Organization of Web Information (CS 728) | **Midterm Exam**
Computer Science and Engineering | **2011-02-26 Saturday**
Indian Institute of Technology Bombay | **17:30–20:00 SIC304**

NAME _____ ROLL _____

This exam has 7 printed page/s. Write your 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. **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 will recast Viterbi decoding with slightly modified notation. Recall that the input sequence is $x_1, \ldots, x_T$, the state sequence is $y_1, \ldots, y_T$, and there are M states. We are given T matrices $A_t \in \mathbb{R}^{M \times M}$ where $A_t(m', m) = \sum_f w_f \phi_f(m', m, x, t)$, with F features $\{\phi_f\}$ as usual. Adjoining a sentinel state y_0, inference seeks $\arg\max_{\vec{y}} \sum_{t=1}^{T} A_t(y_{t-1}, y_t)$. This is equivalent to a shortest path problem (after adjoining another sentinel state y_{T+1}) in the state-by-time *trellis graph*.

1(a) Including the sentinel nodes, how many nodes does the trellis graph have? What is the maximum possible number of edges?

		1

$MT + 2$ nodes, at most $2M + (T-1)M^2$ edges.

1(b) Let (u, v) be an edge in the trellis graph. u is indexed as $(t-1, m')$ and v as t, m. Let the edge cost c_{uv} be $-A_t(m', m)$. Let z_{uv} be a 0/1 decision variable which indicates whether the edge is part of the shortest path. What is the (unnormalized) inference objective, expressed in terms of c and z?

		1

$\sum_{u,v} c_{uv} z_{uv}$

1(c) To constrain the z_{uv}s to form a single path, we need separate sets of constraints at y_0, y_{T+1}, and all other nodes. Write down these three families of constraints. Use $O(u)$ as out-neighbors of node u and $I(v)$ and in-neighbors of node v. Try to make your constraints minimal, i.e., ensure (and justify) that no constraint is subsumed by others.

		3

$$\sum_{v \in O(y_0)} z_{y_0 v} = 1$$

$$\sum_{u \in I(y_{T+1})} z_{u y_{T+1}} = 1$$

$$\forall v \in V \setminus \{y_0, y_{T+1}\} : \quad \sum_{u \in I(v)} z_{uv} = \sum_{w \in O(v)} z_{vw}$$

1(d) The motivation to write what was a simple dynamic program as an integer program was that we can add all kinds of constraints that cannot be handled within the Viterbi framework. Assuming the B-I-O state convention, suppose we want at most K occurrences of segments of a specific label. Let the corresponding state IDs be m_B and m_I. Write constraint/s for this requirement. Note that decision variables z_{uv} can be written with the full set of indices as $z_{t-1,m';t,m}$; use this form.

		1

That zero or more m_Is can follow an m_B will be handled by transition rules already, so we need to just count up m_Bs.

$$\sum_{t=1}^{T} \sum_{m'=1}^{M} z_{t-1,m';t,m_B} \leq K$$

1(e) How would you encode the constraint "if label m appears, then label m' cannot appear"?

		1

In general if we have Boolean variables z_1, z_2 and we want to say $z_1 \implies \neg z_2$, that's equivalent to $\neg z_1 \vee \neg z_2$, or in other words, $(1 - z_1) + (1 - z_2) \geq 1$, or $z_1 + z_2 \leq 1$. Translating to our setting, we require that

$$\sum_{t=1}^{T} \sum_{\mu=1}^{M} z_{t-1,\mu;t,m} + \sum_{t=1}^{T} \sum_{\mu=1}^{M} z_{t-1,\mu;t,m'} \leq 1.$$

1(f) How would you encode the requirement that token positions ℓ through r (inclusive) must all have the same state/label?

		2

This constraint can be paraphrased as "if any position $t \in [\ell, r]$ is labeled m, then all positions $t' \in [\ell, r]$ must be labeled m. For shorthand, define

$$g(t, m) = \sum_{m'=1}^{M} z_{t-1,m';t,m}$$

The condition we want is, for $t, t' \in [\ell, r]$,

$$g(t, m) = 1 \implies g(t', m) = 1$$

In general, $g_1 \implies g_2$ is equivalent to $\neg g_1 \vee g_2$ which can be enforced as $1 - g_1 + g_2 \geq 1$ or $g_2 \geq g_1$. Therefore the constraints are:

$$\forall t, t' \in [\ell, r]: \quad g(t', m) \geq g(t, m).$$

2. In latent Dirichlet allocation (LDA), we start with T topics, W words, and word distributions $\phi^{(t)} \sim \text{Dir}(\vec{\beta})$ for topic t. Then for each document $d \in [1, D]$, we sample a topic distribution $\theta^{(d)} \sim \text{Dir}(\vec{\alpha})$. For each word offset o in document d, we sample a topic $z_{do} \sim \text{Multi}(\theta^{(d)})$, and then a word $w_{do} \sim \text{Multi}(\phi^{(z_{do})})$.

Given a corpus and fixed α, β, the LDA inference problem is to estimate (posterior distributions over) $\phi^{(t)}$ for $t \in [1, T]$, $\theta^{(d)}$ for $d \in [1, D]$, and often z_{do} as well. w_{do} is observable and fixed. This

exercise is about applying LDA for entity disambiguation, where we interpret topics as entity labels.

We will consider here a simple Gibb's sampling approach to get an estimate of $\Pr(z|x)$, and then use this to get estimates for $\phi^{(t)}$ and $\theta^{(d)}$. For simplicity we will assume flat hyperparameters $\vec{\alpha} = (\alpha_1, \ldots, \alpha_T) = (\alpha, \ldots, \alpha)$ and $\vec{\beta} = (\beta_1, \ldots, \beta_W) = (\beta, \ldots, \beta)$ for single numbers α, β. We will drop the vector notation from $\vec{\alpha}, \vec{\beta}$ henceforth.

Let $n_t^{(d)}$ be the number of times a word from document d has been assigned to topic t, and $n_\cdot^{(d)}$ be $\sum_t n_t^{(d)}$. Let $n_t^{(w)}$ be the number of times word w has been assigned to topic t in the vector of assignments $\vec{z}$, and $n_t^{(\cdot)}$ be $\sum_w n_t^{(w)}$.

For your convenience here are some definitions related to the Dirichlet distribution. When we say $\phi \sim \text{Dir}(\beta)$, ϕ takes on values that are parameters of a multinomial distribution, i.e., $\sum_w \phi_w = 1$. For such values,

$$\Pr(\phi|\beta) = \frac{1}{B(\beta)} \prod_w \phi_w^{\beta_w - 1},$$

where the normalizing constant is

$$B(\beta) = \frac{\prod_w \Gamma(\beta_w)}{\Gamma\left(\sum_w \beta_w\right)}.$$

Here $\Gamma(\cdot)$ is the gamma function: $\Gamma(u) = \int_0^\infty t^{u-1} e^{-t} dt$.

2(a) In this application of LDA, interpret in up to four sentences the parameters $\phi_w^{(t)}$ using a concrete entity disambiguation example.

		1

Each entity t has an associated distribution over words w. E.g., if the entity t is Michael Jordan the statistics professors, words w like 'variational', 'graphical' and 'Bayes' have large probabilities (ϕ values). Whereas, for Michael Jordan the basketball player, other words like 'match', 'champion' and 'tournament' will have large probabilities.

2(b) Write down $\Pr(\vec{z})$ by completing the following. Show all important intermediate steps.

$$\Pr(\vec{z}) = \left(\frac{\Gamma(T\_\_\_\_\_)}{\_\_\_\_\_^T}\right)^D \prod_{d=1}^D \frac{\prod_{t=1}^T \Gamma(\_\_\_\_\_ + \alpha)}{\Gamma(n_\cdot^{(d)} + \_\_\_\_\_)}$$

		2

$$\Pr(\vec{z}) = \left(\frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T}\right)^D \prod_{d=1}^D \frac{\prod_{t=1}^T \Gamma(n_t^{(d)} + \alpha)}{\Gamma(n_\cdot^{(d)} + T\alpha)}$$

2(c) Write down $\Pr(\vec{w}|\vec{z})$ by completing the following. Show all important intermediate steps.

$$\Pr(\vec{w}|\vec{z}) = \left(\frac{\Gamma(W\_\_\_\_\_)}{\_\_\_\_\_^W}\right)^T \prod_{t=1}^T \frac{\prod_{w=1}^W \Gamma(\_\_\_\_\_ + \beta)}{\Gamma(n_t^{(\cdot)} + \_\_\_\_\_)}$$

		2

$$\Pr(\vec{w}|\vec{z}) = \left(\frac{\Gamma(W\beta)}{\Gamma(\beta)^W}\right)^T \prod_{t=1}^{T} \frac{\prod_{w=1}^{W} \Gamma(n_t^{(w)} + \beta)}{\Gamma(n_t^{(\cdot)} + W\beta)}$$

2(d) Finding $\Pr(\vec{z}|\vec{w})$ is still intractable. Why?

		1

If we tried to use Bayes rule on the quantities computed above, we would get

$$\Pr(\vec{z}|\vec{w}) = \frac{\Pr(\vec{z},\vec{w})}{\Pr(\vec{w})} = \frac{\Pr(\vec{z},\vec{w})}{\sum_{\vec{z}} \Pr(\vec{z},\vec{w})} = \frac{\Pr(\vec{z})\Pr(\vec{w}|\vec{z})}{\sum_{\vec{\zeta}} \Pr(\vec{\zeta})\Pr(\vec{w}|\vec{\zeta})}$$

The sum in the denominator would have an exponential number of terms.

2(e) We will use Gibbs sampling to get around the problem, by sampling from the $\Pr(\vec{z}|\vec{w})$ distribution. To do this, all but one element z_{do} in $\vec{z}$, called $\vec{z}[\backslash do]$, are held fixed, and the marginal distribution of z_{do} is estimated. More notation: let $n_{\ldots}^{(\ldots)}[\backslash do]$ represent counts as before with the counts associated with position indexed by d, o left out. Using your earlier answers, complete the following, showing all important steps:

$$\Pr(z_{do} = t|\vec{z}[\backslash do], \vec{w}) \propto \frac{n_t^{(w_{do})}[\backslash do] + \underline{\quad}}{\underline{\quad} + W\beta} \quad \frac{\underline{\quad} + \alpha}{n_{\cdot}^{(d)}[\backslash do] + \underline{\quad}}$$

		4

$$\Pr(z_{do} = t|\vec{z}[\backslash do], \vec{w}) \propto \frac{n_t^{(w_{do})}[\backslash do] + \beta}{n_t^{(\cdot)}[\backslash do] + W\beta} \quad \frac{n_t^{(d)}[\backslash do] + \alpha}{n_{\cdot}^{(d)}[\backslash do] + T\alpha}$$

2(f) The procedure is to initialize z_{do} in some fashion, and repeatedly evaluate $\Pr(z_{do} = t|\vec{z}[\backslash do], \vec{w})$ and sample z_{do} from this distribution. At the end, we will get (hopefully) stabilized estimates $n_t^{(w)}$ and $n_t^{(d)}$. Now complete the estimates

$$\hat{\phi}_t^{(w)} = \frac{(\underline{\quad}) \quad n_t \quad + \underline{\quad}}{\underline{\quad} + W\underline{\quad}}$$

$$\hat{\theta}_t^{(d)} = \frac{(\underline{\quad}) \quad n_t \quad + \underline{\quad}}{\underline{\quad} + T\underline{\quad}}$$

Show all important steps.

		4

$$\hat{\phi}_t^{(w)} = \frac{n_t^{(w)} + \beta}{n_t^{(\cdot)} + W\beta}$$

$$\hat{\theta}_t^{(d)} = \frac{n_t^{(d)} + \alpha}{n_{\cdot}^{(d)} + T\alpha}$$

2(g) Which estimated parameters help you finally determine the set of entities mentioned in a document?

		1

$\theta_t^{(d)}$ tells us to what extent entity (topic) t is involved in document d. Note that this may not be sparse, so presumably, for applications, we need to do some threholding.

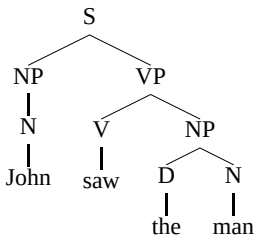
2(h) Point out one desirable feature of entity disambiguation formulations that the above does not have. (Hint: it has to do with the use of Dir(α).)

		1

Using a Dirichlet hypergenerator ensures that each document has few entities and that entity mentions are bursty. But it cannot ensure that there is semantic relatedness among entities. For that, we need to introduce correlations among topics. This can be done using something like CTM, or some kind of category hierarchy as in Wikipedia.

3. Parsing a sentence is fundamental to NLP. Variants are useful in information extraction as well: it is easy to write down a context-free grammar for paper citations, addresses, or even a visual presentation domain such as DOM trees rendered into Web pages with layout clues. Here we will explore kernel-based discriminative training of parsers using such representations.

a)



b)

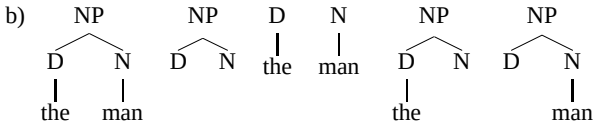


Diagram (a) above shows a parse tree. Diagram (b) enumerates some subtree fragments under the right NP node. In general suppose the corpus has D possible tree fragments, where D is huge. Let a parse tree be denoted x, and $h_d(x)$ be the number of occurrences of the dth tree fragment in x. The overall feature vector representation for x will be $h(x) = (h_1(x), \ldots, h_D(x))$. A tree fragment does not include the specific identity of the leaf words, but only the part-of-speech labels in the internal nodes. The internal nodes immediately connected to the surface words at the leaves are called *pre-terminal* nodes.

3(a) Note that for most x, most $h_d(x) = 0$. We will never write down $h(x)$ because D is huge and $h(x)$ is very sparse (but still quite large). Instead, we will define a kernel $K(x_1, x_2) = h(x_1) \cdot h(x_2)$. Let $N(x)$ be the set of nodes in tree x. Let indicator $I_d(n)$ be 1 iff subtree d is rooted at node n. Then (complete)

		1

$$h_d(x) = \sum_{n \in N(x)} \underbrace{\qquad\qquad}$$

$h_d(x) = \sum_{n \in N(x)} I_d(n)$

3(b) Let $\Delta(n_1, n_2) = \sum_d I_d(n_1) I_d(n_2)$. Complete the following:

		2

$$h(x_1) \cdot h(x_2) = \sum_d h_d(x_1) h_d(x_2)$$
$$= \sum_d \left(\sum_{n_1 \in N(x_1)} \underbrace{\text{(from above)}} \right) \left(\sum_{n_2 \in N(x_2)} \underbrace{\text{(from above)}} \right)$$
$$= \underbrace{\sum}_{\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim} \underbrace{\sum}_{\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim} \Delta(\underbrace{\quad}_{\sim\sim\sim\sim}, \underbrace{\quad}_{\sim\sim\sim\sim}).$$

$\sum_{n_1 \in N(x_1)} \sum_{n_2 \in N(x_2)} \Delta(n_1, n_2)$. 1/2 mark for each blank.

3(c) Next we need to design an efficient algorithm to compute $\Delta(n_1, n_2)$. To this end, complete the following:

- If the labels at n_1 and n_2 are different, then $\Delta(n_1, n_2) =$ $\underbrace{\quad\quad\quad\quad}$.
- If the labels at n_1 and n_2 are the same and they are pre-terminals, then $\Delta(n_1, n_2) =$ $\underbrace{\quad\quad\quad\quad}$.
- Otherwise if the labels at n_1 and n_2 are the same and they are not pre-terminals, then

$$\Delta(n_1, n_2) = \bigoplus_{c=1}^{\text{NumChildren}(n_1)} \left(1 + \underbrace{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}\right),$$

  where NumChildren(n) is the number of children of node n, and Child(n, c) is the cth child of node n. (Replace $\bigoplus$ above with $\sum$ or $\prod$.)

		3

- If the labels at n_1 and n_2 are different, then $\Delta(n_1, n_2) = \underbrace{0}$.
- If the labels at n_1 and n_2 are the same and they are pre-terminals, then $\Delta(n_1, n_2) = \underbrace{1}$.
- Otherwise if the labels at n_1 and n_2 are the same and they are not pre-terminals, then

$$\Delta(n_1, n_2) = \prod_{c=1}^{\text{NumChildren}(n_1)} \left(1 + \underbrace{\Delta(\text{Child}(n_1, c), \text{Child}(n_2, c))}\right),$$

3(d) How much time does it take to compute Δ at the root of trees x_1, x_2?

		1

$O\big(|N(x_1)|\ |N(x_2)|\big)$ time.

3(e) Now we turn to training. We will assume a model w and a scoring function $F(x) = w \cdot h(x)$ which we seek to maximize while picking one of a set of candidate parse trees $X(s)$ for a sentence s. We will write $X(s) = \{x_{s1}, x_{s2}, \ldots\}$. During training, we will assume x_{s1} is the correct parse tree. The (primal) perceptron algorithm proceeds as follows:

initialize $w = \vec{0}$
for sentences s **do**
  let $\bar{j} = \arg\max_j F(x_{sj})$
  if $\bar{j} \neq 1$ **then**
    set $w \leftarrow w + h(x_{s1}) - h(x_{s\bar{j}})$

Because we do not want to deal with $h(x)$ explicitly, we will rewrite the above as a "dual" algorithm using variables α_{sj} in place of w.

initialize $\alpha_{sj} = 0$ for all s, j
for sentences s **do**
  let $\bar{j} = \arg\max_j G(x_{sj})$
  if $\bar{j} \neq 1$ **then**
    set $\alpha_{sj} \leftarrow \alpha_{sj} +$ _____

Fill in the blank above and define the objective $G(x)$ using α_{sj} and various $K(x, x_{sj})$ values only, without using $h(x)$ not involved in a kernel computation, so that the primal and dual forms of the algorithm are completely equivalent and $G(x) = F(x)$.

		3

The blank should be filled with 1.

$$G(x) = \sum_{s,j} \alpha_{sj} \Big(K(x_{s1}, x) - K(x_{sj}, x) \Big)$$

Total: 35