CSc 84020 Neural Networks and Deep Learning
Homework 2: Classification and Prediction with DL Multi-Layer Perceptron
Andrea Ceres
Shao Liu

4. Discussions & Conclusions

We are tasked with determining if our trained models sufficiently recognize correctly a new sample from the test set using feedforward Deep Learning Multi-Layered Perceptron (DLMLP) architecture. We use the optimizer Stochastic Gradient Descent with Nesterov Momentum (*SGD*) [1] and the activation function Rectified Linear Unit (*ReLU*) [2] in our first set of models. In our second set of models, we replaced the optimizer to *Adam* [3] and the activation function to *LeakyReLU* [4]. We tested different combinations of dropout and L2 regularization.

Model-1 (SGD)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Dropout	0.0	0.1	0.2	0.4	0.0	0.1	0.2	0.4	0.0	0.1	0.2	0.4
L2 Regularization	0.0	0.0	0.0	0.0	0.0005	0.0005	0.0005	0.0005	0.01	0.01	0.01	0.01
Accuracy	0.7644	0.7634	0.7527	0.7245	0.7618	0.7553	0.7495	0.7060	0.7211	0.7111	0.7249	0.6897
Loss	0.6877	0.6928	0.7090	0.8019	0.7508	0.7567	0.7769	0.9156	0.9464	0.9581	0.9387	1.0390

Table.1 Experiment result for 3b

Model-2 (Adam)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Dropout	0.0	0.1	0.2	0.4	0.0	0.1	0.2	0.4	0.0	0.1	0.2	0.4
L2 Regularization	0.0	0.0	0.0	0.0	0.0005	0.0005	0.0005	0.0005	0.01	0.01	0.01	0.01
Accuracy	0.7641	0.7571	0.7538	0.7375	0.7613	0.7588	0.7559	0.7245	0.7371	0.7078	0.7220	0.7068
Loss	0.6984	0.6967	0.7123	0.7637	0.7440	0.7538	0.7577	0.8783	0.8738	0.9320	0.9059	0.9366

Table.2 Experiment result for 3c

Conclusion

From *Table.1* and *Table.2* we can conclude that with 30 epochs training, our first model (*SGD* & *ReLU*) with no dropout and L2 regularization achieved the highest accuracy of 0.7644 and the lowest loss of 0.6877. However, the difference between model-1 (using *SGD*) and model-2 (using *Adam*) are miniscule (~0.0003). Thus, after 30 epochs, both *SGD* and *Adam* are efficient and can optimize the model well, since the testing accuracy becomes stable. Our highest multilayer perceptron test accuracy of 0.7644 is nearly 10% higher than the test accuracy of 0.6717 using K-Nearest Neighbors on a small subset of our

dataset in *3a*. However, 0.7644 accuracy is relatively low for a six-class dataset classification. We surmise the reasons to be:

1. Feature vector of a 2D image may lose some information between neighbor pixels. A feature vector is a one dimension vector which cannot include all the information of an 2D image. It cannot reflect the 2D location and value relationships between pixels.
2. The architecture of our model is not complex enough to map the image vectors to target labels.
3. Our model is not adequately trained because 30 epochs are relatively few. We believe the solution to improve the accuracy of our model is to implement 2D Convolutional layers to capture the 2D space information from images. Moreover, we can add more layers and more neurons per layer to increase the complexity, and train for more epochs during the training process.
4. Much more data was available to us from the *Quick, Draw!* dataset, which contains 50 million drawings. We limited our dataset to 25,000 per class. With more processing power and time, we can improve the accuracy of our model by significantly increasing our sample size.

Also interesting is the changes to model-1 and to model-2 after we utilize different dropout rates. We have explained why the accuracy decreases in our previous analysis. Here we want to discuss why the change of two models are different. We can see from *Table.1* and *Table.2* that when we increase the dropout rate from 0.1 to 0.4. The accuracy of model-1 decreases more than model-2. For example, when we fix the L2 regularization rate to 0.0005 and set dropout rate to 0.1, 0.2, and 0.4, the accuracy of model-1 will drop to 0.7553, 0.7495 and 0.7060, respectively. Whereas the model-2 accuracy will drop to 0.7588, 0.7559, 0.7245, respectively. Each model-2 accuracy is higher than the corresponding model-1 accuracy. Because we are training on the same number of epochs, we can make a conclusion that when applying dropout to the model, *Adam* works faster than *SGD* in finding the optimal weights to minimize the loss. This result is also evidence to support the idea agreed by most scientists that *Adam* is more efficient than *SGD* but may not find the global optimal solution, as the best accuracy of *Adam* is lower than that of *SGD*.

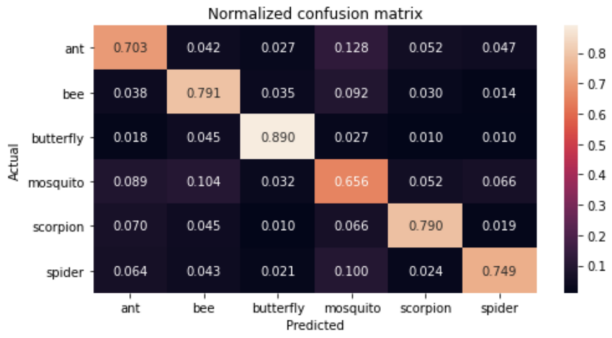

Fig.1

From the Normalized Confusion Matrix (*Fig.1*) we can see that among the six classes, butterfly scores highest, and mosquito scores lowest. One explanation is that butterflies are much easier to draw and more distinct from the other critters due to their unique shape and large wings. However, mosquitoes have the lowest score (0.656) and can be more easily predicted as other classes mistakenly, such as ants (0.128) and as spiders (0.100), which share a fairly similar body shape.

In conclusion, the model architecture we provide improves upon our benchmark set by KNN, but does not perform as well as it may perform with a less simple architecture.

References

[1] Herbert Robbins and Sutton Monro *A Stochastic Approximation Method* The Annals of Mathematical Statistics, Vol. 22, No. 3. (Sep., 1951), pp. 400-407.

[2] Nair, Vinod and Hinton, Geoffrey E. *Rectified linear units improve restricted Boltzmann machines*. In ICML, pp. 807–814, 2010.

[3] Kingma, Diederik & Ba, Jimmy. (2014). *Adam: A Method for Stochastic Optimization.* International Conference on Learning Representations.

[4] Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. *Rectifier nonlinearities improve neural network acoustic models*. In ICML, volume 30, 2013.

[5] Corinna Cortes and Mehryar Mohri and Afshin Rostamizadeh. *L2 Regularization for Learning Kernels.* 2012

[6] Analysis of underfitting
https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance

[7] Analysis of high/low accuracy & high/low loss combinations
https://datascience.stackexchange.com/questions/42599/what-is-the-relationship-between-the-accuracy-and-the-loss-in-deep-learning

[8] Normalized confusion matrix code
https://stackoverflow.com/questions/20927368/how-to-normalize-a-confusion-matrix

[9] Precision-Recall curve code
https://stackoverflow.com/questions/56090541/how-to-plot-precision-and-recall-of-multiclass-classifier

[10] ROC/AUC curve plots code
https://stackoverflow.com/questions/64924911/plotting-multiclass-roc-curve

[11] AUC code
https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html