MMD_PA2

Tobias Winter

May 2025

1 Introduction

In the following you will see how the provided datasets where processed and how the SVM works.

2 Prerequisites

To run the code, it is recommended to first select a new folder for the project and run the following commands.

For macOS and Linux:

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

On windows run the following:

```
python3 -m venv venv
venv\Scripts\Activate.ps1
pip install -r requirements.txt
```

This will setup a new Python virtual environment, activate the environment, and install all the dependencies the project has. To run the code, select the file *main.py* and start the code.

More detailed instructions on what settings can be configured, can be found the the source code.

In the next sections, the questions form the assignment will be answerd.

Please find all the plots bigger in the plot folder.

3 Part 1: Linear SVM with SGD variants

3.1 Question 1

Report how you selected the learning rate, batch size, and regularization parameter. Did you use different parameters for different datasets or optimizer variants?

A grid search is implemented in the function *parameter_tuning*(). The grid is defined as following:

1. learning rate: starting at 0.01 ... 2.0 with a step size of 0.02
2. l2 radii: starting at 0.1 ... 5.0 wit a step size if 0.5
3. with of without random fourier features

Since this takes a long time, computing this for the imdb and higgs dataset is not feasable, thats why the hing loss for the two datasets is around 1.0. The search was done in the first 10k rows of the higgs and imdb dataset.

Why would this take so long? A single training session (meaning calling the *start_svm_run*() function) takes around 42s. For 75k iterations, this would not be feasable. After 16 hours of searchin on the imdb and higgs dataset, I stopped the programm, and tryed to find the best parameters in the already tryed parameters.

If I would have more time, my next stratagie would be to

1. instant of accessing the first 10k rows of the datasets, I should access the rows at random.
2. Find a better grid, which needs the be searched

3.2 Question 2

Include convergence plots for the optimizers (training loss vs. number of passes/epochs).

In this PDF you can find the plots. The plots depict the average hing loss per epoch. All datasets where run for 10 epochs. Per image, there are three subplots, one for each optimiser. The batch size (bs), calculated performace (Pref) and time need for training can be found in the legend of each subplot. Unfortunately, i did not manage to reach good results for all datasets.

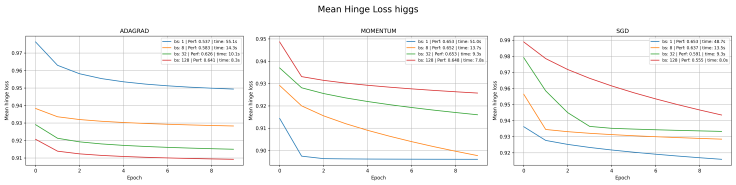

Figure 1: Higgs dataset witout RFF

3.3 Question 3

Report classification accuracy/AUC and runtime (e.g., number of epochs, total wall- clock time) for each dataset and optimizer variant.

This can be found in the plots.

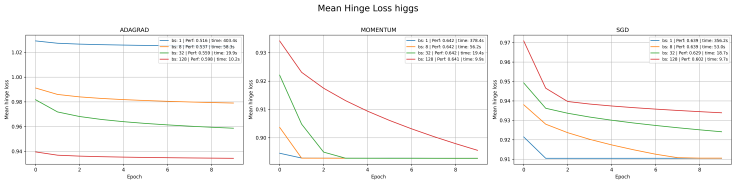

Figure 2: Higgs dataset without RFF

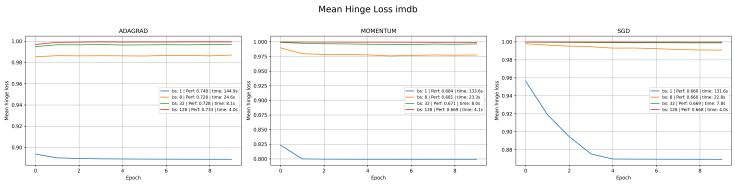

Figure 3: IMDB dataset without RFF

3.4 Question 4

Compare and discuss the effects of optimizer variants and batch sizes. What trends do you observe?

I can only speek for the toydata, since there i managed to get a avg hing loss under 0.5.

3.4.1 Adagrad

Adagrad has a lower avg hing loss then vanilla SGD. Can converge pritty fast with the right parameters.

3.4.2 Momentum

Momentum Usually converges after the first epoch with the right paramets.

3.4.3 SGD

Seems to converge over multiple epochs. Definitely needs more then 10 epochs to converge.

3.5 Question 5

Briefly describe your implementation of each optimizer.

All optimizer are implemented in the *LinearSvmSGD* class directly in the update function. Each optimizer adjust the "step".

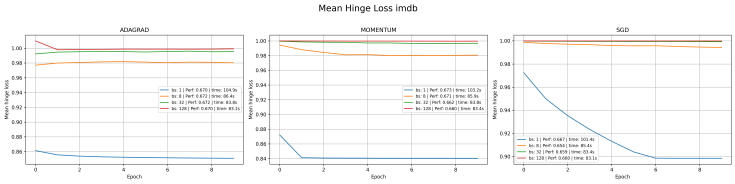

Figure 4: IMDB dataset with RFF

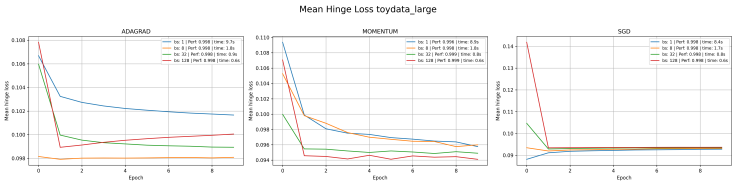

Figure 5: Toydata large without RFF

4 Part 2: Random Fourier Features (RFF) for Kernel SVM

4.1 Question 1

Briefly describe your implementation of RFFs.

This can be found in the *rff_sample.py* file.

1. init: sample random vectors from a normal dist. scaled by $1/\sigma$
2. init: sample random phase from [0, 2π]
3. init: scaling
4. transform: dot product of features and random vectors from init
5. transform: phase shift
6. transfrom: scale and return the feature map

4.2 Question 2

Report how you selected the number of RFFs (minimum 3 different values 100), learning rate, and regularization parameter.

I did not do that. I used 1000 dimensions for the feature map with a $\sigma = 1.0$

4.3 Question 3

Include convergence plots for one optimizer across different RFF sizes.

See the plots in this PDF where its stated that RFF was used.

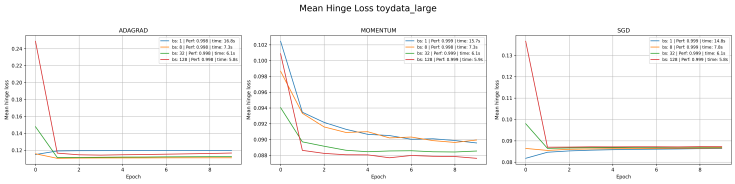

Figure 6: Toydata large with RFF

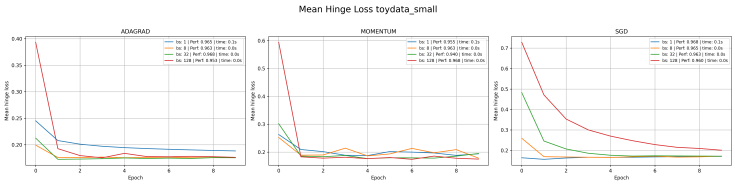

Figure 7: Toydata small without RFF

4.4 Question 4

Report classification accuracy/AUC and runtime for each dataset. Discuss how the number of RFFs affects accuracy and runtime.

See the accuracy/AUC scores in the images.

I did not varied the random features.

4.5 Question 5

For IMDB and Higgs, also report performance on different training set sizes (e.g., 1000, 2000, 3000 samples). Compare your results to using sklearn.svm.SVC.

I did not do that.

5 Part 3

5.1 Question 1

Report training convergence, accuracy/AUC, and runtime for at least three mini-batch sizes for each optimizer (standard SGD, momentum, Adagrad).

You can find the runtimes for the batch sizes in the plots.

5.2 Question 2

Discuss how mini-batch size affects convergence speed, optimization stability, and final accuracy. Do these effects vary across datasets?

Online (bs=1) had the most run time but was potentially out performing the other batch sizes in the avg hing loss. Where as the larges batch size was the fastes, which was to be expected.

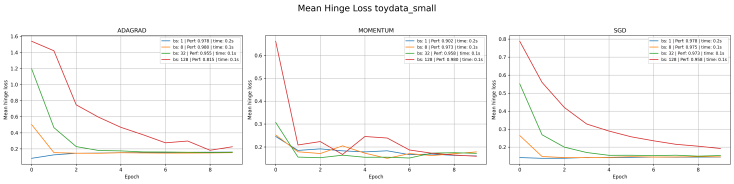

Figure 8: Toydata small with RFF

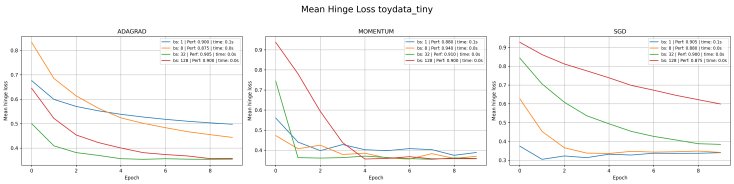

Figure 9: Toydata tiny without RFF

For accuracy, the larges batch sizes can be potentially less accuract then online, but not by much. (in the toydata)

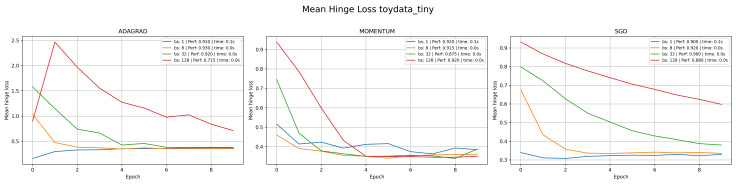

Figure 10: Toydata tiny with RFF