k-NN on Iris data

- In this project, you will try applying k-nn classification method on the Iris data
- FYI, the k-NN model is in `sklean.neighbors.KNeighborsClassifier`
- set k to be 5

1. In this cell, load the iris data, and fit a k-nn (k=5) model to it. Predict the label of data point: (3,5,4,2), show the prediction result

In [66]:
```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets

iris = datasets.load_iris()
X_train = iris.data
Y_train = iris.target

#Splitting dataset into 75% train set and 25% test set
#from sklearn.model_selection import train_test_split
#X_train, X_test, Y_train, Y_test =  train_test_split(iris['data'], iris['targ
et'], random_state = 0)

k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, Y_train)

pred = np.array([[3,5,4,2]])

#Actual prediction of knn model
prediction = knn.predict(pred)
print prediction
print iris.target_names[prediction]
```

```
[1]
['versicolor']
```

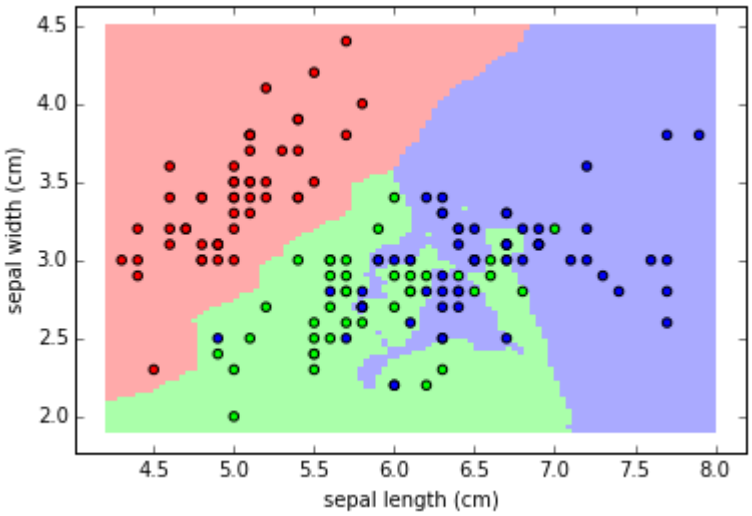
2. In this cell, you will try showcase your k-nn model. Here's what you need to do:

1. To allow visualization, we only use the **first two** features of the data points, remove the other features. Name the new-data set: `X` (for samples) and `Y` (for labels)
2. Fit a k-nn model to the new dataset, specify k as you like.
3. define:

```
x_min, x_max = X[:, 0].min() - .1, X[:, 0].max() + .1
y_min, y_max = X[:, 1].min() - .1, X[:, 1].max() + .1
```

4. Generate a `numpy.meshgrid` of uniformly distributed 100*100 points in the area of (x_min, x_max) $\times$ (y_min, y_max)
5. Plug in these 100*100 data points into the model and obtain their predicted labels: `Z`
6. Plot those 100*100 data points using `pyplot.pcolormesh` , give different classfied points different color, use `cmap=cmap_light`
7. Plot the points from the X as scatter points. give different classified points different color as well, use `cmap=cmap_bold`
8. Show the plot

Just for your reference, your plot should look like this, but not necessarily to be the same.



The cmaps are already defined.

In [67]:
```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
```

Write your code in the following cell:

In [68]:
```
X = iris.data[:, :2] # we only take the first two features.
Y = iris.target

k = 3
knn = KNeighborsClassifier(n_neighbors=k)
# we create an instance of Neighbours Classifier and fit the data.
knn.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 = .05 # step size in the mesh
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max,
100))
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
#plt.set_cmap(plt.cm.Paired)
cmap = plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
plt.scatter(X[:,0], X[:,1],c=Y, cmap=cmap_bold, edgecolor='k', s=20)
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks()
plt.yticks()

plt.show()
```