NLP CS577 Homework 1

Tunazzina Islam, islam32@purdue.edu

March 2020

Q1

We use softmax as the logistic function for Logistic Regression.

Let assume, y_i is the probability predicted by the logistic regression which is a softmax result for class i.

$$y_i = softmax(z_i) = \frac{\exp(z_i)}{\sum_{i=1}^{n} \exp(z_i)}$$

where, $z = xw + b$ and $x =$ input feature vector (we use TF-IDF vector representation), $w =$ weight of corresponding feature, $b =$ bias.

Cross Entropy Loss has been used as loss function.

$$\text{Cross Entropy Loss,} L = -\sum_{i=1}^{n} t_i \log y_i,$$

where t_i is the true label;

Gradient:

We combine computation of the partial derivative of the cross-entropy loss function w.r.t output activation (i.e. softmax) together with the partial derivative of the output activation w.r.t. z which results in a short and clear implementation. The partial derivative will be following:

$$\frac{\partial L_i}{\partial z} = \frac{\partial L}{\partial y_i} . \frac{\partial y_i}{\partial z_i} = \sum y_i t_i - t_i(1 - y_i) = -t_i + t_i y_i + y_i \sum t_i = -t_i + y_i(t_i + \sum t_i) = y_i - t_i$$

The derivative of the loss with respect to z is

$$\frac{\partial L(i)}{\partial z} = y_i - t_i$$

The derivative of the last layer parameters are

$$\frac{\partial L(i)}{\partial w} = \frac{\partial z(i)}{\partial w} \frac{\partial L(i)}{\partial z_i} = x_i(y_i - t_i)$$

The gradient update for N training examples:

$$w = w - \text{learning\_rate} * \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial w}$$

Q2

Back Propagation:

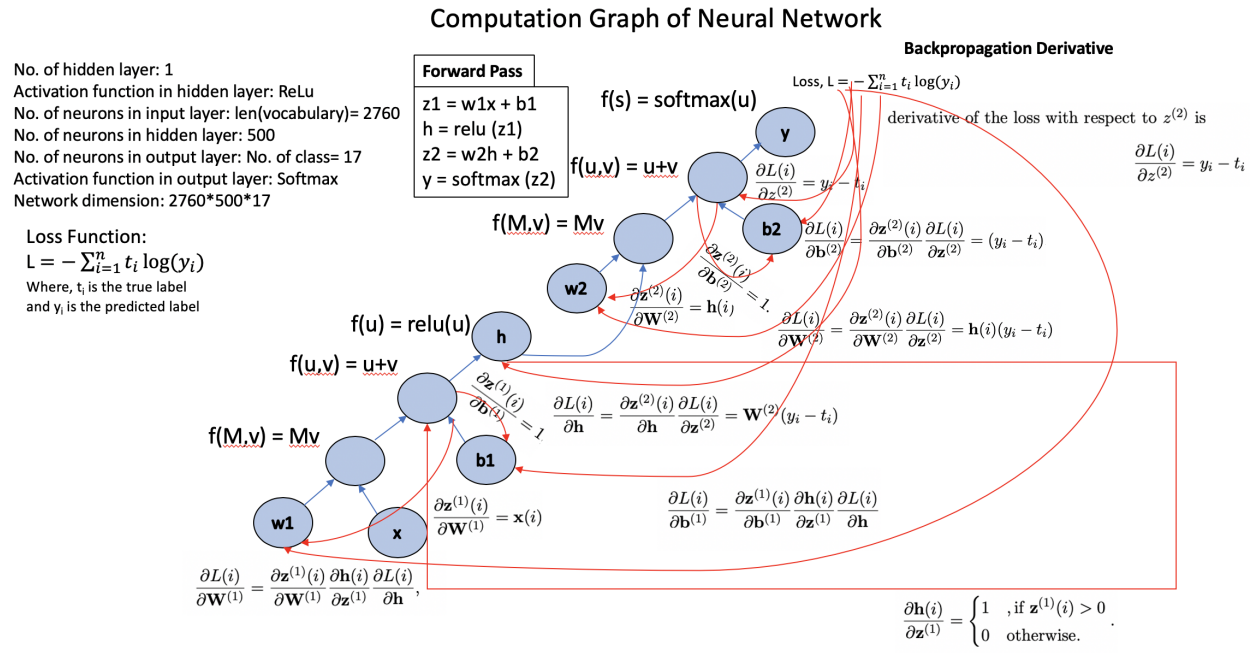

Figure 1: Computation Graph of Neural Network. Blue arrow for Forward computation and Red arrow for Back propagation computation.

Details of Backpropagation Derivative:

The derivative of the loss with respect to $z^{(2)}$ is

$$\frac{\partial L(i)}{\partial z^{(2)}} = y_i - t_i$$

It will compute the following derivatives for each training example: The derivative of the last layer parameters are

$$\frac{\partial L(i)}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{z}^{(2)}(i)}{\partial \mathbf{W}^{(2)}} \frac{\partial L(i)}{\partial \mathbf{z}^{(2)}} = \mathbf{h}(i)(y_i - t_i)$$

where

$$\frac{\partial \mathbf{z}^{(2)}(i)}{\partial \mathbf{W}^{(2)}} = \mathbf{h}(i),$$

and

$$\frac{\partial L(i)}{\partial \mathbf{b}^{(2)}} = \frac{\partial \mathbf{z}^{(2)}(i)}{\partial \mathbf{b}^{(2)}} \frac{\partial L(i)}{\partial \mathbf{z}^{(2)}} = (y_i - t_i),$$

as

$$\frac{\partial \mathbf{z}^{(2)}(i)}{\partial \mathbf{b}^{(2)}} = 1.$$

The derivatives with respect to the hidden values of the previous layer are

$$\frac{\partial L(i)}{\partial \mathbf{h}} = \frac{\partial \mathbf{z}^{(2)}(i)}{\partial \mathbf{h}} \frac{\partial L(i)}{\partial \mathbf{z}^{(2)}} = \mathbf{W}^{(2)}(y_i - t_i)$$

We then backpropagate $\partial L(i)/\partial \mathbf{h}$ to the previous layer. This will be needed in the final derivative the final derivatives are

$$\frac{\partial L(i)}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathbf{z}^{(1)}(i)}{\partial \mathbf{W}^{(1)}} \frac{\partial \mathbf{h}(i)}{\partial \mathbf{z}^{(1)}} \frac{\partial L(i)}{\partial \mathbf{h}},$$

and

$$\frac{\partial L(i)}{\partial \mathbf{b}^{(1)}} = \frac{\partial \mathbf{z}^{(1)}(i)}{\partial \mathbf{b}^{(1)}} \frac{\partial \mathbf{h}(i)}{\partial \mathbf{z}^{(1)}} \frac{\partial L(i)}{\partial \mathbf{h}},$$

where

$$\frac{\partial \mathbf{z}^{(1)}(i)}{\partial \mathbf{W}^{(1)}} = \mathbf{x}(i),$$

and

$$\frac{\partial \mathbf{z}^{(1)}(i)}{\partial \mathbf{b}^{(1)}} = 1,$$

and the derivative of the ReLU

$$\frac{\partial \mathbf{h}(i)}{\partial \mathbf{z}^{(1)}} = \begin{cases} 1 & \text{, if } \mathbf{z}^{(1)}(i) > 0 \\ 0 & \text{otherwise.} \end{cases}.$$

This function outputs

$$\mathrm{d}\mathbf{W}^{(1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial \mathbf{W}^{(1)}},$$

$$\mathrm{d}\mathbf{b}^{(1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial \mathbf{b}^{(1)}},$$

$$\mathrm{d}\mathbf{W}^{(2)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial \mathbf{W}^{(2)}},$$

and

$$\mathrm{d}\mathbf{b}^{(2)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial \mathbf{b}^{(2)}}.$$

Q3

Hyper-parameter Tuning:

For logistic Regression, we have following hyper-parameters:

no of epochs = 100

learning rate = 0.5

lambda = 0.1 (regularization parameter, we used L^2 regularization)

For Weight Initialization, we used *np.random.normal* which draws random samples from a normal (Gaussian) distribution with mean=0 and standard deviation = 0.1

We will explain one of the hyper-parameters(learning rate) tuning procedure here with learning curve:

We did 5-fold cross validation with 80% train and 20% validation data with different learning rates = 0.01, 0.05, 0.1, 0.5, 1.0 and plotted accuracy (Fig. 2) and macro F1 score (Fig. 3) both for training and validation data. For learning rate = 0.5, we obtained highest accuracy (train: 89.47%, validation: 41.3%) and macro F1 score (train: 86.11%, validation: 24.99%) both for train and validation data. So we chose learning rate = 0.5.

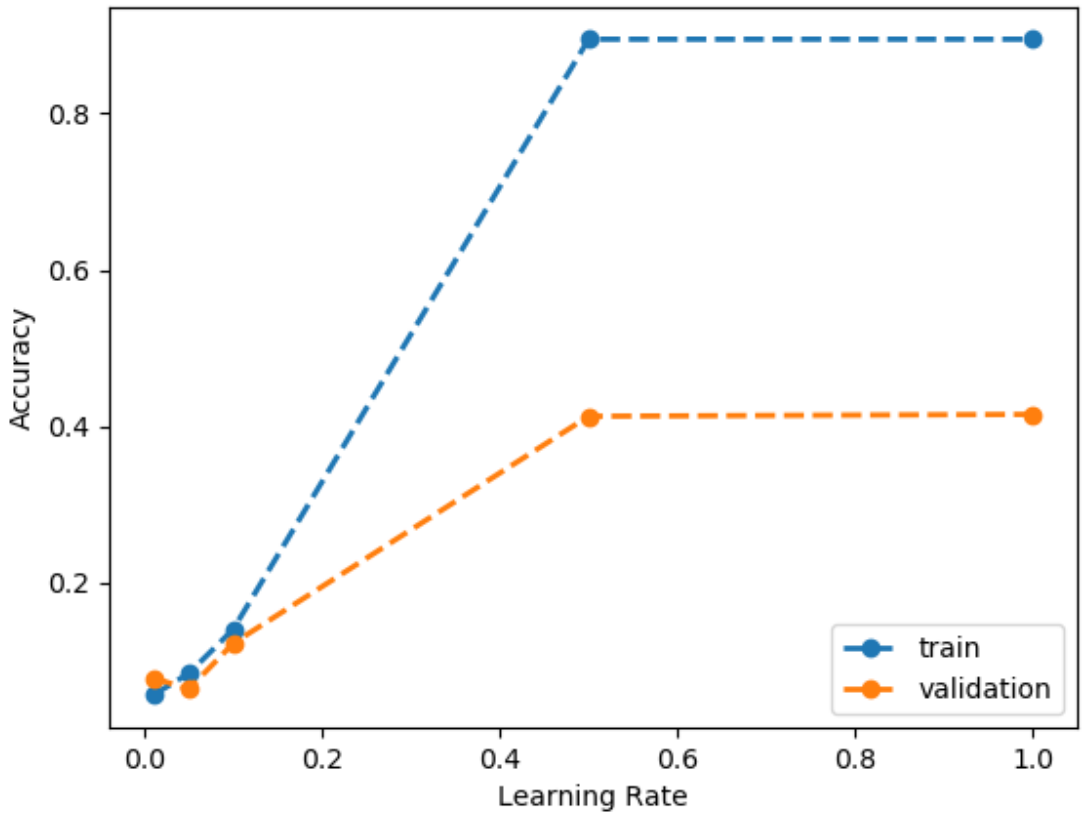

Figure 2: Learning curve Accuracy vs. Learning rate: Logistic Regression

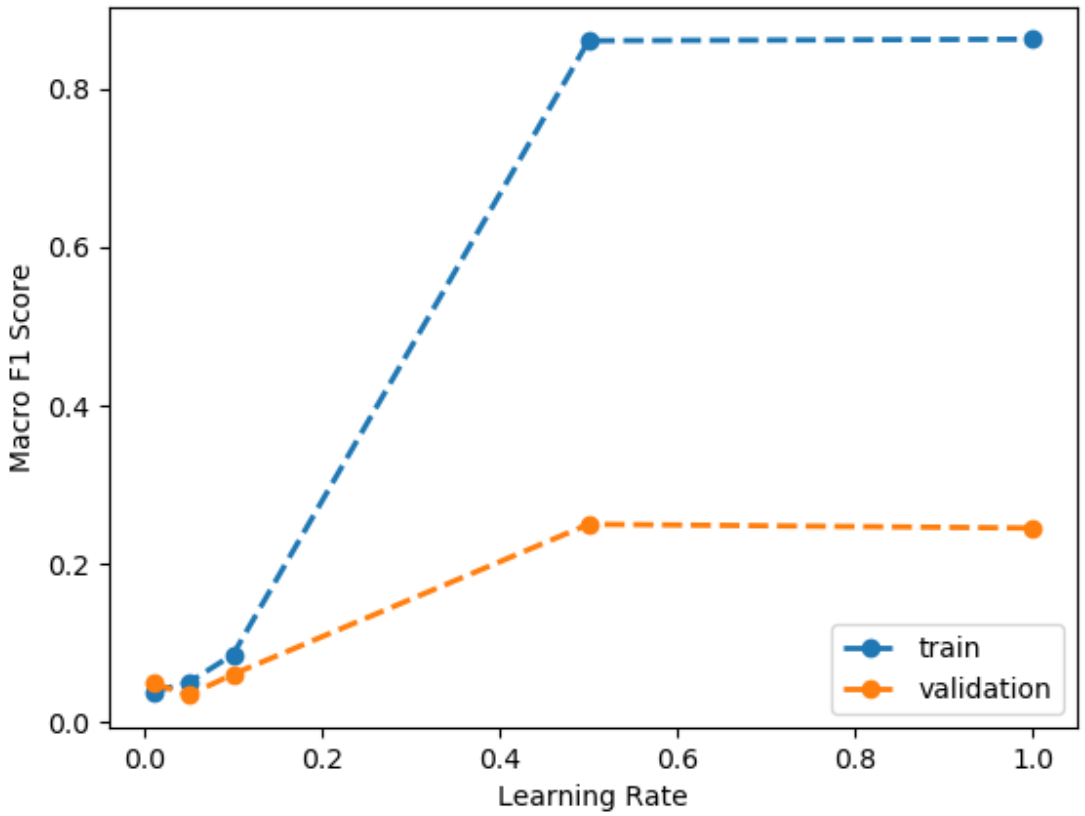

Figure 3: Learning curve Macro F1 Score vs. Learning rate: Logistic Regression

For Multi-layer Neural Network, we use 1 hidden layer with 500 hidden neurons with ReLU activation. Our network dimension is $2760 * 500 * 17$. We have following hyper-parameters:

no of epochs = 200

no of neurons in hidden layer = 500

learning rate = 1.0

_lambda = 0.0001 (regularization parameter, we used L^2 regularization)

For Weight Initialization, we used *np.random.normal* which draws random samples from a normal (Gaussian) distribution with mean=0 and standard deviation = 0.1

We will explain one of the hyper-parameters(learning rate) tuning procedure here with learning curve:

We did 5-fold cross validation with 80% train and 20% validation data with different learning rates = 0.01, 0.05, 0.1, 0.5, 1.0 and plotted accuracy (Fig. 4) and macro F1 score (Fig. 5) both for training and validation data. For learning rate = 1.0, we obtained highest accuracy (train: 99.79%, validation: 41.7%) and macro F1 score (train: 99.83%, validation: 29.29%) both for train and validation data. So we chose learning rate = 1.0 for neural network.

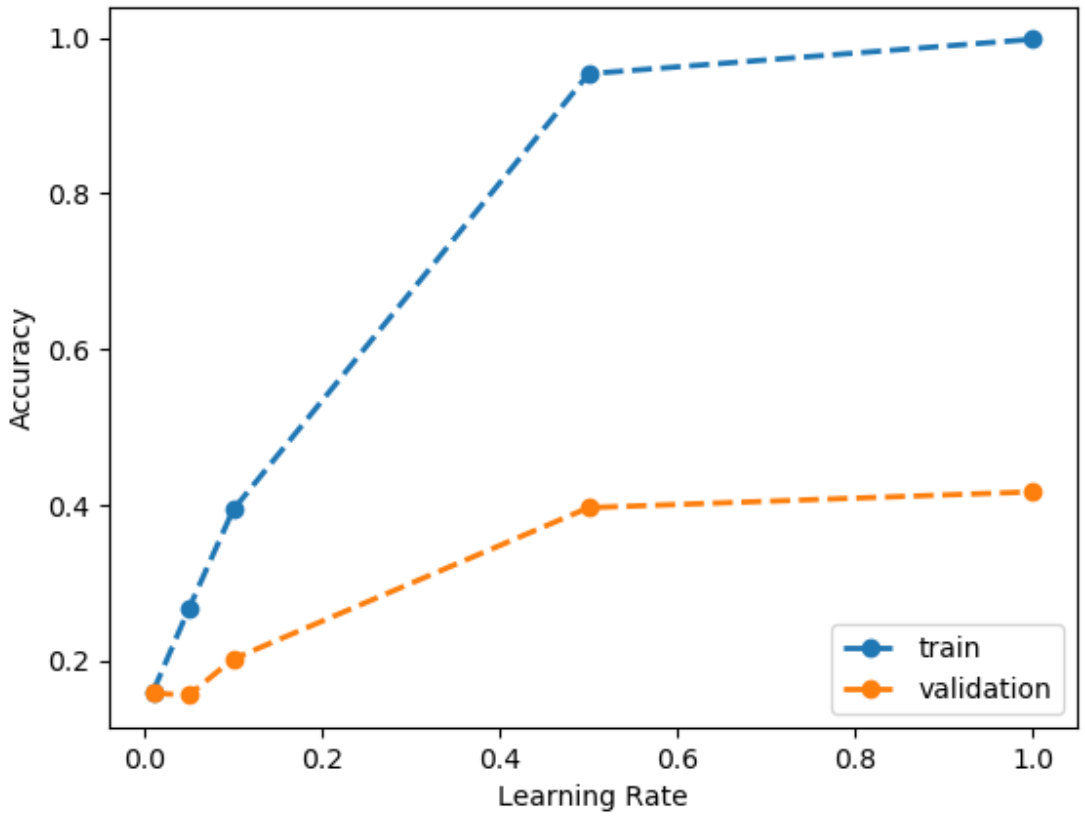

Figure 4: Learning curve Accuracy vs. Learning rate: Neural Network

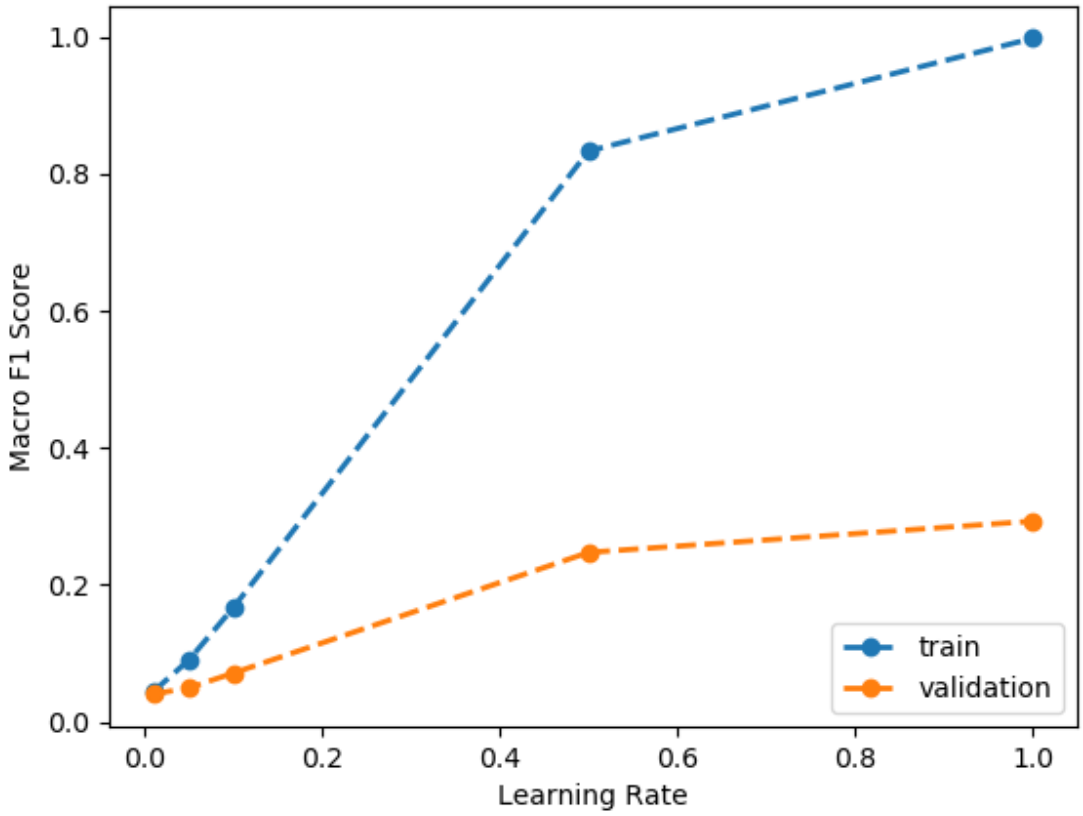

Figure 5: Learning curve Macro F1 Score vs. Learning rate: Neural Network

Q4

Framing: Framing is a political strategy in which politicians present their statements in order to control public perception of issues. In the social sciences, framing comprises a set of concepts and theoretical perspectives on how individuals, groups, and societies, organize, perceive, and communicate about reality. We can do real world analysis of framing patterns over time on different dataset like: vaccination related tweets and health providers can leverage this analysis to know people's sentiment and perspective.