Contents

(75pts) COS495 Midterm

Your name:

(15pts) Short answers

(2pts) What is the object being embedded (i.e. a vector representing this object is computed) when one uses

- the pair-pattern matrix?
- the word-context matrix?

(2pts) Bigrams are consecutive tokens. Let $x_1 =$ `dog bites man more` and $x_2 =$ `man bites dog less`. Write down the dense bigram indicator feature vectors for this dataset, label the dimensions.

(3pts) When faced with a big softmax over the vocabulary, training can be slow. Describe and explain the hierarchical softmax for speeding up training used in word2vec. Identify its parameters and explain the savings.

GloVe has objective $\sum_{ij} f(X_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$

- **(2pt)** What do the indices i, j represent and what is X_{ij}?

- **(3pt)** What is an issue with the alternative objective $\sum_{ij}(w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$?

(3pts) Let $z = Wa$, $W \in R^{m \times n}$,and consider the backward pass. Suppose we have i.i.d samples $\frac{dL}{dz_i} \sim N(0, 1)$ and $W_{ij} \sim N(0, \sigma^2)$. We would like $\text{Var}[\frac{dL}{da_i}] = 1$ by setting σ appropriately. Compute σ and show your steps.

(5pts) Unequal loss

Not all mistakes are created equal. We define $\text{cost}(y, y')$ as the cost when the label is y and the prediction is y'. Then the hinge loss is

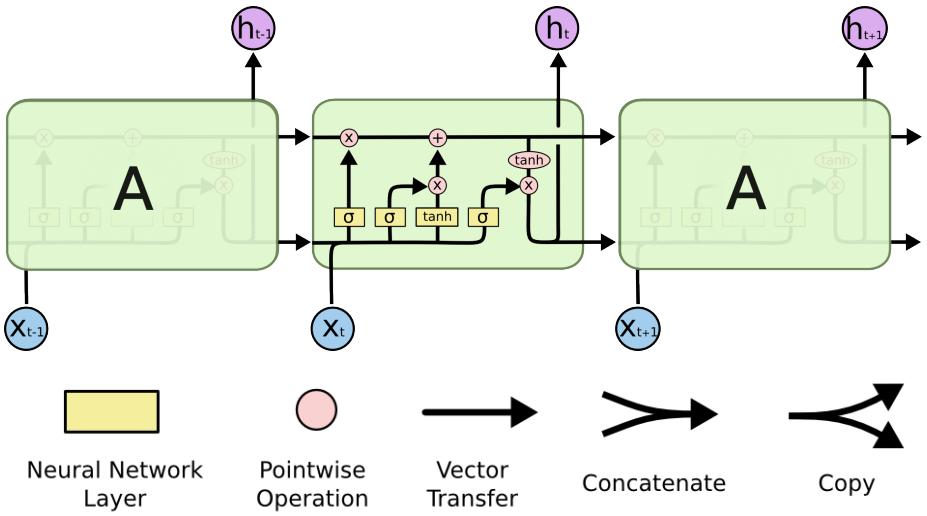
$$\max(0, \max_{y'} \text{cost}(y, y') + s_{y'} - s_y).$$

- **(3pts)** Show that this hinge loss is an upperbound of $\text{cost}(y, y')$.
- **(2pts)** Give the necessary condition on s_i for this upperbound to be tight and an example when it is loose.

(15pts) About LSTMs

Here are the LSTM equations and the corresponding illustration

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
\tilde{c}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$



(illustration from Chris Olah)

- **(5pt)** Label each of $f_t, i_t, o_t, c_t, \tilde{c}_t, c_{t-1}$ on the illustration. Make sure there is no ambiguity in your labeling.

Suppose we have a one-layer LSTM primitive $[h_1, \ldots, h_T] = \text{LSTM}([x_1, \ldots, x_T]) \in R^{h \times T}$.

- **(2pts)** Represent a two-layer LSTM using LSTM

- **(3pts)** Represent a bidirectional LSTM and use its hidden states to perform a sequence tagging task, where the tag at each timestep depends only on forward and backward hidden states at that time step. Clearly define your operations.

LSTM is supposed to alleviate the vanishing and exploding gradient problem. Let us consider a loss function that only depends on the final step $L(c_T)$ where T is the final step.

- **(5pts)** Express $\frac{dL}{dc_t}$ in terms of $\frac{dL}{dc_{t+1}}$, $\frac{dL}{dh_t}$, and non-recursive terms.

- **(5pts)** By analyzing your expression for $\frac{dL}{dc_1}$, explain why the long term dependency does not vanish when the forget gate is $f_t = 1$.

(25pts) Modular implementation of neural networks

Modules makes it easy to construct neural networks with various structures. Each module supports the `forward` and `backward` functions, might maintain internal state, and might be be composed of other modules. `Linear` and `Sequential` are two examples.

```
class Linear(Module):
    def __init__(self, W, b):
        self.W, self.b = W, b

    def forward(self, input):
        return np.dot(self.W, input) + self.b

    def backward(self, gradout):
        return np.dot(self.W.T, gradout)

class Sequential(Module):
    def __init__(self, children):
        self.children = children

    def forward(self, input):
        for child in self.children:
            input = child.forward(input)
        return input

    def backward(self, gradout):
        for child in reversed(self.children):
            gradout = child.backward(gradout)
        return gradout
```

(3pt) While `Linear` is computing the right outputs, it is missing a step that would enable learning in `Linear` itself. What step is missing? What else needs to be stored in order to perform this step?

(5pt) Implement the `Sigmoid` module which computes the element-wise function $\sigma(x) = \frac{1}{1+\exp(-x)}$

(5pt) Implement the `Dropout` module (space below)

```
# pseudo code or python code are both acceptable
# clarity and precision is required, being able to run is not required
class Sigmoid(Module):
    def forward(self, input):









    def backward(self, gradout):









#
class Dropout(Module):
        def __init__(self, p):
        self.p = p # with prob. p, the input stays
    def forward(self, input):









    def backward(self, gradout):
```

#

(7pt) Implement `EMul` that takes some children modules, give all children the same inputs, and outputs the elementwise product of the outputs of all children.

```
class EMul(Module):
    def __init__(self, children):
        self.children = children

    def forward(self, input):









    def backward(self, gradout):
```

#

(5pt) Use existing modules to implement the following network function by writing a single statement invoking constructors of `Sequential`, `EMul`, `Sigmoid`, and `Linear` given weights `W1, W2, W3`

$$
\begin{aligned}
a_1 &= \sigma(W_1 x) \\
a_2 &= \sigma(W_2 x) \\
z &= a_1 \odot a_2 \\
y &= W_3 z
\end{aligned}
$$

(15pts) Design an RNN

We would like an RNN to recognize a string of well-balanced parenthesis. Well balanced parenthesis are strings such as `(())` or `(())()` where

- the number of `(` is the same as the number of `)` in the whole string
- any prefix string does not contain more `)` than `(`.

Let $x = x_1, x_2, \ldots, x_T$ where $x_i \in R^{2\times 1}$ with $[0, 1]$ for `)` and $[1, 0]$ for `(`.
This RNN should perform the following updates

$$h_{i+1} = \text{relu}(Wh_i + Ux_{i+1})$$

In addition, we start with h_0 (dimension 3 with 0 initial value is recommended, but you are free to use anything you like). At the end, a prediction $y = Vh_T$ is made, where $y <= 0$ means balanced and $y > 0$ otherwise.

- **(7pts)** Clearly state your strategy for the RNN
- **(8pts)** Specify explicit values of W, U, V to achieve it