ways to estimate parameters, as well as details of the math see cs281a, cs288

Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:
  \[
  \frac{P(W|\text{ham})}{P(W|\text{spam})}, \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
  \]

<table>
<thead>
<tr>
<th></th>
<th>helvetica</th>
<th>seems</th>
<th>group</th>
<th>ago</th>
<th>areas</th>
<th>...</th>
</tr>
</thead>
</table>
| helvetica| 11.4      | 10.8  | 10.2  | 8.6 | 8.3   |...
| seems    | 10.8      | 10.2  | 8.6   | 8.3 | ...   |
| group    | 10.2      | 8.6   | 8.3   | ... |
| ago      | 8.6       | 8.3   |       |     |
| areas    | 8.3       |       |       |     |

Do these make more sense?

Tuning on Held-Out Data

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

- Where to learn?
  - Learn parameters from training data
  - Tune hyperparameters on different data
  - Why?
    - For each 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

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?

- Can add these information sources as new variables in the NB model

- Next class we’ll talk about classifiers which let you easily add arbitrary features more easily
Baselines

- First step: 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 ham
  - Accuracy might be very high if the problem is skewed
    - E.g. calling everything “ham” gets 66%, so a classifier that gets 70% isn’t very good…

- For real research, usually use previous work as a (strong) baseline

Confidencies from a Classifier

- The confidence of a probabilistic classifier:
  - Posterior over the top label
  - \( \text{confidence}(\hat{y}) = \max_y \ P(y|x) \)
  - 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?

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
- Classifier confidences are useful, when you can get them