CIS192 Python Programming
Machine Learning (part two)

Robert Rand

University of Pennsylvania

October 22, 2015
Outline

1. Review
   - Supervised vs. Unsupervised
   - Variance vs. Bias
   - Previous Models

2. Classification

3. Regression

4. General Tips
**Unsupervised Learning** has no knowledge of the labels, and generally seeks to *cluster* related points.

- Eg. K-Means

**Supervised Learning** has *training data* with labels attached. We want to extrapolate from that data to new data.

- We’ve seen the K-Nearest Neighbor and Decision Tree approaches.
- We’ll see more in this class.
Variance vs. Bias

- **Bias** is error that emerges from incorrect assumptions in the learning model.
- **Variance** is error that emerges from oversensitivity to small fluctuations in the training data.
- The more important the weight of a single datapoint, the higher the variance.
- In K-Nearest Neighbors, a higher k means more bias.
- In Decision Trees, a greater tree height means more variance.
- **Overfitting** occurs when your model is more attuned to the noise in your dataset than the actual underlying pattern.
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.
K-Means cont.

- K-Means is a form of unsupervised learning
- K-Means assumes the data is grouped into (N-dimensional) spherical clusters.
- To fix the problem of one axis being larger than others, use scaling
- For more complex distributions we use Gaussian Mixture Models
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.
- High variance when $k = 1$
- High bias when $k$ is proportional to the size of the dataset.
- 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.
Outline

1. Review
   - Supervised vs. Unsupervised
   - Variance vs. Bias
   - Previous Models

2. Classification

3. Regression

4. General Tips
Classification vs. Regression

- **Classification** assigns each data point to one of $n$ distinct groups.
- **Regression** assigns each data point a real number.
  - Eg. a probability in $(0, 1)$ or an estimated height in $(0, 8)$ feet.
- So far we’ve only seen classification algorithms.
Naive Bayes

- **eg.** `naive_bayes.GaussianNB`, `naive_bayes.MultinomialNB`

- A powerful and efficient algorithm that assumes \textit{independence} between features.

- User specifies the assumed underlying distribution - Gaussian, Bernoulli etc.

- Classifies points using Maximum Likelihood Estimation (MLE) of $P(x, y)$ via $P(x|y)$ and $P(y)$. 
Logistic Regression

(Note: Generally used for classification, not regression, despite its name.)

- **eg.** `linear_model.LogisticRegression`
  - Parameter: `solver` - Specifies the mathematical method used to estimate MLE
  - Tries to directly calculate $P(y \mid x)$
  - Uses an iterative technique like Gradient Descent to estimate MLE.
  - Finds a linear boundary between the two points being classified.
  - Primarily for binary decision problems but can also be used in stages for nary classification.
Support Vector Machines

- 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.
Outline

1 Review
   - Supervised vs. Unsupervised
   - Variance vs. Bias
   - Previous Models

2 Classification

3 Regression

4 General Tips
Linear Regression

- **eg.** `linear_model.LinearRegression`
- Assumes the output is a linear function of the input.
  - $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ for some $\theta$s.
  - Note that we can handle polynomials simply by adding $x_i^2, x_i^3$ etc. to the prediction data.

- We penalize functions by their euclidean ($L_2$) distance from the line to the point.
Outline

1 Review
   - Supervised vs. Unsupervised
   - Variance vs. Bias
   - Previous Models

2 Classification

3 Regression

4 General Tips
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.