Natural Language Processing
CSE 325/425

Sihong Xie

Lecture 10:
- Maximum entropy principle
- Maximum entropy Markov model

Maximum entropy principle

Logistic regression models is a special case of the Maximum Entropy models.

What's the "best" estimation of a prob. distribution with partial information?

- Estimate the probabilities of seeing each of the six faces of a dice:
  - Without any information, the best estimation is $\Pr(i) = 1/6$.
    - Entropy $$H(X) = -\sum_{i=1}^{6} \Pr(i) \log \Pr(i)$$
    - For discrete random variables, maximum entropy <=> uniform distributions.
  - If we know that the odd numbers are twice more likely than odd numbers, then
    $$\Pr(1) = \Pr(3) = \Pr(5) = 2/9 \qquad \Pr(2) = \Pr(4) = \Pr(6) = 1/9$$
    - Still have maximum entropy, but also conform to the constraints (what are they?)

Maximum entropy principle

Predict the POS-tag of the word "*zzfish*".

- Without any information, we give equal probabilities to all tags.
- More information: "*zzfish*" can only be tagged as one of {NN, JJ, NNS, VB}:

$$\Pr(NN) = \Pr(JJ) = \Pr(NNS) = \Pr(VB) = 1/4$$

- More information: "*zzfish*" is a sort of noun in 8 out of 10 times:

$$\Pr(NN) = \Pr(NNS) = 2/5 \qquad \Pr(JJ) = \Pr(VB) = 1/10$$

- More information: "*zzfish*" is a verb in 1 out of 20 times:

$$\Pr(NN) = \Pr(NNS) = 2/5 \qquad \Pr(JJ) = 3/20 \qquad \Pr(VB) = 1/20$$

Maximum entropy principle

Logistic regression is a maximum entropy classifier

- Go through the last HW question of CSE326/426.

Maximum entropy Markov model

Multi-class logistic regression can predict a POS-tag for one word, then predict the POS-tag q_t using the *fixed* previous predicted tag q_{t-1} .

$$\Pr(q_t = c | q_{t-1}, o_t; \boldsymbol{\theta}^c) = \frac{1}{Z(q_{t-1}, o_t)} \exp\left\{ \sum_{i=1}^{d} \theta_i^c f_i(q_{t-1}, o_t, q_t = c) \right\}$$

Secretariat is expected to race tomorrow.

NNP → VBZ → VBN → TO → VB → RB

- Pro: simple and surprisingly good results.
- Cons:
  - Can’t use later predictions to correct previous predictions.
  - Errors propagate.

Maximum entropy Markov model

Marry logistic regression and Markov model

- Predict the whole sequence of POS-tags via Viterbi algorithm.
- When predicting a tag, the decision is made based on information from both directions.
  - Slightly modify the multi-class logistic regression model to predict q_t based on all possible q_{t-1} .

$$\Pr(q_t = c | q_{t-1}, o_t; \boldsymbol{\theta}^c) = \frac{1}{Z(q_{t-1}, o_t)} \exp\left\{\sum_{i=1}^{d} \theta_i^c f_i(q_{t-1}, o_t, q_t = c)\right\}$$

Secretariat is expected to race tomorrow.

Predict optimal sequences

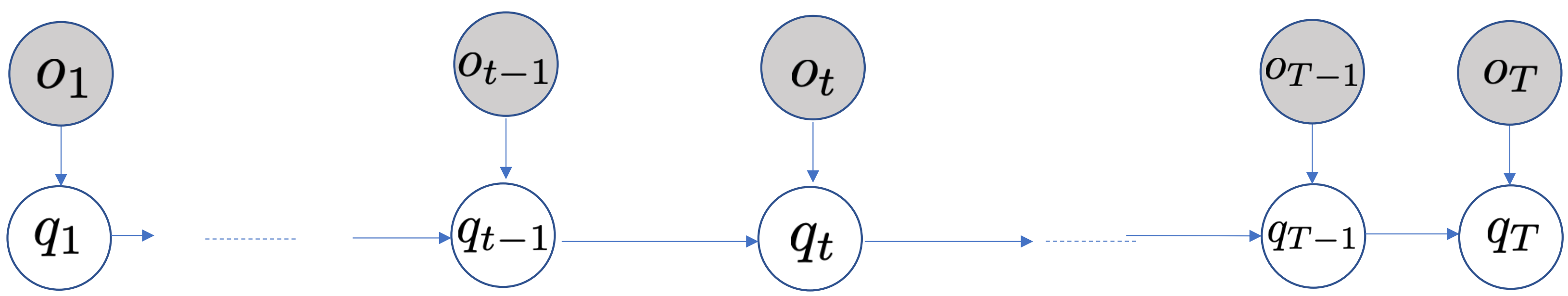
Viterbi algorithm (for MEMM): compute maximum probability and the optimal sequence

1. Initialize $v_1(i)$ for each value i of the first hidden state q_1 .
2. for $t = 2, \ldots, T$

   for $j = 1, \ldots, N$

   compute $v_t(j) = \cancel{\max_k v_{t-1}(k) a_{kj} b_j(o_t)}$

   record back-pointers $p_t(j) = \arg\max_k \cancel{v_{t-1}(k) a_{kj} b_j(o_t)}$
3. Backtracking to find Q^*
4. Return $\Pr(Q^*|O) = \max_j v_T(j)$



Maximum entropy Markov model

Training of MEMM

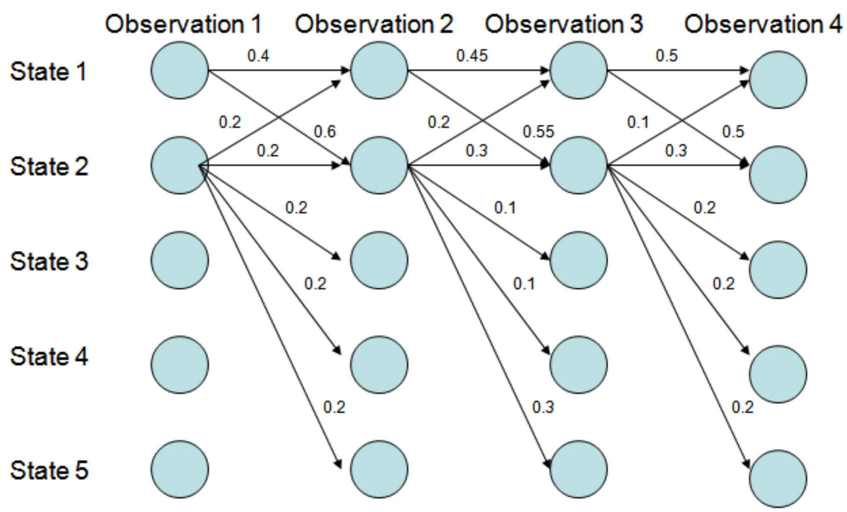
- Can only train with some labeled data (supervised or semi-supervised learning).
- Design feature functions $\mathbf{f}(q_{t-1}, o_t, q_t = c)$ for all possible tags and words.
- Evaluate the feature functions on the POS-tag sentences.
  - Go through each word, based on the observed word, the previous and the current POS-tags, compute the features.
  - Usually need to consider rich features beyond the word and tag identities.
    - consider suffixes, prefixes, Captipalizations, lower-cases, hyphens, dictionaries, tags that are farther away.
- Parameters: $\boldsymbol{\theta}^c, c \in \{\text{All POS-tags}\}$

Maximum entropy Markov model

Drawbacks of MEMM:

- Labeling bias in the tag probabilities, due to local normalization

$$\Pr(q_t = c | q_{t-1}, o_t; \boldsymbol{\theta}^c) = \frac{1}{Z(q_{t-1}, o_t)} \exp\left\{ \sum_{i=1}^{d} \theta_i^c f_i(q_{t-1}, o_t, q_t = c) \right\}$$



1 tends to move to 2 then stay with 2:

$\Pr(1 \to 1 \to 1 \to 1) =$

$\Pr(1 \to 2 \to 2 \to 2) =$

Next lecture:
address this issue using Conditional Random Fields