CS689: Machine Learning - Fall 2020

Homework 3

Submitted by Amit Gopal Hattimare

1.a. The saddle point equation is:

$$\min_{\mathbf{w}} \max_{\alpha_{1:N}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n \mathbf{w} - H(\alpha_n)) \quad \text{s.t.} \quad \forall n \;\; \alpha_i \in [0,1]$$

We can swap the $\min$ and $\max$ operations and then take derivative w.r.t $\mathbf{w}$ to find the minimum $\mathbf{w}$ parameters. Our modified saddle point equation is:

$$\max_{\alpha_{1:N}} \min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n \mathbf{w} - H(\alpha_n)) \quad \text{s.t.} \quad \forall n \;\; \alpha_i \in [0,1] \tag{1}$$

Let the function to optimize be $\mathcal{L}(\mathbf{w}, \alpha, \mathcal{D}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n \mathbf{w} - H(\alpha_n))$. Then,

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, \alpha, \mathcal{D}) = \lambda \mathbf{w} - \sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n^T) = 0$$

$$\implies \mathbf{w} = \frac{1}{\lambda}\sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n^T) \tag{2}$$

Putting (2) in (1), we get our dual objective.

$$\text{Dual}(\alpha, \mathcal{D}) = \max_{\alpha_{1:N}} \left[\frac{\lambda}{2}.\frac{1}{\lambda^2}.\left\|\sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n^T)\right\|_2^2 - \sum_{n=1}^{N}\left(\alpha_n y_n \mathbf{x}_n \frac{1}{\lambda}\sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n^T) - H(\alpha_n)\right)\right]$$

$$= \max_{\alpha_{1:N}} \left[\frac{1}{2\lambda}\left\|\sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n^T)\right\|_2^2 - \frac{1}{\lambda}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n \alpha_m y_n y_m \mathbf{x}_n \mathbf{x}_m^T) + \sum_{n=1}^{N} H(\alpha_n)\right]$$

$$= \max_{\alpha_{1:N}} \frac{-1}{2\lambda}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n \alpha_m y_n y_m \mathbf{x}_n \mathbf{x}_m^T) + \sum_{n=1}^{N} H(\alpha_n) \qquad \left(\because \left\|\sum_{n=1}^{N}(\alpha_n y_n \mathbf{x}_n^T)\right\|_2^2 = \sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n \alpha_m y_n y_m \mathbf{x}_n \mathbf{x}_m^T)\right)$$

This is the dual objective which is a maximization problem over α and does not involve $\mathbf{w}$.

1.b. The primal objective of logistic regression is given by the minimization problem:

$$\mathcal{L}(\mathbf{w}, \mathcal{D}) = \min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{n=1}^{N}(1 + \exp(-y_n \mathbf{x}_n \mathbf{w}))$$

Reasons for choosing to optimize the dual objective instead of the primal can be:

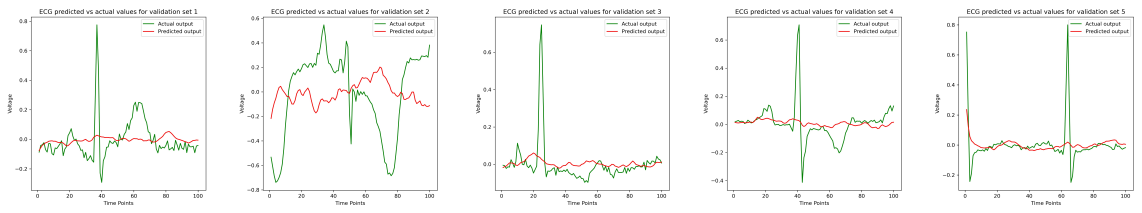


Figure 1: Validation set outputs with their predictions for the first 5 validation sets using the given neural network architecture.

1. The dual has N values to optimize and N^2 $\mathbf{x}_n\mathbf{x}_m^T$ multiplications whereas the primal has D values to optimize and only N $\mathbf{x}_n\mathbf{w}$ multiplications. Dual is better if $N < D$ for the given dataset $\mathcal{D}$ (though for most data $N > D$).
2. Similar to the SVC optimization problem, $\mathbf{x}_n\mathbf{x}_m^T$ term in the dual objective can be replaced by a suitable kernel matrix element $\mathbf{K}_{nm}$ where $\mathbf{K}$ is the $N \times N$ kermel matrix. It can also be replaced by a kernel function $\mathcal{K}(n, m)$ if we know some properties of the data. We can try out some existing kernel functions. Also, simialr to SVC, the constant λ can be removed and $\mathcal{K}$ adjusted appropriately.

2.a. Check out code.

2.b. Check table 1 for results.

Data	Mean squared error loss
Training set	0.0183
Validation set	0.0196

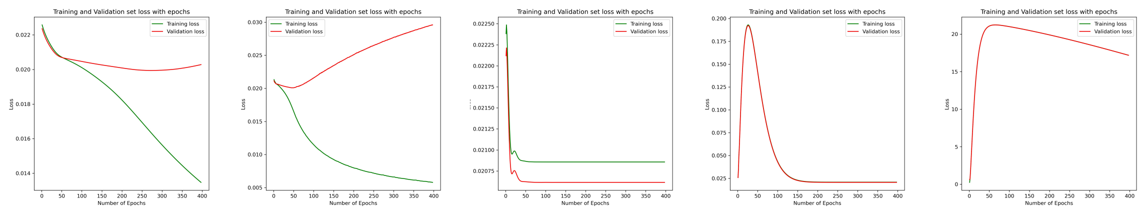
Table 1: Mean squared error loss on training and validation sets for the given model architecture.

2.c. The graphs can be seen in figure 1.

3.a. I tried 3 model architectures - simple neural network (NN), CNN based model, and RNN based model with LSTM cells. CNN and RNN models have fully connected NN as the end layer. I shall describe below how I fine tuned both models:

NeuralNetwork

I used a 2 layer neural network with fixed size of hidden layer units (h1 > h2). I used Adam optimization algorithm and fine tuned the learning rate using a linear search process. Table 2 shows the best validation loss and number of epochs to achieve that loss for varying learning rates. Other parameters were kept fixed as $momentum = 0.9$, $nesterov = False$. The learning rate curve for training and validation data can be



(a) lr=0.0001 (b) lr=0.001 (c) lr=0.01 (d) lr=0.1 (e) lr=1.0

Figure 2: Learning rate curve for a 2 hiddel layer NN model with Adam optimizer for varying learning_rate values.

seen in figure 2. From these tests, we can see that 0.001 is a good choice for learning rate for Adam. I tried the same learning rate with RMSProp optimizer but the validation set loss was still 0.0200.

My second experiment with NN was to choose a one hidden layer model with Adam optimizer using fixed lr=0.001 and seeing effects of hidden layer size. The results are summarized in table 3. This indicates that NN architecture is not very good in capturing the details of the data set and adding more hidden units doesn't help in learning. Even 50 units achieves comparable performance.

Learning rate	0.0001	**0.001**	0.01	0.1	1
Best validation loss	0.0199	0.0200	0.0206	0.0206	0.0222
num of epochs	279	51	115	273	1

Table 2: Effect of learning rate on final validation loss for 2 layer NN model.

Hidden layer units	50	100	300	400	**500**
Best validation loss	0.0205	0.0206	0.0204	0.0204	0.0203
num of epochs	118	63	55	54	51

Table 3: Effects of number of hidden units on learning rate for 1 layer NN model.

CNN

CNN models have lots of hyperparamters to choose from. I selected an architecture having 2 blocks of Conv-ReLU-MaxPool followed by a linear layer. The hyperparameters to tune here were number of filters (out_channels), filter size (kernel_size) and maxpool size for each block. Optimizer was fixed to Adam with lr=0.001. In first test, I tried different number of filters keeping other parameters fixed. The validation set loss is summarized in table 4.
In next test, I fixed the number of filters to 10 and changed kernel size keeping the stride fixed to 1. Results are in table 5. Another test was done to test the effects of max-pooling by keeping filter size as 10 and kernel size as 20. Results are in table 6. As you can see, in each successive test, I used the optimum value optained from the previous tests and kept those constant.

Number of filters	4	**10**	20
Best validation loss	0.0206	0.0206	0.0206
Num of epochs	43	24	45

Table 4: Effects of filter size on 2 block CNN model. Same filter size is used for both blocks.

Kernel size	5	**10**	20
Best validation loss	0.0207	0.0206	0.0206
Num of epochs	42	20	11

Table 5: Effects of kernel size on 2 block CNN model. Same kernel size is used for both blocks.

Maxpool size	**3**	5	8
Best validation loss	0.0207	0.0206	0.0206
Num of epochs	21	30	24

Table 6: Effects of Maxpool size on 2 block CNN model. Same maxpool size is used for both blocks.

RNN

There are different varieties of RNN architectures that can be chosen. For this task, I decided to use multi-layer RNN model with LSTM cells as the first layer and fully connected NN at the end. LSTM is popular for time sequence estimation tasks and combined with RNN architecture, it can work well. The hyperparameters are number of hidden units and number of RNN layers. I am fixing the linear layer to 1 as making it 2 did not provide improved performance in a separate test.
In the first test, I fixed the hidden layers to 1 and tried multiple values of hidden units. Results are in table 7. In second test, I fixed the hidden layers to 2 and tried same set of hidden unit values. Results are in table 8.

Hidden layer units	200	**300**	400	500
Best validation loss	0.0205	0.0204	0.0204	0.0204
Num of epochs	91	100	76	90

Table 7: Effects of hidden layer size RNN model with LSTM cells.

Hidden layer units	200	**300**	400	500
Best validation loss	0.0202	0.0202	0.0203	0.0203
Num of epochs	118	113	110	102

Table 8: Effects of hidden layer size RNN model with LSTM cells.

3.b. To select the best model, I compared three different architectures (NN, CNN, RNN with LSTM) and tuned their hyperparameters. The strategy to select their hyperparameters is discussed in part 3.a. Models are compared based on validation set loss, and training is done by keeping batch size as total size of data. Since the number of examples is less and models were overfitting easily, I did not go with mini batch approach.
From all the experimentation, I notice that all the models have almost similar performance. A statistical test can better compare the models, but looking at the prediction graphs, no model has been able to learn the spikes well.

For each table, I have made bold the best hyperparameter value. The best model is a 2 layer NN with Adam optimizer.

3.c. The architecture for the best model is in figure 3. It is implemented using sequential container in pytorch. It is described as:

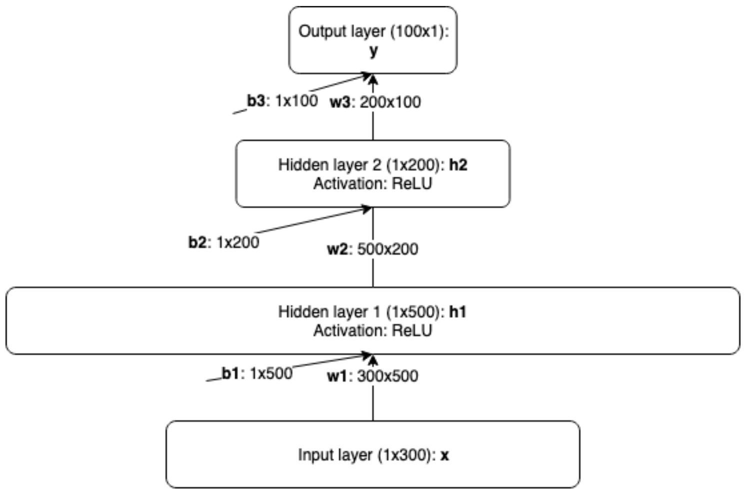


Figure 3: Architecture diagram for the best model: a multi layer neural network.

3.d. The best neural network model selection technique has already been explained in section 3.a. above. The hyperparameters for this model were number of layers, number of hidden units in a layer, optimizer, and learning rate. Number of hidden layers was chosen by doing experiments for 1 and 2 hidden layers and picking the best according to validation set loss. Adam was chosen as the default optimizer but its learning rate was chosen using a 1D grid search approach. Momemtum was kept as the default because conversion rate was not seen as a problem after tuning the learning rate.

3.e. The line plots are in figure 4.

3.f. For this task, I retrained the model using both validation and training data so that the data set size increases.

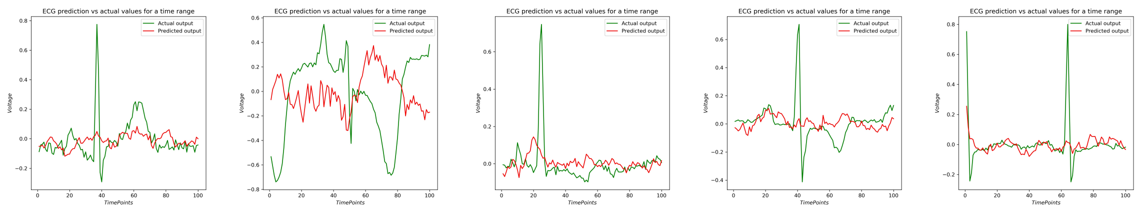


Figure 4: Validation data output and predictions for the first 5 sets using the best model.