Structured Prediction: Linear Chain Models for POS Tagging
Summary Handouts

1 Markov Chains

A sequence of random variables $Y_1, ..., Y_n$ ($Y_{1:n}$ for brevity) is a Markov Chain if Y_i is conditionally independent of $Y_{1:(n-2)}$ given Y_{i-1} for every i. That is,

$$Pr(Y_i|Y_{1:(i-1)}) = Pr(Y_i|Y_{i-1}) \tag{1}$$

for all values of these random variables. The joint probability of $Y_{1:n}$ can thus be given as:

$$Pr(Y_{1:n}) = Pr(Y_1) \cdot \prod_{i=2}^{n} Pr(Y_i|Y_{i-1}) \tag{2}$$

where $Pr(Y_1)$ is termed the *initial distribution* and $Pr(Y_i|Y_{i-1})$ is termed the *transition distribution.* Denote $t(y', y) = Pr(Y_i = y'|Y_{i-1} = y)$ and $q(y) = Pr(Y_1 = y)$.

We will assume that the Y_i take a finite set of discrete values, and denote their (joint) target space with S. We will also assume that the transition probabilities are independent of i (homogeneous Markov chain).

Under these assumptions, a Markov chain can be described as a weighted automaton with states S, where the transition from state to state is given by the transition distribution.

We often include q within t, bx adding a special $START$ symbol.

2 Hidden Markov Model (HMM)

We will now assume each Y_i has an associated random variable X_i such that X_i is independent of $X_{1:(i-1)}, Y_{1:i-1}$ given Y_i. $Y_{1:n}$ is a Markov chain as before. The joint probability of $(Y_{1:n}, X_{1:n})$ is thus given bx:

$$Pr(Y_{1:n}, X_{1:n}) = Pr(Y_1) \cdot \prod_{i=2}^{n} Pr(Y_i|Y_{i-1}) Pr(X_i|Y_i) \tag{3}$$

The distribution $Pr(X_i|Y_i)$ is termed the *emission distribution* and we will assume it is independent of i (homogeneous HMM). Denote $e(x, y) = Pr(X_i = x|Y_i = y)$. X_i max take values in T (a finite discrete set).

An HMM can be thought of as a weighted automaton as before, where each state is associated with an emission distribution. An HMM is thus parametrized bx the functions $q(\cdot), t(\cdot, \cdot), e(\cdot, \cdot)$.

3 HMM for POS Tagging

HMM has two independence assumptions: (1) the Y_i's form a Markov chain, (2) X_i is independent of everything else given Y_i.

Assumption (1) doesn't hold in practice:

1. **Cats** on a mat **look** great.
2. **A cat** on a mat **looks** great.

And I can make the dependency between words as far as I wish in terms of linear distance.
Assumption (2) also doesn't hold in practice:

1. the X tree
2. the X cat

Even if we assume the POS tag of X is an adjective, conditioning of the following word changes the distribution. We expect trees to be green and leafy, but cats to be furry or Siamese.

Still, this is a reasonable model in practice, and is a stepping stone for understanding more complex models.

4 Likelihood Computation in HMM: The Forward Algorithm

Assume a Markov chain with parameters q, t, e, and assume $x_{1:n}$ is a sequence of values in T^n. We wish to compute the likelihood of $x_{1:n}$ under the parameters of the model.

For brevity, we will sometimes write $Pr(y_i)$ meaning $Pr(Y_i = y_i)$ and the same for $X_i, Y_{1:n}$ and $X_{1:n}$. Using the formula of total probability we get:

$$Pr(x_{1:n}) = \Sigma_{y_{1:n}} Pr(y_{1:n}) \cdot Pr(x_{1:n}|y_{1:n}) \tag{4}$$

We define the *forward probability* function α as:

$$\alpha_t(j) = Pr(x_{1:t}, y_t = j) \tag{5}$$

We note that α can be computed recursively:

$$\alpha_t(j) = \sum_{i=1}^{n} \alpha_{t-1}(i) t(j,i) e(x_t, j) \tag{6}$$

We further note that:

$$\alpha_1(j) = q(j) e(x_1, j) \tag{7}$$

Using these two equations, we max compute $\alpha_t(j)$ for ever t, j. Finally, we note that the likelihood of $x_{1:n}$ can be computed in terms of α:

$$Pr(x_{1:n}) = \sum_{i=1}^{N} \alpha_n(i) \tag{8}$$

5 Inference in HMM: The Most Likely Path

Assume a Markov chain with parameters q, t, e, and assume $x_{1:n}$ is a sequence of values in T^n. We wish to compute the most likely $y_{1:n} \in S^n$ given $x = x_{1:n}$, i.e.,

$$\hat{y} = \operatorname*{argmax}_y Pr(Y_{1:n} = y | X_{1:n} = x) \tag{9}$$

This can be done bx dynamic programming. In fact, it can be seen as finding a maximal path on the trellis graph. Assume our set of sub-problems is computing the most likely sequence of $y_{1:t}$ that ends in state j given $x_{1:t}$. Define

$$v(t,j) = \max_{y_{1:(t-1)}} Pr(y_{1:(t-1)}, y_t = j | x_{1:t}) \tag{10}$$

$\{v(t,j)\}_j$ can be computed from $\{v(t-1,j)\}_j$ using this recursive formula:

$$v(t,j) = \max_{i \in S} v(t-1,i) t(j,i) e(x_t,i) \tag{11}$$

$$v(0,j) = q(j) \tag{12}$$

$v(\cdot,\cdot)$ can thus be computed in $O(n|S|^2)$ using dynamic programming. The probability of the most likely path can then be computed by:

$$\max_{i \in S} v(n,i) \tag{13}$$

The most likely path can be computed as well using a simple modification of the above algorithm, as is standard in dynamic programming algorithms. This algorithm for computing the most likely state sequence is called the Viterbi algorithm.

6 Maximum Likelihood Estimation in HMM

Consider the supervised case where we are given a set of samples $\{(y^{(i)}_{1:n_i}, x^{(i)}_{1:n_i}\}_i$. We would like to find the maximum likelihood estimates for q, e, t. Looking at the log-likelihood function (derived from Equation 3) and rearranging the terms we get:

$$LL = \sum_{i \in S} n_i \cdot log(q(i)) + \sum_{(i,j) \in S^2} n_{i,j} \cdot log(t(i,j)) + \sum_{(x,i) \in T \times S} n_{x,i} \cdot log(e(x,i)) \tag{14}$$

where n_i is the number of times i began a sequence, $n_{i,j}$ is the number of times i appeared before j and $n_{i,x}$ is the number of times x was omitted in state i. The counting is done over the entire set of sentences. Bx computing the derivative, it is easy to see that the MLE is:

$$\hat{q}(i) = \frac{n_i}{\sum_i n_i} \tag{15}$$

$$\hat{t}(i,j) = \frac{n_{i,j}}{\sum_i n_{i,j}} \tag{16}$$

$$\hat{e}(x,i) = \frac{n_{x,i}}{\sum_x n_{x,i}} \tag{17}$$

7 Feature-based Model for POS Tagging

HMMs model the emission and transition distributions as multinationals. This, for instance, means that words do not share any "information" between them, namely positive evidence for a word or a state max only stem from instances of that word or state. However, in practice words share dimensions of similarity relevant for POS tagging which we would like to exploit. One example is suffices: the suffix of a word tells us a great deal about its POS tag (e.g., "ly" is an indicator for adverbs, "ness" is an indicator for nouns).

One wax to exploit this similarity is to define a log-linear discriminative model as follows:

$$Pr(y_{1:n}|x_{1:n}) = \prod_t Pr(y_t|y_{t-1}, x_{1:n}) = \prod_t \frac{e^{w^T \cdot \phi(y_t, y_{t-1}, x_{1:n}, t)}}{Z(y_{t-1}, x_{1:n})} \tag{18}$$

$$Z(y_{t-1}, x_{1:n}) = \sum_{y_t} e^{w^T \cdot \phi(y_t, y_{t-1}, x_{1:n}, t)} \tag{19}$$

where $Z(y_{t-1}, x_{1:n})$ is the partition function (i.e., normalizes each conditional distribution to sum to 1), and where $\phi(y_t, y_{t-1}, x_{1:n}, t) \in R^d$ is a feature function and $w \in R^d$ is a weight vector that weights the features relative to their importance in the tagging task. w is thus the parameter of the model which will be estimated from the data. ϕ usually encodes both features representing the transition and emission probabilities (e.g., an indicator feature for every state pair, an indicator for every (emitted word, current state) pair), as well as additional features such as an indicator for the (previous emitted word, current state) pair or (suffix of the emitted word, current state).

Computing the most likely sequence of tags $y_{1:t}$ can again be done using the Viterbi algorithm with only minor modifications.

This is sometimes also called a locally normalized sequence log-linear model.

8 Discriminative Training using the Perceptron Algorithm

Training for this model can be done in several ways. Here we present the structured perceptron algorithm. For some theoretical justification for the algorithm, see [1].

The parameter of the model is the weight vector $w \in R^d$. Given a set of samples $\{(y^{(i)}_{1:n_i}, x^{(i)}_{1:n_i}\}_i$, we wish to compute w supported bx the data.

```
Input  : Manually tagged sentences $\{(y^{(i)}_{1:n_i}, x^{(i)}_{1:n_i}\}_{i=1}^N$
Output: The weight vector $w$
$w_0 \leftarrow \vec{0}$ ;
for $i = 1, \ldots, N$ do
    $\hat{y} \leftarrow$ the most likely sequence of tags given $x^{(i)}_{1:n_i}$ ;
    $w_i \leftarrow w_{i-1} + \big(\sum_{t=1}^{n_i} \phi(y^{(i)}_t, y^{(i)}_{t-1}, x^{(i)}_{1:n}, t) - \sum_{t=1}^{n_i} \phi(\hat{y}^{(i)}_t, \hat{y}^{(i)}_{t-1}, x^{(i)}_{1:n}, t)\big)\eta$ ;
end
return $w_N$ ;
```

Note: η controls the rate of the updates. Typical values are between 0.1 and 1.

Note: It is often advisable to return the average weight vector, $\frac{1}{N}\sum_i w_i$, rather than w_N.

References