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)

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: allCaps, CAPS
  - Non-text: SenderInContacts
  - ...

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9
- Setup:
  - Get a large collection of example images, each labelled 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
  - ...

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!
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
  - Learning 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

Bayes Nets for Classification

- One method of classification:
  - Use a probabilistic model!
  - Features are observed random variables \( F \)
  - Y is the query variable
  - Use probabilistic inference to compute most likely Y

\[
y = \text{argmax}_y P(y|F_1 \ldots F_n)
\]

- You already know how to do this inference

Simple Classification

- Simple example: two binary features

\[
P(m|s, f) \quad \text{direct estimate} \quad P(S|M) \quad P(F|M)
\]

\[
P(m|s, f) = \begin{cases} 
1 & \text{Bayes estimate (no assumptions)} \\
\frac{P(s, f|m)P(m)}{P(s, f)} & \text{Conditional independence} \\
\end{cases}
\]

\[
P(m|s, f) = P(s|m)P(f|m)P(m) + \begin{cases} 
P(\text{not}, m, s, f) = P(s|m)P(f|m)P(m) \\
P(\text{not}, m, s, f) = P(s|m)P(f|m)P(m) \\
\end{cases}
\]

Inference for Naive Bayes

- Goal: compute posterior over causes
  - Step 1: get joint probability of causes and evidence

\[
P(Y, F_1 \ldots F_n) = \begin{bmatrix} P(Y_1, F_1 \ldots F_n) \\
P(Y_2, F_1 \ldots F_n) \\
\vdots \\
P(Y_k, F_1 \ldots F_n) \\
\end{bmatrix}
\]

\[
P(Y_1)P(F_1|Y_1)P(F_2|Y_2)P(F_3|Y_3) \ldots P(F_n|Y_n)
\]

- Step 2: get probability of evidence
- Step 3: renormalize

\[
P(F_1 \ldots F_n)
\]

General Naive Bayes

- A general naïve Bayes model:

\[
P(Y, F_1 \ldots F_n) = P(Y) \prod_{i} P(F_i|Y)\]

\[
\text{parameters } |Y| \times n \times |F| 
\]

- We only specify how each feature depends on the class
- Total number of parameters is linear in n

General Naive 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|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
A Digit Recognizer

- Input: pixel grids

- Output: a digit 0–9

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.
    \[ F_{0,0} = 0 \quad F_{0,1} = 0 \quad F_{0,2} = 1 \quad F_{0,3} = 0 \quad \ldots \quad F_{15,15} = 0 \]
  - Here: lots of features, each is binary valued

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

Examples: CPTs

Parameter Estimation

- Estimating distribution of random variables like $X$ or $X | Y$

- Empirically: use training data
  - For each outcome $x$, look at the empirical rate of that value:
    \[ \hat{P}(x) = \frac{\text{count}(x)}{\text{total samples}} \]
  - This is the estimate that maximizes the likelihood of the data

- Elicitation: ask a human!
  - Usually need domain experts, and sophisticated ways of eliciting probabilities (e.g. betting games)
  - Trouble calibrating

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
  - Generative model
    \[ P(C, W_1, \ldots, W_n) = P(C) \prod_i P(W_i|C) \]

- 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: $P(C, W_1, ..., W_n) = P(C) \prod_{i} P(W_i | C)$
- What are the parameters?

| P(C)   | P(W|spam) | P(W|ham) |
|--------|-----------|----------|
| ham: 0.66 |          |          |
| spam: 0.33 | 0.0156   | 0.0210   |
| the: 0.0153 | to: 0.0133 | of: 0.0119 |
| and: 0.0115 | you: 0.0095 | 2002: 0.0110 |
| of: 0.0095 | with: 0.0086 | and: 0.0105 |
| you: 0.0093 | from: 0.0107 | a: 0.0100 |
| a: 0.0086 | with: 0.0080 | to: 0.01517 |
| ...     | ...      | ...      |

- Where do these tables come from?

Example: Overfitting

- Posteriors determined by relative probabilities (odds ratios):

  - south-west: inf
  - nation: inf
  - morally: inf
  - nicely: inf
  - seriously: inf
  - ... screens: inf
  - minute: inf
  - guaranteed: inf
  - $205.00$: inf
  - delivery: inf
  - signature: 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.
  - 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

Example: Overfitting

- Posteriors determined by relative probabilities (odds ratios):

  | P(W|ham) | P(W|spam) | P(W|ham) |
  |---------|-----------|----------|
  | (prior) | 0.33333   | 0.66666  |
  | Tot Spam| -1.1      | -0.4     |
  | Tot Ham |           |          |

P(spam | w) = 98.9

Example: Overfitting

- Posteriors determined by relative probabilities (odds ratios):

  | P(W|ham) | P(W|spam) |
  |---------|-----------|
  | P(W|ham) |
  | south-west: inf
  | nation: inf
  | morally: inf
  | nicely: inf
  | seriously: inf
  | ... screens: inf
  | minute: inf
  | guaranteed: inf
  | $205.00$: inf
  | delivery: inf
  | signature: inf
  | ...

What went wrong here?

Estimation: Smoothing

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it’s heads, what’s the estimate for P(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
  \[
  \theta_{ML} = \arg \max_{\theta} P(X|\theta) = \frac{\text{count}(x)}{\text{total samples}}
  \]
- In Bayesian statistics, we think of the parameters as just another random variable, with its own distribution
  \[
  \theta_{MAP} = \arg \max_{\theta} P(\theta|X) = \arg \max_{\theta} \frac{P(X|\theta)P(\theta)}{P(X)} = \arg \max_{\theta} P(X|\theta)P(\theta)
  \]

**Estimation: Laplace Smoothing**

- Laplace’s estimate:
  - Pretend you saw every outcome once more than you actually did
  \[
  P_{MAP}(x) = \frac{c(x) + 1}{N + |X|} \quad P_{ML}(X) = \frac{c(x)+1}{N+|X|}
  \]
  - Can derive this as a MAP estimate with Dirichlet prior (see cs281a)

**Estimation: Laplace Smoothing**

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

**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?

- 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

<table>
<thead>
<tr>
<th>Font</th>
<th>Odds Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helvetica</td>
<td>11.4</td>
</tr>
<tr>
<td>Arial</td>
<td>10.8</td>
</tr>
<tr>
<td>Group</td>
<td>10.2</td>
</tr>
<tr>
<td>Age</td>
<td>8.4</td>
</tr>
<tr>
<td>Area</td>
<td>8.3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Tuning on Held-Out Data**

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

- Where to learn?
  - Learn parameters from training data
  - Must tune hyperparameters on different data
  - \(\alpha\)?
  - 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

Do these make more sense?
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

Confidences from a Classifier

• The confidence of a probabilistic classifier:  
  • Posterior over the top 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  
  • 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  
  • Precision: 3 correct / 3 guessed = 0.4  
  • Recall: 3 correct / 3 true = 1.0  

• Which is more important in customer support email automation?  
• Which is more important in airport face recognition?

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
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

Case-Based Reasoning

- Similarity for classification
  - Case-based reasoning
  - Predict an instance’s label using similar instances
- Nearest-neighbor classification
  - K-NN: copy the label of the most similar data point
  - E-NN: let the k nearest neighbors vote (have to devise a weighting scheme)
- Key issues: how to define similarity
  - Trade-off:
    - Small k gives relevant neighbors
    - Large k gives smoother functions
    - Sound familiar?
- [DEMO]

Recap: Nearest-Neighbor

- Nearest neighbor:
  - Classify test example based on closest training example
  - Requires a similarity function (kernel)
  - Lazy learning: extract classifier from data
  - Eager learning: keep data around and predict from it at test time

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:
  \[ \mathbf{I} = (0.0 \ 0.0 \ 0.3 \ 0.6 \ 0.7 \ 0.1 \ldots 0.0) \]
- What’s the similarity function?
  - Dot product of two images vectors:
    \[ \text{sim}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i \]
  - Usually normalize vectors so \(|\mathbf{x}| = 1\)
  - min = 0 (where?), max = 1 (where?)