HW 1 Report MAS.S61 Personalized Machine Learning

Samuel Spaulding, Ishwarya Ananthabhotla, Nikhita Singh, Ishaan Grover, Akshay Mohan

Problem Statement

The report should include a discussion of how you have personalized your model, a short description of pros/cons of your approach, your architecture and design, your results, and how the choices of architecture and hyperparameters affected your model. The running code you used to generate the results should be submitted as well (to be commented and have the main function to rerun the experiments - training and prediction). Your code will mainly be used to verify that you were able to successfully implement your approach. After the submission, individual meetings will be organized to run the code together with TAs/Instructors.

Non-personalized Hyperparameter Selection

We started off by modifying the original code provided in *neural_net_practice.ipynb* to select different hyperparameters for the baseline model. We removed the personalized features (*subject_num*, *personal:age*, *personal:sex*, *test_time*) from the dataset, giving us 16 features as input to the non-personalized model. As a start, we tried 10,000 steps, 50,000 steps and 100,000 steps to see where performance would level off. 50,000 steps provided an efficient tradeoff between performance and run time.

Originally, we wanted to do a 2D grid search across 6 Architectures x 3 Learning Rates x 3 Weight Penalties x 3 Dropout Probabilities, running each model for 50,000 steps and averaging performance for each selection of hyperparameters across 10 runs. The values of each of these variables we chose is shown below.

- Architectures: [128x32, 64x32, 128x64x32, 32x32x32, 32x16x8, 16x16x16]
- LR: [0.1, .01, .005]
- WP : [0.1, .01, .05]
- DOP : [1.0, 0.9, 0.8]

However, one combination of variables within this grid search took 16.123 minutes to run. Since running this across every combination would take too long, we chose to limit our scope to 4 Architectures, 2 Learning Rates, 2 Weight Penalities, 2 Dropout Probabilities. We parallelized the process by running various combinations on different laptops. The 128x64x32 architecture performed the best overall. This architecture was evaluated across different learning rates, dropout probabilities and weight penalities. Results from the baseline and various combinations of hyperparameters are shown below.

	Baseline	Model 1	Model 2	Model 3	Model 4	Model 5*
Hidden Layer	128x32	128x64x32	128x64x32	128x64x32	128x64x32	128x64x32
Learning Rate	0.001	0.001	0.01	0.1	0.01	0.001
Weight Penalty	0.0	0.0	0.01	0.01	0.01	0.01
1 - (Dropout Rate)	1.0	1.0	0.8	0.9	0.9	0.9
Minibatch Size	25	25	25	25	25	25
Clip Gradients	**false**	**false**	**true**	**true**	**true**	**true**
Validation RMSE	**7.656558**	**8.2133**	**8.1467**	**9.701**	**7.955**	**7.178**

Comparing the baseline to Model 1, it is clear that depth alone does not improve the model. Adding additional free parameters in the form of depth without compensating via some form of regularization likely leads to overfitting and worse performance on the validation set.

Next we decided to increasing the learning rate and add regularization in the form of weight penalties and dropout. Increasing the learning rate too high ended up hurting our performance (compare Model 4 and Model 5), but regularization via clipped gradients, weight penalties, and dropout all impacted

our model performance positively (compare Model 2 and Model 4). Model 5 represents the best set of hyperparameters we found, evaluated by performance across the validation set. After selecting the hyperparameters for the non-personalized model, we tested it on the **Test** data, and achieved an RMSE of 7.23328, averaged over 3 trials.

Personalized Architecture

As an simple test to see whether personalization would be likely to improve performance, we trained the baseline model (Model 5 in the table above), this time using personalized features including subject number, age, sex, and test time. This *drastically* decreased the RMSE to <2, showing that even basic feature augmentation using personalized features can be beneficial to improving the model.

We chose a two-pronged approach to personalizing the neural net. First, we added the personalized features (age, sex, and test time) as a form of feature augmentation and second, we developed a multi-task learning architecture like the one we discussed in class, modeling the outputs for each individual separately. To our baseline model, we added another hidden layer of 8 nodes per person (each group of 8 nodes was fully connected to the 32 node layer), with each cluster of 8 nodes fully connected to a personal 2-node output layer. The final architecture we chose to use is shown in the diagram below:

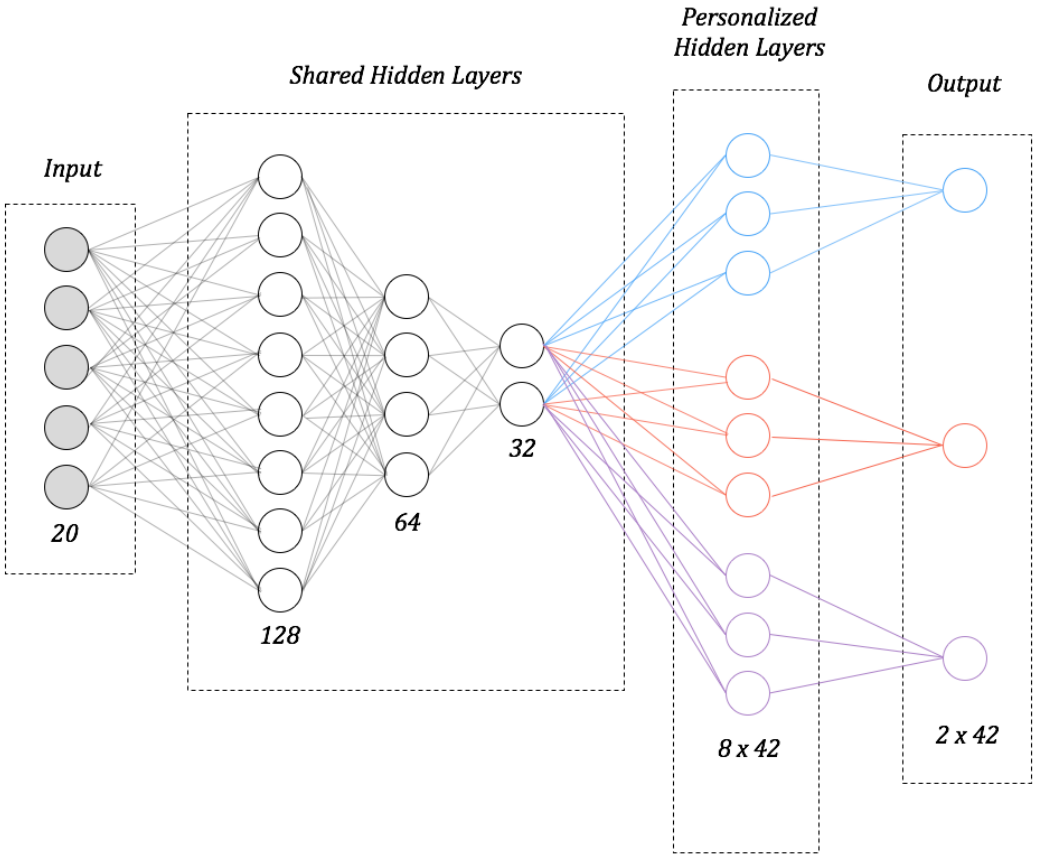

Figure 1: Personalized Architecture

Hyperparameters for the Personalized Model

After confirming that the Personalized Model was working, we decided to try tweaking the hyperparameters a little bit to see if we could get improved performance. We found that dialing up the learning rate to .01 substantially improved the performance, giving us a final personalized set of hyperparameters

shown below

	Personalized Model
Hidden Layer	128 x 64 x 32 x (8 x **42**) x (2 x **42**)
Learning Rate	0.01
Weight Penalty	0.01
1 - (Dropout Rate)	0.9
Minibatch Size	25
Clip Gradients	**true**

Final Results

Results reported here are from Subjects 1,5,7,10,15,18,25,29,35,40. We found that performance on this subset is generally slightly better than performance on the entire population, but not substantially.

Results from Personalized Neural Net

Subject #	Test RMSE	Mean Test label_motor_UPDRS	Mean Test label_total_UPDRS
1	2.903	30.285±1.050	38.364±1.729
5	1.906	33.290±0.965	43.860±1.538
7	0.755	16.267±0.877	23.338±1.212
10	0.298	13.213±1.562	19.777±1.018
15	1.759	13.978±0.732	19.888±1.384
18	0.351	6.000±0.028	7.372±0.037
25	3.084	32.252±2.862	51.300±3.471
29	1.454	24.326±0.355	32.709±1.146
35	1.965	34.468±0.130	52.504±0.355
40	0.821	15.896±0.172	26.552±1.368

Test mean RMSE	**1.530**
Test total RMSE	**1.8025**
Test total MAE	**.16**

Results from Baseline Neural Net

Subject #	Test RMSE	Mean Test label_motor_UPDRS	Mean Test label_total_UPDRS
1	1.255	32.724±2.372	42.058±3.461
5	2.371	30.797±2.151	41.033±2.807
7	1.749	16.439±0.880	23.778±1.028
10	1.640	14.757±0.952	20.439±1.633
15	1.873	13.182±2.340	19.068±3.341
18	3.271	7.411±1.092	10.864±2.010
25	2.401	30.660±2.986	47.410±3.484
29	1.822	25.305±0.746	32.748±1.275
35	4.180	35.188±2.074	50.265±3.548
40	1.334	16.884±0.663	24.997±0.677

Test mean RMSE	**2.19**
Test total RMSE	**2.343**
Test total MAE	**0.22**

We found that feature augmentation with personalized features significantly dropped root mean squared error. Moreover, we see that the personalized architecture for multitask learning further drops the root mean squared error

Files of interest

Please refer to `https://github.com/SamSpaulding/PML-DNN-hw1` for our source code.

The most relevant files we contributed to were:

- neural_net/personalized_neural_net.py, **a neural net for multi-task/personalized learning**
- neural_net/neural_net.py, **we added functions to allow for baseline testing on personalized data**
- data_funcs.py, **we added functions for selection, manipulation, and testing on personalized data**
- neural_net/personalized_practice.ipynb **an iPython notebook with examples of how to train, test, and analyze results from the personalized and baseline models**
- personalized_neural_net_2.py, **a second, similar, implementation of a neural net for multi-task/personalized learning**