Homework 2

Submitted by Vishantan Kumar
March 3, 2024

Part 1: Short Answers

Ans 1. Perceptrons learn a linear decision Boundary, separating data by a line, plane or hyperplane.

Ans 2. For a C-Class classification we can use a One-vs-Rest strategy, training C perceptrons and separating classes as - Belongs to class C / Doesn't belong to class C. When classifiying a test - point, we calculate the relative probability of it belonging to each of the C classes, and label it as the class with the highest probability (or confidence score).

Ans 3. Stochastic Gradient Descent (SGD) is an optimization algorithm that is slight modification of the Gradient descent algorithm. In gradient descent, the objective function is optimized by updating the parameters in a direction opposite to that of the gradient at that point. In SGD, the same principle is followed, however, instead of looking at the entire dataset for calculating the gradient, a single sample or a smaller subset is chosen at a time. This makes SGD much faster than traditional gradient descent but introduces a lot of noise due to the different subsets chosen at each iteration.

Ans 4. Mini-batch GD is a hybrid of SGD and Batch-GD, wherein the dataset is split into smaller subsets, not just for the calculation of gradients but for parameter optimization. This differs from batch since it uses smaller subset instead of the single subset used in Batch-GD. For instance, for a dataset with 1000 samples, and a batch size of 50, the mini-batch GD splits it into 20 batches and hence updates the parameters 20 times - once for each batch.

Ans 5. Given:

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

Let $1 + e^{-a} = f(a)$, therefore

$$\sigma(f) = \frac{1}{f}$$
$$\sigma'(a) = \sigma'(f(a)).f'(a) = \frac{-1}{f^2}.(-1).e^{-a}$$
$$\sigma'(a) = \frac{e^{-a}}{(1+e^{-a})^2} = (\frac{1}{(1+e^{-a})}).(1 - \frac{1}{(1+e^{-a})})$$

hence,

$$\sigma'(a) = \sigma(a).(1 - \sigma(a))$$

Ans 6. An epoch refers to a complete pass over the dataset. In one epoch, the algorithm passes atleast once through the entire dataset, updating the parameters at each iteration. The number of epochs is predefined while the number of iterations are dependant on number of data-points and the batch size. After all iterations of updation are completed, one epoch is marked complete.

Ans 7. Number of iterations = Size of dataset/Batch size. Dataset = 2000, Batch size = 50. Therefore, number of iterations = $2000/50 = 40$ iterations.

Ans 8. Dropout regularization is the method of deactivating neurons in a neural layer by setting the input/output to 0 with a predefined probability called the dropout rate. This is used to prevent overfitting, improve generalisation, and provide robustness to the model.

Ans 9. Early stopping is a regularisation method for Neural Networks wherein the performance on the validation is monitored at each epoch and if the performance stops improving significantly, the training is stopped. This has the advantage of preventing the model from overfitting to the training set which can lead to poor generalisation on the test sets.

Ans 10. Regularization is primarily used to reduce model overfitting which happens when the network starts modelling the noise and outliers in the data, decreasing it's performance on unseen test data-sets. Regularization improves generalisation on datapoints, minimizing the probability of bias towards a certain data feature. There are various methods of regularization including - L1/L2, Early stopping, Dropout etc.

Ans 11. There are multiple ways to reduce overfitting:

- Increase size of training set - fresh samples or data augmentation
- Using cross-validation along with early stopping
- Introducing regularization in different stages - L1/L2 or Dropout etc.
- Reducing feature complexity by dimensionality reduction methods such as LDA or PCA (if applicable)

Ans 12. Batch normalisation is a technique that normalizes the inputs to each layer within a neural network. The idea is to minimize the 'internal covariance shift', which refers to the change in distribution of each layers' input as the parameters of the proceeding layer change. Batch-norm normalizes the inputs to each layer such that they lie

on a $\mathcal{N}(0,1)$ distribution. This mitigates the effects of covariance shift, increasing the training speed and making it more stable. This also has a regularization effect wherein it prevents overfitting and hence can reduce the need for specialised regularisation techniques.

Part 2: Perceptron

2.1 2-way classifications

The following plot showcases the individual classification accuracies for each of the 10 2-way classifications. The average classification accuracy was found to be 95% with a maximum of 98.4% for digit 0.

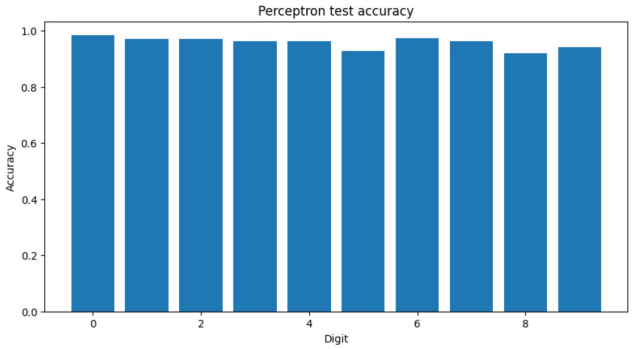

Figure 1. Perceptron accuracies for each digit

2.2 Overall classification

The overall classification accuracy for a combination of the 10 perceptron models was found to be around 85% which is lower than the individual classification accuracies, but this can be due to the fact that even if a digit is correctly classified as **not belonging** to class **k**, it might still be misclassified to another class **m** such that **m** $\neq$ **k** but **m** is not the correct label.

Part 3: Logistic Regression

3.1 Gradient Computations

To show:

$$\frac{\partial J(w)}{\partial w_j} = \sum_{i=1}^{N} x_j^{(i)} \left(g_w \left(x^{(i)} \right) - y^{(i)} \right)$$

We know that $J(w)$ is given by:

$$-\sum_{i=1}^{N} \left(y^{(i)} \log(g_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - g_w(x^{(i)})) \right)$$

where,

$$g_w(x) = \sigma(w^T x)$$

$$g'_w(x) = \frac{x e^{-w^T x}}{\left(1+e^{-w^T x}\right)^2} = x.\sigma(w^T x) \left(1 - \sigma(w^T x)\right)$$

using Chain rule:

$$\frac{-\partial J(w)}{\partial w_j} = \sum_{i=1}^{N} \left(y^{(i)} \frac{g'_w(x_j^{(i)})}{g_w(x_j^{(i)})} + (1 - y^{(i)}) . \frac{-g'_w(x_j^{(i)})}{1 - g_w(x_j^{(i)})} \right)$$

Substituting the value of $g'_w(x)$ from the previous equation:

$$\frac{-\partial J(w)}{\partial w_j} = \sum_{i=1}^{N} \left(y^{(i)} \frac{x\sigma(1-\sigma)}{\sigma} + (1 - y^{(i)}) \frac{-x\sigma(1-\sigma)}{1-\sigma} \right)$$

$$= \sum_{i=1}^{N} \left(y^{(i)} x_j^{(i)} (1 - \sigma) + (1 - y^{(i)})(-x_j^{(i)} \sigma) \right)$$

$$= \sum_{i=1}^{N} \left(y^{(i)} x_j^{(i)} - y^{(i)} x_j^{(i)} \sigma + -x_j^{(i)} \sigma + y^{(i)} x_j^{(i)} \sigma \right)$$

Substituting for σ:

$$\frac{\partial J(w)}{\partial w_j} = \sum_{i=1}^{N} \left(g_w(x^{(i)}) x_j^{(i)} - y^{(i)} x_j^{(i)} \right)$$

$$\frac{\partial J(w)}{\partial w_j} = \sum_{i=1}^{N} \left(x_j^{(i)} (g_w(x^{(i)}) - y^{(i)}) \right)$$

We hence prove the required expression

3.2 Gradient Descent

Gradient descent based Logistic regression was run on all 10 samples for 6 different learning rates - $[1, 0.5, 0.1, 0.05, 0.01, 0.001]$ to compare the performance. The following figure shows the performance for each of the learning rates-

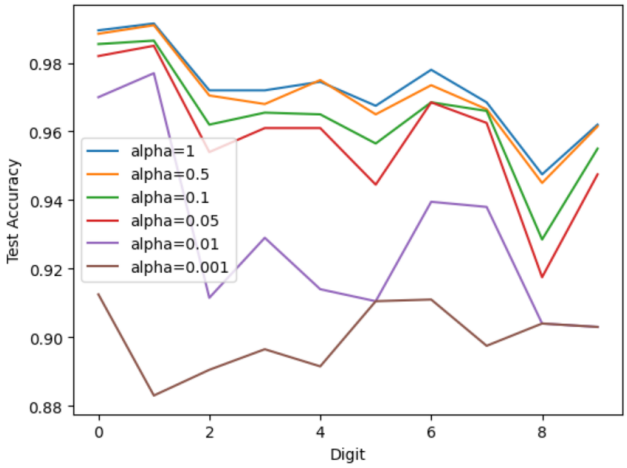

Figure 2. Logistic Regression accuracies

The best performance was observed on Learning rate of 1 and 0.5 with accuracies in the range of $96 - 99\%$ which are pretty high learning rates. while learning rate of 0.001 showed the poorest performance with and average accuracy of 90%. Accuracies were also seen to be positively correlated to the learning rate, i.e., a higher learning rate led to a better accuracy. Each training was performed for 300 iterations.

3.3 Overall classification

The overall classification using the 10 logit models was run for 300 iterations on the following learning rates - $[1, 0.5, 0.1, 0.05, 0.01]$. The results for the same can be observed in 3

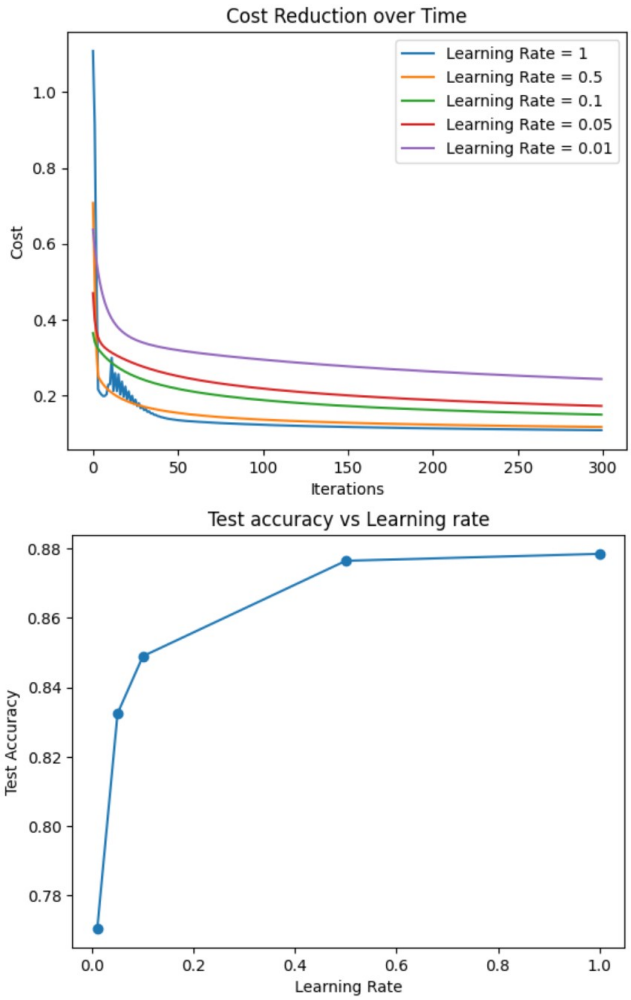

Figure 3. Overall model cost and Accuracy

The cost function is seen to be monotonically decreasing for each learning rate, however, for $\alpha = 1$ the cost function is seen to be noisy for the initial iterations, smoothing out as the training progresses. The cost is also lowest for $\alpha = 1$ and increases as the learning rate decreases.

The average accuracy is seen to follow a similar trend, being around 87% for $\alpha = 1$ and decreasing to around 77% for lower learning rates such as $\alpha = 0.01$.

Part 4: Results

We observed smooth cost functions and good classifications accuracies for learning rates as high as 0.5 and 1. This is because the logistic regression function is strictly convex under our assumptions, hence it hardly ever overshoots. An overall classification accuracy $> 85\%$ is fairly good for a non-complicated model like logistic regression, however, it can be made better by using other classifiers such as Neural networks, adding regularisation features such as L_1 / L_2 or early stopping.

The accuracies for the perceptrons were also fairly good, with a $92\%+$ accuracy on each individual classifier and an 85% accuracy on the group classification. This is good since a Single Perceptron is a linear classifier and the data features are not strictly linear.