CS F441 - Selected Topics from CS

A1 Report

Varun Gumma[1], Abhinav Sukumar Rao[2], and Raj Kashyap Mallala[3]

[1]2017A7PS0165H
[2]2018A7PS0172H
[3]2017A7PS0025H

1 Deep Learning Models

We have implemented thirteen different Deep learning models for the assignment. Each one of them is a neural network-based architecture with variations in either the number of layers, neurons, activation functions, weight initializations and a few other parameters. Early Stopping has been used to reduce training time.

The initial 3 networks are to test the effect of adding a new layer onto the model and also check the effect of batch size:

1. Shallow network (2 layers)
2. Shallow network (3 layers)
3. Network with different batch size

Next we compare the effect of optimizers on convergence We had to tweak the learning rate to a lower value for SGD owing to it not converging with the same learning rate as Adam:

1. 5 layer neural network with Adam
2. 5 layer neural network with SGD

We now compare the effect of weight initialization on the models by choosing 2 algorithms: Glorot-Uniform and He- uniform, which use a uniform and normal distribution for initialization:

1. Weights initialized using Glorot-Uniform
2. Weights initialized using He-Uniform

Another type of architecture we experimented with was an artificial neural network that looked like an autoencoder. The basic idea behind an autoencoder is to be able to extract features (and eliminate unnecessary details) by *bottlenecking* the image through smaller layers, and regenerating the new image upon resizing.

1. Autoencoder-type

We look at the effect of activation functions on our model, namely Tanh, Sigmoid and ReLU. Note that these operate only on the hidden layers and we always use softmax before the output layer for classification:

1. 5 layered NN with Tanh
2. 5 layered NN with Sigmoid

3. 5 layered NN with ReLU

Lastly, we inspect the effects of overfitting by comparing the train and test accuracy of the above models. Dropout is an algorithm where we *deactivate* a fraction of our neurons while training, in each epoch, to prevent them from being updated in that epoch:

1. A 5 layered NN without Dropout
2. A 5 layered NN with Dropout

Accuracies of various DL Models		
Feature of Significance	Train Acc.	Test Acc.
2-Layered NN	0.983	0.966
3-Layered NN	0.990	0.973
Batch size variation	0.9999	0.950
Adam	0.993	0.977
SGD	0.999	0.962
Glorot Uniform	0.995	0.978
He Uniform	0.995	0.980
Autoencoder-type	0.995	0.983
Tanh	0.929	0.928
Sigmoid	0.973	0.968
ReLU	0.997	0.981
No Dropout	0.981	0.924
Dropout	0.960	0.940

1.1 Observations and Explanations

1. **Effect of adding a layer:** We clearly see the increase in both train and test accuracy after increasing the number of layers for a shallow neural network on MNIST. This is because the model generalizes with addition of layers to some extent.

2. **Reduction in batch size:** The training accuracy for a larger batch size is much more than for a smaller batch size, but the model performs worse on test data for a larger batch size. Small batches can offer a regularizing effect (*Wilson et.al*), perhaps due to the noise they add to the learning process.

3. **SGD vs Adam:** We see that Adam is performing better in the test set than SGD. This way, we can infer that usage of an optimizer depends on the usecase, and we can't always decide what the best optimizer is for ALL models. Adam converges with lesser number of iterations compared to SGD, but takes more time since SGD operates on a single sample each iteration.

Accuracies and Time difference between SGD and Adam				
Model	Train Acc.	Test Acc.	Time(mm:ss)	epochs
Adam	0.993	0.977	02:44	22
SGD	0.999	0.962	02:43	79

4. **Weight initializations:** There was no significant difference in the train and test accuracies using the different initializers, except that Glorot-Uniform began with a much lower accuracy ($\sim$0.62) than He-Uniform ($\sim$0.72), but both models reached above 0.95 accuracy by 20 epochs. This could mean that it is just a matter of preference on what to choose.

5. **Autoencoder-type-ANN:** This model performed by far the best in terms of test accuracy, and this could be owing to the fact that we have a lot more layers than our other models, and also because we force the model to figure out the features by decreasing layers and regenerate the image using those features.

6. **Effect of activation functions:** ReLU activation performs the best of all three, while tanh performs the worst of them all. This is because the ReLU function doesn't saturate for larger weights while both tanh and sigmoid do. In fact, tanh saturates fastest among all three which might be the reason for lower accuracy.

7. **Effect of Dropout:** Adding dropout to our NN improves our test accuracy by $\sim$2% as described in the table. This is because the machine learns to not consider every feature for training. So all features will have equal contribution and no feature will have a very high corresponding weight value. Redundant features are also dropped this way, since the machine will assign very low values in the corresponding weights.

2 Machine Learning Models

We have implemented the following five ML models to compare against the aforementioned neural network architectures:

1. Logistic Regression
2. Support Vector Machine
3. Decision Tree
4. Random Forest
5. K-Nearest Neighbors

Accuracies of various ML Models with avg pooling		
Model	Test Acc.	Time(s)
Logistic Regression	0.926	143.39
Support Vector Machine	0.980	315.34
Decision Tree	0.889	17.51
Random Forest	0.968	68.04
K-Nearest Neighbors	0.974	185.85

Accuracies of various ML Models with max pooling		
Model	Test Acc.	Time(s)
Logistic Regression	0.918	135.54
Support Vector Machine	0.976	315.32
Decision Tree	0.890	13.31
Random Forest	0.966	48.17
K-Nearest Neighbors	0.967	166.14

The MNIST dataset provided consists of 60000 28×28 images of handwritten digits. Since feeding a feature vector of 784 was taking too long to train, we have first applied avg-pooling or max-pooling on the image and reduced its size to 14×14. Now that the feature vector was 4 times smaller, the training took considerably lesser time for all the models, without any loss of accuracy.

2.1 Observations and Explanations

1. SVMs (with RBF) produced more accurate ($\sim$0.97) results among ML models and sometimes even out performed DL models.

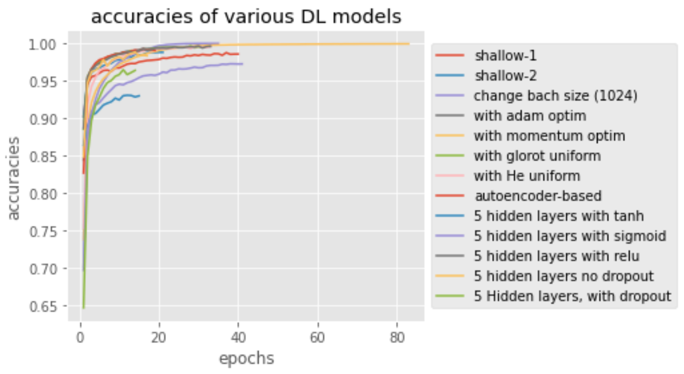
2. As we observe KNN (with K as 5) performing well ($\sim$0.96), we can conclude that the local constancy assumption holds good for this feature space. This can be observed from the images where no instance of a class is very similar to an instance of another class.

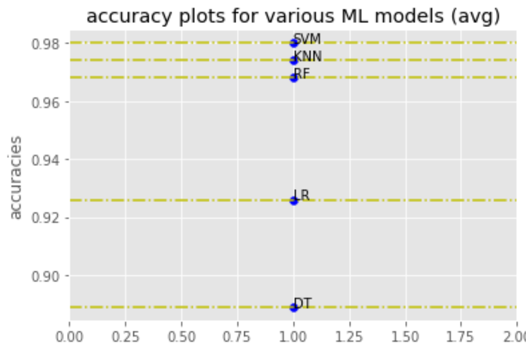
3. Logistic Regression has provided a comparatively lower accuracy (∼0.92) as it is a Single Layer Perceptron, i.e. without any hidden layers and hence is unable to learn any matured features and cannot generalize well.

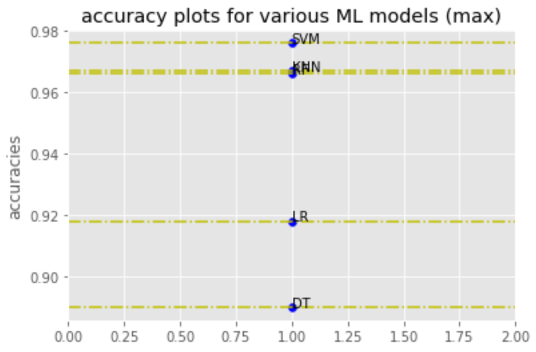
4. Decision Trees do not have any non-linear functions and hence these models cannot generate any matured features as well. The result of a Decision Tree is a piecewise-constant function, with one piece per leaf. Each leaf requires at least one training example to define, so it is not possible for the decision tree to learn a function that has more local maximae than the number of training examples (Goodfellow *et al.*) This could be the major reason for its poor performance, which was the lowest among all models.

5. Random Forests outperform Decision Trees and various DL models owing to its *ensemble* properties. The major reasons for the better performance is that the trees in the Random Forest are fully grown and unpruned (unlike in DTs) and hence the feature space is split into more smaller regions. Secondly, these trees are diverse as at each tree node a random set of features are considered for splitting.

3 Accuracy Plots


accuracies of various DL models
accuracies
epochs
shallow-1
shallow-2
change bach size (1024)
with adam optim
with momentum optim
with glorot uniform
with He uniform
autoencoder-based
5 hidden layers with tanh
5 hidden layers with sigmoid
5 hidden layers with relu
5 hidden layers no dropout
5 Hidden layers, with dropout


accuracy plots for various ML models (avg)
accuracies
SVM
KNN
RF
LR
DT


accuracy plots for various ML models (max)
accuracies
SVM
KNN
RF
LR
DT