# COMP4901K and MATH4824B Final Exam

December 12, 2018, Wednesday

Time: 8:30am-10:30am

Instructor: Yangqiu Song

**Name:** ________________ **Student ID:** ________________

| **Question** | **Score** | **Question** | **Score** |
|---|---|---|---|
| 1 | / 10 | 6 | / 10 |
| 2 | / 5 | 7 | / 10 |
| 3 | / 15 | 8 | / 10 |
| 4 | / 10 | 9 | / 10 |
| 5 | / 10 | 10 | / 10 |
| **Total:** | **/ 100** | | |

## Q1. Yes/No Questions (10 Points)

Indicate whether each statement is true (✓) or false (×).

1. For a fully connected deep network with one hidden layer, increasing the number of hidden units could decrease bias and increase variance of the prediction.
2. Recurrent neural networks can handle a sequence of arbitrary length, while feedforward neural networks can not.
3. Multi-layer neural network model trained using stochastic gradient descent on the same dataset with different initializations for its parameters is guaranteed to learn the same parameters.
4. Stochastic gradient descent results in a smoother convergence plot (loss vs epochs) as compared to batch gradient descent.
5. In supervised learning, training data includes both inputs and desired outputs.

TFFFT

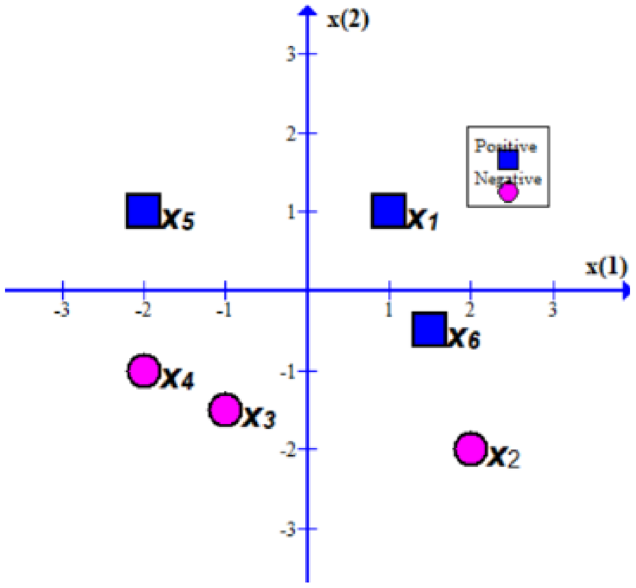
## Q2. Short-Answer Questions (5 Points)

Please provide short answers to following questions:

1. What are epoch, batch size, and learning rate in the context of machine learning? (**(2pts)**
   (a) One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.
   (b) The batch size is a hyper-parameter that defines the number of samples to work through before updating the internal model parameters.
   (c) Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient.
2. What's the risk with tuning hyper-parameters using the test dataset? **(2pts)** Tuning model hyper-parameters to a test set means that the hyper-parameters may overfit to that test set. If the same test set is used to estimate performance, it will produce an overestimate. Using a separate validation set for tuning and test set for measuring performance provides unbiased, realistic measurement of performance.
3. Provide a typical goal of (good) weight initialization for deep neural networks. **(2pts)**The goal of good weight initialization is to break symmetry and generate activations that lead to non-zero initial gradients. Weights should be large enough that gradients don't decay to zero through a deep network, and not so large that non-linear layers saturate.

## Q3. Perceptron (15 Points)

The perceptron algorithm is to predict the label $y_i' = sgn(w^T x_i)$ for each data point $x_i$. For the following dataset, please derive the perceptron algorithm and give the solution to $w$.



We set: learning rate $\alpha$=0.2 and initial weight vector $w_0 = [1, 0.5, 0]^T$. The input sequence is:

$$x_1 = [1, 1, 1]^T$$
$$x_2 = [2, -2, 1]^T$$
$$x_3 = [-1, -1.5, 1]^T$$
$$x_4 = [-2, -1, 1]^T$$
$$x_5 = [-2, 1, 1]^T$$
$$x_6 = [1.5, -0.5, 1]^T$$

3 points for the algorithm. 2 points for each step (1 point for classification result and 1 point for update)

## Q4. Logistic Regression (10 Points)

Remember the form of Logistic Regression function

$$P(y=1|x;\theta) = \frac{1}{1+e^{-\theta^T x}}$$

Let us assume

$$P(y=1|x;\theta) = \sigma_\theta(x) \quad \text{and} \quad P(y=0|x;\theta) = 1-\sigma_\theta(x)$$

then

$$P(y|x;\theta) = (\sigma_\theta(x))^y (1-\sigma_\theta(x))^{1-y}$$

Now you can learn your logistic model using maximum likelihood estimation, for which you should use gradient descent to maximize the log-likelihood function iteratively. Derive the **stochastic gradient descent** update rule for logistic regression.

$$\max P(y|x;\theta) \Rightarrow$$

$$\begin{aligned}\max \log P(y|x;\theta) &= \max \log\left(\sigma_\theta(x)^y\,(1-\sigma_\theta(x))^{1-y}\right)\\ &= \max\left(y\log\left(\sigma_\theta(x)\right) + (1-y)\log\left(1-\sigma_\theta(x)\right)\right) \quad \textbf{(3 pts)}\\ &= \max(J(y|x;\theta))\end{aligned}$$

Since $\sigma_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$, we have $\sigma'_\theta(x) = \sigma_\theta(x)\left(1-\sigma_\theta(x)\right)$ We can get it's gradients as:

$$\begin{aligned}\nabla_\theta J(y|x;\theta) &= \frac{y}{\sigma_\theta(x)} \times \sigma'_\theta(x) - \frac{1-y}{1-\sigma_\theta(x)} \times \sigma'_\theta(x)\\ &= \underbrace{-y\left(1-\sigma_\theta(x)\right)x}_{\textbf{(3pts)}} + \underbrace{(1-y)\sigma_\theta(x)x}_{\textbf{(3pts)}}\\ &= x\left(\sigma_\theta(x)-y\right)\end{aligned}$$

So $\boldsymbol{\theta}_j := \boldsymbol{\theta}_j - \eta\nabla_\theta J(y^{(i)}|x_j^{(i)};\theta) = \boldsymbol{\theta}_j - \eta\left(\sigma_{\boldsymbol{\theta}}(x_j^{(i)}) - y^{(i)}\right)x_j^{(i)}$ **(1pts)**

## Q5. Statistical Language Models (10 Points)

We build a uni-gram language model based on the following word frequency table. (**Do NOT use any smoothing technique.**)

| word | frequency | word | frequency |
|---|---|---|---|
| i | 100 | hate | 50 |
| you | 100 | book | 100 |
| he | 50 | desk | 50 |
| she | 50 | paper | 200 |
| like | 100 | reading | 60 |
| love | 100 | writing | 40 |

(1) What is the likelihood of the sequence "i love reading" based on the language model you build? P("i")=0.1 (1 points)
P("love"=0.1) (1 points)
P("reading"=0.06) (1 points)
P("i love reading") = 0.0006 (3 points)

(2) List a disadvantage of the uni-gram language model. (4 points) 1. do not consider the context information (previous words).
2. ignore the order of words.
3. words with high frequency have high probability, which will make the sequence consisting of high frequency words, i.e., "paper paper paper" very high , but the sequence itself is meaningless.

## Q6. Language Model Applications (10 Points)

Nathan L. Pedant would like to build a spelling corrector focused on the particular problem of there vs. their. The idea is to build a model that takes a sentence as input, for example

He saw their football in the park (1)

He saw their was a football in the park (2)

and for each instance of their or there predict whether the true spelling should be their or there. So for sentence (1) the model should predict their, and for sentence (2) the model should predict there. Note that for the second example the model would correct the spelling mistake in the sentence.

Nathan decides to use a language model for this task. Given a language model $p(w_1, \ldots, w_n)$, he returns the spelling that gives the highest probability under the language model. So for example for the second sentence we would implement the rule

If

$$p(He\ saw\ their\ was\ a\ football\ in\ the\ park) > p(He\ saw\ there\ was\ a\ football\ in\ the\ park)$$

Then Return their

Else Return there.

(1): The first language model Nathan designs is of the form

$$p(w_1, \ldots, w_n) = \prod_{i=1}^{n} q(w_i),$$

where

$$q(w_i) = Count(w_i)/N$$

and $Count(w_i)$ is the number of times that word $w_i$ is seen in the corpus, and $N$ is the total number of words in the corpus. Let's assume

$$N = 10,000,$$

$$Count(there) = 50,$$

and

$$Count(their) = 100.$$

Assume in addition that for every word $v$ in the vocabulary, $Count(v) > 0$. What does the rule given above return for "He saw their was a football in the park?" (there or their?)

their (100/10000) > there (50/10000)

2+2 points for probabilities and 1 point for the answer.

(2): The second language model Nathan designs is of the form

$$p(w_1, \ldots, w_n) = q(w_1) \prod_{i=2}^{n} q(w_i | w_{i-1}),$$

where

$$q(w_i|w_{i-1}) = Count(w_i, w_{i-1})/Count(w_{i-1})$$

and $Count(w_{i-1})$ is the number of times that word $w_{i-1}$ is seen in the corpus, and $Count(w_i, w_{i-1})$ is the number of co-occurrences of two words in the corpus. Let's assume

$$Count(there, saw) = 20,$$

$$Count(their, saw) = 40,$$

$$Count(saw) = 200,$$

$$Count(was, there) = 20,$$

$$Count(was, their) = 2,$$

$$Count(there) = 50,$$

$$Count(their) = 100.$$

We also assumes all other probabilities are non-zero. What does the rule given above return for "He saw their was a football in the park?" (there or their?)

there (20/200 * 20/50) > their (40/200 * 2/100)
2+2 points for probabilities and 1 point for the answer.

## Q7. Convolutional Neural Network (10 Points)

A convolutional neural network has one `2x3` convolutional layer with stride `1` and a max pooling layer. Suppose the input sequence is "a cat sits on the mat"; the filter $K$ is defined as follows,

$$K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}$$

and embeddings (2 dimensional) of input sequence are listed as follows,

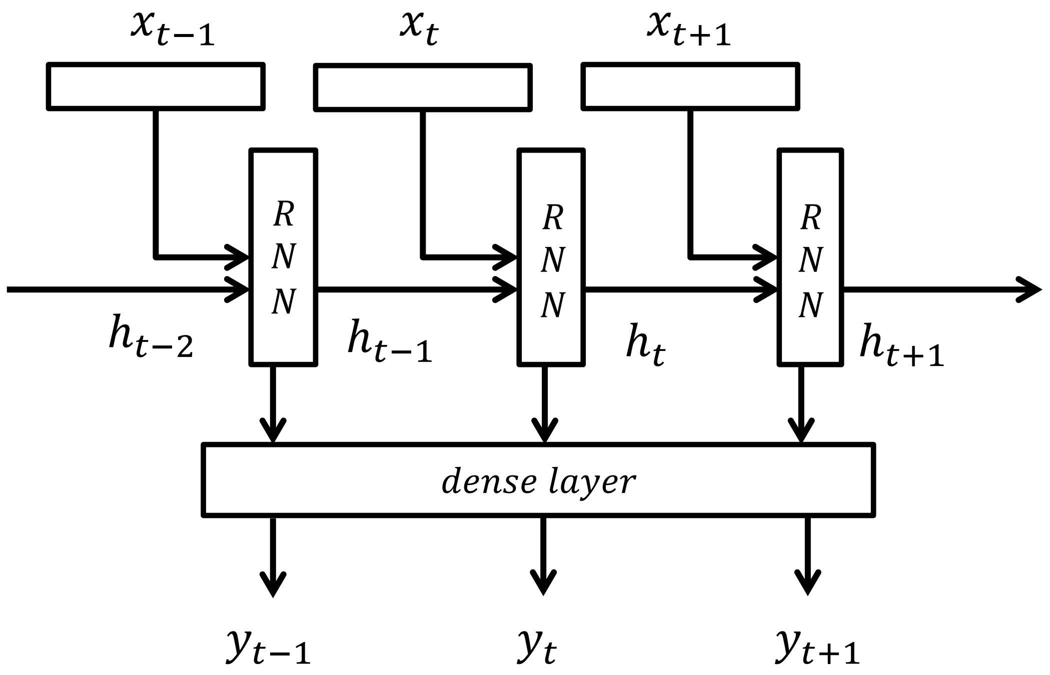
| a | cat | sits | on | the | mat |
|---|---|---|---|---|---|
| $\begin{bmatrix} 0.1 \\ 1.2 \end{bmatrix}$ | $\begin{bmatrix} 0.2 \\ 1.0 \end{bmatrix}$ | $\begin{bmatrix} 0.3 \\ 0.8 \end{bmatrix}$ | $\begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}$ | $\begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}$ | $\begin{bmatrix} 0.6 \\ 0.2 \end{bmatrix}$ |

(i) What is the output of the max pooling layer? (6 points) max ([0.6,0.8,1.0,1.2]) = 1.2
0.6, 0.8, 1.0, 1.2 worth 1 point. 1.2 worth 2 point.

(ii) What is the role of a pooling layer in a convolutional neural network (4 points)? 1. can convert the variant length inputs into fixed length outputs.
2. dimension reduction.
3. extract the most significant features.

## Q8. Recurrent Neural Network (10 Points)

Considering the following architecture: This architecture can be formalized as follows:



$$\begin{aligned} h_t &= tanh\left(W_h h_{t-1} + W_i x_t\right) \\ v_t &= relu(W_v h_t + b) \\ y_t &= \text{softmax}\left(W_o v_t\right) \end{aligned}$$

where $W_h, W_i, W_v, W_o$ are the parameters, and $x_t \in R^D$, $h_t \in R^H$, $v_t \in R^V$, $y_t \in R^Y$. Now given $D = 50, H = 100, V = 10, Y = 1000$. Please compute the number of all the parameters in these network in two cases: word embeddings $x_t$ are trainable and not trainable (*hint:* here parameters are those updated while training).

If word embedding is not trainable:

$$\begin{aligned} N_{param} &= N_{W_h} + N_{W_i} + N_{W_v} + N_b + N_{W_o} \\ &= H \times H + H \times D + H \times V + V + V \times Y \\ &= 10,000 + 5,000 + 1,000 + 10 + 10,000 \\ &\quad (1pts) \quad (1pts) \quad (1pts) \quad (1pts) \quad (1pts) \\ &= 26,010 \end{aligned}$$

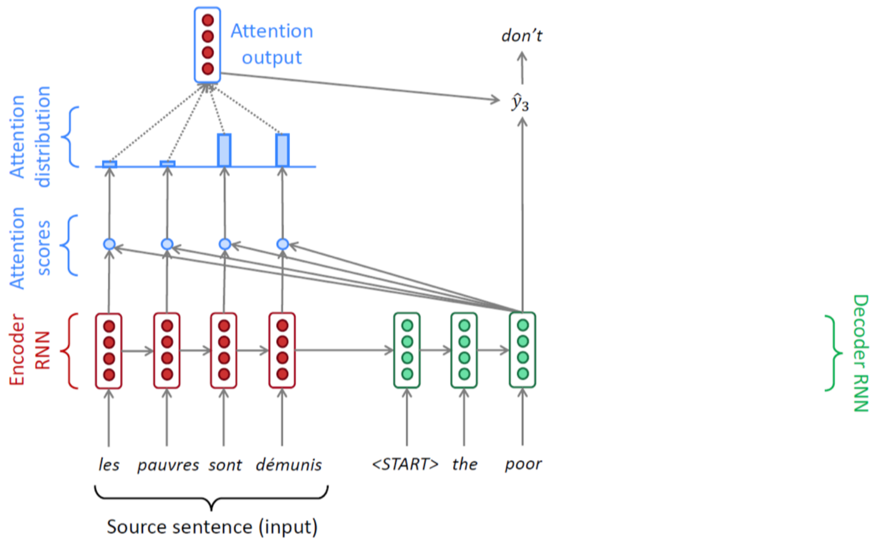
If word embedding is trainable:

$$\begin{aligned} N_{param} &= H \times H + H \times D + H \times V + V + V \times Y + D \times VocabSize \\ &= 10,000 + 5,000 + 1,000 + 10 + 10,000 + 50,0000 \\ &\quad (1pts) \quad (1pts) \quad (1pts) \quad (1pts) \quad (1pts) \quad (2pts) \\ &= 526,010 \end{aligned}$$

PS: (1) If you miss any term, then you will loss the corresponding point. (2) For each question, if every term is true but the final results are not true, you will loss 1 points

## Q9. Dot-Product Attention (10 Points)

Sequence to Sequence (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in French) to sequences in another domain (e.g. the same sentences translated to English), introduced in 2014 by Sutskever et al. But the simple version Seq2Seq utilizes limited information. Attention mechanism is a simple but effective way to improve the performance of Seq2Seq.



The details of Dot-product Attention are as follows:

1. Compute the dot product: $e_{t,j} = \boldsymbol{s_{t-1}^T h_j}$, where $\boldsymbol{s_{t-1}^T}$ is the transpose of $\boldsymbol{s_{t-1}}$ which is the decoder hidden state at time $t-1$, $\boldsymbol{h_j}$ is the encoder hidden state at time $j$.
2. Compute the weight: $\alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=0}^{T-1}\exp(e_{t,k})}$
3. Compute the attention output: $\boldsymbol{c_t} = \sum_{j=0}^{T-1} \alpha_{t,j}\boldsymbol{h_j}$
4. Concatenate the attention output with decoder hidden state before sending the decoder RNN, or concatenate the attention output with the decoder output (we use the latter method in lab 10).

According to the description above, please compute the attention output at time 2.

| $h_0$ | $h_1$ | $h_2$ | $h_3$ | $s_0$ | $s_1$ |
|---|---|---|---|---|---|
| $\begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}$ | $\begin{bmatrix}0.2\\0.3\\0.1\\0.4\end{bmatrix}$ | $\begin{bmatrix}0.1\\0.4\\0.3\\0.2\end{bmatrix}$ | $\begin{bmatrix}0.3\\0.4\\0.2\\0.1\end{bmatrix}$ | $\begin{bmatrix}0.4\\0.1\\0.2\\0.3\end{bmatrix}$ | $\begin{bmatrix}0.2\\0.4\\0.3\\0.1\end{bmatrix}$ |

$$e_{2,:} = \begin{bmatrix} 0.23 & 0.23 & 0.29 & 0.29 \end{bmatrix} \text{ (4 * 1pts)} \tag{1}$$

$$\alpha_{2,:} = \begin{bmatrix} 0.2425 & 0.2425 & 0.2575 & 0.2575 \end{bmatrix} \text{ (4 * 1pts)} \tag{2}$$

$$\boldsymbol{c_2} = \begin{bmatrix} 0.1758 \\ 0.3273 \\ 0.2258 \\ 0.2713 \end{bmatrix} \text{ (2pts)} \tag{3}$$

We consider the computational process if your final result is not correct due to the wrong intermediate computation result(s).

## Q10. Attention Variants (10 Points)

We have introduced the dot-product attention in the previous question, and there are other attention variants. Please write down the $e_{t,j}$ formulas according to the Keras code below.

| | | |
|---|---|---|
| Dot-Product | `weight = Activation('softmax')(`<br>`    dot([decoder_outputs[-1], encoder_output],`<br>`        axes=[2, 2]))` | $e_{t,j} = \boldsymbol{s_{t-1}^T h_j}$ |
| Multiplicative | `encoder_output_maped = Dense(units=2*hidden_dim,`<br>`                                bias=False)(encoder_output)`<br>`weight = Activation('softmax')(`<br>`    dot([decoder_outputs[-1], encoder_output_maped],`<br>`        axes=[2, 2]))` | |
| Additive | `decoder_output_maped = Dense(units=2*hidden_dim,`<br>`                                bias=False)(decoder_output)`<br>`encoder_output_maped = Dense(units=2*hidden_dim,`<br>`                                bias=False)(encoder_output)`<br>`tanh_result = Activation('tanh')(Add()([`<br>`    Lambda(lambda x: K.expand_dims(x, axis=2))(`<br>`        decoder_output_maped),`<br>`    Lambda(lambda x: K.expand_dims(x, axis=1))(`<br>`        encoder_output_maped)]))`<br>`weight = Activation('softmax')(`<br>`    Lambda(lambda x: K.squeeze(x, axis=3))(`<br>`        Dense(units=1, bias=False)(tanh_result)))` | |

$e_{t,j} = \boldsymbol{s_{t-1}^T W h_j}$ (5pts) $e_{t,j} = \boldsymbol{v^T} \tanh\left(\boldsymbol{W_s s_{t-1}} + \boldsymbol{W_h h_j}\right)$ (5pts)