NYU

Maximum Entropy Models and Feature Engineering

CSCI-GA.2590 – Lecture 6B

Ralph Grishman

So Far …

- So far we have relied primarily on HMMs as our models for language phenomena
    - simple and fast to train and to use
    - effective for POS tagging (one POS ←→ one state)
    - can be made effective for name tagging (can capture context) by splitting states
    - but further splitting could lead to sparse data problems

We want …

- We want to have a more flexible means of capturing our linguistic intuition that certain conditions lead to the increased likelihood of certain outcomes
  - that a name on a 'common first name' list increases the chance that this is the beginning of a person name
  - that being in a sports story increases the chance of team (organization) names
- Maximum entropy modeling (logistic regression) provides one mathematically well-founded method for combining such features in a probabilistic model.

Maximum Entropy

The features provide constraints on the model.

We'd like to have a probability distribution which, outside of these constraints, is as uniform as possible -- has the maximum entropy among all models which satisfy these constraints.

Indicator Functions

- Suppose we have a tagging task, where we want to assign a tag t to a word w based on the 'context' h of w (the words around w, including w itself).
  - In other words, we want to compute p(h,t).
- We will specify a set of K features in the form of binary-valued indicator functions f_i (h, t).
- Example:

  f_1(h, t) = 1 if the preceding word in h is "to" and t = "VB"

  = 0 otherwise

A log-linear model

We will use a log-linear model

$$p(h, t) = (1/Z) \prod_{i=1 \text{ to } K} \alpha_i^{f_i(h, t)}$$

where α_i is the weight for feature i, and Z is a normalizing constant.

If $\alpha_i > 1$, the feature makes the outcome t more likely;
If $\alpha_i < 1$, the feature makes the outcome t less likely;

The goal of the learning procedure is to determine the values of the α_i's so that the expected value of each f_i

$$\Sigma_{h,t}\ p(h, t)\ f_i(h, t)$$

is equal to its average value over the training set of N words (whose contexts are $h_1, \ldots, h_N$):

$$(1/N)\ \Sigma_j\ f_i(h_j, t)$$

Training

Training a ME model involves finding the α_i's.

Unlike HMM training, there is no closed-form solution; an iterative solver is required.

The first ME packages used *generalized iterative scaling.* Faster solvers such as BFGS and L-BFGS are now available.

Overfitting and Regularization

- If a feature appears only a few times, and by chance each time with the same outcome, it will get a high weight, possibly leading to poor predictions on the test data
- this is an example of overfitting
  - not enough data to train many features
- a simple solution is a threshold: a minimum count of a feature—outcome pair
- a fancier approach is regularization—favoring solutions with smaller weights, even if the result is not as good a fit to the training data

Using MaxENT

- MaxEnt is typically used for a multi-class classifier.
- We are given a set of training data, where each datum is labeled with a set of features and a class (tag). Each feature-class pair constitutes an indicator function.
- We train a classifier using this data, computing the αs.
- We can then classify new data by selecting the class (tag) which maximizes p(h ,t).

Using MaxENT

- Typical training data format:

f_1 f_2 f_3 … outcome

f_1 f_2 f_3 … outcome

f_1 f_2 f_3 … outcome

MEMM

Maximum Entropy Markov Model

- a type of Hidden Markov Model (a sequence model)
  - next-state probabilities computed by MaxEnt model
  - MaxEnt model has access to entire sentence, but only to immediate prior state (not to earlier states)
    - first-order HMM
- use Viterbi for tagging
    - time still $O(s^2n)$, but larger factor for MaxEnt eval

Feature Engineering

- The main task when using a MaxEnt classifier is to select an appropriate set of features
  - words in the immediate neighborhood are typical basic features: w_{i-1}, w_i, w_{i+1}
  - patterns constructed for rule-based taggers are likely candidates: w_{i+1} is an initial
  - membership on word lists: w_i is a common first name (from Census)

FE and log-linear models

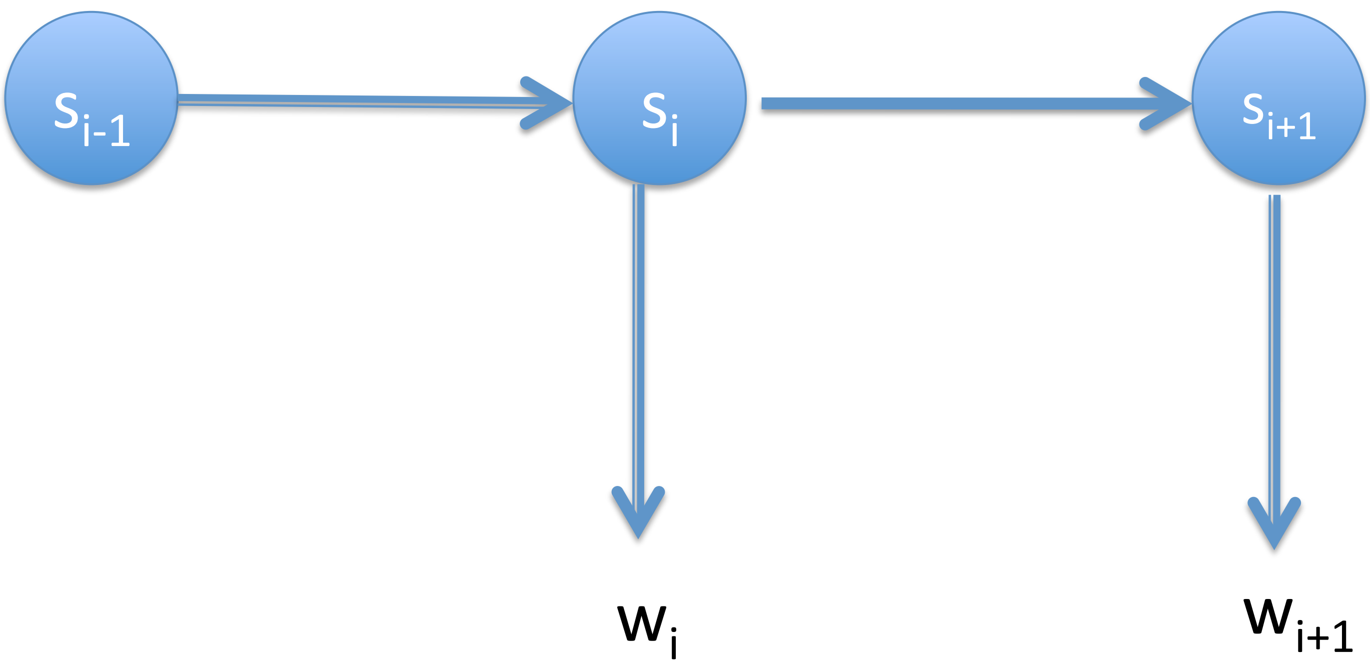
- MaxEnt model combines features multiplicatively
- you may want to include the conjunction of features as a separate feature
  - treat bigrams as separate features: $w_{i-1} \times w_i$

Combining MaxEnt classifiers

- One can even treat the output of individual classifiers as features ("system combination"), potentially producing better performance than any individual classifier
  - weight systems based on
    - overall accuracy
    - confidence of individual classifications
(margin = probability of most likely class –
probability of second most likely class

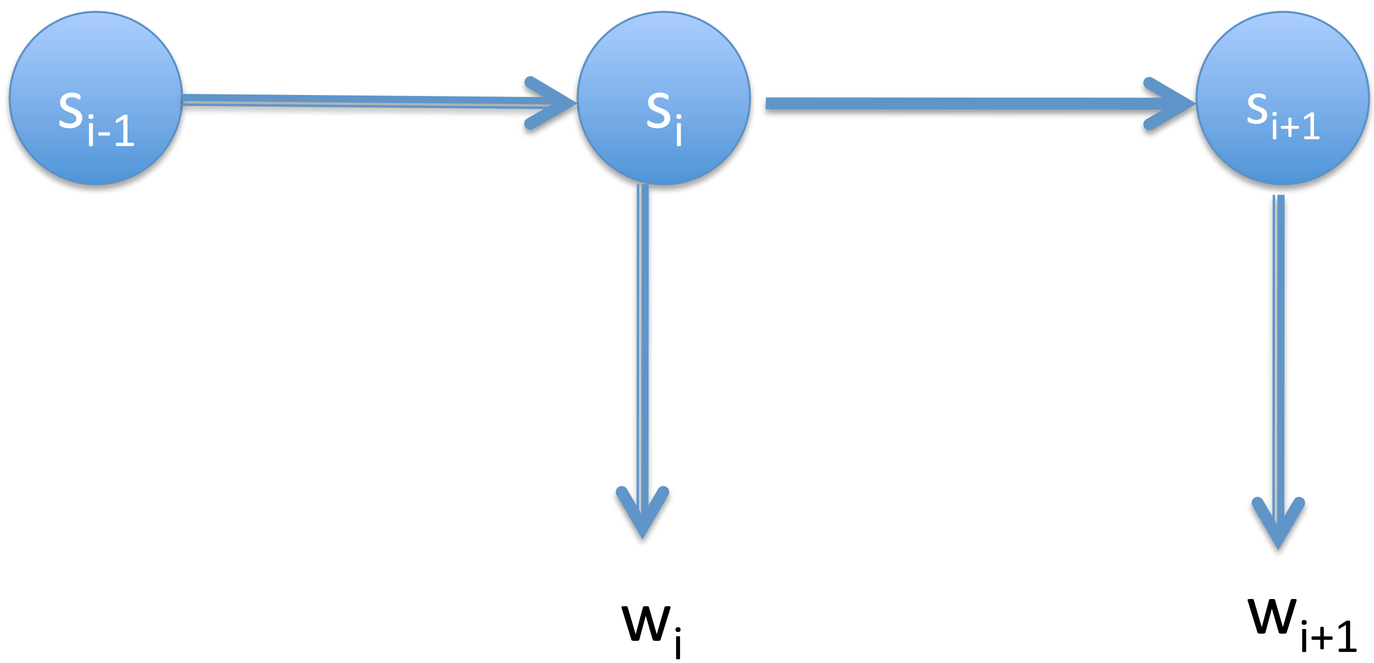
HMM vs MEMM

- HMM: a generative model



HMM vs MEMM

- MEMM: a discriminative model



MaxEnt vs. Neural Network

- MaxEnt
  - simple form for combining inputs (log linear)
  - developer must define set of features to be used as inputs
- Neural Network
  - much richer form for combining inputs
  - can use simpler inputs (in limiting case, words)
  - useful features generated internally as part of training

CRF

- MEMMs are subject to label bias, particularly if there are states with only one outgoing arc
- this problem is avoided by conditional random fields (CRFs), but at a cost of higher training and decoding times
  - linear-chain CRFs reduce decoding time but still have high training times