CS 535 - Machine Learning

Assignment 5

Due Date : Monday, December 4, 2017 (11:55 p.m)

Instructions for Report:

- First thing first. Assignment seems difficult, but it isn't. Purpose of this assignment is to give you idea of different architectures of neural network and how to optimize the parameters. This also act as foundation for you to understand the buzz word **Deep Learning**. This assignment will also familiarize you with the reporting tools that are widely used in research community.
- Deliverable/output is mentioned for every question at the end of question statement. You are only required to provide that output along with comments on the results if asked.
- You are required to prepare your final report in *LaTex* including any figures, graphs or tables mentioned in question statement.
- You will be provided *LaTex* template and have to fill in the details. So Simple!
- So now you are thinking we have to install a new tool just to write report!! No you don't have to. Let's take 5 minutes to setup our reporting environment.
- Create an account on https://www.overleaf.com. Click on **New Project** and then on **Blank Paper**. Remove anything in this blank paper and copy and paste everything from provided template. You will see an error in the right panel. Ignore it.
- In left panel, click on files and upload image file from your computer that is provided with template. Error will be removed and you will see a nice document that you will be filling. Template will act as guide on how to include images, start sections (headings in MS Word) or how to start subsections. It also has table that you might want to use for reporting different things.
- Output in PDF format is shown at the right and source for that PDF in the middle. On the left panel, you can see files, which for now have only two files. One image file

that you uploaded and one **main.tex** file. Right click on main.tex file and Rename it to **yourRollNo.tex**.

- Start coding and getting graphs and other results. Upload them on overleaf and include them in the report.
- It just took five minutes to setup did it not? It would seem frustrating to write a report in a new environment, but you will thank your instructors later.
- You can further Google anything that you might need. For example, how to insert figure on the top of the page, or move it to the bottom of the page or change figure's size relative to the text etc.
- When you have finalized your report by uploading every graph and including it in the LaTex Report, download **yourRollNum.tex** file from the left panel and PDF by clicking on the top PDF button. Make a **yourRollNum.zip** file including these two files and your **jupyter code notebook**. Upload this Zip file on LMS. Please note that your zip file must have tex file, pdf file and jupyter notebook (3 files in total) and everything must be uploaded on LMS. No submission via email would be entertained.
- Any late submission will be penalized according to policy announced by instructor. Whole assignment would be considered late even if a part is submitted late.

Instructions for Coding Environment:

- We will be using same environment that you setup in Assignment 2 i.e Python with Jupyter notebook.
- You need to install two additional libraries known as *tensorflow* and *keras*. Just run following command in Terminal if you are using Linux.

  pip install –ignore-installed –upgrade tensorflow

  pip install –ignore-installed –upgrade keras

  or Following commands for Anaconda

  conda install tensorflow

  conda install keras

  That's it. It's that simple.
- If you are having trouble installing it, further details could be fetched from TensorFlow and Keras install guides.
- This assignment focuses on getting hands on experience on new tools and analyzing effect of varying different parameters of the learning algorithms. Therefore, you are provided a basic Jupyter notebook to serve as starting point for neural network part.
- Try to code in functions so that you can just change arguments to get different results that are asked in the questions. Otherwise ,things get messy by copying and pasting same version of code over and over with just one parameter changed. Why not make this parameter a variable!
- Here is how you can create a sequential model in keras. . Easy Right!!
- Don't worry by comparing your results with other students. Every environment and run would get different results. But overall trend would be the same.
- You can either use libraries in python to generate plots, or note down accuracies and loss in Excel and generate graph from there. But you have to include them as a figure in the final report.
- You are expected to give this assignment at least as much time as your instructors have spent to make it.
- Start Early!

Good Luck! ☺

Key Terms:
Batch: If system memory is low enough and we are unable to load all dataset, we split the dataset into n batches and train our model batch by batch. We update (weights) our model based on error calculated after every batch.
Epoch: When all batches i.e all dataset is given to our model for training, it is known as one epoch. We start second epoch from first batch again and keep on updating weights by looking at the error after every batch.

Problem 0. [10 Points]

- You are given a binary classification dataset namely credit card default prediction. We will be using this dataset for all problems except last one. Your task is to predict whether a person would default credit card payment of next month or not. Details could be fetched from Here if you are interested.
- Output of this problem is a single bar plot showing train and test accuracies of all experiments (single graph showing results for LDA,QDA and NB).
    1. You are familiar with sklearn libaray from assignment 2. Implement Gaussian naive bayes algorithm using sklearn.
    2. Implement Quadratic Discriminant Analysis (QDA) using sklearn with default parameters.
    3. Implement Fisher Linear Discriminant/Linear Discriminant Analysis (Both are same things) in sklearn library. Use number of components equal to $n\_classes - 1$ (as discussed in the class). Don't use any priors.
    4. Comment in the report on why it is asked to use Guassian Naive Bayes and not Multinomial Naive Bayes and which method among these three gave you best results and why?

Problem 1. [10 Points]

- This problem focuses on Logistic Regression and eventually developing a neural network. Output for this problem is going to be a single graph with line plot for every variation. Something like figure below. Where accuracy for both training and testing

dataset is on the Y-axis, while epoch number is on the x-axis. So there are two lines for every experiment. One showing training accuracy, while second one is showing testing/validation accuracy. For simplicity, we are assuming that our validation and testing sets are same so saying validation or testing dataset would not make any difference. You will have to comment on the observation and results obtained from these experiments in your report.

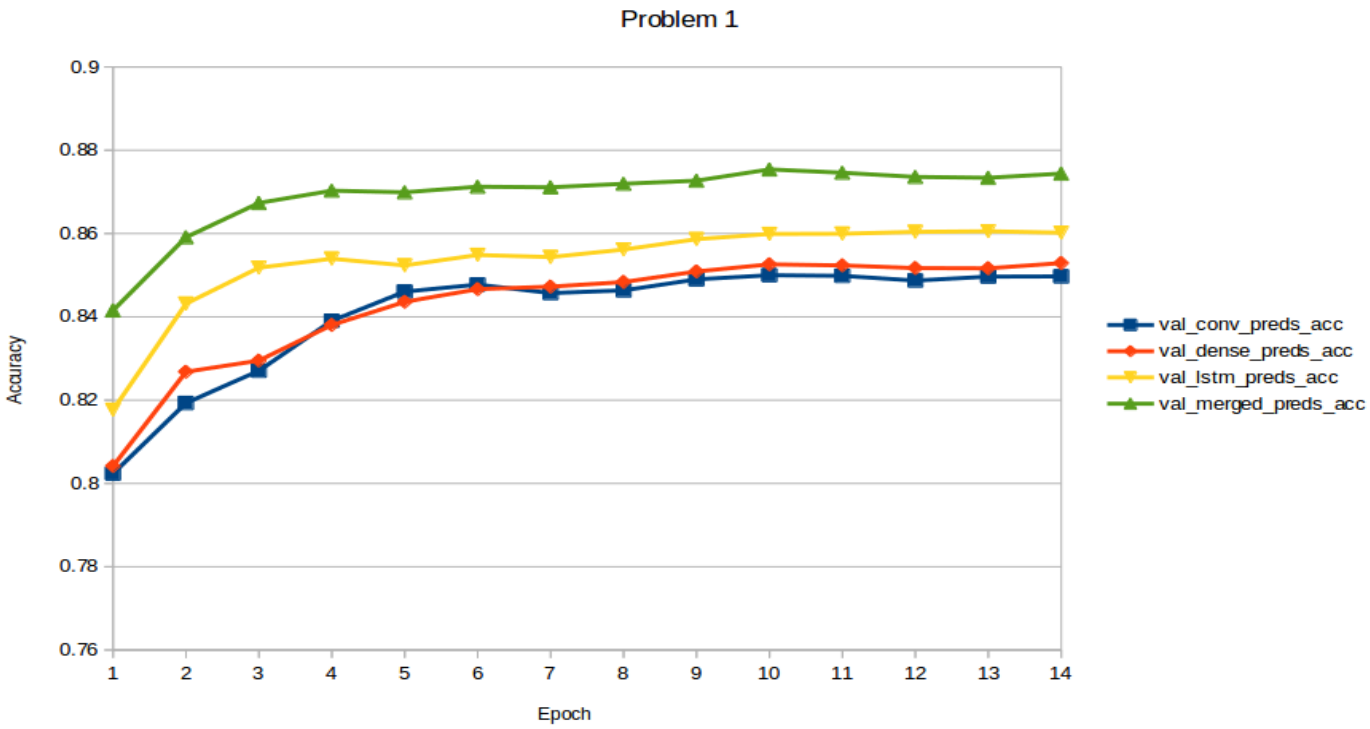

Figure 1: Problem 1: Comparison of Varying Layers Size

- Following Configurations are fixed for all experiments in this problem.

  Activation: Sigmoid

  No of Output Units: 1

  Epochs: 5

  Optimizer: Stochastic Gradient Decent (SGD)

  Error Function: Sum of Squared Errors (SSE)

1. In your sample notebook, Implement a simple single layer perception with one output unit (looks like logistic regression does it not?). Plot the accuracy of every epoch for both training and testing.

2. Add an additional layer between input and output layer with 5 units.This layer is known as hidden layer. Plot train and test accuracies for every epoch on the same graph as above.
3. Change number of units of hidden layer from 5 to 10. Plot the accuracies for train and test on the same graph.
4. Change the number of units of hidden layer from 10 to 50. Plot the accuracies for both train and test set.

Problem 2. [10 Points]

- Purpose of this problem is to see the effect of using different activation functions and loss function.
- Output of this problem is two graphs with multiple lines for different experiments. One graph shows accuracies for both train and test data, while second graph shows error(also known as loss) for train and test data. Comment on the observations of results in the report. We will build upon what we were doing in part 4 of Problem 1. Plot results of accuracies and loss only from Problem 1, part 4 on two new and empty graphs so that we can compare our results with upcoming experiments.
- Following parameters are fixed for this experiment, unless specifically mentioned to change it in the statement.

  Epochs: 5

  No of Output Units: Activation Function Dependant

  Optimizer: Stochastic Gradient Decent (SGD)

  Error Function: Sum of Squared Errors (SSE)

  Number of hidden units: 50

1. Use tanh activation function on both hidden and output layer. Plot the accuracies and loss for every epoch for train and test dataset on the graph.
2. Use Rectified Linear Unit (relu) activation function on hidden layer, while sigmoid on output layer. Plot accuracies and loss for every epoch for both train and test data.

3. Use Rectified Linear Unit (relu) activation function on hidden layer, while sigmoid on output layer. Change cost function to binary cross entropy. Plot accuracies and loss for every epoch for both train and test data.
4. Use relu activation function on hidden layer, while softmax activation function on output layer with SSE as your cost function. Encode labels to one-hot encoding (Google it how to do it in keras). Plot accuracies and loss for every epoch for both train and test data. Comment in report why we use one-hot encoding for softmax but not for sigmoid or tanh.
5. Use relu activation function on hidden layer, while softmax activation function on output layer. Labels are still one-hot encoded. Change cost function to categorical cross entropy. Plot accuracies and loss for every epoch for both train and test data. Comment in report why we are using categorical cross entropy and not binary cross entropy.

Problem 3. [20 Points]

- This problem is to analyze the effect of varying learning rate and adding momentum.
- Output of this problem is two graphs with multiple lines for different experiments. One graph shows accuracies for both train and test data, while second graph shows error(also known as loss) for train and test data. Comment on the observations of results in the report.
- Following parameters are fixed for this experiment, unless specifically mentioned to change it in the statement.

  Epochs: 5

  No of Output Units: 2

  Activation Function on Hidden Layer : relu

  Activation Function on Output Layer: softmax

  Optimizer: Stochastic Gradient Decent (SGD)

  Error Function: Categorical Cross Entropy

  Number of hidden units: 50

  Labels: One-hot encoded

1. Use Learning Rate = 0.001 with no momentum. Plot accuracies and loss for both train and test dataset.
2. Use Learning Rate =0.2 with no momentum. Plot accuracies and loss for both train and test dataset.
3. Use Learning Rate =0.5 with no momentum. Plot accuracies and loss for both train and test dataset.
4. Use Learning Rate =0.9 with no momentum. Plot accuracies and loss for both train and test dataset.
5. Use Learning Rate =0.01 with momentum = 0.9. Plot accuracies and loss for both train and test dataset.

Problem 4. [10 Points]

- This problem is to analyze different optimizers.
- Output of this problem is two graphs with multiple lines for different experiments. One graph shows accuracies for both train and test data, while second graph shows error(also known as loss) for train and test data. Comment on the observations of results in the report.
- Create two new graphs from the results of problem 3, part 5 so that we can compare it with our upcoming experiments. Following parameters are fixed for this experiment, unless specifically mentioned to change it in the statement.

  Epochs: 5

  No of Output Units: 2

  Activation Function on Hidden Layer : relu

  Activation Function on Output Layer: softmax

  Error Function: Categorical Cross Entropy

  Number of hidden units: 50

  Labels: One-hot encoded

1. Use *rmsprop* with default parameters as optimizer. Plot accuracies and loss for both train and test dataset.

2. Use *adam* with default parameters as optimizer. Plot accuracies and loss for both train and test dataset.
3. Use *nadam* with default parameters as optimizer. Plot accuracies and loss for both train and test dataset. Comment in the report about the difference between SGD, rmsprop, adam and nadam optimizers.

Problem 5. [40 Points]

Easy up till now. You must've completed these tasks within an hour. Let's make things a little bit interesting. For this problem, you have to do a little research and learning for yourself. Now change the dataset from credit card default binary classification to a 10-class image classification. Download python version of the CIFAR-10 dataset from This link. Information about the data, labels and python code to open the data is given on the site. Each batch is stored in a numpy dictionary. We are only interested in **data** and **labels** for experiments. Rest is metadata for your understanding. Convert labels to one-hot encoding. We will be using data_batch_1 to data_batch_4 for training (for your convenience, it is suggested to merge all these four files into a single variable/file for training purposes), data_batch_5 for validation and test_batch for testing. Yes, this time we have all three splits and now validation and testing mean separate things. For this problem, you have to do a little bit search and learning for yourself. Please look at https://keras.io/callbacks/.

- This problem is to analyze the difference between traditional neural network and convolutional neural network.
- Output of this problem is two **tables**. One for accuracies and one for loss. One Table is already given in *LaTex* template. Either create second table for loss or add additional rows in existing table. Please note that epoch wise loss and accuracy is not required. Only final accuracy and loss should be reported. Accuracy and loss will be reported for all three splits i.e train, validation and test. Please see the table in report template.
- Useful tip for learning enthusiast: If you have a matrix as an input and you want to convert it into a vector, you use *flatten* layer. If you want to convert a vector into a matrix, you use *reshape* layer. If you are working on matrix, you use 2D convolution, If you are working on a vector, you use 1D Convolution. This would help those who want

to convert the image vectors given in dataset to image matrix for visualization. This is not part of assignment though. If you want to visualize your model in a graphical form, Here is how you can do that.

1. Till now we were working with binary classification problem. Our output layer had 2 units with softmax activation. Use the network that we developed till problem 4, part 3, with exactly same parameters except for number of output units. Figure out yourself how many those should be now. We were using fixed number of epochs till now, but let's make it more adaptive to our dataset. Set number of epochs to 1000. Use validation set for early stopping of training. That is, if validation loss does not decrease for 3 epochs, stop the training. Please see callbacks link shared above to learn more about early stopping. Furthermore, we were using the model that we achieved after fifth epoch, although there was a possibility that a model achieved after 2nd epoch yields better accuracy on testing but we had no means to verify that. Let's implement that little check also. Save best weights only. That is, only those weights of the models should be saved that have given maximum validation accuracy. This is implemented already in save checkpoint callback. You just have to use it. Please check above link. Now, once training stops, load those best saved weights, and give the model our test split of dataset to get testing accuracy and testing loss. There are plenty of examples on official keras website on how to save and load weights and then get testing accuracy. Report all three accuracies i.e train, validation and test in the table along with three losses either in another table or by adding more rows to existing one. For train and validation accuracies and losses, only best ones should be reported, not for all epochs. This network is known as feed forward network.
2. Now let's add a convolutional layer after input layer with 32 filter, 2 kernel size and 1 strides with *relu* as activation and after that, add a Max Pooling layer with pool size 2. Rest of the network remains the same. Report accuracy and loss in a new table (don't use table used in part 1) for train, validation and test set.
3. Now let's change the filter size of convolutions to 64, kernel size 4. Rest of the things remain the same. Report accuracy and loss in a new table (don't use table used in part 1 or 2) for train, validation and test set.
4. Comment in the report on following.
   - What convolutional layer learns about the data?
   - What are the improvements after including convolutional layer?

- Why Max Pooling is necessary to add after convolution layer and what it does?
- Is adding more and more layers is always a better choice? Argue on why or why not?
- Now that you have implemented so many things, what do you think deep learning is?