MATH 637: Mathematical Techniques in Data Science
Model selection

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

March 02, 2020

Comparison of regression methods seen so far

1. Ordinary least squares (OLS)
    - Minimizes sum of squares.
    - Best linear unbiased estimator.
    - Solution not unique when $n < p$.
    - Estimate unstable when the predictors are collinear.
    - Generally does not lead to best prediction error. Bias-variance trade-off.

Comparison of regression methods seen so far

1. Ordinary least squares (OLS)
    - Minimizes sum of squares.
    - Best linear unbiased estimator.
    - Solution not unique when $n < p$.
    - Estimate unstable when the predictors are collinear.
    - Generally does not lead to best prediction error. Bias-variance trade-off.
2. Ridge regression (ℓ_2 penalty)
    - Regularized solution.
    - Estimator exists and is stable, even when $n < p$.
    - Easy to compute (add multiple of identity to $X^T X$).
    - Coefficients not set to zero (no model selection).

Comparison of regression methods seen so far (cont.)

3. Subset selection methods (best subset, stepwise and stagewise approaches)
    - Generally leads to a favorable bias-variance trade-off.
    - Model selection. Leads to models that are easier to interpret and work with.
    - Can be computationally intensive (e.g. best subset can only be computed for small p)
    - Some of the approaches are greedy/less-rigorous.

Comparison of regression methods seen so far (cont.)

3. Subset selection methods (best subset, stepwise and stagewise approaches)
   - Generally leads to a favorable bias-variance trade-off.
   - Model selection. Leads to models that are easier to interpret and work with.
   - Can be computationally intensive (e.g. best subset can only be computed for small p)
   - Some of the approaches are greedy/less-rigorous.
4. Lasso (ℓ_1 penalty)
   - Shrinks and sets to zero the coefficients (shrinkage + model selection).
   - Generally leads to a favorable bias-variance trade-off.
   - Model selection. Leads to models that are easier to interpret and work with.
   - Can be efficiently computed.
   - Supporting theory. Active area of research.

Choosing parameters: cross-validation

- Ridge, lasso, elastic net have regularization parameters.

Choosing parameters: cross-validation

- Ridge, lasso, elastic net have regularization parameters.
- We obtain a family of estimators as we vary the parameter(s).

Choosing parameters: cross-validation

- Ridge, lasso, elastic net have regularization parameters.
- We obtain a family of estimators as we vary the parameter(s).
- An *optimal* parameter needs to be chosen in a principled way.

Choosing parameters: cross-validation

- Ridge, lasso, elastic net have regularization parameters.
- We obtain a family of estimators as we vary the parameter(s).
- An *optimal* parameter needs to be chosen in a principled way.
- **Cross-validation** is a popular approach for rigorously choosing parameters.

Choosing parameters: cross-validation

- Ridge, lasso, elastic net have regularization parameters.
- We obtain a family of estimators as we vary the parameter(s).
- An *optimal* parameter needs to be chosen in a principled way.
- **Cross-validation** is a popular approach for rigorously choosing parameters.

K-**fold cross-validation:**

```
Split data into $K$ equal (or almost equal) parts/folds at random.
for each parameter $\lambda_i$ do
    for $j = 1, \ldots, K$ do
        Fit model on data with fold $j$ removed.
        Test model on remaining fold $\to$ $j$-th test error.
    end for
    Compute average test errors for parameter $\lambda_i$.
end for
Pick parameter with smallest average error.
```

K-fold CV

More precisely,

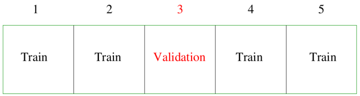
- Split data into K folds $F_1, \ldots, F_K$.

1	2	3	4	5
Train	Train	Validation	Train	Train

K-fold CV

More precisely,

- Split data into K folds $F_1, \dots, F_K$.

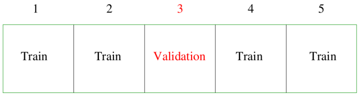


- Let $L(y, \hat{y})$ be a *loss function*. For example, $L(y, \hat{y}) = \|y - \hat{y}\|_2^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$.

K-fold CV

More precisely,

- Split data into K folds $F_1, \dots, F_K$.

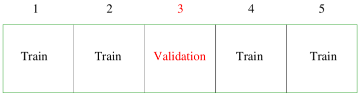


- Let $L(y, \hat{y})$ be a *loss function*. For example, $L(y, \hat{y}) = \|y - \hat{y}\|_2^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$.
- Let $f_\lambda^{-k}(\mathbf{x})$ be the model fitted on all, but the k-th fold.

K-fold CV

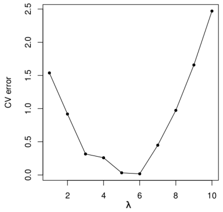
More precisely,

- Split data into K folds $F_1, \dots, F_K$.



- Let $L(y, \hat{y})$ be a *loss function*. For example, $L(y, \hat{y}) = \|y - \hat{y}\|_2^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- Let $f_\lambda^{-k}(\mathbf{x})$ be the model fitted on all, but the k-th fold.
- Let

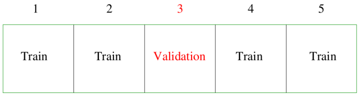
$$CV(\lambda) := \frac{1}{n}\sum_{k=1}^{n}\sum_{i \in F_k} L(y_i, f_\lambda^{-i}(\mathbf{x}_i))$$



K-fold CV

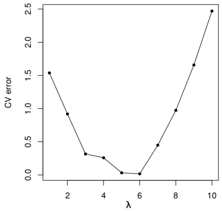
More precisely,

- Split data into K folds $F_1, \dots, F_K$.



- Let $L(y, \hat{y})$ be a *loss function*. For example, $L(y, \hat{y}) = \|y - \hat{y}\|_2^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$.
- Let $f_\lambda^{-k}(\mathbf{x})$ be the model fitted on all, but the k-th fold.
- Let

$$CV(\lambda) := \frac{1}{n} \sum_{k=1}^{n} \sum_{i \in F_k} L(y_i, f_\lambda^{-i}(\mathbf{x}_i))$$



- Pick λ among a *relevant* set of parameters

$$\hat{\lambda} = \operatorname*{argmin}_{\lambda \in \{\lambda_1, \dots, \lambda_m\}} CV(\lambda)$$

Python

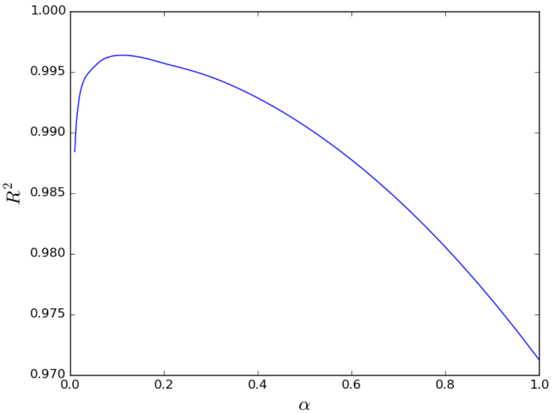
Scikit-learn has nice general methods for splitting data.

```
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.linear_model import Lasso

# Generate random data
n = 100
p = 5

X = np.random.randn(n,p)
epsilon = np.random.randn(n,1)
beta = np.random.rand(p)
y = X.dot(beta) + epsilon

# Train-test split
X_train, X_test, y_train, y_test =
  train_test_split(X, y, test_size=0.25)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

# K-fold CV
from sklearn.model_selection import KFold
kf = KFold(n_splits=10)
for train, test in kf.split(X):
        print("Train %s \n Test %s" % (train, test))
```

Python: Implementing CV

```
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.model_selection import KFold

# Generate random data
n = 100
p = 100

X = np.random.randn(n,p)
epsilon = np.random.randn(n,1)
beta = np.zeros((p,1))
beta[0:8] = 10*np.random.rand(8,1)
y = X.dot(beta) + epsilon

K = 10  # K-fold CV
alphas = np.exp(np.linspace(np.log(0.01),np.log(1),100))
N = len(alphas) # Number of lasso parameters
scores = np.zeros((N,K))
kf = KFold(n_splits=10)

for i in range(N):
   clf = Lasso(alphas[i])
   for j, (train, test) in enumerate(kf.split(X)):
      X_train, X_test, y_train, y_test =
        X[train], X[test], y[train], y[test]
      clf.fit(X_train,y_train)
      scores[i,j] = clf.score(X_test, y_test)   # Returns R^2
# Compute average CV score for each parameter
scores_avg = scores.mean(axis=1)
```

Implementing CV



Note: Here we want to choose α to *maximize* the R^2.

Exercise: Implement 10-fold CV for Ridge regression. Plot CV error.

LassoCV

Scikit-learn sometimes has automatic methods for performing cross-validation.

```
import numpy as np
from sklearn.linear_model import LassoCV
import matplotlib.pyplot as plt

# Generate random data
n = 100
p = 100

X = np.random.randn(n,p)
epsilon = np.random.randn(n,1)
beta = np.zeros((p,1))
beta[0:8] = 10*np.random.rand(8,1)
y = X.dot(beta) + epsilon

K = 10  # K-fold CV

y = y.reshape(n) # LassoCV doesn't work if y is (n x 1)
clf = LassoCV(n_alphas = 100, cv = K)

clf.fit(X,y)
```

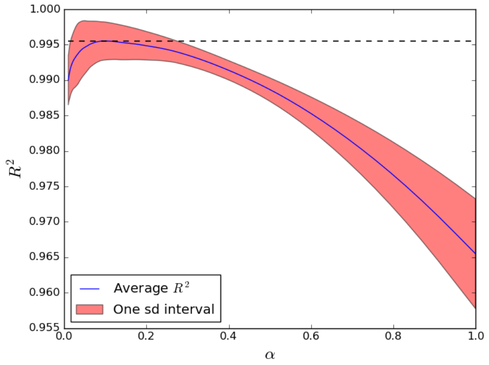
Remark: safer to examine CV curve.

One SD rule

For each parameter, one can also naturally report the standard deviation of the error acroos the different folds.

```
# Compute average CV score for each parameter
scores_avg = scores.mean(axis=1)
scores_std = scores.std(axis=1)

plt.plot(alphas, scores_avg,'-b')
plt.fill_between(alphas, scores_avg-scores_std, s
  cores_avg+scores_std,facecolor='r',alpha=0.5)

plt.legend([r'Average $R^2$', r'One sd interval'],
  loc = 'lower left')

plt.plot(alphas, np.ones((len(alphas),1))*scores_avg.max(),
  '--k', linewidth=1.2)

plt.xlabel(r'$\alpha$', fontsize=18)
plt.ylabel(r'$R^2$', fontsize = 18)
plt.show()
```

One sd rule (cont.)



- Provides an idea of the error made when estimating the R^2.
- Can pick a lasso parameter for which the maximum R^2 is within a one standard deviation interval of the actual value.
- Useful technique to select a model that is more sparse in a principled way (when necessary).

Model selection vs Model assessment

Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the “best” one.

Model selection vs Model assessment

Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the "best" one.
- **Model assessment:** having chosen a final model, estimating its prediction error (generalization error) on new data.

Model selection vs Model assessment

Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the "best" one.
- **Model assessment:** having chosen a final model, estimating its prediction error (generalization error) on new data.

Model assessment: is the estimator really good? compare different models with their own sets of parameters.

Model selection vs Model assessment

Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the "best" one.
- **Model assessment:** having chosen a final model, estimating its prediction error (generalization error) on new data.

Model assessment: is the estimator really good? compare different models with their own sets of parameters.

Generally speaking, the CV error provides a good estimate of the prediction error.

Model selection vs Model assessment

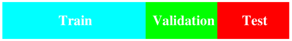
Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the "best" one.
- **Model assessment:** having chosen a final model, estimating its prediction error (generalization error) on new data.

Model assessment: is the estimator really good? compare different models with their own sets of parameters.

Generally speaking, the CV error provides a good estimate of the prediction error.

- When *enough* data is available, it is better to separate the data into three parts: train/validate, and test.



Model selection vs Model assessment

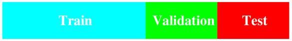
Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the "best" one.
- **Model assessment:** having chosen a final model, estimating its prediction error (generalization error) on new data.

Model assessment: is the estimator really good? compare different models with their own sets of parameters.

Generally speaking, the CV error provides a good estimate of the prediction error.

- When *enough* data is available, it is better to separate the data into three parts: train/validate, and test.



- Typically: 50% train, 25% validate, 25% test.

Model selection vs Model assessment

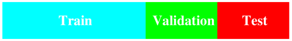
Two related, but different goals:

- **Model selection:** estimating the performance of different models in order to choose the "best" one.
- **Model assessment:** having chosen a final model, estimating its prediction error (generalization error) on new data.

Model assessment: is the estimator really good? compare different models with their own sets of parameters.

Generally speaking, the CV error provides a good estimate of the prediction error.

- When *enough* data is available, it is better to separate the data into three parts: train/validate, and test.



- Typically: 50% train, 25% validate, 25% test.
- Test data is "kept in a vault", i.e., not used for fitting or choosing the model.
- Other methods (e.g. AIC, BIC, etc.) can be used when working with very little data.