1 K-Nearest neighbors classifier

1.1 The implementation of KNN

- First,calculate the euclidean distance between each test data and each of the other training data

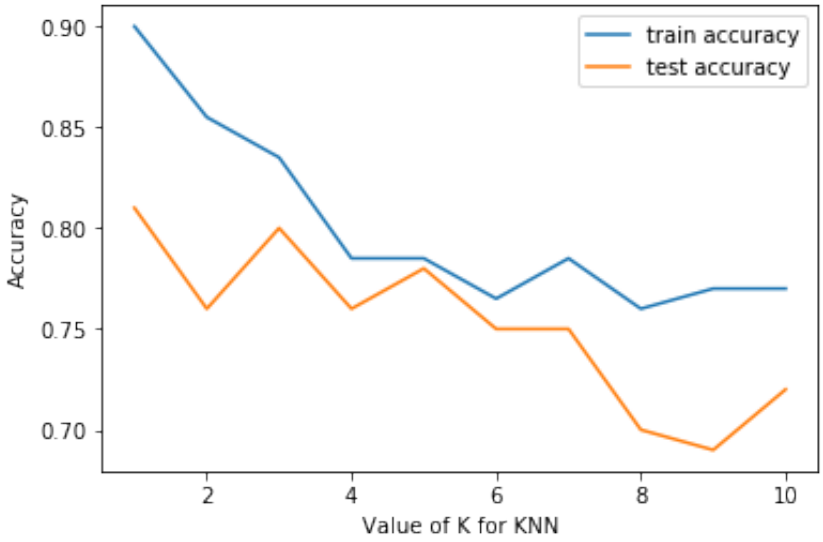
```
def compute_distances(self, X):
        num_test = X.shape[0]
        num_train = self.X_train.shape[0]
        x2 = np.sum(X ** 2, axis=1).reshape((num_test, 1))
        y2 = np.sum(self.X_train ** 2, axis=1).reshape((1, num_train))
        xy = -2 * np.matmul(X, self.X_train.T)
        dists = np.sqrt(x2 + xy + y2)
        # print(dists)
        return dists
```

- Second， enter the distance matrix of the test data and the training data to predict the category of each test data.

```
def predict_labels(self, dists, k=1):
        num_test = dists.shape[0]
        y_pred = np.zeros(num_test, dtype=int)
        for i in range(num_test):
            closest_x = np.argsort(dists[i])[:k]
            closest_y = [self.y_train[val] for val in closest_x]
            labels, counts = np.unique(closest_y, return_counts=True)
            y_pred[i] = labels[np.argmax(counts)]
        return y_pred
```

1.2 Explanation of results

Try different K values to predict the data, the results are as follows:



It can be seen from the above picture that as the value of K increases, the accuracy decreases.When K=1, the accuracy is the highest; when K=10, the accuracy is the smallest.

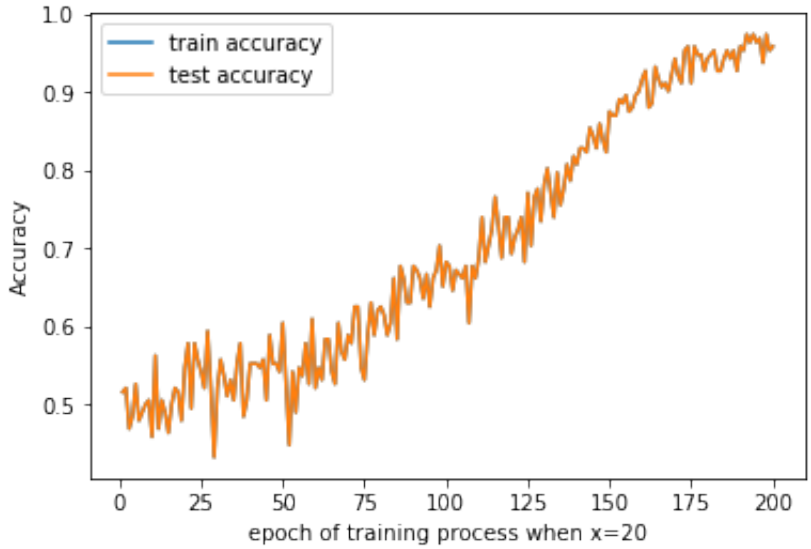
2 Convolutional neural network classifier

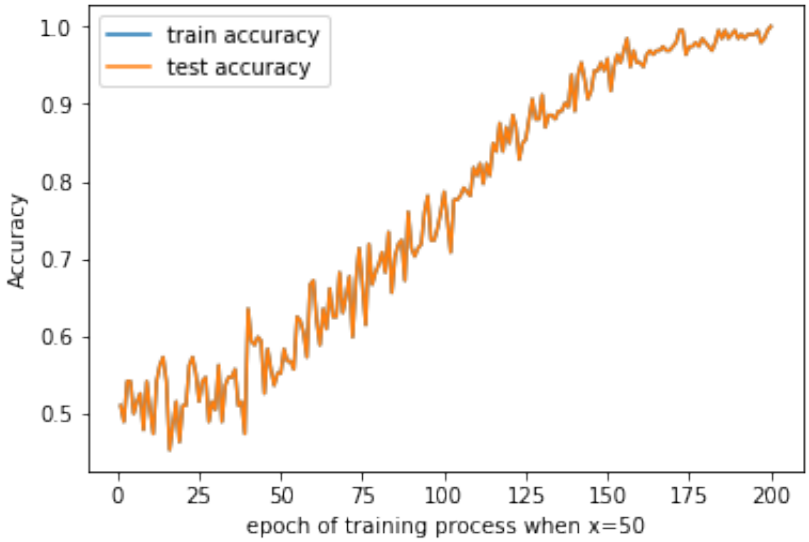
TensorFlow is used here to implement the LeNet network, with different hidden layers size (20, 50, 100) in the fully connected layer.

2.1 LeNet's network structure

Input → Layer 1 → ReLu → Pooling → Layer 2 → ReLu → Pooling → FC1 → ReLu → FC2 → ReLu → FC3 → Yhat (using Softmax)

2.2 The different training results





You may not see the different X training effects from the above picture, but you can see from the table in the next section that there may be over-fitting when X is larger.

3 Accuracy for Classifiers

Classifier	Training Accuracy	Testing Accuracy
K-NN(k=1)	0.9	0.81
K-NN(k=2)	0.855	0.76
K-NN(k=3)	0.835	0.8
K-NN(k=4)	0.785	0.76
K-NN(k=5)	0.785	0.78
K-NN(k=6)	0.765	0.78
K-NN(k=7)	0.785	0.75
K-NN(k=8)	0.76	0.7
K-NN(k=9)	0.77	0.69
K-NN(k=10)	0.77	0.72
CNN(x=20)	0.975	0.94
CNN(x=50)	0.99	0.9
CNN(x=100)	0.98	0.94