11-763
Fall 2015
Homework 2
Due: 10/15/2015, 11:59 pm

Please submit your homework as a LaTeX-typeset PDF. Hand-drawn diagrams are ok. Direct clarification questions to `cmu-instructors-11763-fa2015@googlegroups.com`

1 Fun with HMMs

Given a finite label set Ω and a finite observation alphabet Σ, a bigram HMM defines a joint distribution over label sequences $\mathbf{y} = \langle y_1, y_2, \ldots, y_n \rangle \in \Omega^n$ and observation sequences $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle \in \Sigma^n$ where $n \geq 0$ as

$$p(\mathbf{x}, \mathbf{y}) = \left(\prod_{i=1}^{n} \gamma(y_i \mid y_{i-1}) \cdot \eta(x_i \mid y_i) \right) \cdot \gamma(\text{STOP} \mid y_n)$$

where y_0 is assumed to be START.

1. Define an algorithm that runs in $O(|\Omega|^2 nk)$ time and generates k independent, random samples over the *label sequence* variable $\mathbf{y}$, drawn according to the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$. (The inputs to the algorithm are the label set Ω, the HMM parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$, and the sequence $\mathbf{x}$). Can you make the runtime even smaller? (**Hint**: Try sampling y_i from the distribution $p(y_i \mid \mathbf{x})$)

2. Suppose I have an HMM such that $\Omega = \{B, I, O\}$. This model gives a joint distribution over word sequences bracketed with base noun phrases; this is also called "NP chunking." Given a structure (y_1^n, x_1^n) generated by this model, a "base noun phrase" corresponds to $x_i^j = \langle x_i, x_{i+1}, \ldots, x_j \rangle$ whenever $(y_i^{j+1} = BI^{j-i}O) \vee (y_i^{j+1} = BI^{j-i}B) \vee ((j = n) \wedge (y_i^j = BI^{j-i}))$ (the first argument of the disjunction corresponds to a noun phrase followed by non-NP material, the second corresponds to a base NP that is followed by another base NP, and the third corresponds to a base NP at the end of the sentence). Define an algorithm that calculates the exact (marginal) probability that subsequence x_i^j is a base noun phrase, given the sequence and the HMM. (The inputs to the algorithm are the HMM parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$, i, j, and the sequence $\mathbf{x}$.)

2 Fun with CRFs

HMMs define joint distributions over sequences of labels and tags. However, in many structured prediction applications, we are only interested in the conditional distribution of labels given observations, $p(\mathbf{y} \mid \mathbf{x})$. Conditional random fields (CRFs) permit this distribution to be modeled directly, using the following parametric form:

$$p(\mathbf{y} \mid \mathbf{x}; \boldsymbol{w}) = \frac{\exp \boldsymbol{w}^\top \sum_{i=1}^{n} \boldsymbol{f}(x_i, y_i, y_{i-1})}{Z(\mathbf{x}; \boldsymbol{w})},$$

$$\text{where } Z(\mathbf{x}; \boldsymbol{w}) = \sum_{\mathbf{y}' \in \Omega^n} \exp \boldsymbol{w}^\top \sum_{i=1}^{n} \boldsymbol{f}(x_i, y'_i, y'_{i-1}),$$

$\boldsymbol{w} \in \mathbb{R}^d$ is parameter vector, and $\boldsymbol{f} : \Sigma \times \Omega \times \Omega \to \mathbb{R}^d$ is a d-dimension feature vector function. Feature functions extract properties of local parts of the sequence that are useful for the prediction problem.

To learn the parameters of a CRF, the conditional log likelihood of a sample of training data $\mathcal{T} = \{\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}\}_{i=1}^n$,

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}) &= \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \log p(\mathbf{y} \mid \mathbf{x}; \boldsymbol{w}) \\ &= \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \left[\boldsymbol{w}^\top \sum_{i=1}^{|\mathbf{x}|} [\boldsymbol{f}(x_i, y_i, y_{i-1})] - \log Z(\mathbf{x}; \boldsymbol{w}) \right] \end{aligned} \tag{1}$$

is maximized by adjusting the parameters $\boldsymbol{w}$. There are a variety of ways to do this, but the most common ones involve computing the derivatives of the log likelihood function with respect to $\boldsymbol{w}$. Differentiating Eq. 1 using the chain rule for derivatives, we obtain

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \left[\sum_{i=1}^{|\mathbf{x}|} \left[\boldsymbol{f}(x_i, y_i, y_{i-1}) - \underbrace{\mathbb{E}_{p(\mathbf{y}'|\mathbf{x};\boldsymbol{w})} \boldsymbol{f}(x_i, y_i', y_{i-1}')}_{\text{Model expectation}} \right] \right].$$

2. Show how to obtain the "Model expectation" term in the above equation by differentiating $\log Z(\mathbf{x}; \boldsymbol{w})$ with respect to $\boldsymbol{w}$. Show your work.

3. Give an algorithm that computes the expected value: $\sum_{i=1}^{|\mathbf{x}|} \left[\mathbb{E}_{p(\mathbf{y}'|\mathbf{x};\boldsymbol{w})} \boldsymbol{f}(x_i, y_i', y_{i-1}') \right]$ for a training instance $(\mathbf{x}, \mathbf{y})$. What is the complexity of calculating the expectation term?

Let us now consider an alternative parameterization. Assume that each symbol $\omega \in \Omega$ is associated with a vector $\mathbf{v}_\omega \in \mathbb{R}^m$, and that the feature vector function applies to a single symbol of $\boldsymbol{f}(x_i) \in \mathbb{R}^d$, and let:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\exp \sum_{i=1}^n \left[\boldsymbol{f}(x_i)^\top \mathbf{W} \mathbf{v}_{y_i} + \mathbf{v}_{y_{i-1}}^\top \mathbf{V} \mathbf{v}_{y_i} \right]}{Z(\mathbf{x})},$$

where $\mathbf{W} \in \mathbb{R}^{d\times m}$ and $\mathbf{V} \in \mathbb{R}^{m\times m}$ are parameter matrices.

4. Given a set of training data give the derivatives of the log-likelihood with respect to $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{v}_\omega$.

3 Context Free Grammars

Following are a few rules[1] from a probabilistic context free grammar (PCFG) for English, along with their probabilities. Nonterminals and terminals are written in capital and small letters, respectively.

- $p(\text{S} \rightarrow \text{NP VP}) = 0.1$, $p(\text{S} \rightarrow \text{VP}) = 0.01$
- $p(\text{NP} \rightarrow \text{N}) = 0.1$
- $p(\text{VP} \rightarrow \text{V}) = 0.01$, $p(\text{VP} \rightarrow \text{VP NP}) = 0.1$
- $p(\text{V} \rightarrow \text{time}) = 0.0001$, $p(\text{V} \rightarrow \text{flies}) = 0.001$
- $p(\text{N} \rightarrow \text{time}) = 0.001$, $p(\text{N} \rightarrow \text{flies}) = 0.0001$

[1] There are additional rules not shown; you do not need them for this problem.

1. Find two complete parsing derivations of the sentence "time flies". Which is more probable?

2. Why can't the basic CYK[2] algorithm be used directly with this grammar?

3. Modify the rules of the grammar such that the CYK algorithm can be used.[3] Then, simulate, on paper, the CYK algorithm to find whether "time flies" can be parsed with this grammar. Show your work[4].

4. The basic CYK algorithm solves the membership problem (i.e., whether a sentence belongs to the language defined with a CFG). There is an easy but inefficient way to modify CYK to also return the k-best parses.[5] Briefly and informally explain this easy modification.

4 Parsing with Integer Linear Programming

A linear program in standard form looks like this:

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \text{ such that } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$$

Let N be the dimensionality of $\mathbf{x}$ and let M be the number of linear constraints (not including the positivity constraints); $\mathbf{A} \in \mathbb{R}^{M \times N}$ and $\mathbf{b} \in \mathbb{R}^M$. Any linear program can be transformed to this form.

An *integer linear program* adds the constraint that each x_i is integer-valued. Often bounds constraints and integer constraints are merged into a constraint that each $x_i \in \{0, 1\}$.

1. Let $\mathbf{w} = \langle w_0 = \$, w_1, w_2, \ldots, w_n \rangle$ be a sentence. ($ is the root or "wall" symbol.) Let $cost(w_i, w_j)$ be the cost associated with making w_i the parent of w_j in a dependency tree and let the cost of a dependency tree be

$$\sum_j cost(w_{\text{parent}(j)}, w_j)$$

   Your job is to define an ILP such that minimizing $\mathbf{x}$ can be converted back into a dependency tree such that:

   - every word in $\langle w_1, \ldots, w_n \rangle$ has exactly one parent in $\{w_0, w_1, \ldots, w_n\}$;
   - w_0 does not have a parent
   - the solution to the ILP will correspond to the lowest-cost dependency tree.

   To answer the question, you must define $\mathbf{x}$ (in terms of the dependency tree), $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$. Hint: you should be able to define a solution in which $N = O(n^2)$ and $M = O(n)$. You don't need to convert to standard form, and it's okay to use equality and inequality constraints as long as they are linear.

2. Do part 1 again, but this time add an overall **projectivity** constraint on the tree:

[2] Some people call it CKY, others call it CYK. They are the same thing.

[3] You can use the algorithm described at `http://www.cs.nyu.edu/courses/fall07/V22.0453-001/cnf.pdf` – but it's worth trying to work out how to do this manually first. The modified grammar should be equivalent to the original one (i.e. the languages defined by both grammars are identical.)

[4] For example, you could show the CYK tables. This is not the only way to simulate CYK, though. So just pick a convenient way for you.

[5] It is probably easier to work it out on your own, but feel free to look at section 4.1 in (**?**) The rest of section 4 in this paper describes less obvious and more efficient alternatives; we are not asking about these, although you should be aware they exist.

- if w_i is the parent of w_j, then $\forall k \in (i, j) \cup (j, i)$, the parent of w_k is also $\in [i, j] \cup [j, i]$. (Another equivalent statement is that for any i, w_i and all its descendants for a contiguous substring.) Hint: this will require at least another $O(n^2)$ linear constraints. Note: do not worry about cyclicity yet, unless you find it helpful to do so.

3. **Bonus question:** The dependency structures recovered above have not constrained to be *acyclic*. In practice we typically want dependency structures to be trees. Can you define constraints that will enforce acyclicity? You may introduce more variables, too, if you need to. You may also use non-linear constraints.