SI 670 Notes

Suggested books

- Introduction to Machine Learning with PythonA Guide for Data ScientistsBy Andreas C. Müller and Sarah Guido
- Deep Learning with Pythonby Francois Chollet

Top libraries

- Scikit-learn
- SciPy
- Numpy
- Pandas
- Matplotlib

Cycle

- Feature representation
- Training
- Evaluation
- Refine cycle (hyperparameterization)

Data quality checks

- Min/max summaries
- Wrong data type, units
- Equal class reppresentation
- Outliers
- Data distribution
- Correlations among variables

KNN Notes

Category

- Supervised
  - Classification
  - Regression

High level algorithm

Given a training set X-train with labels y-train, and given a new instance x-test

1. Find the observations that resamble x-test that are in X-train. Call this set of observation(s) Xnn
2. Get the labels of Ynn for the instances in Xnn
3. Predict label for x-test by combining the labels Ynn (majority vote).

Parameters

- Distance metric (Euclidian)
- Choice of k (k=1 very flexibel, k=100 rigid)
- Weighting function (neighbors that are far less influence on final prediction)

Evaluation

- Accuracy (correctly predicted / total observations) (for classification)
- R^2 (for regression, measure how does the data fit the model 0-1)

Extras

- Ensure that all observations are on the same scale
  - if not standardize them (standard scalar)

Classification

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_C1, y_C1, random_state = 0)
knnc = KNeighborsClassifier(   n_neighbors = 5).fit(X_train, y_train)
print(knnc.predict(X_test))
print('Accuracy test score: {:.3f}'.format(knnc.score(X_test, y_test))
```

Regression

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =train_test_split(X_R1, y_R1, random_state = 0)
knnreg  = KNeighborsRegressor(n_neighbors =5).fit(X_train, y_train)

print(knnreg.predict(X_test))
print('R-squared test score: {:.3f}'.format(knnreg.score(X_test, y_test)
```

Linear Regression

Supervised

By now it should be clear that when we have a qunatitative, or qualitative response, and we want to train a model and then predict new,unseen observations with our model we are in the world of supervised learning.

Assumption

Now one of the main assumptions with this type of learning is that we expect the training data to have the same structure/relationships as the test data.

Linear Regression

Is type of linear model in which we attempt to predict the target value (quantitative response) with a weighted sum of features (predictors). The usual formula is Y = b0 +B1X1+E

- the goal is to minimize the residual sum of squares, which is the sum of the squared differences between y and y^, or actual minus predicted.

Evaluation

- Rˆ2, a.k.a *coefficient of determination* measures how well a prediction model fis a determinate dataset. The value is between 0 & 1.
- Look at the distribution of the residuals. Are they constant, or is there some systematic bias?

Cross-validatation

It uses multiple test-train splits. Each split is used to train the model and get an error metric. At the end, an overall average is computed. This way we are able to get a more stable and realistic performance of our model.

- k-Fold CV, divide the dataset in k folds, run k models
- Stratified CV, so as to allow class proportions to be preserved in the splits.
- Leave-one-out CV, run n models

Linear Regression vs KNN-Regression

Linear	KNN
few parameters	Non-Parametric
Small dataset	Large dataset needed
Generalizes beyond	Limited generalization

Polynomial feature expansion

Generate new features consisting of all polynomial combinations of the original two features. The degree of the polynomial specifies how many variables participate at a time in each new feature. it is still a linear model, and can use same least-squares estimation method

Why?

- To capture interactions between the original features by adding them as features to the linear model.
- To make a classifiercation problem easier

Pitfasll?

- Beware of polynomial feature expansion with high degree as this can lead to complex models that overfit

Generalization

Refers to an algorithm's ability to give accurate predictions for new, previously unseen data

- Models that are too complex for the amount of training data available are said to overfit and are not likely to generalize well to new examples.
- Models that are too simple, that don't even do well on the training data, are said to underfit and also not likely to generalize well.
- Refer to bias-variance trade off for an indepth analysis of why over/underfitting happens

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =train_test_split(X_R1, y_R1, random_state = 0)
linreg =LinearRegression().fit(X_train, y_train)
print("linear model intercept (b): {}".format(linreg.intercept_))
print("linear model coeff (w): {}".format(linreg.coef_))
```