Assignment #3: Support Vector Machines [Group: max 2 students]

**1 Getting Started**Make sure you installed the sklearn Python library.  
sklearn contains some datasets for testing Machine Learning algorithms. You can download a  
first dataset with the intructions:  
from sklearn import datasets  
iris = datasets.load\_iris()

This dataset is made of examples of irises, each represented with a feature vector of dimension 4.  
The examples belong to one of 3 categories (setosa, versicolor, and virginica). The feature vectors  
contain the width and length of the sepal and of the petals.  
Retrieve this information by printing the iris variable.

You can store the feature vectors in a variable with  
iris\_X = iris.data  
and the labels with  
iris\_y = iris.target

Make sure you understand what iris\_X and iris\_y contain exactly.  
We will separate the data into training and test data with:

import numpy as np  
indices = np.random.permutation(len(iris\_X))  
iris\_X\_train = iris\_X[indices[:-40]]  
iris\_y\_train = iris\_y[indices[:-40]]  
iris\_X\_valid = iris\_X[indices[-40:-20]]  
iris\_y\_valid = iris\_y[indices[-40:-20]]  
iris\_X\_test = iris\_X[indices[-20:]]  
iris\_y\_test = iris\_y[indices[-20:]]

**3 Linear Support Vector Machine Classifier**You can try a linear support vector machine using the following code (svc is for support vector  
machine for classification):  
from sklearn import svm  
svc = svm.SVC(kernel=’linear’)  
svc.fit(iris\_X\_train, iris\_y\_train)  
and compare the output of  
svc.predict(iris\_X\_test)  
with  
iris\_y\_test  
Check the documentation at http://scikit-learn.org/stable/modules/generated/sklearn.  
svm.SVC.html#sklearn.svm.SVC to see how to change the parameter *C*, the weight between the  
classification score and the regularization term.  
The SVM classifier is a binary classifier. sklearn uses a One-vs-One strategy to apply it in  
multiclass classification problems (such as this Iris problem). Read the documentation: http://  
scikit-learn.org/stable/modules/multiclass.html#ovo-classification to learn about the  
one-vs-one strategy.  
Use the validation set (iris\_X\_valid, iris\_y\_valid) to identify the optimal value for *C*, at  
least given the validation set.

**4 Non-Linear Support Vector Machine Classifier**You can create a non-linear support vector machine classifier with Gaussian kernels using:  
svc = svm.SVC(kernel=’rbf’)  
See if you can tune *C* and *γ*, the standard deviation of the kernels to improve the classification  
results.  
**5 Visualization**For visualizing in 2D the effect of an SVM classifier, let consider only the 2 first coordinates of the  
feature vectors:

iris\_X = iris.data[:, :2]  
indices = np.random.permutation(len(iris\_X))  
iris\_X\_train = iris\_X[indices[:-40]]  
iris\_y\_train = iris\_y[indices[:-40]]  
iris\_X\_valid = iris\_X[indices[-40:-20]]  
iris\_y\_valid = iris\_y[indices[-40:-20]]  
iris\_X\_test = iris\_X[indices[-20:]]  
iris\_y\_test = iris\_y[indices[-20:]]  
We can create a plot of the regions created by the SVM with the following code:  
%matplotlib inline  
import matplotlib.pyplot as plt  
def plot\_contours(ax, clf, xx, yy, \*\*params):  
"""Plot the decision boundaries for a classifier.  
Parameters  
----------  
ax: matplotlib axes object  
clf: a classifier  
xx: meshgrid ndarray  
yy: meshgrid ndarray  
params: dictionary of params to pass to contourf, optional  
"""  
Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
out = ax.contourf(xx, yy, Z, \*\*params)  
return out  
def make\_meshgrid(x, y, h=.02):  
x\_min, x\_max = x.min() - 1, x.max() + 1  
y\_min, y\_max = y.min() - 1, y.max() + 1  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),  
np.arange(y\_min, y\_max, h))  
return xx, yy  
svc = svm.SVC(kernel=’rbf’, gamma=0.7, C=2)  
pred = svc.fit(iris\_X\_train, iris\_y\_train)  
U, V = iris\_X\_train[:, 0], iris\_X\_train[:, 1]  
xx, yy = make\_meshgrid(U, V)  
figsize = 10  
fig = plt.figure(figsize=(figsize,figsize))  
ax = plt.subplot(111)  
plot\_contours(ax, svc, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)  
ax.scatter(U, V, c=iris\_y\_train, cmap=plt.cm.coolwarm, s=20, edgecolors=’k’)  
ax.set\_xlim(xx.min(), xx.max())

ax.set\_ylim(yy.min(), yy.max())  
plt.show()  
Observe the effects of parameters *C* and *γ* on the regions.