CSCI567 2017 Homework Assignment 1

Submission instructions

Due date and time Sept. 24th (Sunday), 11:59pm

Github submission Please create a directory 'homework1/' at the base level of your git repository. We will expect to find your solutions there. You will turn in 4 files under 'homework1/' directory.

- a PDF written report named as `firstname_lastname_USCID.pdf`, which contains your solutions for questions in Sect. 1 and Sect. 2.
- python script `linear_regression.py`, which contains your code for questions in Sect. 4.
- python script `knn.py`, which contains your code for questions in Sect. 5.
- python script `logistic.py`, which contains your code for questions in Sect. 6.

For your written report, your answers must be typeset with LaTeX and generated as a PDF file. No handwritten submission will be permitted. There are many free integrated LaTeX editors that are convenient to use, please google search them and choose the one(s) you like the most. This `http://www.andy-roberts.net/writing/latex` seems like a good tutorial.
For your programming solutions, please use python3 on the virtual machine (VM) and write your own solutions from scratch. Do not import any packages yourself. We will run the 3 aforementioned scripts on VM to grade your homework.

Collaboration You may discuss with your classmates. However, you need to write your own solutions and submit separately. Also in your written report, you need to list with whom you have discussed for each problem. Please consult the syllabus for what is and is not acceptable collaboration.

Algorithmic component

1 Linear Regression

Review In the lectures, we have described the least mean square (LMS) solution for linear regression as

$$\boldsymbol{w}^{\mathrm{LMS}} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\boldsymbol{y} \tag{1}$$

where X is our design matrix (N rows, $D+1$ columns) and $\boldsymbol{y}$ is the N-dimensional column vector of the true values in the training data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. We have assumed that each row of X has been appended with the constant 1 in Eqn. 1.

Question 1.1 $X^{\mathbf{T}}X$ is not invertible [Recommended maximum time spent: 15 mins]

In the lecture, we have described a practical challenge for linear regression. Specifically, the least square solution is not possible when $X^{\mathrm{T}}X$ is not invertible. Please use a concise mathematical statement (**in one sentence**) to summarize the relationship between the training data X and the dimensionality of $\boldsymbol{w}$ when this bad scenario happens.
What to submit: your one sentence answer in the written report.

Question 1.2 Bias solution [Recommended maximum time spent: 60 mins]

At the end of lecture 3, we mentioned that under certain assumption, the bias b has a solution being the mean of the samples

$$b^* = \frac{1}{N}\mathbf{1}_N^{\mathrm{T}}\boldsymbol{y} = \frac{1}{N}\sum_n y_n, \tag{2}$$

where $\mathbf{1}_N = [1, 1, \ldots, 1]^{\mathrm{T}}$ is a N-dimensional all one column vector.

We can prove that it is true when $D = 0$, i.e. we ignore the features such that the design matrix is a column of 1's, by the following procedure.

$$b^* = \arg\min_b \|\boldsymbol{y} - b\mathbf{1}_N\|^2 \qquad \text{Residual sum of squares} \tag{3}$$

$$\mathbf{1}_N^{\mathrm{T}}(\boldsymbol{y} - b^*\mathbf{1}_N) = 0 \qquad \text{Taking derivatives w.r.t } b \tag{4}$$

$$b^* = \frac{1}{N}\mathbf{1}_N^{\mathrm{T}}\boldsymbol{y} \tag{5}$$

In this question, we would like you to generalize the proof above to arbitrary D and arrive at something similar to Eqn. 2.

Please follow the three-step recipe: 1) write out the residual sum of squares objective function; 2) take derivatives w.r.t. the variable you are interested in and set the gradient to 0; 3) solve for b^* and conclude. You will find out that you need one more condition to arrive at Eqn. 2, which is

$$\frac{1}{N}\sum_n x_{nd} = 0, \quad \forall d = 1, 2, \ldots, D. \tag{6}$$

This is to center the input data (excluding the appended constant) to be zero mean, which is a common preprocessing technique people use in practice. There is a simple explanation to Eqn. 6 — if the feature values are zero on average, the average response can only be caused by the bias (Eqn. 2).
What to submit: your fewer-than-10-line proof in the written report.

2 Logistic Regression

Review Recall that the logistic regression model is defined as:

$$p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) \tag{7}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

Given a training set $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, where $y_n \in \{0, 1\}$, we will minimize the cross entropy error function to solve for $\boldsymbol{w}$ and b.

$$\min_{\boldsymbol{w},b} \mathcal{E}(\boldsymbol{w}, b) = \min_{\boldsymbol{w},b} -\sum_n \{y_n \log[p(y_n = 1|\boldsymbol{x}_n)] + (1 - y_n) \log[p(y_n = 0|\boldsymbol{x}_n)]\} \tag{9}$$

$$= \min_{\boldsymbol{w},b} -\sum_n \{y_n \log \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + b) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + b)]\} \tag{10}$$

Question 2.1 Bias solution [Recommended maximum time spent: 45 mins]

Consider if one does not have access to the feature $\boldsymbol{x}$ of the data, and is given a training set of $\mathcal{D} = \{y_n\}_{n=1}^{N}$, where $y_n \in \{0, 1\}$. What would be the optimal logistic regression classifier in that case? What is the probability that a test sample is labeled as 1? You could also compare your solution with Eqn. 2 in Question 1.2.

Specifically, please write out the objective function as in Eqn. 10. And solve for the optimal bias term b^*.

What to submit: your fewer-than-5-line derivation and your formula for the optimal bias in the written report.

Programming component

3 Pipeline and Datasets

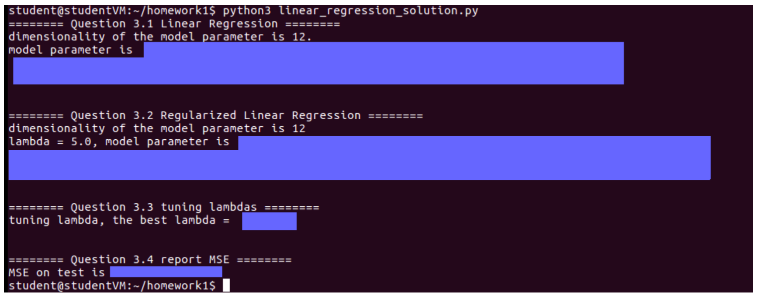
In this section, we will first explain the general coding pipeline you will use in Sect. 4, 5 and 6, and then describe the datasets.

3.1 Pipeline

A standard machine learning pipeline usually consists of three parts. 1) Load and preprocess the data. 2) Train a model on the training set and use the validation set to tune hyperparameters. 3) Test the final model and report the result. In this assignment, we will provide you a template for each section (`linear_regression.py, knn.py`, and `logistic.py`), which follows this 3-step pipeline. We provide step 1's data loading and preprocessing code, and define step 3's output format to report the results. You will be asked to implement step 2's algorithms to complete the pipeline. Do not make any modification to our implementations for step 1 and 3.

Please do not import packages that are not listed in the provided scripts. Follow the instructions in each section strictly to code up your solutions. **DO NOT CHANGE THE OUTPUT FORMAT. DO NOT MODIFY THE CODE UNLESS WE INSTRUCT YOU TO DO**

SO. A homework solution that does not match the provided setup, such as format, name, initializations, etc., **will not** be graded. It is your responsibility to make sure that your code runs with python3 on the VM.



```
student@studentVM:~/homework1$ python3 linear_regression_solution.py
======== Question 3.1 Linear Regression ========
dimensionality of the model parameter is 12.
model parameter is [redacted]


======== Question 3.2 Regularized Linear Regression ========
dimensionality of the model parameter is 12
lambda = 5.0, model parameter is [redacted]


======== Question 3.3 tuning lambdas ========
tuning lambda, the best lambda = [redacted]


======== Question 3.4 report MSE ========
MSE on test is [redacted]
student@studentVM:~/homework1$ 
```

Figure 1: Example output

Example output For `linear_regression.py` in Sect. 4, you should be able to run it on VM and see output similar to Fig. 1.

3.2 Datasets

Regression Dataset The UCI Wine Quality dataset lists 11 chemical measurements of 4898 white wine samples as well as an overall quality per sample, as determined by wine connoisseurs. See **winequality-white.csv**. We split the data into training, validation and test sets in the preprocessing code. You will use linear regression to predict wine quality from the chemical measurement features.

Classification Datasets We describe three datasets that will be used for classification problems in this homework. Each dataset corresponds to a JSON file named as **\$dataset\$.json**. JSON is a lightweight data-interchange format, similar to a dictionary. After loading a dataset, you can access its training, validation, and test splits using the keys 'train', 'valid', and 'test', respectively. For example, suppose we load **mnist_subset.json** to the variable x. Then, $x['train']$ refers to the training set of **mnist_subset**. This set is a list with two elements: $x['train'][0]$ containing the features of size N (samples) $\times D$ (dimension of features), and $x['train'][1]$ containing the corresponding labels of size N.

Next we will describe each one of the three datasets in more detail.

- **toydata1** includes 240 data instances with binary labels that are linearly separable. The data is split into a training set and a test set. You can look up training and test sets in **toydata1.json** with $['train']$ and $['test']$, respectively.
- **toydata2** includes 240 data instances with binary labels that are not linearly separable. The data is split into a training set and a test set. You can look up training and test sets in **toydata2.json** with $['train']$ and $['test']$, respectively.
- **mnist_subset**: MNIST is one of the most well-known datasets in computer vision, consisting of images of handwritten digits from 0 to 9. We will be working with a subset of the official

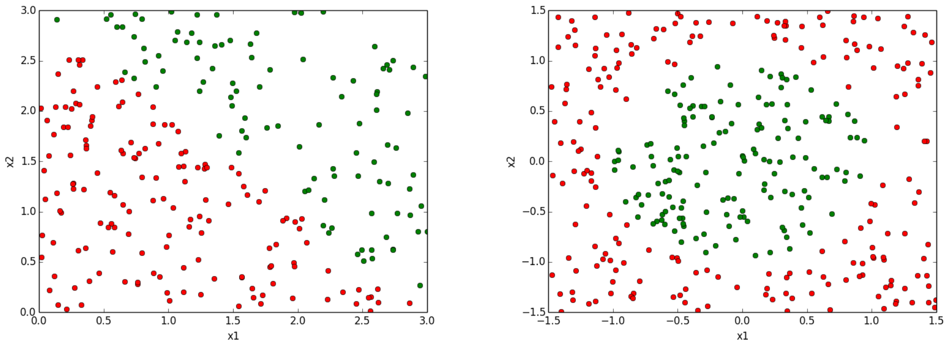


Figure 2: Left: **toydata1**, linearly separable; Right: **toydata2**, not linearly separable

version of MNIST. In particular, we randomly sample 700 images from each category and split them into training, validation, and test sets, which can be reached in **mnist_subset.json** via keys $['train']$, $['valid']$ and $['test']$, respectively.

4 Linear Regression

You are asked to implement 4 python functions for linear regression. The input and output of the functions are specified in `linear_regression.py`. You should be able to run `linear_regression.py` after you finish the implementation. Note that we have already appended the column of 1's to the feature matrix, so that you do not need to modify the data yourself.

Question 4.1 Linear regression Implement linear regression and return the model parameters.
What to submit: fill in the function `linear_regression_noreg(X, y)`.

Question 4.2 Regularized linear regression To address the challenge described in Question 1.1, as well as other issues such as overfitting, we add regularization. For now, we will focus on L_2 regularization. In this case, the optimization problem is:

$$\boldsymbol{w}^{\lambda} = \arg\min_{\boldsymbol{w}} ||X\boldsymbol{w} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_2^2 \tag{11}$$

where $\lambda \geq 0$ is a hyper-parameter used to control the complexity of the resulting model. When $\lambda = 0$, the model reduces to the usual (unregularized) linear regression. For $\lambda > 0$ the objective function balances between two terms: (1) the data-dependent quadratic loss function $||X\boldsymbol{w} - \boldsymbol{y}||_2^2$, and (2) a function of the model parameters $||\boldsymbol{w}||_2^2$.

Implement your regularized linear regression algorithm.
What to submit: fill in function `regularized_linear_regression(X, y, λ)`.

Question 4.3 Tuning the regularization hyper-parameter Use the validation set to tune the regularization parameter $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2\}$. We select the best one that results in the lowest mean square error on the validation set.
What to submit: fill in the function `tune_lambda(Xtrain, ytrain, Xval, yval, lambds)`.

Question 4.4 Mean square error Report the mean square error of the model on the given test set.
What to submit: fill in the function `test_error(w, X, y)`.

5 k Nearest Neighbors

Review In the lecture, we define the classification rule of the k-nearest neighbors (kNN) algorithm for an input example x as

$$v_c = \sum_{x_i \in \mathrm{knn}(x)} \mathbb{1}(y_i == c), \forall c \in [C] \tag{12}$$

$$y = \arg\max_{c \in [C]} v_c \tag{13}$$

where $[C]$ is the set of classes.

A common distance measure between two samples x_i and x_j is the Euclidean distance:

$$d(x_i, x_j) = \|x_i - x_j\|_2 = \sqrt{\sum_d (x_{id} - d_{jd})^2}. \tag{14}$$

You are asked to implement 4 python functions for kNN. The input and output are specified in `knn.py`. You should be able to run `knn.py` after you finish the implementation.

Question 5.1 Distance calculation Compute the distance between test data points in X and training data points in X_{train} based on Eqn. 14.
What to submit: fill in the function `compute_distances(Xtrain, X)`.

Question 5.2 kNN classifier Implement kNN classifier based on Eqn. 13. Your algorithm should output the predictions for the test set. *Important:* When there are ties among predicted classes, you should return the class with the smallest value. For example, when $k = 5$, if the labels of the 5 nearest neighbors happen to be $1, 1, 2, 2, 7$, your prediction should be the digit 1.
What to submit: fill in the function `predict_labels(k, ytrain, dists)`.

Question 5.3 Report the accuracy The classification accuracy is defined as:

$$\text{accuracy} = \frac{\#\text{ of correctly classified test examples}}{\#\text{ of test examples}} \tag{15}$$

(The accuracy value should be in the range of $[0, 1]$)
What to submit: fill in the code for function `compute_accuracy(y, ypred)`.

Question 5.4 Tuning k Find k among $[1, 3, 5, 7, 9]$ that gives the best classification accuracy on the validation set.
What to submit: fill in the code for function `find_best_k(K, ytrain, dists, yval)`.

6 Logistic Regression

You are asked to finish 3 python functions for logistic regression to solve the binary classification problem. The input and output are specified in `logistic.py`. You should be able to run `logistic.py` after you finish the implementation.

Question 6.1 Logistic regression classifier Find the optimal parameters for logistic regression using gradient descent. The objective is the cross-entropy error function described in class and in Eqn. 10.

We have initialized $\boldsymbol{w}$ and b to be all 0s (please do not modify this initialization procedure.). At each iteration, we compute the average gradients from all training samples and update the parameters using the chosen step size (or learning rate). We have also specified values for the step size and the maximum number of iterations (please do not modify those values).
What to submit: fill in the function
`logistic_train(Xtrain, ytrain, w, b, step_size, max_iterations)`.

Question 6.2 Test accuracy After you train your model, run your classifier on the test set and report the accuracy, which is defined as:

$$\frac{\text{\# of correctly classified test examples}}{\text{\# of test examples}}$$

What to submit: fill in the function `logistic_test(Xtest, ytest, w, b)`.

Question 6.3 Feature transformation **toydata2** is not linearly separable and logistic regression does not perform well. Let's try a simple modification to make it work better! Take element-wise square over the features of both $['train']$ and $['test']$ of **toydata2**. That is, each element in the feature matrix becomes the square of that element. Repeat the training process and report the final test accuracy. You should see the big difference between the two experiments.
What to submit: fill in the function `feature_square(Xtrain, Xtest)`.