2. Linear Regression with Closed Form Solution

Extension Note: Project 2 due date has been extended by 2 days to **July 18 23:59UTC** (Note the UTC time zone).

After seeing the problem, your classmate Alice immediately argues that we can apply a linear regression model, as the labels are numbers from 0-9, very similar to the example we learned from the lecture. Though being a little doubtful, you decide to have a try and start simple by using the raw pixel values of each image as features.

Alice wrote a skeleton code run_linear_regression_on_MNIST in main.py, but she needs your help to complete the code and make the model work.

Closed Form Solution of Linear Regression

5.0/5.0 points (graded)

To solve the linear regression problem, you recall the linear regression has a closed form solution:

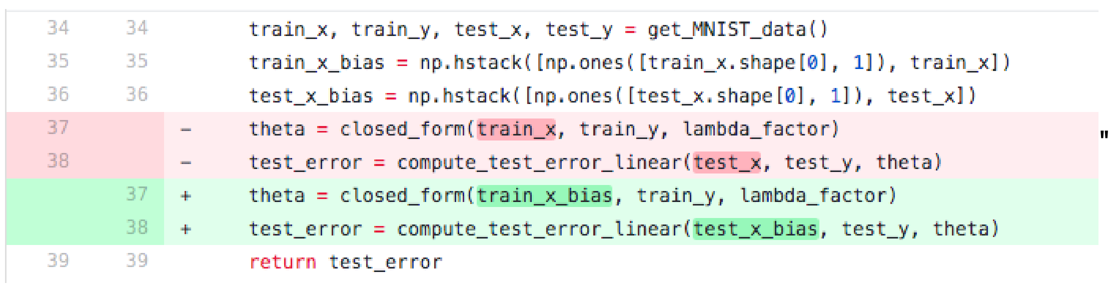
$$\theta = (X^T X + \lambda I)^{-1} X^T Y$$

where I is the identity matrix.

Write a function `closed_form` that computes this closed form solution given the features X, labels Y and the regularization parameter λ.

Available Functions: You have access to the NumPy python library as `np`; No need to import anything.

Correction Note: The version of project archive before July 12 contains an error in the function `run_linear_regression_on_MNIST`. Please either download the correct version mnist.tar.gz now, or make the following correction to lines 37-38 in `main.py`:



```
34   34       train_x, train_y, test_x, test_y = get_MNIST_data()
35   35       train_x_bias = np.hstack([np.ones([train_x.shape[0], 1]), train_x])
36   36       test_x_bias = np.hstack([np.ones([test_x.shape[0], 1]), test_x])
37        -   theta = closed_form(train_x, train_y, lambda_factor)
38        -   test_error = compute_test_error_linear(test_x, test_y, theta)
     37   +   theta = closed_form(train_x_bias, train_y, lambda_factor)
     38   +   test_error = compute_test_error_linear(test_x_bias, test_y, theta)
39   39       return test_error
```

```
    Computes the closed form solution of linear regression with L2 regularization

    Args:
        X - (n, d + 1) NumPy array (n datapoints each with d features plus the bias feature in the first dimension)
        Y - (n, ) NumPy array containing the labels (a number from 0-9) for each
            data point
        lambda_factor - the regularization constant (scalar)
    Returns:
        theta - (d + 1, ) NumPy array containing the weights of linear regression. Note that theta[0]
        represents the y-axis intercept of the model and therefore X[0] = 1
    """
    d = X.shape[1]
    theta = np.linalg.pinv(X.T @ X + lambda_factor * np.identity(d)) @ (X.T @ Y)
    return theta

```

Press ESC then TAB or click outside of the code editor to exit

Correct

```
def closed_form(X, Y, lambda_factor):
    """
    Computes the closed form solution of linear regression with L2 regularization

    Args:
        X - (n, d + 1) NumPy array (n datapoints each with d features plus the bias feature in the first dimension)
        Y - (n, ) NumPy array containing the labels (a number from 0-9) for each
            data point
        lambda_factor - the regularization constant (scalar)
    Returns:
        theta - (d + 1, ) NumPy array containing the weights of linear regression. Note that theta[0]
        represents the y-axis intercept of the model and therefore X[0] = 1
    """
    I = np.identity(X.shape[1])
    theta = np.linalg.inv(X.T @ X + lambda_factor * I) @ X.T @ Y
    return theta
```

Test results

```
CORRECT
```

See full output

See full output

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Test Error on Linear Regression

1.0/1.0 point (graded)
Apply the linear regression model on the test set. For classification purpose, you decide to round the predicted label into numbers 0-9.

Note: For this project we will be looking at the error rate defined as the fraction of labels that don't match the target labels, also known as the "gold labels" or ground truth. (In other context, you might want to consider other performance measures such as precision and recall, which we have not discussed in this course).

Please enter the **test error** of your linear regression algorithm for different λ (copy the output from the `main.py` run).

Grading note: The tolerance of this problem has been lowered to accept answers from both the correct and incorrect versions of `run_linear_regression_on_MNIST`.

$\mathrm{Error}|_{\lambda=1} =$ 0.744 ✓ **Answer:** 0.7697

$\mathrm{Error}|_{\lambda=0.1} =$ 0.7442 ✓ **Answer:** 0.7698

$\mathrm{Error}|_{\lambda=0.01} =$ 0.744 ✓ **Answer:** 0.7702

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

What went Wrong?

1.0/1.0 point (graded)
Alice and you find that no matter what λ factor you try, the test error is large. With some thinking, you realize that something is wrong with this approach.

☐ Gradient descent should be used instead of the closed form solution.

- [✓] The loss function related to the closed-form solution is inadequate for this problem. ✓
- [ ] Regularization should not be used here.

✓

Solution:

The closed form solution of linear regression is the solution of optimizing the mean squared error loss. This is not an appropriate loss function for a classification problem.

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Discussion

Show Discussion

Topic: Unit 2 Nonlinear Classification, Linear regression, Collaborative Filtering (2 weeks):Project 2: Digit recognition (Part 1) / 2. Linear Regression with Closed Form Solution