CIS192 Python Programming

Machine Learning in Python

Robert Rand

University of Pennsylvania

October 22, 2015

Outline

Installation

Recommended:

```
pip install -U numpy scipy ipython scikit-learn
```

You should install pip first if you don't have it.

Packages like Anaconda and Canopy also include the relevant libraries.

Numpy and Scipy

- Libraries for sophisticated mathematics and mathematical computing in Python.
- Include libraries for linear algebra.
- Optimized for efficient machine learning.

Matplotlib

- Library for plotting datasets.
- Use it to look at data before attempting machine learning techniques.
- Interfaces well with the IPython (an alternative shell for Python development).
- Should come bundled with scikit-learn, otherwise add `matplotlib` to the pip command on slide 1.

Scikit Learn

- A machine learning library for Python.
- Uses numpy and scipy.
- Comes with a broad array of built in machine learning algorithms

Outline

Coats and Bars

Suppose you didn't know the meaning of "boat" or "car" but you had a stack of photographs of boats and cars.

Could you somehow sort them into two stacks, which corresponded to boats and cars?

K-Means

- `cluster.KMeans()`
  - Parameter: `n_clusters` - specifies the number of clusters desired.
- Randomly assign initial position for each cluster.
- Repeat until stable:
  - Assign every point to its closest cluster c_i.
  - Move c_i to the center of points that are assigned to it.

  Every point is labeled with its cluster.

Other Models

- Gaussian Mixture Models general K-Means - K-means assumes clusters look like circle, where GMM can handle arbitrary elliptic clusters.
- Affinity Propagation ((cluster.AffinityPropagation()) identifies *exemplars* - points that can stand in for a given cluster. It may vary the number of clusters depending on the exemplars it finds.

Outline

Is that a squirrel?

Often, we *do* have labels. But learning is still a problem. Machine Learning is about *generalizing* from the example's you've seen (say, 100 pictures of squirrels) to things you haven't yet seen.

Squirrel

Squirrel

Squirrel?

K Nearest Neighbors

- `neighbors.KNeighborsClassifier`
  - Parameter: `n_neighbors` - specifies the number of neighbors k.
- What it sounds like - looks for the k points most similar to a given point in the data and returns the most common label of those points.
- Where $k = 1$ matches the closest point (high variance).
- Where $k = n$ always returns the most common label (high bias).
- Slow: Compares points to every point in the training data.

Decision Trees

- `tree.DecisionTreeClassifier`
  - Parameter: `max_depth` - specifies the maximum tree depth (we can alternatively specify `max_leaf_nodes`) .
- Each nodes splits the data according to a specific feature.
- Greater tree height -> more variance.
- Smaller tree height -> more bias.

Support Vector Machine

- eg. `svm.SVC, svm.LinearSVC`
  - Parameter: `kernel` - a function that specifies the form of the separation
- Carves up the space using *support vectors* - lines of maximum thickness that divide the space into two.
- Primarily for binary decision problems but can also be used in stages for nary classification.

Naive Bayes

- eg. `naive_bayes.GaussianNB`
  - Parameter: `max_depth` - specifies the maximum tree depth (we can alternatively specify `max_leaf_nodes`) .
- A powerful and efficient algorithm that assumes *independence* between features.
- User specifies the assumed underlying distribution - Gaussian, Bernoulli etc.
- Classifies points using Bayes' Theorem.

Outline

Tips and Tricks

- Visualize the data before running an algorithm on it - what kind of approach is appropriate to the problem.
- Partition your data (randomly!) into 3 sets of data:
  - Training (80%): The core training data.
  - Cross Validation (10%): Data left out, test the model with different parameters on it.
  - Test Data (10%): Also withheld, used to determine the ultimate accuracy of the model.
- You'll generally find that your model has either too much bias or too much variance - try to find a sweet spot.
- Avoid *overfitting* - this is generally the point where accuracy on your training set is substantially better than on your cross-validation set.
- Don't become too attached to one algorithm - no amount of tweaking will make the wrong algorithm into the right one.