```
Dec 9, Lui
 1 import numpy as np
 2 import util
   def main(train_path, valid path, save path):
        """Problem: Gaussian discriminant analysis (GDA)
 7
 8
       Args:
                                                     Dec 9, 2021, 3:45:16 PM PST
 9
           train path: Path to CSV file containing dataset for training.
10
           valid path: Path to CSV file containing dataset for validation.
11
           save_path: Path to save predicted probabilities using np.savetxt().
12
13
       # Load dataset
14
       x_train, y_train = util.load_dataset(train_path, add_intercept=False)
15
       # *** START CODE HERE ***
16
17
       # Train a GDA classifier
       clf = GDA()
18
19
       clf.fit(x_train, y_train)
20
       # Plot decision boundary on validation set
21
22
       x_eval, y_eval = util.load_dataset(valid_path, add_intercept=False)
       plot_path = save_path.replace('.txt', '.png')
23
24
       util.plot(x_eval, y_eval, clf.theta, plot_path)
25
       x_eval = util.add_intercept(x_eval)
26
                                                    7ford.edu - Dec 9, 2021, 3:45:16 PN
27
       # Use np.savetxt to save outputs from validation set to save path
28
       p_eval = clf.predict(x_eval)
       yhat = p_eval > 0.5
29
       print('GDA Accuracy: %.2f' % np.mean( (yhat == 1) == (y_eval == 1)))
30
31
       np.savetxt(save_path, p_eval)
32
       # *** END CODE HERE ***
33
34
35
   class GDA:
36
        """Gaussian Discriminant Analysis.
37
38
       Example usage:
           > clf = GDA()
39
40
           > clf.fit(x_train, y_train)
           > clf.predict(x_eval)
41
        H/H/H
42
       def __init__(self, step_size=0.01, max_iter=10000, eps=1e-5,
43
                    theta_0=None, verbose=True):
44
            11 11 11
45
46
           Args:
47
               step_size: Step size for iterative solvers only.
                                                                   inford.edu-Dec 9, 2021, 3
48
               max iter: Maximum number of iterations for the solver.
49
               eps: Threshold for determining convergence.
50
               theta_0: Initial guess for theta. If None, use the zero vector.
51
               verbose: Print loss values during training.
52
53
54
           self.theta = theta_0
           self.step_size = step_size
55
           self.max_iter = max_iter
56
           self.eps = eps
57
           self.verbose = verbose
58
59
       def fit(self, x, y):
60
           """Fit a GDA model to training set given by x and y by updating
61
           self.theta.
62
63
           Args:
64
               x: Training example inputs. Shape (n_examples, dim).
65
               y: Training example labels. Shape (n_examples,).
           11 11 11
66
           # *** START CODE HERE ***
67
68
           m, n = x.shape
69
70
           # Find phi, mu 0, mu 1, and sigma
           phi = 1 / m * np.sum(y == 1)
71
72
           mu_0 = (y == 0).dot(x) / np.sum(y == 0)
```

adu-Dec

```
mu_1 = (y == 1).dot(x) / np.sum(y == 1)
 73
             mu_yi = np.where(np.expand_dims(y == 0, -1),
 74
 75
                               np.expand_dims(mu_0, 0),
 76
                               np.expand_dims(mu_1, 0))
 77
             sigma = 1 / m * (x - mu_yi).T.dot(x - mu_yi)
 78
             # Write theta in terms of the parameters
 79
               cuself.theta))

cuself.theta))

rgs:

x: Inputs of shape (n_examples, dim).

rs:

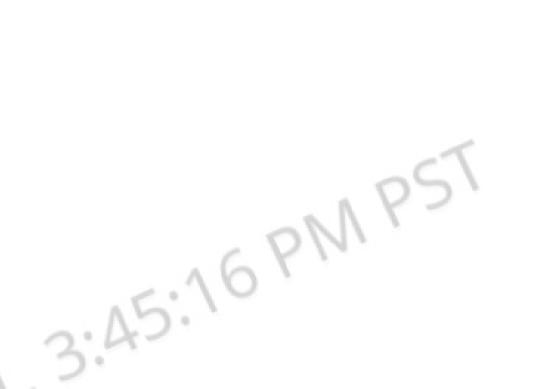
rputs of shape (n_examples, dim).

RT COOP "
 80
             self.theta = np.zeros(n + 1)
 81
             sigma_inv = np.linalg.inv(sigma)
 82
             mu_diff = mu_0.T.dot(sigma_inv).dot(mu_0) \
 83
 84
             self.theta[0] = 1 / 2 * mu_diff - np.log((1 - phi) / phi)
 85
             self.theta[1:] = -sigma_inv.dot(mu_0 - mu_1)
 86
 87
             if self.verbose:
 88
             # *** END CODE HERE ***
 89
 90
 91
         def predict(self, x):
 92
              """Make a prediction given new inputs x.
 93
 94
             Args:
 95
 96
 97
             Returns:
 98
           cid.csv',
oua_pred_1.txt')

\text{valid_path='ds2_train.csv',}
valid_path='ds2_valid.csv',
save_path='gda_pred_2.txt')
 99
100
101
102
103
104
105
         @staticmethod
106
         def _sigmoid(x):
107
108
109
110 if
        __name__ == '__main ':
111
         main(train_path='ds1_train.csv',
112
113
114
115
         main(train_path='ds2_train.csv',
116
117
```

vgPatel1@stanford.edu-Dec9,2021,3

edil-Dec



5:16 PM PST