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: $dd, 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 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
  • ...

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
  • Very important: never "peek" at the 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
General Naïve Bayes

- A general naïve Bayes model:

\[ P(Y, F_1 \ldots F_n) = P(Y) \prod_i P(F_i|Y) \]

| Y parameters | n x (F x Y) parameters |

- We only specify how each feature depends on the class.
- Total number of parameters is \( \text{linear} \) in \( n \).

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

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

- 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 prob \( P(W|C) \).
    - Why make this assumption?

Example: Spam Filtering

- Model:

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

- What are the parameters?

\[
\begin{align*}
P(C) & \\
\text{ham} : & 0.46 \quad \text{spam} : 0.33
\end{align*}
\]

\[
\begin{align*}
P(W|\text{spam}) & \\
\text{the} : & 0.0166
\end{align*}
\]

\[
\begin{align*}
P(W|\text{ham}) & \\
\text{the} : & 0.0230
\end{align*}
\]

- Where do these tables come from?

Parameter Estimation

- Estimating distribution of random variables like \( X \) or \( X | Y \)
- Empirically: use training data
  - For each outcome \( x_i \), look at the empirical rate of that value:

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

- This is the estimate that maximizes the likelihood of the data

\[ L(x, \theta) = \prod_i P(x_i|\theta) \]

Example: Overfitting

- Posterior determined by relative probabilities (odds ratios):

\[
\begin{align*}
P(W|\text{ham}) & \\
P(W|\text{spam})
\end{align*}
\]

| south-west | inf | nation | inf | minute | inf | guaranteed | inf | $205.00 | inf | delivery | inf |
| south-west | inf | nation | inf | minute | inf | guaranteed | inf | $205.00 | inf | delivery | 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
  - 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

- Relative frequencies are the maximum likelihood estimates
  \[ \theta_{ML} = \arg \max_\theta P(X|\theta) \]
  \[ P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}} \]
- In Bayesian statistics, we think of the parameters as just another random
  variable, with its own distribution
  \[ \theta_{MAP} = \arg \max_\theta P(\theta|X) \]
  \[ \theta_{MAP} = \arg \max_\theta P(X|\theta) P(\theta)/P(X) \]
  \[ = \arg \max_\theta P(X|\theta) P(\theta) \]

Estimation: Laplace Smoothing

- Laplace’s estimate:
  \[ P_{LAP}(x) = \frac{c(x) + 1}{N + |X|} \]
  \[ P_{LAP}(X) = \]

Estimation: Linear Interpolation

- Laplace’s estimate (extended):
  \[ P_{LAP}(x) = \frac{c(x) + k}{N + k|X|} \]
  \[ P_{LAP}(X) = \]
  - What’s Laplace with k = 0?
  - k is the strength of the prior
  - Laplace for conditionals:
    - Smooth each condition independently:
      \[ P_{LAP}(x|y) = \frac{c(x, y) + k}{c(y) + k|X|} \]
  - Other 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?!
**Real NB: Smoothing**

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

  - **helvetica**: 11.4
  - **seem**: 10.8
  - **group**: 10.2
  - **ago**: 8.4
  - **areas**: 8.3
  - **...**
  
  - **verdana**: 28.8
  - **Credit**: 28.4
  - **ORDER**: 27.2
  - **<PURT>**: 26.9
  - **money**: 26.5
  - **...**

  *Do these make more sense?*

**Tuning on Held-Out Data**

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

**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
    \[
    \text{confidence}(x) = \max P(y|x)
    \]
  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee confidence is correct

**Measuring Performance**

- Two main measures
  1. ROC curves
    - Receiver Operating Characteristic
  2. Recall Precision

**ROC Curves**

- In two class problem, we have
  - True Positives
  - False Positives
  - True Negatives
  - False Negatives
  - ROC curve shows trade-off btw. two types of error
**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 \( r \):
    \[
    F_1 = \frac{2}{1/p + 1/r}
    \]

**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 . . .

- 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.

**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.
- Classifier confidences are useful, when you can get them.

**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.

- 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.
Discriminative vs. Generative

- Generative model (The artist)
  - \[ p(Data, No Zebra) \]
  - \[ p(Data | No Zebra) \]

- Discriminative model (The lousy painter)
  - \[ p(No Zebra | Data) \]
  - \[ p(Data | No Zebra) \]

  - Classification function \[ f(Data) \]

Some (Simplified) Biology

- Very loose inspiration: human neurons

Linear Classifiers

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

activation \( w(x) = \sum w_i f_i(x) = w \cdot f(x) \)

- If the activation is:
  - Positive, output +1
  - Negative, output -1

Example: Spam

- Imagine 4 features (spam is "positive" class):
  - free (number of occurrences of "free")
  - money (occurrences of "money")
  - BIAS (intercept, always has value 1)

  \[
  \begin{align*}
  \text{BIAS} & : -3 \\
  \text{free} & : 4 \\
  \text{money} & : 2 \\
  \ldots \\
  \text{BIAS} & : 1 \\
  \text{free} & : 1 \\
  \text{money} & : 1 \\
  \ldots \\
  \end{align*}
  \]

Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to \( Y=+1 \)
  - Other corresponds to \( Y=-1 \)

\[
W = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}
\]

Binary Perceptron Update

- Start with zero weights
- For each training instance:
  - Classify with current weights

  \[
  y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}
  \]

  - If correct (i.e., \( y = y^* \)), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if \( y^* \) is -1.

  \[
  w = w + y^* \cdot f
  \]
Multiclass Decision Rule

- If we have more than two classes:
  - Have a weight vector for each class: $w_y$
  - Calculate an activation for each class
    \[
    \text{activation}_y(x, y) = w_y \cdot f(x)
    \]
  - Highest activation wins
    \[
    y = \arg \max_y (\text{activation}_y(x, y))
    \]

Example

"win the vote"

$w_{SPORTS}$ $w_{POLITICS}$ $w_{TECH}$

<table>
<thead>
<tr>
<th>BIAS</th>
<th>win</th>
<th>game</th>
<th>vote</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

The Perceptron Update Rule

- Start with zero weights
- Pick up training instances one by one
- Classify with current weights
  \[
  y = \arg \max_y w_y \cdot f(x) = \arg \max_y \sum_i w_{yi} \cdot f_i(x)
  \]
  - If correct, no change!
  - If wrong: lower score of wrong answer, raise score of right answer
    \[
    w_y = w_y - f(x)
    \\
    w_y' = w_y' + f(x)
    \]

Example

"win the vote"
"win the election"
"win the game"

$w_{SPORTS}$ $w_{POLITICS}$ $w_{TECH}$

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

Examples: Perceptron

- Separable Case