COMPSCI 371D Homework 7

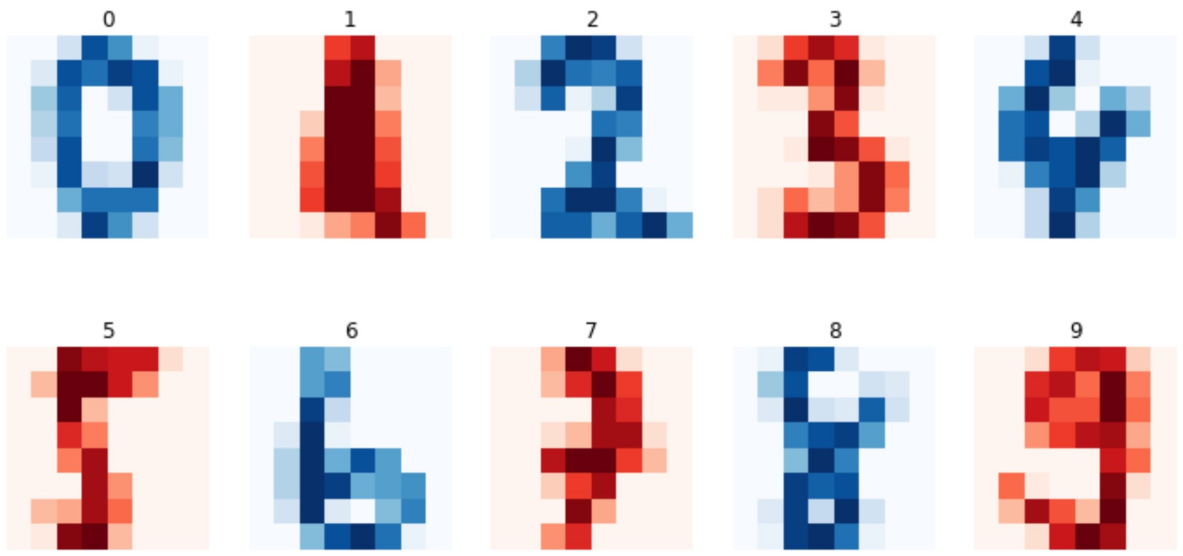
Part 1: Digit Classification

In [1]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

digits = datasets.load_digits()
images, values = shuffle(digits.images, digits.target, random_state=0)
parity = values % 2    # 1 odd, 0 even

plt.figure(figsize=(12, 6))
for digit in range(10):
    plt.subplot(2, 5, digit+1)
    index = np.argwhere(values == digit)[0][0]
    color_map = 'Reds' if parity[index] else 'Blues'
    plt.imshow(images[index, :, :], cmap=color_map)
    plt.axis('off')
    plt.title('{}'.format(digit))
```



In [2]:
```
data = {'x': images.reshape((len(images), -1)), 'y': parity}

test_fraction = 0.4
training_set, test_set = {}, {}
training_set['x'], test_set['x'], training_set['y'], test_set['y'] = \
    train_test_split(data['x'], data['y'], test_size=test_fraction, random_state=0)
```

In [3]:
```
def evaluate(h, train, test, name):
    def error_rate(predictor, samples):
        x, y = samples['x'], samples['y']
        return (1 - predictor.score(x, y)) * 100

    f = '{:s}: training error rate is {:.2f} percent on {} samples,' + \
        '\n\ttest error rate is {:.2f} percent on {} samples'
    info = (name, error_rate(h, train), len(train['y']), error_rate(h, test), len(t
est['y']))
    print(f.format(*info))
```

Problem 1.1

In [4]:
```
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
pred = LogisticRegression(C = 1e5, solver = 'lbfgs', max_iter = 10000, random_state
= 0)
pred.fit(training_set['x'], training_set['y'])
```

Out[5]:
```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=10000, multi_class='warn', n_jobs=None,
                   penalty='l2', random_state=0, solver='lbfgs', tol=0.0001,
                   verbose=0, warm_start=False)
```

In [6]:
```
evaluate(pred, training_set, test_set, 'logistic classifier')
```

```
logistic classifier: training error rate is 7.14 percent on 1078 samples,
        test error rate is 8.62 percent on 719 samples
```

Problem 1.2 (Exam-Style)

No, because if the training set was linearly separable, there would be an error rate of 0% for the training set, but since there isn't there is no classifier that could perfectly split up the classification.

Problem 1.3 (Exam Style)

Our classifier slightly overfits the training data. This can be seen by the fact that the error rate of the test set is slightly higher than the training error rate. If the classifier wasn't overfitted, we would hope to see the same or even a lower error rate for any new data.

Problem 1.4

In [7]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

```
In [8]: params = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}
        h = GridSearchCV(KNeighborsClassifier(), params, scoring='accuracy', iid=False, cv=
        15)
        h.fit(training_set['x'], training_set['y'])
```

```
Out[8]: GridSearchCV(cv=15, error_score='raise-deprecating',
                     estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                                    metric='minkowski',
                                                    metric_params=None, n_jobs=None,
                                                    n_neighbors=5, p=2,
                                                    weights='uniform'),
                     iid=False, n_jobs=None,
                     param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]},
                     pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                     scoring='accuracy', verbose=0)
```

```
In [9]: print('Using k = 3 as the best parameter')
        evaluate(h, training_set, test_set, 'k-NN classifier')
```

```
Using k = 3 as the best parameter
k-NN classifier: training error rate is 0.65 percent on 1078 samples,
        test error rate is 0.56 percent on 719 samples
```

Problem 1.5 (Exam Style)

No because if the k-NN classifier did overfit the training data, the classifier would perform worse on any data that wasn't the training data, but in this case, the test error rate was lower than the training error rate so that means the classifier was not overfit.

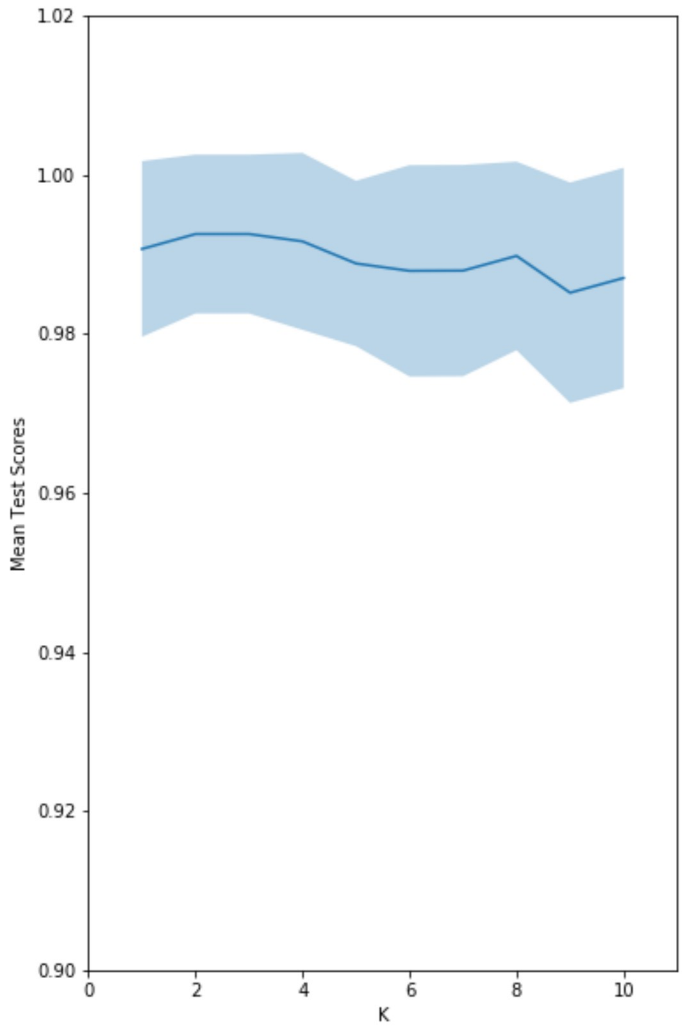
Problem 1.6

For the logistic-regression classifier, its test error rate was 8.62%, while the test error rate for the k-NN classifier was 0.56%, so the k-NN classifier performed much better. The k-NN classifier performed much better because it is easier to compare the numbers to similar numbers as compared to taking all of the data and turning it into one score.

Problem 1.7

```
In [10]: mean, std = h.cv_results_['mean_test_score'], h.cv_results_['std_test_score']
         k = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
         fig = plt.figure(figsize=(6,10))
         plt.plot(k, mean)
         plt.axis([0,11,.9,1.02])
         plt.fill_between(k, mean-std, mean+std, alpha=.3)
         plt.ylabel('Mean Test Scores')
         plt.xlabel('K')

Out[10]: Text(0.5, 0, 'K')
```



Problem 1.8

```
In [11]: bad_h = KNeighborsClassifier(n_neighbors=9)
         bad_h.fit(training_set['x'], training_set['y'])
         evaluate(bad_h, training_set, test_set, 'k-NN classifier')

         k-NN classifier: training error rate is 0.93 percent on 1078 samples,
                 test error rate is 1.25 percent on 719 samples
```

Using the worst hyper-parameter we get a test error rate of 1.25% which is about double the error rate of the best hyper-paramter. The value of k returned by GridSearchCV can be trusted because since the std. deviation is pretty consistant for each mean, that means that there was no outlier score for any k value which would exaggerate the mean score that we recieve.

Part 2: Set Convexity

Problem 2.1 (Exam Style)

For any set S, it is convex if for every $u, v \in S$ and $t \in [0,1]$:

$$tu + (1-t)v \in S$$

So let $C + D = E$, $c_1, c_2 \in C$, $d_1, d_2 \in D$, $e_1, e_2 \in E$, and $e_1 = c_1 + d_1$, $e_2 = c_2 + d_2$. C and D already have the property stated above since we know they are convex.

Then:

$$te_1 + (1-t)e_2 = t(c_1 + d_1) + (1-t)(c_2 + d_2) = tc_1 + td_2 + (1-t)c_2 + (1-t)d_2$$
$$= (tc_1 + (1-t)c_2) + (td_1 + (1-t)d_2) \in C + D = E$$

This means that E is convex.

Part 3: KKT Conditions

Problem 3.1 (Exam Style)

Lets find the Hessian matrix of f. This turns out to be $\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ which is postive semidefinite so f is convex.

Problem 3.2 (Exam Style)

$A(v_1) = \{1, 2\}$

$A(v_2) = \{2\}$

$A(v_3) = \{1\}$

$A(v_4) = \{\}$

Problem 3.3 (Exam Style)

$b_1, b_2 = 0$

$a_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$a_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Problem 3.4 (Exam Style)

(For v_1)

KKT Condition: $\begin{bmatrix} -2 \\ -2 \end{bmatrix} = \alpha_1 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

This can't be satisfied for any α_1, α_2 so v_1 is not a solution.

(For v_2)

KKT Condition: $\begin{bmatrix} -2 \\ 0 \end{bmatrix} = \alpha_1 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

This can't be satisfied, so v_2 is not a solution.

(For v_3)

KKT Condition: $\begin{bmatrix} 0 \\ -2 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

This can't be satisfied, so v_3 is not a solution.

(For v_4)

KKT Condition: $\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

This can be satisfied with $\alpha_1, \alpha_2 = 0$, so v_4 is a solution of the KKT Condition.

Problem 3.5 (Exam Style)

$\nabla f(v_2) = \begin{bmatrix} 2(u_1 + 1) \\ 2(u_2 - 1) \end{bmatrix}$

(v_4)

KKT Condition: $\begin{bmatrix} 4 \\ 0 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

This can be satisfied with $\alpha_1 = 4, \alpha_2 = 0$, so v_4 is a solution of the KKT Condition.

Problem 3.6 (Exam Style)

$$\nabla f(v_2) = \begin{bmatrix} 2(u_1+1) \\ 2(u_2+2) \end{bmatrix}$$

(v_4)

KKT Condition: $\begin{bmatrix} 4 \\ 6 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

This can be satisfied with $\alpha_1 = 4, \alpha_2 = 6$, so v_4 is a solution of the KKT Condition.