Maxent Models and Discriminative Estimation

Christopher Manning
CS224N/Ling237
2004

Introduction

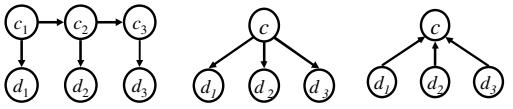
- In recent years there has been extensive use of *conditional* or *discriminative* probabilistic models in NLP, IR, and Speech
- Because:
  - They give high accuracy performance
  - They make it easy to incorporate lots of linguistically important features
  - They allow automatic building of language independent, retargetable NLP modules

Joint vs. Conditional Models

- Joint (generative) models place probabilities over both observed data and the hidden stuff (generate the observed data from hidden stuff): P(c,d)
  - All the best known StatNLP models:
    - *n*-gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars
- Discriminative (conditional) models take the data as given, and put a probability over hidden structure given the data: P(c|d)
  - Logistic regression, conditional loglinear models, maximum entropy markov models, (SVMs, perceptrons)

Bayes Net/Graphical Models

- Bayes net diagrams draw circles for random variables, and lines for direct dependencies
- Some variables are observed; some are hidden
- Each node is a little classifier (conditional probability table) based on incoming arcs



Generative Discriminative

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

- 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)

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

Features

- In these slides and most maxent work: *features* are elementary pieces of evidence that link aspects of what we observe *d* with a category *c* that we want to predict.
- A feature has a (bounded) real value: $f: C \times D \rightarrow \mathbf{R}$
- Usually features are indicator functions of properties of the input and a particular class (*every one we present is*). They pick out a subset.
  - $f_i(c, d) \equiv [\Phi(d) \wedge c = c_i]$ [Value is 0 or 1]
- We will freely say that $\Phi(d)$ is a feature of the data d, when, for each c_i, the conjunction $\Phi(d) \wedge c = c_i$ is a feature of the data-class pair (c, d).

Features

- For example:
  - $f_1(c, d) \equiv [c = \text{“NN”} \wedge \text{islower}(w_0) \wedge \text{ends}(w_0, \text{“d”})]$
  - $f_2(c, d) \equiv [c = \text{“NN”} \wedge w_{-1} = \text{“to”} \wedge t_{-1} = \text{“TO”}]$
  - $f_3(c, d) \equiv [c = \text{“VB”} \wedge \text{islower}(w_0)]$

IN NN *in bed* | TO NN *to aid* | TO VB *to aid* | IN JJ *in blue*

- Models will assign each feature a *weight*
- 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:

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

Feature-Based Models

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

Data	Data	Data
BUSINESS: Stocks hit a yearly low …	… to restructure bank:MONEY debt.	DT JJ NN … The previous fall …
Label: BUSINESS	Label: MONEY	Label: NN
Features: {…, stocks, hit, a, yearly, low, …}	Features: {…, P=restructure, N=debt, L=12, …}	Features: {W=fall, PT=JJ PW=previous}
Text Categorization	Word-Sense Disambiguation	POS Tagging

Example: Text Categorization

(Zhang and Oles 2001)

- Features are a word in document and class (they do feature selection to use reliable indicators)
- Tests on classic Reuters data set (and others)
  - Naïve Bayes: 77.0% F_1
  - Linear regression: 86.0%
  - Logistic regression: 86.4%
  - Support vector machine: 86.5%
- Emphasizes the importance of *regularization* (smoothing) for successful use of discriminative methods (not used in most early NLP/IR work)

Example: NER

(Klein et al. 2003; also, Borthwick 1999, etc.)

- Sequence model across words
- Each word classified by local model
- Features include the word, previous and next words, previous classes, previous, next, and current POS tag, character *n*-gram features and *shape* of word
  - Best model had > 800K features
- High (> 92% on English devtest set) performance comes from combining many informative features.
- With smoothing / regularization, more features never hurt!

Decision Point: State for *Grace*

Local Context

	Prev	Cur	Next
Class	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Example: NER

(Klein et al. 2003)

Decision Point: State for *Grace*

Local Context

	Prev	Cur	Next
Class	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Feature Weights

Feature Type	Feature	PERS	LOC
Previous word	*at*	-0.73	0.94
Current word	*Grace*	0.03	0.00
Beginning bigram	*<G*	0.45	-0.04
Current POS tag	NNP	0.47	0.45
Prev and cur tags	IN NNP	-0.10	0.14
Previous state	Other	-0.70	-0.92
Current signature	Xx	0.80	0.46
Prev state, cur sig	O-Xx	0.68	0.37
Prev-cur-next sig	x-Xx-Xx	-0.69	0.37
P. state - p-cur sig	O-x-Xx	-0.20	0.82
…			
Total:		**-0.58**	**2.68**

Example: POS Tagging

- Features can include:
  - Current, previous, next words in isolation or together.
  - Previous (or next) one, two, three tags.
  - Word-internal features: word types, suffixes, dashes, etc.

Decision Point

Local Context

-3	-2	-1	0	+1
DT	NNP	VBD	???	???
The	Dow	fell	22.6	%

Features

W_0	22.6
W_{+1}	%
W_{-1}	fell
T_{-1}	VBD
T_{-1}–T_{-2}	NNP–VBD
hasDigit?	true
…	…

(Ratnaparkhi 1996; Toutanova et al. 2003, etc.)

Other Maxent Examples

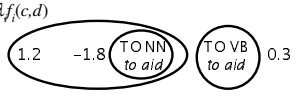
- Sentence boundary detection (Mikheev 2000)
  - Is period end of sentence or abbreviation?
- PP attachment (Ratnaparkhi 1998)
  - Features of head noun, preposition, etc.
- Language models (Rosenfeld 1996)
  - $P(w_0|w_{-n},...,w_{-1})$. Features are word n-gram features, and trigger features which model repetitions of the same word.
- Parsing (Ratnaparkhi 1997; Johnson et al. 1999, etc.)
  - Either: Local classifications decide parser actions or feature counts choose a parse.

Conditional vs. Joint Likelihood

- We have some data $\{(d, c)\}$ and we want to place probability distributions over it.
- A *joint* model gives probabilities $P(d,c)$ and tries to maximize this likelihood.
  - It turns out to be trivial to choose weights: just 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.

Feature-Based Classifiers

- "Linear" classifiers:
  - Classify from features sets $\{f_i\}$ to classes $\{c\}$.
  - Assign a weight λ_i to each feature f_i.
  - For a pair (c,d), features vote with their weights:
    - $\text{vote}(c) = \Sigma\lambda_i f_i(c,d)$



  - Choose the class c which maximizes $\Sigma\lambda_i f_i(c,d) = \text{VB}$
  - There are many ways to chose weights
    - Perceptron: find a currently misclassified example, and nudge weights in the direction of a correct classification

Feature-Based Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
  - Use the linear combination $\Sigma\lambda_i f_i(c,d)$ to produce a probabilistic model:

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

Makes votes positive.

Normalizes votes.

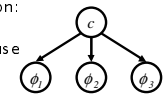
    - $P(\text{NN}|\textit{to}, \textit{aid}, \text{TO}) = e^{1.2}e^{-1.8}/(e^{1.2}e^{-1.8} + e^{0.3}) = 0.29$
    - $P(\text{VB}|\textit{to}, \textit{aid}, \text{TO}) = e^{0.3}/(e^{1.2}e^{-1.8} + e^{0.3}) = 0.71$
  - The weights are the parameters of the probability model, combined via a "soft max" function
  - Given this model form, we will choose parameters $\{\lambda_i\}$ that *maximize the conditional likelihood* of the data according to this model.

Other Feature-Based Classifiers

- The exponential model approach is one way of deciding how to weight features, given data.
- It constructs not only classifications, but probability distributions over classifications.
- There are other (good!) ways of discriminating classes: SVMs, boosting, even perceptrons – though these methods are not as trivial to interpret as distributions over classes.

Comparison to Naïve-Bayes

- Naïve-Bayes is another tool for classification:
  - We have a bunch of random variables (data features) which we would like to use to predict another variable (the class):



  - The Naïve-Bayes likelihood over classes is:

$$P(c|d,\lambda) = \frac{P(c)\prod_i P(\phi_i|c)}{\sum_{c'} P(c')\prod_i P(\phi_i|c')} \Rightarrow \frac{\exp\left[\log P(c) + \sum_i \log P(\phi_i|c)\right]}{\sum_{c'} \exp\left[\log P(c') + \sum_i \log P(\phi_i|c')\right]}$$

Naïve-Bayes is just an exponential model.

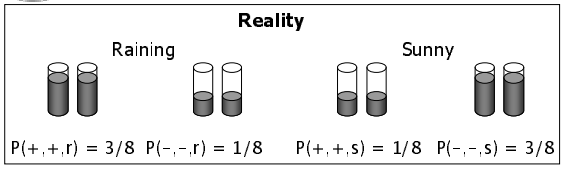
$$\Rightarrow \frac{\exp\left[\sum_i \lambda_{ic} f_{ic}(d,c)\right]}{\sum_{c'} \exp\left[\sum_i \lambda_{ic} f_{ic}(d,c')\right]}$$

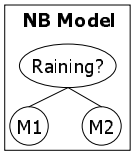
Comparison to Naïve-Bayes

- The primary differences between Naïve-Bayes and maxent models are:

Naïve-Bayes	Maxent
Trained to maximize joint likelihood of data and classes.	Trained to maximize the conditional likelihood of classes.
Features assumed to supply independent evidence.	Features weights take feature dependence into account.
Feature weights can be set independently.	Feature weights must be mutually estimated.
Features must be of the conjunctive $\Phi(d) \wedge c = c_i$ form.	Features need not be of the conjunctive form (but usually are).

Example: Sensors





NB FACTORS:
- P(s) = 1/2
- P(+|s) = 1/4
- P(+|r) = 3/4

PREDICTIONS:
- P(r,+,+) = (½)(¾)(¾)
- P(s,+,+) = (½)(¼)(¼)
- P(r|+,+) = 9/10
- P(s|+,+) = 1/10

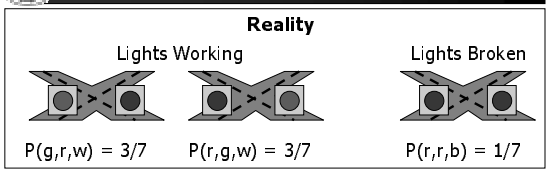
Example: Sensors

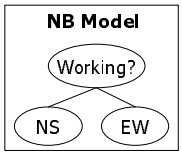
- Problem: NB multi-counts the evidence.

$$\frac{P(r|+...+)}{P(s|+...+)} = \frac{P(r)}{P(s)}\frac{P(+|r)}{P(+|s)}...\frac{P(+|r)}{P(+|s)}$$

- Maxent behavior:
  - Take a model over $(M_1,...M_n,R)$ with features:
    - f_{ri}: M_i=+, R=r weight: λ_{ri}
    - f_{si}: M_i=+, R=s weight: λ_{si}
  - $\exp(\lambda_{ri}-\lambda_{si})$ is the factor analogous to P(+|r)/P(+|s)
  - ... but instead of being 3, it will be $3^{1/n}$
  - ... because if it were 3, E[f_{ri}] would be far higher than the target of 3/8!

Example: Stoplights





NB FACTORS:
- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2
- P(b) = 1/7
- P(r|b) = 1
- P(g|b) = 0

Example: Stoplights

- What does the model say when both lights are red?
  - P(b,r,r) = (1/7)(1)(1) = 1/7 = 4/28
  - P(w,r,r) = (6/7)(1/2)(1/2) = 6/28 = 6/28
  - P(w|r,r) = 6/10!
- We'll guess that (r,r) indicates lights are working!
- Imagine if P(b) were boosted higher, to 1/2:
  - P(b,r,r) = (1/2)(1)(1) = 1/2 = 4/8
  - P(w,r,r) = (1/2)(1/2)(1/2) = 1/8 = 1/8
  - P(w|r,r) = 1/5!
- Changing the parameters, bought conditional accuracy at the expense of data likelihood!

Exponential Model Likelihood

- Maximum Likelihood (Conditional) Models :
  - Given a model form, choose values of parameters to maximize the (conditional) likelihood of the data.
- Exponential model form, for a data set (C,D):

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

Building a Maxent Model

- Define features (indicator functions) over data points.
  - Features represent sets of data points which are distinctive enough to deserve model parameters.
  - Usually features are added incrementally to "target" errors.
- For any given feature weights, we want to be able to calculate:
  - Data (conditional) likelihood
  - Derivative of the likelihood wrt each feature weight
    - Use expectations of each feature according to the model
- Find the optimum feature weights (next part).

The Likelihood Value

- The (log) conditional likelihood is a function of the iid data (C,D) 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:

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

- We can separate this into two components:

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

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

- The derivative is the difference between the derivatives of each component

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)

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)

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:

$$E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$$