Conditional Random Fields and Structured Perceptron

Instructor: Wei Xu
Ohio State University
CSE 5525

Some slides from Eric Xing and Michael Collins
Previously, we saw MEMMs...

\[
P(t_1 \ldots t_n | w_1 \ldots w_n) = \prod_{i=1}^{n} q(t_i | t_{i-1}, w_1 \ldots w_n, i)
\]

\[
= \prod_{i=1}^{n} \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \ldots w_n, i)}}
\]
MEMMMs: The Label Bias Problem

\[ P(t_1, \ldots, t_n | w_1 \ldots w_n) = \prod_{i=1}^{n} \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \ldots w_n, i)}} \]

These are forced to sum to 1 Locally

Q: Is that really necessary?
MEMMs: The Label Bias Problem

\[ P(t_1, \ldots, t_n | w_1 \ldots w_n) = \prod_{i=1}^{n} \frac{e^v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)}{\sum_{t'} e^v \cdot f(t', t_{i-1}, w_1 \ldots w_n, i)} \]

These are forced to sum to 1 Locally

Q: Is that really necessary?

MEMMs: The Label Bias Problem
MEMMs: The Label Bias Problem

What the local transition probabilities say:

- State 1 almost always prefers to go to state 2
- State 2 almost always prefer to stay in state 2
MEMMs: The Label Bias Problem

**Diagram:**
- **Observation 1**
  - State 1: 0.4
  - State 2: 0.2
  - State 3: 0.2
  - State 4: 0.2
  - State 5: 0.2

- **Observation 2**
  - State 1: 0.45
  - State 2: 0.6
  - State 3: 0.2
  - State 4: 0.2
  - State 5: 0.6

- **Observation 3**
  - State 1: 0.5
  - State 2: 0.3
  - State 3: 0.55
  - State 4: 0.1
  - State 5: 0.5

- **Observation 4**
  - State 1: 0.5
  - State 2: 0.3
  - State 3: 0.5
  - State 4: 0.3
  - State 5: 0.5

**Probability of path 1->1->2->2:**
- $0.4 \times 0.55 \times 0.3 = 0.066$

**Other paths:**
- 1->1->1->1: 0.09
- 2->2->2->2: 0.018
- 1->2->1->2: 0.06
MEMMs: The Label Bias Problem

Most Likely Path: 1 -> 1 -> 1 -> 1

• Although **locally** it seems state 1 wants to go to state 2 and state 2 wants to remain in state 2.

• why?
MEMMs: The Label Bias Problem

Most Likely Path: 1 -> 1 -> 1 -> 1

- State 1 has only two transitions but state 2 has 5:
  - Average transition probability from state 2 is lower
MEMMs: The Label Bias Problem

Label bias problem in MEMM:
- Preference of states with lower number of transitions over others
Solution: Do not normalize probabilities locally

From local probabilities to local potentials

- States with lower transitions do not have an unfair advantage!
MEMMs: The Label Bias Problem

• States with low entropy distributions effectively ignore observations

\[
P(t_1, \ldots, t_n | w_1 \ldots w_n) = \prod_{i=1}^{n} \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \ldots w_n, i)}}
\]

These are forced to sum to 1 Locally

Q: is that really necessary?
From MEMMs to Conditional Random Fields

\[ P(t_1, \ldots, t_n | w_1 \ldots w_n) \propto \prod_{i=1}^{n} e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)} \]

Q: how can we make the distribution over tag sequences sum to 1?
From MEMMs to Conditional Random Fields

\[ P(t_1, \ldots, t_n|w_1 \ldots w_n) = \frac{1}{Z(v, w_1, \ldots, w_n)} \prod_{i=1}^{n} e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)} \]

\[ Z(v, w_1, \ldots, w_n) = \sum_{t_1, \ldots, t_n} \prod_{i=1}^{n} e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)} \]

CRF uses global normalizer to overcome the label bias problem of MEMM
- MEMMs use a per-state exponential model
- CRFs have a single exponential model for the joint probability of the entire label sequence
Conditional Random Fields

• Learning:
  - similar to MEMM (gradient descent or MAP perceptron)

• Inference:
  - similar to HMM (dynamic programming)

  1) given model parameters, find best tag sequence
  2) during learning, compute marginal probabilities and the Z
Gradient ascent

Loop While not converged:
For all features $j$, compute and add derivatives

$$w_j^{\text{new}} = w_j^{\text{old}} + \eta \frac{\partial}{\partial w_j} \mathcal{L}(w)$$

$\mathcal{L}(w)$: Training set log-likelihood

$$\left( \frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \cdots, \frac{\partial \mathcal{L}}{\partial w_n} \right)$$
Gradient ascent
Gradient of Log-Linear Models

\[
\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{D} f_j(y_i, d_i) - \sum_{i=1}^{D} \sum_{y \in Y} f_j(y, d_i) P(y \mid d_i)
\]
MAP-based Learning (perceptron)

\[ \frac{\partial L}{\partial w_j} \approx \sum_{i=1}^{D} f_j(y_i, d_i) - \sum_{i=1}^{D} f_j(\arg \max_{y \in Y} P(y|d_i), d_i) \]
Conditional Random Field Gradient

\[
\frac{\partial L}{\partial w_j} = \sum_{i=1}^{D} \sum_k f_j(t_k, t_{k-1}, w_1, \ldots, w_n, k) - \\
\sum_{i=1}^{D} \sum_{t_1, \ldots, t_n} \sum_k f_j(t_k, t_{k-1}, w_1, \ldots, w_n, k) P(t_1, \ldots, t_n | w_1, \ldots, w_n)
\]

Tractable! Can be computed with the dynamic programming (Forward-Backward) algorithm
MAP-based learning (perceptron)

\[
\frac{\partial L}{\partial w_j} \approx \sum_{i=1}^{D} \sum_{k} f_j(t_k, t_{k-1}, w_1, \ldots, w_n, k) - \\
\sum_{i=1}^{D} \sum_{k} f_j(\text{arg max}_{t_1, \ldots, t_n} P(t_1, \ldots, t_n \mid w_1, \ldots, w_n), w_1, \ldots, w_n, k)
\]
Training a Tagger using the Perceptron Algorithm

can be computed with the dynamic programming (Viterbi) algorithm

**Algorithm 40 StructuredPerceptronTrain(D, MaxIter)**

1: \( w \leftarrow 0 \)  
   // initialize weights
2: for \( iter = 1 \ldots MaxIter \) do  
3:    for all \( (x, y) \in D \) do  
4:       \( \hat{y} \leftarrow \arg\max_{\hat{y} \in Y(x)} w \cdot \phi(x, \hat{y}) \)  
5:       // compute prediction  
6:       if \( \hat{y} \neq y \) then  
7:          \( w \leftarrow w + \phi(x, y) - \phi(x, \hat{y}) \)  
8:          // update weights  
9:       end if  
10:    end for  
11:  end for  
12: return \( w \)  
   // return learned weights
An Example

Say the correct tags for $i$’th sentence are

the/DT man/NN bit/VBD the/DT dog/NN

Under current parameters, output is

the/DT man/NN bit/NN the/DT dog/NN

Assume also that features track: (1) all bigrams; (2) word/tag pairs

Parameters incremented:

$\langle$NN, VBD$\rangle, \langle$VBD, DT$\rangle, \langle$VBD $\rightarrow$ bit$\rangle$

Parameters decremented:

$\langle$NN, NN$\rangle, \langle$NN, DT$\rangle, \langle$NN $\rightarrow$ bit$\rangle$
Experiments

- Wall Street Journal part-of-speech tagging data
  
  Perceptron = 2.89% error, Log-linear tagger = 3.28% error

- [Ramshaw and Marcus, 1995] NP chunking data
  
  Perceptron = 93.63% accuracy, Log-linear tagger = 93.29% accuracy
Summary

**HMM**

**MEMM**

\[
P(y_{1:n} | x_{1:n}) = \prod_{i=1}^{n} P(y_i | y_{i-1}, x_{1:n}) = \prod_{i=1}^{n} \frac{\exp(w^T f(y_i, y_{i-1}, x_{1:n}))}{Z(y_{i-1}, x_{1:n})}
\]

**CRF**

\[
P(y_{1:n} | x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, x_{1:n}) = \frac{1}{Z(x_{1:n}, w)} \prod_{i=1}^{n} \exp(w^T f(y_i, y_{i-1}, x_{1:n}))
\]