Chapter 7

Unsupervised Sequence Labeling

7.1 Problem

In the last chapter, we saw how to train finite transducers to learn how to tag sequences from training data consisting of (character or word) sequences together with their correct tag sequences. In this (short) chapter, we'll consider how to do this using training data consisting of character or word sequences *only*, without any correct tag sequences.

For example:

- We want to decipher a writing system we've never seen before, and we want to know what the vowels and consonants are.
- We want to find part-of-speech tags for a language that doesn't have any part-of-speech tag training data.

Is this possible? Yes, but, to be perfectly honest, it often doesn't work all that well. The reason is that with absolutely no training data, there's nothing to tell the model what we want it to find, and the model is free to go off and find something different.

7.2 Model

The model we use is a hidden Markov model, just like before.

$$P(\mathbf{t}, \mathbf{w}) = P(\mathbf{t}) P(\mathbf{w} \mid \mathbf{t}) \tag{7.1}$$

$$P(\mathbf{t}) = p(t_1 \mid \texttt{<s>}) \times \left(\prod_{i=2}^{n} p(t_i \mid t_{i-1}) \right) \times p(\texttt{</s>} \mid t_n) \tag{7.2}$$

$$P(\mathbf{w} \mid \mathbf{t}) = \prod_{i=1}^{n} p(w_i \mid t_i). \tag{7.3}$$

Here, the tags t are drawn from the set $\{1, \ldots, T\}$. You have to choose T in advance.

7.3 Example

Suppose our alphabet is just $\Sigma = \{\text{a}, \text{b}, \text{c}, \text{o}\}$, and the training data is

cabocoba.

Let $T = 2$. Consider a model that doesn't use one of the tags:

$$p(1 \mid \texttt{<s>}) = 1 \qquad p(2 \mid \texttt{<s>}) = 0$$
$$p(1 \mid 1) = 8/9$$
$$p(2 \mid 1) = 0$$
$$p(\texttt{</s>} \mid 1) = 1/9$$
$$p(\text{a} \mid 1) = 1/4$$
$$p(\text{b} \mid 1) = 1/4$$
$$p(\text{c} \mid 1) = 1/4$$
$$p(\text{o} \mid 1) = 1/4$$

Then the likelihood (probability of the training data) is

$$\left(\tfrac{8}{9} \cdot \tfrac{1}{4}\right)^8 \cdot \tfrac{1}{9} = 6.6 \cdot 10^{-7}.$$

Now, consider a model that uses both tags:

$$p(1 \mid \texttt{<s>}) = 1 \qquad p(2 \mid \texttt{<s>}) = 0$$
$$p(1 \mid 1) = 0 \qquad p(1 \mid 2) = 1$$
$$p(2 \mid 1) = 1 \qquad p(2 \mid 2) = 0$$
$$p(\texttt{</s>} \mid 1) = 0 \qquad p(\texttt{</s>} \mid 2) = 1/3$$
$$p(\text{a} \mid 1) = 0 \qquad p(\text{a} \mid 2) = 1/3$$
$$p(\text{b} \mid 1) = 1/2 \qquad p(\text{b} \mid 2) = 0$$
$$p(\text{c} \mid 1) = 1/2 \qquad p(\text{c} \mid 2) = 0$$
$$p(\text{o} \mid 1) = 0 \qquad p(\text{o} \mid 2) = 1/3$$

Then the likelihood (probability of the training data) is

$$\left(\tfrac{1}{2} \cdot \tfrac{1}{3}\right)^4 \cdot \tfrac{1}{3} = 2.6 \cdot 10^{-4}.$$

This model is much better because it groups the characters into two classes that alternate with each other, making it much easier to predict what the next character is.

So, the model's only job is to be the best character predictor it can be, and it gets to use the tags as a way to help it make its predictions. The question is – will the way that model uses the tags correspond to the tags that we wanted it to learn? Or anything meaningful at all?

For this particular problem, yes, it turns out that an HMM can learn that characters should be classed into vowels and consonants. For other problems, what the HMM learns may be harder to interpret.

7.4 Training

Our training data is a collection of word sequences. Let $\mathbf{w}_i$ be the i'th word sequence. When our training data was word sequences together with tag sequences, it was easy to estimate all the model probabilities by counting and dividing. But now, we don't know what the tag sequences are, so we don't know what to count. So training is more difficult.

7.4.1 Objective function

As always, we want to maximize the log-likelihood over the training data:

$$\log L = \sum_i \log P(\mathbf{w}_i) \tag{7.4}$$

The probability $P(\mathbf{w}_i)$ is the total probability of all possible tag sequences for $\mathbf{w}_i$:

$$P(\mathbf{w}_i) = \sum_{\mathbf{t}} P(\mathbf{t}, \mathbf{w}_i) \tag{7.5}$$

This summation is over a potentially exponential number of paths, but we can do this using dynamic programming. We saw how to do this when computing Z in a conditional random field; here it is again, specialized for the HMM:

forward[<s>, 0] $\leftarrow 1$
forward[t, 0] $\leftarrow 0$ **for** $t \neq$ <s>
for $j \leftarrow 1, \ldots, n$ **do**
  for each tag t' **do**
    forward[t', j] $\leftarrow 0$
    for each tag t **do**
      forward[t', j] $\leftarrow$ forward[t', j] + forward[t, $j-1$] $\times\ p(t' \mid t)\, p(w_j \mid t')$
    end for
  end for
end for
$Z \leftarrow 0$
for each tag t **do**
  $Z \leftarrow Z +$ forward[t, n] $\times\ p($</s> $\mid t)$
end for

Then $P(\mathbf{w}) = Z$.

7.4.2 Example

Continuing with our example above, let's suppose that our current probabilities are:

$$p(1 \mid \texttt{<s>}) = 0.5 \qquad p(2 \mid \texttt{<s>}) = 0.5$$
$$p(1 \mid 1) = 0.2 \qquad p(1 \mid 2) = 0.5$$
$$p(2 \mid 1) = 0.6 \qquad p(2 \mid 2) = 0.3$$
$$p(\texttt{</s>} \mid 1) = 0.2 \qquad p(\texttt{</s>} \mid 2) = 0.2$$
$$p(\text{a} \mid 1) = 0.1 \qquad p(\text{a} \mid 2) = 0.4$$
$$p(\text{b} \mid 1) = 0.2 \qquad p(\text{b} \mid 2) = 0.3$$
$$p(\text{c} \mid 1) = 0.3 \qquad p(\text{c} \mid 2) = 0.2$$
$$p(\text{o} \mid 1) = 0.4 \qquad p(\text{o} \mid 2) = 0.1$$

And our input string is $\mathbf{w}$ = coba.

	w_j = c	w_j = o	w_j = b	w_j = a
$t = 1$	$0.5 \cdot 0.3 = 0.15$	$0.15 \cdot 0.2 \cdot 0.4 +$ $0.1 \cdot 0.5 \cdot 0.4 = 0.032$	$0.032 \cdot 0.2 \cdot 0.2 +$ $0.012 \cdot 0.5 \cdot 0.2 = 0.00212$	
$t = 2$	$0.5 \cdot 0.2 = 0.1$	$0.15 \cdot 0.6 \cdot 0.1 +$ $0.1 \cdot 0.3 \cdot 0.1 = 0.012$		

Question 9. Fill in the rest of the table, and then compute $P(\mathbf{w})$.

7.4.3 Optimization

To make this an unconstrained optimization problem, we make a change of variables (just like we did with the topic model), expressing each of the tag-tag probabilities $p(t' \mid t)$ in terms of weights $\lambda(t' \mid t)$,

and each of the tag-word probability $p(w \mid t)$ in terms of weights $\mu(w \mid t)$. The λ's and μ's can be any positive or negative real number:

$$p(t' \mid t) = \frac{\exp \lambda(t' \mid t)}{\sum_{t''} \exp \lambda(t'' \mid t)}$$

$$p(w \mid t) = \frac{\exp \mu(w \mid t)}{\sum_{w'} \exp \mu(w' \mid t)}$$

Initialize all the λ's and μ's randomly. Then, use automatic differentiation to compute the gradient of $\log L$ with respect to the λ's and μ's, and use stochastic gradient ascent to update them:

randomly initialize $\lambda(t' \mid t)$ and $\lambda(w \mid t)$ for all t, t', w
repeat
  $LL \leftarrow 0$ ▷ log-likelihood $\log L$
  for training examples $\mathbf{w}_i$ **do**
    compute $\log P(\mathbf{w}_i)$
    $LL \leftarrow LL + \log P(\mathbf{w}_i)$
    compute $\frac{\partial \log P(\mathbf{w}_i)}{\partial \lambda(t' \mid t)}$ and $\frac{\partial \log P(\mathbf{w}_i)}{\partial \mu(w \mid t)}$ at the current point
    $\lambda(t' \mid t) \leftarrow \lambda(t' \mid t) + \eta \cdot \frac{\partial \log P(\mathbf{w}_i)}{\partial \lambda(t' \mid t)}$
    $\mu(w \mid t) \leftarrow \mu(w \mid t) + \eta \cdot \frac{\partial \log P(\mathbf{w}_i)}{\partial \lambda(w \mid t)}$
  end for
  print LL ▷ should increase
until done

7.5 Labeling and Evaluation

After we're done training the model, we go through the training data again and use the Viterbi algorithm to find the best labeling of every training sequence. There's no need here to separate training and test data. (Why?)

Evaluating an unsupervised sequence labeling problem is tricky. If we wanted characters to be labeled as consonants or vowels, all we get back is characters that are labeled as 1 or 2. If we wanted words to be labeled with part-of-speech tags, all we get back is words that are labeled $1, 2, 3, \ldots, T$.

If the tags are to be used in some downstream application, then the best evaluation is to evaluate the downstream application. Otherwise, the simplest evaluation is called *many-to-one accuracy*: for each tag t, look at all the words that were tagged t, and select the most frequent *correct* tag t^* for those words. Replace t with t^*. After processing each tag this way, compute accuracy as usual.