

CIS192 Python Programming

Supervised Learning

Harry Smith

University of Pennsylvania

October 25, 2017



Outline

- 1 ML: Concepts
- 2 ML: Classification
- 3 ML: Regression
- 4 ML: General Tips



Supervised vs. Unsupervised Learning

- *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.
 - ▶ K-NN, Decision Tree, NB, Logistic Regression, SVM
 - ▶ Linear Regression



Boats and Cars Redux

- Suppose we have a stack of photos, and we know for each one whether it's a coat or a bar.
- Can we use this information to decide whether future photographs are coats or bars?
 - ▶ (yes)



Today's Goals and Motivation

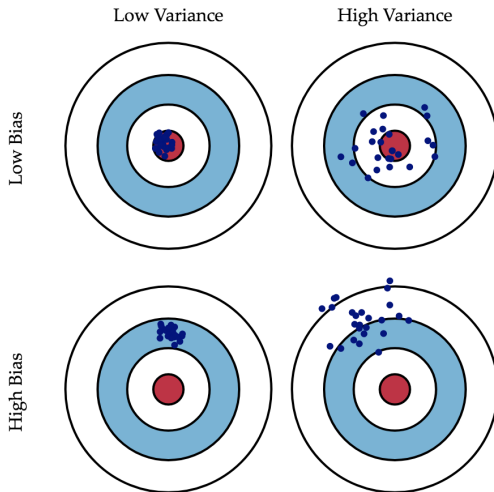


Variance vs. Bias

- *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.



Variance and Bias, Graphically



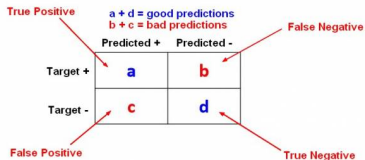
Classification vs. Regression

- *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.



Precision vs. Recall

- *Precision* is the ability of the classifier to not label a negative sample as positive.
- *Recall* is the ability of a classifier to find all positive samples.



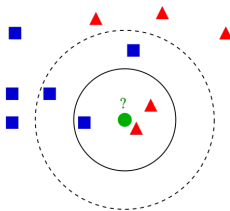
Outline

- 1 ML: Concepts
- 2 ML: Classification
- 3 ML: Regression
- 4 ML: General Tips



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.



Naive Bayes

- eg. `naive_bayes.GaussianNB`, `naive_bayes.MultinomialNB`
- A powerful and efficient algorithm that assumes *independence* between features.
- User specifies the assumed underlying distribution - Gaussian, Bernoulli etc.
- Multinomial NB is used frequently in text classification (hint, hint)
- Classifies points using Maximum Likelihood Estimation (MLE) of $P(x, y)$ via $P(x|y)$ and $P(y)$.



Logistic Regression

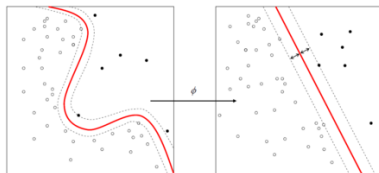
(Note: Generally used for classification, not regression, despite its name.)

- eg. `linear_model.LogisticRegression`
 - ▶ Parameter: `solver` - Specifies the mathematical method used to estimate MLE
- Tries to directly calculate $P(y | x)$
- Minimizes a cost function across all possible choices (e.g. what's the choice that I can make for this point that's the least likely to be wrong?)
- 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 (or n).
- Versatile, memory efficient, and excel in high-dimensional spaces.
- Struggle for high feature to dimension ratios, relatively opaque (lack clear probability functions).



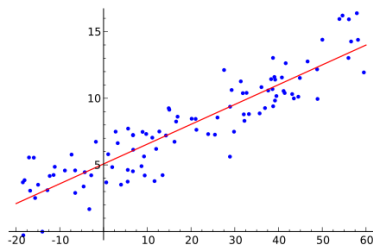
Outline

- 1 ML: Concepts
- 2 ML: Classification
- 3 ML: Regression
- 4 ML: General Tips



Linear Regression

- eg. `linear_model.LinearRegression`
- Assumes the output is a linear function of the input.
 - ▶ $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$ for some θ 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 ML: Concepts
- 2 ML: Classification
- 3 ML: Regression
- 4 ML: 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.

