ITCS 6101/8101 Homework 4: Theory (50 points)

March 5, 2026

1 Language modeling (40 points)

Consider a bigram language model (LM) with the following probabilities, where S means beginning of sentence:

- P(S | fear) = 0.5, P(fear | S) = 0.4, P(fear | to) = 0.3, P(fear | leads) = 0.1
- P(leads | fear) = 0.1, P(leads | anger) = 0.2, P(leads | hatred) = 0.3
- P(to | leads) = 0.2, P(to | anger) = 0.1, P(to | hatred) = 0.2 P(to | conflict) = 0.1
- P(leads | to) = 0.2 , P(anger | to) = 0.3 P(conflict | to) = 0.2 P(hatred | to) = 0.1

Consider the text in the sentence below:

```
fear leads to anger leads to hatred leads to conflict
```

Answer the following questions (show your work in detail):

1. (20p) Compute the **probability** that the LM assigns to the sentence.
2. (10p) Compute the **perplexity** that the LM assigns to the sentence.
3. (10p) Is it theoretically possible to have a text corpus such that a bigram LM (any implementation, e.g., n-gram or neural) trained on that corpus ends up learning the bigram probabilities above? Explain.

2 Sampling Methods (40 points)

Consider a language model where, for an input sequence $\mathbf{x}$, the output softmax layer computes the following word probabilities:

$$\begin{aligned} \mathbf{p} &= [p(Horton|\mathbf{x}), p(Morton|\mathbf{x}), p(Vlad|\mathbf{x}), p(Ned|\mathbf{x})] \\ &= [0.1, \quad 0.5, \quad 0.3, \quad 0.1] \end{aligned}$$

Answer the following questions, **justifying each answer** in detail:

1. (5p) If we were to use **greedy sampling** according to this probability distribution 100 times, how many times would we expect to see $Morton$ sampled?
2. (5p) If we were to use **multinomial sampling** according to this probability distribution 100 times, how many times would we expect to see $Vlad$ sampled?
3. (10p) Show the general formula for **softmax with temperature**. Explain all notation used.

4. (10p) If we were to use **multinomial sampling** where the softmax that computed the distribution above was changed to use a **temperature** τ, how many times we would expect to see the *Horton* sampled if:

   (a) $\tau = 0$

   (b) $\tau = 1$

   (c) $\tau = +\infty$

5. (5p) I want *Morton* to be sampled less often than **p** above would allow for. Which interval should τ belong to in order for this to happen?

6. (5p) I want *Ned* to be sampled less often than **p** above would allow for. Which interval should τ belong to in order for this to happen?

3 Language Modeling with FCNs vs. RNNs (40 points)

Consider implementing a language model LM, where:

- The size of the vocabulary is $|V| = 1,000$.
- The token embeddings have size $E = 100$ and are frozen.

Answer the following questions, justifying each answer in detail:

1. (10p + 5p) If the LM is implemented using a fully connected network (FCN) that takes as input the previous $K = 10$ tokens concatenated, has one hidden layer with size $H = 50$ and a softmax output layer, how many parameters are in total?

   (a) Implement the FCN above in PyTorch and verify your explanation by printing the number of parameters for each layer and their total number.

2. (10p + 5p) If the RNN is implemented using a recurrent neural network (RNN) that takes as input the previous $K = 10$ tokens, has a hidden state of size $H = 50$ and and a softmax output layer, how many parameters are in total?

   (a) Implement the RNN above in PyTorch and verify your explanation by printing the number of parameters for each layer and their total number.

3. (10p) Specify one advantage and one disadvantage of using the RNN vs. the FCN approach for language modeling.

4 RNN Design (30 points)

Consider an RNN where the input at each time step is binary $x_t \in \{0, 1\}$ and the output is a binary label $y_t \in \{0, 1\}$. The RNN should output the positive label $y_t = 1$ if and only if the number of values 1 seen so far is strictly greater than the number of values 0 seen so far. For example:

- If the input sequence is $\langle x_1, x_2, x_3, x_4, x_5 \rangle = \langle 1, 1, 0, 0, 1 \rangle$.

- Then the output sequence should be $\langle y_1, y_2, y_3, y_4, y_5 \rangle = \langle 1, 1, 1, 0, 1 \rangle$.

The RNN is specified through:

- A linear state transition equation $\mathbf{h}_t = W\mathbf{h}_{t-1} + \mathbf{u}x_t + \mathbf{b}_h$, where W is a $k \times k$ matrix, $\mathbf{h}_{t-1}$, $\mathbf{u}$, and bias $\mathbf{b}_h$ are $k \times 1$ vectors, and input x_t is a 1×1 scalar.
- The output is produced by a binary logistic model $\hat{y}_t = \sigma(\mathbf{v}^T\mathbf{h}_t + \mathbf{b}_y)$, where $\mathbf{v}$ is a $k \times 1$ vector and bias $\mathbf{b}_y$ is a 1×1 scalar.

Solve the following exercises:

1. (20p) Specify values for the initial hidden state $\mathbf{h}_0$ and the RNN parameters W, $\mathbf{u}$, $\mathbf{b}_h$, $\mathbf{v}$, and $\mathbf{b}_y$ such that the RNN solves the problem above. Explain why your model will work on any binary sequence. Keep it simple, e.g., try to use as small a value for k as possible.
2. (10p) Implement the model you design above in **NumPy** and verify empirically that it works on 10 random bit sequences that you generate, with lengths going from 1 to 10.

4.1 Within vs. Out of Distribution Length Generalization (40 bonus points)

Create the following datasets, based on a hyperparameter K:

1. **Train**: Random binary sequences of length 1, 3, 4, 7, 10, 11, 13, 15. For each length, generate K random binary sequences.
2. **Validation**: Random binary sequences of length 1, 2, ..., 100. For each length, generate 10 random binary sequences.
3. **Test-WiD**: Random binary sequences of length 1, 3, 4, 7, 10, 11, 13, 15. For each length, generate 10 random binary sequences.
4. **Test-OoD1**: Random binary sequences of length 2, 5, 6, 8, 9, 12, 14. For each length, generate 10 random binary sequences.
5. **Test-OoD2**: Random binary sequences of length 16, 17, 18, 19, 20, 25, 30, 50, 75, 100. For each length, generate 10 random binary sequences.

Implement and evaluate the following:

1. (20p) Train and tune a vanilla RNN that is the same as above, but use a `tanh` non-linear activation in the state transition equation. Evaluate it separately on the 3 test datasets and report accuracy.
2. (20p) Train and tune a gated RNN, e.g. LSTM and/or GRU, and then evaluate it separately on the 3 test datasets and report accuracy.

Report learning curves showing test accuracy as a function of the size of the training dataset, by varying K from 10 to 100 in steps of 10. Do your best in terms of tuning the models on the Validation data, e.g., the number of layers, the size of the hidden state, regularization, etc. To automate this step, you may consider using the Model Selection procedures provided by `sklearn`, or the Optuna hyperparameter optimization framework.

5 Submission

Submit your theory responses on Canvas as one file named `hw04-theory.pdf` and the PyTorch solutions as one file named `hw04-code.py`. It is important that you show clearly all the derivation steps. We recommend using an editor such as Overleaf for Latex, Word, or Jupyter-Notebook that allows proper formatting of equations. Alternatively, if you choose to write your solutions on paper, submit an electronic scan / photo of it exported to **PDF**. Make sure that your writing is **legible** and the scan has good quality (we will not grade solutions that we struggle to read).