Natural Language Processing

Lecture 7: Classification
Some Administrivia

• Read the book
  – (Really Read the Book!)

• Project Teams
  – Let us know if someone drops

• Read the book
  – (Really Read the Book!)
Acknowledging Collaboration

• **MUST** include explicit acknowledgements of people/software/websites in assignments/projects

• Included in written submissions
  – “No collaboration made” (if none)
Notation

• Training examples: \( x = (x_1, x_2, ..., x_N) \)
• Their categories: \( y = (y_1, y_2, ..., y_N) \)
• A classifier \( C \) seeks to map \( x_i \) to \( y_i \)
  \[ x \rightarrow [C] \rightarrow y \]
• A learner \( L \) infers \( C \) from \((x, y)\)
  \[ x \rightarrow [L] \rightarrow C \]
  \[ y \rightarrow [L] \rightarrow C \]
Three different classifiers

• Noisy Channel
• Naïve Bayes
• Linear Models
• Perceptron
Probabilistic Classifiers

return \arg\max_y p(y' | x)
Noisy Channel Model (General)

What proportion of emails are expected to be spam vs. not spam?

What proportion of product reviews are expected to get 1,2,3,4,5 stars?
Noisy Channel Classifiers

\[
\text{return } \arg\max_y p(y) \times p(x \mid y)
\]
Representing Text: Features

• Any object $x \in \mathcal{X}$ you might be given to classify can be represented as a vector in a vector space
  – Vectors of representing text are often sparse and high-dimensional

• Designing $\Phi$ (“Feature engineering”)
  – What information do you need to solve the problem?
  – What information do you need to avoid mistakes?
  – Very common: bag-of-words
Example: Spam Detection

• A training set of email messages
  – Spam or Not-Spam

• A set of features for each message
  – ???
Example: Spam Detection

• A training set of email messages
  – Spam or Not-Spam

• A set of features for each message
  – Bag of words
    • For each word: Number of occurrences
    • Nigerian Prince, email quota full, won ONE HUNDRED MILLION DOLLARS
  – From someone you know
  – A reply
  – From @andrew.cmu.edu (your domain)
Example: Movie Ratings

• A training set of movie reviews
  – Stars 1-5

• A set of features for each message
  – Bag of words
    • For each word: Number of occurrences
    • Excellent, sucks, blockbuster, biggest, Star Wars, Disney, Adam Sandler
  – ...

Feature Representation

• Discrete vs Continuous
  – Finite number of categories
  – Floating point scale

• Many classifiers require continuous features

• Convert discrete to continuous
  – “very bad”, “bad”, “okay”, “good”, “very good” →
  – 1-5 linear scale, if its a scale
  – Or [1,0,0,0,0], [0,1,0,0,0], [0,0,1,0,0] ...
  – “one hot” vector if its not a scale
Naïve Bayes Classifier

\[ X \rightarrow \mathcal{C} \rightarrow y \]

\[ \phi_j \leftarrow [\Phi(x)]_j \]

\[ \text{return } \arg\max_{y'} p(y') \times \prod_j p(\phi_j \mid y') \]
Naïve Bayes Learner

∀y, \( p(y) \leftarrow \frac{\text{count}(y)}{N} \)

∀y, ∀j, ∀f, \( p(\phi_j(x) = f \mid y) \leftarrow \frac{\text{count}(f, y)}{\text{count}(y)} \)
Linear Classifiers

C:
1. Use $\Phi(x)$ to map $x$ onto a real-valued feature space.
2. Calculate the linear score $z = w^T \Phi(x)$.
3. If $z > 0$, then return $y = \text{YES}$, else $y = \text{NO}$.

$x \rightarrow \boxed{C} \rightarrow y$
Linear Classifiers

\[ x \rightarrow y \]

C
Linear Classifiers

\[ u : w^T u = 0 \]
Linear Classifiers

\[ u : w^\top u = 0 \]
Linear Classifiers

\[ \text{C} \cdot \text{y} = \text{w} \cdot \text{u} = 0 \]
Linear Classifiers (> 2 Classes)

\[
\text{return } \arg\max_y w^T \Phi(x, y)
\]
Perceptron Learner

\[ w \leftarrow 0 \]

for \( t = 1 \) \ldots \( T \):

select \((x_t, y_t)\)

\# run current classifier

\[ y \leftarrow \text{argmax}_{y'} \quad w^T \Phi(x, y') \]

if \( y \neq y_t \) then \# mistake

\[ w \leftarrow w + \alpha \left[ \Phi(x_t, y_t) - \Phi(x_t, y) \right] \]

return \( w \)
Perceptrons

• Single Layer Perceptron
  – Multilayer perceptrons used in neural networks

• Weights are found iteratively
  – May converge (or not) depending on problem
  – May find local maximum
  – May take longer to train
Which Classifier to Use?

• The Best one for the Problem
• Machine Learning Specialist
  – Best algorithm to train from given features
• Language Technologist
  – Feature engineer the best features
• NLP Specialist
  – Care about *both*
  – What features should be there, and what algorithm will be able to exploit those features