Maximum Entropy Classifiers

Dr. Demetrios Glinos
University of Central Florida

CAP6640 – Computer Understanding of Natural Language

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Discriminative Probabilistic Models

- So far, we have examined "generative" models
  - language modeling
  - Naïve Bayes
  - HMM

- Increasing use of conditional or "discriminative" probabilistic models
  - in NLP, speech processing, IR (and machine learning generally)
  - give high accuracy performance
  - easy to incorporate lots of linguistically important features
  - support automatic building of language-independent NLP components

Generative Models

- Given some data { (d,c) } of paired observations d and hidden classes c
- Generative models compute *joint* probabilities over both the observed and the hidden variables

$$P(c, d)$$

  - which are used to generate the observed data from the hidden
  - examples: n-gram models, NB, HMM, PCFG

Discriminative Models

- Given some data { (d,c) } of paired observations d and hidden classes c
- Discriminative models compute *conditional* probabilities over the hidden variables given the data

P(c | d)

  - which are used to generate the observed data from the hidden
  - examples:
    - logistic regression, conditional loglinear or maximum entropy models, conditional random fields
    - also (although not directly probabilistic): perceptrons, NNs, SVMs

Bayes Nets

- Bayes net diagrams draw circles for random variables and lines for direct dependincies
- Some variables are observed; some are hidden
- Each node is a CPT over the incoming arcs
    - CPT serves as a small classifier

Joint vs. Conditional Likelihood

- **Joint model**
  - computes joint probabilities **P(d, c)**
  - tries to maximize joint likelihood
  - trivial to choose weights: just use relative frequencies

- **Conditional model**
  - computes conditional probabilities P(c | d)
  - given the data, models only the conditional probability of the class
  - tries to maximize the conditional likelihood
    - this is more difficult to do, as we shall see
    - but we do it to get increased accuracy

Usefulness of Discriminative Modeling

- Klein and Manning 2002, using Senseval-1 Data on a word sense disambiguation task:

Training Set	
Objective	Accuracy
Joint likelihood	86.8
Conditional likelihood	**98.5**

Test Set	
Objective	Accuracy
Joint likelihood	73.6
Conditional likelihood	**76.1**

- Comparison tests used the ***same*** word/class features and same smoothing

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Features

- Now, we *could* build a MEMM based on just the current word and previous tag

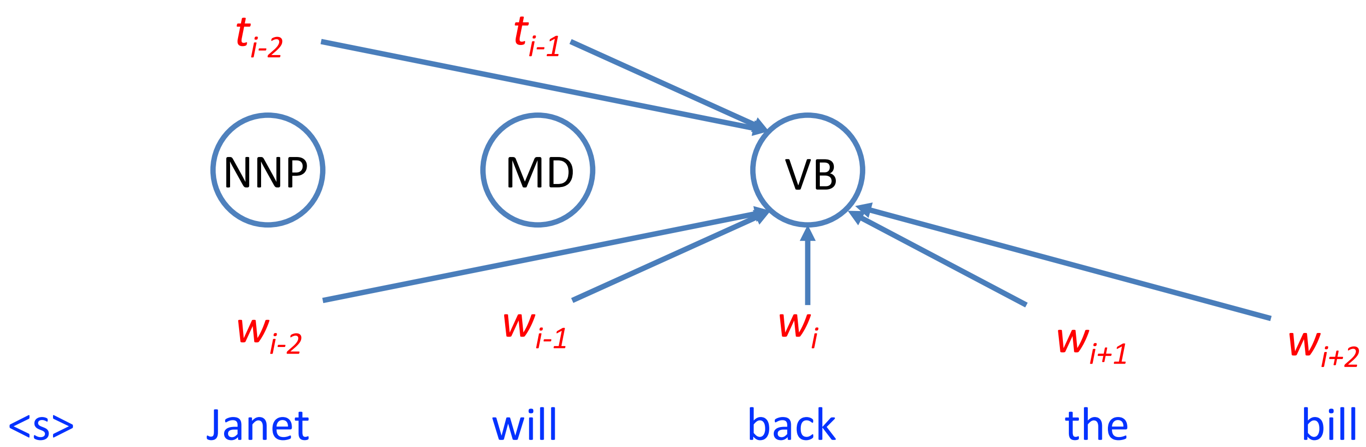
$$\hat{T} = \underset{T}{\operatorname{argmax}} \prod_{i} P(t_i | w_i, t_{i-1})$$

  - But this would be no more accurate than the generative HMM model, and could even be worse

- The attraction of MEMMs is that we can use this formulation to incorporate additional, linguistically relevant features
  - The current word and previous tag are just 2 features
  - Many others features of the context are also possible

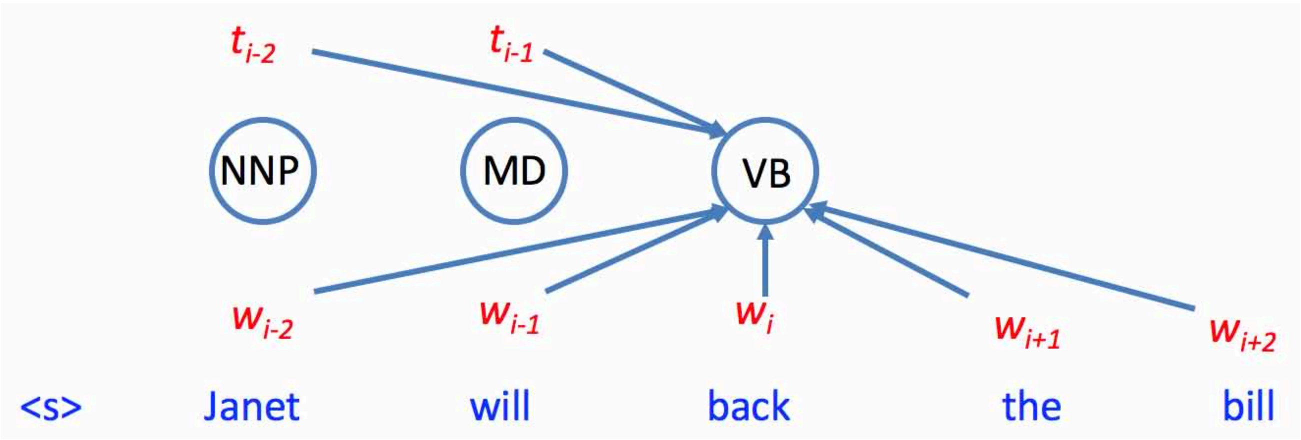
Incorporating More Features

- Typical features used in MEMMs
  - current word
  - neighboring words
  - previous tags
  - combinations of the above
  - plus additional features



Feature Templates

- Feature templates are used to specify combinations of features



- Example templates

$< t_i, w_{i-2} >$ $< t_i, w_{i-1} >$ $< t_i, w_i >$ $< t_i, w_{i+1} >$ $< t_i, w_{i+1} >$ ← *word-based*

$< t_i, t_{i-1} >$ $< t_i, t_{i-2}, t_{i-1} >$ ← *token-based*

$< t_i, t_{i-1}, w_i >$ $< t_i, w_{i-1}, w_i >$ $< t_i, w_i, w_{i+1} >$ ← *combined*

Example: Features generated

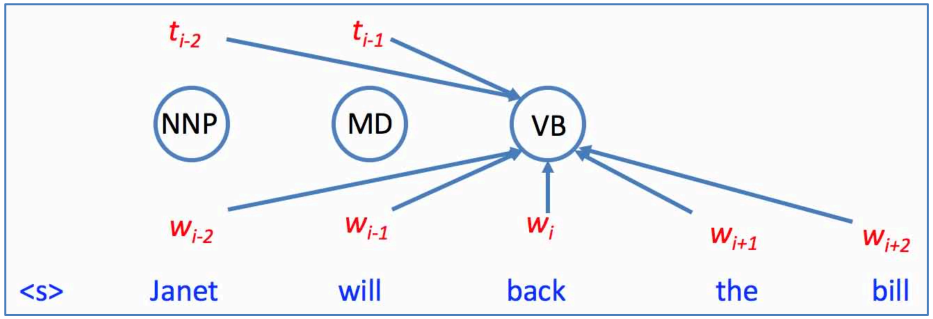
Given

sentence "<s> Janet will back the bill"

Janet tagged NNP

will tagged MD

index i points to word "back"



Features generated

t_i = VB and w_{i-2} = Janet

t_i = VB and w_{i-1} = will

t_i = VB and w_i = back

t_i = VB and w_{i+1} = the

t_i = VB and w_{i+2} = bill

t_i = VB and t_{i-1} = MD

t_i = VB and t_{i-1} = MD and t_{i-2} = NNP

t_i = VB and w_{i-1} = will and w_i = back

t_i = VB and w_i = back and w_{i+1} = the

Word Signature Features

- MEMMs also typically use word signature features for unknown words
  - word-spelling properties
  - word shape

- Examples

  w_i contains a particular prefix (from all prefixes of length ≤ 4)

  w_i contains a particular suffix (from all suffixes of length ≤ 4)

  w_i contains a number

  w_i contains an upper case letter

  w_i contains a hyphen

  w_i is all upper case

  w_i's word shape

  w_i's short word shape

  w_i is upper case and has a digit following a hyphen (e.g., CRC-12)

  w_i is upper case and is followed within 3 words by Co., Inc., etc.

Word Shape Features

- Similar to regular expressions

- Basic word shape characteristics
  - map lower case letters to 'x', upper case to 'X', and digits to 'd'
  - examples
    - **X.X.X.** would match I.M.F.
    - **XXdd-dd** would match DC10-30

- Short word shape
  - same as above, but remove consecutive duplicate specification characters
  - examples
    - **Xd-d** would match DC10-30 and also B747-300

Feature Space

- MEMMs compute
  - Template features for every word seen in the training data set
  - Unknown word features for
    - all words in training set
    - or, just the low frequency ones (below some threshold)
- This produces a very large set of features
- **Feature cutoff**
  - features are not computed if they have a count < 5 in training set

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Feature-Based Linear Classifiers

- Linear classifiers
  - Linear function from feature sets { f_i } to classes { c_j }
  - Assign a weight λ_i to each feature f_i
  - We consider each class for an observed datum d (feature set)
  - For the pair (c, d), the features vote with their weights

$$vote(\ c_j\ ) = \sum_i \lambda_i f_i(c_j, d)$$

  - the winner is the class with the highest score

$$c^* = \underset{c_j}{\operatorname{argmax}} \sum_i \lambda_i f_i(c_j, d)$$

Methods for Choosing the Weights

- Perceptron
  - find a currently misclassified example, and nudge the weights in the dorection of the correct classification
- Margin-based methods
  - e.g., Support Vector Machines
- Exponential methods
  - e.g., log-linear, **maxent**, logistic, Gibbs models
  - make a probabilistic model from the linear voting combination

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

← makes votes positive

← normalizes votes

  - the **weights** are the **parameters** of the probability model

Example: Exponential Model

Given

vote(LOC | in Boulder) = 1.8

vote(PER | in Boulder) = - 0.6

vote(ORG | in Boulder) = 0.3

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

Then

P(LOC | in Boulder, λ) = $e^{1.8}$ / ($e^{1.8}$ + $e^{-0.6}$ + $e^{0.3}$) = 0.586

P(PER | in Boulder, λ) = $e^{-0.6}$ / ($e^{1.8}$ + $e^{-0.6}$ + $e^{0.3}$) = 0.176

P(ORG | in Boulder, λ) = $e^{0.3}$ / ($e^{1.8}$ + $e^{-0.6}$ + $e^{0.3}$) = 0.238

→ relative ordering of vote results is preserved, but normalized to [0,1]

Note on Logistic Regression

- Exponential models
  - Goal of training is to choose the set of parameters { λ_i } that **maximizes the conditional likelihood** of the data
  - To do this, we must construct not only classifications, but probability distributions over classifications
- Related to **logistic regression**
  - Maxent models in NLP are essentially the same as multiclass logistic regression models in statics or machine learning
  - parameterization is slightly different in a way that is useful for NLP-style models that have tons of sparse features
  - features are more general in that a feature is also a function of the class

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Features in NLP

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

$$f_i(c,d) \equiv [\Phi(d) \wedge c = c_j]$$ ← value is 0 or 1

- Each feature selects a data subset and suggests a label for it

Feature Expectations

- We will make use of two kinds of expectations
  - Empirical (actual) expectation of a feature

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

  - Model (predicted) expectation of a feature

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

- Goal of training a maxent model: to have the model expectations match the observed (empirical) expectations

Building a Maxent Model

- Features are often added during model development to target errors
  - Often, the easiest features to think of are the ones that indicate bad combinations
- Then, for any given feature weights, we wish to calculate
  - Data conditional likelihood
  - Derivative of the likelihood with respect to each feature weight
- We can then find the optimum feature weights (discussed later)

Exponential Model Likelihood

- Maximum (conditional) likelihood models
  - Given a model form, choose values of the parameters to maximize the (conditional) likelihood of the data

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

The Likelihood Value

- The (log) conditional likelihood of iid (independent, identically-distributed data) data (C, D) according to a maxent model is a function of the data and the parameters $\boldsymbol{\lambda}$

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

- If there aren't many values of c (i.e., data is sparse), it is easy to calculate

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

The Likelihood Value

- We can separate the last equation into two components

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

$$- \sum_{(c,d)\in(C,D)} \log \sum_{c'} exp \sum_{i} \lambda_i f_i(c',d)$$

- Which is in the form

$$\log \; P(C|D,\lambda) = \mathrm{N}(\lambda) - \mathrm{M}(\lambda)$$

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

Derivative (Part 1): Numerator

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

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

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

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

Thus, the derivative of the numerator is the empirical count(f_i, C)

Derivative (Part 2): Denominator

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

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

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

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

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

Thus, the derivative of the denominator is the predicted count(f_i, λ)

Derivative (Part 3): Log Likelihood

- Combining the ptrvious results, we have

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

- The optimum parameters will be the ones for which each feature's predicted expectation equals its empirical expectation

- Optimal distribution
  - Always exists if feature counts are from actual data
  - Will always be unique (but parameters may not be unique)

- Such models are called maximum entropy models because the optimal parameters correspond to a model with maximum entropy
  - recall Shannon's definition of information entropy: $\mathrm{E(x)} = -ln\left(\frac{1}{P(x)}\right)$

Finding the Optimal Parameters

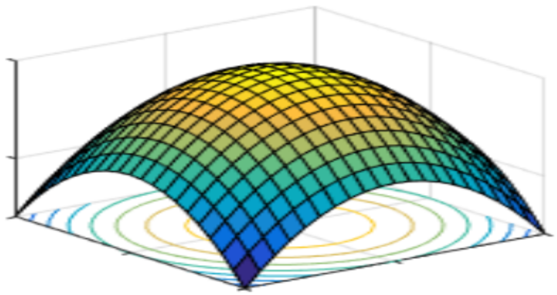
- We wish to choose parameters $\boldsymbol{\lambda_1}$, $\boldsymbol{\lambda_2}$, $\boldsymbol{\lambda_3}$, .. that maximize the conditional log-likelihood of the training data

$$CLogLik(\ D\ ) = \log\ \ P(C|D, \boldsymbol{\lambda})$$

- To be able to find the maximum, we have worked out how to calculate
  - the function value
  - and its partial derivatives (gradients) with respect to each model parameter

Finding the Optimal Parameters

- Use your favorite numerical optimization package
  - typically, one must minimize the negative of CLogLik



1. Gradient descent (GD); stochastic gradient descent (SGD)
2. Generalized iterative scaling (GIS) and improved iterative scaling (IIS)
3. Conjugate gradient (CG), maybe with preconditioning
4. Quasi-Newton methods: limited variable metric (LMVM) methods, e.g., L_BFGS

Most Likely Sequence

- Most likely sequence computed based on words within $\pm\ell$ words and the previous k tags

$$\hat{T} = \underset{T}{\operatorname{argmax}} \prod_{i} P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1})$$

- Given the model (developed from a corpus)
  - Can use greedy algorithm
    - make a hard classification on the first word in the sentence, then on the second word, and so on
  - Can use the Viterbi algorithm
    - same approach as for HMM

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Bidirectionality

- Both HMM and MEMM work essentially left-to-right
  - even though Viterbi allows some influence on current tag by subsequent ones
- Conditional Random Fields (CRF)
  - allow explicit dependence on subsequent tags
  - more powerful
  - but at substantion computational cost
- Other approaches
  - Stanford tagger uses a bidirectional version of MEMM called a cyclic dependency network
  - can also turn any sequence model into a bidirectional model by performing multiple passes (e.g., first pass use only preceding tags, second pass, use tags on right as well)

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Morphologically Rich Languages

- Languages similar to English
  - HMM and MEMM tagger accuracies comparable to English

- Morphologically rich languages
  - e.g., Czech, Hungarian, Turkish
  - for comparable size dictionary
    - Hungarian contains 2x word types as English
    - Turkish contains 4x word types as English
  - larger vocabularies mean more unknown word types and result in degraded performance
  - word morphology also encodes more information, like case and gender, so POS taggers need to label these features as well
  - tagsets can be 4 to 10 times larger than the 50-100 we use for English

Non-Segmented Languages

- Languages like Chinese do not segment written words
  - word segmentation generally applied before tagging
    - but some methods perform segmentation and tagging jointly
  - unknown words are a significant problem
    - despite short (~2.4 characters/unknown ~7.7 char/unk for English)
    - in English, unknown words tend to be proper nouns
    - in Chinese, unknowns tend to be common nouns and verbs
    - features for unknowns
      - use prefixes and suffixes (same as in English)
      - but also use radicals (not used in English)

Today

- Discriminative Models
- Features for Discriminative Models
- Feature-Based Linear Classifiers
- Building a Maxent Model
- Bidirectionality
- POS Tagging for Other Languages
- POS Tagging Performance

Baseline Tagging Performance

- Performance metric: Tag accuracy
  - how many tags are correct
  - about 97% currently
- Baseline method
  - tag every word with its most frequent tag
  - tag unknown words as common nouns
- Baseline performance is 90%
  - reason is many words are unambiguous
  - unambiguous words (like "the", "a", etc.) and for punctuation marks are included in the accuracy calculation

Comparison of POS Tagging Methods

- Rough accuracies for various POS tagging methods
  - on all words
  - on unknown words

Method	Accuracy (total)	Accuracy (unknowns)
Most frequent tag	~90%	~50%
Trigram HMM	~95%	~55%
Maxent P(t|w)	93.7%	82.6%
MEMM tagger	96.9%	86.9%
Bidirectional dependencies	97.2%	90%
Upper bound	~98% (human agreement)	