[CMPUT 466/566] Machine learning

Coding Assignment 1

Lili Mou

Submission Information

The student should submit a zip file containing a pdf report and all the code which should replicate the results.

Problem 1 [50%]

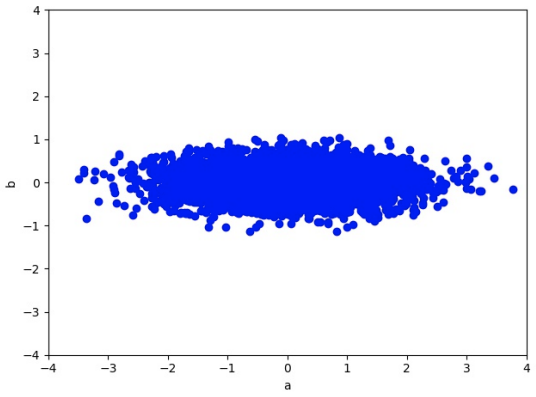
Download: Codebase

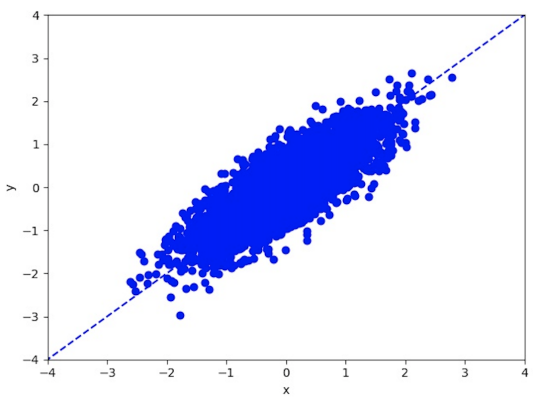
In this coding problem, we will implement the closed-form solution to linear regression. The student will use matrix-vector representations for data representation and get some experience with data visualization. In addition, the student will explore the phenomenon of "regression to the mean."

We generate data with two steps:

1. Independently generate $a \sim \mathcal{N}(0, \sigma_a^2)$ and $b \sim \mathcal{N}(0, \sigma_b^2)$
2. Rotate $(a, b)^\top$ to a degree (e.g., $45°$) and obtain $(x, y)^\top$

The data generation is given by the `generate_data(M, var1, var2, degree)` function. After data generation, we will obtain these two plots of data. In the second plot, the dashed line is where data is mostly generated around.





Now, we would like to do two linear regressions:

1. Predict y from x (denoted by x2y), and
2. Predict x from y (denoted by y2x).

To accomplish this, the student needs to implement the `leastSquares(X, Y)` function. Note: In the function `leastSquares(X, Y)`, X is simply the input and Y is simply the output. It is not related to x2y regression or y2x regression.

Then, we would like to use the learned models x2y and y2x to predict a series of points in (-4, 4) as input, and plot the input and the predicted output.

To accomplish this, the student needs to

1. Implement feature augmentation, i.e., each data sample is concatenated with a dummy feature 1. Notice the augmented feature during prediction has to comply with that in training.
2. Implement the prediction function `model(X, w)`, where the definition of `w` should comply with that in the `leastSquares` function.
3. Plot the learned models. Notice that a linear regression from $\mathbb{R}$ to $\mathbb{R}$ is just a line. The student is required to plot x2y regression in red and y2x regression in green. And, these two models MUST be plotted in the same x-y coordinate system, with x being the horizontal axis and y being the vertical axis.

Submission:

In the PDF report (the only accepted format), the student should report:

1) [20% for correct implementation of linear regression] With settings M = 5000, var1 = 1, var2 = 0.3, degree = 45, report the weight and bias for x2y and y2x regressions. The submission should be four numbers in two rows:

   w_x2y b_x2y
   w_y2x b_y2x

   Note: getting the above numbers correctly is a necessary (not sufficient) condition for correct implementation.

2) [10%] Three plots of regression models in a row, each with var2 = 0.1, 0.3, 0.8, respectively. (Other settings remain intact: M = 5000, var1 = 1, degree = 45).

3) [5%] A description of the phenomena found in 1) and 2).

4) [15%] We now set var2 = 0.1, but experiment with different rotation degrees. The student should design a controlled experimental protocol, plot three figures in a row with different settings, and describe the findings.

Note: For 1) and 2), only give the result. Do NOT explain anything, because the setting is clear in the problem description. For 3) and 4), do NOT try to write an over-lengthy description merely to get high marks. On the contrary, an unnecessarily long report indicates poor representation skills and may result in mark deduction.

Lessons learned:

This experiment tells us that, even for the same data $\{(x^{(i)}, y^{(i)})\}_{i=1}^{M}$, predicting y from x is different from predicting x from y. This verifies what we learned in the first lecture: Formulating a machine learning task as a supervised or an unsupervised task, or how can we formulate a supervised task really **depends on the task itself**, i.e., what the goal of your prediction is.

Problem 2 [50%]

Download: 2a, 2b, 2c, dataset

In this problem, we will explore gradient descent optimization for linear regression, applied to the Boston house price prediction. Download the dataset and we can load it by

```
import pickle as pickle
with load(<REPLACE/PATH/TO/FILE/filename.pkl>, “rb”) as f:
    (X, y) = pickle.load(f)
```

where pickle is a popular Python module for serializing and de-serializing Python object structure. Note that the above code is already included in the codebase and will load the dataset. Finally, we shall again use linear algebra routines (e.g., numpy) for the assignment.

The dataset contains 506 samples, each with 13 features. We first randomly shuffle all samples, and then take 300 samples as the training set, 100 samples as the validation test, and 106 as the test set.

We normalize features and output by

$$\tilde{x}_i = \frac{x_i - \text{mean}(x_i)}{\text{std}(x_i)}$$

$$\tilde{t} = \frac{t - \text{mean}(t)}{\text{std}(t)}$$

We use mean square error as our ***loss*** to train our model. The measure of success, however, is the mean of the absolute difference between the predicted price and true price. Here, we call

the measure of success the ***risk*** or ***error***. This reflects how much money we would lose for a bad prediction. The lower, the better.

In other words, the training loss is

$$J = \frac{1}{2M}\sum_{m=1}^{M}(\widetilde{y}^{(m)} - \widetilde{t}^{(m)})^2$$

where we compute loss on the normalized output $\widetilde{t}^{(m)}$ and the prediction $\widetilde{y}^{(m)}$.

The measure of success (the lower, the better) is

$$E = \frac{1}{M}\sum_{m=1}^{M}|y^{(m)} - t^{(m)}|$$

Here, the risk is defined on the original output (thinking of it's the real money).

Notice that we will use mini-batch gradient descent, and thus, M should be the number of samples in a batch.

Submission:

(a) [30%] We implement the train-validation-test framework, where we train the model by mini-batch gradient descent, and validate model performance after each epoch. After reaching the maximum number of iterations, we pick the epoch that yields the best validation performance (the lowest risk), and test the model on the test set.

Without changing default hyperparameters, we report three numbers

1. The number of epoch that yields the best validation performance,
2. The validation performance (risk) in that epoch, and
3. The test performance (risk) in that epoch.

and two plots:

1. The learning curve of the training loss, and
2. The learning curve of the validation risk.

where x-axis is the number of epochs, and y-axis is training loss and validation risk, respectively.

(b) [10%] We now explore non-linear features in the linear regression model. In particular, we adopt point-wise quadratic features. Suppose the original features are $(x_1, x_2, \cdots, x_d)^\top$. We now extend it as $(x_1, x_2, \cdots, x_d, x_1^2, x_2^2, \cdots, x_d^2)^\top$.

At the same time, we tune ℓ_2-penalty to prevent overfitting by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha\big(\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) + \lambda_{\text{decay}} \boldsymbol{w}\big)$$

The hyperparameter λ_{decay} should be tuned from the set {3, 1, 0.3, 0.1, 0.03, 0.01}.

Report the best hyperparameter λ_{decay}, i.e., the one yields the best performance, and under this hyperparameter, the three numbers and two plots required in Problem 2(a).

(c) [10%] Ask a meaningful scientific question on this task by yourself, design an experimental protocol, report experimental results, and draw a conclusion.