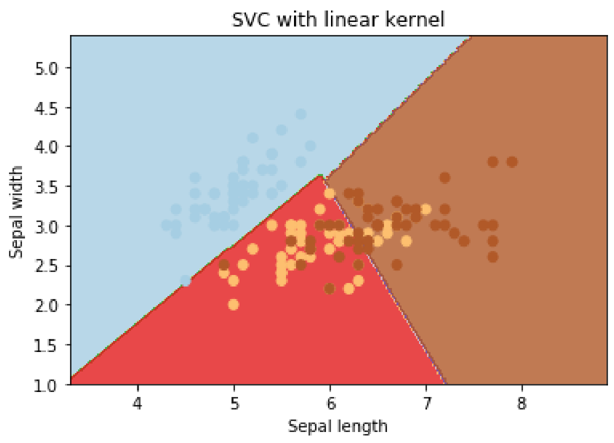
Machine Learning InClass Activity 6 - SVMs

Name: Denny Anderson Computind ID: dra2zp

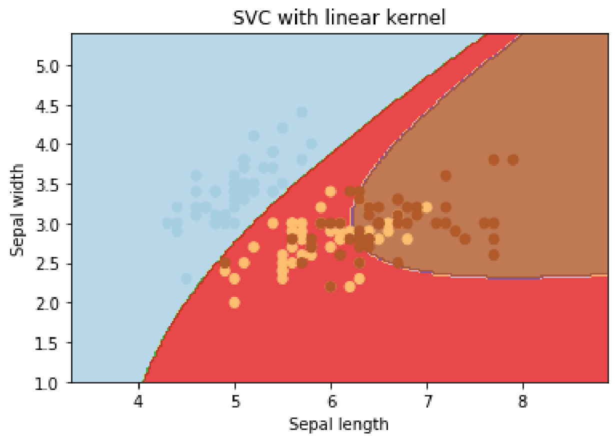
In [2]:
```
#Question 1

#Understand and run the code given in the above webpage. Capture the output of the code.
#Label and include the output in your report.
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features. We could
# avoid this ugly slicing by using a two-dim dataset
y = iris.target
C = 1.0 # SVM regularization parameter
svc = svm.SVC(kernel='linear', C=1,gamma=1).fit(X, y)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
plt.subplot(1, 1, 1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```



In [6]:
```
#Question 2

#Modify your code (add to it) to include an SVM example (on the same data set) that uses
#a polynomial kernel (of degree=3). Plot and label the result and include the new output in
#your report (you may submit your code as an appendix to the report - add comments!)
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features. We could
# avoid this ugly slicing by using a two-dim dataset
y = iris.target
C = 1.0
# SVM regularization parameter
svc = svm.SVC(degree=3,kernel='poly', C=1,gamma=1).fit(X, y)
#Here we change the kernel from linear to polynomial and include the parameter degree which we set to 3.
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
plt.subplot(1, 1, 1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```



Here we change the kernel from linear to polynomial and include the parameter degree which we set to 3.

Question 3: C: C is the penalty parameter of the error term. It can be considered as the cost for classification. If the value of C is too small, we will underfit and if it is too large, then we will overfit. It can also be considered as the factor that affects the complexity of the model. Kernel: A kernel is basically a similarity function. It takes a certain number of inputs and tells you how similar they are to each other. Having a kernel allows you to find the similarity between the inputs without having to define a large number of features. Also, computing the kernel is an easy task. Gamma: Gamma influences the Gaussian function used by the SVM. A high gamma leads to low variance and a low gamma leadds to high variance. It handles non-linear classification. Graphically, we can see that the value of gamma controls the shape of the peaks of the line of separation.

Question 4: A good way to handle multi-class classification using SVMs is to use a one vs all approach. This means that, suppose we have 3 classes A, B and C, we can always find a feature plane that separates A from B & C and recursively we can do the same for classes B and C.