11-763 Spring 2018: Homework 2

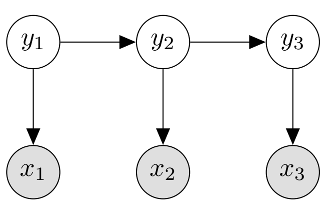
Due: March 8 2018, 23:59 PM

Instructions

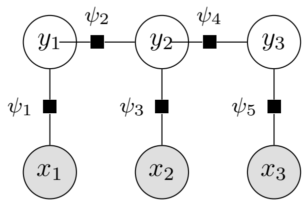
- Please submit the homework as a LaTeX-typeset PDF.
- We would suggest using the tikz-tree and tikz-bayesnet packages to draw trees and graphical models.
- Hand drawn images are also accepted as long as they are clear enough in the pdf.
- We encourage a healthy discussion on homeworks, but the final proof/solution must be written down individually.
- Cite any references that were used to solve any of the questions.

1 Factor Graphs

By converting a graphical model to a factor graph, we can more directly represent its component factors. The distribution under the model can be written as the product of all the factors in the graph. Specicially, a factor graph is a bipartite graph: one set of nodes contains all the random variables, $\mathcal{V}$, while the other set contains all the potential functions (also called factors), Ψ. Each potential function is connected by an undirected edge to each variable it depends on.



(a) Graphical Model



(b) Factor Graph

Figure 1: Hidden Markov Models

For example, Figure 1b shows a factor graph representation of the directed graphical model in Figure 1a. The two sets of nodes in this factor graph are:

$$\mathcal{V} = \{y_1, y_2, y_3, x_1, x_2, x_3\} \tag{1}$$
$$\Psi = \{\psi_1(y_1, x_1), \psi_2(y_1, y_2), \psi_3(y_2, x_2), \psi_4(y_2, y_3), \psi_5(y_3, x_3)\} \tag{2}$$

If $\mathbf{y} = (y_1, y_2, y_3)$ and $\mathbf{x} = (x_1, x_2, x_3)$, the model's distribution is given by:

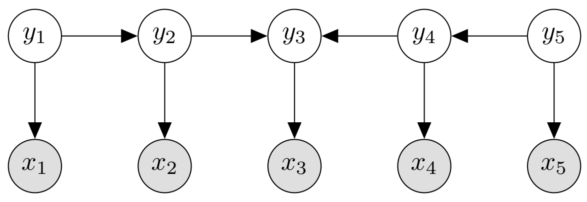
$$p(\mathbf{y}, \mathbf{x}) = \frac{\psi_1(y_1, x_1)\psi_2(y_1, y_2)\psi_3(y_2, x_2)\psi_4(y_2, y_3)\psi_5(y_3, x_3)}{Z} \tag{3}$$

Here, $Z = \sum_{\mathbf{y},\mathbf{x}} \psi_1(y_1, x_1)\psi_2(y_1, y_2)\psi_3(y_2, x_2)\psi_4(y_2, y_3)\psi_5(y_3, x_3)$ is the normalization constant. Because the example is a directed graphical model, each of the factors is a conditional probability (or a product of conditional probabilities) and thus the normalization constant is simply $Z = 1$. For undirected graphical models this may not be the case.

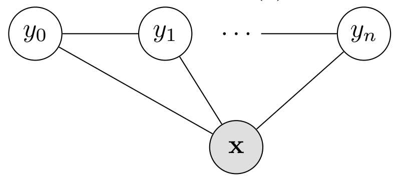
When a factor graph is written in canonical form, each factor corresponds to a **maximal clique** in the *undirected* representation of the model. A clique is a subset of vertices such that every two distinct vertices are adjacent[1]. A maximal clique is a clique that cannot be extended by including one or more adjacent vertices. The corresponding cliques of a *directed* graphical model are the cliques of the minimal undirected graphical model that can represent the original model. For example, in Figure 1a, each maximal clique contains only two variables, for example $\{y_1, y_2\}$, $\{y_1, x_1\}$ and so on.

[1]Adjacent vertices are sometimes called connected vertices

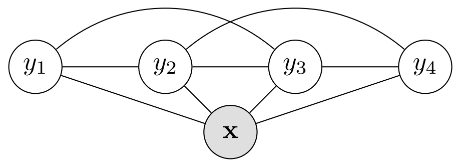
1. Draw canonical factor graphs for each of the graphical models shown in Figure 2. Write their distributions in terms of the factors you introduced (as in Equation 3).



(a) Hidden Markov Model with a twist



(b) Linear Chain CRFs



(c) CRFs with skip connections

Figure 2: Graphical Models

2. Consider the graphical model in Figure 2b and consider Equation 4 in the next section. Give an explicit definition of the potential functions you introduced in your factor graph that would yield Equation 4 as the conditional distribution $p(\mathbf{y}|\mathbf{x})$ under this model.

2 Fun with CRFs

We had a brief look at inference in a HMM model in the last homework and hence we shall move on to Conditional Random Fields (CRFs) in this one. 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, $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\intercal$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^\intercal$. CRFs permit this distribution to be modeled directly, using the following parametric form:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp \mathbf{w}^\intercal \sum_{i=1}^{n} \mathbf{f}(x_i, y_i, y_{i-1})}{Z(\mathbf{x}; \mathbf{w})}, \tag{4}$$

$$\text{where } Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{y}' \in \Omega^n} \exp \mathbf{w}^\intercal \sum_{i=1}^{n} \mathbf{f}(x_i, y'_i, y'_{i-1}) \tag{5}$$

$\mathbf{w} \in \mathbb{R}^d$ is parameter vector, and $\mathbf{f} : \Sigma \times \Omega \times \Omega \to \mathbb{R}^d$ is a d-dimension feature vector function. Feature vectors 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 $\mathcal{T} = \{\mathbf{x}^{(t)}, \mathbf{y}^{(t)}\}_{\forall t}$,

$$\mathcal{L}(\mathbf{w}) = \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \log p(\mathbf{y}|\mathbf{x}; \mathbf{w}) \tag{6}$$

$$= \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \left[\mathbf{w}^\intercal \sum_{i=1}^{n} [\mathbf{f}(x_i, y_i, y_{i-1})] - \log Z(\mathbf{x}; \mathbf{w}) \right], \tag{7}$$

is maximized by adjusting the parameters $\mathbf{w}$. There are a variety of ways to do this but the most common ones involve computing derivatives of the log likelihood function with respect to $\mathbf{w}$. Differentiating Eq. 7 using the chain rule for derivatives, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \left[\sum_{i=1}^{n} \left[\mathbf{f}(x_i, y_i, y_{i-1}) - \underbrace{\mathbb{E}_{p(\mathbf{y}'|\mathbf{x};\mathbf{w})} \mathbf{f}(x_i, y'_i, y'_{i-1})}_{\text{Model Expectation}} \right] \right] \tag{8}$$

1. Show how to obtain the "Model Expectation" term in Eq. 8 by differentiating $\log Z(\mathbf{x};\mathbf{w})$ with respect to $\mathbf{w}$. Show your work.
2. Give an algorithm that computes the expected value: $\sum_{i=1}^{n} \mathbb{E}_{p(\mathbf{y}'|\mathbf{x};\mathbf{w})} \mathbf{f}(x_i, y'_i, y'_{i-1})$ for a training instance $(\mathbf{x}, \mathbf{y})$. What is the complexity of your algorithm? (Assume that computing $\mathbf{f}(x_i, y_i, y_{i-1})$ is $\mathcal{O}(1)$)

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 $\mathbf{f}(x_i) \in \mathbb{R}^d$, and let:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp \sum_{i=1}^{n} \left[\mathbf{f}(x_i)^\intercal \mathbf{W} \mathbf{v}_{y_i} + \mathbf{v}_{y_{i-1}}^\intercal \mathbf{V} \mathbf{v}_{y_i} \right]}{Z(\mathbf{x})}, \tag{9}$$

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

3. Given a set of training data give the derivatives of the log-likelihood with respect to $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{v}_\omega$. You can leave the answer as an Expectation formulation similar to the one in Eq. 8

3 Context Free Grammars

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

- p(S → NP VP) = 0.1, p(S → VP) = 0.01
- p(NP → N) = 0.1
- p(VP → V) = 0.01, p(VP → VP NP) = 0.1
- p(V → time) = 0.0001, p(V → flies) = 0.001
- p(N → time) = 0.001, p(N → flies) = 0.0001

1. Is this PCFG complete? Explain your answer.
2. Find 2 complete parsing derivations of the sentence "time flies". Which is more probable?
3. Why can't the basic CYK algorithm be used directly with this grammar?
4. Modify the rules of the grammar such that the CYK algorithm can be used. Then simulate, on paper, the CYK algorithm to find weather "time flies" can be parsed with this grammar. Show your work[2].
5. 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 k-best parses. Briefly and informally explain this easy modification.

[2]You could use the CYK tables as discussed in class or any other method that you prefer.

4 Parsing with Integer Linear Programming

A linear program in the standard form looks like this:

$$\min_{x} \mathbf{c}^{\mathsf{T}}\mathbf{x} \text{ such that } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0 \tag{10}$$

Let N be the dimensionality of $\mathbf{x}$ and let M be the number of linear constraints (excluding 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 the each x_i is integer-valued. Often bounds constraints and integer constraints are merged into a constraint that each $x_i \in \{0, 1\}$.

With this background, now we are going to model dependency parsing as an ILP problem.

1. Let $\mathbf{w} = (w_0 = \$, w_1, w_2, \ldots, w_n)$ 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 hence the cost of a dependency tree would be:

$$\sum_{j} cost(w_{\text{parent}(j)}, w_j) \tag{11}$$

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

   - every word in $(w_1, w_2, \ldots, w_n)$ has exactly one parent in $(w_0, w_1, w_2, \ldots, w_n)$
   - 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), A, $\mathbf{b}$, and $\mathbf{c}$. You don't need to convert it to a standard form as long as all the inequality and equality constraints are linear. *Hint: You should be able to define a solution in which $N = \mathcal{O}(n^2)$ and $M = \mathcal{O}(n)$.*

2. What all constraints would you need to add to impose overall **projectivity** on the tree.

   - *Projectivity*: 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 form a contiguous substring. *Hint: This will require at least another $\mathcal{O}(n^2)$ linear constraints.*

     Note: do not worry about cyclicity yet, unless you find it helpful to do so.