CS6120 Midterm Examination

Khoury College of Computer Sciences
Northeastern University

13 March 2020

Please answer each part of all six questions. This is an open book exam: you may use your own notes, course notes, textbooks, etc. Write your own answers in a document, preferably a text file or PDF. If you write your answers out by hand, compile the images into a PDF. If you write your answers in another word-processing program, please export the final document as a PDF or text file.

1 Text Classification

Say that you are training a model to classify headlines into $K = 3$ classes, $\{politics, business, sports\}$. You are using word unigram features and get a training document with the tokens $x^{(i)} = \{huge, loss, ahead\}$. The correct output class is $y^{(i)} = business$, but the current version of the classifier outputs $\hat{y} = sports$. For this question, consider only the *active* features, i.e., the nine components of $f(x^{(i)}, y)$ in the cross product of the input tokens and possible output classes.

1. Say that you are training a perceptron classifier, with a parameter update

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta^{(t)}[f(x^{(i)}, \hat{y}) - f(x^{(i)}, y^{(i)})]$$

   What are the updates (i.e., the part inside the brackets) for the nine active features? You can write nine updates as $\Delta\theta(huge, politics) = \ldots, \Delta\theta(loss, politics) = \ldots$, and so on. Note that you do not need to know the *current* value $\theta^{(t)}$, just the updates to it. You should also ignore the learning rate $\eta^{(t)}$.

2. Say that you are training a logistic regression classifier. Furthermore, say that you are at the beginning of the training process and the model simply predicts a uniform distribution over all outputs, i.e., $\forall y, p(y \mid x^{(i)}) = \frac{1}{3}$. What are the updates for the nine active features? You can write them as above. You do not need to know the current values of the parameters or the learning rate.

2 Language Models

For both n-gram and recurrent language models, we commonly factor the joint probability of a string into the probabilities of tokens (words, morphemes, or letters) in order from the beginning of the sequence to the end. In some situations, such as for a bidirectional LSTM, it can be helpful to compute the probabilities of tokens in the reverse order.

Say that you wanted to build a reverse-order trigram model with add-λ smoothing. You had a vocabulary of V words and a training corpus with N words. You want to compute the likelihood of the sentence:

the rain is raining all around

and you thus need to compute probabilities such as $p(\text{the} \mid \text{rain}, \text{is})$. Write down the formulas you would use to estimate each of the probabilities necessary to compute the likelihood of this sentence under a reverse trigram model.

3 Naive Bayes Classification

1. When we use Laplace (add 1) smoothing on the maximum likelihood estimates, are any feature likelihoods $p(\text{feature} \mid \text{class}) = 1$? Why or why not?

2. We discussed in class how to compute the posterior probability of a class Y given a document X using Bayes' rule:

$$p(Y \mid X) = \frac{p(X \mid Y)p(Y)}{p(X)}$$

   Furthermore, the naive Bayes model commonly writes the probability of an N-word document given a class as the product of individual words:

$$p(X \mid Y) = \prod_{i=1}^{N} p(x_i \mid Y)$$

   We don't usually compute the denominator $p(X)$ in Bayes' rule above, since we're usually only interested in the class with the maximum posterior probability. It is sometimes useful, however, to get an actual probability from a model (e.g., to compute perplexity, or to combine with other models). How would you compute $p(X)$ for an N-word document when the number of classes $K = 3$?

4 Feedforward Neural Networks

1. Say that you are building a feedforward neural network for text classification, with an input vocabulary of V unique words, K output classes, and H hidden units. How many parameters would you need to estimate?

2. Show that the softmax and sigmoid activation functions are equivalent when the number of possible output labels is two. Specifically, for any parameters to a softmax layer $\Theta^{(z \to y)}$ (omitting the bias terms for simplicity), show how to construct a vector of weights θ for a sigmoid layer such that,

$$\text{SoftMax}(\Theta^{(z \to y)} z)[0] = \sigma(\theta \cdot z)$$

i.e., the activation of the first output of the two-class softmax is equal to the activation of the single output of the sigmoid.

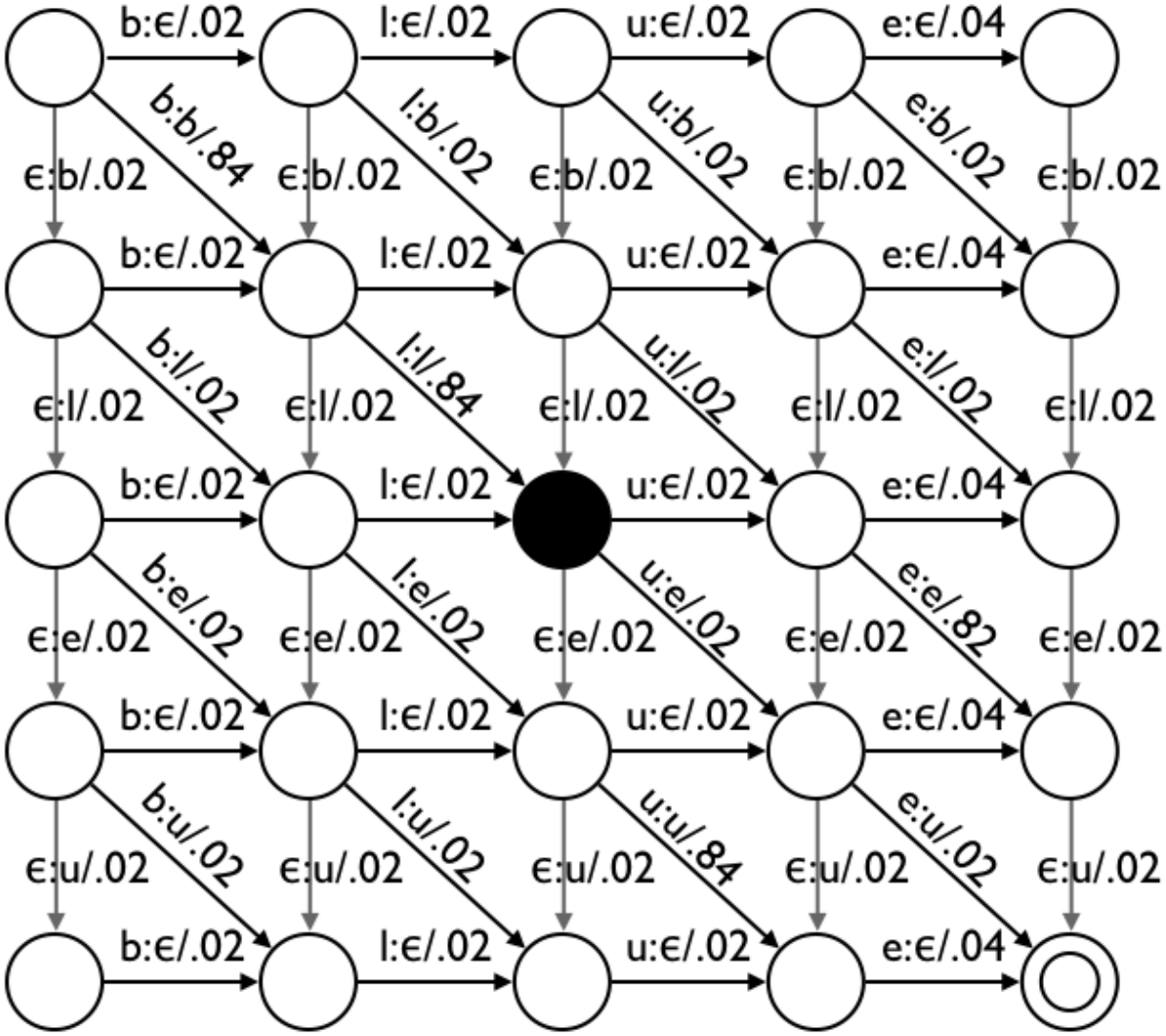
5 Sequence Labeling

1. In a sequence labeling problem with K labels and M tokens, how many nodes are in the trellis?

2. When using sequence labeling for part-of-speech tagging an M-word sentence with K tags, describe a situation where there might be fewer nodes than this in the trellis.

6 Viterbi and Forward Algorithms for Edit Distance

The Viterbi and forward algorithms we learned for hidden Markov models and conditional random fields can be easily adapted to edit distance computations. As we mentioned briefly in class, the graph or "chart" for computing the edit distance has the following form:



In this example, we are using edit distance for the common problem of comparing cognate words in two languages: English *blue* and French *bleu*. We use ϵ for the empty string.

1. The weights on the edges in this graph represent probabilities. We thus wish to find paths with the *highest product* of edge weights, rather than the lowest sum. In other words, we are working in the (max, times) probability semiring, not the (min, plus) shortest distance semiring. Use the Viterbi algorithm to find the probability of the highest-probability path between the black node in the center of this graph and the final state in the lower right of this graph. In addition, write down the series of edges that this highest-probability path traverses.

2. Assume now that all edges in the complete graph have weight 1. Working in the (plus, times) semiring, use the forward algorithm to compute the sum of the score of all paths from the upper left to the lower right of the graph. (Don't start from the black node this time.) Since each individual path will now have a weight of $1 \cdot 1 \cdot \cdots 1 = 1$, you will therefore simply be *counting* the number of distinct paths through the graph.