Cross-Validation

Justin Post

Recap

- Judge the model's effectiveness at predicting using a metric comparing the predictions to the observed value
- Often split data into a training and test set
  - Perhaps 70/30 or 80/20
- Next: Cross-validation as an alternative to just train/test (and why we might do both!)

Issues with Trainging vs Test Sets

Why may we not want to just do a basic training/test split?

- If we don't have much data, we aren't using it all when fitting the models
- Data is randomly split into training/test
  - May just get a weird split by chance
  - Makes metric evaluation a somewhat variable measurement depending on number of data points

Issues with Trainging vs Test Sets

Why may we not want to just do a basic training/test split?

- If we don't have much data, we aren't using it all when fitting the models

- Data is randomly split into training/test

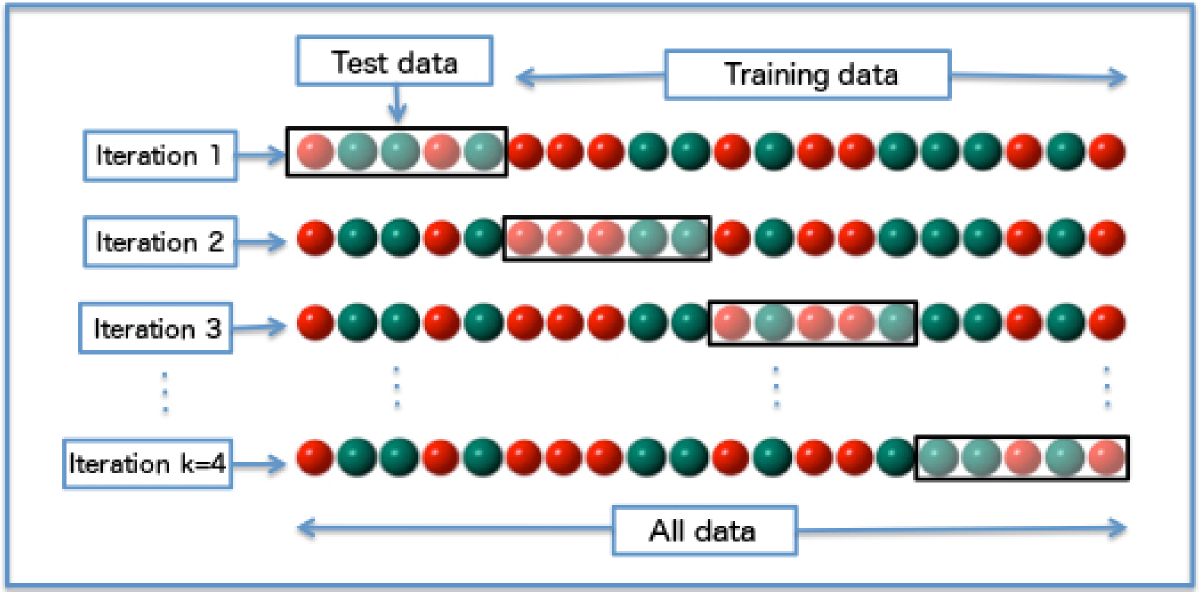
  - May just get a weird split by chance
  - Makes metric evaluation a somewhat variable measurement depending on number of data points

- Instead, we could consider splitting the data multiple ways, do the fitting/testing process, and combine the results!

  - Idea of cross validation!
  - A less variable measurement of your metric that uses all the data
  - Higher computational cost!

Cross-validation

Common method for assessing a predictive model



Cross-Validation Idea

k fold Cross-Validation (CV)

- Split data into k folds
- Train model on first k-1 folds, test on kth to find metric value
- Train model on first k-2 folds and kth fold, test on (k-1)st fold to find metric value
- ...

Cross-Validation Idea

k fold Cross-Validation (CV)

- Split data into k folds
- Train model on first k-1 folds, test on kth to find metric value
- Train model on first k-2 folds and kth fold, test on (k-1)st fold to find metric value
- …

Find CV error

- Combine test metrics across test folds
- For example, average all MSE metrics
- **Key = no predictions used in the value of the metric were found on data that were used to train that model!**

CV on MLR Models

- Let's consider our three linear regression models

$$\text{Model 1: log\_selling\_price} = \text{intercept} + \text{slope*year} + \text{Error}$$

$$\text{Model 2: log\_selling\_price} = \text{intercept} + \text{slope*log\_km\_driven} + \text{Error}$$

$$\text{Model 3: log\_selling\_price} = \text{intercept} + \text{slope*log\_km\_driven} + \text{slope*year} + \text{Error}$$

```
import pandas as pd
import numpy as np
bike_data = pd.read_csv("https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv")
bike_data['log_selling_price'] = np.log(bike_data['selling_price'])
bike_data['log_km_driven'] = np.log(bike_data['km_driven'])
```

CV on MLR Models

- Can use CV error to choose between these models
- In `scikit-learn` use the `cross_validate()` function from the `model_selection` submodule
    - Uses a `scoring` input to determine the metric

CV on MLR Models

- Can use CV error to choose between these models
- In `scikit-learn` use the `cross_validate()` function from the `model_selection` submodule
  - Uses a `scoring` input to determine the metric

```
from sklearn.model_selection import cross_validate
from sklearn import linear_model
reg1 = linear_model.LinearRegression()
cv1 = cross_validate(reg1,
    bike_data["year"].values.reshape(-1,1),
    bike_data["log_selling_price"].values,
    cv=5,
    scoring=('neg_mean_squared_error'),
    return_train_score=True)
print(cv1.keys())
```

```
## dict_keys(['fit_time', 'score_time', 'test_score', 'train_score'])
```

```
print(cv1)
```

```
## {'fit_time': array([0.00728607, 0.        , 0.        , 0.        , 0.        ]), 'score_time': array([0., 0., 0., 0., 0
```

CV on MLR Models

- Can use CV error to choose between these models
- In `scikit-learn` use the `cross_validate()` function from the `model_selection` submodule
  - Uses a `scoring` input to determine the metric

```
print(cv1["test_score"])
```

```
## [-0.33432825 -0.39699181 -0.22164746 -0.20264027 -0.38421905]
```

```
#get CV RMSE
round(np.sqrt(-sum(cv1["test_score"])/5),4)
```

```
## 0.5549
```

CV on MLR Models

- Fit our other models

```
reg2 = linear_model.LinearRegression()
cv2 = cross_validate(reg2,
    bike_data["log_km_driven"].values.reshape(-1,1),
    bike_data["log_selling_price"].values,
    cv=5, scoring='neg_mean_squared_error')
reg3 = linear_model.LinearRegression()
cv3 = cross_validate(reg3, bike_data[["year", "log_km_driven"]],
    bike_data["log_selling_price"].values,
    cv = 5, scoring='neg_mean_squared_error')
```

CV on MLR Models

- Compare the MSE values

```
print(round(np.sqrt(-sum(cv1["test_score"])/5),4),
      round(np.sqrt(-sum(cv2["test_score"])/5),4),
      round(np.sqrt(-sum(cv3["test_score"])/5),4))
```

```
## 0.5549 0.6021 0.518
```

- Now we would refit the 'best' model on the full data set!

Recap

Cross-validation gives a way to use more of the data while still seeing how the model does on test data

- Commonly 5 fold or 10 fold is done
- Once a best model is chosen, model is refit on entire data set
- **We'll see how CV can be used to select tuning parameters for certain models**
    - In this case, we often use both CV and a train/test split together!