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
- ❑Set regularizer based on a probabilistic prior
- ❑Feature selection methods
- ❑How to determine final regression function from cross validation

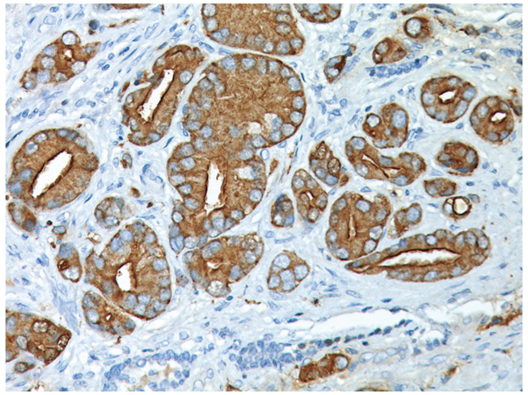
Outline

- ❑ Motivating Example: Predicting prostate cancer from a PSA test
- ❑ Model selection from LASSO regularization
- ❑ Probabilistic interpretation

Prostate Specific Antigen Testing

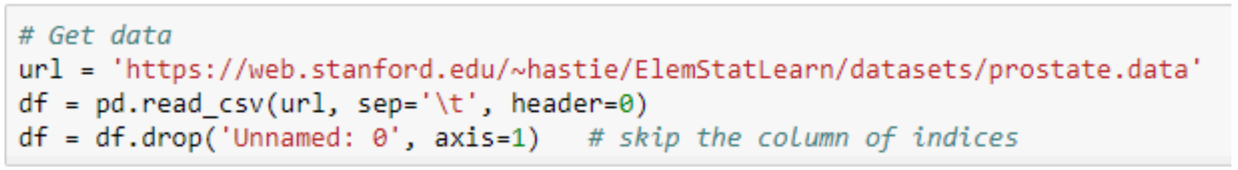
- ❑ PSA levels easily tested
- ❑ High PSA believed to be associated with prostate cancer
  - Potential tool for screening
- ❑ Classic 1989 study by Thomas et al:
  - Measured PSA level of 102 men prior to prostate removal
  - Measured characteristics of prostate from samples
  - Characteristics include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

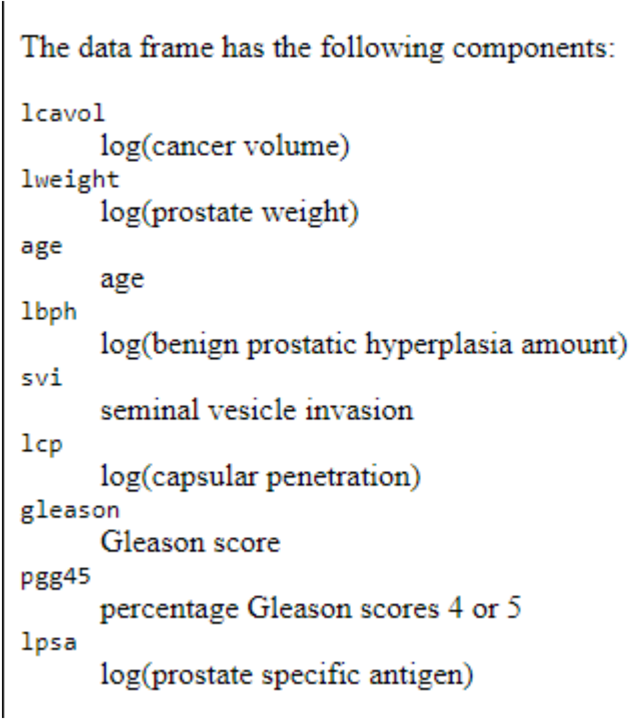


Data

- ❑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)



```python
# 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)

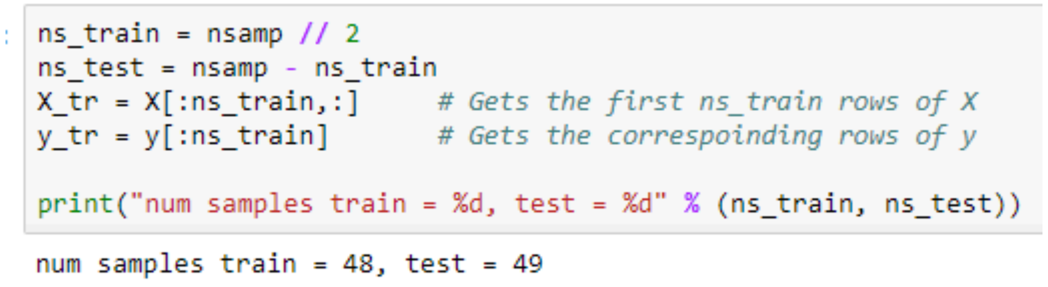
First Try: Linear Model

- Simple idea: Use linear regression

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

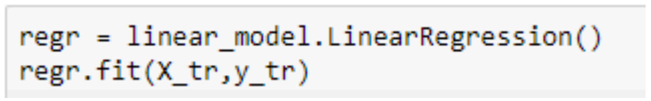
  - $y =$ lpsa (target PSA level)
  - $x_1, \ldots, x_d =$ prostate features (d = 8)

- Why linear regression?
  - Easy to compute / interpret
  - Coefficients are easy to interpret
  - Larger coefficients ⇒ larger influence of feature on PSA



```
ns_train = nsamp // 2
ns_test = nsamp - ns_train
X_tr = X[:ns_train,:]     # Gets the first ns_train rows of X
y_tr = y[:ns_train]       # Gets the corresponding rows of y

print("num samples train = %d, test = %d" % (ns_train, ns_test))
```

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

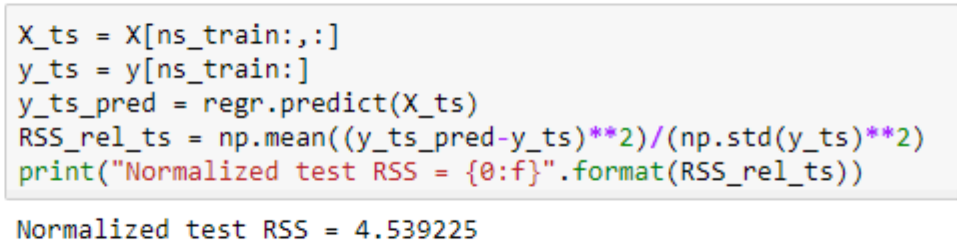


```
regr = linear_model.LinearRegression()
regr.fit(X_tr,y_tr)
```

Model Does Not Generalize

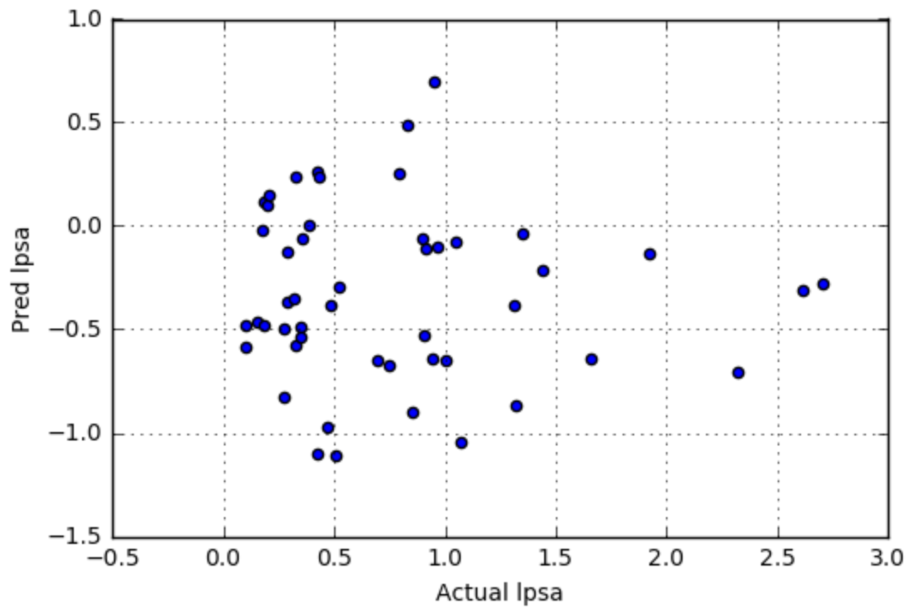
- Evaluate model with cross validation
  - Train on 48 samples
  - Measure RSS on 49 samples
- Test RSS is very high
- Scatter plot shows no predictive ability

- What happened?
- Can we do a better model?



```
X_ts = X[ns_train:,:]
y_ts = y[ns_train:]
y_ts_pred = regr.predict(X_ts)
RSS_rel_ts = np.mean((y_ts_pred-y_ts)**2)/(np.std(y_ts)**2)
print("Normalized test RSS = {0:f}".format(RSS_rel_ts))
```

```
Normalized test RSS = 4.539225
```

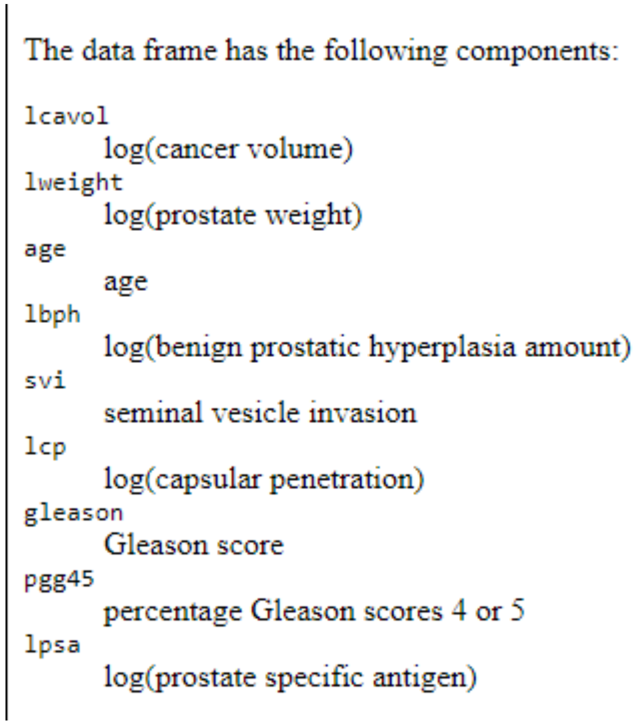


Outline

- ❑Motivating Example: Predicting prostate cancer from a PSA test
- ❑Model selection from LASSO regularization
- ❑Probabilistic interpretation

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):

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

- Regularized estimator:

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

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

- Concept: Regularizer penalizes β that are "unlikely"
  - Constrains estimate to smaller set of parameters

Two Common Regularizers

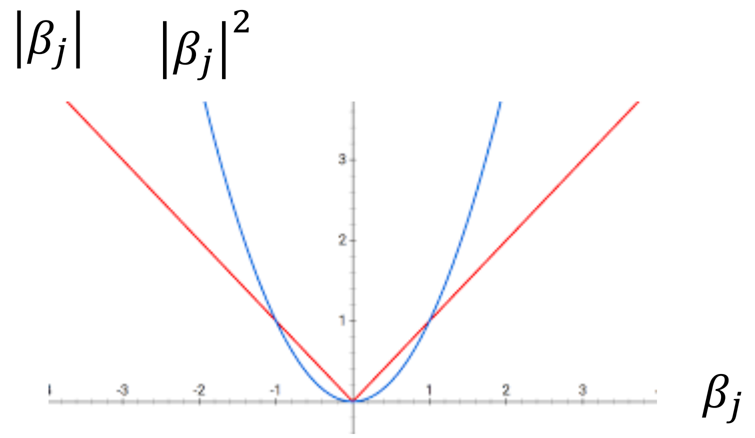
- Ridge regression (called L2)

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

- 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, as this terms depends on the mean of the target, and 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)
  - $x_{i,j} \rightarrow (x_{i,j} - E(x_{i,j}))/\mathrm{STD}(x_{i,j})$
  - $y_i \rightarrow \left(y_i - \mathrm{E}(y_i)\right)/\mathrm{STD}(y_i)$
  - Once the predictor for the scaled data are determined, we can 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:

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

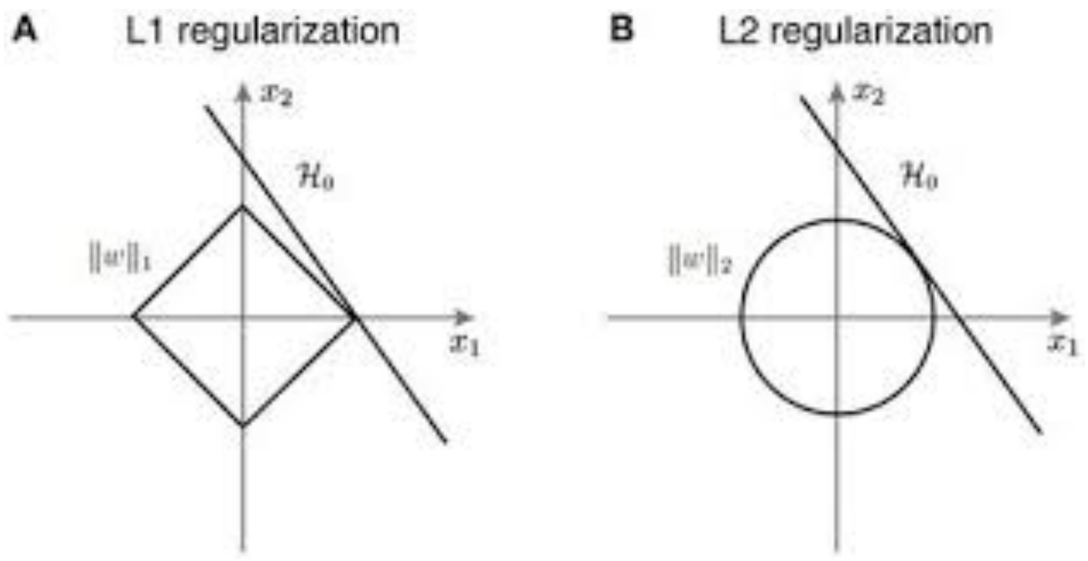
- LASSO cost function:

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

  - $\|\boldsymbol{\beta}\|_1$ = 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



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 $||\beta||^2$?
- Without regularization, large positive and negative coefficients cancel each other for correlated features, resulting in high variance of the resulting models

Ridge Regression

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

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

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

- 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.

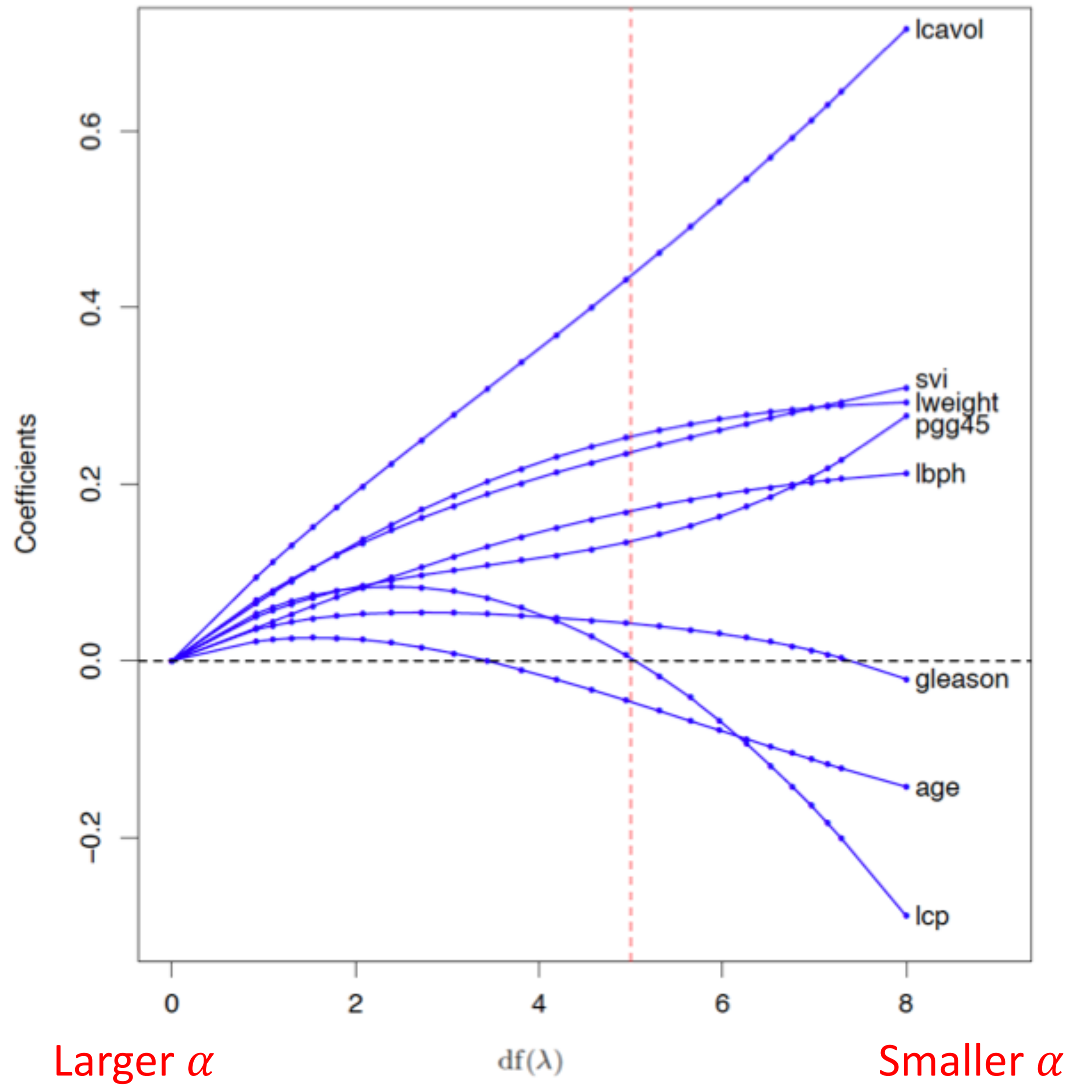


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $\mathrm{df}(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $\mathrm{df} = 5.0$, the value chosen by cross-validation.*

LASSO Regression

- ❑ LASSO cost function:

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

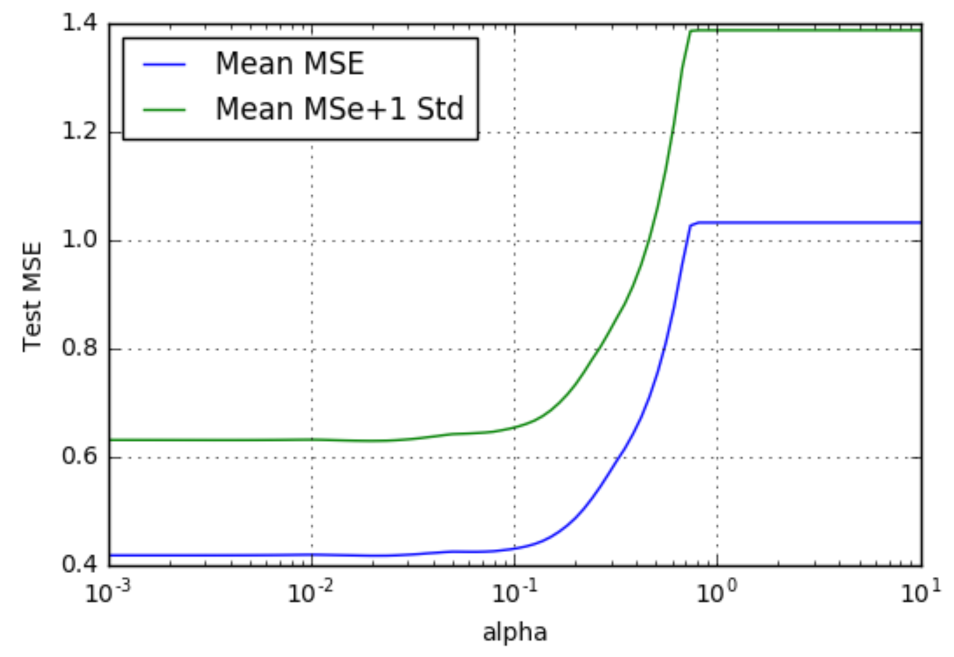
- ❑ Because derivative of $|\beta_j|$ is not continuous, there is no closed-form solution.
- ❑ However, there is a unique minimum because the cost function is convex.
- ❑ Many methods to solve iteratively
  - Least angle regression (LAR), coordinate descent, ADMM
  - Beyond the scope of this class
 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

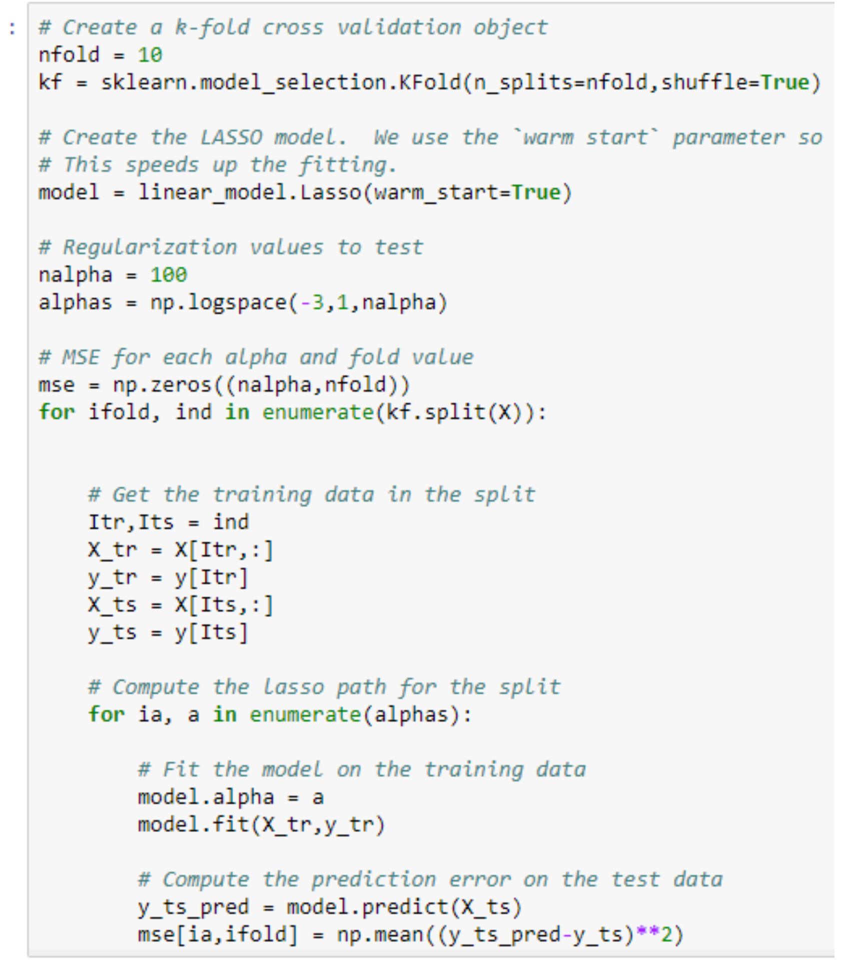
Selecting Regularization Level

- 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

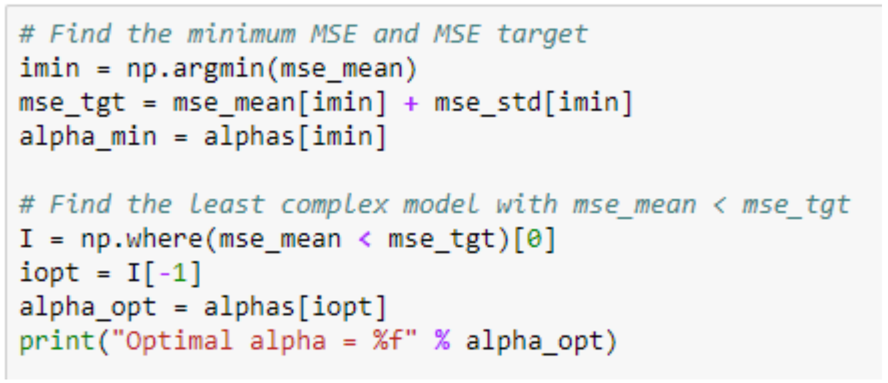




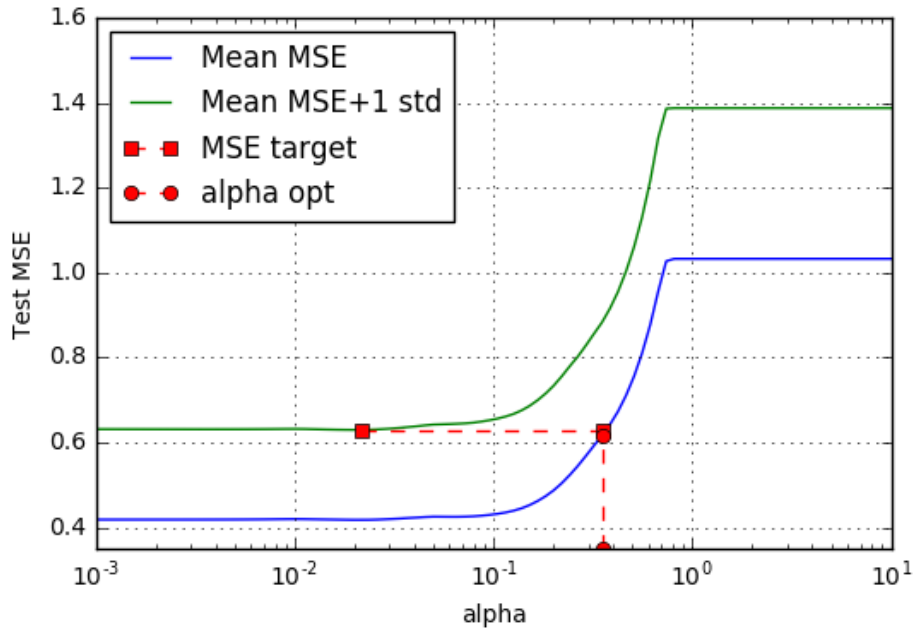
```
# Create a k-fold cross validation object
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)

# 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 $\text{mse\_tgt} = \text{mse\_mean}[\alpha_0] + \text{mse\_std}[\alpha_0]$
  - Find largest α where mse_mean[α] < mse_tgt

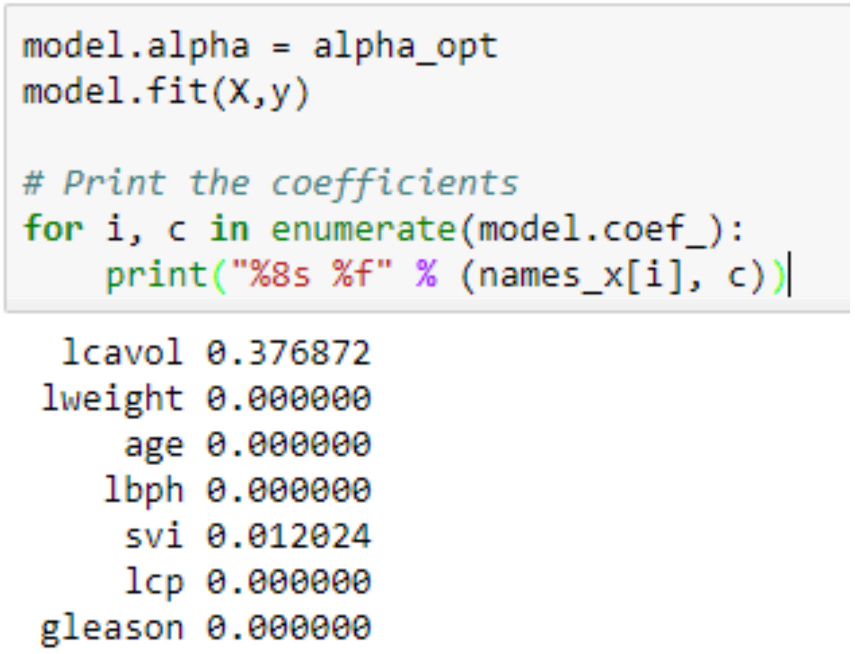


```python
# Find the minimum MSE and MSE target
imin = np.argmin(mse_mean)
mse_tgt = mse_mean[imin] + mse_std[imin]
alpha_min = alphas[imin]

# 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 two factors are non-zeros
  - Lcavol: log cancer volume
  - Svi: seminal vesicle invasion
- Use only features corresponding to non-zero coefficients for linear regression

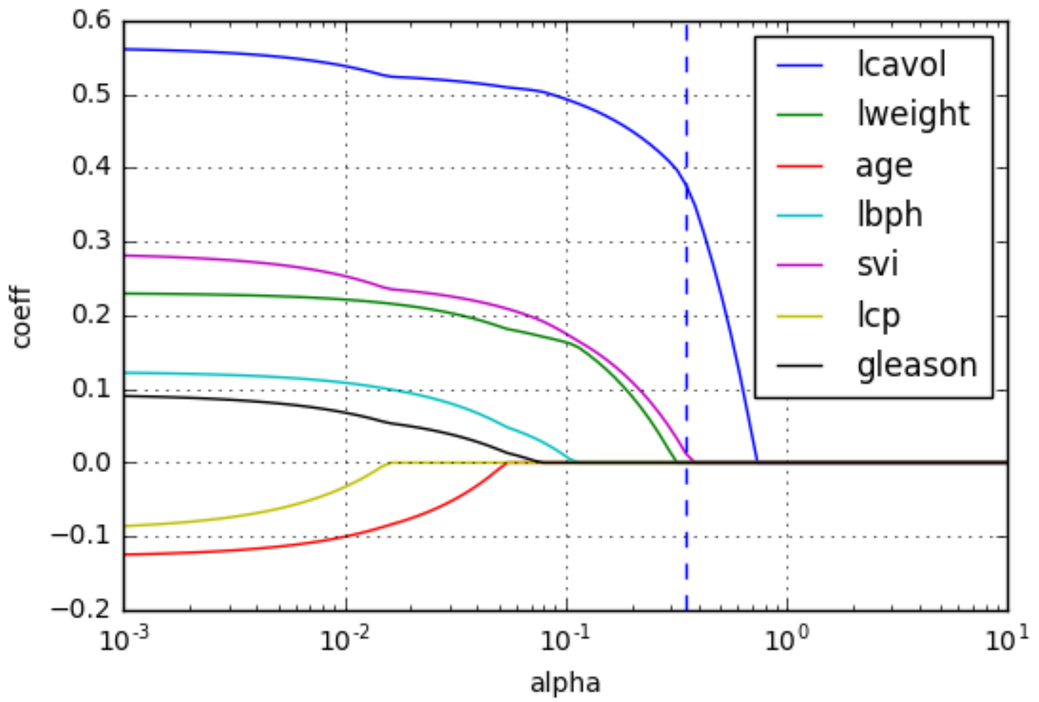


```
model.alpha = alpha_opt
model.fit(X,y)

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

```
  lcavol 0.376872
 lweight 0.000000
     age 0.000000
    lbph 0.000000
     svi 0.012024
     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…



How to determine the final regressor from cross validation?

- ❑ K-folds yield K regression functions
- ❑ We can produce K estimates for each test sample, and use the average (=mean estimate)
- ❑ When the regressor is linear with respect to its parameters, we can simply average the parameters! (HW)

Go through Demo on LASSO

Outline

- ❑ Motivating Example: Predicting prostate cancer from a PSA test
- ❑ Model selection from LASSO regularization
- ❑ Probabilistic interpretation
  - Least squares estimate is Maximum Likelihood Estimate
  - Ridge and Lasso are Maximum a Posterior (MAP) Estimates with different prior distributions for β

Maximum Likelihood Estimate

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

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

- Maximum likelihood estimator:

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

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

- Hence

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

Bayes Estimation (MAP Estimate)

- Maximum a posterior (MAP) estimator of $\boldsymbol{\beta}$:

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

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

- Bayes Rule: $p(\boldsymbol{\beta}\,|\boldsymbol{y}, A) = p(\boldsymbol{y}|A, \boldsymbol{\beta}\,)p(\boldsymbol{\beta})/p(\boldsymbol{y}|A)$

- Hence: $\widehat{\boldsymbol{\beta}} = \arg\max_{\beta} p(y|A, \boldsymbol{\beta}\,)p(\boldsymbol{\beta})$ (because $\boldsymbol{y}$ and A are fixed)
  - Likelihood: $p(y|A, \boldsymbol{\beta}\,)$ How well $\boldsymbol{\beta}$ matches data
  - Prior: $p(\boldsymbol{\beta}\,)$: How well $\boldsymbol{\beta}$ agrees with prior knowledge about its distribution (constraints)

- More in probability class…

Bayes Estimation with Logarithms

- Often easier to use logarithms:

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

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

- Hence

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

- 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:

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

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

$$p(\beta_j) = \tfrac{1}{2\pi\sigma} exp(-\beta_j{}^2/2\gamma^2),\ -\log p(\beta_j) = \beta_j{}^2/2\gamma^2 + constants$$

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

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

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

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

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 (or other statistics) with the target
  - 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.

$$ftest = \frac{r^2}{1-r^2}(\text{nsample-2})$$

- ❑Mutual information between a feature and the target

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

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 for each feature subset.
- ❑ 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

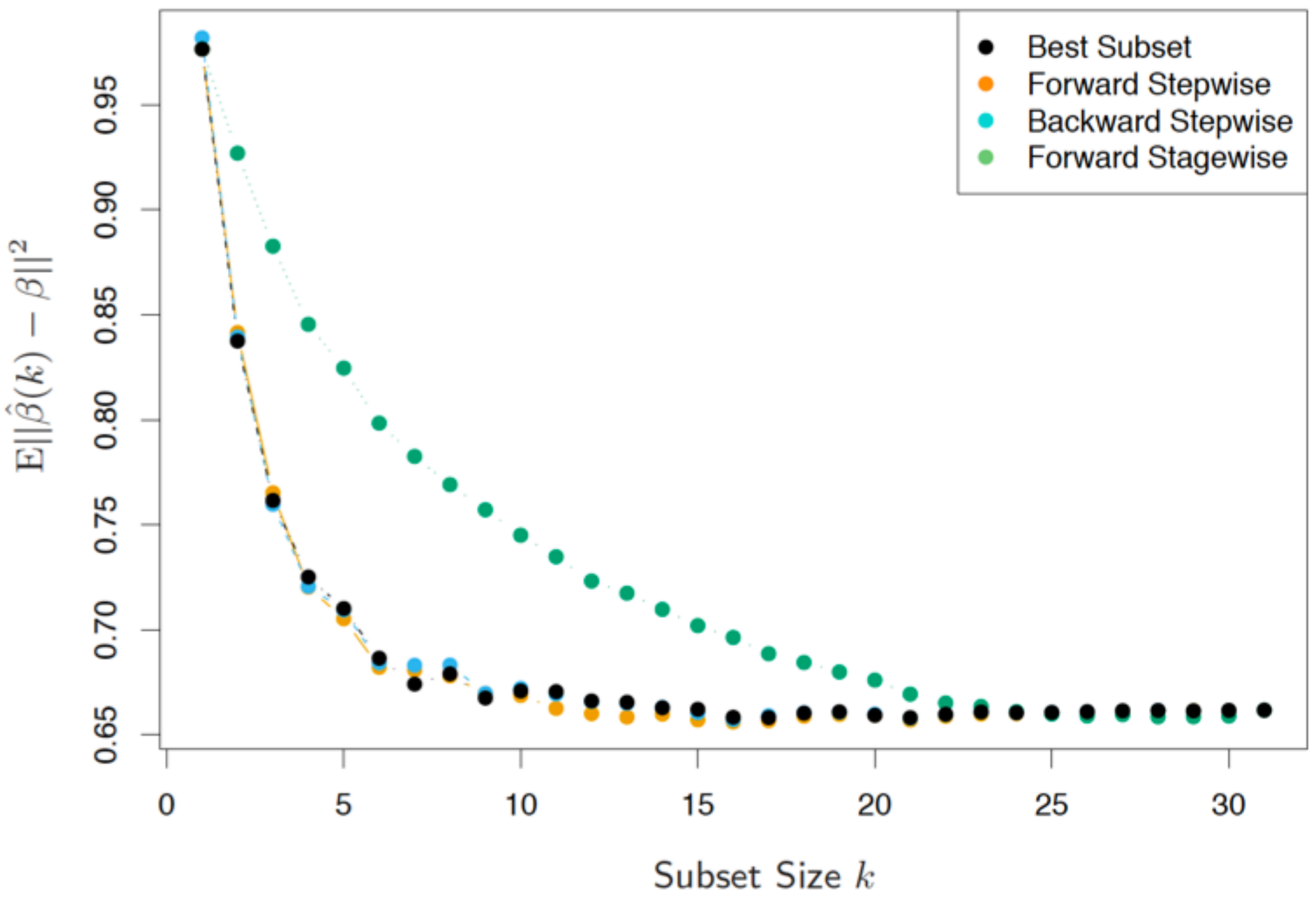


FIGURE 3.6. *Comparison of four subset-selection techniques on a simulated linear regression problem $Y = X^T\beta + \varepsilon$. There are $N = 300$ observations on $p = 31$ standard Gaussian variables, with pairwise correlations all equal to 0.85. For 10 of the variables, the coefficients are drawn 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

What you should know

- ❑Formulate a linear estimation problem with a proper regularization term
- ❑Compute an L1-regularized estimate (LASSO) using sklearn tools
- ❑Compute the optimal regularization level using cross validation
- ❑Interpret results from a LASSO path
- ❑Set the regularizer based on a probabilistic prior
- ❑Different feature selection methods