3. Support Vector Machine

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 digits. 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 digit 1-9 to 1 and keeps the label 0 for digit 0. He also found that `sklearn` package contains a SVM model where you can use directly. He gave you the link to that model and hope 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

5/5 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 classifciation

    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

Test results

Hide output

CORRECT

Test: output1

Output:

```
train_x: [[0.74021551 0.43195563]
 [0.40684059 0.44127392]
 [0.99915919 0.52231402]
 [0.29457754 0.45500002]
 [0.25464916 0.50020259]
 [0.54634547 0.96572388]
 [0.27012457 0.04665474]
 [0.76504382 0.82860708]
 [0.68958708 0.77897273]
 [0.38031509 0.81979835]
 [0.84065229 0.79751435]
 [0.95114222 0.29275591]
 [0.06657971 0.70887626]
 [0.37758739 0.57958282]
 [0.59278639 0.24966864]
 [0.46846513 0.82311161]]
train_y: [0 0 0 0 0 0 1 1 1 0 1 1 1 0 0]
test_x: [[0.08559828 0.35649492]
 [0.58029473 0.48915356]
 [0.95243623 0.31967552]
 [0.3185904  0.50873137]
 [0.48851915 0.9699501 ]
 [0.2590593  0.80253755]
 [0.73428524 0.88636573]
 [0.54701302 0.1022148 ]
 [0.10810793 0.8151251 ]
 [0.97802428 0.58690018]
 [0.46722661 0.55904237]]
Submission output: [0 0 0 0 0 0 0 0 0 0 0]
```

Test: output2

Output:

```
train_x: [[0.65447353 0.20028434 0.48239213]
 [0.64171897 0.56762668 0.88175238]
 [0.8447331  0.4998508  0.90182049]
 [0.63844322 0.55868315 0.5755569 ]
 [0.24214408 0.51659304 0.76089543]
 [0.54918455 0.26628602 0.22222436]
 [0.2604621  0.47727055 0.97995404]
 [0.63602578 0.42497548 0.32957748]
 [0.72721527 0.3376802  0.32333993]
 [0.5883944  0.72746542 0.62569041]
 [0.72004228 0.52763629 0.71935559]]
train_y: [0 0 1 0 1 0 1 0 0 1 1]
test_x: [[0.77964731 0.81920037 0.25430089]
 [0.66557967 0.78041885 0.28126548]
 [0.0423602  0.02576799 0.57016067]
 [0.13564713 0.7721141  0.34966991]
 [0.15671801 0.3125784  0.30351681]
 [0.33528671 0.38876823 0.19711744]
 [0.28158124 0.20749997 0.24147398]
 [0.2879429  0.07601778 0.26567582]
 [0.28807974 0.57639994 0.42400284]
 [0.53160431 0.03197762 0.89518054]
 [0.23845549 0.73092687 0.44063511]
 [0.74996488 0.64791171 0.08330063]
 [0.30355642 0.72450119 0.75979937]
 [0.55121859 0.60209421 0.06778405]
 [0.65039024 0.35902444 0.78136871]]
Submission output: [0 0 1 1 0 0 0 0 0 1 1 0 1 0 0]
```

Test: output3

Output:

```
train_x: [[8.11279050e-01 4.52141831e-01]
 [2.10601269e-01 4.60914351e-01]
 [7.56697823e-01 1.80289453e-01]
 [7.06835399e-01 3.74637157e-01]
 [8.38588837e-01 6.44637236e-01]
 [4.74836638e-01 7.96392933e-04]
 [8.62446759e-01 8.94038847e-01]
 [3.79948573e-01 3.09683287e-02]
 [6.67359468e-03 9.36768967e-01]
 [8.78075213e-01 1.76902314e-01]
 [4.77391055e-01 4.33192373e-01]
 [4.43228794e-01 1.01555361e-02]
 [5.76255389e-01 7.91439018e-01]
 [9.68005407e-01 5.91789850e-01]
 [5.14374622e-01 4.40404097e-01]
 [6.03612168e-01 6.45366923e-02]
 [2.19506810e-01 9.68929440e-01]
 [8.73675061e-01 7.23780916e-02]
 [2.21873555e-01 8.56756712e-02]]
train_y: [0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 1 0]
test_x: [[0.81382256 0.15376929]
 [0.23824971 0.33176453]
 [0.02746353 0.83300153]
 [0.17765254 0.40144322]
 [0.32422425 0.18189357]
 [0.87095893 0.57074161]
 [0.67063566 0.14012018]
 [0.11496733 0.14235777]
 [0.96174966 0.90831115]
 [0.83568426 0.16983058]
 [0.42290956 0.21958185]
 [0.12218531 0.46372719]
 [0.60522441 0.29051125]
 [0.07373684 0.62676753]
 [0.20730928 0.7869615 ]
 [0.35580997 0.46428361]
 [0.78131668 0.69473551]
 [0.87077188 0.91518852]]
Submission output: [0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1]
```

Test: outputr4

Output:

```
train_x: [[0.40818312 0.38159886 0.24241605 0.38366628]
 [0.27033749 0.00799247 0.68875155 0.90568713]
 [0.7260674  0.45649055 0.18981836 0.21927554]
 [0.86720485 0.37834952 0.12136992 0.04190428]
 [0.76908809 0.13222895 0.75322116 0.57165623]
 [0.2783364  0.08799754 0.44468719 0.14840139]
 [0.6340726  0.84103611 0.87859193 0.58786321]
 [0.75026416 0.28905957 0.85672734 0.26242404]
 [0.51058753 0.874677   0.64296309 0.74521429]
 [0.21944755 0.75669826 0.65450055 0.07229256]
 [0.00590142 0.57580201 0.04343592 0.64508964]
 [0.55628962 0.60103265 0.32649291 0.44102935]]
train_y: [1 1 1 0 1 0 0 1 1 1 0 0]
test_x: [[0.37860324 0.13043979 0.81896389 0.26955548]
 [0.71999832 0.0391397  0.0892016  0.00906736]
 [0.80751849 0.84190127 0.63861565 0.20653685]
 [0.44203692 0.51044744 0.11300264 0.96149892]
 [0.55899333 0.30114035 0.65591599 0.02560668]
 [0.43083021 0.6259706  0.56773372 0.2968281 ]
 [0.16241506 0.59334928 0.35972919 0.55718829]
 [0.93006913 0.88344789 0.23734165 0.15495788]
 [0.67133825 0.23542507 0.25029202 0.22982249]
 [0.6335627  0.66607768 0.62857216 0.64588296]
 [0.67391828 0.13442488 0.51791969 0.62855356]
 [0.83447665 0.33478579 0.7688467  0.82935934]
 [0.2353971  0.62110565 0.55023205 0.09117792]
 [0.61131201 0.80061459 0.54041534 0.86326114]
 [0.0276005  0.62259065 0.59953523 0.31203471]]
Submission output: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
Test: size
```

Output:

```
Output size: (19,)
```

Hide output

Submit

You have used 2 of 20 attempts

✓ Correct (5/5 points)

Binary classification error

5/5 points (graded)

Report the test error by running `run_svm_one_vs_rest_on_MNIST`.

$\mathrm{Error} =$ 0.007499999999999951 ✓

Submit You have used 1 of 20 attempts

✓ Correct (5/5 points)

Implement C-SVM

5/5 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.

☑ Larger C gives smaller tolerance of violation.

☐ Larger C gives a larger-margin separating hyperplane.

☑ Larger C gives a smaller-margin separating hyperplane.

✓

Submit You have used 1 of 2 attempts

✓ Correct (5/5 points)

Multiclass SVM

5/5 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)
    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

Test results

Hide output

CORRECT

```
Test: output1
```

Output:

```
train_x: [[6.99578995e-01 5.53887478e-01]
 [6.85779158e-01 6.31515219e-01]
 [7.34893186e-01 3.85731936e-01]
 [8.10086613e-01 6.77818801e-01]
 [4.61409485e-01 8.28452598e-01]
 [4.42838547e-01 9.63062400e-01]
 [2.18127805e-01 2.58773532e-01]
 [5.99256262e-04 9.77156129e-02]
 [9.08444152e-01 2.19172952e-01]
 [6.81866122e-01 4.31294184e-01]
 [5.82768423e-01 2.83390240e-01]
 [7.43304318e-01 2.04510477e-01]
 [8.39572330e-01 1.92496580e-01]]
train_y: [0 1 2 3 3 0 2 3 3 0 2 3 3]
test_x: [[0.54078115 0.90424663]
 [0.30404909 0.72121012]
 [0.9573102  0.54364279]
 [0.78040913 0.4710401 ]
 [0.4121865  0.85725592]
 [0.38522828 0.47252131]
 [0.8350261  0.98659789]
 [0.12303874 0.56828995]
 [0.8581817  0.11108622]
 [0.53666229 0.7750237 ]
 [0.498776   0.30302112]
 [0.0691167  0.80248447]
 [0.81882352 0.71891433]
 [0.11216922 0.30237022]
 [0.95167904 0.58994716]]
Submission output: [3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

```
Test: output2
```

Output:

```
train_x: [[0.89067904 0.10903467 0.85626622]
 [0.05025322 0.63078981 0.24687799]
 [0.19626511 0.37969011 0.89536792]
 [0.65222839 0.0262676  0.06054459]
 [0.561349   0.22484606 0.09531271]
 [0.27482608 0.63519925 0.08724291]
 [0.61660784 0.25366854 0.14584416]
 [0.02046448 0.12280837 0.66150661]
 [0.04786739 0.30779211 0.88858349]
 [0.34419749 0.80909343 0.13903339]]
train_y: [3 0 0 0 2 1 0 0 1 2]
test_x: [[0.96478668 0.65586213 0.57584791]
 [0.73845802 0.45179801 0.98143644]
 [0.65385843 0.34829907 0.94578283]
 [0.19782722 0.04934819 0.9415929 ]
 [0.24559923 0.66968244 0.99222757]
 [0.05170463 0.39427076 0.06792789]
 [0.55638174 0.74970882 0.51336906]
 [0.00490819 0.84121466 0.34129819]
 [0.06985683 0.0293824  0.17444237]
 [0.92886513 0.90440363 0.16560619]
 [0.17015272 0.05538263 0.36379799]
 [0.49996439 0.95682557 0.5311442 ]
 [0.00390922 0.71012281 0.83096032]
 [0.32613772 0.18519645 0.73048977]
 [0.00377797 0.67512294 0.26130904]
 [0.93387774 0.02629033 0.08979667]
 [0.87688284 0.71547598 0.82567368]
 [0.46420684 0.99850153 0.22575559]
 [0.96575274 0.31854653 0.62150942]]
Submission output: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
Test: output3
```

Output:

```
train_x: [[0.75002148 0.63795031 0.02730299]
 [0.06007064 0.92071922 0.95779576]
 [0.32853301 0.87291641 0.86004731]
 [0.24639947 0.2632307  0.47235945]
 [0.05084266 0.76128843 0.94864315]
 [0.47737458 0.88383301 0.50558399]
 [0.04090603 0.78157897 0.09865894]
 [0.75605868 0.75261111 0.4173839 ]
 [0.20532656 0.09058549 0.4575146 ]
 [0.99151513 0.8016871  0.31725206]
 [0.63073877 0.19078004 0.42412705]]
train_y: [1 2 0 3 2 3 4 3 3 3 5]
test_x: [[0.6167434  0.37653891 0.32302156]
 [0.52221961 0.09780199 0.8823878 ]
 [0.61515609 0.18121164 0.14613885]
 [0.59838007 0.50415017 0.54354423]
 [0.07601381 0.39909316 0.41933142]
 [0.69729208 0.18875622 0.73619069]
 [0.09658408 0.68687144 0.54239389]
 [0.12313501 0.80736059 0.72992841]
 [0.04087764 0.13969736 0.85577583]
 [0.36916368 0.25326968 0.46333089]]
Submission output: [3 3 3 3 3 3 3 3 3 3]
```

Test: outputr4

Output:

```
train_x: [[0.15113639 0.83737355]
 [0.46977631 0.61996902]
 [0.49968899 0.11361127]
 [0.8626747  0.8358077 ]
 [0.66348748 0.36102256]
 [0.85746564 0.95724941]
 [0.27952378 0.95736125]
 [0.22205355 0.63287309]
 [0.67004944 0.61885252]
 [0.79073745 0.07174858]
 [0.27565991 0.08344264]]
train_y: [0 1 0 2 1 0 3 3 2 0 2]
test_x: [[0.44307091 0.37645626]
 [0.25656735 0.98210213]
 [0.54969574 0.77414383]
 [0.95812598 0.70050533]
 [0.18251712 0.1872745 ]
 [0.95824287 0.98384452]
 [0.78242729 0.06993624]
 [0.96004066 0.62649782]
 [0.48468815 0.79527531]
 [0.14305605 0.55477047]
 [0.93962422 0.16525797]
 [0.14296584 0.46005    ]
 [0.30824337 0.49591007]]
Submission output: [0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Test: size

Output:

```
Output size: (12,)
```

Hide output

Submit

You have used 1 of 20 attempts

✓ Correct (5/5 points)

Multiclass SVM error

5/5 points (graded)

Report the overall test error by running `run_multiclass_svm_on_MNIST`.

Error = 0.08189999999999997 ✓

Submit You have used 1 of 20 attempts

✓ Correct (5/5 points)

Discussion

Hide Discussion

Topic: Unit 2 Nonlinear Classification, Linear regression, Collaborative Filtering (2 weeks):Project 2: Digit recognition (Part 1) / 3. Support Vector Machine

Add a Post

Show all posts ▼ by recent activity ▼

How to view the misclassified images along with their labels 3
To get an idea of what the digit images that were classified incorrectly actually look like, I added a bit of function to display just the misclassified images and put a label above ...

☑ [STAFF] Problem with "Binary classification error"? 8

Implement C-SVM 6

[Staff] Multiclass SVM
Grader erroneously asks to implement `one_vs_rest_svm` twice instead of `multi_class_svm`. 11
Community TA

Error in Answer: Implement C-SVM 7

☑ Multiclass SVM error 6
Multiclass SVM works fine and passes the grader, but the error -- which is lower than for one vs. rest SVM -- is marked as wrong.

Learn About Verified Certificates