Machine Learning

1. Supervised
   1. Regression
      1. Linear Models
         1. Basic
         2. Regularized
            1. Lasso (L1)
            2. Ridge (L2)
            3. Elastic Net
   2. Classification
      1. KNN ("K Nearest Neighbors)
      2. Logistic Regression
         1. Basic (Binary)
         2. Multinomial
2. Unsupervised
   1. Clustering
      1. Density Models (DB Scans)
      2. Centroid Models (K Means)
      3. Connectivity Models (Hierarchical)
         1. Decision Trees
         2. Random Forests
      4. Distribution Models

Bias v. Variance

* Bias = Difference between the predicted value and the actual value
* Variance = Difference between each of the predicted values
* Bias and Variance are inversely related
* Minimizing bias, at the expense of greater variance, makes models more generalizable/ useful

Supervised: Regression: Linear Models: Basic

Def: Explanation of a continuous variable given a series of independent variables

The simplest version is just a line of best fit: y = mx + b. Explains the relationship between x and y, giving starting point b and explanation power of m.

Linear regression works best when:

* The data is normally distributed (but doesn’t have to be)
* X’s significantly explain y (have low p-values)
* X’s are independent of each other (low multicollinearity)
* Resulting values pass linear assumption (depends upon problem) 🡨["The relationship is actually linear."]

If data is not normally distributed, we could introduce bias.

General format for **sklearn** model classes and methods

* *# generate an instance of an estimator class*estimator = base\_models.AnySKLearnObject()
* *# fit your data*estimator.fit(X, y)
* *# score it with the default scoring method (recommended to use the metrics module in the future)*  
  estimator.score(X, y)
* *# predict a new set of data*estimator.predict(new\_X)
* *# transform a new X if changes were made to the original X while fitting*estimator.transform(new\_X)

**EVALUATING MODEL:**

**R2** is a measure of how well the line fits the data, but is NOT a measure of the size of the error between real and predicted. Adjusted R2 is a little lower than R2 and takes into account the fact that an R2 can be made 'artificially high' just by using lots and lots of data.

For linear models, **residual errors** are the differences between actual and predicted values. Distribution of the residual errors should be normal with median = 0

**Mean Squared Error (MSE)** is the mean of the squared residual errors. A measure of the size of the error.

Supervised: Regression: Linear Models: Regularized

Regularization = a technique to improve the generalizability of a learned model / to reduce the possibility of overfitting the model.

Methods:

1. Lasso [least absolute shrinkage /selection operator] (L1)
   1. Method that performs variable selection and regularization
   2. Weights to potentially zero
2. Ridge (L2)
   1. xxx
3. Elastic Net
   1. Xxxx

Questions about regularizataion

**GRADIENT DESCENT**

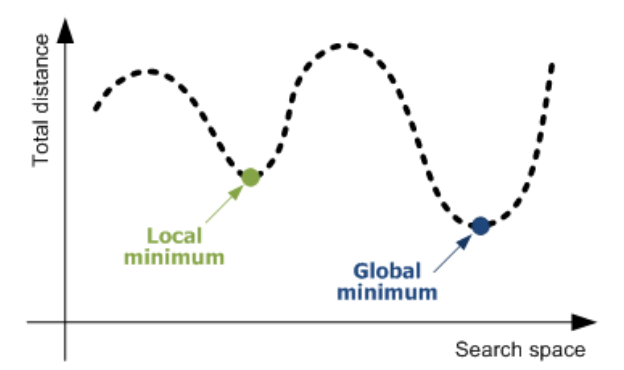
A random linear solution is provided as a starting point. The solver attempts to find a next “step”: take a step in any direction and measure the performance. If the solver finds a better solution (i.e. lower MSE), this is the new starting point. Repeat these steps until the performance is optimized and no “next steps” perform better. The size of steps will shrink over time.

lm = linear\_model.SGDRegressor()

# SGD is Stochastic Gradient Descent

Gradient Descent works best when:

* We are working with a large dataset. Smaller datasets are more prone to error.
* Data is cleaned up and normalized.
* Gradient Descent is significantly faster than OLS. This becomes important as data gets bigger.
* Potential pitfall – might settle on a local minimum as the solution rather than global minimum.



Tuning parameters are:

* the learning rate: how aggressively we solve the problem
* epsilon: at what point do we say the error margin is acceptable
* iterations: when should be we stop no matter what

Supervised: Classification: K-Nearest Neighbors

K Nearest Neighbors (KNN) is a classification algorithm that makes a prediction based upon the closest data points.

The KNN algorithm:

1. For a given point, calculate the distance to all other points.
2. Given those distances, pick the k closest points.
3. Calculate the probability of each class label given those points.
4. The original point is classified as the class label with the largest probability (“votes”).

Parameters to be adjusted are 1) the number of neighbors to consider and 2) whether to use weights. Weights can differentiate whether all neighbors are considered equally or whether some are seen as more predictive based on something like distance.

CROSS VALIDATION

k-fold cross validation

* Split the data into k groups
* Train the model on all segments except one
* Test model performance on the remaining set
* If k = 5, split the data into five segments and generate five models.

kf = cross\_validation.KFold(len(modeldata), n\_folds=5, shuffle=True)

mse\_values = []

scores = []

n= 0

print "~~~~ CROSS VALIDATION each fold ~~~~"

for train\_index, test\_index in kf:

lm = linear\_model.LinearRegression().fit(modeldata.iloc[train\_index], y.iloc[train\_index])

# initializes a model and then immediately fits it using training data.

mse\_values.append(metrics.mean\_squared\_error(y.iloc[test\_index], lm.predict(modeldata.iloc[test\_index])))

# gets MSE values for test data related to the training data, and adds them to a list.

# MSE function needs true y and predicted y

scores.append(lm.score(modeldata, y))

# gets R2 values for test data related to the training data, and adds them to a list.

n+=1

print 'Model', n

print 'MSE:', mse\_values[n-1]

print 'R2:', scores[n-1]

print "~~~~ SUMMARY OF CROSS VALIDATION ~~~~"

print 'Mean of MSE for all folds:', np.mean(mse\_values)

print 'Mean of R2 for all folds:', np.mean(scores)

**GRIDSEARCH**

A method of repeating a model many times with different tuning parameters to find the best set of parameters. A version of model selection.

**What are list comprehensions?**

<http://treyhunner.com/2015/12/python-list-comprehensions-now-in-color/>

List comprehensions are a tool for transforming one list (any [iterable](https://docs.python.org/3/glossary.html#term-iterable) actually) into another list. During this transformation, elements can be conditionally included in the new list and each element can be transformed as needed.

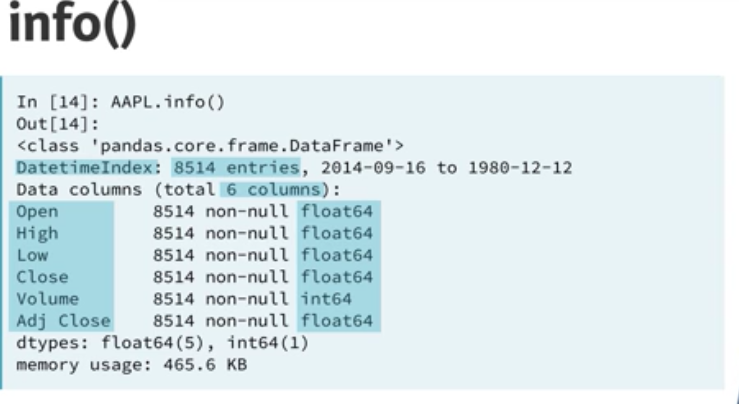
If you’re familiar with functional programming, you can think of list comprehensions as syntactic sugar for a filter followed by a map:

>>> doubled\_odds = map(lambda n: n \* 2, filter(lambda n: n % 2 == 1, numbers))

>>> doubled\_odds = [n \* 2 for n in numbers if n % 2 == 1]

Useful Pandas DataFrame methods

.info()



Broadcasting is assigning values to cells in DF?