Exercise 2
Structured Prediction for PoS Tagging
Advanced Practical Course In Machine Learning

Alon Netser
31160253-6

December 8, 2019

1 Theoretical Questions

1.1 MEMM "contains" HMM

Cancelled.

1.2 Higher Order Markov Model

Cancelled.

1.3 Energy Based Model Gradient

1.3.1 Part 1

As we saw in class, the probability of a single transition $\Pr[y_t \mid y_{t-1}, x_{1:T}]$ is the following expression:

$$\Pr[y_t \mid y_{t-1}, x_{1:T}] = \frac{1}{Z} \cdot e^{w^T \cdot \phi(y_t, y_{t-1}, x_{1:T})}$$

where

$$Z = \sum_{y_t} e^{w^T \cdot \phi(y_t, y_{t-1}, x_{1:T})}$$

This means that the energy function of the MEMM (for a single transition) is

$$-E(y_t, y_{t-1}, x_{1:T}; \theta) = w^T \cdot \phi(y_t, y_{t-1}, x_{1:T})$$

So the energy function equals minus the dot-product between the weight vector w and the vector representing the transition.

1.3.2 Part 2

Let's calculate the log probability of a general energy-based model:

$$\begin{aligned}
\log \Pr_\theta[x] &= \log(\frac{1}{Z} \cdot e^{-E(x;\theta)}) \\
&= -E(x;\theta) - \log(Z) \\
&= -E(x;\theta) - \log(\sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)})
\end{aligned}$$

Now, taking the gradient of this term with respect to the parameters of the model θ we get

$$\begin{aligned}
\frac{\partial}{\partial\theta} \log \Pr_\theta[x] &= \frac{\partial}{\partial\theta}(-E(x;\theta) - \log(\sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)})) \\
&= -\frac{\partial}{\partial\theta}E(x;\theta) - \frac{\partial}{\partial\theta}\log(\sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)})) \\
&= -(\frac{\partial}{\partial\theta}E(x;\theta) + \frac{\partial}{\partial\theta}\log(\sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)})) \\
&= -(\frac{\partial}{\partial\theta}E(x;\theta) + \frac{1}{\sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)}} \cdot \sum_{x' \sim \mathbb{P}_\theta} \frac{\partial}{\partial\theta} e^{-E(x';\theta)}) \\
&= -(\frac{\partial}{\partial\theta}E(x;\theta) - \frac{1}{Z} \cdot \sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)} \cdot \frac{\partial}{\partial\theta}E(x';\theta)) \\
&= -\frac{\partial}{\partial\theta}E(x;\theta) + \frac{1}{Z} \cdot \sum_{x' \sim \mathbb{P}_\theta} e^{-E(x';\theta)} \cdot \frac{\partial}{\partial\theta}E(x';\theta)
\end{aligned}$$

Plugging this term in the expectation with respect to the distribution $\mathcal{D}$ gives:

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\frac{\partial}{\partial\theta}\log\Pr_{\theta}[x]] &= \sum_{x\sim\mathcal{D}}\Pr_{\mathcal{D}}[x]\cdot\frac{\partial}{\partial\theta}\log\Pr_{\theta}[x] \\
&= \sum_{x\sim\mathcal{D}}\Pr_{\mathcal{D}}[x]\cdot\left(-\frac{\partial}{\partial\theta}E(x;\theta)+\frac{1}{Z}\cdot\sum_{x'\sim\mathbb{P}_\theta}e^{-E(x';\theta)}\cdot\frac{\partial}{\partial\theta}E(x';\theta)\right) \\
&= \sum_{x\sim\mathcal{D}}\left(\Pr_{\mathcal{D}}[x]\cdot\sum_{x'\sim\mathbb{P}_\theta}\frac{1}{Z}\cdot e^{-E(x';\theta)}\cdot\frac{\partial}{\partial\theta}E(x';\theta)\right)-\sum_{x\sim\mathcal{D}}\Pr[x]\cdot\frac{\partial}{\partial\theta}E(x;\theta) \\
&= \sum_{x\sim\mathcal{D}}\left(\Pr_{\mathcal{D}}[x]\cdot\sum_{x'\sim\mathbb{P}_\theta}\Pr_{\theta}[x']\cdot\frac{\partial}{\partial\theta}E(x';\theta)\right)-\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\frac{\partial}{\partial\theta}E(x;\theta)\right] \\
&= \sum_{x\sim\mathcal{D}}\left(\Pr_{\mathcal{D}}[x]\cdot\mathop{\mathbb{E}}_{x'\sim\mathbb{P}_\theta}\left[\frac{\partial}{\partial\theta}E(x';\theta)\right]\right)-\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\frac{\partial}{\partial\theta}E(x;\theta)\right] \\
&= \mathop{\mathbb{E}}_{x'\sim\mathbb{P}_\theta}\left[\frac{\partial}{\partial\theta}E(x';\theta)\right]\cdot\sum_{x\sim\mathcal{D}}\Pr_{\mathcal{D}}[x]-\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\frac{\partial}{\partial\theta}E(x;\theta)\right] \\
&= \mathop{\mathbb{E}}_{x'\sim\mathbb{P}_\theta}\left[\frac{\partial}{\partial\theta}E(x';\theta)\right]-\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\frac{\partial}{\partial\theta}E(x;\theta)\right]
\end{aligned}$$

1.4 (Bonus) MLE for HMM

Let's denote the number of words by N, the number of PoS tags by P. The number $t_{i,j}$ is the probability the transition from the PoS i to the PoS j (we assume the PoS tags are numbered from 1 to P).

In order to solve this question we'll use Lagrange multipliers. We want to find $t = arg\max_t\{\ell(S,\theta)\}$ (t is a vector containing an entry for each two PoS tags i,j) subject to the constraints $\sum_{j=1}^{P} t_{i,j} = 1$ for every $0 \le i < p$. The constraints force the transition from the PoS i to all other PoS tags to be a distribution.

$$\frac{\partial}{\partial t_{p,q}}\left(\sum_{i=1}^{N}\sum_{t=1}^{T(i)}\log(t_{y_{t-1}^{(i)},y_t^{(i)}})-\sum_{i=1}^{N}\sum_{t=1}^{T(i)}\log(e_{x_t^{(i)},y_t^{(i)}})-\lambda\cdot\left(\sum_{k=1}^{P}t_{p,k}-1\right)\right)=0$$

Note that the derivative of log is one divided by what's inside the log, and the amount of times the variable $t_{p,q}$ appears in the expression is $\#(y_p \to y_q)$ and one more time in the sime over k.

Therefore we get that the deriviative equals zero iff

$$\frac{\#(y_p \to y_q)}{t_{p,q}} - \lambda = 0$$

and this is iff

$$t_{p,q} = \frac{\#(y_p \to y_q)}{\lambda}$$

Now, using the constraint $\sum_{k=1}^{P} t_{p,k} = 1$ we get

$$\sum_{k=1}^{P} \frac{\#(y_p \to y_k)}{\lambda} = 1$$

And this is iff

$$\lambda = \sum_{k=1}^{P} \#(y_p \to y_k)$$

And plugging λ in the expression we found above for $t_{p,q}$ gives us

$$t_{p,q} = \frac{\#(y_p \to y_q)}{\sum_{k=1}^{P} \#(y_p \to y_k)}$$

as claimed.

2 Practical Exercise

I used NumPy string arrays to store the dataset (instead of list of lists). This helped me do a lot of things in a vectorized way (such as counting the occurrences of elements, handling rare-words, etc). Since the dataset contains sentences of different lengths, I found the maximal length of a sentence (denote by ℓ_{max}), and created a 2D array of strings, that contains ℓ_{max} columns (and the amount of rows is of course the amount of sentences in the dataset). Then, each sentence id padded with empty-strings - from the last word until the ℓ_{max} cell.
Note that since storing strings in a NumPy array requires knowing the maximal length of a string, as a pre-processing step the maximal length of a string is calculated, both in the words and in the PoS tags.

The train/test was done randomly, meaning that the training-set is sampled uniformly from the dataset. The train/test ratio I chose was 90%/10%.

The rare words were replaced with a RARE_WORD symbol, and I chose to define a word as 'rare' if the number of occurrences in the dataset is less than 5 times.

2.1 Baseline Model

The baseline model reaches about 90.5-91% accuracy on the test-set (depends on the train/test random split). The running-time is pretty fast - it takes less than a minute (on my personal laptop).

2.2 HMM

The HMM model reaches about 94.5-95% accuracy on the test-set. The running-time is also pretty fast -t takes about one minute (on my personal laptop).

Here are some sentences (and their corresponding PoS tags) drawn according to the trained model parameters:

- – Whittle is called for officer

  - NNP VBZ VBN IN NN

- - at it 's stock-market
  - IN PRP POS NN

- - the further songs who is the most serious Minneapolis , those Dutch buying with a accounting it were to predict
  - DT JJ NNS WP VBZ DT RBS JJ NNP , DT JJ NN IN DT NN PRP VBD TO VB

The drawn sentences of course do not seem logical in the language-sense, but at least in the sense of the Part-of-speech tags they are alright.

2.3 MEMM

The MEMM model depends on the mapping function ϕ. I implemented a basic mapping function ϕ, that given a triplet y_{t-1}, y_t, x_y returns exactly 2 indices (that correspond to 1 in the binary representation vector of that triplet).

- The first index corresponds to the transition from the PoS tag y_{t-1} to the PoS tag y_t, and the index equals $y_{t-1} \cdot N_{PoS-tags} + y_t$. This means that if the index of the binary vector in the coordinate $0 \leq q \leq N^2_{PoS-tags}$ is 1, it means that this triplet contains a transition from the PoS tag $(q\%N_{PoS-tags})$ to the PoS tag $(q//N_{PoS-tags})$.

- The second index corresponds to the emission of the word and the PoS tag. Having 1 in the index $0 \leq r \leq N_{PoS-tags} \cdot N_{words}$ corresponds to having the word $r\%N_{words}$ with the PoS tag $r//N_{words}$.

The MEMM model with this basic mapping function reaches about 91.5%-92%. Adding more features will probably do better.

2.4 Model Comparison

After splitting to 90%/10% train/test, I tried to train on increasing proportion of the data (10%, 25%, 90%). Each experiment was done 5-10 times, and the results depend on the randomness of the train/test split. The reported values are about the mean of the accuracies the models reached among the runs.

Note that the accuracy is calculated ignoring the "success" of predicting the START_STATE and END_STATE, because they are not really a part of the sentence.

Note that the MEMM was trained for 1 epoch only, and the mapping function used is quite naive. Adding more features and/or training for more epochs will probably do much better.

- Training on a portion of 10% of the training-data reaches 87.5% in the baseline model, 90% in the HMM model, and about 83-84% in the MEMM model.

- Training on a portion of 25% of the training-data reaches about 89.5% in the baseline model, about 93% in the HMM model, and about 87-88% in the MEMM model.

- Training on all of the training-data (90% of the total data) reaches about 90.5% in the baseline model, about 94.3% in the HMM model, and about 91-92% in the MEMM model.