Data Science

First Quarter Course Review

October 13th, 2014

Jarret Petrillo, jarret.petrillo@gmail.com

Using the Command Line w/ Git

Helpful Commands

cd ~/notebooks
changes current directory to the notebook folder in vagrant

cp ~/vagrant/file2movefromlocaldrive ~/notebooks
Moves a file from the local drive into the virtual environ

cd ~/notebooks/fall-2014-assignments
git remote add origin https://github.com/gads14-nyc/fall_2014_assignments.git
Bookmarks the typed git repo with the tag name "origin"
Note: different folders can use the same remote name

git pull origin master
Copies any changes into your local directory from origin repo

git add filetosubmit
git commit -m "Added filetosubmit"
git push origin master
Uploads added file to online repo

Linear Regression

Framework

- *Simple linear regression uncovers basic correlations*
- *Least squares estimation is sensitive to overfitting and heteroskedasticity*
- *Overfitting is the inclusion of extra covariates*
- *Heteroskedasticity is the presence of correlation between error terms and means that the iid (independently identically distributed) assumption of linear regression doesn't hold*
- *Ridge regression is an alternative linear estimation algorithm that works even in the presence of mild heteroskedasticity*
- *Lasso regression is another estimation algorithm that is robust to overfitting*

Helpful Functions

```
import statsmodels.api as sm
model = sm.ols(y, X)
results = model.fit()
results.summary()
```

```
from sklearn.linear_models import LinearRegression
linear_model = LinearRegression(fit_intercept=True)
linear_model.fit(X,y)
y_hat = linear_model.predict(X)
```

Covariate Selection Using Cross Validation

1-Fold CV: Pseudo Code

1. **Start with a list of potential models saved in a dictionary**
   models = {'model01': ['Infrared02'], 'model02':['ELEV','Infrared02']}
2. **Divide data set into test and train subsets**
3. **On the training subset fit each model**
4. **Save the mean squared error for each model in a dictionary**
5. **Sort the dictionary**
   results = {'model01': 0.553, 'model02': 0.434}
6. **Choose the model with the lowest mean squared error**

K-Fold CV: Pseudo Code

1. **Start with a list of potential models saved in a dictionary**
2. **for each k repeat steps 2, 3, and 4 above saving the results in a list in a dictionary**

   results = {'model01': [0.533, 0.513, 0.567], 'model02': [0.475, 0.469, 0.458]}
3. **Convert list of mean squared errors into a single value by taking the average**

   results = {'model01': 0.536, 'model02': 0.464}
4. **Sort the dictionary**
5. **Choose the model with the lowest average mean squared error**

Helpful Functions

from sklearn.cross_validation import KFold
Returns a tuple (train, test) of 0/1 vectors
data[train] returns training set

from sklearn.metrics import mean_squared_error
For two vector inputs returns the mean sqaured error

results = {'model01': 0.536, 'model02': 0.464}
sort(results, key=results.get, reverse=True)
returns a sorted list from a dictionary