Music Instructments

April 3, 2018

In [1]:
```
import librosa
import librosa.display
import matplotlib.pyplot as plt
from scipy import signal
import IPython.display as ipd
import numpy as np
from scipy.io import wavfile
import peakutils

%matplotlib notebook

plt.ion()
```

In [2]:
```
# sample of instructment from http://www.philharmonia.co.uk/explore/sound_samples

# Load sample of french horn and violin
(sample_french_horn, sr_french_horn) = librosa.load("mp3/viola/viola_A3_1_forte_arco-nor

(sample_violin, sr_violin) = librosa.load("mp3/violin/violin_A3_1_forte_arco-normal.mp3"
```

In [3]:
```
ipd.Audio(rate=sr_french_horn,data=sample_french_horn)
```

Out[3]:
```
<IPython.lib.display.Audio object>
```

In [4]:
```
ipd.Audio(rate=sr_violin,data=sample_violin)
```

Out[4]:
```
<IPython.lib.display.Audio object>
```

In [5]:
```
# Plot the wave form of french horn
t = np.arange(len(sample_french_horn))/sr_french_horn
plt.figure()
plt.plot(t, sample_french_horn)
#plt.xlim(0.2, .21)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

iris_KNN {Ming}

April 3, 2018

0.1 Implement the K-Neareast Neighbor classifer as in Machine Learning Recipes

In [819]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
```

In [820]:
```
iris = datasets.load_iris()
```

In [821]:
```
x = iris.data
y = iris.target
```

In [822]:
```
# split data into 80% of training set and 20% of testing set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [823]:
```
classifier = KNeighborsClassifier()

classifier.fit(x_train, y_train)
```

Out[823]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [824]:
```
classifier.score(X=x_test, y=y_test)
```

Out[824]:
```
1.0
```

In [825]:
```
y_pred = classifier.predict(x_test)
```

In [826]:
```
confusion_matrix(y_pred=y_pred, y_true=y_test)
```

Out[826]:
```
array([[11,  0,  0],
       [ 0,  9,  0],
       [ 0,  0, 10]])
```

Yes_and_No_and_Up_and_Down_Spectrum-The-Real-Copy

April 3, 2018

In [1]:
```
import os
import random
import numpy as np
import librosa

from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix

YES_DIR="data/train/audio/yes"
NO_DIR="data/train/audio/no"
UP_DIR="data/train/audio/up"
DOWN_DIR="data/train/audio/down"

#X = [[0., 0.], [1., 1.]]
#y = [0, 1]
```

In [2]:
```
# list all the "yes" .wav files
yes_files = os.listdir(YES_DIR)

# and make them full path names
# adds attribute filename
yes_files = [os.path.join(YES_DIR, filename) for filename in yes_files]

# list all the "no" .wav files
no_files = os.listdir(NO_DIR)

# and make them full path names
no_files = [os.path.join(NO_DIR, filename) for filename in no_files]

#list all the "up" .wav files
up_files = os.listdir(UP_DIR)

#and make the full path names
up_files = [os.path.join(UP_DIR, filename) for filename in up_files]
```