Homework 03: Neural Networks

CS 6316 Machine Learning

Due on Nov 14, 2023 11:59 PM

Name: Tongxuan Tian, **Computing ID:** nua3jz

1. **The backpropagation algorithm** (12 points) introduces the fundamental algorithm for training neural networks, called the backpropagation algorithm. Essentially, the calculation of backpropagation is an application of the chain rule in calculus. For example, if function $h(x)$ is composed by two simple function $g_1(\cdot)$ and $g_2(\cdot)$ as

$$h(x) = g_1(g_2(x)) \tag{1}$$

then, the gradient of $h(x)$ with respect to x is

$$\frac{dh}{dx} = \frac{dg_1(g_2(x))}{dg_2(x)} \frac{dg_2(x)}{dx}. \tag{2}$$

In this problem, let us consider a special neural network architecture, called *recurrent neural networks* (RNNs). RNNs are often used to model sequential data, such as word sequences in NLP or time series data. Along a sequence, a RNN is often specified by a recursive function as

$$h_t = \sigma(w_1 x_t + w_2 h_{t-1}) \tag{3}$$

where $\sigma(\cdot)$ is the Sigmoid function, w_1 and w_2 are the parameters of this RNN. h_t is called the *hidden state* at time stamp t, which is a function of the current input x_t and the previous hidden state h_{t-1}. As a simple starting point, we usually set $h_0 = 0$.

   (a) (2 points) Assume that we have a sequential data with total time stamp $T = 3$, please write down the explicit expression of h_1, h_2, and h_3. In other words, for each h_t, please only use the composition of the Sigmoid functions together with $\{x_1, \ldots, x_t\}$ and $\{w_1, w_2\}$.

   (b) (2 points) Based on the explicit expressions of $\{h_t\}_{t=1}^3$ in (a), please write down the derivative of h_t with respect to w_1,

$$\frac{dh_t}{dw_1}$$

   where $t = 1, 2, 3$.

   (c) (2 points) Without specify the value of t, please write down the *general* form of

$$\frac{dh_t}{dw_1},$$

   where t can be any positive integer.

   (d) (2 points) A typical way of using RNNs is to take the hidden states for some prediction tasks. For simplicity, we assume that for the following classification problem, we only use the last hidden state h_T. Then, the corresponding classifier is defined as

$$p(y \mid \boldsymbol{x}) = \sigma(y(w_o h_T + b)) \tag{4}$$

where w_o and b are parameters associated with this classifier, and $y \in \{-1, +1\}$.

In the case of stochastic gradient descent, given one training example $(\boldsymbol{x}, y)$, if we use the following loss function L, please compute the gradient of L with respect to w_1:

$$L(\theta) = -log(\sigma(y(w_o h_T + b))) \tag{5}$$

(e) (2 points) Now, let us consider the Sigmoid function $\sigma(r)$, based on its definition

$$\sigma(r) = \frac{1}{1 + \exp(-r)} \tag{6}$$

please show that, for $r \in \mathbb{R}$

$$0 < \frac{d\sigma(r)}{dr} < 0.5 \tag{7}$$

(f) (2 points) Combine (d) and (e), please explain the *vanishing gradient* problem in learning RNNs with a large T (say, $T \geq 50$). Your explanation needs to be as *specific* as possible. In other words, the explanation should be established on the results from (d) and (e).

Solution

(a)

$$\begin{aligned}
h_1 &= \sigma(w_1 x_1 + w_2 h_0) = \sigma(w_1 x_1) \\
h_2 &= \sigma(w_1 x_2 + w_2 h_1) = \sigma(w_1 x_2 + w_2(\sigma(w_1 x_1))) \\
h_3 &= \sigma(w_1 x_3 + w_2(\sigma(w_1 x_2 + w_2(\sigma(w_1 x_1)))))
\end{aligned}$$

(b) Since σ is a Sigmoid function, we have $\sigma'(x) = \sigma(x)(1 - \sigma(x))$, hence we can write the detrivative of each h_t with respect to w_1 as:

$$\begin{aligned}
\tfrac{dh_1}{dw_1} &= h_1(1 - h_1)x_1 = h_1 x_1(1 - h_1) \\
&= \sigma(w_1 x_1)(1 - \sigma(w_1 x_1))x_1 \\
\tfrac{dh_2}{dw_1} &= h_2(1 - h_2)(x_2 + w_2 x_1 h_1(1 - h_1)) \\
&= x_2 h_2(1 - h_2) + w_2 x_1 h_1 h_2(1 - h_1)(1 - h_2) \\
\tfrac{dh_3}{dw_1} &= h_3(1 - h_3)[x_3 + w_2 h_2(1 - h_2)(x_2 + w_2 h_1(1 - h_1))] \\
&= x_3 h_3(1 - h_3) + w_2 x_2 h_2 h_3(1 - h_2)(1 - h_3) + w_2^2 x_1 h_1 h_2 h_3(1 - h_1)(1 - h_2)(1 - h_3)
\end{aligned}$$

(c)

$$\begin{aligned}
\frac{dh_t}{dw_1} &= x_t h_t(1 - h_t) + w_2 x_{t-1} h_{t-1}(1 - h_{t-1})(1 - h_t) + ... + w_2^{t-1} x_1 \prod_{i=1}^{t} h_i(1 - h_i) \\
&= \sum_{i=1}^{t} (w_2^{t-i} x_i \prod_{k=i}^{t} h_k(1 - h_k))
\end{aligned}$$

(d) According to chain rule, we have:

$$\begin{aligned}
\frac{dL}{dw_1} &= \frac{dL}{d\sigma} \cdot \frac{d\sigma}{dz} \cdot \frac{dz}{dh_T} \cdot \frac{dh_T}{dw_1} \\
&= -\frac{1}{\sigma(y(w_0 h_T + b))} \cdot \sigma(z)(1 - \sigma(z)) \cdot w_0 y \cdot \frac{dh_T}{dw_1} \\
&= -y w_0 (1 - \sigma(y(w_0 hT + b))) \cdot \frac{dh_T}{dw_1}
\end{aligned}$$

(e)

$$\frac{d\sigma(r)}{dr} = \frac{e^{-r}}{(1+e^{-r})^2} = \frac{1}{\frac{1}{e^{-r}} + e^{-r} + 2}$$

Since $e^{-r} > 0$, we have $\frac{e^{-r}}{(1+e^{-r})^2} > 0$ and $\frac{1}{e^{-r}} + e^{-r} \geq 2$.
Hence, we have $\frac{1}{\frac{1}{e^{-r}}+e^{-r}+2} \leq 0.25 < 0.5$.
Finally, we have $0 < \frac{d\sigma(r)}{dr} < 0.5$

(f) Based on the gradient of L with respect to w_1 we get in (c), we can know that the gradient of σ is multiplied many times in the gradient of L. Since $0 < \frac{d\sigma(r)}{dr} < 0.5$, when $T > 50$, the gradient of some early input of the sequence will be very close to zero because $\frac{d\sigma(r)}{dr}$ will be multiplied many times in their coefficient. Hence, the gradient of early input of the sequence will have little effect on the final gradient of loss, in other words, the gradient of final loss does not contain much information about the initial gradient; the longer it is from the current time step, the less significant its feedback gradient signal will be, thus resulting in gradient vanishing and RNN fail to capture global information.

2. **Model selection** (8 points) Please create an iPython notebook file for answering the following questions. For this part,

   (a) for each code block, mark it clearly which question it is for

   (b) please make sure the code is execuatable with any modification, for example, importing necessary packages

   (c) **report the accuracy numbers in the pdf file**, and our TAs will re-run the code and compare with the numbers reported in the pdf file

   On piazza, along with this pdf file, there are five files in `data.tgz`

   - trnX-3d.csv, trnY-3d.csv: the input/output files for the training set
   - devX-3d.csv, devY-3d.csv: the input/output files for the development set

   Questions:

   (a) (3 points) Please use the MLPClassifier from sklearn with the default hyper-parameter setting for training.
   (https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html)
   For your convenience, you may want to use the accuracy function provided by sklearn.
   (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html)
   Report the accuracy on both the training and development sets in the pdf file.

   (b) (5 points) MLPClassifier provides about 20 hyper-parameters for tuning. In this question, we will limit to just a few of them.
   - hidden_layer_sizes: note that, this parameter will let you specify the number of layers and the hidden units in each layer
   - activation
   - solver: please only use SGD or Adam for hyper-parameter tuning
   - learning_rate_init
   - batch_size

   For these five hyper-parameters, you can choose any values for them as you wish (except the specified constraint for "solver"). The goal of hyper-parameter tuning is to find the best results on the development set. For your convenience, you may want to use the GridSearchCV function in the sklearn.

   Please report the following information in the PDF file

- the best development accuracy
- the corresponding hyper-parameters that produced them
- the difference between the best accuracy and the accuracy produced by the default hyper-parameters

Report

(a) Accuracy on training set: 82.51%
Accuracy on development set: 83.33%

(b) Best Accuracy on development set: 86.67%
Corresponding hyper-parameter:

- hidden_layer_sizes: (1000, 500)
- activation: ReLU
- solver: Adam
- learning_rate_init: 0.0005
- batch_size: 128

Accuracy difference:

- The best accuracy on the development set is 3.34% higher than the accuracy under default parameters.