cs577 Assignment 3 - Report

Girish Rajani-Bathija
A20503736
Department of Computer Science
Illinois Institute of Technology
October 25, 2022

Problem Statement

Select a classification and regression problem from the UCI repository, load the dataset, clean and vectorize it and split it into train/validation/test sets. Evaluate different loss functions on the dataset. Evaluation/Comparison using different optimizers. Lastly, evaluation needs to be done using different regularization methods. Summarize the results and draw conclusions.

Proposed Solution

The proposed solution is to perform classification on the Iris dataset and regression on the crime dataset. It will be cleaned, normalized, and vectorized for training. Various losses, optimizers, and regularization methods will be evaluated for both problems. Appropriate conclusions will be drawn from the analysis/evaluation conducted.

Implementation Details

When preprocessing the iris dataset, an error was encountered when vectorizing the label. Just using categorical encoding was not enough to vectorize the iris data labels. In doing so, the following error occurred.

“invalid literal for int() with base 10: 'Iris-versicolor' ”

This meant that first the labels must be encoded into integers which were done using LabelEncoder from sklearn and then it was one hot encoded.

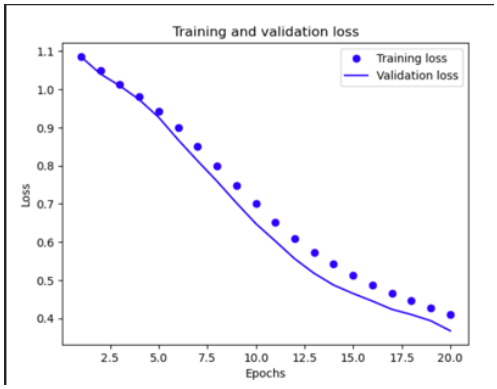
Additionally, when evaluating using the sparse categorical crossentropy loss function, the previous evaluation had been done using categorical crossentropy, so the training labels were already one-hot encoded. This resulted in some errors since I had used sparse categorical crossentropy while the labels were one-hot encoded. This type of loss requires the data to be in the form of integers and not 0’s and 1’s, so to fix this, I had to remove the vectorized labels.

Results and Discussion

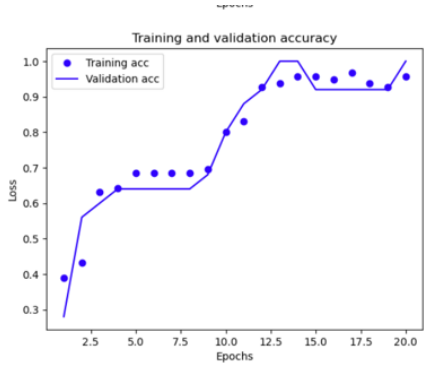
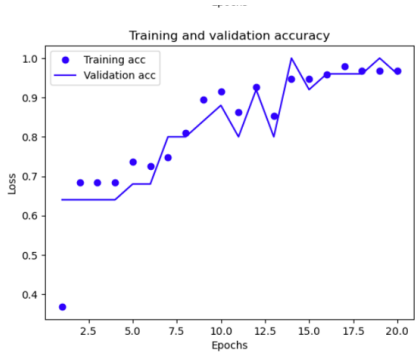
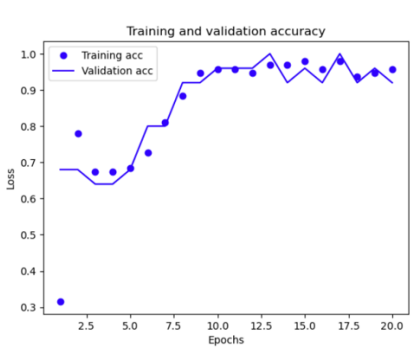
Classification Example - Iris Dataset:

When building the model, various layers and hyperparameters were used to classify the iris dataset. Both of the hidden layers contained a dense layer of 64 units with a relu activation. The output layer used a softmax activation function with a 3-unit output size since the sample belonged to one of three classes. The data was also split into training and testing data, whereby 20% of the dataset was test data, and 80% was training data(30 samples for testing and 120 for training). From that training data, it was further split into training/validation sets (25 samples for validation and 105 for training).

When evaluating loss functions on the iris dataset, the appropriate loss functions for a multiclass classification problem are categorical crossentropy, sparse categorical crossentropy, and kullback leibler. It is important to note that throughout this assignment, there was very little to no overfitting occurring within the training and validation loss. Example:

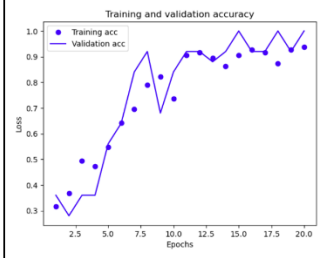
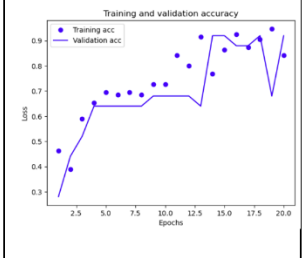
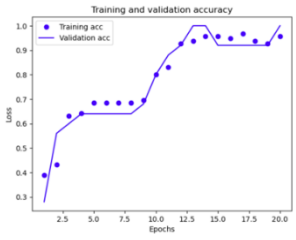


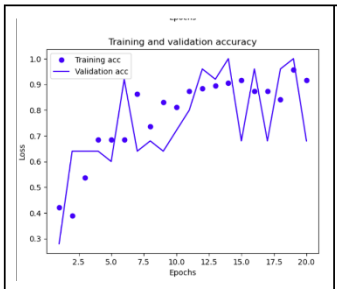
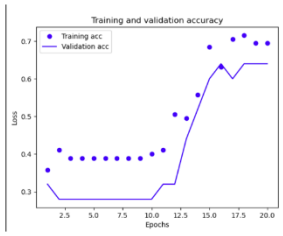
All of the other evaluations followed a very similar pattern, so instead, analysis was done on training/validation accuracy, and so for the purpose of not repeating similar information, I will only be showing the training/validation accuracy during analysis.

		
Categorical Crossentropy	Sparse Categorical Crossentropy	Kullback Leibler Divergence

The above table shows the training and validation accuracy of the training using the 3 previously mentioned losses. It is important to note that all 3 losses were trained on the same model architecture, 20 epochs, and a batch size of 10 units. This configuration seemed to generate optimal results. We can see from the results above that although all 3 losses reach a significantly high accuracy in the 90's, they all overfit at some point. Categorical crossentropy model overfits around 14 epochs, sparse categorical crossentropy starts to overfit around 10 epochs, and KL divergence starts to overfit relatively early. Both categorical crossentropy and KL divergence were used on vectorized labels, and then sparse categorical crossentropy was used on integer labels. Based on the results obtained above, categorical cross entropy performed the best. Therefore, categorical cross entropy was used throughout, when evaluating various optimizers, which will be analyzed below.

When evaluating different optimizers, the following were used: SGD, RMSprop, Adam, Adadelta, Adagrad, and Adamax.

		
SGD	RMSprop	Adam
(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)	(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)	(learning_rate=0.001, rho=0.9, epsilon=None, decay=0.0)

		
Adadelta	Adagrad	Adamax
(learning_rate=1.0, rho=0.95, epsilon=None, decay=0.0)	(learning_rate=0.01, epsilon=None, decay=0.0)	(learning_rate=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0)

	SGD	RMSprop	Adam	Adadelta	Adagrad	Adamax
Epochs	2	14	15	5	2	7

As shown above, the Adam optimizer took the longest to converge without overfitting, which meant that it generalized the best when compared to the other optimizers. Now that we know Categorical Crossentropy loss and Adam optimizer gave the best results together, we need to minimize overfitting by introducing regularization.

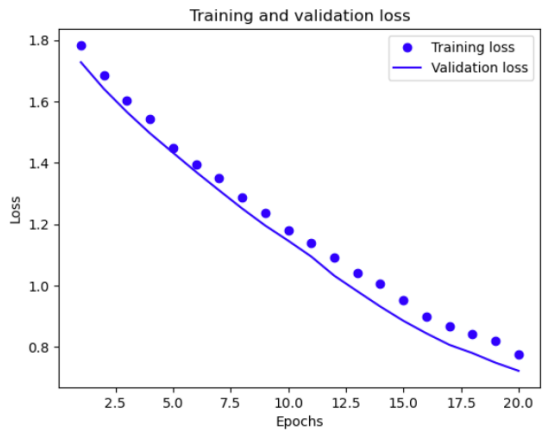
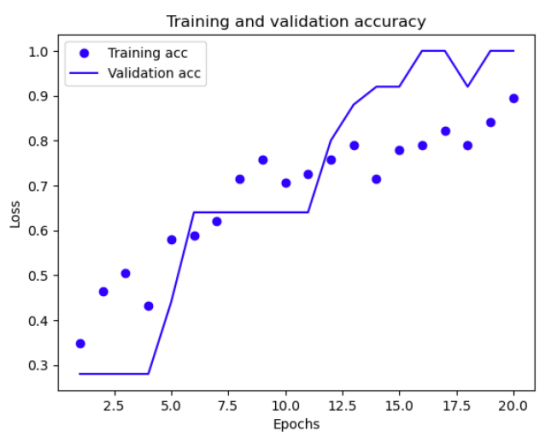
The following regularization measures were evaluated on test data: no regularization, weight decay (l1, l2, and l1_l2 kernel), dropout, batch normalization, and ensemble classifier. The results of these can be seen below.

`Test loss: 0.3700619339942932` `Test accuracy: 0.8666666746139526`	`Test loss: 0.4198187291622162` `Test accuracy: 0.9666666388511658`	`Test loss: 0.5797714591026306` `Test accuracy: 0.9666666388511658`
No Regularization	Weight Decay - Using L2 Kernal Regularization	Weight Decay - Using L1 Kernal Regularization

Test loss: 1.3661830425262451 Test accuracy: 0.6000000238418579	Test loss: 0.4031350314617157 Test accuracy: 0.9666666388511658	Test loss: 1.1605370044708252 Test accuracy: 0.20000000298023224
Weight Decay - Using l1_l2 Kernal Regularization	Dropout - 0.4 on Last Hidden Layer	Batch Normalization

The above results were evaluated on test data, and we can see that Batch Normalization did not have much effect in this situation nor did using an l1_l2 weight decay. However, all other types of regularization resulted in a 10% increase in test accuracy (from 86% using no regularization to 96% using either L1, L2, or Dropout regularization). In the end, however, L2 kernel regularization and a dropout of 0.4 resulted in the highest accuracy with a significantly low loss compared to other methods.

After having performed all of this evaluation, we can conclude that in this particular case, it was observed that categorical crossentropy was the best loss, adam was the best optimizer, and dropout and L2 kernel regularization were the best regularization methods to help reduce overfitting. A final training and testing on a new model was conducted this time on the optimal hyperparameters observed throughout this experiment to see how well this new model generalizes. The results can be seen below.

	
Training and Validation Loss	Training and Validation Accuracy

```
Test loss: 0.6924996972084045
Test accuracy: 0.9666666388511658
```

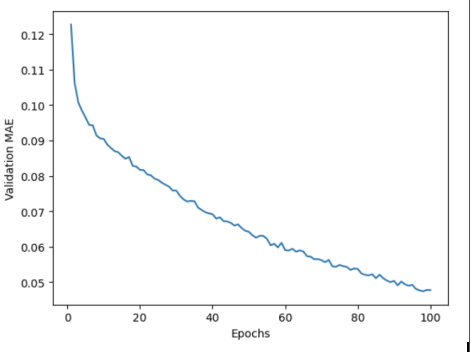
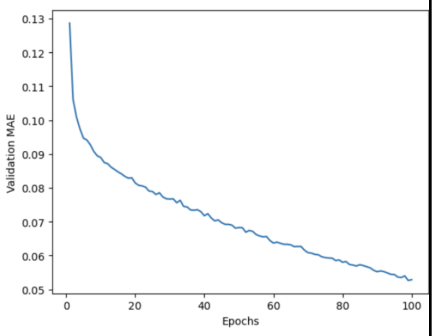
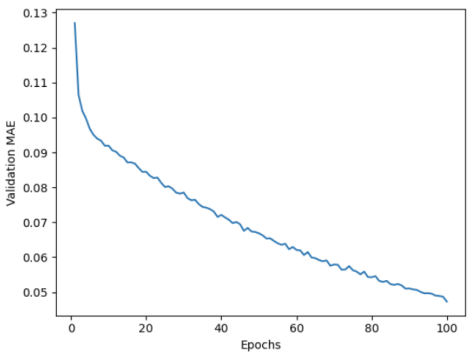
It can be observed now, after training a new model using the best hyperparameters mentioned in the conclusion above that now, we see the model start to overfit after 18 epochs and has a satisfactory test accuracy. The only downside observed is that the test loss had increased when we combined both a dropout of 0.4 and the L2 kernel regularization together.

Regression Example - Crime Dataset:

When building the regression model, various layers and hyperparameters were used. Three hidden layers were used with relu activation (the first having 64 units, second having 32 units and third having 16 units) and the output layer had 1 unit as output with no activation.

The data was also split into training and testing data whereby 30% of the dataset was test data and 70% was training data. For the validation data, 4 folds were used to perform k-fold cross validation so the validation data and partial training data were created from the training data for training each fold.

When evaluating loss functions on the crime dataset, the appropriate loss functions used for the regression problem are mse, mae, and logcosh. The model was trained once using each loss function while using a constant optimizer of rmsprop and mae metrics.

		
mse	mae	logcosh

	mse	mae	logcosh
Avg mae	0.1064	0.1008	0.1082
Avg loss	0.0248	0.1008	0.0127

All evaluations in this assignment were trained on the same model architecture, 100 epochs and a batch size of 10 units. The validation score after each fold was computed, and then it was plotted for each epoch, as shown above. It can be seen that all 3 losses performed quite well when training. Although, it is important to note that the mae loss had the highest loss amongst the 3 losses. For the rest of the evaluation on optimizers and regularizations, we will continue to use mse as loss.

	SGD	RMSprop	Adagrad	Adadelta	Adam	Adamax
mae	0.0963	0.1076	0.1006	0.1124	0.1090	0.1041
loss	0.0201	0.0253	0.0213	0.0276	0.0265	0.0238

The above shows the mae and loss for each optimizer when trained. The same learning rates, decay, and other hyperparameters were used in the optimizers as in the classification example earlier. Although all 6 optimizers performed well, it can be seen above that the SGD optimizer performed the best amongst all optimizers evaluated since it had the lowest mae and loss scores. Even though SGD performed the best, for consistency with the analysis, rmsprop will still be used to test the regularization on test data then, at the end, a final model will be trained using the best loss (mse), the best optimizer (sgd), and the best regularization/s (tbd).

	No Regularization	Weight Decay L2	Weight Decay L1
Test MAE Score	0.0979	0.0946	0.1771
Test MSE Score	0.0216	0.0249	0.1059

	Weight Decay l1_l2	Dropout (0.4)	Batch Normalization
Test MAE Score	0.1749	0.1014	0.1297
Test MSE Score	0.1030	0.0221	0.0330

When evaluating on test data using different regularization methods, it was observed that normally in this regression problem, the validation MAE decreases but when using L1 and l1_l2 kernel weight decay, the MAE increased which resulted in higher scores (worse results). Based on the results shown above, it seems that using L2 kernel regularization and dropout helped to generalize better and achieve better results when compared to other regularization techniques. I tried testing a new model using both dropout and l2 kernel weight decay, but the results were not better. In general, all forms of regularization helped in generalization except for l1 kernel weight decay and l1_l2 kernel weight decay which caused the MAE scores to increase instead of decrease.