Midterm

Midterm

- Midterm date is set to **April 10th**.
- The midterm will be closed notes, books, laptops, smartphones, and people.

  You may (optionally) bring a calculator, if you want to.
- 75 minutes in class.

Midterm

- **Preparation:**
  - Lecture slides + textbook readings
  - Written homework (PS0, PS1, and PS2)
  - Programming Project 0, 1 and 2

- **Expectation:**
  - **As a close-book close-note exam,** you are not expected to be able to answer all the questions correctly.
  - You may get challenged on some of the questions.
  - Try your best! We may curve for final letter grades, if it is needed …

Midterm

- Broad types of questions (similar to PS1/2):
  - Long answers, e.g., simple toy example, pseudocode
  - Short answers
  - True/False
  - Multiple choice

- Make sure you **understand** the fundamentals in addition to being able to procedurally execute a few key algorithms (e.g., Viterbi, Beam search).
- But, in general, we will have a greater emphasis on open-ended and conceptual questions in midterm (e.g., what are the main drawbacks/advantages of XXX model? Or, we will describe a new technique and ask you to reason about it).

Example Questions

Multi-choice Questions

(1 point) Which of the following best describes the BLEU score (select the best answer below):

(a) The objective function used for training neural machine translation systems.

(b) A metric that is used to evaluate translation quality in which human judges rate translations on a scale from 1-5.

(c) A metric that is used to automatically evaluate machine translation systems.

(d) A way to measure the difficulty of translation between a given language pair (e.g., English/French).

(1 point) What problem do LSTMs address that vanilla recurrent neural networks (e.g., Elman Networks) suffer from? (select the best answer from the choices below)

(a) Elman Networks suffer from slow training times

(b) Elman Networks suffer from the problem of vanishing gradients

(c) LSTMs have fewer parameters, so they are likely to overfit

(d) Elman Networks are more difficult to parallelize

(e) Elman Networks have lower accuracy

Example Questions

This question is dedicated to my PhD advisor, Ralph Grishman. He designed this lovely toy example with a two-word language, which namely consists of only two words: $fish$ and *sleep*. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden state. It can be used to solve Hidden Markov Models (HMMs) as well as many other problems.

Suppose we have a small training corpus. In this training corpus, word $fish$ appears 8 times as a noun (NN) and 5 times as a verb (VB); word *sleep* appears twice as a noun and 5 times as a verb.

(1) What are the emission probabilities?

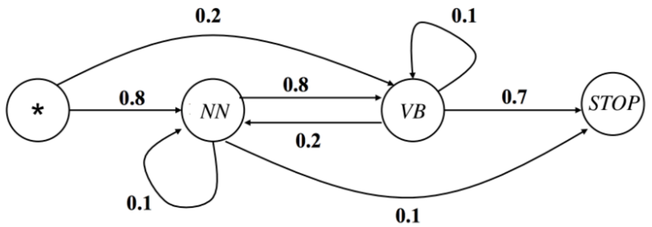
$e(fish|NN)$

$e(sleep|NN)$

$e(fish|VB)$

$e(sleep|VB)$

Also suppose we already have a simple Part-of-Speech HMM model, a bigram one:



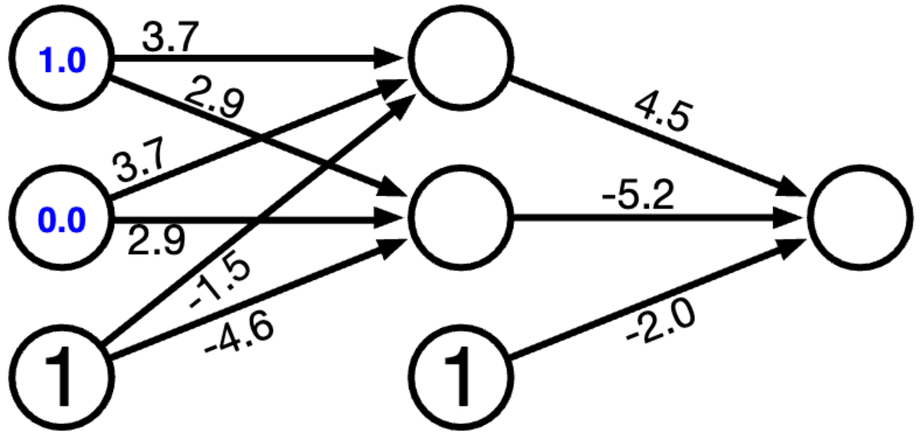
The arrows in the diagram denote conditional probabilities, e.g. $q(NN|*) = 0.8$ and $q(NN|VB) = 0.2$.

(2) Use the Viterbi algorithm to find the most likely POS tag sequence for a test sentence *"Fish sleep."* For simplicity, we ignore the punctuation in the calculation for this toy example.

Small toy example to walk through some algorithms/models

Example Questions

Below is a simple neural network with one hidden layer:



$$\mathbf{W}_0 = \begin{bmatrix} 3.7 & 3.7 \\ 2.9 & 2.9 \end{bmatrix}, \mathbf{b}_0 = [-1.5, -4.6], \mathbf{W}_1 = \begin{bmatrix} 4.5 \\ -5.2 \end{bmatrix}, \mathbf{b}_1 = [-2.0]$$

Assume we use sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$ as activation function. Let's try out two input values $\mathbf{x} = [1.0, 0.0]^T$ and calculate the hidden layer $\mathbf{h}$ and outputs y of the network.

Midterm will not involve as much real-number multiplication as in this question, but single-digit multiplications.

Example Questions

1 Perception Algorithm

$\mathbf{x}$: vector of n features for a single instance; $\mathbf{w}$: vector of n weights; b: bias; y: class label for this instance; .

Activation a is the outcome score, used in both training and testing. It is about making prediction for a single instance (online learning) with the current set of weights:

$$a = \sum_n w_n x_n = \mathbf{w}^T \mathbf{x} + b \qquad \hat{y} = SIGN(a)$$

Training:

Start with some initial weight vector $\mathbf{w}$ and bias term b (such as 0)
Loop for K iterations
  For each training instance, compute activation a
    if $ya > 0$, do nothing
    if $ya \leq 0$, update the weights:
      $\mathbf{w} = \mathbf{w} + y\mathbf{x}$
      $b = b + y$

(1) What does checking $ya \leq 0$ do? Would $ya < 0$ work as well?
(2) Let's try an example. Suppose we have the following data points, and no bias term (draw 2D plot):
  $x_1 = (1, 2), y_1 = 1$
  $x_2 = (0, -1), y_2 = -1$
  $x_3 = (2, 1), y_3 = -1$

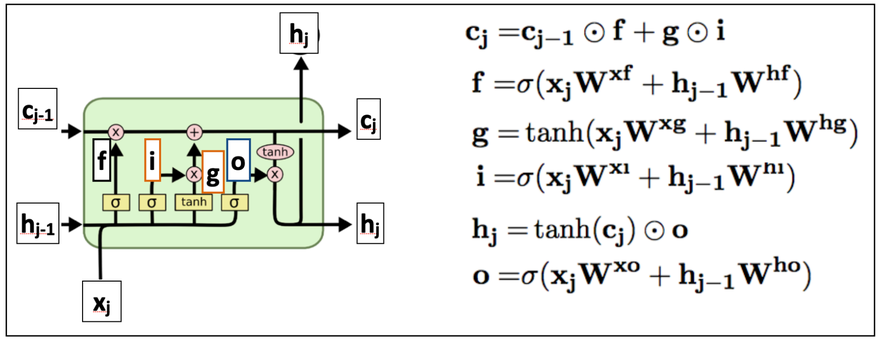
(3) Show, mathematically, why the parameter updates will make it do better on the same training instance next time around?

You may see some pseudocode,

or be asked to write some

Topics

- **Topics that will not be tested:**
  - Naive Bayes
  - SVM
- **Topics that will not have very involved questions, but possibly conceptual or short-answer questions:**
  - No execution of forward-backward algorithm on toy example
  - LSTM/GRU, if tested, we will provide corresponding equations
  - No need to memorize BLEU score's equation
  - Back-propagation



Topics

- **Important topics that we may test on (not limited to):**
  - Perceptron
  - Logistic regression: model, training objective, gradient update, etc.
  - Optimization: stochastic gradient descent, learning rate, initialization, etc.
  - Training neural networks
  - Feedforward, CNN, RNN, LSTM: model architecture, dimensionality, pros/cons, etc.
  - Word2vec/skip-gram
  - Sequence-to-sequence model, attention mechanism
  - Transformer, self-attention
  - Subword, tokenization

Topics

- **Important topics that we may test on (not limited to):**
  - Sequential task: NER BIO tagging scheme
  - Evaluation: Precision/Recall/F1, BLEU score
  - HMM: definition, parameter estimation, Viterbi Algorithm
  - CRF: advantages, forward-backward algorithm
  - MT: Beam search, language models
  - Runtime complexity of different algorithms


KEEP
CALM
AND
HAVE A NICE
SPRING BREAK