Machine Learning:
Assignment Sheet #6

Due on March 22, 2022 at 10:00

Group HB

Henri Sota, Enis Mustafaj

Problem 6.1

Consider the following training data:

$$\mathcal{T} = \left\{((-1,0)^\top, 3), ((1,3)^\top, -1), ((-2,1)^\top, 0), ((0,4)^\top, 2)\right\}$$

We want to compute the predictor from linear regression by least squares. However, instead of using Theorem 4.1, we use batch gradient descent to compute the coefficient vector $\hat{\beta}$. Choose the initial guess $\beta^{(0)} = (0,0,1)^\top$ and calculate the first two steps of batch gradient descent with learning rate $\eta = 0.25$.
To calculate the coefficients of each step, we use the formula for the batch gradient descent

$$\beta^{(n+1)} = \beta^{(n)} - \eta \cdot \sum_{i=1}^{N} \nabla L(y_i, f(x_i))$$

First, we calculate the partial derivatives of the loss function with respect to coefficient vector:

$$\nabla L(y_i, f(x_i)) = \begin{pmatrix} \frac{\partial L(y,f(x))}{\beta_0} \\ \frac{\partial L(y,f(x))}{\beta_1} \\ \frac{\partial L(y,f(x))}{\beta_2} \end{pmatrix} = \begin{pmatrix} -2 \cdot (y - \beta_0 - \beta_1 \cdot x_0 + \beta_2 \cdot x_2) \\ -2 \cdot (y - \beta_0 - \beta_1 \cdot x_0 + \beta_2 \cdot x_2) \cdot x_1 \\ -2 \cdot (y - \beta_0 - \beta_1 \cdot x_0 + \beta_2 \cdot x_2) \cdot x_2 \end{pmatrix}$$

By applying this to the general formula, we get:

$$\beta^{(n+1)} = \beta^{(n)} + 2 \cdot \eta \cdot \sum_{i=1}^{N} \begin{pmatrix} (y_i - \beta_0 - \beta_1 \cdot x_{0(i)} - \beta_2 \cdot x_{2(i)}) \\ (y_i - \beta_0 - \beta_1 \cdot x_{0(i)} - \beta_2 \cdot x_{2(i)}) \cdot x_{1(i)} \\ (y_i - \beta_0 - \beta_1 \cdot x_{0(i)} - \beta_2 \cdot x_{2(i)}) \cdot x_{2(i)} \end{pmatrix}$$

We substitute the values of training data in the formula to compute the coefficients from the first 2 iteration steps.

$$\beta^1 = \beta^0 + \frac{1}{2} \cdot \left[\begin{pmatrix} 3 \\ -3 \\ 0 \end{pmatrix} + \begin{pmatrix} -4 \\ -4 \\ -12 \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix} + \begin{pmatrix} -6 \\ 0 \\ -24 \end{pmatrix} \right]$$

$$\beta^1 = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} + \frac{1}{2} \cdot \begin{pmatrix} -8 \\ -5 \\ -37 \end{pmatrix} = \begin{pmatrix} -4 \\ -\frac{5}{2} \\ -\frac{35}{2} \end{pmatrix}$$

To compute the second step we use the values of the new coefficients derived from above.

$$\beta^2 = \beta^1 + \frac{1}{2} \cdot \left[\begin{pmatrix} \frac{9}{2} \\ -\frac{9}{2} \\ 0 \end{pmatrix} + \begin{pmatrix} 58 \\ 28 \\ 174 \end{pmatrix} + \begin{pmatrix} \frac{33}{2} \\ -33 \\ \frac{33}{2} \end{pmatrix} + \begin{pmatrix} 76 \\ 0 \\ 304 \end{pmatrix} \right]$$

$$\beta^2 = \begin{pmatrix} -4 \\ \frac{-5}{2} \\ \frac{-35}{2} \end{pmatrix} + \frac{1}{2} \cdot \begin{pmatrix} 155 \\ \frac{41}{2} \\ \frac{849}{2} \end{pmatrix} = \begin{pmatrix} \frac{147}{2} \\ \frac{31}{2} \\ \frac{919}{4} \end{pmatrix}$$

Problem 6.2

In this task, we consider again the training data

$$\mathcal{T} = \left\{((4,1)^\top, 2), ((2,8)^\top, -14), ((1,0)^\top, 1), ((3,2)^\top, -1)\right\}$$

However, this time we do not use the linear model. Instead we use the quadratic model

$$f(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 \beta_5 X_2^2$$

and stick to the L_2 loss.

a) Derive the gradient $\nabla_\beta L_2(y_i, f(\mathbf{x_i}))$ of the loss with respect to the coefficient vector.

$$\nabla_\beta L_2(y_i, f(\mathbf{x_i})) = \begin{pmatrix} \frac{\partial L_2(y_i,f(\mathbf{x_i}))}{\beta_0} \\ \frac{\partial L_2(y_i,f(\mathbf{x_i}))}{\beta_1} \\ \frac{\partial L_2(y_i,f(\mathbf{x_i}))}{\beta_2} \\ \frac{\partial L_2(y_i,f(\mathbf{x_i}))}{\beta_3} \\ \frac{\partial L_2(y_i,f(\mathbf{x_i}))}{\beta_4} \\ \frac{\partial L_2(y_i,f(\mathbf{x_i}))}{\beta_5} \end{pmatrix} = \begin{pmatrix} -2 \cdot (y_1 - (\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \beta_4 \cdot x_1^2 + \beta_5 \cdot x_2^2)) \\ -2 \cdot (y_1 - (\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \beta_4 \cdot x_1^2 + \beta_5 \cdot x_2^2)) \cdot x_1 \\ -2 \cdot (y_1 - (\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \beta_4 \cdot x_1^2 + \beta_5 \cdot x_2^2)) \cdot x_2 \\ -2 \cdot (y_1 - (\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \beta_4 \cdot x_1^2 + \beta_5 \cdot x_2^2)) \cdot x_1 x_2 \\ -2 \cdot (y_1 - (\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \beta_4 \cdot x_1^2 + \beta_5 \cdot x_2^2)) \cdot x_1^2 \\ -2 \cdot (y_1 - (\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \beta_4 \cdot x_1^2 + \beta_5 \cdot x_2^2)) \cdot x_2^2 \end{pmatrix}$$

b) Use the initial guess $\beta^{(0)} = \mathbf{0}$ and the learning rate $\eta = 0.1$ and perform the first two steps of stochastic gradient descent. Recall that in stochastic gradient descent, we at some point pick random samples from the training set. In order to get a unique solution in this task, we assume that a random selection of samples from the training set would give you the training samples in their original order.

To calculate the coefficients we use the formula:

$$\beta^{(n+1)} = \beta^{(n)} + 2 \cdot \eta \cdot X^T \cdot (y - X \cdot \beta^n)$$

For the first step, we calculate:

$$X = \begin{pmatrix} 1 & 4 & 1 & 4 & 16 & 1 \end{pmatrix} \qquad X^T = \begin{pmatrix} 1 \\ 4 \\ 1 \\ 4 \\ 16 \\ 1 \end{pmatrix} \qquad y = 2$$

$$\beta^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{5} \cdot \begin{pmatrix} 1 \\ 4 \\ 1 \\ 4 \\ 16 \\ 1 \end{pmatrix} \cdot \left(2 - \begin{bmatrix} 1 & 4 & 1 & 4 & 16 & 1 \end{bmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} \frac{2}{5} \\ \frac{8}{5} \\ \frac{2}{5} \\ \frac{8}{5} \\ \frac{32}{5} \\ \frac{2}{5} \end{pmatrix}$$

For the second step, we calculate:

$$X = \begin{pmatrix} 1 & 2 & 8 & 16 & 4 & 64 \end{pmatrix} \qquad X^T = \begin{pmatrix} 1 \\ 2 \\ 8 \\ 16 \\ 4 \\ 64 \end{pmatrix} \qquad y = -14$$

$$\beta^2 = \begin{pmatrix} \frac{2}{5} \\ \frac{8}{5} \\ \frac{2}{5} \\ \frac{8}{5} \\ \frac{32}{5} \\ \frac{2}{5} \end{pmatrix} + \frac{1}{5} \cdot \begin{pmatrix} 1 \\ 2 \\ 8 \\ 16 \\ 4 \\ 64 \end{pmatrix} \cdot \left(-14 - \begin{pmatrix} 1 & 2 & 8 & 16 & 4 & 64 \end{pmatrix} \cdot \begin{pmatrix} \frac{2}{5} \\ \frac{8}{5} \\ \frac{2}{5} \\ \frac{8}{5} \\ \frac{32}{5} \\ \frac{2}{5} \end{pmatrix} \right) = \begin{pmatrix} -\frac{478}{25} \\ -\frac{936}{25} \\ -\frac{3894}{25} \\ -\frac{7768}{25} \\ -\frac{1792}{25} \\ -\frac{31222}{25} \end{pmatrix}$$

c) Use the initial guess $\beta^{(0)} = \mathbf{0}$ and the learning rate $\eta = 0.1$ and perform the first two steps of mini-batch gradient descent with a batch size of $N_b = 2$. Use the same approach for the randomization part as in the previous sub-task.

For the first step we have:

$$X = \begin{pmatrix} 1 & 4 & 1 & 4 & 16 & 1 \\ 1 & 2 & 8 & 16 & 4 & 64 \end{pmatrix} \qquad X^T = \begin{pmatrix} 1 & 1 \\ 4 & 2 \\ 1 & 8 \\ 4 & 16 \\ 16 & 4 \\ 1 & 64 \end{pmatrix} \qquad y = \begin{pmatrix} 2 \\ -14 \end{pmatrix}$$

Now we calculate the coefficient vector for the first step:

$$\beta^1 = \beta^0 + 2 \cdot \eta \cdot X^T \cdot (y - X \cdot \beta^0)$$

$$\beta 1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{5} \cdot \begin{pmatrix} 1 & 1 \\ 4 & 2 \\ 1 & 8 \\ 4 & 16 \\ 16 & 4 \\ 1 & 64 \end{pmatrix} \cdot \left(\begin{pmatrix} 2 \\ -14 \end{pmatrix} - \begin{pmatrix} 1 & 4 & 1 & 4 & 16 & 1 \\ 1 & 2 & 8 & 16 & 4 & 64 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right)$$

$$\beta^1 = \begin{pmatrix} -\frac{12}{5} \\ -4 \\ -22 \\ -\frac{216}{5} \\ -\frac{24}{5} \\ -\frac{894}{5} \end{pmatrix}$$

For the second step we have:

$$X = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 3 & 2 & 6 & 9 & 4 \end{pmatrix} \qquad X^T = \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 0 & 2 \\ 0 & 6 \\ 1 & 9 \\ 0 & 4 \end{pmatrix} \qquad y = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Now we calculate the coefficient vector for the second step:

$$\beta^2 = \beta^1 + 2 \cdot \eta \cdot X^T \cdot (y - X \cdot \beta^1)$$

$$\beta 2 = \begin{pmatrix} -\frac{12}{5} \\ -4 \\ -22 \\ -\frac{216}{5} \\ -\frac{24}{5} \\ -\frac{894}{5} \end{pmatrix} + \frac{1}{5} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 0 & 2 \\ 0 & 6 \\ 1 & 9 \\ 0 & 4 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 \\ -11 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 3 & 2 & 6 & 9 & 4 \end{pmatrix} \cdot \begin{pmatrix} -\frac{12}{5} \\ -4 \\ -22 \\ -\frac{216}{5} \\ -\frac{24}{5} \\ -\frac{894}{5} \end{pmatrix} \right)$$

$$\beta 2 = \begin{pmatrix} -\frac{12}{5} \\ -4 \\ -22 \\ -\frac{216}{5} \\ -\frac{24}{5} \\ -\frac{894}{5} \end{pmatrix} + \frac{1}{5} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 0 & 2 \\ 0 & 6 \\ 1 & 9 \\ 0 & 4 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 \\ -11 \end{pmatrix} + \begin{pmatrix} \frac{56}{5} \\ \frac{5380}{5} \end{pmatrix} \right)$$

$$\beta^2 = \begin{pmatrix} -\frac{12}{5} \\ -4 \\ -22 \\ \frac{216}{5} \\ -\frac{24}{5} \\ -\frac{894}{5} \end{pmatrix} + \begin{pmatrix} \frac{5436}{25} \\ \frac{16186}{25} \\ 430 \\ 1290 \\ \frac{48436}{25} \\ 860 \end{pmatrix} = \begin{pmatrix} \frac{5376}{25} \\ \frac{16086}{25} \\ 408 \\ \frac{6234}{5} \\ \frac{48316}{25} \\ \frac{3406}{5} \end{pmatrix}$$

Programming Problem 6.1

In this programming exercise, you will implement mini-batch gradient descent in order to train a linear model using least squares. Note that you can use an implementation of mini-batch gradient descent to get both, batch gradient descent and stochastic gradient descent.
As training data, you will use again the Energy efficiency Data Set from the UCI Machine Learning Repository, where we consider the required heating energy as output quantity.
To be more specific, you are supposed to reproduce Example 5.2 from the lecture notes, however with your own implementation of mini-batch gradient descent.

The implementation of gradient descent can be found in the file `programming_exercises.ipynb`. There are three classes, `MiniBatchGradientDescentLinearRegressor`, `SGDLinearRegressor`, and `BGDLinearRegressor`, the latter two being simply proxies for the first one. The first class implements the algorithm for gradient descent in the `fit` method. The class expects the training data to be NumPy arrays. The latter classes expect only two arguments, `learning_rate` and `epochs`, while the parent class also expects the `batch_size` argument. In case `batch_size` argument is not provided, it is set to match the whole training set size given. Example 5.2 is also provided at the end of the notebook for comparison.