Midterm for 6.864
Name:

\[\begin{array}{cccc}
40 & 30 & 30 & 30 \\
\end{array}\]

Good luck!
Part #1

Question 1 (10 points)

We define a PCFG where non-terminal symbols are \( \{S, A, B\} \), the terminal symbols are \( \{a, b\} \), and the start non-terminal (the non-terminal always at the root of the tree) is \( S \). The PCFG has the following rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow SS )</td>
<td>0.3</td>
</tr>
<tr>
<td>( S \rightarrow AA )</td>
<td>0.2</td>
</tr>
<tr>
<td>( S \rightarrow BA )</td>
<td>0.5</td>
</tr>
<tr>
<td>( A \rightarrow a )</td>
<td>0.3</td>
</tr>
<tr>
<td>( A \rightarrow b )</td>
<td>0.7</td>
</tr>
<tr>
<td>( B \rightarrow a )</td>
<td>0.4</td>
</tr>
<tr>
<td>( B \rightarrow b )</td>
<td>0.6</td>
</tr>
</tbody>
</table>

For the input string \( aabb \), show two possible parse trees under this PCFG, and show how to calculate their probability.
**Question 2** (10 points)

Consider a model merging algorithm for HMM induction. Given two strings *baab* and *aaab*, the initial model will have the following structure:

![Diagram of the initial model](image)

Note that we use the notation $e(b)$ or $e(a)$ to refer to emission probabilities; for example, for state 8 we have $e(b) = 1$, meaning that the symbol $b$ is emitted with probability 1 from state 8 (Note that by implication, $e(a) = 0$ for state 8).

Show the structure of the model after merging two states. You should choose the two states to be merged in a way that the new model preserves the likelihood of the corpus (i.e., $P(baab) = 0.5$ and $P(aaab) = 0.5$).
Question 3 (20 points)

Consider an agglomerative single-link clustering with a Euclidean similarity measure. As shown in class, the complexity of this algorithm in the general case is $O(n^2)$. We wish to apply this clustering method to a set of points in a one dimensional space (i.e., points that on a line), for example the following:

\[ \text{\begin{center} \includegraphics[width=0.5\textwidth]{line_cluster.png} \end{center}} \]

Design an algorithm that identifies the highest two clusters in the agglomerative clustering. Your algorithm should be more efficient than $O(n^2)$. Specify the complexity of your algorithm. (Hint: you may want to consider the connection between single-link clustering and the minimum spanning tree over the $n$ points.)
Part #2 30 points

Nathan L. Pedant generates \((x, y)\) pairs as follows. Take \(\mathcal{V}\) to be set of possible words (vocabulary), e.g., \(\mathcal{V} = \{\text{the, cat, dog, happy, ...}\}\). Take \(\mathcal{V}'\) to be the set of all words in \(\mathcal{V}\), plus the reversed string of each word, e.g., \(\mathcal{V}' = \{\text{the, eht, cat, tac, dog, god, happy, yppah, ...}\}\).

First, Nathan chooses the symbol \(x\) to be some word from the vocabulary \(\mathcal{V}\). He then does the following:

- With 0.4 probability, he chooses \(y\) to be identical to \(x\).
- With 0.3 probability, he chooses \(y\) to be the reversed string of \(x\).
- With 0.3 probability, he chooses \(y\) to be some string that is neither \(x\) nor the reverse of \(x\). In this case he chooses \(y\) from the uniform distribution over words in \(\mathcal{V}'\) that are neither \(x\) nor the reverse of \(x\).

**Question 4** (10 points)

Define the features for a log-linear model that can model this distribution \(P(y|x)\) perfectly (Note: you may assume that there are no palindromes in the vocabulary, i.e., no words like eye which stay the same when reversed.) Your model should make use of as few features as possible (we will give you 10 points for using 2 features, 8 points for using 3 features, 5 points for using more than 3 features.)
**Question 5** (10 points)

Write an expression for each of the probabilities

\[ P(\text{the}|\text{the}) \]
\[ P(\text{eht}|\text{the}) \]
\[ P(\text{dog}|\text{the}) \]

as a function of the parameters in your model.
Question 6  (10 points)

What value do the parameters in your model take to give the distribution used by Nathan, described above?
Part #3

We define a PCFG where the non-terminal symbols are \( \{S, A, B\} \), the terminal symbols are \( \{a, b\} \), and the start non-terminal (the non-terminal always at the root of the tree) is \( S \). The PCFG has the following rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \to A \ S )</td>
<td></td>
</tr>
<tr>
<td>( S \to A \ A )</td>
<td></td>
</tr>
<tr>
<td>( S \to A \ B )</td>
<td></td>
</tr>
<tr>
<td>( A \to a )</td>
<td></td>
</tr>
<tr>
<td>( A \to b )</td>
<td></td>
</tr>
<tr>
<td>( B \to a )</td>
<td></td>
</tr>
<tr>
<td>( B \to b )</td>
<td></td>
</tr>
</tbody>
</table>

We are now going to use EM to estimate the probabilities on rules in the grammar. As training data, we have a single string, \( x = aab \). If \( \mathcal{Y} \) is the set of parse trees for \( x \) under the above grammar, our goal is to maximize

\[
L(\vec{\theta}) = \log P(x | \vec{\theta}) = \sum_{y \in \mathcal{Y}} \log P(x, y | \vec{\theta})
\]

with respect to \( \vec{\theta} \). Here \( \vec{\theta} \) is a set of parameters in the model—a vector of values \( \theta_r \) for each \( r \) in the grammar where \( \theta_r \) is the probability associated with rule \( r \). For example, we write

\[
\theta_{S \to AB}
\]

to denote the conditional probability \( P(S \to AB | S) \). \( P(x, y | \vec{\theta}) \) is the probability given to tree \( y \) paired with string \( x \) under parameters \( \vec{\theta} \).
Question 7  (5 points)

For the input string $aab$, there are two parses under the grammar:

\[ y_1 = \begin{pmatrix} S \\ A & S \\ a & A & A \\ a & a \end{pmatrix} \]

\[ y_2 = \begin{pmatrix} S \\ A & S \\ a & A & B \\ a & a \end{pmatrix} \]

Given that $x = aab$, write expressions for $P(x, y_1 | \theta)$ and $P(x, y_2 | \theta)$ as a function of the parameters in the model.
**Question 8** (25 points)

We are now going to derive the EM updates. Assume that we have a current set of parameters $\theta^-1$, and we’d like to estimate updated parameters $\theta^1$. Show how to calculate parameter values $\theta^1$ from the previous parameters $\theta^-1$, using the EM algorithm. Assume again that we have $x = aab$ as the only string in the training data.
Part #4

We define a PCFG where the non-terminal symbols are \{S, A, B\}, the terminal symbols are \{a, b\}, and the start non-terminal (the non-terminal always at the root of the tree) is S. The PCFG has the following rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → A B</td>
<td>0.2</td>
</tr>
<tr>
<td>S → A A</td>
<td>0.2</td>
</tr>
<tr>
<td>S → A S</td>
<td>0.2</td>
</tr>
<tr>
<td>S → B S</td>
<td>0.4</td>
</tr>
<tr>
<td>A → a</td>
<td>0.6</td>
</tr>
<tr>
<td>A → b</td>
<td>0.4</td>
</tr>
<tr>
<td>B → a</td>
<td>0.7</td>
</tr>
<tr>
<td>B → b</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**Question 9** (5 points)

Draw one of the possible parse trees for the string aabb, and show how to calculate its probability.
**Question 10** (25 points)

The following algorithm is the pseudo-code from lecture, which finds the maximum probability for any tree when given as input a sequence of words \( w_1, w_2, \ldots, w_n \).

We take \( K = 3 \), and \( N_1 = S, N_2 = A, N_3 = B \). The value stored in \( \pi[1, n, 1] \) is then the highest probability for any tree spanning the words that is rooted in \( S \).

**Initialization:**

For \( i = 1 \ldots n, k = 1 \ldots K \)

\[
\pi[i, i, k] = P(N_k \rightarrow w_i|N_k)
\]

**Main Loop:**

For \( \text{length} = 1 \ldots (n - 1), i = 1 \ldots (n - \text{length}), k = 1 \ldots K \)

\[
j \leftarrow i + \text{length}
\]

\[
\max \leftarrow 0
\]

For \( s = i \ldots (j - 1) \),

For \( N_1, N_m \) such that \( N_k \rightarrow N_1 N_m \) is in the grammar

\[
\text{prob} \leftarrow P(N_k \rightarrow N_1 N_m) \times \pi[i, s, l] \times \pi[s + 1, j, m]
\]

If \( \text{prob} > \max \)

\[
\max \leftarrow \text{prob}
\]

\[
\pi[i, j, k] = \max
\]

This algorithm will take \( O(n^3) \) time to find the highest scoring tree for a sentence of length \( n \). For our particular PCFG, how would you modify the algorithm to run in \( O(n) \) time? (Hint: look closely at the parse tree you drew for question 5(i). You should see that it has a particular form, and that all other parse trees for this or other sentences will have that particular form.)