6.864, Fall 2005: Problem Set 3

Total points: 110 regular points

Due date: 5 pm, 1st November 2005

Late policy: 5 points off for every day late, 0 points if handed in after 5pm on November 4th 2005

Question 1 (15 points)

Clarissa Linguistica decides to build a log-linear model for language modeling. She has a training sample (x_i, y_i) for $i = 1 \ldots n$, where each x_i is a prefix of a document (e.g., $x_i =$ "Yesterday, George Bush said") and y_i is the next word seen after this prefix (e.g., $y_i =$ "that"). As usual in log-linear models, she defines a function $\bar{\phi}(x, y)$ that maps any x, y pair to a vector in $\mathbb{R}^d$. Given parameter values $\bar{\theta} \in \mathbb{R}^d$, the model defines

$$P(y|x, \bar{\theta}) = \frac{e^{\bar{\theta} \cdot \bar{\phi}(x,y)}}{\sum_{y' \in \mathcal{V}} e^{\bar{\theta} \cdot \bar{\phi}(x,y')}}$$

where $\mathcal{V}$ is the *vocabulary*, i.e., the set of possible words; and $\bar{\theta} \cdot \bar{\phi}(x, y)$ is the inner product between the vectors $\bar{\theta}$ and $\bar{\phi}(x, y)$.

Given the training set, the training procedure returns parameters $\bar{\theta}^* = \arg\max_{\bar{\theta}} L(\bar{\theta})$, where

$$L(\bar{\theta}) = \sum_i \log P(y_i|x_i, \bar{\theta}) - C \sum_k \theta_k^2$$

and $C > 0$ is some constant.

Clarissa makes the (rather odd) choice of her first two features in the model:

$$\phi_1(x, y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and previous word in } x \text{ is } \texttt{the} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_2(x, y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and previous word in } x \text{ is } \texttt{the} \\ 0 & \text{otherwise} \end{cases}$$

So $\phi_1(x, y)$ and $\phi_2(x, y)$ are *identical features*.

Question (15 points): Show that for any training set, with ϕ_1 and ϕ_2 defined as above, the optimal parameters $\bar{\theta}^*$ satisfy the property that $\theta_1^* = \theta_2^*$.

Question 2 (15 points)

Nathan L. Pedant now decides to build a bigram language model using log-linear models. He gathers a training sample (x_i, y_i) for $i = 1 \ldots n$. Given a vocabulary of words $\mathcal{V}$, each x_i and each y_i is a member of $\mathcal{V}$. Each (x_i, y_i) pair is a *bigram* extracted from the corpus, where the word y_i is seen following x_i in the corpus.

Nathan's model is similar to Clarissa's, except he chooses the optimal parameters $\bar{\theta}^*$ to be $\arg\max L(\bar{\theta})$ where

$$L(\bar{\theta}) = \sum_i \log P(y_i|x_i, \bar{\theta})$$

The features in his model are of the following form:

$$\phi_i(x, y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and } x = \texttt{the} \\ 0 & \text{otherwise} \end{cases}$$

i.e., the features track pairs of words. To be more specific, he creates one feature of the form

$$\phi_i(x, y) = \begin{cases} 1 & \text{if } y = w_2 \text{ and } x = w_1 \\ 0 & \text{otherwise} \end{cases}$$

for every (w_1, w_2) in $\mathcal{V} \times \mathcal{V}$.

Question (15 points): Assume that the training corpus contains all possible bigrams: i.e., for all $w_1, w_2 \in \mathcal{V}$ there is some i such that $x_i = w_1$ and $y_i = w_2$. The optimal parameter estimates $\bar{\theta}^*$ define a probability $P(y = w_2 | x = w_1, \bar{\theta}^*)$ for any bigram w_1, w_2. Show that for any w_1, w_2 pair, we have

$$P(y = w_2 | x = w_1, \bar{\theta}^*) = \frac{Count(w_1, w_2)}{Count(w_1)}$$

where $Count(w_1, w_2) =$ number of times $(x_i, y_i) = (w_1, w_2)$, and $Count(w_1) =$ number of times $x_i = w_1$.

Question 3 (15 points)

Clarissa now decides to build a bigram language model that is fancier than Nathan's. She again has a training sample (x_i, y_i) for $i = 1 \ldots n$ where each (x_i, y_i) pair is a bigram. She introduces an additional "hidden" variable h which can take any one of the values $1 \ldots k$. The log-linear model has the form

$$P(y, h | x, \bar{\theta}) = \frac{e^{\bar{\theta} \cdot \bar{\phi}(x,h,y)}}{\sum_{y' \in \mathcal{V}, h' \in 1 \ldots k} e^{\bar{\theta} \cdot \bar{\phi}(x,h',y')}} \tag{1}$$

where $\bar{\phi}(x, h, y)$ is a feature-vector.

Clarissa defines a new likelihood function,

$$L(\bar{\theta}) = \sum_i \log P(y_i | x_i, \bar{\theta}) = \sum_i \log \sum_{h=1}^{k} P(y_i, h | x_i, \bar{\theta}) \tag{2}$$

where $P(y_i, h | x_i, \bar{\theta})$ takes the form in Eq. 1.

Question (15 points): Recall that we showed in lecture that the gradient of $L(\bar{\theta})$ for regular log-linear models is

$$\left. \frac{\partial L}{\partial \bar{\theta}} \right|_{\bar{\theta}} = \sum_i \bar{\phi}(x_i, y_i) - \sum_i \sum_{y'} P(y' | x_i, \bar{\theta}) \bar{\phi}(x_i, y') \tag{3}$$

Derive a similar expression for

$$\left. \frac{\partial L}{\partial \bar{\theta}} \right|_{\bar{\theta}}$$

where $L(\bar{\theta})$ is as defined in Eq. 2. You should write the gradient in terms of $\bar{\phi}$ and $P(y, h | x, \bar{\theta})$ (in a similar way that Eq. 3 is written in terms of $\bar{\phi}$ and $P(y | x, \bar{\theta})$).

Question 4 (15 points)

In class, we introduced several measures of similarity between probability distributions. The table below summarizes three of these measures: KL divergence, information radius (IRad) and L_1 norm. In the following questions, you will analyze properties of these similarity measures.

Similarity Measure	**Definition**
KL divergence	$D(p\|\|q) = \sum_i p_i \log \frac{p_i}{q_i}$
IRad	$D(p\|\|\frac{p+q}{2}) + D(q\|\|\frac{p+q}{2})$
L_1 norm	$\sum_i \|p_i - q_i\|$

1. Show that IRad is bounded by $2 \log 2$
2. Show that the KL divergence is not symmetric by finding an example of two distributions p and q for which $D(p||q) \neq D(q||p)$
3. Describe the performance of these measures in the presence of low counts.

Question 6 (50 points)

(50 points)

In this question, you will explore corpus-based approaches to lexical semantics. More concretely, you will implement and evaluate a method for clustering verbs based on their distributional properties. In this experiment, the context of a verb is represented by its object.

To train your method, you are provided with a file `verb-object.gz`, which contains a list of (verb, object) pairs extracted from the Wall Street Journal corpus. For example, the first line of `verb-object.gz` is "`Take Stage 5`". This means that the verb "take" was observed with object "Stage" five times.

- You will first have to construct a word-by-word matrix P that captures the distribution of verbs over their objects. The dimensionality of P is $|V| \times |N|$, where $|V|$ is the number of verbs in the corpus and $|N|$ is the number of nouns. The entry $P_{i,j}$ gives the conditional probability that verb v_i has object n_j (i.e., $P(n_j|v_i)$). Note that each row $\vec{p_i}$ of the matrix P defines the conditional distribution $P(.|v_i)$.
- Define a similarity measure $sim(v_i, v_j)$ that gives the similarity between verbs v_i and v_j. Specifically, $sim(v_i, v_j)$ should give the cosine similarity between the distribution vectors $\vec{p_i}$ and $\vec{p_j}$ (i.e. the cosine of the angle between the vectors).

  In order for us to test this part of the algorithm, your code should provide the following interface:

  - Read in a file containing pairs of verbs "$verb_1$ $verb_2$", one pair per line.
  - Print $sim(verb_1, verb_2)$ to standard output, one value per line.

  We will provide development data for this part in the form of two files: `sim.in`, which contains verb pairs as described above, and `sim.out`, which contains the corresponding similarities.
- You should cluster the verbs using the *complete link algorithm*. Your code should provide a clustering function $cluster(k)$ where k specifies the desired number of clusters.

While the complete link algorithm can be implemented in $O(|V|^2 \log |V|)$ steps, your implementation does not have to be that efficient. However, you may want to store some intermediate similarity computations to speed up your program.

NB: It may be possible for two similarity measures to be equal for different pairs of words, in which case a tie-break is necessary. You should break ties by using string comparison, as follows:

- Define the string comparison $str_1 \triangleright str_2$ to be true iff at least one of the following is true
  - $str_1[1]$ (the ASCII value for the first character) is strictly greater than $str_2[1]$. If str is the empty string, substitute -1 for $str[1]$.
  - $str_1[1] = str_2[1]$ and $str_1[2, n_1] \triangleright str_2[2, n_2]$, where $str[2, n]$ is the substring of str that excludes the first character.
- If $sim(verb_1, verb_2) = sim(verb_3, verb_4)$
  - Assume, without loss of generality, that $verb_1 \triangleright verb_2$ and $verb_3 \triangleright verb_4$.
  - Merge the clusters containing $verb_1$ and $verb_2$ if $verb_1 verb_2 \triangleright verb_3 verb_4$, where $str_1 str_2$ indicates concatenation of str_1 and str_2.
  - Merge the clusters containing $verb_3$ and $verb_4$ otherwise.

If your language has a built-in string comparison operator, it will most likely be defined as $\triangleright$ above. For example, $str_1 \triangleright str_2$ is the same as `str1.compareTo(str2) > 0` in Java, `$str1 cmp $str2 > 0` in Perl, and `str1 > str2` in Python.

We will provide development data for this part in the form of two files: `cluster1` and `cluster2`, which represent the output of $cluster(2)$. Each file will contain the list of verbs in the corresponding cluster, one verb per line.

- You will evaluate your clustering approach using the pseudoword disambiguation task. Our corpus contains twenty synthetic pairs of synonyms (see file synrev). Each pair was created by substituting half of the occurrences of the given verb in the corpus with its reverse (e.g., "kill" $\rightarrow$ "llik"). You should define a function $comp(word_1, word_2)$ that returns the number of cluster merging steps required to place $word_1$ and $word_2$ in the same cluster.

  In order for us to test this part of the assignment, please create a file `pseudoword-comp` that contains 20 lines of the form "$word\ reversedword\ comp(word, reversedword)$" for example, "`kill llik 123`"