Assignment 2

Team: Naveksha and Billy

Part 2 Answers

1. **Cross entropy loss function**

$$L(W) = -\sum_{n=1}^{N} y\,log(y') + (1-y)log(1-y') \text{ where } y' = \sigma(Wx)$$

During training y is either 1 or 0 so one of the two terms will become 0.

Derivative of cross entropy loss w.r.t W

$$\frac{d(L(w))}{dW} = -\frac{d(ylog(y'))}{dW} - \frac{d((1-y)log(1-y'))}{dW}$$

$$\frac{d(L(w))}{dW} = -y\frac{d(log(y'))}{dW} - (1-y)\frac{d(log(1-y'))}{dW}$$

Using chain rule,

$$\frac{d(L(w))}{dW} = \left(-\frac{y}{y'} - \frac{(1-y)}{(1-y')}\right)\frac{dy'}{dW}$$

$$\frac{d(L(w))}{dW} = \left(-\frac{y}{y'} - \frac{(1-y)}{(1-y')}\right)\frac{d(\sigma(Wx))}{dW}$$

Taking derivative of sigmoid function,

$$\frac{d(L(w))}{dW} = \left(-\frac{y}{y'} - \frac{(1-y)}{(1-y')}\right)\sigma(Wx)(1-\sigma(Wx))\frac{d(Wx)}{dW}$$

Simplify

$$\frac{d(L(w))}{dW} = -1 * \left(\frac{y(1-y')+y'(1-y)}{y'(1-y')}\right)y'(1-y')x$$

$$\frac{d(L(w))}{dW} = -1 * (y(1-y') + y'(1-y)) * x$$

$$\frac{d(L(w))}{dW} = (y(y'-1) - y'(y-1))) * x$$

$$\frac{d(L(w))}{dW} = (yy' - y - yy' + y') * x$$

$$\frac{d(L(w))}{dW} = (y'-y) * x$$

2. **Gradient of output loss w.r.t W1**

$$\frac{d(L(W))}{dW1} = \frac{d}{dW1}(\sigma(W2x') - y) * x'$$

Rule of derivative of terms in multiplication

$$\frac{d(L(W))}{dW1} = (\sigma(W2x') - y) * \frac{d}{dW1}x' + \frac{d}{dW1}(\sigma(W2x') - y) * x'$$

Where x' is output from the hidden layer, which is input to the last layer.

$$x' = \sigma(W1x)$$

Taking Gradient w.r.t W2

$$\frac{d^2(L(w))}{dW1dW2} = \frac{d}{dW2}(((\sigma(W2x') - y) * \frac{d}{dW1}x') + \frac{d}{dW2}(\frac{d}{dW1}(\sigma(W2x') - y) * x')$$

Simplify using rule of multiplication and chain rule

$$\frac{d^2(L(w))}{dW1dW2} = ((\sigma(W2x') - y)\frac{d}{dW2}(\frac{d}{dW1}x') + \frac{d}{dW1}x'\frac{d}{dW2}((\sigma(W2x') - y) + \frac{d}{dW2}(\frac{d}{dW1}(\sigma(W2x') - y) * x')$$

Solving a small part of equation

$$\frac{d}{dW2}(\frac{d}{dW1}x') = \frac{d}{dW2}(\sigma(W1x) * (1 - \sigma(W1x) * x) = 0$$

$$\frac{d^2(L(w))}{dW1dW2} = (\frac{d}{dW1}x'\frac{d}{dW2}((\sigma(W2x') - y) + \frac{d}{dW2}(\frac{d}{dW1}(\sigma(W2x') - y) * x')$$

$$\frac{d^2(L(w))}{dW1dW2} = (\frac{d}{dW1}x'\frac{d}{dW2}((\sigma(W2x') - y) + \frac{d}{dW1}(\sigma(W2x') - y) * \frac{d}{dW2}(x') + x'\frac{d}{dW2}(\frac{d}{dW1}(\sigma(W2x') - y))$$

$$\frac{d^2(L(w))}{dW1dW2} = (\frac{d}{dW1}x'\frac{d}{dW2}((\sigma(W2x') - y) + x'\frac{d^2}{dW1dW2}(\sigma(W2x') - y))$$

Solving a small part of equation

$$\frac{d}{dW1}x' = \frac{d}{dW1}(\sigma(W1x)) = \sigma(W1x)(1 - \sigma(W1x))x = x'x - x'^2x$$

$$\frac{d}{dW2}(\sigma(W2x') - y) = \sigma(W2x')(1 - \sigma(W2x'))x' = y'x' - y'^2x'$$

$$\frac{d^2}{dW1dW2}(\sigma(W2x') - y)) = \frac{d}{dW1}(\sigma(W2x')(1 - \sigma(W2x')))$$

$$\frac{d^2}{dW1dW2}(\sigma(W2x') - y)) = \sigma(W2x')\frac{d}{dW1}(1 - \sigma(W2x')) + (1 - \sigma(W2x'))\frac{d}{dW1}\sigma(W2x')$$

Further solving a smaller part of equation

$$\sigma(W2x')\frac{d}{dW1}(1 - \sigma(W2x')) = -\sigma(W2x') * \frac{d}{dW1}(\sigma(W2x'))$$

$$\frac{d^2}{dW1dW2}(\sigma(W2x') - y)) = -\sigma(W2x')\frac{d}{dW1}(\sigma(W2x')) + \frac{d}{dW1}\sigma(W2x') - \sigma(W2x')\frac{d}{dW1}\sigma(W2x')$$

$$\frac{d^2}{dW1dW2}(\sigma(W2x') - y)) = -2\sigma(W2x')\frac{d}{dW1}(\sigma(W2x')) + \frac{d}{dW1}\sigma(W2x')$$

$$\frac{d^2}{dW1dW2}(\sigma(W2x') - y)) = (1- 2y')\frac{d}{dW1}(\sigma(W2x'))$$

Putting the small parts back

$$\frac{d^2(L(w))}{dW1dW2} = x'(y' - y)$$

3. **Generalize the equation for h hidden layers**

$$\frac{d^h(L(w))}{dW1dWh} = x'_{h-1}(x'_h - x_h)$$

4. **Optimization on vanilla gradient descent**

4.1 Add regularization: (L2 regularization: add a function of square of weights to the loss function)

4.2 Learning rate annealing: Reduce the learning rate with increasing epochs.

Results

1. **Results for random classifier, Naive Bayes', Logistic regression, Vanilla NN and 2 versions of optimized NN(s)**

   Dataset: Disease.csv

   The data was normalized and downsampled.

```
(venv) navekshasood@Navekshas-MacBook-Pro MLCL % python script_classify
.py
Split 768 rows into train=426 and test=110 rows.
Running learner = Random
Accuracy for Random: 48.18181818181818
Running learner = Naive Bayes
Accuracy for Naive Bayes: 73.63636363636363
Running learner = Logistic Regression
Accuracy for Logistic Regression: 50.0
Running learner = Neural Network
Accuracy for Neural Network: 61.81818181818181
Running learner = Neural Network with LR Annealling
Accuracy for Neural Network with LR Annealling: 58.18181818181818
Running learner = Neural Network with Regularization
Accuracy for Neural Network with Regularization: 75.45454545454545
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 70.0
```

   Results show that Naive Bayes' and Neural Network perform best on this dataset. Regularization was able to improve the performance of neural network further, but learning rate annealing did not help that much. Logistic regression performed quite poorly on this dataset, which may mean that this data was not linearly separable and a logistic regressor may not be able to segregate it correctly.

   Dataset: IMDB.csv

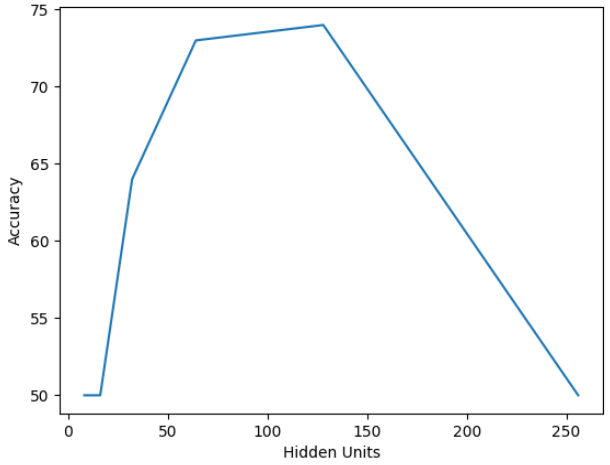
   Max_features = 300

```
(venv) navekshasood@Navekshas-MacBook-Pro MLCL % python script_classify
.py
Split 50000 rows into train=40000 and test=10000 rows
Running learner = Random
Accuracy for Random: 49.74
Running learner = Naive Bayes
Accuracy for Naive Bayes: 76.85
Running learner = Logistic Regression
Accuracy for Logistic Regression: 75.9
Running learner = Neural Network
Accuracy for Neural Network: 81.0
Running learner = Neural Network with LR Annealling
Accuracy for Neural Network with LR Annealling: 81.17
Running learner = Neural Network with Regularization
Accuracy for Neural Network with Regularization: 79.63
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 80.55
```

Results show that Naive Bayes', logistic regression as well as neural network performed pretty well on this dataset. It goes on to show that this was a linearly separable dataset with clearer decision boundary. Here, LR annealing marginally helped neural network perform better, but regularization did not. Because this dataset was manifold larger that disease data, it took relatively longer to run the algorithms on this dataset (approximately 5 minutes for 7 algorithms). If you'd like to run faster, consider using a lesser number of max_features in the CountVectorizer function; slight degradation in performance is expected due to loss of information.

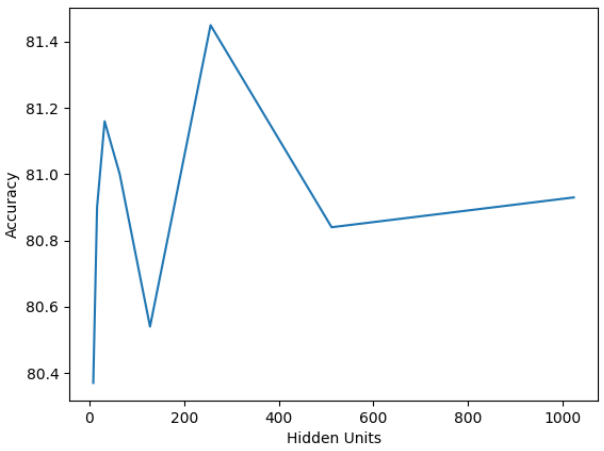
2. **Experiments with hidden neurons**
   Dataset: Disease.csv
   Neurons = [8, 16, 32, 64, 128, 256]

```
(venv) navekshasood@Navekshas-MacBook-Pro MLCL % python script_classify.py
Split 768 rows into train=436 and test=100 rows.
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 50.0
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 50.0
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 64.0
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 73.0
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 74.0
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 50.0
```



Dataset: IMDB.csv
Neurons = [8, 16, 32, 64, 128, 256, 512, 1024]

```
(venv) navekshasood@Navekshas-MacBook-Pro MLCL % python script_classify.py
Split 50000 rows into train=40000 and test=10000 rows
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 80.36999999999999
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 80.9
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 81.16
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 81.0
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 80.54
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 81.45
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 80.84
Running learner = Neural Network with LR Annealling & Regularization
Accuracy for Neural Network with LR Annealling & Regularization: 80.93
```



The plots depict that the accuracy keeps increasing as we increase the number of hidden neurons or the capacity of the neural net until such time that it starts overfitting and the test accuracy starts declining. As per the experiments, the ideal number of neurons is

128 for the disease as well as the IMDB dataset. However, the disease dataset saw much variation in accuracy across the number of hidden neurons, whereas for the IMDB dataset it was in a tighter range.

Note: We started working on the IMDB dataset before it was posted on canvas, and hence we are working on a csv file we found online. We have checked and made sure that the data is the same.

References:

1. https://www.dropbox.com/s/rxrtz3auu845fuy/Softmax.pdf?dl=0
2. https://prvnk10.medium.com/sigmoid-neuron-and-cross-entropy-962e7ad090d1\
3. https://jmlb.github.io/ml/2017/12/26/Calculate_Gradient_Softmax/
4. https://towardsdatascience.com/improving-vanilla-gradient-descent-f9d91031ab1d