NLP: Classification

Dan Garrette
dhg@cs.utexas.edu

September 12, 2013

1 Classification Tasks

- Language identification: determine the language that a text is written in
- Spam filtering: label emails, tweets, blog comments as spam or not spam
- Routing: label emails to approprate people in an organization (complaints, tech support, order status, etc)
- Sentiment analysis: label some text as being positive or negative (polarity classification)

Example: Sentiment

- Determine the sentiment (positive vs. negative) of, for example, a tweet
- "Probably the worst movie of the century"
- One idea: Compare of "positive" words (good, great, best) to "negative" words (bad, terrible, worst).
  - Humans give insufficent lists. Learning a sufficent list is hard.
  - Words mean different things in different contexts:
    "This thing is *a great deal*. Definitely worth the money."
    "*A great deal* of media attention surrounded the event."
    "It's *a great deal*... if you're looking to looking to waste your money."
  - Some words can flip polarity:
    "It's **not** a **good** investment."
    "I thought it would be **terrible**, **but** I was so wrong."
  - Multi-word expressions:
    "The movie was <u>shit</u>."
    "The movie was <u>the shit</u>."
  - Subtlety:
    "If this movie's your thing, don't bother talking to me."
    "Great plot, great acting, great cast, but it just doesn't hold up."

- Can depend on the target:
  "Unpredicatable plot"
  "Unpredicatable steering"

- Additional Features
  - Ngrams: "must buy", "couldn't care less"
  - Casing: uppercase words are often subjective
  - Punctuation: lots of ! or ? can indicate subjectivity
  - Emoticons: :) vs. :(

2 Rule-Based System

- Write prediction rules: "If contains X and Y but not Z, then 'positive'"
- What happens when multiple rules apply but conflict?
  - Order the rules according to their accuracy?
  - Assign weights to the rules?
- Problems
  - Time-consuming and expensive to write rules.
  - High precision, low recall
  - Rules have to be manually tailored to each dataset (unpredictable vs. unpredictable)
  - Expensive to update (new expressions, new slang, new abrvs)

3 Learning

- If we have examples, we can learn a function mapping texts to categories
- Often probabilistic
- Instead of rules, we use features
- Features are automatically weighted based on statistics in the training data.
- Features are dimensions in space.
- Learn a boundary between classes.
- Boundary used to classify new texts.

Some Data

```
start=B, end=ia, location
start=B, end=er, person
start=M, end=ia, person
start=L, end=ia, location
start=N, end=er, location
start=B, end=ia, location
start=E, end=nd, location
start=N, end=ia, location
start=A, end=er, person
start=L, end=ke, person
```

Our Goal

- Learn a function that maps features to the most likely label
- best_label = $\text{argmax}_{\text{label}}\ p(label \mid \text{features})$

Direct Posterior Parameter Estimation from Data

- $p(label \mid features) = \frac{C(instances\ with\ label\ and\ feature)}{C(instances\ with\ features)}$
  - $p(label = location \mid start = B, end = er) = \frac{0}{1} = 0.0$
  - $p(label = person \mid start = B, end = er) = \frac{1}{1} = 1.0$

  - $p(label = location \mid start = B, end = ia) = \frac{2}{2} = 1.0$
  - $p(label = person \mid start = B, end = ia) = \frac{0}{2} = 0.0$
- This doesn't work very well
- Sparsity: any particular feature combination is rare, hard to generalize
- Even worse when every word in a text is a feature
  - Every text would be unique, and there would be no generalization at all
  - Couldn't label new instances

Bayes Rule

- $p(label \mid features) = \frac{p(features|label) \cdot p(label)}{p(features)}$
- best_label = $\text{argmax}_{\text{label}}\ \frac{p(features|label) \cdot p(label)}{p(features)}$
- If labels = {A,B}: $\frac{p(features|A) \cdot p(A)}{p(features)}$ vs. $\frac{p(features|B) \cdot p(B)}{p(features)}$
- Denominator is always the same, so: $p(features \mid A) \cdot p(A)$ vs. $p(features \mid B) \cdot p(B)$
- Thus, to compute the **posterior**, we need two things
  - the likelihood of the evidence: $p(features \mid label)$

- the prior: $p(label)$

Direct Evidence Likelihood Estimation from Data

- $p(features \mid label) = \frac{C(instances\ with\ features\ and\ label)}{C(instances\ with\ label)}$
  - $p(start = B, end = er \mid label = location) = \frac{0}{6} = 0.0$
  - $p(start = B, end = er \mid label = person) = \frac{1}{4} = 0.25$

  - $p(start = B, end = ia \mid label = location) = \frac{2}{6} = 0.33$
  - $p(start = B, end = ia \mid p(label = person) = \frac{0}{4} = 0.0$
- Still problematic: still sparse, still lots of zeros, still hard to generalize

Naïve Bayes

- We want to disentangle the features for better generalization
- Compute each feature's probability independently
- Will be able to compute the probability of an instance from the features even if we haven't seen that particular combination of features before.
- Requires us to assume that features are independent
  - Not actually true! Language doesn't work like that.
  - But it's a simplifying assumption
  - "Naïve" assumption
- $p(features \mid label) = p(F_1, F_2, F_3, ... \mid label) = p(F_1 \mid label) \cdot p(F_2 \mid label) \cdot p(F_3 \mid label) \cdot ...$

Parameter Estimation from Data

- The prior
  - $p(label = location) = \frac{C(label{=}location)}{\sum_l C(label{=}l)} = \frac{6}{10} = 0.6$
  - $p(label = person) = \frac{C(label{=}person)}{\sum_l C(label{=}l)} = \frac{2}{10} = 0.2$
- Likelihood of the evidence
  - $p(start = B \mid label = location) = \frac{C(start{=}B,label{=}location)}{C(label{=}location)} = \frac{2}{6} = 0.33$
  - $p(start = B \mid label = person) = \frac{C(start{=}B,label{=}person)}{C(label{=}person)} = \frac{1}{4} = 0.25$

  - $p(end = ia \mid label = location) = \frac{C(end{=}ia,label{=}location)}{C(label{=}location)} = \frac{4}{6} = 0.67$
  - $p(end = ia \mid label = person) = \frac{C(end{=}ia,label{=}person)}{C(label{=}person)} = \frac{1}{4} = 0.25$

- $p(end = nd \mid label = location) = \frac{C(end{=}nd, label{=}location)}{C(label{=}location)} = \frac{1}{6} = 0.17$
- $p(end = nd \mid label = person) = \frac{C(end{=}nd, label{=}person)}{C(label{=}person)} = \frac{0}{4} = 0.0$

Naïve Probabilities

- Before
  - $p(start = B, end = ia \mid label = location) = \frac{2}{6} = 0.33$
  - $p(start = B, end = ia \mid p(label = person) = \frac{0}{4} = 0.0$
- Now
  - $p(start = B \mid label = location) \cdot p(end = ia \mid label = location) = 0.33 \cdot 0.67 = 0.22$
  - $p(start = B \mid label = location) \cdot p(end = ia \mid label = person) = 0.25 \cdot 0.25 = 0.06$

Classifying

- We get a **new** instance. Need to determine its label.
- Works even if we haven't seen the particular combination of features
- `start=L, end=er`
- $p(features \mid A) \cdot p(A)$ vs. $p(features \mid B) \cdot p(B)$
- $p(start = L \mid location) \cdot p(end = er \mid location) \cdot p(location) = \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{6}{10} = 0.02$
- $p(start = L \mid person) \cdot p(end = er \mid person) \cdot p(person) = \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{4}{10} = 0.06$
- So, it's more likely a *person*

Why Naïve Bayes?

- Modularity: separate out individual features and prior
- Helps us deal with sparsity
  - Particular feature combinations are rare
  - Individual features are less sparse
  - Still have some features that appear only with one label (meaning zero probaiblities), but this is less common
- Priors
  - Controls how much the base label distribution affects the probabilities
  - Can be automatically calculated from data (as we've seen)
  - Can be set from outside knowledge, if available
    * Imagine we are explicitly told that 3/4 of named entities are people
    * $p(label = person) = 0.75$

        - $p(start = L \mid location) \cdot p(end = er \mid location) \cdot p(location) = \frac{1}{6} \cdot \frac{1}{6} \cdot 0.25 = 0.007$
        - $p(start = L \mid person) \cdot p(end = er \mid person) \cdot p(person) = \frac{1}{4} \cdot \frac{2}{4} \cdot 0.75 = 0.09$
        - So the likelihood of *person* is even higher
    - Useful for injecting linguistic knowledge into a learned model

Can set the prior however we want

P & R

Smoothing - dev set