Assignment 1

	ELEC 425
Date	@October 19, 2022
Status	Not started

ELEC 425 Assignment 1

Hailing Rao, 20121206, 18hlr@queensu.ca

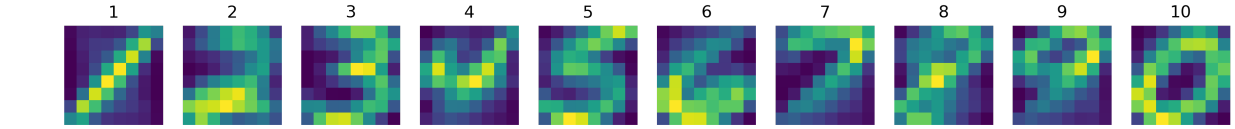
Language: Python

1. Exploring the Dataset

(No marks)

2. Gaussian Classifiers

Gaussian Classifier Trained Model



Value of σ^2:

0.06335051229114672

Code (shortened):

```
class gaussian_classifier:
    def __init__(self, data):
        self.mean_vectors = list()
        self.variance_squared = list()
        self.training_data = np.array(data['digits_train'])
        self.test_data = np.array(data['digits_test'])

    def calculate_mean_vector(self, class_index):
        digits_train = self.training_data
        mean_vector = [0] * 64 # 64 features
        num_training_points = np.size(digits_train, 1)
        for i in range(num_training_points): # loop through training data
            datapoint = digits_train[:,i,class_index]
            # for each training datapoint, add the value to the mean
            mean_vector = mean_vector + datapoint

        # divide meanvector by num of training data
        mean_vector = mean_vector / num_training_points
        return mean_vector

    def calculate_variance_squared(self):
        digits_train = self.training_data
        mean_vectors = self.mean_vectors
        num_classes, num_datapoints, num_features = np.size(digits_train, 2), np.size(digits_train, 1), np.size(digits_train, 0)
```

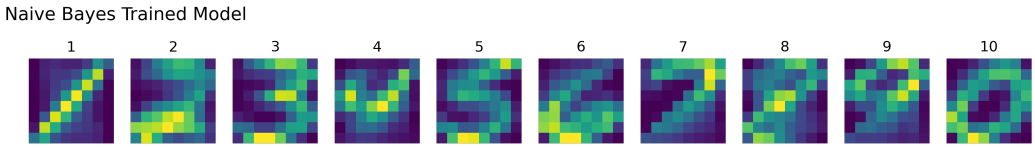
```
        variance = 0

        for class_num in range(num_classes):
            mean_vector = self.mean_vectors[class_num]
            for data_point in range(num_datapoints):
                for feature in range(num_features):
                    x, u = digits_train[feature, data_point, class_num], mean_vector[feature]
                    variance += (x-u)**2
        DM = num_features*num_datapoints * 10 # K = 10

        return variance / DM

    def train(self):
        digits_train = self.training_data
        num_classes = np.size(digits_train, 2)
        for i in range(num_classes):
            mean_vector = self.calculate_mean_vector(i)
            self.mean_vectors.append(mean_vector)
        self.variance_squared= self.calculate_variance_squared()
```

3. Naïve Bayes Classifiers



Code (shortened):

```
class bayes_classifier:
    def __init__(self, data):
        self.n_k_values = list()
        self.training_data = np.array(data['digits_train'])
        self.test_data = np.array(data['digits_test'])
        ## convert the data to binary
        def convert_data(dataset):
            K, m_Val, D = np.size(dataset, 2), np.size(dataset, 1), np.size(dataset, 0)
            for k in range(K):
                for m in range(m_Val):
                    for d in range(D):
                        val = dataset[d, m, k]
                        if val > 0.5:
                            val = 1
                        else:
                            val = 0
                        dataset[d, m, k] = val
            return dataset
        self.training_data = convert_data(self.training_data)
        self.test_data = convert_data(self.test_data)
    def calculate_n_k(self, class_index):
        digits_train = self.training_data
        p_ck = 1/10
        n_k = [0] * TRAIN_D
        for datapoint_id in range(TRAIN_m):
            n_k = n_k + digits_train[:, datapoint_id, class_index]
        n_k = n_k / TRAIN_m
        return n_k
    def train(self):
        digits_train = self.training_data
        num_classes = np.size(digits_train, 2)
        for i in range(num_classes):
            n_k = self.calculate_n_k(i)
            self.n_k_values.append(n_k)
```

4. Test Performance:

Gaussian Classifier	Naïve Bayes Classifier
Total Error: 721/4000 | 0.18025 Error by digit: Digit <<1>> : 69/400 Digit <<2>> : 81/400 Digit <<3>> : 63/400 Digit <<4>> : 61/400 Digit <<5>> : 68/400 Digit <<6>> : 44/400 Digit <<7>> : 63/400 Digit <<8>> : 109/400 Digit <<9>> : 110/400 Digit <<0>> : 53/400	Total Error: 939/4000 | 0.23475 Error by digit: Digit <<1>> : 87/400 Digit <<2>> : 104/400 Digit <<3>> : 91/400 Digit <<4>> : 85/400 Digit <<5>> : 111/400 Digit <<6>> : 60/400 Digit <<7>> : 89/400 Digit <<8>> : 121/400 Digit <<9>> : 133/400 Digit <<0>> : 58/400