CS/ECE/ME532 Assignment 9

1. **Face Emotion Classification with Kernel Classifier.** In this problem you will apply a kernel classifier to the face emotion dataset. You may find it very helpful to use code from an activity.

   a) Build a kernel classifier using

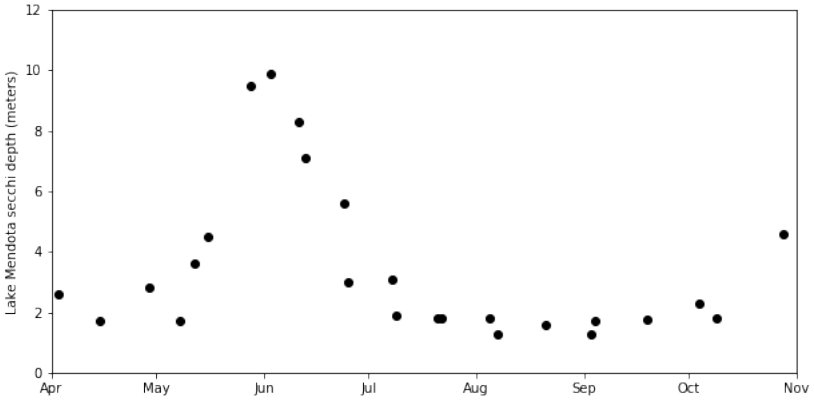
   - the squared error loss function
   - an ℓ_2 regularizer with $\lambda = 0.5$.
   - the Guassian Kernel $K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-||\boldsymbol{u} - \boldsymbol{v}||^2/(2\sigma^2)$.

   b) Train your classifier choosing for different values of σ and create a plot with σ on the horizontal axis and accuracy on the vertical axis and comment on the plot. Does your classifier achieve 0% training error?

   c) Find a more realistic estimate of the accuracy of your classifier by using 8-fold cross validation. Can you achieve perfect test accuracy?

2. **Kernel Regression, Lake Mendota Clarity**. The *Secchi depth* is a measure of water clarity obtained by lowering a black and white disk off the shady side of a boat and recording the depth at which the disk is no longer visible.

   A dataset obtained from the University of Wisconsin's Limnology department contains Secchi disk readings (in meters) on Lake Mendota from 2019 and 2020. A Secchi depth of less than 2 meters is consider poor clarity, while a Secchi depth greater than 6 meters is consider very clear. Lake Mendota can have very clear water in late spring when native zooplankton *daphnia pulicaria* consume large amounts of algae and phytoplankton (for more details, see https://blog.limnology.wisc.edu/2019/06/12/whats-behind-this-extended-phase-of-crazy-clear-water-in-lake-mendota/).



   a) Use kernel ridge regression with a Gaussian kernel to fit the measurements. You may find it useful to use code from an activity. Use regularization parameter

$\lambda = 0.01$ and scale parameter $\sigma = 10$. Plot the resulting fit, and comment on the results. Do these parameters overfit or underfit the data? Adjust the regularization parameter to find a visually better fit.

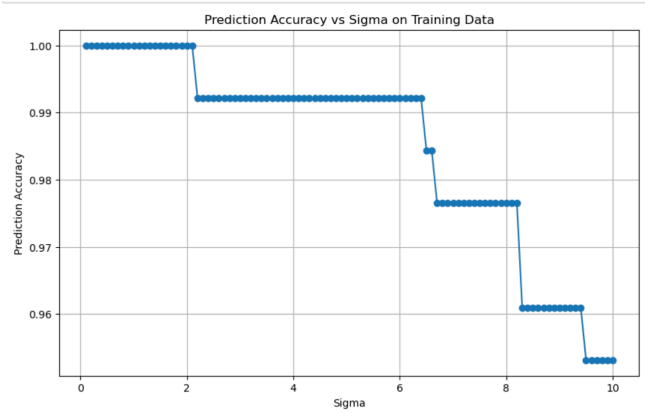
b) Describe how you could use k-fold cross validation to systematically find a good value of σ and λ.

1a

In [214]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import euclidean_distances
from scipy.io import loadmat

dataset = loadmat('face_emotion_data.mat')

x, y = dataset['X'], dataset['y']
n, p = np.shape(X)

X = np.hstack((np.ones((n,1)), x))  # append a column of ones

# Problem 1a - Devin Bresser

def gaussian_kernel(u, v, sigma):
    return np.exp(-euclidean_distances(u, v, squared=True) / (2 * sigma**2))

def get_K_and_alpha(X_train, y_train, sigma_, lmda_):
    K_train = gaussian_kernel(X_train, X_train, sigma_)
    alpha = np.linalg.inv(K_train + lmda_*np.eye(len(K_train))) @ y_train
    return K_train, alpha
```

1b

In [215]:
```
# Problem 1b

sigmas = np.linspace(0.1,10,100)
lmda = 0.5
accuracies = []

for sigma_ in sigmas:
    K, alpha = get_K_and_alpha(X, y, sigma_, lmda)
    predictions = np.sign(K @ alpha)
    accuracy = np.mean(predictions == y)
    accuracies.append(accuracy)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(sigmas, accuracies, marker='o')
plt.xlabel('Sigma')
plt.ylabel('Prediction Accuracy')
plt.title('Prediction Accuracy vs Sigma on Training Data')
plt.grid(True)
plt.show()
```



In [216]:
```
# Problem 1b comment: The classifier achieves 0 training error
# when sigma < approximately 2.1
# We observe that increasing sigma decreases the accuracy on the training data
# which makes sense, as it is a parameter that tunes how "smooth" the fit is.
```

1c

In [223]:
```
# Problem 1c

from sklearn.model_selection import KFold

kf = KFold(n_splits = 8)

sigmas = np.linspace(0.1,10,100)
lmda = 0.5

avg_accuracies = []

for sigma_ in sigmas:

    fold_accuracies = []

    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # Compute K and alpha for the training data
        K_train, alpha = get_K_and_alpha(X_train, y_train, sigma_, lmda)

        # Compute predictions on the test set
        K_test = gaussian_kernel(X_test, X_train, sigma_)
        predictions = np.sign(K_test @ alpha)

        # Compute accuracy
        accuracy = np.mean(predictions == y_test)
        fold_accuracies.append(accuracy)

    avg_accuracies.append(np.mean(fold_accuracies))

sigma_accuracies = list(zip(sigmas, avg_accuracies))
highest_accuracy_pair = max(sigma_accuracies, key=lambda pair: pair[1])
print(f"Highest avg. (8-fold CV) test accuracy: {highest_accuracy_pair[1]} achieved at sigma={highest_accuracy_pair[

# Comment: Sigma=3.4 gives the highest average testing accuracy 0.961
```

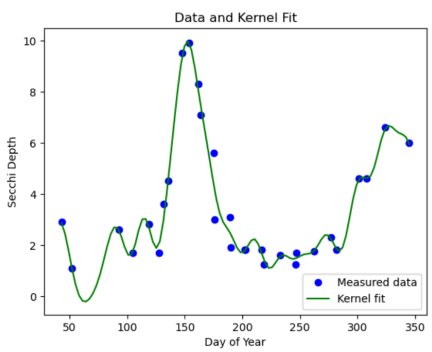
```
Highest avg. (8-fold CV) test accuracy: 0.9609375 achieved at sigma=3.4000000000000004
```

2a-i

In [13]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df = pd.read_csv('mendota_secchi_depth.txt', delimiter='\t')
x = df['day_of_year']
y = df['secchi_depth']
```

In [82]:
```
# Problem 1a

sigma = 10  # gaussian kernel parameter
lam = 0.01  # ridge regression parameter
n = len(x)  # number of data points
p = 100 # number of x-axis points for plotting
x_axis_pts = np.linspace(min(x), max(x), p)  # plotting points

# code from activity

distsq = np.zeros((n, n), dtype=float)

for i in range(n):
    for j in range(n):
        distsq[i, j] = (x[i] - x[j])**2

K = np.exp(-distsq / (2 * sigma**2))

alpha = np.linalg.inv(K + lam * np.identity(n)) @ y

distsq_x_axis_pts = np.zeros((p, n), dtype=float)

for i in range(p):
    for j in range(n):
        distsq_x_axis_pts[i, j] = (x_axis_pts[i] - x[j])**2

dtest = np.exp(-distsq_x_axis_pts / (2 * sigma**2)) @ alpha

print('Sigma = ', sigma)
print('Lambda = ', lam)
plt.plot(x, y, 'bo', label='Measured data')
plt.plot(x_axis_pts, dtest, 'g', label='Kernel fit')
plt.title('Data and Kernel Fit')
plt.legend(loc='lower right')
plt.xlabel('Day of Year')
plt.ylabel('Secchi Depth')
plt.show()
```

```
Sigma =  10
Lambda =  0.01
```

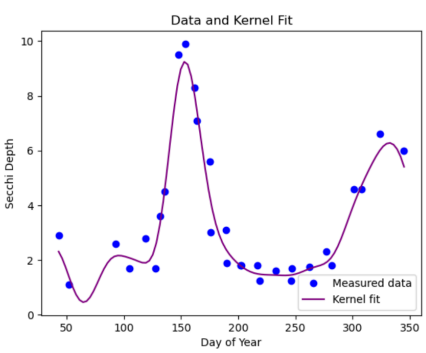


In [83]:
```
# Problem 1a comment: These parameters overfit the data in my opinion.
# We are modeling a lot of variation in the data in high detail,
# when in reality it is probably just random variation in the measurements.
# A smoother fit would predict new points more accurately.
```

2a-ii

```
# I have fine tuned sigma and lmda to get this result which captures the
# broader story of the data (a big spike around June) but smooths over
# some of the smaller variations in it.
# The sparsity of training data available around April-May causes a dip in that
# part of the curve that is difficult to remedy by adjusting the parameters.

# Problem 1a

sigma = 15  # gaussian kernel parameter
lam = 0.15  # ridge regression parameter
n = len(x)  # number of data points
p = 100 # number of x-axis points for plotting
x_axis_pts = np.linspace(min(x), max(x), p)  # plotting points
```

```
Sigma =  15
Lambda =  0.15
```



2b

1. Divide data into k folds (training and validation sets)
2. Create a 2d array with the ranges of lambda and sigma values that you want to test.
3. For each (sigma, lambda) pair, train the model on all of the folds (train on testing, test on testing) and compute the average mean squared error (MSE) on the testing points.
4. Select the (sigma, lambda) pair with the lowest average MSE on the testing points to be your model.