CMME139

Introduction to Deep Learning

Assignment #3

Due Date: 07/06/2023 Wednesday (23:59)

Structure: Group Assignment

Score Contribution: 20%, evaluated over 20 points.

This is a group assignment. There is no code submission involved in this assignment. There will be a report submission in this assignment

Objectives

- **To develop experience and intuition around neural networks**
- **To observe the effects of parameter selection**

Overview

The objective of this assignment is to experiment with parametrization in neural networks. Students are expected to do experiments with different parameter settings, compare the results and communicate their findings by a report. We will also do some experimentation around major factors that would affect the performance of neural networks.

We expect you to work with a classification task: Ideally, we expect you to work with the outcome of Assignment #2: The neural network for the mobile games data (please indicate your setup for classification in the beginning of the report. i.e., do you have 2 or 3 classes?). However, you can also work with the Mnist dataset or any other example from the Web, with a one point deduction. (However, a word of warning: Mnist is a very sanitary example with very high accuracies. You may not observe performance changes with such data). Note that the assignment does not involve creation of new code: You will work with your past codes while changing parametrization and make observations.

Performance indicators

In this assignment, you will examine two aspects of the performance of a neural network: Accuracy and Computational Cost. In addition, we may/would like to look at the effects of these parameters for overfitting during training (a suggestion towards your conclusions). In terms of accuracy, you don't need any additional explanations: You already worked with accuracy in the assignments, and you worked with classification accuracy in the workshops.

In terms of computation cost, you will need a function to measure the computation time. I suggest you use system.time() function in the base R functions. The system.time() function provides 3

numeric results: User time, system time and elapsed time. User time describes the time spend in the user process. However, this often includes some additional costs due to operating system and etc. running in the background. Elapsed time describes the total time spent until the task is finished in real time. System time, or Wall clock time, describes time spent by your program on the CPU. In most academic studies, we use system time to describe ideal conditions, and you should use this as a metric in your reports (However, in some cases system time is almost negligible. In those cases, we usually report user time instead). Another alternative is to use sys.time() function from the base R functions which can be used to log a specific time when your code is being executed. Check the Web for their use-cases.

Overfitting describes the condition where the trained models behave differently in data that they have not seen before. In the most informal sense, overfitting can be observed by comparing training loss and validation loss: If losses in both conditions show similar responses, we conclude that there is no indication of overfitting. However, if losses show different behavior, we conclude that there is a possibility of overfitting (Note that it does not necessarily mean overfitting. It may be due to some other effect).

Experimental Setting

Here, several parameters of a neural network are described. In this assignment, you are expected to evaluate these parameters in your code and report your observations. These parameters are:

- **Number of Epochs (2 pts):** What happens if you change the number of Epochs in a neural network? How does the accuracy get effected? How does the computation time get effected? Can we use only a small number of Epochs?
- **Number of Layers in a Neural Network (2 pts):** What happens when you add more hidden layers in a neural network? (Here I am focusing only on dense layers) How does the accuracy get effected? Does linearity play a part in the change in accuracy (or at least can you observe it)? What about computation time (you should consider this along with number of parameters)?
- **Number of Neurons in the Layers (2 pts):** Recall that in lecture sessions we mention that increase in number of neurons in a layer means a transformation to a higher dimensional representation, and this may be an answer for non-linear problems. We also mentioned that a decrease in number of neurons in a layer means a transformation towards generalization. Notice that, in terms of accuracy we are describing a double-edged sword: Higher number of neuron representations are good for solving non-linear problems but increase probability of possible overfitting. On the other side, lower number of neurons in a layer support generalization and avoid overfitting but they are not good for non-linear problems.
  Can you describe what happens when you change number of neurons in layers? (Note that this may be very difficult to observe in a substantial manner. We ask you to design an experimental setting based on your theoretical foundations and report your results. It is possible that you may not observe effects in a well-founded matter. For this task, it is the design and proposed idea that matters) Here, you can also skip evaluating the computational cost, as next experiment will summarize it.

- **Number of Parameters (2 pts):** What happens when the total number of parameters change in a neural network (For computation time? For accuracy?)
- **The Structure of the Neural Network (3 pts):** With a fixed number of parameters, you can build different multi-layer neural networks. For example, one in a pyramid shape or one with uniform size in each layer. Does it effect accuracy or computation time?
- **Effect of Dropout (2 pts):** What is the effect of dropout and rate of dropout? Computationally? Accuracy wise? What do you think happens?
- **Effect of Class Imbalance (4 pts):** The datasets exhibiting a pattern where classes are not balanced is described as having a class imbalance problem. Classes that have a significantly higher number of instances may benefit from this at training time. As a result, a classification model may contain a bias toward the class that has a higher number of cases. Class imbalance is one of the biggest reasons of overfitting. When examining parameters, it would be a good idea to experiment on class imbalance. We expect that you, by manipulating your data frame compositions, create imbalance in your training samples and experiment on the class imbalanced training.
  When working with real life data, many times class imbalance is an unavoidable situation. As neural networks are huge "equation machines involving weight calculations", they also provide a intuitive way to battle such conditions: It is possible during training, to instruct a neural network to "balance out" class imbalance problems by assigning weights to observations from different classes. This can be done by using the "class_weight " parameter. By using this, it is possible to level the playing field for instances from different classes. Below, you are provided with an example use of the class_weight parameter:

  ```
  model <- fit( trainingData,
                trainingTarget,
                epochs = 100, batch_size = 32, validation_split = 0.2,
                class_weight = list("0"= 1, "1"=2, "2"= 4) )
  ```

  In this part of the assignment, perform a small experiment, starting with a class imbalanced training case
- **Number of Training Samples (3 pts):** What happens if you change the number of training samples in your models? There are some super simple, yet effective ways to do this part of the assignment.

Note that we are aiming to build our understanding. In some of the cases, you will observe no difference, or in some cases although you observe changes, it would be hard to attribute them to a single reason. You should experiment and convey your observations through the report. (It would also hugely benefit if you had a discussion with your group members)

What to Submit

This is a report-based assignment. You should prepare a report, containing a cover page in pdf form and submit the report. There is no need for code submission.

What will be graded

- Report Quality
- Experiment Design
- Correctness of the Experimental Design
- Soundness of the conclusions

Some Tips and Tricks

I suggest you work with at least with a neural network having 2 or 3 layers as a baseline in your experiment designs: More will be too costly to run multiple tests, and less would be too simple.

Note that it is possible to decide on the code to work with along with other group members and divide tasks among group members.

This is a straightforward assignment, and the objective of the assignment is to make you observe the effects of parametrizations in a neural network.

Good Luck!

DL TEAM