# Homework 8: Feature Learning

| Contact TA | Yiquan Xiao |
| --- | --- |
| Released date | Sunday, Nov. 12 |
| Submission deadline | Midnight, Tuesday, Nov. 28 |

## Task Descriptions

In this last homework assignment, you have only one programming task. Complete the task by filling in the TODOs of the provided code template: [hw8.py](https://drive.google.com/file/d/1pmgx4GQpNt3jQ_pbiIDqapo_xpp38qyf/view?usp=sharing). Feel free to define additional functions if they make your code cleaner.

### Task 1: Classification of diabetes with k-fold cross-validation

Perform K-fold cross-validation using a linear model and the *l1* regularizer over

a popular **two-class classification** genomics dataset consisting of P = 72 datapoints, each of which has input dimension N = 7128. You can download the dataset [here](https://drive.google.com/file/d/1x__kaWjelaHTBqwGIhRQ3dfp3s_CVfH2/view?usp=sharing).

This will tend to produce a sparse predictive linear model, which is helpful in determining a small number of genes that correlate with the output of this two-class classification dataset (which is whether each individual represented in the dataset has diabetes or not).

**Please approach the problem as you are a data analyst who is trying to analyze the data:** if your boss gives you this dataset, how would you proceed with a linear classification model and report the insights you can draw from the data to your boss? Come up with the hyperparameters (e.g., **learning rate** and **the penalty of the regularizer**) and necessary preprocessing (e.g., normalization) that you think will help you get insights from the data. **Justify your choices in the report. And report your observations.**

After preprocessing the data, you should first perform K-Fold Cross Validation on the training set to find the best set of hyperparameters. Report the average validation accuracy of the model with the best set of hyperparameters during the K-Fold Cross Validation process.

Then, use the best set of hyperparameters you found to train the model on the entire training set and report the training accuracy. This will be your final model. And last, you should evaluate your final model on the testing set and report the test accuracy.   
  
Given the final model, **report the 5 most influential genes**, in descending order. To do this, you can identify the top 5 weights with the largest absolute values (exclusion bias).

For example, if w = [10, 1, -8, 1, 4, 6, 2, 2, -3, -5, 7], then the gene are the 5 genes with the greatest impact.

Background: Genome-wide association studies (GWAS) aim at understanding the connections between tens of thousands of genetic markers (input features), taken from across the human genome of several subjects, with medical conditions such as high blood pressure, high cholesterol, heart disease, diabetes, various forms of cancer, and many others (see [Figure 11.52 in the textbook](https://github.com/jermwatt/machine_learning_refined/blob/main/sample_chapters/2nd_ed/chapter_11.pdf)). These studies typically involve a relatively small number of patients with a given affliction (as compared to the very large dimension of the input). As a result, regularization based cross-validation is a useful tool for learning meaningful (linear) models for such data. Moreover, using a (sparsity-inducing) regularizer like the l1 norm can help researchers identify the handful of genes critical to the affliction under study, which can both improve our understanding of it and perhaps provoke development of gene-targeted therapies.

Tips:

* You should split your dataset into train/test splits. You can decide what’s a reasonable train/test split ratio.
* You should conduct K-fold cross validation on the train split by further splitting the train split into k-folds. You can decide the number of folds you think will make sense.
* You should use the *validation accuracy* to decide the best set of hyperparameters – that is, the best hyperparameters should be the one that achieves the lowest validation accuracy (on your validation split)
* For the hyperparameters to tune, you can tune (1) the learning rate and (2) the penalty parameter.
* You should be able to observe **averaged validation accuracy (across folds)** to be higher than 90% if your hyperparameter is proper.
* If you have NaN issues, it is probably because your initial weights are too large such that you encounter overflow problems. Try to decrease your initial weight magnitude.
* When the lambda (the penalty parameter) is set to be too large and yet your learning rate is not diminishing, you will encounter divergence (or your cost won’t decrease). In this case, you can try to decrease your learning rate. Below is the code snippet that works for me. I choose to decrease the learning rate by 0.1 whenever the cost stops decreasing anymore in the last 200 iterations. Other heuristics could also work but make sure your model converges!

| # When cost stops decreasing, decrease learning rate by 0.1   if k % 200 == 0 and np.mean(cost\_history[-100:-50]) - np.mean(cost\_history[-50:]) < 1e-8:   if alpha > 1e-3:   alpha = alpha \* 0.1   # early stop as we have achieved the lowest alpha   else:   print(f"early stop at k = {k}")   break |
| --- |

## 

## Deliverables

You should report:

* Choice of K (used for K-fold cross validation)
* Choice of normalization method (no normalization/Standard Normalization/PCA sphering) and other preprocessing method used (if any)
* Cost function used
* Hyperparameter setting of your final model (learning rate, penalty of the regularizer, and other hyperparameters if any)
* Average validation accuracy of the model with the best set of hyperparameters during the K-Fold Cross Validation process.
* Plot of cost vs. iteration of your final model **over the entire training set**
* Accuracy of your final model **on the testing set**
* The 5 most influential genes, in descending order
  + For example, if w = [10, 1, -8, 1, 4, 6, 2, 2, -3, -5, 7], then the gene are the 5 genes with the greatest impact