

Lecture 5


LASSO Regularization and Feature Selection

EE-UY 4563/EL-GY 9123: INTRODUCTION TO MACHINE LEARNING
PROF. SUNDEEP RANGAN (WITH MODIFICATION BY YAO WANG)

Learning Objectives

- ☐ Formulate a linear estimation problem with a regularization
- ☐ Compute an L1-regularized estimate (LASSO) using sklearn tools
- ☐ Compute the optimal regularization level using cross validation
- ☐ Interpret results from a LASSO path
- ☐ Determine final regression function from cross validation
- ☐ Set regularizer based on a probabilistic prior
- ☐ Perform other feature selection methods

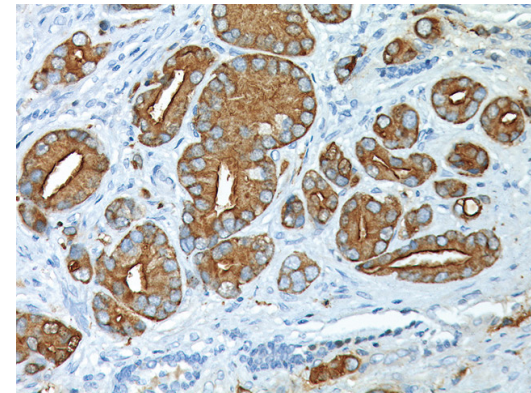
Outline

 Motivating Example: Predicting prostate cancer from a PSA test

- ☐ Model Selection
- ☐ Model Selection from LASSO regularization
- ☐ Probabilistic interpretation
- ☐ Other Model Selection Methods
- ☐ In-Class Exercise: Audio Pitch Detection

Prostate Specific Antigen Testing

- ❑ PSA levels easily tested
- ❑ High PSA believed to be associated with prostate cancer
 - Potential tool for screening
- ❑ Classic 1989 study by Thomas et al:
 - Measured PSA level of 102 men prior to prostate removal
 - Measured characteristics of prostate from samples
 - Characteristics include cancer volume, weight, ...
- ❑ Data analysis:
 - What characteristics predict PSA?



Stamey, Thomas A., et al. "[Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients.](#)" The Journal of urology 141.5 (1989): 1076-1083.

Data

- ❑ Prostate dataset widely-used in ML classes
- ❑ Can be downloaded from many sites
- ❑ Samples = 97 patients
- ❑ 8 features of the prostate
- ❑ Target variable = lpsa (log PSA)

```
# Get data
url = 'https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data'
df = pd.read_csv(url, sep='\t', header=0)
df = df.drop('Unnamed: 0', axis=1) # skip the column of indices
```

The data frame has the following components:

```
lcavol      log(cancer volume)
lweight     log(prostate weight)
age         age
lbph        log(benign prostatic hyperplasia amount)
svi         seminal vesicle invasion
lcp         log(capsular penetration)
gleason     Gleason score
pgg45       percentage Gleason scores 4 or 5
lpsa        log(prostate specific antigen)
```

First Try: Linear Model

□ Simple idea: Use linear regression

$$y \approx \hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d$$

- y = lpsa (target PSA level)
- x_1, \dots, x_d = prostate features ($d = 8$)

□ Why linear regression?

- Easy to compute / interpret
- Coefficients are easy to interpret
- Larger coefficients \Rightarrow larger influence of feature on PSA

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

```
X_tr, X_ts, y_tr, y_ts = train_test_split(X,y,test_size=0.5,shuffle=True)
ntr = X_tr.shape[0]
nts = X_ts.shape[0]
print("num samples train = %d, test = %d" % (ntr, nts))
```

```
num samples train = 48, test = 49
```

```
regr = linear_model.LinearRegression()
regr.fit(X_tr,y_tr)
```

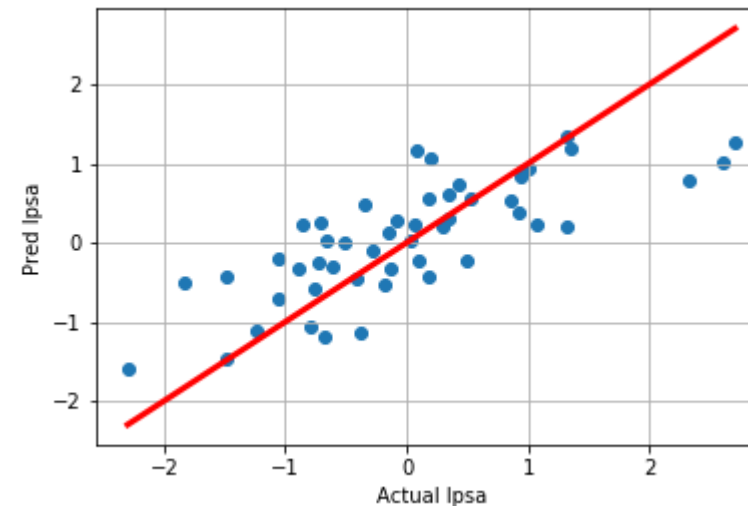
Model Fit

- ❑ Evaluate model with cross validation
 - Train on 48 samples
 - Measure RSS on 49 samples
- ❑ We obtain reasonable fit on test data
 - $R^2 \approx 0.58$

```
y_ts_pred = regr.predict(X_ts)
rss_ts = np.mean((y_ts_pred-y_ts)**2)/(np.std(y_ts)**2)
rsq_ts = 1-rss_ts
print("Normalized test RSS = %f" % rss_ts)
print("Normalized test R^2 = %f" % rsq_ts)
```

Normalized test RSS = 0.419799

Normalized test R^2 = 0.580201



Looking at the Coefficients

□ Recall that model is: $\hat{y} = b + w_1x_1 + \dots + w_dx_d$

□ Weights w_i indicates dependence of feature i on target y

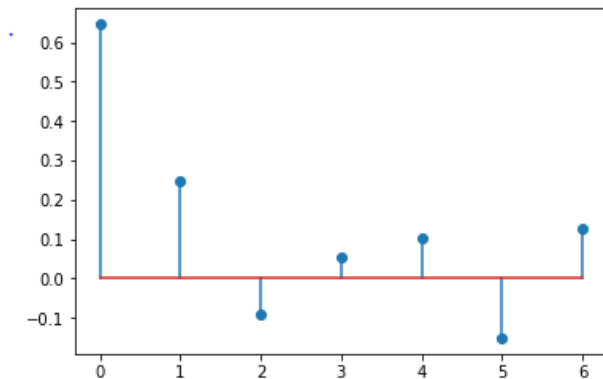
□ For PSA test:

- Highest weight on lcavol (log cancer volume)
- But, weights on all features are non-zero
- Hard to eliminate features

□ How can we tell if some features are not significant?


```
w = regr.coef_  
for name, wi in zip(names_x, w):  
    print('%10s %9.4f' % (name, wi))
```

lcavol	0.6457
lweight	0.2466
age	-0.0895
lbph	0.0543
svi	0.1034
lcp	-0.1508
gleason	0.1253



Outline

☐ Motivating Example: Predicting prostate cancer from a PSA test

 ☐ Model Selection


☐ Model Selection from LASSO regularization

☐ Probabilistic interpretation

☐ Other Model Selection Methods

☐ In-Class Exercise: Audio Pitch Detection

Model Selection

- Consider linear model: $y \approx \hat{y} = b + w_1x_1 + \cdots + w_dx_d$ 
- Models target y as function of features $\mathbf{x} = (x_1, \dots, x_d)$
- In many problems, we know only a few features are likely relevant
- This means we want $w_j = 0$ for most features
- But, we don't know a priori which features are relevant
- Model selection problem: Fit a model with a small number of features
- Mathematically:
 - Determine a subset of features $I \subseteq \{1, \dots, d\}$ with $|I|$ small
 - Fit a model: $\hat{y} = b + w_1x_1 + \cdots + w_dx_d$ with $w_j = 0$ for all $j \notin I$

Example 1: Medical Modeling

- ❑ PSA test:
 - We have a number of features
- ❑ But, likely that only a small number of features are relevant
- ❑ Want a method that can determine which ones matter

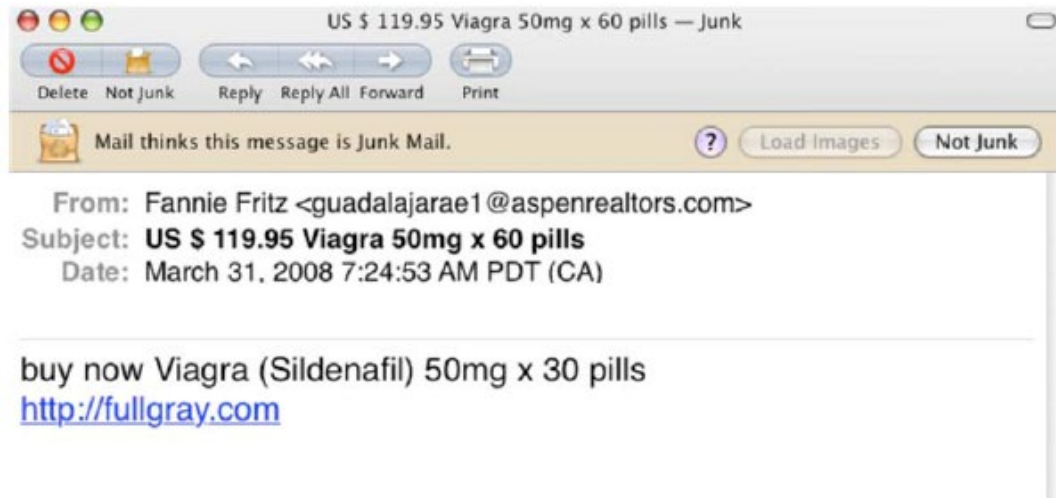
The data frame has the following components:

```
lcavol      log(cancer volume)
lweight     log(prostate weight)
age         age
lbph        log(benign prostatic hyperplasia amount)
svi         seminal vesicle invasion
lcp         log(capsular penetration)
gleason     Gleason score
pgg45       percentage Gleason scores 4 or 5
lpsa        log(prostate specific antigen)
```

Model Selection with Limited Data

- ❑ Model selection is particularly valuable when data is limited
- ❑ Ex: Consider linear model: $\hat{y} = b + w_1x_1 + \dots + w_dx_d$
 - Model has $d + 1$ parameters
- ❑ From previous lecture, we need $N > d + 1$ data points (x_i, y_i)
- ❑ In many cases we have $N \ll d$
 - Examples below
 - Many few data points than features
 - Classic linear fit will not work
- ❑ But, suppose we can restrict to $K \ll N$ non-zero parameters
 - Then, we can find a good fit on those parameters
- ❑ Challenge: How do we find a small number K of relevant features

Example 2: Spam Detection



❑ Classification problem:

- Is email junk or not junk?

❑ Typical bag-of-words model:

- Enumerate all words, $i = 1, \dots, d$
- Represent email via word count $x_i = \text{num instances of word } i$

❑ Model selection:

- d = vocabulary size is typically very large
- But, only a few words are likely relevant
- Want to find $K \ll d$ relevant words

Example 3: EEG

- ❑ EEG: Electroencephalography
- ❑ Measure brain activity from electrodes on scalp
- ❑ Source localization problem:
 - Find brain region responsible for evoked response
- ❑ Problem:
 - Many possible brain regions
Typically use $d > 10,000$ voxels
 - But, limited number of measurements:
100s of electrodes
 - Cannot fit a model from all brain regions
- ❑ Model selection:
 - We know that responses are likely from a small brain region
 - Find a small number of voxels that explain response

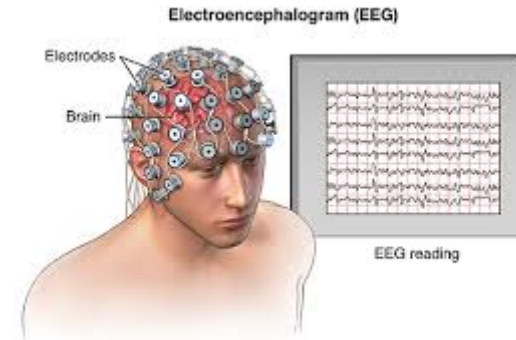
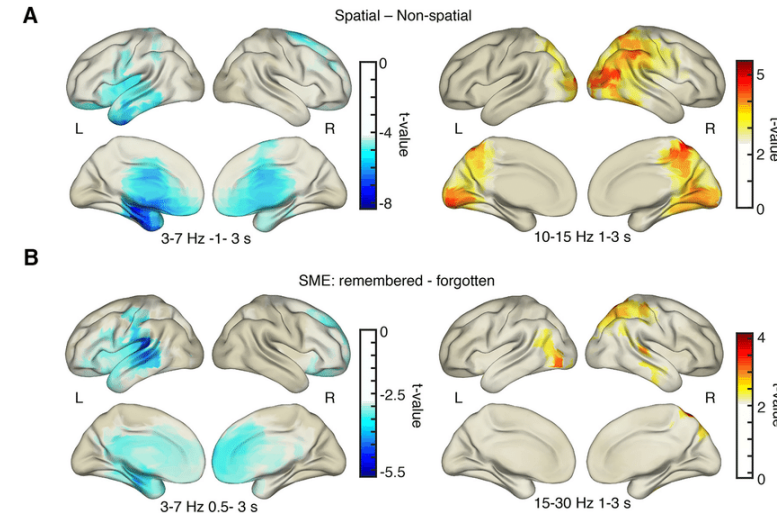
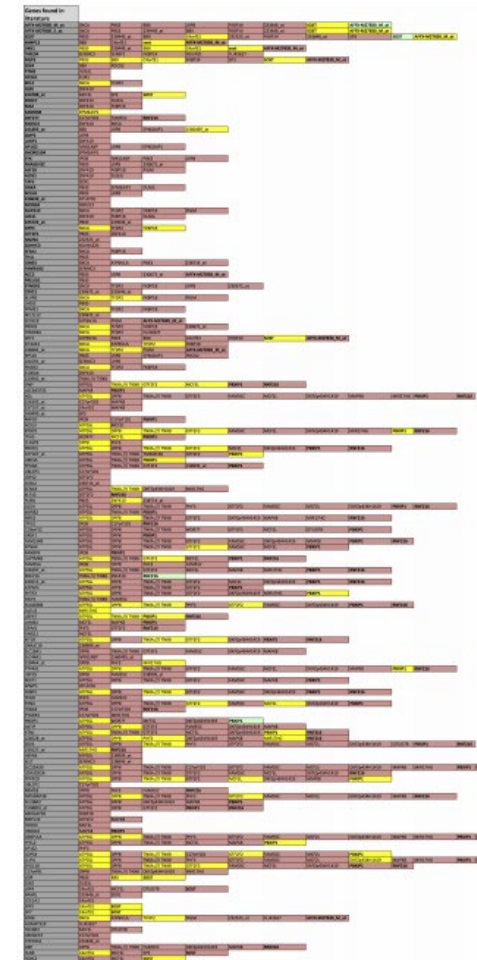


Image: mayoclinic.org




Example 4: DNA MicroArray Data

- ❑ Basic genetic problem
 - Which genes determine some characteristic (i.e. phenotype)?
- ❑ DNA microarrays:
 - Measure “expression” levels of large numbers of genes
 - Expression levels = amount of protein produced by gene
- ❑ Data modeling:
 - Fit phenotype to expression levels
 - Usually have large numbers of genes ($d \sim 1000$)
 - But, small number of data points ($n \sim 100$)
 - We know only a small number of genes are responsible
 - So, we can use model selection



Outline

- ❑ Motivating Example: Predicting prostate cancer from a PSA test
- ❑ Model Selection
- ❑ Model Selection from LASSO regularization
- ❑ Probabilistic interpretation
- ❑ Other Model Selection Methods
- ❑ In-Class Exercise: Audio Pitch Detection

Intuition

- ❑ We know from last lecture:
 - Too many parameters \Rightarrow Large generalization error
- ❑ In this data set, only a few factors are likely significant
- ❑ But, we don't know which one
- ❑ Can we automatically identify them?
 - Use correlation between features and target
 - Do not always work well
 - Exhaustive search can be expansive!
- ❑ Idea: Fit model under constraint:
 - Force only a few parameters to be non-zero
- ❑ General idea of **regularization**:
 - Constrain the parameters with prior knowledge

The data frame has the following components:

```
lcavol      log(cancer volume)
lweight     log(prostate weight)
age         age
lbph        log(benign prostatic hyperplasia amount)
svi         seminal vesicle invasion
lcp         log(capsular penetration)
gleason     Gleason score
pgg45       percentage Gleason scores 4 or 5
lpsa        log(prostate specific antigen)
```

Regularized LS Estimation

□ Standard least squares estimation (from Lecture 3):

$$\hat{\beta} = \arg \min_{\beta} RSS(\beta), \quad RSS(\beta) = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

□ Regularized estimator:

$$\hat{\beta} = \arg \min_{\beta} J(\beta), \quad J(\beta) = RSS(\beta) + \phi(\beta)$$

- $RSS(\beta)$ = prediction error from before
- $\phi(\beta)$ = regularizing function.

□ Concept: Regularizer penalizes β that are “unlikely”

- Constrains estimate to smaller set of parameters

Two Common Regularizers

❑ Ridge regression (called L2)

$$\phi(\beta) = \alpha \sum_{j=1}^d |\beta_j|^2$$

❑ LASSO regression (called L1)

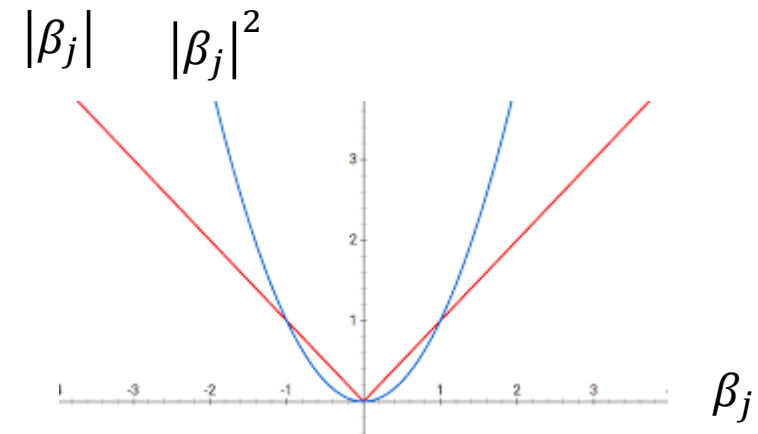
$$\phi(\beta) = \alpha \sum_{j=1}^d |\beta_j|$$

❑ Both penalize large β_j

❑ Level of regularization controlled by α

❑ Note the regularization sum

- Does not include the intercept β_0 ,
- This term depends on the mean of the target
- Should not be arbitrarily constrained to be small



Minimize $|\beta_j|^2$ do not penalize small non-zero coef., overly penalize large coef.

Minimize $|\beta_j|$ tend to make coefficients either 0 or large (SPARSE!)

Data Scaling

□ Scaling:

- Scale each feature and the target to have zero mean and unit variance (or STD)
- $x_{i,j} \rightarrow (x_{i,j} - E(x_{i,j}))/STD(x_{i,j})$
- $y_i \rightarrow (y_i - E(y_i))/STD(y_i)$

□ After predictor for the scaled data are determined:

- Derive the equivalent predictor on the original data (HW!)

□ Motivation:

- Without scaling, the regularization level depends on the data range
- With mean removal, we do not need the intercept term β_0
- So that the regularization term is simply a L2 or L1 norm of coefficient vector

L1 and L2 Norm

□ Assuming the data have been scaled to have zero mean and unit variance

□ Ridge cost function:

$$J(\boldsymbol{\beta}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^d |\beta_j|^2 = \|\mathbf{y} - A\boldsymbol{\beta}\|^2 + \alpha \|\boldsymbol{\beta}\|^2$$

□ LASSO cost function:

$$J(\boldsymbol{\beta}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^d |\beta_j| = \|\mathbf{y} - A\boldsymbol{\beta}\|^2 + \alpha \|\boldsymbol{\beta}\|_1$$

◦ $\|\boldsymbol{\beta}\|_1$ = L1 norm (pronounced ell-1)

Ridge vs LASSO

□ Optimization can be easily performed for L1 and L2 regularizers

- Regularizer is convex
- More on this later

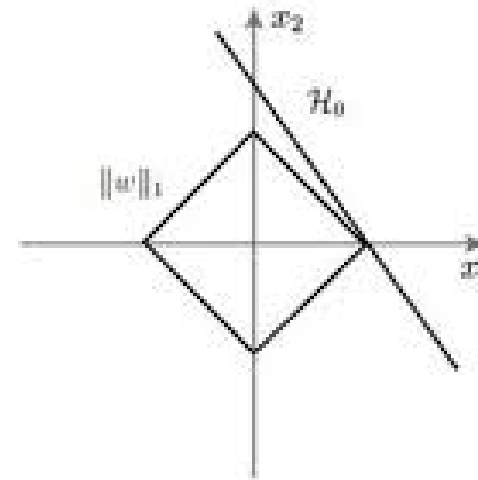
□ L2 tends to lead to many “small” coefficients

- Not great for feature selection
- Closed-form solution possible

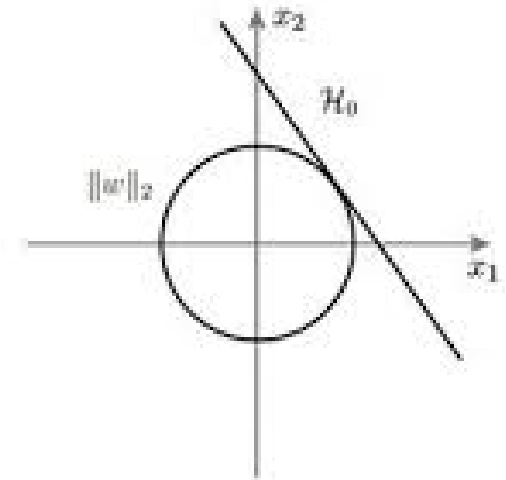
□ L1 tends to lead to more sparse solutions

- Several coefficients are zero
- No closed-form solution
- Will focus this lecture on L1

A L1 regularization



B L2 regularization



Ridge Regression

❑ Loss function

$$J(\beta) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^d |\beta_j|^2 = \|y - A\beta\|^2 + \alpha \|\beta\|^2$$

❑ Why minimize $\|\beta\|^2$?

- Tries to keep coefficients small
- Only use coefficients when it reduces prediction error

❑ Without regularization:

- Parameters are unrestricted
- Models have high variance
- Large positive and negative coefficients cancel each other for correlated features

Ridge Regression

- ❑ Solution for given regularization level
 - Easily obtainable by setting gradient to zero (HW!)

$$J(\boldsymbol{\beta}) = \|\mathbf{y} - A\boldsymbol{\beta}\|^2 + \alpha\|\boldsymbol{\beta}\|^2$$

$$\boldsymbol{\beta}_{ridge} = (A^T A + \alpha I)^{-1} A^T \mathbf{y}$$

- ❑ How to determine the right regularization level α ?
 - Through cross validation!
- ❑ Sklearn function for ridge regression:
 - http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Coefficient path with ridge regression

Note that larger α does not lead to fewer non-zero coefficients, but only smaller (and mostly positive) coefficients!

Figure from [Hastie2008]: Hastie, Tibshirani, Friedman, The elements of statistical learning.

For more on this subject, see Sec. 3.4.1.

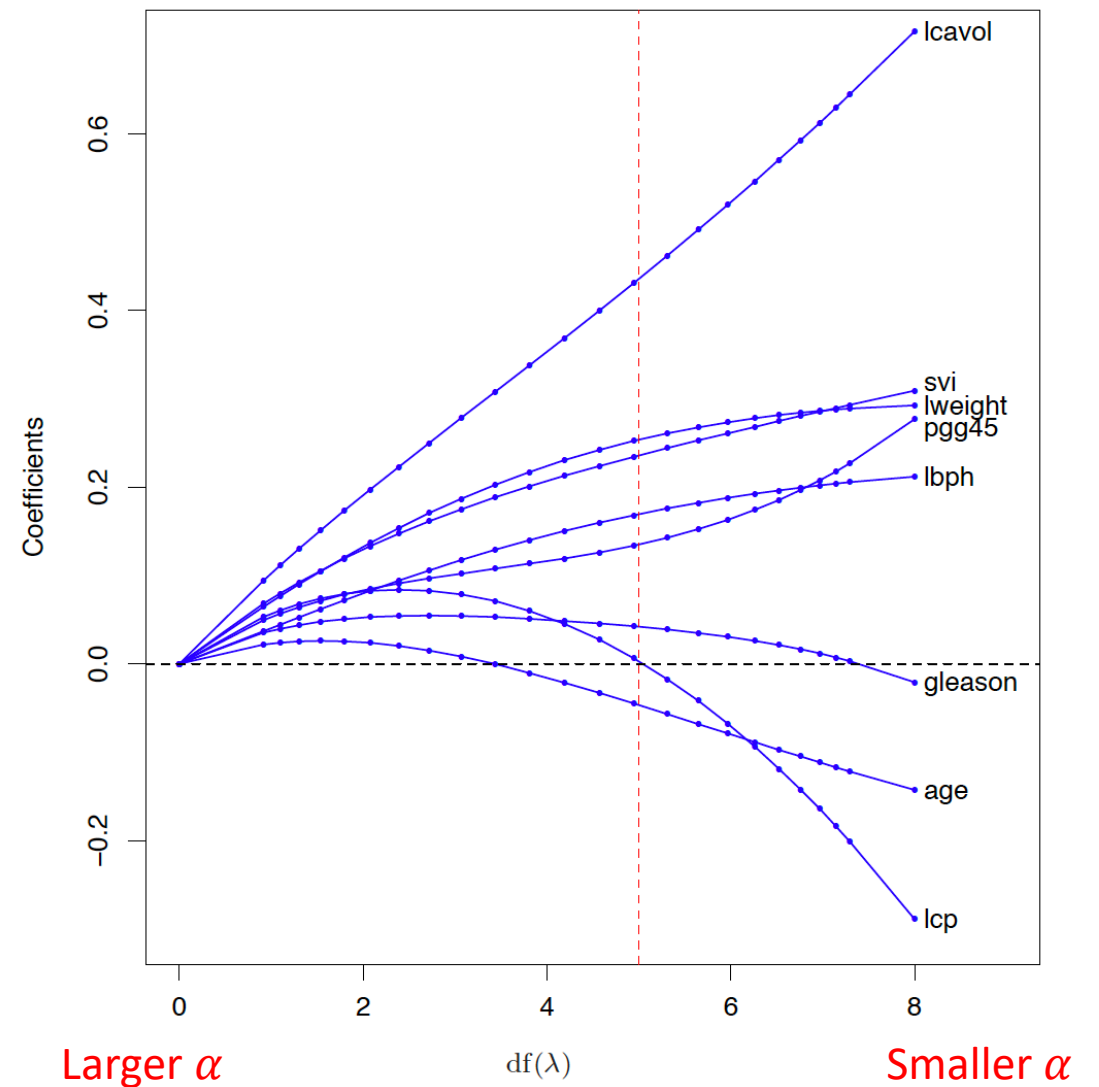


FIGURE 3.8. Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $df(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $df = 5.0$, the value chosen by cross-validation.

LASSO Regression

□ LASSO cost function:

$$J(\boldsymbol{\beta}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^d |\beta_j| = \|\mathbf{y} - A\boldsymbol{\beta}\|^2 + \alpha \|\boldsymbol{\beta}\|_1$$

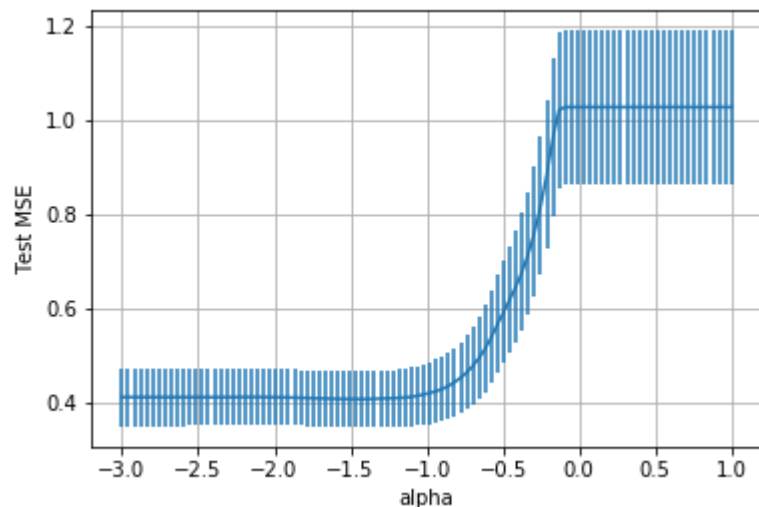
- Because derivative of $|\beta_j|$ is not continuous, there is no closed-form solution.
- However, there is a unique minimum because the cost function is convex.
- Many methods to solve iteratively
 - Least angle regression (LAR), coordinate descent, ADMM
 - Beyond the scope of this class
 - See textbook [Hastie2008] for LAR method

Selecting Regularization Level

- ❑ How do we select regularization level α ?
 - Higher $\alpha \Rightarrow$ More constrained / simpler model
 - Lower $\alpha \Rightarrow$ More complex model
- ❑ Similar to inverse of model order
- ❑ Find α via cross-validation

Computing LASSO in python

- ❑ Use sklearn Lasso method
 - Solve using coordinate descent
- ❑ Cross validation loop
 - Outer loop: Loop over folds
 - Inner loop: Loop over α
 - Measure mean and std deviation of MSE



```
: # Create a k-fold cross validation object
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)

# Create the LASSO model. We use the `warm start` parameter so
# This speeds up the fitting.
model = linear_model.Lasso(warm_start=True)

# Regularization values to test
nalpha = 100
alphas = np.logspace(-3,1,nalpha)

# MSE for each alpha and fold value
mse = np.zeros((nalpha,nfold))
for ifold, ind in enumerate(kf.split(X)):

    # Get the training data in the split
    Itr,Its = ind
    X_tr = X[Itr,:]
    y_tr = y[Itr]
    X_ts = X[Its,:]
    y_ts = y[Its]

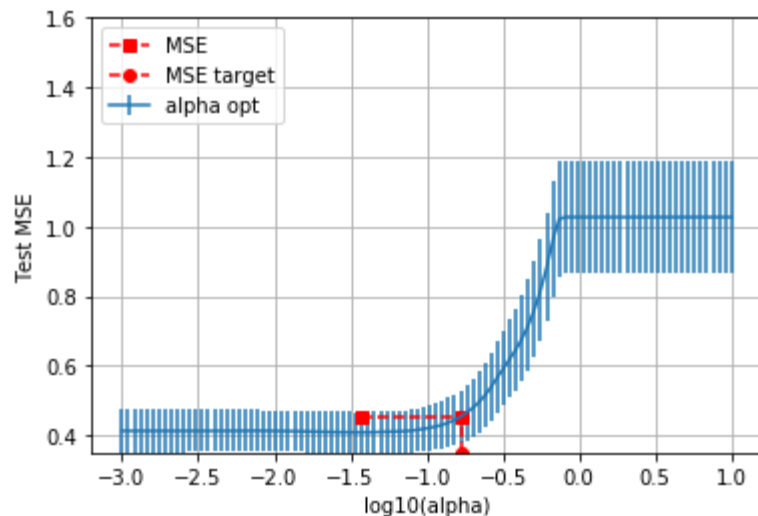
    # Compute the lasso path for the split
    for ia, a in enumerate(alphas):

        # Fit the model on the training data
        model.alpha = a
        model.fit(X_tr,y_tr)

        # Compute the prediction error on the test data
        y_ts_pred = model.predict(X_ts)
        mse[ia,ifold] = np.mean((y_ts_pred-y_ts)**2)
```

Using One Standard Deviation Rule

- Use one standard deviation rule from before
 - Find α_0 with minimum mean MSE, `mean_mean`
 - Set `mse_tgt = mse_mean[α_0] + mse_se[α_0]`
 - Find largest α where `mse_mean[α] < mse_tgt`



```
# Find the minimum MSE and MSE target
imin = np.argmin(mse_mean)
mse_tgt = mse_mean[imin] + mse_se[imin]
alpha_min = alphas[imin]

# Find the least complex model with mse_mean < mse_tgt
I = np.where(mse_mean < mse_tgt)[0]
iopt = I[-1]
alpha_opt = alphas[iopt]
print("Optimal alpha = %f" % alpha_opt)
```

Coefficients

- ❑ Select α via cross-validation
- ❑ Then, find coefficients using all training data.
- ❑ Final coefficients are sparse:
 - Only three factors are non-zeros
 - Lcavol: log cancer volume
 - Lweight: Log weight
 - Svi: seminal vesicle invasion
- ❑ Use only features corresponding to non-zero coefficients for linear regression

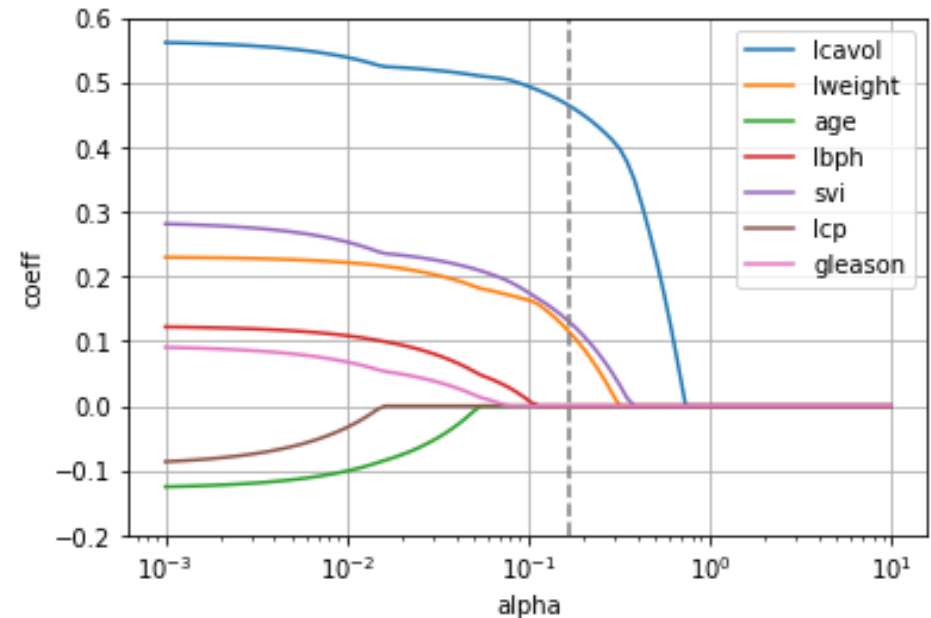
```
model.alpha = alpha_opt
model.fit(X,y)

# Print the coefficients
for i, c in enumerate(model.coef_):
    print("%8s %f" % (names_x[i], c))
```

```
lcavol 0.464526
lweight 0.115832
age 0.000000
lbph 0.000000
svi 0.131102
lcp 0.000000
gleason 0.000000
```

LASSO path

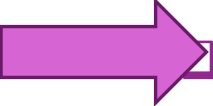
- Useful to plot coefficients as a function of α .
- Called the LASSO path
- Indicates relative importance of different factors
- For this data set:
 - lcavol most important
- Don't draw medical conclusions
 - Need more detailed significance testing
 - Complex subject for another class...



Finding the Final Regressor

- ❑ Select features from cross-validation
- ❑ Re-run ordinary (un-regularized) regression on reduced features
- ❑ Use K –fold validation
- ❑ K -folds yield K weights and biases
- ❑ Take mean of the weights and biases for the final parameter estimate
- ❑ Take mean of the test MSE for the estimate of the test MSE

Outline

- ❑ Motivating Example: Predicting prostate cancer from a PSA test
- ❑ Model Selection
- ❑ Model Selection from LASSO regularization
- ❑ Probabilistic interpretation
- ❑ Other Model Selection Methods
- ❑ In-Class Exercise: Audio Pitch Detection

Maximum Likelihood Estimate

- Suppose that true data generated from probabilistic model with Gaussian noise:

$$\mathbf{y} = A\boldsymbol{\beta} + \mathbf{w}, \quad w_i \sim N(0, \sigma^2)$$

- Maximum likelihood estimator:

$$\hat{\boldsymbol{\beta}} = \arg \max_{\boldsymbol{\beta}} p(\mathbf{y}|A, \boldsymbol{\beta}) = \arg \min_{\boldsymbol{\beta}} [-\ln p(\mathbf{y}|A, \boldsymbol{\beta})]$$

- Gaussian density for noise in \mathbf{y} : $\ln p(\mathbf{y}|A, \boldsymbol{\beta}) = -\frac{1}{2\sigma^2} \|\mathbf{y} - A\boldsymbol{\beta}\|^2$

- Hence

$$\hat{\boldsymbol{\beta}} = \arg \max_{\boldsymbol{\beta}} p(\mathbf{y}|A, \boldsymbol{\beta}) = \arg \min_{\boldsymbol{\beta}} [\|\mathbf{y} - A\boldsymbol{\beta}\|^2] = \text{Least Squares Solution}$$

Bayes Estimation (MAP Estimate)

□ Maximum a posterior (MAP) estimator of β :

$$\hat{\beta} = \arg \max_{\beta} p(\beta | y, A)$$

- $\hat{\beta}$ = Most likely parameter value given evidence y, A

□ Bayes Rule: $p(\beta | y, A) = p(y|A, \beta)p(\beta)/p(y|A)$

□ Hence: $\hat{\beta} = \arg \max_{\beta} p(y|A, \beta)p(\beta)$ (because y and A are fixed)

- Likelihood: $p(y|A, \beta)$ How well β matches data
- Prior: $p(\beta)$: How well β agrees with prior knowledge about its distribution (constraints)

□ More in probability class...

Bayes Estimation with Logarithms

- Often easier to use logarithms:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \arg \max_{\boldsymbol{\beta}} p(\mathbf{y} | A, \boldsymbol{\beta}) p(\boldsymbol{\beta}) = \arg \min_{\boldsymbol{\beta}} [-\ln p(\mathbf{y} | A, \boldsymbol{\beta}) p(\boldsymbol{\beta})] \\ &= \arg \min_{\boldsymbol{\beta}} [-\ln p(\mathbf{y} | A, \boldsymbol{\beta}) - \ln p(\boldsymbol{\beta})]\end{aligned}$$

- Gaussian density for noise in \mathbf{y} : $\ln p(\mathbf{y} | A, \boldsymbol{\beta}) = -\frac{1}{2\sigma^2} \|\mathbf{y} - A\boldsymbol{\beta}\|^2$

- Hence

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\frac{1}{2\sigma^2} \|\mathbf{y} - A\boldsymbol{\beta}\|^2 - \ln p(\boldsymbol{\beta}) \right] = \arg \min_{\boldsymbol{\beta}} [\|\mathbf{y} - A\boldsymbol{\beta}\|^2 + \phi(\boldsymbol{\beta})]$$

- **Conclusion:** MAP estimate = regularized LS with $\phi(\boldsymbol{\beta}) = -2\sigma^2 \ln p(\boldsymbol{\beta})$
 - Penalize $\boldsymbol{\beta}$ proportional to $-\ln p(\boldsymbol{\beta})$: Less likely $\boldsymbol{\beta}$ penalized more

Ridge and Lasso as Bayesian Estimators

□ Bayesian Estimator:

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\frac{1}{2\sigma^2} \|\mathbf{y} - A \boldsymbol{\beta}\|^2 - \ln p(\boldsymbol{\beta}) \right]$$

□ Assuming β_j are i.i.d. Gaussian with zero mean:

$$p(\beta_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\beta_j^2 / 2\sigma^2), \quad -\log p(\beta_j) = \beta_j^2 / 2\sigma^2 + \text{constants}$$

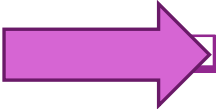
$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\|\mathbf{y} - A \boldsymbol{\beta}\|^2 + \frac{\sigma^2}{\gamma^2} \|\boldsymbol{\beta}\|^2 \right] = \text{Ridge Regression!}$$

□ Assuming β_j are i.i.d. Laplacian with zero mean:

$$p(\beta_j) = \frac{1}{2\sigma} \exp(-|\beta_j|/\sigma), \quad -\log p(\beta_j) = |\beta_j|/\sigma + \text{constant}$$

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\|\mathbf{y} - A \boldsymbol{\beta}\|^2 + \frac{2\sigma^2}{\gamma} \|\boldsymbol{\beta}\|_1 \right] = \text{Lasso Regression!}$$

Outline

- ❑ Motivating Example: Predicting prostate cancer from a PSA test
- ❑ Model Selection
- ❑ Model Selection from LASSO regularization
- ❑ Probabilistic interpretation
- ❑ Other Model Selection Methods
- ❑ In-Class Exercise: Audio Pitch Detection

Other feature selection methods

❑ Filtering method:

- Rank the features based on their correlation or mutual information with the target and possibly the redundancy among the features
- Simple but not very good

❑ Wrapper method:

- For each candidate feature subset, apply a chosen classifier/regressor, evaluate the cross validation accuracy. Go through all possible feature subsets, or test the subsets in some greedy way
- Computationally expensive

❑ Embedded method:

- Some regression/classification method naturally lead to feature ranking and selection

❑ What is available in Python:

- http://scikit-learn.org/stable/modules/feature_selection.html

Filtering method

- Rank the features based on their correlation with the target
 - Can use other metrics: Correlation, F-test, mutual information, ...
- Also should consider the redundancy (correlation) among chosen features
 - **Minimal Redundancy Maximum Relevance (mRMR)**
 - Peng, H.C., Long, F., and Ding, C., "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 8, pp. 1226–1238, 2005.
 - <http://home.penglab.com/proj/mRMR/>
 - <https://www.mathworks.com/matlabcentral/fileexchange/14916-minimum-redundancy-maximum-relevance-feature-selection>

Ranking metrics

❑ Correlation coefficient between a feature and the target

❑ F-test: test the significance of using one feature vs. not using any (use the mean of y only. Essentially measure the difference in the MSE when using only the mean value of y vs. using a single feature.

$$f_{test} = \frac{r^2}{1-r^2}(n_{sample}-2)$$

❑ Mutual information between a feature and the target

$$I(X, Y) = \iint p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy$$

Embedded Method

- ❑ Results from some regression/classification methods allow feature selection
 - Linear regression: based on coefficient magnitude
 - Neural net: based on weight magnitude
 - Decision tree: based on tree level
 - Can add regularization terms on the coefficients/weights to encourage sparsity
 - LASSO regression
- ❑ Recursive feature elimination
 - Starting with all features, remove one feature that has the lowest importance (e.g. smallest coefficient magnitude)
 - Recursive feature elimination in sklearn
 - http://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_digits.html#sphx-glr-auto-examples-feature-selection-plot-rfe-digits-py
 - http://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-selection-plot-rfe-with-cross-validation-py

Wrapper method

- For each candidate feature subset, apply a chosen classifier/regressor, evaluate the cross validation accuracy. Go through all possible feature subsets, or test the subsets in some greedy way
 - Exhaustive search
 - Genetic algorithm
 - Forward stepwise
 - Backward stepwise

Exhaustive search for feature selection

- ❑ Suppose you want to consider feature subset of size up to p
- ❑ For all possible feature subsets of size 1 to p :
 - use cross validation to find mean RSS mean and standard deviation
- ❑ Choose the subset with the minimal RSS mean,
 - Or use the one standard error rule.
- ❑ When the number of features is large, may not be computationally feasible
- ❑ Fast search algorithms:
 - Genetic algorithm

Greedy feature selection

❑ Forward-Stepwise Selection

- Select one feature from all features that provides the lowest RSS with cross validation
- Select one new feature from all remaining features, so that previously chosen features plus the new feature provides the lowest RSS
- Repeat until the maximum feature number is reached, or when the RSS starts to increase

❑ Backward-Stepwise

- First use all features and find the RSS (using cross validation)
- Remove one feature and find the new RSS. Go through all possible features to remove.
- Find the one that leads to the least RSS increase. Remove this feature.
- Repeat the above, remove one from the remaining features, to find the next most important feature.

❑ Except exhaustive search, can all lead to suboptimal solution

Comparison of feature selection methods

Figure from [Hastie2008]: Hastie, Tibshirani, Friedman, The elements of statistical learning.

For more on this subject, see Sec. 3.3

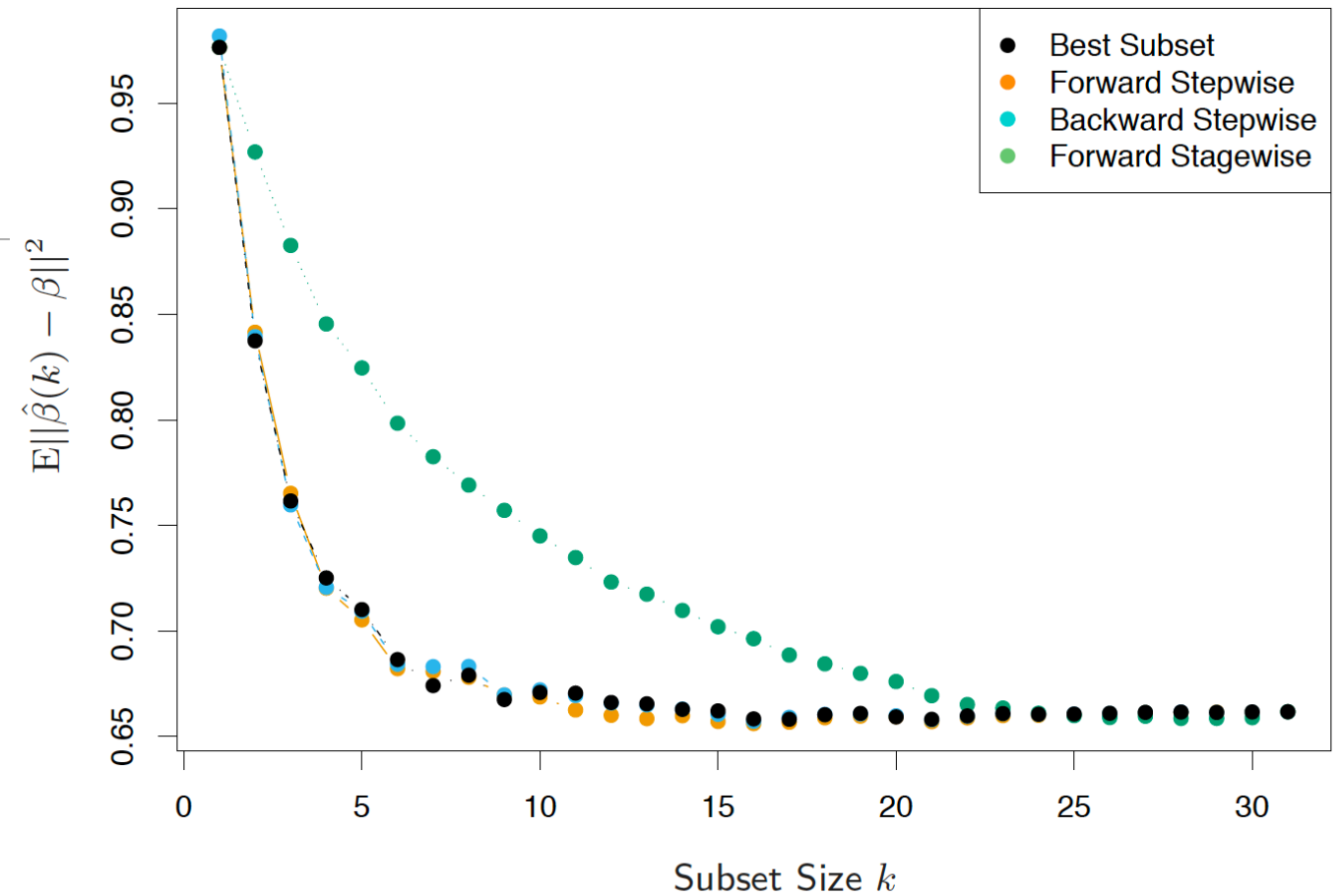


FIGURE 3.6. Comparison of four subset-selection techniques on a simulated linear regression problem $Y = X^T \beta + \varepsilon$. There are $N = 300$ observations on $p = 31$ standard Gaussian variables, with pairwise correlations all equal to 0.85. For 10 of the variables, the coefficients are drawn at random from a $N(0, 0.4)$ distribution; the rest are zero. The noise $\varepsilon \sim N(0, 6.25)$, resulting in a signal-to-noise ratio of 0.64. Results are averaged over 50 simulations. Shown is the mean-squared error of the estimated coefficient $\hat{\beta}(k)$ at each step from the true β .

Going through demo comparing different feature selection methods

More about cross validation

❑ Why do we use cross validation?

- To estimate the test error when there are insufficient training data so that we can partition the total data to a **large** training set and a **large** test set.
- Whether a dataset is large depends on the number of parameters of the model to be trained.
- Ideally the number of samples should be $>100x$ of the number of parameters, but at least $10x$.

❑ When you have sufficient training data, you can just use a certain percentage (e.g. 50%) for training and remaining for testing. The error on the testing set would be a reliable estimate of the test error.


❑ Two ways of using cross validation

- When the “best” model class, model order, and feature set are known:
 - Use CV to estimate the test error
- Use CV to determine the appropriate model class, model order and feature subset
 - For each candidate model class, model order, and feature subset, evaluate CV error
 - Determine which candidate yields the least CV error.

More about cross validation

- ❑ How to use the multiple estimated models from multiple trials?
 - Apply each on a test sample and take the average (for regression) or majority (for classification) of results
 - For linear regression, equivalent to average the model coefficients
- ❑ When your data is limited, you may want to go beyond K-folds
 - Ex: 5-fold means that you partition the data to 5 parts in some way, each part has 20% of data, and only do 5 fold training and testing
 - When your data is small, the average CV error is still very sensitive to how the data is partitioned to 5 parts. If you use random shuffling, you will get different result each time.
 - Instead, you could do L trials ($L \gg 5$) of random sampling, each time using 80% for training and 20% for testing
- ❑ How to handle limited data in machine learning is still a challenging topic!

Outline

- ❑ Motivating Example: Predicting prostate cancer from a PSA test
- ❑ Model Selection
- ❑ Model Selection from LASSO regularization
- ❑ Probabilistic interpretation
- ❑ Other Model Selection Methods
- ❑ In-Class Exercise: Audio Pitch Detection

In Class Exercise

❏ https://github.com/sdrangan/introml/blob/master/unit05_lasso/lasso_in_class.ipynb

LASSO Regression In-Class Exercise

In this exercise, we will see how to use LASSO for pitch detection in audio.

We load the following packages.

```
import numpy as np
import matplotlib.pyplot as plt
import pickle
```

Load the data

The data is taken from a sample of about 20~ms of audio from a viola. I have already pre-processed the data. You can load it with following command. The value t is the time (in seconds) and y is the sample of audio (this is a mono recording).

```
fn_src = 'https://raw.githubusercontent.com/sdrangan/introml/master/unit05_lasso/viola_sample.p'
fn_dst = 'viola_sample.p'

import os
from six.moves import urllib

if os.path.isfile(fn_dst):
```

What You Should Know to Do

- ☐ Formulate a linear estimation problem with a regularization
- ☐ Compute an L1-regularized estimate (LASSO) using sklearn tools
- ☐ Compute the optimal regularization level using cross validation
- ☐ Interpret results from a LASSO path
- ☐ Determine final regression function from cross validation
- ☐ Set regularizer based on a probabilistic prior
- ☐ Perform other feature selection methods