Machine Learning, with scikit-learn

Jeffrey Skonhovd

Georgia Institute of Technology

May 19, 2014

Outline

Who am I?

- Jeffrey Skonhovd
- Works at FTN Financial
- Twitter: @jskonhovd
- Github: jskonhovd

Overview

- What is Machine Learning?
  - Machine Learning is the study of computer algorithms that improve automatically through experience.
- How should I go about learning Machine Learning?
  - MOOCs
  - Don't get caught up in the implementations.
- Tools
  - WEKA
  - scikit-learn

Types

- Supervised Learning
  - Supervised Learning is the task of inferring a function from labeled training data.
- Unsupervised Learning
  - Unsupervised Learning is the tasks of finding hidden structure in unlabeled data.
- Reenforcement Learning
  - Reenforcement Learning is concerned with how agents ought to take actions in an environment as to maximize some notion of cumulative reward.

Some Boring, but important Definitions.

- Inductive Bias
  - The inductive bias of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered.
  - Occam's Razor assumes that the hypotheses with the fewest assumptions should be selected.
- Cross-validation
  - The basic idea of Cross-validation to leave out some of the data when fitting the model.

Scikit-learn

- Scikit-learn is a set of simple and efficient tools for data mining and data analysis.
- Uses Python!!!
- `http://scikit-learn.org/`

Decision Trees

- Decision Tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented a decision tree.
- Maximize Information Gain
  - Information Gain measures how well a given attribute separates the training examples according to their target classifcation.

Decision Trees: Example


```
import numpy as np
import pylab as pl
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
# Parameters
# Load data
iris = load_iris()
clf = DecisionTreeClassifier()
X = iris.data[:, [1, 2]]
y = iris.target
clf = clf.fit(X, y)
plotCustom(X, y, [1, 2], clf)‘
```

kNN: Example


```
from sklearn import neighbors
import numpy as np
import pylab as pl
from sklearn import cross_validation
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:, [1, 2]]
y = iris.target
clf = neighbors.KNeighborsClassifier(3, 'distance')
plotCustom(X, y, [1,2], clf)
```

SVM


```
from sklearn import svm
import numpy as np
import pylab as pl
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:, [1, 2]]
y = iris.target
C = 1.0
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C)
rbf_svc.fit(X,y)
plotCustom(X, y, [1,2], rbf_svc)
```

kMeans


```
import numpy as np
import pylab as pl
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
n_digits = len(np.unique(y))
kmeans = KMeans(init='k-means++', n_clusters=n_digits
kmeans.fit(X)
kmeans_plots(X,y,[2, 3],kmeans)
```

- Resources