Organizing Web Information (CS 728) — **Quiz**
Computer Science and Engineering — **2018-03-26 Monday**
Indian Institute of Technology Bombay — **15:30–17:30 SIC301**

NAME $\underset{\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim}{}$ ROLL $\underset{\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim}{}$

This exam has 7 printed page/s. Write your name and roll number on **EVERY SIDE (and not just sheet)**, because we may take apart your answer book and/or xerox it for correction. Write your answer clearly within the spaces provided and on any last blank page. Do not write inside the rectangles to be used for grading. **If you need more space than is provided, you probably made a mistake in interpreting the question.** Start with rough work elsewhere, but you need not attach rough work. Use the marks alongside each question for time management. **Illogical or incoherent answers are worse than wrong answers or even *no* answer, and may fetch negative credit.** You may not use any computing or communication device during the exam. You may use textbooks, class notes written by you, approved material downloaded **prior to the exam** from the course Web page, course news group, or the Internet, or notes made available by me for xeroxing. If you use class notes from other student/s, you must obtain them **prior to the exam** and **write down his/her/their name/s and roll number/s** here.

1. We are given input sequence $\boldsymbol{x} = (x_1, \dots, x_T)$ where each $x_t \in \mathbb{R}^F$, and state/label sequence $\boldsymbol{y} = (y_1, \dots, y_T)$ where each label $y_t \in \{1, \dots, M\}$. To capture rich transition dynamics of output states, we introduce hidden states $\boldsymbol{z} = (z_1, \dots, z_T)$. The number of hidden states will be $N \gg M$. We write the energy, or log-potential, as

$$\mathcal{E}(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x}) = \sum_{t=1}^{T-1} f(x_t, z_t) + g(z_t, y_t) + h(z_t, z_{t+1}).$$

1.a Draw an unrolled plate diagram showing the dependencies between $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$.

		1

1.b Complete: based on the energy expression, it is natural to express

$$\Pr(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{A(\boldsymbol{x})} \sum_{\sim\sim} \exp\big(\mathcal{E}(\sim\sim, \sim\sim | \sim\sim)\big).$$

		2

$$\Pr(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{A(\boldsymbol{x})} \sum_{\boldsymbol{z}} \exp\big(\mathcal{E}(\underset{\sim}{\boldsymbol{y}}, \underset{\sim}{\boldsymbol{z}} | \underset{\sim}{\boldsymbol{x}})\big)$$

1.c Write down the expression for the partition function $A(\boldsymbol{x})$.

		2

$$A(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \sum_{\boldsymbol{z}} \exp\big(\mathcal{E}(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})\big)$$

1.d Show how to compute $\sum_{\boldsymbol{z}} \exp\big(\mathcal{E}(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})\big)$ in time that is polynomial in T, N, M, F. State the time in $O(\cdot)$ notation.

		2

We are given a fixed $\boldsymbol{x}$ and $\boldsymbol{y}$ and must sum over all $\boldsymbol{z}$, which we do one position at a time. For $t = 0, 1, \ldots, T$, let $n \in [1, N]$ and

$$V(t, n) = \sum_{\substack{z_{1:t} \in [1,N]^t \\ z_t = n}} \exp\big(\mathcal{E}(\boldsymbol{y}, z[1:t]|\boldsymbol{x})\big)$$

$V(t, n)$ is filled via dynamic programming:

$$V(0, \star) = 1$$
$$V(t, n) = \sum_{n'=1}^{N} V(t-1, n') \exp\big(f(x_t, n) + g(n, y_t) + h(n', n)\big)$$

The total time needed is $O(TN^2)$ and we read off $\sum_{n=1}^{N} V(T, n)$ in the end.

1.e Show how to compute $A(\boldsymbol{x})$ in time that is polynomial in T, N, M, F. State the time in $O(\cdot)$ notation.

		2

If we do not use any special structure in g, we can simply extend the DP table to

$$V(t, m, n) = \sum_{\substack{y_{1:t} \in [1,M]^t \\ y_t = m}} \sum_{\substack{z_{1:t} \in [1,N]^t \\ z_t = n}} \exp\big(\mathcal{E}(y[1:t], z[1:t]|\boldsymbol{x})\big)$$

which is filled via DP in $O(TM^2N^2)$ time:

$$V(0, \star, \star) = 1$$
$$V(t, m, n) = \sum_{m'=1}^{M} \sum_{n'=1}^{N} V(t-1, m', n') \exp\big(f(x_t, n) + g(n, m) + h(n', n)\big)$$

1.f A specific design would be to allocate $N = MD$ labels for Z_t, and deterministically associate blocks of D states of Z_t to each state of Y_t, using the g log-potential. Assuming this specific form of g, can you improve you previous answer?

		1

For indexing simplicity, let $y_t \in [0, M-1]$ and $z_t \in [0, N-1]$, with states

$mD, \ldots, (m+1)D-1$ of Z mapping to state m of Y. Accordingly, we define

$$g(n,m) = \begin{cases} 0, & n \in \{mD, \ldots, (m+1)D-1\}, \\ -\infty, & \text{otherwise} \end{cases}$$

Having thus defined g, we can simplify the above DP as

$$V(0,m,n) = 1$$
$$V(t,m,n) = \sum_{m'=0}^{M-1} \sum_{n'=m'D}^{m'(D+1)-1} V(t-1,m',n') \exp\big(f(x_t,n) + h(n',n)\big)$$

Initializing the table takes $O(TMN)$ time. After that, to fill each of TMN cells takes $O(MD)$ time, for a total of $O(TM^2ND)$ time.

(Based on Low-Rank Hidden State Embeddings for Viterbi Sequence Labeling.)

2. Suppose we want to embed each synset x in WordNet as a vector $\boldsymbol{u}_x \in \mathbb{R}^D$. Synsets are related via a (transitive) partial order: $x \prec y$ means "x is a hyponym of y", e.g., "a camel is a (hyponym of) mammal". To represent transitivity, if $x \prec y$, Order Embedding (OE) requires that $\boldsymbol{u}_x \geq \boldsymbol{u}_y$, where $\geq$ applies elementwise. Given labeled positive instances $x \prec y$ and negative instances $x \not\prec y$, the overall loss is the sum of two parts:

$$\ell(x,y) = \|\max\{\mathbf{0}, \boldsymbol{u}_y - \boldsymbol{u}_x\}\|_2^2$$
$$\mathcal{L}_+ = \sum_{x \prec y} \ell_+(x,y) = \sum_{x \prec y} \ell(x,y)$$
$$\mathcal{L}_- = \sum_{x \not\prec y} \ell_-(x,y) = \sum_{x \not\prec y} \max\{0, \alpha - \ell(x,y)\},$$

where α is a tuned additive margin. The intuition is that when $x \not\prec y$, we want $\ell(x,y) \geq \alpha$.

2.a Apart from the limitation to be explored in subsequent parts of this question, point out a potential problem with the definition of $\ell(x,y)$. There may be multiple valid answers, but state only one.

		1

The hinge/ReLU is being squared, which increases the influence of outliers and 'hopeless' instances.

2.b Complete: A problem with the above loss definition is that, when $x \not\prec y$, OE may unnecessarily try to make $u_{y,d} > u_{x,d}$ for ________ (some/many/all) $d \in [1,D]$ rather than ________ (all/some/at least one) $d \in [1,D]$.

		2

A problem with the above loss definition is that, when $x \not\prec y$, OE may unnecessarily try to make $u_{y,d} > u_{x,d}$ for many $d \in [1,D]$ rather than at least one $d \in [1,D]$.

2.c The losses we really want to minimize are:

- When $x \prec y$, we want $\ell_+(x,y) = \begin{cases} 0, & \boldsymbol{u}_x \ge \boldsymbol{u}_y \\ 1, & \exists d : u_{x,d} < u_{y,d} \end{cases}$.
- When $x \not\prec y$, we want $\ell_-(x,y) = \begin{cases} 0, & \exists d : u_{x,d} < u_{y,d} \\ 1, & \boldsymbol{u}_x \ge \boldsymbol{u}_y \end{cases}$.

Using $\text{ReLU}(\bullet) = [\bullet]_+ = \max\{0, \bullet\}$ and sigmoid non-linearity $\sigma(\bullet) = 1/(1+e^{-\bullet})$, propose smooth approximations to the above losses.

		3

For $x \not\prec y$, loss should be zero if $u_{x,d} < u_{y,d}$ for any $d \in [1, D]$. Accordingly, we redefine

$$\ell_-(x,y) = \min_{d\in[1,D]} [u_{x,d} - u_{y,d}]_+, \tag{1}$$

so that the loss is zero if dominance fails in at least one dimension. To balance this L_∞ form in case of positive instances, we redefine

$$\ell_+(x,y) = \max_{d\in[1,D]} [u_{y,d} - u_{x,d}]_+, \tag{2}$$

so that the loss is zero only if dominance holds in all dimensions.
The unbounded hinge losses above mean a few outliers can hijack the aggregate losses $\mathcal{L}_+$ and $\mathcal{L}_-$. Moreover, the absence of a geometric margin (as in SVMs, as against the *loss* margin α above) also makes confident separation between $\prec$ and $\not\prec$ cases tricky. Our final design introduces a nonlinearity (sigmoid function) to address outliers, additive margin Δ and a standard stiffness hyperparameter ψ.

$$\ell_+(x,y) = \sigma\left(\psi \max_{d\in[1,D]} [u_{y,d} + \Delta - u_{x,d}]_+\right) - 1/2 \tag{3}$$

$$\ell_-(x,y) = \sigma\left(\psi \min_{d\in[1,D]} [u_{x,d} + \Delta - u_{y,d}]_+\right) - 1/2. \tag{4}$$

(Obviously the '$-1/2$' terms are immaterial for optimization, but bring the loss expression to zero when there are no constraint violations.)

3. FrameNet defines relations with canonical relation type IDs r and their (possibly typed) arguments $a_1, \ldots, a_j, \ldots$ in the form $r(a_0, a_1)$. E.g., Theft(Perpetrator,Goods). Each relation type can have aliases, which may be WordNet synsets like *steal.v.01* or *lift.v.02*, or strings such as *stole* and *lifted*. Sample annotated sentences supporting this frame are "Yesterday, $\underline{\text{John}}_{\textsf{Perpetrator}}$ $\underline{\text{stole}}_{\textsf{Theft}}$ $\underline{\text{my car}}_{\textsf{Goods}}$." and "$\underline{\text{Mary}}_{\textsf{Perpetrator}}$ easily $\underline{\text{lifted}}_{\textsf{Theft}}$ $\underline{\text{a purse}}_{\textsf{Goods}}$ at the mall." We can assume that the number of arguments in each frame is a small constant J. We are given a sentence $x_1, \ldots, x_T$ with spans $S = \{s_i\}$ identified by a suitable NLP module (which might identify more spans than needed). We assume a frame $r(a_1, a_2)$ has been identified. The task of semantic role labeling (SRL) is to mark spans that are mentions of $\{a_j\}$. Each role/argument a_j needs to be satisfied by exactly one span. Given any span s_i and a role a_j, suppose we have available a compatibility scoring function $g(s_i, a_j)$.

3.a For this part, assume no pair of spans in S overlap. Using 0/1 indicator variables

z_{ij}, write down an integer linear program (ILP) to find the optimal SRL. Do you really need an ILP or is there a simpler solution?

		1

The objective to maximize is $\sum_{i,j} z_{ij} g(s_i, a_j)$. Each a_j should be satisfied exactly once: for each j, $\sum_i z_{ij} = 1$. Each s_i should be 'used' at most once: for each i, $\sum_j z_{ij} \le 1$.

3.b Now assume the NLP module can collect overlapping spans in S. However, overlapping distinct spans are not allowed to satisfy the same or different arguments of r. Let $S_t \subseteq S$ be the spans that include token x_t. Suggest modifications to the ILP above to enforce the above constraint in the face of overlapping spans in S.

		1

For each position t, $\sum_{s_i \in S_t} \sum_j z_{i,j} \le 1$.

3.c List two potential shortcomings of the ILP approach, one related to inference time, the other related to training of model parameters inside g.

		1

ILP gives only the MAP estimate and not a normalized probability. We cannot calculate loss gradients wrt model parameters in g through the ILP to update the model. Only something like voted perceptron can be used, rather than more efficient CRF training methods.

3.d We will develop an alternative dynamic programming (DP) approach. Let $a_0 = \varnothing$ be the null label to be assigned to spans that do not fulfill any role. Let $g(s_i, \varnothing)$ be the 'profit' from assigning span s_i to no role. The DP lattice graph has nodes $\{v_t : t = 1, \ldots, T+1\}$. The null edge from node v_t to v_{t+1} will be denoted as $(t, t+1, \varnothing)$, having weight $g((t, t+1), \varnothing)$. If span s_i includes tokens ℓ_i through h_i both inclusive, for each role a_j, we add the edge $(\ell_i, h_i + 1, a_j)$, having weight $g(s_i, a_j)$. (Adding all possible such edges may result in parallel edges.) Which of these statements is/are true? Justify informally.

- Given a role assignment to non-overlapping spans with a certain (ILP) score, we can find a path from node 1 to $T+1$ with total edge weight equal to the ILP score.
- Given a path from node 1 to $T+1$ with a certain total edge weight, we can find a role assignment to non-overlapping spans having ILP score equal to the total edge weight.

		2

Both statements are correct. Given a role assignment to non-overlapping spans, we will choose the corresponding edges in the DP graph. For all tokens not assigned roles, we will use null edges. The other direction is similarly trivial.

3.e The above construction does not guarantee that an optimal path satisfies each role/argument exactly once. Extend the node space $\{v_t\}$ to $\{v_t^k\}$ for a suitable space of k, add any small number of additional nodes you may need, and redesign edges suitably to satisfy this constraint. About how many nodes will there be in your DP lattice at most? How much time will your DP take at most?

		4

k will take values over all subsets of $\{1, \dots, J\}$, the set of arguments to the chosen frame. I.e., we will make 2^J copies of the DP graph defined above, one for each subset of arguments satisfied. Let $k = (k_1, \dots, k_J) \in \{0,1\}^J$. If argument j is satisfied in copy k, then we no longer allow choosing any edge labeled a_j in that copy. If argument j is not satified in copy $(k_1, \dots, k_j = 0, \dots, k_J)$, then what used to be edge $(\ell_i, h_i + 1, a_j)$ within one DP graph now goes from copy $(k_1, \dots, k_j = 0, \dots, k_J)$ to copy $(k_1, \dots, k_j = 1, \dots, k_J)$. We are interested in the best path from node 0 in copy $(0, \dots, 0)$ to node $T + 1$ in copy $(1, \dots, 1)$.

(Based on Efficient Inference and Structured Learning for Semantic Role Labeling.)

4. Consider a version of TransE, where subject, relation, object s, r, o are embedded to vectors $\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{o} \in \mathbb{R}^D$. We have large confidence in the triple (s, r, o) iff $\boldsymbol{s} + \boldsymbol{r} \approx \boldsymbol{o}$, i.e., if $\|\boldsymbol{s} + \boldsymbol{r} - \boldsymbol{o}\|_2^2$ is small.

4.a Simplify $\|\boldsymbol{s} + \boldsymbol{r} - \boldsymbol{o}\|_2^2$ using $\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}$ and showing all steps.

		1

4.b Assuming all entity and relation embeddings are kept scaled to unit L_2 norm, simplify further to:

$$\|\boldsymbol{s} + \boldsymbol{r} - \boldsymbol{o}\|_2^2 \propto \text{constant} + \utilde{\quad\quad} \cdot \utilde{\quad\quad} - \utilde{\quad\quad} \cdot \utilde{\quad\quad} - \utilde{\quad\quad} \cdot \utilde{\quad\quad}.$$

		2

$$\|\boldsymbol{s} + \boldsymbol{r} - \boldsymbol{o}\|_2^2 \propto \text{constant} + \utilde{\boldsymbol{r}} \cdot \utilde{\boldsymbol{s}} - \utilde{\boldsymbol{s}} \cdot \utilde{\boldsymbol{o}} - \utilde{\boldsymbol{r}} \cdot \utilde{\boldsymbol{o}}.$$

4.c Suppose we are training all relation and entity embeddings via gradient descent. For positive sample (s, r, o), if $\boldsymbol{r}, \boldsymbol{o}$ are fixed for the moment, in which direction does $\boldsymbol{s}$ move? Justify why that makes sense.

		1

Gradient descent will try to reduce $\boldsymbol{s} \cdot \boldsymbol{r} - \boldsymbol{s} \cdot \boldsymbol{o}$, by moving $\boldsymbol{s}$ toward $\boldsymbol{o} - \boldsymbol{r}$. Given we want $\boldsymbol{s} \approx \boldsymbol{o} - \boldsymbol{r}$, this makes sense.

4.d For negative sample (s, r, o'), if $\boldsymbol{r}, \boldsymbol{o}'$ are fixed for the moment, in which direction does $\boldsymbol{s}$ move? Justify why that makes sense.

		1

Gradient descent will try to increase $\boldsymbol{s}\cdot\boldsymbol{r}-\boldsymbol{s}\cdot\boldsymbol{o}'$, by moving $\boldsymbol{s}$ toward $\boldsymbol{r}-\boldsymbol{o}'$, or away from $\boldsymbol{o}'-\boldsymbol{r}$, which also makes sense.

Total: 30