HW6 report

Maša Kljun, 63170011

Results

Task 1

We implement this problem in `hw6.py` and implement it in a way that classification and regression ANNs share a lot of code. The only difference between the models is in the activation functions of the last layer and in the loss functions. In classification there are *n_unique* output neurons, where *n_unique* is the number of unique classes in the target variable, while in regression there is only one neuron in the output layer. The activation of the last layer in classification is the softmax function, which enables us to get probabilities of each class, while the activation of the last layer in regression is the identity function. As mentioned before, the second difference is in the loss functions. In classification we use cross entropy, while in regression we use mean squared error (MSE). Most of the other code (computing gradient, etc.) is shared among both models.

Task 2

We verify the gradient with the help of the following definition of the derivative:

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}. \tag{1}$$

We loop through the whole vector of weights and biases wb and in each iteration obtain vector $\bar{wb}^{(i)}$, where the i-th component of vector wb is changed by adding a small value $h = 1e-06$, while leaving other components of the vector as they are. We then use the above Equation 1 and compute the i-th component of the estimated gradient:

$$\nabla f^{(i)}(wb) = \frac{f(\bar{wb}^{(i)}) - f(wb)}{h}. \tag{2}$$

We compare the obtained component $\nabla f^{(i)}(wb)$ with the true value of derivative obtained by the backpropagation function $\nabla f_{bp}^{(i)}(wb)$. If the absolute difference between $\nabla f^{(i)}(wb)$ and $\nabla f_{bp}^{(i)}(wb)$ is smaller than predefined threshold $thresh = 1e-02$, we conclude the gradient for i-th component is correct. If gradient for all components is correct, we conclude the gradient is correct. We verify the gradient on the initial set of weights everytime we fit the model.

Task 3

We apply `housing2r` and `housing3` data sets to the regression and the classification models, respectively. We use 5-fold CV for both data sets in order to find the best values for regularization parameter λ and the number of hidden layers *units*. We choose some options for these two parameters manually, by checking which settings work, and then apply grid search with 5-fold cross-validation. We perform fitting of the final model and optimal parameter search on 80% of the data, and then evaluate the model on the remaining 20% of the data. Also note that we shuffle the data before 80/20 split, and after the split we scale features of train and test sets using StandardScaler from Scikit–Learn.

For the regression model we see from the CV that the best setting is $units : [10]$ (one hidden layer with 10 nodes) and $\lambda = 0.01$. We then evaluate the obtained model with the test set and get $MSE = 50.2$ with standard error 5.35. We compare the obtained results with the Support Vector Regression (SVR) with RBF kernel and the following settings: $\sigma = 5, \lambda = 1, \varepsilon = 1$ and get $MSE = 60.2$ with standard error 4.72. We can see that with the use of ANN we get better (lower) MSE than with the SVR with RBF kernel, which we set to the settings that gave good results in the HW4.

For the classification model we see from the CV that the best setting is $units : [5]$ (one hidden layer with 5 nodes) and $\lambda = 0.001$. We then evaluate the obtained model with the test set and get cross entropy $= 0.12$ with standard error 0.002. We compare the obtained results with the Multinomial Logistic Regression (MLR) and get cross entropy $= 0.14$ with standard error 0.002. We can see that with the use of ANN we once again get better (lower) cross entropy than with the MLR.

Task 4

Before performing the grid search to find the optimal parameters on the train set, we shuffle the train set, split it 80/20, and manually find some parameters that work. After having some initial guesses, we proceed with the exercise. The code for this last exercise can be found in the function `create_final_predictions()`. We shuffle the train data and scale the features of train (`train.csv.gz`) and test (`test.csv.gz`) data. We choose the best parameters using 5-fold CV which are the following: $units : [10, 10, 10]$ (three hidden layers with 10 nodes each) and $\lambda = 0.001$. We measure how much time does the process of grid search with 5-fold CV take and see that the time required for execution of 80 fitting procedures (5 fold of CV for each of 16 combinations of λ and *units*) is 114 minutes and 24 seconds. We also check how much time is required for the fitting of the final model (with the best parameters from CV) and see that the model is fitted in 3 minutes and 45 seconds. Note that in each fit we also numerically verify the gradient in the beginning. From the model evaluated on the train set we get cross entropy $= 0.54$. Additionally, we estimate the cross entropy on the test set by averaging results from CV and get cross entropy $= 0.58$, which is likely to be the better estimate (tested on unseen data). We provide the predictions in `final.txt`.

Note that throughout the report we report the results without stating uncertainty (although we know reporting uncertainty is a good practice), as the computation time for ANNs is really long as it is.