Feature-based Discriminative Classifiers

Making features from text for discriminative NLP models

Christopher Manning

Classifiers

- A classifier is a function g which assigns an input datum d to one of $|C|$ classes, $c \in C$: $g: D \to C$

- The classes might be:
  - {PERSON, ORGANIZATION, LOCATION, O} for named entity recognition
  - {politics, sports, finance, technology, arts, leisure, …} for news
  - {spam, notspam} for an email message
  - {coreferent, not-coreferent} for a coreference candidate mention pair

Example problem

- Classify a capitalized proper noun as a class:
  - LOCATION, DRUG, PERSON
- For a data example *d*
  - *taking Zantac*
- We work by considering each class *c* for the word:
  - (LOCATION, *taking Zantac*,)
  - (DRUG, *taking Zantac*,)
  - (PERSON, *taking Zantac*,)
- and using features to score each candidate classification

Features for a classifier

- *Features* f are elementary pieces of evidence that link aspects of what we observe *d* with a category *c* that we want to predict
- A feature is a function with a bounded real value: $f\colon C \times D \rightarrow \mathbb{R}$
  - Common special case in NLP:
    - binary features $f\colon C \times D \rightarrow \{0, 1\}$

Example binary features

- $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{“in”} \wedge \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \text{“c”})]$

1.8 −0.6

LOCATION *in Arcadia* | LOCATION *in Québec* | 0.3 DRUG *taking Zantac* | PERSON *saw Sue*

- Models will assign to each feature a *weight:*
  - A positive weight votes that this configuration is likely correct
  - A negative weight votes that this configuration is likely incorrect

Binary Features

- Very commonly, a feature specifies
  1. an indicator function – a yes/no boolean matching function – of properties of the input Φ *and*
  2. a particular class

$$f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$$

[Value is 0 or 1]

  - Each feature picks out a data subset and suggests a label for it
- The decision about a data point is based only on the values of the features active at that point.

Christopher Manning

More General Features

- Features can be more general than just binary matching:
  - Can compute a real value from input, e.g., log(word length)
  - Can match a set of values – e.g., perhaps a partial structure – across "classes"
    - This leads to structured classification, which is common in NLP, for example to match parse tree candidates, etc.
      - A discriminative can have features that match a tree with a unary S to VP
      - A coreference model can not like a cluster with different gender items

Building a Simple Discriminative Model

- We define features (indicator functions) over data points
  - Features represent sets of data points which are distinctive enough to deserve model parameters.
    - Words, but also "word contains number", "word ends with *ing*", POS, syntactic structure, relation between two phrases, etc.
- We might simply encode each Φ feature as a unique String
  - A datum will give rise to a set of Strings: the active Φ features
  - Each feature $f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$ gets a real number weight
- We concentrate on Φ features, but one weight for each i of f_i

Building a Simple Discriminative Model

- Features are normally added in big batches via feature templates
  - E.g., one feature template adds $\forall$ *i,j* observed: lastWord=$w_i \wedge c = c_j$
  - Another is: nextWord=$w_i \wedge c = c_j$. Each may add tens of thousands of features
- A model may be specified by the set of feature templates used
- Features are often added during model development to target errors
  - Often, the easiest thing to think of are features that mark bad combinations

Linear classifiers at classification time

- Linear function from feature sets $\{f_i\}$ to classes $\{c\}$.
- Assign a weight λ_i to each feature f_i.
- We consider each class for an observed datum d
- For a pair (c,d), features vote with their weights:
  - $\text{vote(c)} = \Sigma\lambda_i f_i(c,d)$

PERSON
in Québec

LOCATION
in Québec

DRUG
in Québec

- Choose the class c which maximizes $\Sigma\lambda_i f_i(c,d)$

Christopher Manning

Linear classifiers at classification time

- Linear function from feature sets $\{f_i\}$ to classes $\{c\}$.
- Assign a weight λ_i to each feature f_i.
- We consider each class for an observed datum d
- For a pair (c,d), features vote with their weights:
  - $\text{vote}(c) = \Sigma\lambda_i f_i(c,d)$

PERSON *in Québec*

1.8

LOCATION *in Québec*

–0.6

0.3

DRUG *in Québec*

- Choose the class c which maximizes $\Sigma\lambda_i f_i(c,d)$ = LOCATION

Feature-based Discriminative Classifiers

Making features from text for discriminative NLP models

Feature-based softmax/maxent linear classifiers

How to put features into a classifier

Feature-Based Linear Classifiers

- Linear classifiers are a linear function from feature sets $\{f_i\}$ to classes $\{c\}$
- At test time, we consider each class c for a datum d
  - We generate a feature set $\{f_i\}$ for an observed datum-class pair (c,d)
  - Each feature f_i has a weight λ_i
  - We then score each possible class assignment: $\text{vote(c)} = \Sigma\lambda_i f_i(c,d) = \lambda \cdot f$
  - We choose the class c which maximizes $\Sigma\lambda_i f_i(c,d)$
- At training time we have observed (c,d) pairs from labeled examples
  - We generate sets of features $\{f_i(c,d)\}$ for them
  - We use information about what features occur and don't occur to set a weight λ_i for each feature

Example features

- $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{"in"} \wedge \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \text{"c"})]$

1.8 LOCATION *in Arcadia*

−0.6 LOCATION *in Québec*

0.3 DRUG *taking Zantac*

PERSON *saw Sue*

Christopher Manning

Maxent models (softmax, multiclass logistic, exponential, conditional log-linear, Gibbs)

- Make a probabilistic model from the linear combination $\Sigma\lambda_i f_i(c,d)$

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

Makes votes positive

Normalizes votes

  - P(LOCATION|*in Québec*) = $e^{1.8}e^{-0.6}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.586$
  - P(DRUG|*in Québec*) = $e^{0.3} / (e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.238$
  - P(PERSON|*in Québec*) = $e^{0} / (e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.176$
- The weights are the parameters of the probability model, combined via a “soft max” function

Feature-Based Linear Classifiers

- Maxent models:
  - Given this model form, we choose parameters $\{\lambda_i\}$ that *maximize the conditional likelihood* of the data according to this model (as discussed later): $\max_{\Lambda} P(D|C, \Lambda)$
  - We construct not only classifications, but probability distributions over classifications.

Christopher Manning

Feature-Based Linear Classifiers

There are other (good!) ways to chose weights for features

- Perceptron: find a currently misclassified example, and nudge weights in the direction that corrects classification
- Margin-based methods (Support Vector Machines)
- Boosting algorithms

But these methods are not as trivial to interpret as probability distributions over classes

Feature-based softmax/maxent linear classifiers

How to put features into a classifier