3. Support Vector Machine

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

Bob thinks it is clearly not a regression problem, but a classification problem. He thinks that we can change it into a binary classification and use the support vector machine we learned in Lecture 4 to solve the problem. In order to do so, he suggests that we can build an one vs. rest model for every digit. For example, classifying the digits into two classes: 0 and not 0.

Bob wrote a function `run_svm_one_vs_rest_on_MNIST` where he changed the labels of digits 1-9 to 1 and keeps the label 0 for digit 0. He also found that `sklearn` package contains an SVM model that you can use directly. He gave you the link to this model and hopes you can tell him how to use that.

You will be working in the file `part1/svm.py` in this problem

Important: For this problem, you will need to use the scikit-learn library. If you don't have it, install it using `pip install sklearn`

One vs. Rest SVM

5.0/5.0 points (graded)
Use the `sklearn` package and build the SVM model on your local machine. Use `random_state = 0`, `C=0.1` and default values for other parameters.

Available Functions: You have access to the sklearn's implementation of the linear SVM as `LinearSVC`; No need to import anything.

```
def one_vs_rest_svm(train_x, train_y, test_x):
    """
    Trains a linear SVM for binary classficiation

    Args:
        train_x - (n, d) NumPy array (n datapoints each with d features)
        train_y - (n, ) NumPy array containing the labels (0 or 1) for each training data point
        test_x - (m, d) NumPy array (m datapoints each with d features)
    Returns:
        pred_test_y - (m,) NumPy array containing the labels (0 or 1) for each test data point
    """
    clf = LinearSVC(random_state = 0, C=0.1)
    clf.fit(train_x, train_y)
    return clf.predict(test_x)

```

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

Correct

```
def one_vs_rest_svm(train_x, train_y, test_x):
    """
    Trains a linear SVM for binary classficiation

    Args:
        train_x - (n, d) NumPy array (n datapoints each with d features)
        train_y - (n, ) NumPy array containing the labels (0 or 1) for each training data point
        test_x - (m, d) NumPy array (m datapoints each with d features)
    Returns:
        pred_test_y - (m,) NumPy array containing the labels (0 or 1) for each test data point
    """
    clf = LinearSVC(C=0.1, random_state=0)
    clf.fit(train_x, train_y)
    pred_test_y = clf.predict(test_x)
    return pred_test_y
```

Test results

CORRECT

See full output

See full output

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Binary classification error

5.0/5.0 points (graded)
Report the test error by running `run_svm_one_vs_rest_on_MNIST`.

$\text{Error} =$ 0.0075 ✔ **Answer:** 0.007499999999999951

Submit You have used 2 of 20 attempts

Answers are displayed within the problem

Implement C-SVM

5.0/5.0 points (graded)
Play with the C parameter of SVM, what statement is true about the C parameter?

(Choose all that apply.)

- [ ] Larger C gives larger tolerance of violation.
- [x] Larger C gives smaller tolerance of violation. ✔
- [ ] Larger C gives a larger-margin separating hyperplane.
- [x] Larger C gives a smaller-margin separating hyperplane. ✔

Solution:

C represents the tolerance of error. A larger C means we are punishing more on the classification error, thus being less tolerant to misclassifications. Therefore, we will get a smaller margin hyperplane.

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Multiclass SVM

5.0/5.0 points (graded)
In fact, `sklearn` already implements a multiclass SVM with a one-vs-rest strategy. Use `LinearSVC` to build a multiclass SVM model

Available Functions: You have access to the sklearn's implementation of the linear SVM as `LinearSVC`; No need to import anything.

```
def multi_class_svm(train_x, train_y, test_x):
    """
    Trains a linear SVM for multiclass classifciation using a one-vs-rest strategy
```

```
    """
    Args:
        train_x - (n, d) NumPy array (n datapoints each with d features)
        train_y - (n, ) NumPy array containing the labels (int) for each training data point
        test_x - (m, d) NumPy array (m datapoints each with d features)
    Returns:
        pred_test_y - (m,) NumPy array containing the labels (int) for each test data point
    """
    clf = LinearSVC(random_state = 0, C = 0.1, multi_class = 'ovr')
    clf.fit(train_x, train_y)
    return clf.predict(test_x)

```

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

Correct

```
def multi_class_svm(train_x, train_y, test_x):
    """
    Trains a linear SVM for multiclass classifciation using a one-vs-rest strategy

    Args:
        train_x - (n, d) NumPy array (n datapoints each with d features)
        train_y - (n, ) NumPy array containing the labels (int) for each training data point
        test_x - (m, d) NumPy array (m datapoints each with d features)
    Returns:
        pred_test_y - (m,) NumPy array containing the labels (int) for each test data point
    """
    clf = LinearSVC(C=0.1, random_state=0)
    clf.fit(train_x, train_y)
    pred_test_y = clf.predict(test_x)
    return pred_test_y
```

Test results

See full output

CORRECT

See full output

Solution:

As you see, we are using the same code for both SVM functions. Indeed, the default argument for `multi_class` in `LinearSVC` is `ovr`.

Submit You have used 4 of 20 attempts

Answers are displayed within the problem

Multiclass SVM error

5.0/5.0 points (graded)
Report the overall test error by running `run_multiclass_svm_on_MNIST`.

$\text{Error} =$ 0.0819 ✔ **Answer:** 0.08189999999999997

Submit You have used 4 of 20 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) / 3. Support Vector Machine