Conditional Random Fields

Alan Ritter

CSE 5525

Last time we saw MEMMs…

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

$$= \prod_{i=1}^{n} \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \ldots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \ldots w_n, i)}}$$

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

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

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

Gradient ascent

Loop While not converged:

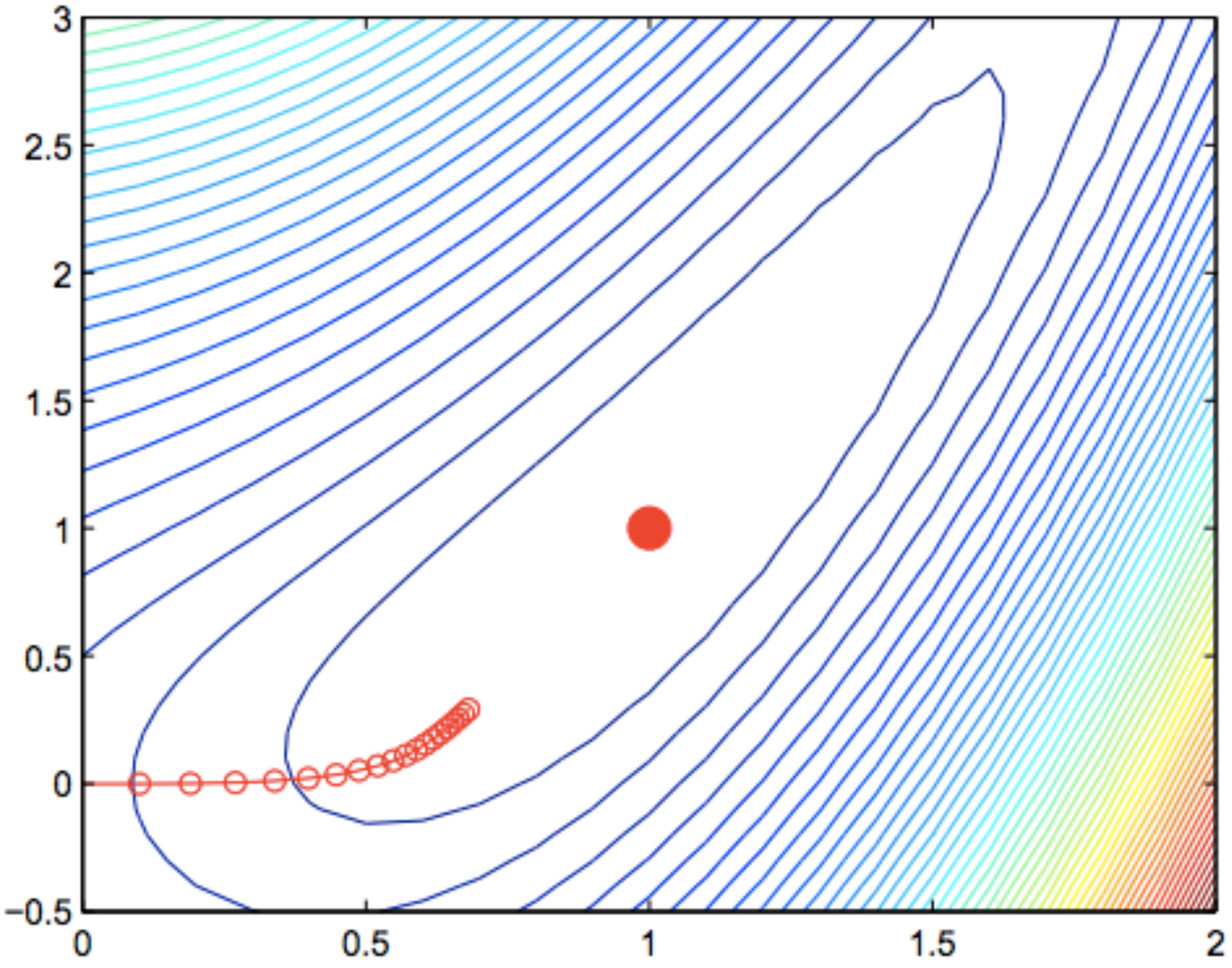
For all features **j**, compute and add derivatives

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

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

$$\left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \ldots, \frac{\partial \mathcal{L}}{\partial w_n} \right)$$

Gradient ascent



Gradient of Log-Linear Models

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{D} f_j(y_i, d_i) - \sum_{i=1}^{D} \sum_{y \in Y} f_j(y, d_i) P(y|d_i)$$

MAP-based Learning (perceptron)

$$\frac{\partial \mathcal{L}}{\partial w_j} \approx \sum_{i=1}^{D} f_j(y_i, d_i) - \sum_{i=1}^{D} f_j(\arg\max_{y \in Y} P(y|d_i), d_i)$$

Conditional Random Field Gradient (log-linear model)

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{D} \sum_{k} f_j(t_k, t_{k-1}, w_1, \ldots, w_n, k) -$$

$$\sum_{i=1}^{D} \sum_{t_1, \ldots, t_n} \sum_{k} f_j(t_k, t_{k-1}, w_1, \ldots, w_n, k) P(t_1, \ldots, t_n | w_1, \ldots, w_n)$$

MAP-based learning (perceptron)

$$\frac{\partial \mathcal{L}}{\partial w_j} \approx \sum_{i=1}^{D} \sum_{k} f_j(t_k, t_{k-1}, w_1, \ldots, w_n, k) -$$

$$\sum_{i=1}^{D} \sum_{k} f_j(\arg \max_{t_1, \ldots, t_n} P(t_1, \ldots, t_n | w_1, \ldots, w_n), w_1, \ldots, w_n, k)$$

Training a Tagger Using the Perceptron Algorithm

Inputs: Training set $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \ldots n$.

Initialization: $\mathbf{v} = 0$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$$z_{[1:n_i]} = \arg \max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \mathbf{v} \cdot \mathbf{f}(w^i_{[1:n_i]}, u_{[1:n_i]})$$

$z_{[1:n_i]}$ can be computed with the dynamic programming (Viterbi) algorithm

If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then

$$\mathbf{v} = \mathbf{v} \quad + \quad \mathbf{f}(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \mathbf{f}(w^i_{[1:n_i]}, z_{[1:n_i]})$$

Output: Parameter vector $\mathbf{v}$.

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:

$$\langle \text{NN, VBD} \rangle, \langle \text{VBD, DT} \rangle, \langle \text{VBD} \rightarrow \text{bit} \rangle$$

Parameters decremented:

$$\langle \text{NN, NN} \rangle, \langle \text{NN, DT} \rangle, \langle \text{NN} \rightarrow \text{bit} \rangle$$

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