Support Vector Machine classifiers (1.5 points)

In this assignment, we use the scikit-learn package to train an SVM classifier. To do so, we need to tune 2 hyperparameters: the cost C and precision γ (gamma). We are going to use K-fold cross-validation to determine the best combination of values for this pair.

- Question 0 (.0) Have a look at the 3 first cells. In the third one, take note of how the `SVC` object is instantiated and trained, how labels are predicted, and finally how the fitting error is computed. In this assignment, the prediction error after a given training is simply defined as the *number of misclassified labels*.
- Question 1 (.9) Using an SVM classifier with an RBF kernel, use 10-fold cross-validation to find the best cost and precision parameters. The range of test values for each parameter is provided.
  - a. First compute the cross-validation error matrix: for each parameter combination, instantiate an SVM classifier; for each split provided by the `KFold` object, re-train this classifier and compute the prediction error; the cross-validation error is the average of these errors over all splits.
  - b. Use the error matrix to select the best parameter combination.
  - c. Visualize the error matrix using `imshow` and the 'hot' colormap.
- Question 2 (.5) Plot the decision boundaries of this classifier, by appropriately modifying the code from the previous assignments. Display the support vectors on the same figure.
- Question 3 (.1) Evaluate and print the generalization error of this classifier, computed on the test set.

Code

Imports

In [1]:

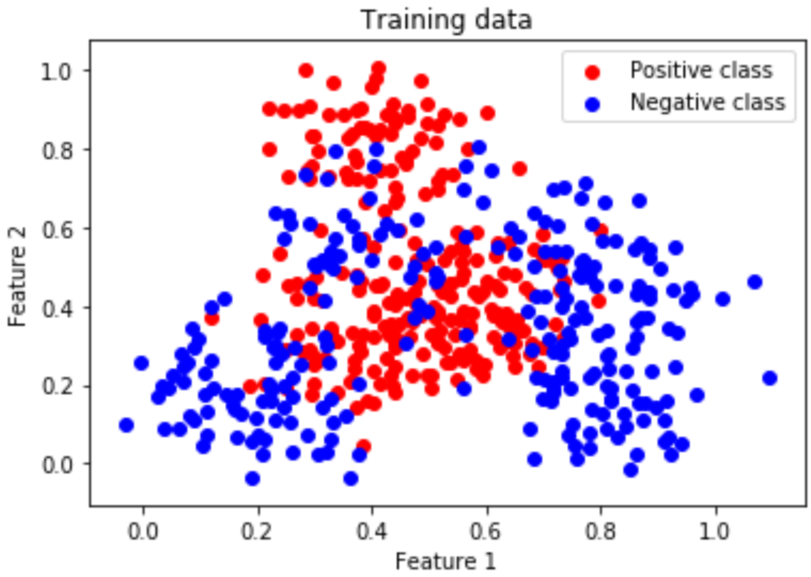
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import KFold
from sklearn.svm import SVC
%matplotlib inline
```

Load and display the training data

In [2]:

```
features = np.load("features.npy")
labels = np.load("labels.npy")
print("features size:", features.shape)
print("labels size:", labels.shape)

# Extract features for both classes
pos = labels == 1  # 1D array of booleans, with pos[i] = True if labels[i] == 1
features_pos = features[pos]  # filter the array with the boolean array
neg = labels != 1
features_neg = features[neg]

# Display data
fig, ax = plt.subplots()
ax.scatter(features_pos[:, 0], features_pos[:, 1], c="red", label="Positive class")
ax.scatter(features_neg[:, 0], features_neg[:, 1], c="blue", label="Negative class")
ax.set_title("Training data")
ax.set_xlabel("Feature 1")
ax.set_ylabel("Feature 2")
ax.legend()

plt.show()
```

```
features size: (500, 2)
labels size: (500,)
```



Training the SVM classifier with arbitrary hyperparameters

In [3]:

```
cost = 1
gamma = 1

# Train the SVM classifier.
svm = SVC(C=cost, kernel='rbf', gamma=gamma)
svm.fit(features, labels)

# Predict labels.
# Note that here we use the same set for training and testing,
# which is not the case in the remainder of the assignment.
predicted_labels = svm.predict(features)

# Compute the error.
# Note: since in Python, True and False are equivalent to 1 and 0,
we can
# directly sum over the boolean array returned by the comparison op
erator.
error = sum(labels != predicted_labels)
print("Prediction error:", error)
```

```
Prediction error: 98
```

Training with K-fold cross-validation

Define test values for the cost and precision parameters

In [4]:

```
def logsample(start, end, num):
    return np.logspace(np.log10(start), np.log10(end), num, base=10
.0)

num_gammas = 20
num_costs = 20
gamma_range = logsample(1e-1, 1e3, num_gammas)
cost_range = logsample(1e-1, 1e3, num_costs)
```

Compute the cross-validation error for each parameter combination

The `KFold` class from scikit-learn is a "cross-validation" object, initialized with a number of folds. For each fold, it randomly partitions the input data into a training set and a validation set. The [documentation (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html)](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html) provides an example of use.

In [5]:

```
K = 10  # number of folds for cross validation
kf = KFold(n_splits=K)
cv_error = np.zeros((num_gammas, num_costs))  # error matrix

# TODO (Question 1)
for i in range(num_gammas):
    for j in range(num_costs):
        svm = SVC(C=cost_range[j], kernel='rbf', gamma=gamma_range[
i])
        error=0
        for train_index,test_index in kf.split(features):
            svm.fit(features[train_index],labels[train_index])
            predicted_labels = svm.predict(features[test_index])
            error+=sum(labels[test_index] != predicted_labels)
        cv_error[i][j]=error/K
# /TODO (Question 1)
```

Train the classifier with the best parameter combination

In [6]:

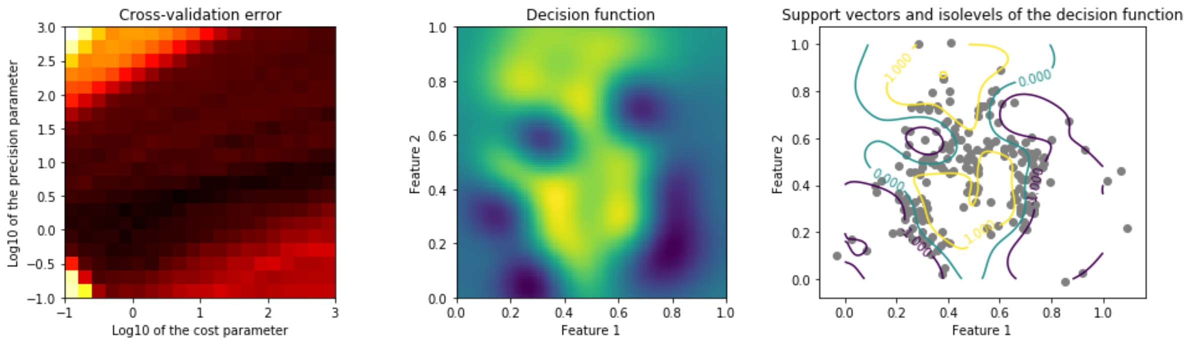
```
# Find gamma and cost giving the smallest error
# TODO (Question 1)
min_error=np.inf
for i in range(num_gammas):
    for j in range(num_costs):
        if cv_error[i][j]<=min_error:
            min_error=cv_error[i][j]
            m=i
            n=j
            continue
cost=cost_range[n]
gamma=gamma_range[m]
print (min_error)
# /TODO (Question 1)

# Train the SVM classifier using these parameters
svm = SVC(C=cost, kernel='rbf', gamma=gamma)
svm.fit(features, labels)
support_vectors = svm.support_vectors_
```

```
7.4
```

Display cross-validation results and decision function

In [7]:

```
# Sample points on a grid
num_points = 100
x_rng = np.linspace(0, 1, num_points)
y_rng = np.linspace(0, 1, num_points)
grid_x, grid_y = np.meshgrid(x_rng, y_rng)

# Evaluate decision function for each point
xy_list = np.column_stack((grid_x.flat, grid_y.flat))
values = svm.decision_function(xy_list)
values = values.reshape((num_points, num_points))

# Display
fig = plt.figure(figsize=plt.figaspect(0.25))

ax = fig.add_subplot(1, 3, 1)
ax.set_title("Cross-validation error")
ax.set_xlabel("Log10 of the cost parameter")
ax.set_ylabel("Log10 of the precision parameter")
# TODO (Question 1)

ax.imshow(cv_error,extent=[-1,3,-1,3],cmap=plt.cm.hot)

# /TODO (Question 1)

ax = fig.add_subplot(1, 3, 2)
ax.set_title("Decision function")
ax.set_xlabel("Feature 1")
ax.set_ylabel("Feature 2")
ax.imshow(values, extent=[0, 1, 0, 1], origin='lower')

ax = fig.add_subplot(1, 3, 3)
ax.set_title("Support vectors and isolevels of the decision function")
ax.set_xlabel("Feature 1")
ax.set_ylabel("Feature 2")
# TODO (Question 2)
CS=ax.contour(grid_x, grid_y, values, [-1.0,0.0,1.0], cmap=plt.cm.viridis)
ax.clabel(CS)
ax.scatter(support_vectors[:,0],support_vectors[:,1], c='gray', marker='o')
# /TODO (Question 2)

plt.show()
```



Generalization error

Load the test data

In [8]:

```
# Load the training data
test_features = np.load("test_features.npy")
test_labels = np.load("test_labels.npy")
print(test_features.shape)
print(test_labels.shape)
```

```
(500, 2)
(500,)
```

Print the number of misclassified points in the test set

In [9]:

```
# TODO (Question 3)
cost=cost_range[n]
gamma=gamma_range[m]

# Train the SVM classifier.
svm = SVC(C=cost, kernel='rbf', gamma=gamma)
svm.fit(test_features, test_labels)

# Predict labels.
# Note that here we use the same set for training and testing,
# which is not the case in the remainder of the assignment.
predicted_labels = svm.predict(test_features)

# Compute the error.
# Note: since in Python, True and False are equivalent to 1 and 0, we can
# directly sum over the boolean array returned by the comparison operator.
error = sum(test_labels != predicted_labels)
print("Prediction error:", error)

# /TODO (Question 3)
```

```
Prediction error: 81
```

In [ ]: