Feature-based Discriminative Classifiers

Making features from text for discriminative NLP models

Christopher Manning

Classifiers

- A classifier is a function *f* which assigns an input datum *d* to one of |*C*| classes, $c \in C$
- The classes might be:
  - {spam, notspam} for an email message
  - {politics, sports, finance, technology, arts, leisure, …} for news
  - {we-are-coreferent, we-are-not-coreference}
    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:
    - binary features $f\colon C \times D \rightarrow \{0, 1\}$

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*

- 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

---

Features

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

$$f_i(c, d) \equiv [\Phi(d) \wedge c = c_j] \qquad \text{[Value is 0 or 1]}$$

- Each feature picks out a data subset and suggests a label for it

Christopher Manning

Feature-Based Models

- The decision about a data point is based only on the features active at that point.

Data BUSINESS: Stocks hit a yearly low …	Data … to restructure bank:MONEY debt.
Label: BUSINESS Features {…, stocks, hit, a, yearly, low, …}	Label: MONEY Features {…, w_{-1}=restructure, w_{+1}=debt, Leng=12, …}
Text Categorization	Word-Sense Disambiguation

Christopher Manning

Feature-Based Linear Classifiers

- 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

Feature-Based Linear Classifiers

- 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*

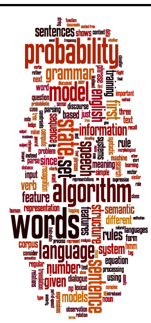
  - Choose the class c which maximizes $\Sigma\lambda_i f_i(c,d)$ = LOCATION

Christopher Manning

Feature-Based Linear Classifiers

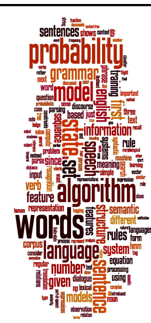
There are many ways to chose weights for features

- Perceptron: find a currently misclassified example, and nudge weights in the direction that corrects classification
- Margin-based methods (Support Vector Machines)
- Maximum entropy models ("softmax regression"; roughly logistic regression), which we will look at next



Feature-based Discriminative Classifiers

Making features from text for discriminative NLP models



Feature-based Linear Classifiers

How to put features into a classifier

Christopher Manning

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: $\text{vote}(c) = \Sigma\lambda_i f_i(c,d)$
  - 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

Christopher Manning

Example features

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

Christopher Manning

Feature-Based Linear Classifiers

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

Christopher Manning

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)
  - We construct not only classifications, but probability distributions over classifications.
    - There are other (good!) ways of discriminating classes – SVMs, boosting, even perceptrons – but these methods are not as trivial to interpret as distributions over classes.

Christopher Manning

Feature Expectations

- We will crucially make use of two *expectations*
  - actual or predicted counts of a feature firing:
  - Empirical count (expectation) of a feature:

$$\text{empirical } E(f_i) = \sum_{(c,d)\in \text{observed}(C,D)} f_i(c,d)$$

  - Model expectation of a feature:

$$E(f_i) = \sum_{(c,d)\in(C,D)} P(c,d) f_i(c,d)$$

Christopher Manning

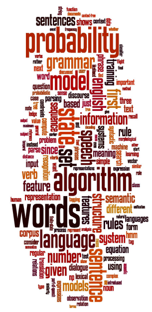
Building a Maxent Model

- 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 the math uses i indices of f_i

Christopher Manning

Building a Maxent 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



Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

Christopher Manning

Christopher Manning

Introduction

- So far we've mainly looked at "generative models"
  - Language models, IBM alignment models, PCFGs
- But there is much use of conditional or discriminative models in NLP, Speech, IR, and ML generally
- Because:
  - They give high accuracy performance
  - They make it easy to incorporate lots of linguistically important features
  - They allow automatic building of language independent, retargetable NLP modules

Christopher Manning

Joint vs. Conditional Models

- We have some data $\{(d, c)\}$ of paired observations d and hidden classes c.
- Joint (generative) models place probabilities over both observed data and the hidden stuff $P(c,d)$
  - They generate the observed data from the hidden stuff
  - All the classic StatNLP models:
    - *n*-gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars, IBM machine translation alignment models

Christopher Manning

Joint vs. Conditional Models

- Discriminative (conditional) models take the data as given, and put a probability/score over hidden structure given the data: $P(c|d)$
  - Logistic regression, conditional loglinear or maximum entropy models, conditional random fields
  - Also, SVMs, (averaged) perceptron, etc. are discriminative classifiers (but not directly probabilistic)

Christopher Manning

Conditional vs. Joint Likelihood

- A *joint* model gives probabilities $P(d,c) = P(c)P(d|c)$ and tries to maximize this joint likelihood.
  - It ends up trivial to choose weights: just count! (relative frequencies)
- A *conditional* model gives probabilities $P(c|d)$. It takes the data as given and models only the conditional probability of the class.
  - We seek to maximize conditional likelihood.
  - Harder to do (as we'll see…)
  - More closely related to classification error.

Christopher Manning

Conditional models work well: Word Sense Disambiguation

Training Set	
Objective	Accuracy
Joint Like.	86.8
Cond. Like.	98.5

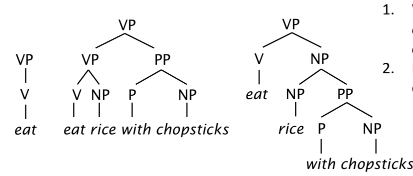
Test Set	
Objective	Accuracy
Joint Like.	73.6
Cond. Like.	76.1

(Klein and Manning 2002, using Senseval-1 Data)

- Even with exactly the same features, changing from joint to conditional estimation increases performance
- That is, we use the same smoothing, and the same word-class features, we just change the numbers (parameters)

Christopher Manning

PCFGs Maximize Joint, not Conditional Likelihood



46 6 2

1. What parse for *eat rice with chopsticks?*
2. How can you get the other parse?

Based on an example by Mark Johnson

Maxent Models and Discriminative Estimation

Maximizing the likelihood

Christopher Manning

Exponential Model Likelihood

- Maximum (Conditional) Likelihood Models :
  - Given a model form, we choose values of parameters λ_i to maximize the (conditional) likelihood of the data.
- For any given feature weights, we can calculate:
  - Data conditional likelihood
  - Derivative of the likelihood wrt each feature weight

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

Christopher Manning

The Likelihood Value

- The (log) conditional likelihood of iid* data (*C*,*D*) according to a maxent model is a function of the data and the parameters λ:

$$\log P(C \mid D, \lambda) = \log \prod_{(c,d)\in(C,D)} P(c \mid d, \lambda) = \sum_{(c,d)\in(C,D)} \log P(c \mid d, \lambda)$$

- If there aren't many values of *c*, it's easy to calculate:

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

*A fancy statistics term meaning "independent and identically distributed". You normally need to assume this for anything formal to be derivable, even though it's never quite true in practice.

Christopher Manning

The Likelihood Value

- We can separate this into two components:

$$\log P(C \mid D, \lambda) = \sum_{(c,d)\in(C,D)} \log \exp \sum_i \lambda_i f_i(c,d) - \sum_{(c,d)\in(C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)$$

$$\log P(C \mid D, \lambda) = N(\lambda) - M(\lambda)$$

- We can maximize it by finding where the derivative is 0
- The derivative is the difference between the derivatives of each component

Christopher Manning

The Derivative I: Numerator

$$\frac{\partial N(\lambda)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \log \exp \sum_i \lambda_{ci} f_i(c,d)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{\partial \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} f_i(c,d)$$

Derivative of the numerator is: the empirical count(f_i, c)

Christopher Manning

The Derivative II: Denominator

$$\frac{\partial M(\lambda)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \frac{\partial \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c',d)}{1} \frac{\partial \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c',d)}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \frac{\partial \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \sum_{c'} P(c' \mid d,\lambda) f_i(c',d)$$

= predicted count(f_i, λ)

Christopher Manning

The Derivative III

$$\frac{\partial \log P(C \mid D,\lambda)}{\partial \lambda_i} = \text{actual count}(f_i, C) - \text{predicted count}(f_i, \lambda)$$

- The optimum parameters are the ones for which each feature's predicted expectation equals its empirical expectation. The optimum distribution is:
  - Always unique (but parameters may not be unique)
  - Always exists (if feature counts are from actual data).
- These models are also called maximum entropy models because we find the model having maximum entropy and satisfying the constraints: $E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$

Christopher Manning

Finding the optimal parameters

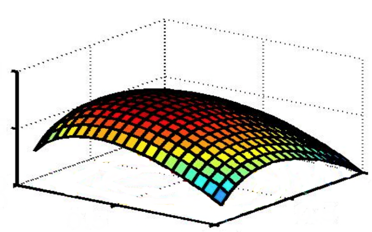
- We want to choose parameters λ_1, λ_2, λ_3, … that maximize the conditional log-likelihood of the training data

$$CLogLik(D) = \sum_{i=1}^{n} \log P(c_i \mid d_i)$$

- To be able to do that, we've worked out how to calculate the function value and its partial derivatives (its gradient)

Christopher Manning

A likelihood surface



Christopher Manning

Finding the optimal parameters

- Use your favorite numerical optimization package….
  - Commonly (and in our code), you **minimize** the negative of *CLogLik*

1. Gradient descent (GD); Stochastic gradient descent (SGD)
2. Iterative proportional fitting methods: Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS)
3. Conjugate gradient (CG), perhaps with preconditioning
4. Quasi-Newton methods – limited memory variable metric (LMVM) methods, in particular, L-BFGS