6. Changing Labels

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

We now wish to classify the digits by their (mod 3) value, such that the new label $y^{(i)}$ of sample i is the old $y^{(i)} \pmod 3$. (Reminder: Return the `temp_parameter` to be 1 if you changed it for the last section)

You will be working in the file `part1/main.py` and `part1/softmax.py` in this problem

Using the Current Model - update target

3.0/3.0 points (graded)

Given that we already classified every $x^{(i)}$ as a digit, we could use the model we already trained and just calculate our estimations (mod 3).

Implement `update_y` function, which changes the old digit labels for the training and test set for the new (mod 3) labels.

Available Functions: You have access to the NumPy python library as `np`

```
    labels.

    Args:
        train_y - (n, ) NumPy array containing the labels (a number between 0-9)
                  for each datapoint in the training set
        test_y - (n, ) NumPy array containing the labels (a number between 0-9)
                 for each datapoint in the test set

    Returns:
        train_y_mod3 - (n, ) NumPy array containing the new labels (a number between 0-2)
                       for each datapoint in the training set
        test_y_mod3 - (n, ) NumPy array containing the new labels (a number between 0-2)
                      for each datapoint in the test set
    """
    return (np.mod(train_y,3), np.mod(test_y,3))

```

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

Correct

```
def update_y(train_y, test_y):
    """
    Changes the old digit labels for the training and test set for the new (mod 3)
    labels.

    Args:
        train_y - (n, ) NumPy array containing the labels (a number between 0-9)
                  for each datapoint in the training set
        test_y - (n, ) NumPy array containing the labels (a number between 0-9)
                 for each datapoint in the test set

    Returns:
        train_y_mod3 - (n, ) NumPy array containing the new labels (a number between 0-2)
                       for each datapoint in the training set
        test_y_mod3 - (n, ) NumPy array containing the new labels (a number between 0-2)
                      for each datapoint in the test set
    """
    return np.remainder(train_y, 3), np.remainder(test_y, 3)
```

Test results

See full output

CORRECT

See full output

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Using the Current Model - compute test error

3.0/3.0 points (graded)

Implement `compute_test_error_mod3` function, which takes the test points `X`, their correct labels `Y` (digits (mod 3) from 0-2), `theta`, and the `temp_parameter`, and returns the error.

Example:

	Estimated Y	Estimated Y (mod 3)	Correct Y	Correct Y (mod 3)
x_1	9	0	8	2
x_2	6	0	6	0
x_3	5	2	8	2

The error of the regression with the original labels would be 0.66667

However, the error of the regression when comparing the (mod 3) of the labels would be 0.33333

Available Functions: You have access to the NumPy python library as `np` and to the `get_classification` function from the project release

```
    Returns the error of these new labels when the classifier predicts the digit. (mod 3)
4
5   Args:
6       X - (n, d - 1) NumPy array (n datapoints each with d - 1 features)
7       Y - (n, ) NumPy array containing the labels (a number from 0-2) for each
8           data point
9       theta - (k, d) NumPy array, where row j represents the parameters of our
10              model for label j
11      temp_parameter - the temperature parameter of softmax function (scalar)
12
13  Returns:
14      test_error - the error rate of the classifier (scalar)
15  """
16  pred_Y = get_classification(X, theta, temp_parameter)
17  return 1 - np.mean(np.mod(pred_Y,3) == Y)
18
```

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

Correct

```
def compute_test_error_mod3(X, Y, theta, temp_parameter):
    """
    Returns the error of these new labels when the classifier predicts the digit. (mod 3)

    Args:
        X - (n, d - 1) NumPy array (n datapoints each with d - 1 features)
        Y - (n, ) NumPy array containing the labels (a number from 0-2) for each
            data point
        theta - (k, d) NumPy array, where row j represents the parameters of our
                model for label j
        temp_parameter - the temperature parameter of softmax function (scalar)

    Returns:
        test_error - the error rate of the classifier (scalar)
    """
    assigned_labels = get_classification(X, theta, temp_parameter)
    return 1 - np.mean(np.remainder(assigned_labels,3) == Y)
```

Test results

CORRECT

See full output

See full output

Submit You have used 9 of 20 attempts

Answers are displayed within the problem

Using the Current Model - test error

1.0/1.0 point (graded)
Find the error rate of the new labels (call these two functions at the end of `run_softmax_on_MNIST`). See the functions' documentation for detailed explanations of the inputs and outputs.

Error rate for labels mod 3: 0.0768 ✔ **Answer:** 0.0768

Submit You have used 2 of 20 attempts

Answers are displayed within the problem

Retrain with New Labels

3.0/3.0 points (graded)
Now suppose that instead we want to retrain our classifier with the new labels. In other words, rather than training the model to predict the original digits and then taking those predictions modulo 3, we explicitly train the model to predict the digits modulo 3 from the original image.

How do you expect the performance to change using the new labels?

- Increase
- Decrease ✔
- Stay the same

Implement `run_softmax_on_MNIST_mod3` in **main.py** to perform this new training; report the new error rate.

Error rate when trained on labels mod 3: 0.187200 ✔ **Answer:** 0.1881

Solution:

We are trying to find common features of all numbers that have the same mod 3 value, however a lot of them look widely different, so it is harder to separate the data set into 3 groups since, for example, 2 does not share many features with 5 or 8. Therefore one would expect the performance to decrease, and this is what happens.

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Discussion

Show Discussion