Machine Learning Practical (MLP)
Coursework 4
Georgios Pligoropoulos - s1687568

Million Song Dataset Classification Problem

The songs have been preprocessed to extract timbre, chroma and loudness from various segments within the song. Each segment is represented by 25 features in total.

We have at our disposal two kinds datasets:

- Fixed input length where we are keeping the central 120 segments. This gives a fixed dimensionality of 3000 features as input.
- Variable input length where the songs vary in the available number of segments. So the number of features vary proportionally. We still have each segment represented by 25 features though. Depending on the sampling that was applied and the length of the song we may have from 120 up to 4000 segments.

We are given a subset of the MSD dataset which contains 40000 instances in the training set and 10000 instances in the validation dataset.

MSD-10 Classification Problem

A subset of the MSD dataset which contains 40000 instances in the training set and 10000 instances in the validation dataset. In addition we are given 9991 instances as a test set.

Our goal is to classify which of the ten(10) genres correspond to which song, as accurately as possible using artificial neural networks.

MSD-25 Classification Problem

A subset of the MSD dataset that contains 50000 examples for a balanced classification task and each of the 25 classes corresponds to 2000 labelled examples for training and validation. In addition 400 examples are provided for testing.

Our goal is to classify which of the 25 genres correspond to which song, as accurately as possible using artificial neural networks.

Tools

For all experiments we are using **Python** Programming Language and **Tensorflow** Framework.

Experimentation

Note: When in our explanations refer to **accuracy** we mean the final validation accuracy and when we refer to **performance** we mean how fast we reached the optimal accuracy for the current experiment.

Note that we will not be reporting final errors and accuracies after each experiment because in most cases the final error is not the minimum error and the final accuracy is not the maximum accuracy. We are most interested in the approximate results since we are far away from 100% accuracy to care for the decimal part of the metrics.

Baseline Classifier

In coursework 3 we managed to achieve ~50% validation accuracy on the validation set using our simple dynamic dropout algorithm. However this required a lot of tweaking of newly introduced hyperparameters.

Here we are building a new simpler model with the parameters for the keep probabilities of dropout being static.

So we are targetting MSD-10 classification problem with MLPs and receiving the fixed segments size dataset as input.

In order to be efficient on **development-experiment cycle** we are going to pick a variant of one of the simpler architectures used in coursework3 which provided promising results.

We are choosing to have three hidden layers and the dimensionality from input to output as follows:

(inputs) 3000 → 500 → 500 → 25 → 10 (outputs)

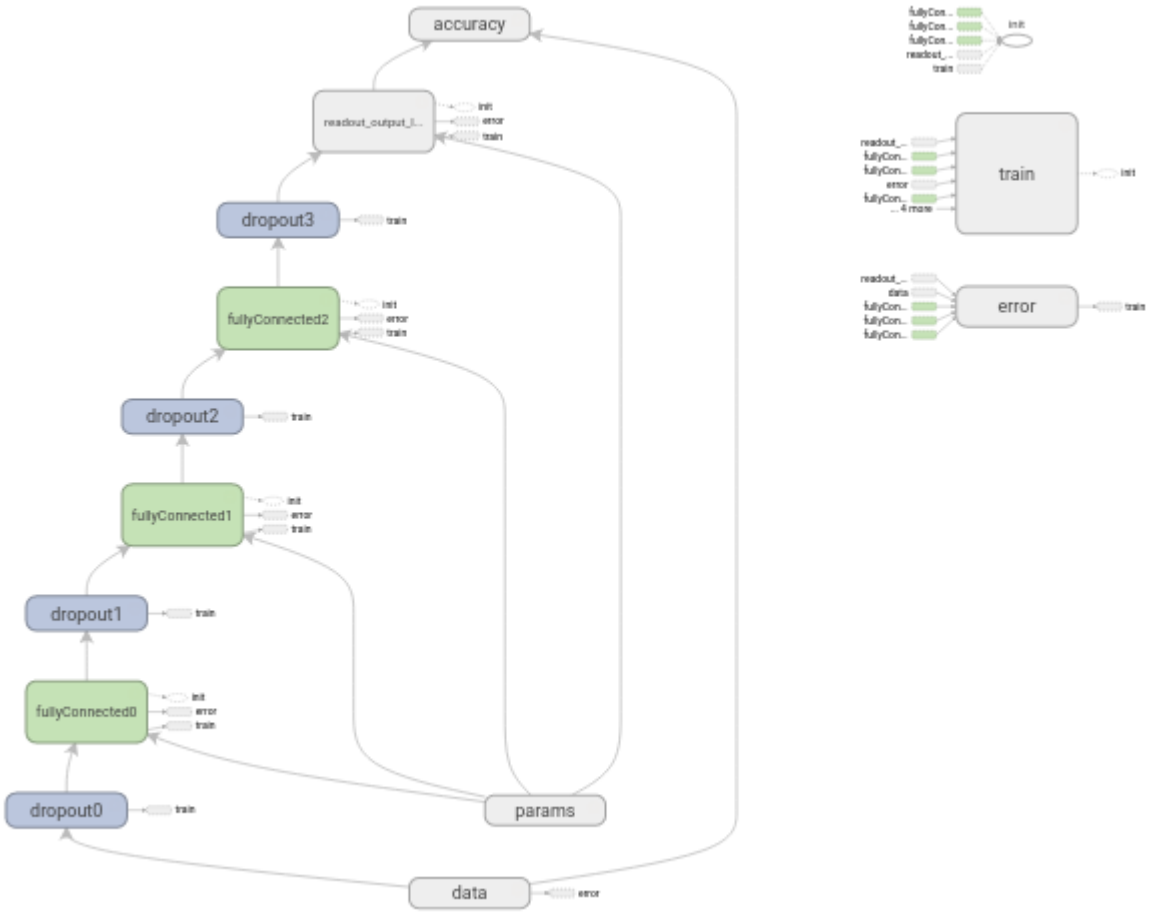
We will be using **affine transformations** which are interleaved with **tanh non-linearity**, and at the end we always have a **softmax output layer**.

Note that we are using **Batch Normalization**, **Dropout** and **L2 regularization** on all layers.

L2 regularization factor is common for all layers and is parametric via a placeholder.

Dropout keep probability of inputs and of hidden layers are also two parameters controlled via two placeholders. Note that the dropout keep probability of all the hidden layers is controlled by the same parameter.

As in coursework 3 the **error metric** is the **mean cross entropy of the batch** which is minimized using **Adam Optimizer**.



Graph 1: Multilayer Perceptron Architecture with dropout, L2 regularization and batch normalization on all layers. Three hidden layers and one readout layer

We are **pretraining** the weights and biases of all hidden layers using the **non-linear stacked autoencoder** from coursework 3. This is a straight forward process without any extra hyperparameters involved.

From coursework 3 we are choosing the **L2 regularization factor** that worked best which was **1e-2**.

We are also keeping the **Learning Rate** at a lower lever than the default value of Adam class in tensorflow. Learning rate is **1e-4**.

We will be using a **batch size of 50** for training.

Bayesian Optimization

Because it would have been difficult to tweak all of the parameters ourselves, we are exploiting the library **scikit-optimize** to get **Bayesian Optimization** to help with finding the optimal dropout keep probabilities.

The space of **dropout keep probabilities** has set to have the **uniform** distribution.

The dropout keep probabilities of the **input layer** are in the real space with **minimum value of 0.7** and maximum of 1.0.

The dropout keep probabilities of the **hidden layers** are in the real space with **minimum value of 0.4** and maximum of 1.0.

The function used for optimization is the **gp_minimize** which includes the hyperparameter **kappa**.
The kappa hyperparameter is a trade-off between exploration versus exploitation and it is set to **1.9**.

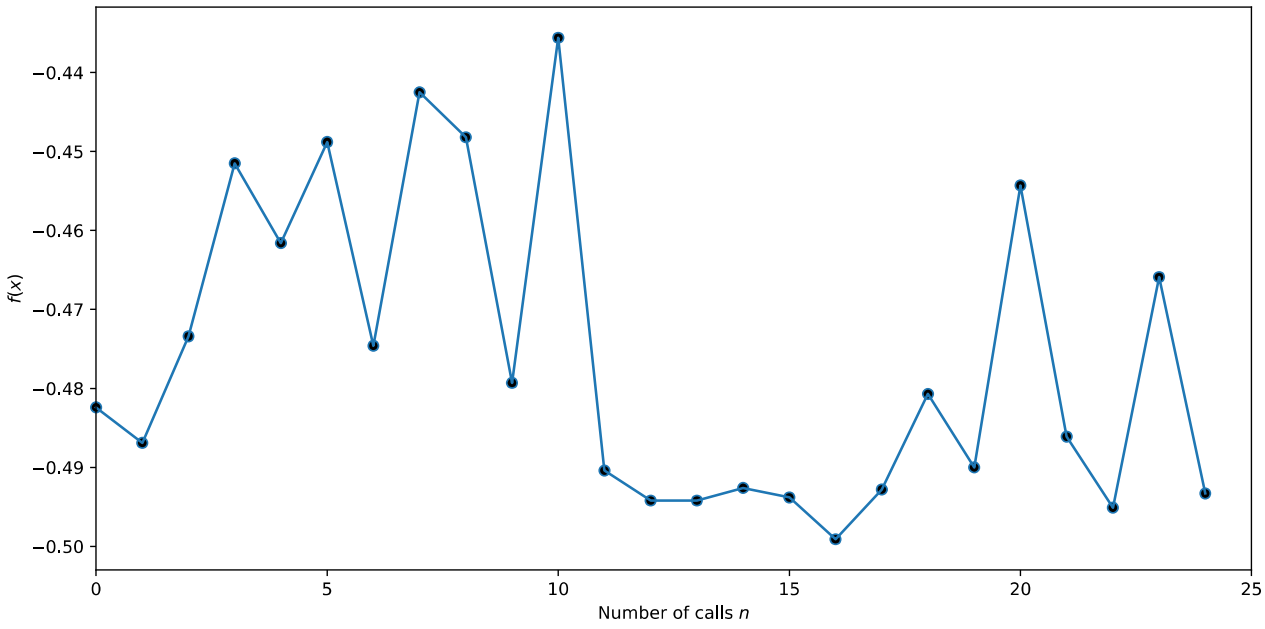
Our objective function is training and validating our deep neural network and we are trying to **maximize** the **maximum validation accuracy** we encountered after **35 epochs**.

We have configured our bayesian optimization to start after executing **5 random initializations**.

We are **evaluating** the objective function **25 times**.

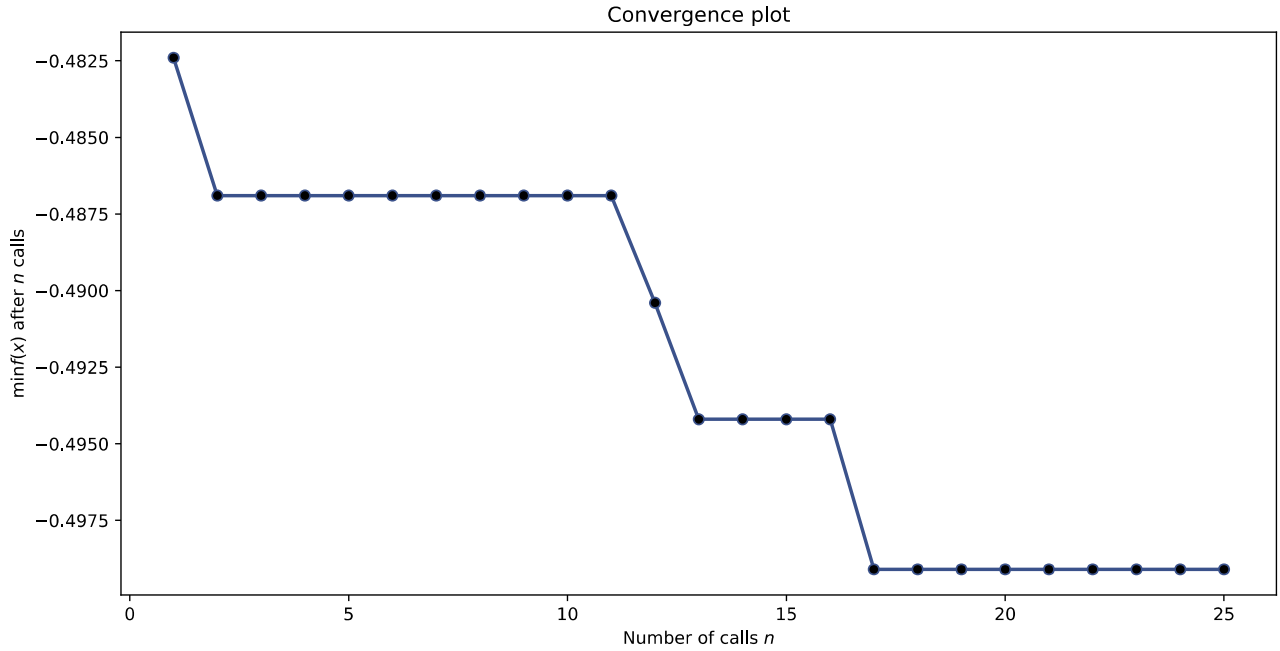
Bayesian Optimization Results

It follows the result of the objective function on every call/evaluation.



Plot 1: Objective Function result of Bayesian Optimization for 35 epochs per call. The objective is the maximum validation accuracy

Here we see the convergence plot which shows the rate at which the bayesian optimization was able to find the minimum value.



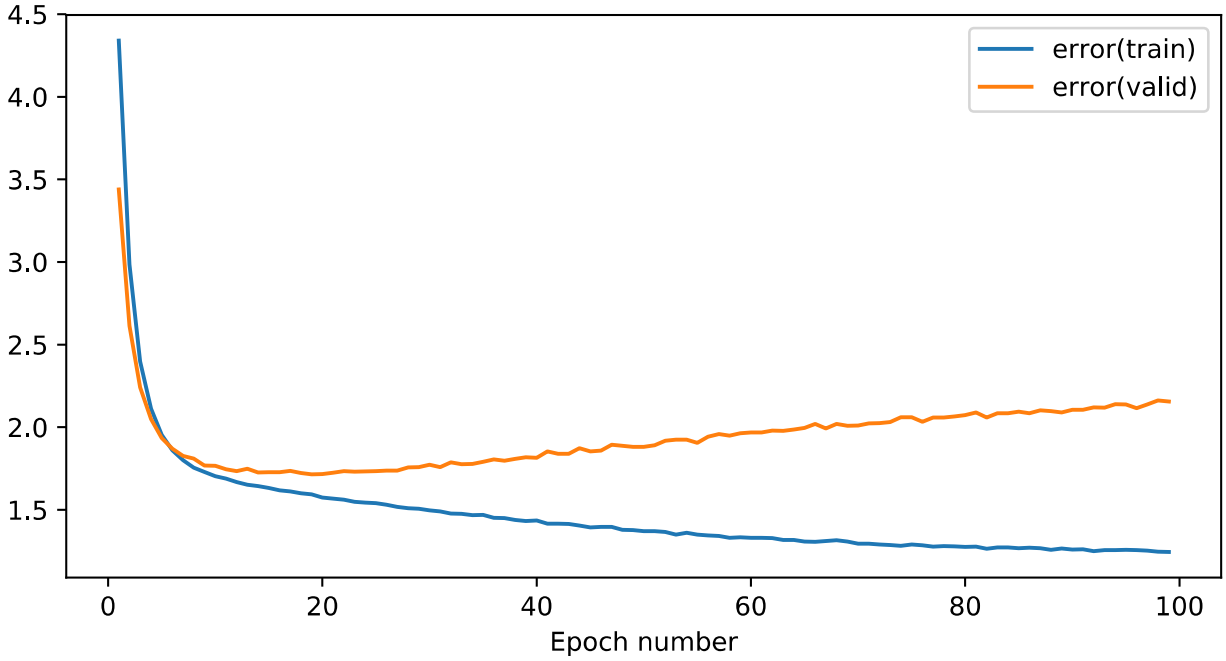
Plot 2: Convergence Plot of Bayesian Optimization for 35 epochs per call. The objective is the maximum validation accuracy. Displays the minimally achieved result for the objective function after every call

So we see that from a ~48% validation accuracy we managed to reach at ~50% through the usage of bayesian optimization. The dropout keep probabilities suggested as the best from the bayesian optimization are:

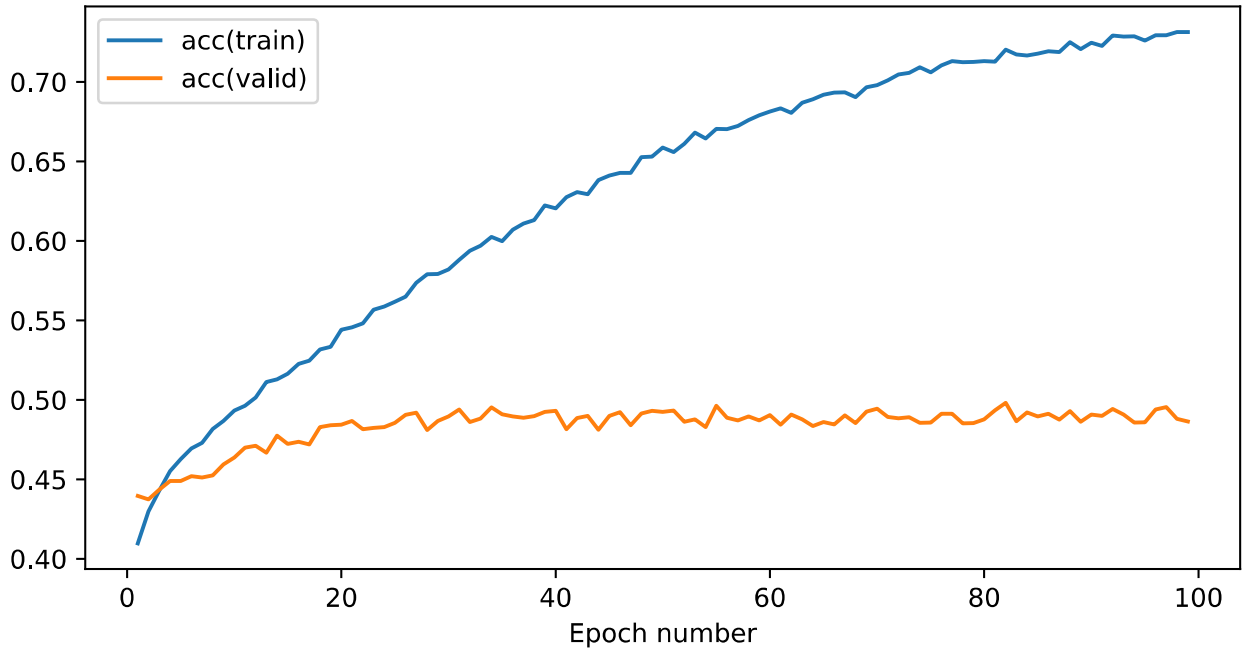
Best dropout keep probability for input layer: 0.7 (or 70%)

Best dropout keep probability for hidden layers: 0.83712466 (or ~84%)

Results by Training Classifier with best parameters



Plot 3: Training & Validation Error – Dropout Keep Prob Input 70% – Dropout Keep Prob Hidden Layers 84% - Layers Dimensionality [3000, 500, 500, 25, 10] – Batch Normalization – Tanh Activation Function – L2 regularization



Plot 4: Training & Validation Accuracy – Dropout Keep Prob Input 70% – Dropout Keep Prob Hidden Layers 84% - Layers Dimensionality [3000, 500, 500, 25, 10] – Batch Normalization – Tanh Activation Function – L2 regularization

Conclusion

We see that we have managed to get a steady validation accuracy performance for the final classifier with the same level of maximum validation accuracy as we had achieved with the dynamic dropout in coursework 3.

However this time there are no extra hyperparameters and there are no such severe oscillations in the plots.

On the other hand we have not achieved optimal regularization since we see a slow increase of the validation error after epoch 20 without being so severe that would impact the classification accuracy.