LASSO Models

Justin Post

Recap

- Judge the model's effectiveness using a **Loss** function
- Often split data into a training and test set
    - Perhaps 70/30 or 80/20
- 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

Recap

- Judge the model's effectiveness using a **Loss** function
- Often split data into a training and test set
  - Perhaps 70/30 or 80/20
- 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
- Often use both! Let's see why by introducing a model with a **tuning parameter**

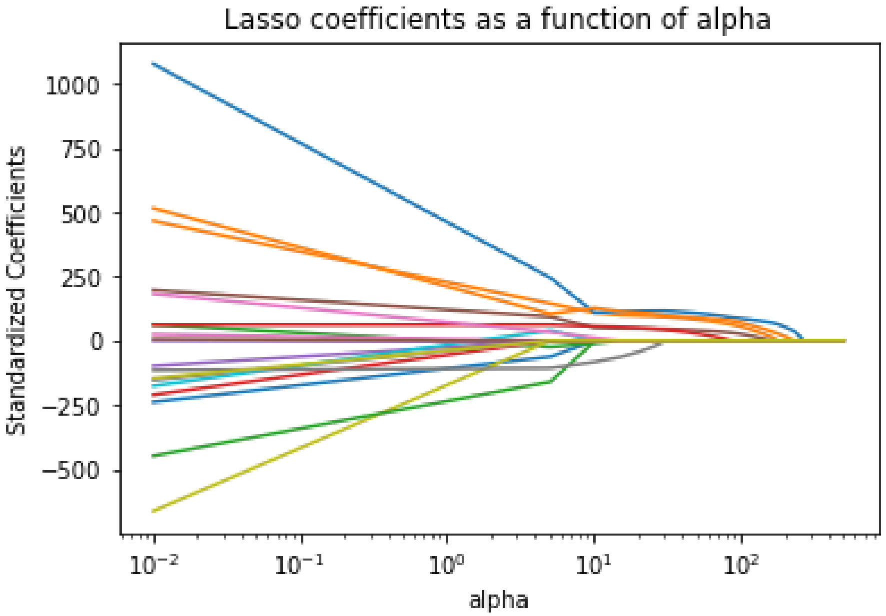
LASSO Model

- Least Angle Subset and Selection Operator or LASSO
  - Similar to Least Squares but a penalty is placed on the sum of the absolute values of the regression coefficients
  - α (>0) is called a tuning parameter

$$\min_{\beta's} \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi}))^2 + \alpha \sum_{j=1}^{p} |\beta_j|$$

LASSO Model

- Least Angle Subset and Selection Operator or LASSO
    - Similar to Least Squares but a penalty is placed on the sum of the absolute values of the regression coefficients
    - Sets coefficients to 0 as you 'shrink'!



Tuning Parameter

- When choosing the tuning parameter, we are really considering a **family of models**!
- Consider an $\alpha = 0.1$ (small amount of shrinkage here)

```
from sklearn import linear_model
lasso = linear_model.Lasso(alpha=0.1)
lasso.fit(bike_data[["year", "log_km_driven"]].values, bike_data["log_selling_price"].values)
```

```
print(lasso.intercept_,lasso.coef_)
```

```
## -164.61209472866094 [ 0.08761607 -0.11092474]
```

Tuning Parameter

- When choosing the tuning parameter, we are really considering a **family of models**!
- Consider an $\alpha = 0.1$ (small amount of shrinkage here)

```
from sklearn import linear_model
lasso = linear_model.Lasso(alpha=0.1)
lasso.fit(bike_data[["year", "log_km_driven"]].values, bike_data["log_selling_price"].values)
```

```
print(lasso.intercept_,lasso.coef_)
```

```
## -164.61209472866094 [ 0.08761607 -0.11092474]
```

- Consider an $\alpha = 1.05$ (a larger amount of shrinkage)

```
lasso = linear_model.Lasso(alpha=1.05)
lasso.fit(bike_data[["year", "log_km_driven"]].values, bike_data["log_selling_price"].values)
```

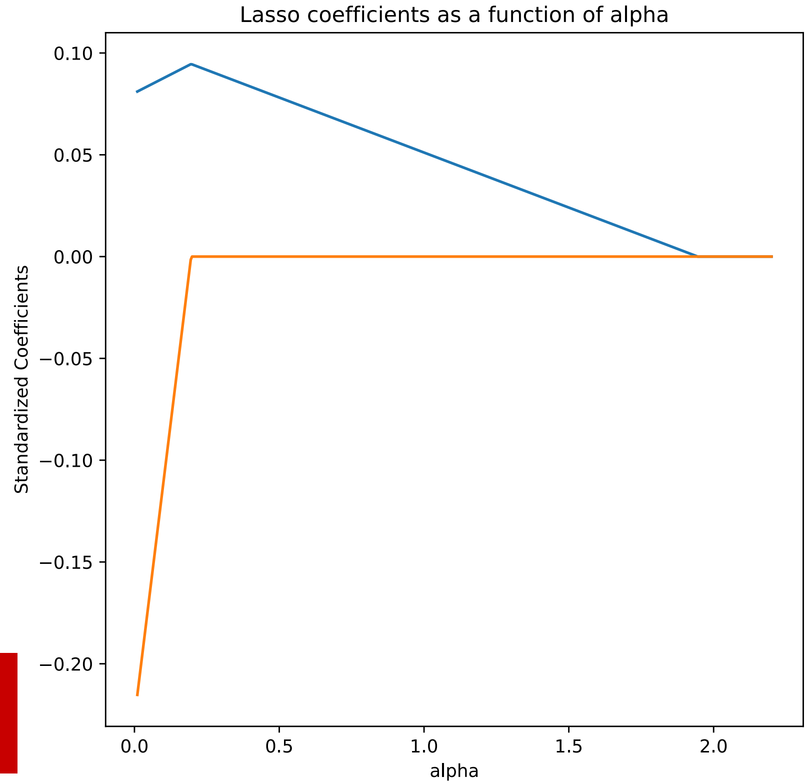
```
print(lasso.intercept_,lasso.coef_)
```

```
## -86.65630892150762 [ 0.04835598 -0.         ]
```

LASSO Fits Visual

- Perfect place for CV to help select the best α!

```
## (-0.09950000000000003, 2.3095000000000003, -0.23074900910594773, 0.1099672686377952)
```



NC STATE UNIVERSITY

Using CV to Select the Tuning Parameter

- Return the optimal α using `LassoCV`

```
from sklearn.linear_model import LassoCV
lasso_mod = LassoCV(cv=5, random_state=0, alphas = np.linspace(0,2.2,100)) \
    .fit(bike_data[["year", "log_km_driven"]].values,
          bike_data["log_selling_price"].values)
```

Using CV to Select the Tuning Parameter

- Return the optimal α using `LassoCV`

```
pd.DataFrame(zip(lasso_mod.alphas, lasso_mod.mse_path_), columns = ["alpha_value", "MSE_by_fold"])
```

```
##     alpha_value                                        MSE_by_fold
## 0      0.000000  [0.5496710875578059, 0.6805679103740427, 0.500...
## 1      0.022222  [0.5496710875578059, 0.6805679103740427, 0.500...
## 2      0.044444  [0.5496710875578059, 0.6805679103740427, 0.500...
## 3      0.066667  [0.5496710875578059, 0.6805679103740427, 0.500...
## 4      0.088889  [0.5496710875578059, 0.6805679103740427, 0.500...
## ..          ...                                                ...
## 95     2.111111  [0.3046546135682859, 0.3626276356362568, 0.191...
## 96     2.133333  [0.2998496943347488, 0.35411026477928204, 0.18...
## 97     2.155556  [0.29626758731408676, 0.34645956641174064, 0.1...
## 98     2.177778  [0.2939082925063055, 0.3396755405336282, 0.182...
## 99     2.200000  [0.29277214247542543, 0.33375857421410476, 0.1...
##
## [100 rows x 2 columns]
```

Using CV to Select the Tuning Parameter

- Return the optimal α using `LassoCV`

```
fit_info = np.array(list(zip(lasso_mod.alphas_, np.mean(lasso_mod.mse_path_, axis = 1))))
fit_info[fit_info[:,0].argsort()]
```

```
## array([[0.        , 0.26832555],
##        [0.02222222, 0.26904464],
##        [0.04444444, 0.27086204],
##        [0.06666667, 0.27377741],
##        [0.08888889, 0.27779157],
##        [0.11111111, 0.28290414],
##        [0.13333333, 0.28911508],
##        [0.15555556, 0.29642441],
##        [0.17777778, 0.30483239],
##        [0.2       , 0.30967893],
##        [0.22222222, 0.31098715],
##        [0.24444444, 0.31159763],
##        [0.26666667, 0.31226415],
##        [0.28888889, 0.31298674],
##        [0.31111111, 0.31376537],
##        [0.33333333, 0.31460007],
##        [0.35555556, 0.31549081],
##        [0.37777778, 0.31643761],
##        [0.4       , 0.31744046],
                                ],
                                ],
                                ],
##        [0.48888889, 0.32201242]
```

NC STATE UNIVERSITY

Using CV to Select the Tuning Parameter

- Now fit using that optimal α

```
lasso_best = linear_model.Lasso(lasso_mod.alpha_) #warning thrown since we are using 0, but can ignore
lasso_best.fit(bike_data[["year", "log_km_driven"]].values, bike_data["log_selling_price"].values)
```

```
print(lasso_best.intercept_,lasso_best.coef_)
```

```
## -148.7932910778814 [ 0.0803366  -0.22686129]
```

So Do We Just Need CV?

Sometimes!

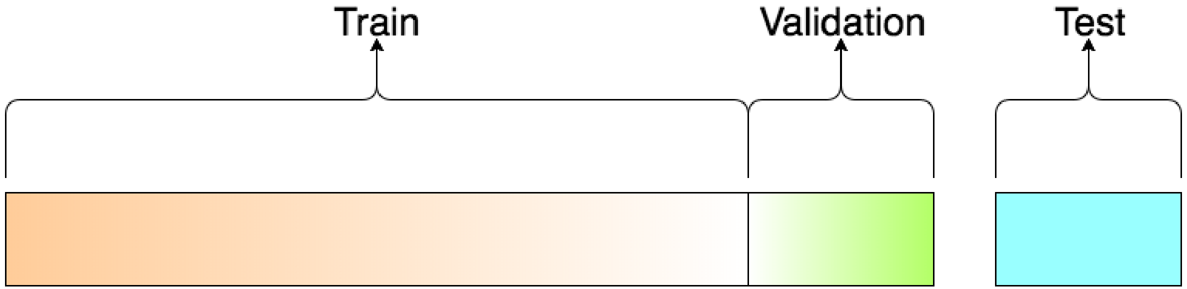
- If you are only considering one type of model, you can use just a training/test set or k-fold CV to select the best version of that model

- When you have multiple types of models to choose from, usually use both!

    - When we use the test set too much, we may have '**data leakage**'
    - Essentially we end up training our models to the test set by using it too much

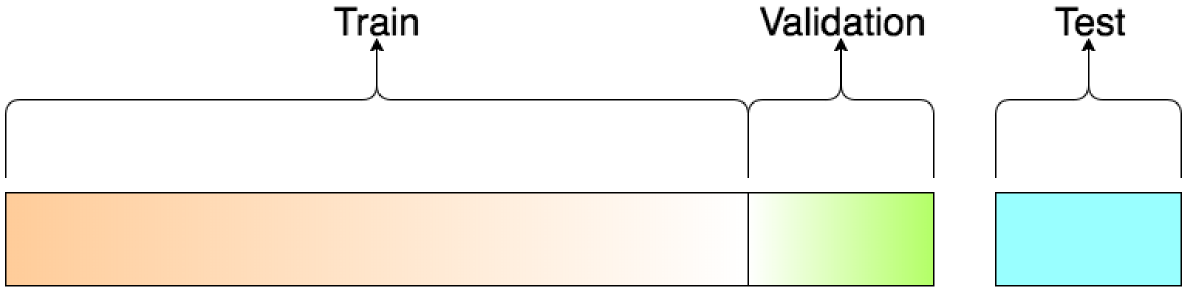
Training/Validation/Test or CV/Test

- Instead, we sometimes split into a training, validation, and test set
- CV can be used to replace the validation set!



Training/Validation/Test or CV/Test

- Instead, we sometimes split into a training, validation, and test set
- CV can be used to replace the validation set!



- Compare only the **best** model from each model type on the test set

Recap

- LASSO is similar to an MLR model but shrinks coefficients and may set some to 0
    - Tuning parameter must be chosen (usually by CV)
- Training/Test split gives us a way to validate our model's performance
    - CV can be used on the training set to select **tuning parameters**
    - Helps determine the 'best' model for a class of models
- With many competing model types, determine the best from each type check performance on the test set