CS 4650 (Spring 2021) HW2 – Neural Network Basics and Sequence Models

February 2021

Instruction

1. We will be using Gradescope to collect your assignments. Please carefully read the following instructions for submitting to Gradescope.
   (a) Each subproblem must be submitted on a separate page. When submitting to Gradescope (under **HW2 Writing**), make sure to mark which page(s) correspond to each problem or subproblem.
   (b) Note: this is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.
2. LaTeX solutions are strongly encouraged (solution template available on the class website), but scanned handwritten copies are also acceptable. Hard copies are not accepted.
3. We generally encourage collaboration with other students. You may discuss the questions and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with on the submission site.

1 Named Entity Recognition (4 points)

Consider a sentence that contains three named entities (one organization name, one product name, one location name) and the predictions from four automatic name entity recognition systems. What is the entity-level Precision, Recall, and F1-score of each system's performance? Here, we do not consider giving any credits to partial matches.

Sentence	`Nintendo`	`confirms`	`RingFit`	`Adventure`	`shortage`	`in`	`the`	`Americas`
Gold Labels	B-Org	O	B-Prod	I -Prod	O	O	O	B-Loc
System #1	B-Org	O	O	O	O	O	O	B-Loc
System #2	B-Org	O	B-Prod	I -Prod	O	O	O	B-Loc
System #3	B-Org	O	B-Prod	I -Prod	O	O	B-Loc	B-Loc
System #4	B-Org	O	B-Prod	I -Prod	O	O	B-Loc	I-Loc

For each system compute:

(a) Precision

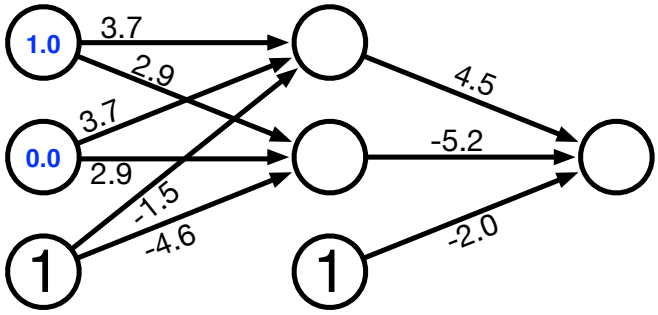
(b) Recall

(c) F-1 score

2 Neural Network

We learned about neural networks in class. The combination of different model architectures, non-linearity through different activation functions, and various ways to avoid overfitting make neural networks a flexible and robust technique to learn from data.

(a) Below is a simple neural network with one hidden layer:



$$\mathbf{W}_1 = \begin{bmatrix} 3.7 & 3.7 \\ 2.9 & 2.9 \end{bmatrix}, \mathbf{b}_1 = \begin{bmatrix} -1.5 \\ -4.6 \end{bmatrix}, \mathbf{W}_2 = \begin{bmatrix} 4.5 & -5.2 \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} -2.0 \end{bmatrix}$$

Assume we use the sigmoid function, $\sigma(z) = \frac{1}{1+e^{-z}}$, as the activation function. Suppose the input $\mathbf{x} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$. Calculate the hidden layer $\mathbf{h}$ (after activation), and the output $\mathbf{y}$ (after activation). (2 points)

(b) Assume a neural network with one hidden layer, $\mathbf{y} = f(\mathbf{x})$, such that

$$\begin{aligned} \mathbf{a} &= \mathbf{W}_1\mathbf{x} + \mathbf{b}_1 \\ \mathbf{h} &= \sigma(\mathbf{a}) \\ \mathbf{y} &= \mathbf{W}_2\mathbf{h} + \mathbf{b}_2 \end{aligned}$$

where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the weights and biases for the i-th linear layer, and $\sigma(\cdot)$ stands for the sigmoid activation function.

You may view this neural network as a blackbox — it takes an input $\mathbf{x}$, and return an output $\mathbf{y}$. Now assume we modify this neural network a little bit by replacing $\sigma(\cdot)$ with $\tanh(\cdot)$ [$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$]. No doubt that for the same input $\mathbf{x}$, this new neural network will return a different output.

But we still want the original output, i.e. return $\mathbf{y}$ when given $\mathbf{x}$. So the question is, how would you adjust $\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2$ such that this requirement would be satisfied?

[Hint: derive the relationships between $\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2$ and $\mathbf{W}_1^{\text{new}}, \mathbf{b}_1^{\text{new}}, \mathbf{W}_2^{\text{new}}, \mathbf{b}_2^{\text{new}}$] (4 points; think of the relationship between *sigmoid* and *tanh* functions)

(c) Among many regularization techniques to avoid overfitting in neural networks, dropout is the simplest. During training, dropout randomly sets units in the hidden layer $\mathbf{h} \in \mathbb{R}^n$ to zero with probability p, and then multiplies $\mathbf{h}$ by a constant γ.

We can write this as $\mathbf{h}^{\text{drop}} = \gamma d \odot \mathbf{h}$, where $d \in \mathbb{R}^n$ is a mask vector where each entry is 0 with probability p, and 1 with probability $1-p$. $\odot$ is the element-wise product operation (also known as the Hadamard product). γ is chosen such that the expected value of $\mathbf{h}^{\text{drop}}$ equals $\mathbf{h}$, i.e. $\mathbb{E}_p[\mathbf{h}_i^{\text{drop}}] = \mathbf{h}_i$, $i = 1, \ldots, n$.

What should the value of γ be? [Hint: in terms of p] (2 points)

3 Hidden Markov Models and the Viterbi Algorithm

We have a toy language with 2 words - "clean" and "home". We want to tag the parts of speech in a test corpus in this toy language. There are only 2 parts of speech — NN (noun) and VB (verb) in this language.

We have a corpus of text in which we the following distribution of the 2 words:

	NN	VB
home	8	3
clean	2	7

Assume that we have an HMM model with the following transition probabilities (* is a special start of the sentence symbol).

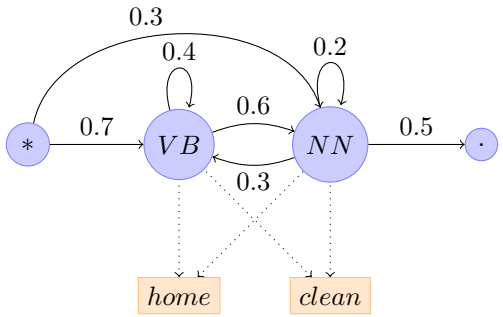


Figure 1: HMM model for POS tagging in our toy language.

(a) Compute the emission probabilities for each word given each POS tag. (2 points)

(b) Draw the Viterbi trellis for the sequence "clean home.". Highlight the most likely sequence. (6 points)