Section #8:
How to do project #4

Matthew Tong
May 30, 2007
Some suggestions from Kristen Branson’s section, 2006
Project #4

• Build a Naïve Bayes spam filter classifier
Project #4

• Build a Naïve Bayes spam filter classifier
  – What is a spam filter?
Project #4

• Build a Naïve Bayes spam filter classifier
  – What is a spam filter?
  – What is spam?
Project #4

• Build a Naïve Bayes spam filter classifier
  – What is a spam filter?
  – What is spam?
  – What is a classifier?
Classifier

• A mapping from elements to labels
  – Often binary labels (spam vs. ham)
  – Related to regression, where “labels” are real-valued
  – Usually elements represented as a feature vector
    • Can have observed and unobserved features
 Classifier

- Say you had two features: how many times the word “diploma” and “guarantee” appear in a document and you’re trying to determining whether or not the document is spam.

(a) Plot of training examples.    (b) Classifier.
Classifier

- Supervised learning
  - Given labels (as in previous example)

- Unsupervised learning
  - No labels
Classifier

• Semi-supervised learning
  – Some elements have labels

• Reinforcement learning
  – Learn proper actions (planning)
  – Difficult since reward/punishment could be ages away!
Project #4

• Build a Naïve Bayes spam filter classifier
  – What is a spam filter?
  – What is spam?
  – What is a classifier?
  – What is Naïve Bayes?
Bayesian Learning

• Bayesian learning finds a classifier that labels according to:

\[
\text{label} = \arg \max_y P(Y = y \mid X = x)
\]
Bayesian Learning

- Using Bayes rule:

\[ p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} = \alpha p(x \mid y)p(y) \]
Bayesian Learning

\[ p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} = \alpha p(x \mid y)p(y) \]

- \( p(y \mid x) \) is the posterior
  - If we knew this, we’d be golden
- \( p(y) \) is the class/hypothesis prior
  - Easy to get - estimate proportion is in each class?
Bayesian Learning

\[ p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} \]

\[ = \alpha p(y)p(x \mid y) \]

- \( p(y \mid x) \) is the posterior
- \( p(y) \) is the class/hypothesis prior
- \( p(x \mid y) \) is the likelihood
  - Estimate by seeing what features members of each class have
  - Easier to estimate than the posterior
  - Can still be VERY hard when \( x \) is high dimensional
Naïve Bayes

- Likelihood hard to calculate due to high dimensionality
  - How can we make this simpler?
Naïve Bayes

- Assume independence of features!

\[ p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \]
Naïve Bayes

• Why is this good?
Naïve Bayes

• Why is this good?
  – Simplifies problem immensely
    • With n binary attributes, $2n+1$ parameters
    • More for real values, but still simple given some assumptions
    • No search required to set parameters!
  – Sometimes true or close to true
  – If given noisy or sparse data, fails gracefully
Naïve Bayes

• Why is this bad?
Naïve Bayes

• Why is this bad?
  – Not generally perfectly true
    • Often some dependencies between data
  – Often DRASTICALLY wrong
So...

• For project, trying to estimate $p(y|x)$ using

$$p(y \mid x) = \alpha p(y) p(x \mid y)$$

$$= \alpha p(y) \prod_{i=1}^{n} p(x_i \mid y)$$
So...

\[ P(Y = y) = \frac{\text{\# of examples with label } y}{\text{total \# of examples}}. \]

\[ P(X_j = x_{ij} | Y = y) = \frac{\text{\# examples with feature } j = x_{ij} \text{ and label } y}{\text{total \# of examples with label } y}. \]
Learning algorithm

```plaintext
Learned.Counts = NB.Learner(\{(x_i, y_i)\}_{i=1}^N)
1) For each label y, feature j, and feature value x_j:
   Initialize Feature.Count[y, j, x_j] ← 0.
2) For each label y:
   Initialize Label.Count[y] ← 0.
3) For i = 1, ... N:
   3-1) Label.Count[y_i]++.
   3-2) For j = 1, ..., d: Feature.Count[y_i, j, x_{ij}]++.
```
Classification algorithm

\[ \hat{y} = f(x, \text{Learned\_Counts}) \]

1) For each label \( y \), compute:
   \[ l_y \leftarrow -(d - 1) \log \text{LabelCount}[y] + \sum_{j=1}^{d} \log \text{FeatureCount}[y, j, x_j]. \]

2) Return \( \text{argmax}_y \ l_y \).

- Why no total # of examples? Don’t care
- Also note that one LabelCount has cancelled out (d-1 instead of d)
Classification algorithm

\[ \hat{y} = f(x, \text{Learned Counts}) \]

1) For each label \( y \), compute:
   \[ l_y \leftarrow -(d - 1) \log \text{LabelCount}[y] + \sum_{j=1}^{d} \log \text{FeatureCount}[y, j, x_j]. \]

2) Return \( \text{argmax}_y \ l_y \).

- Where logs are introduced to overcome rounding errors in:

\[ p_y = \text{LabelCount}[y]^{-(d-1)} \prod_{j=1}^{d} \text{FeatureCount}[y, j, x_j]. \]
Smoothing

• What happens if a bin is 0?
Smoothing

• What happens if a bin is 0?
  – Leads to total prob problems
  – Undesirable, particularly with small data sets
  – Couple solutions, easiest being bigger bins (makes rareer) or smoothing (this is simplest flavor)

\[ P(X_j = x_{ij}|Y = y) = \frac{(# \text{ examples with feature } j = x_{ij} \text{ and label } y) + c}{(\text{total # of examples with label } y) + c(# \text{ of possible values for feature } j)}. \]
Feature selection

• Probably the hardest part of this assignment
• In 151 may cover some automated methods for doing this, happy to discuss in office hours with you
Comparison with other methods

• Rule based
• More advanced probabilistic models