Computer Science

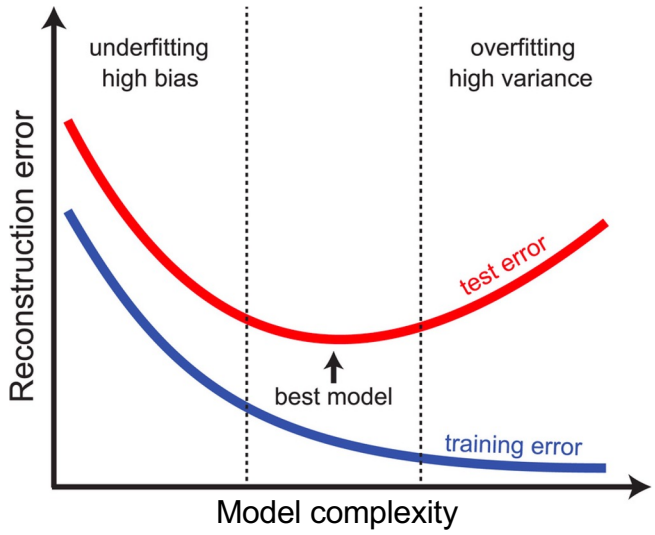
CSC380: Principles of Data Science

Linear Models 3

Credit:
- Jason Pacheco,
- Kwang-Sung Jun,
- Chicheng Zhang
- Xinchen yu

1

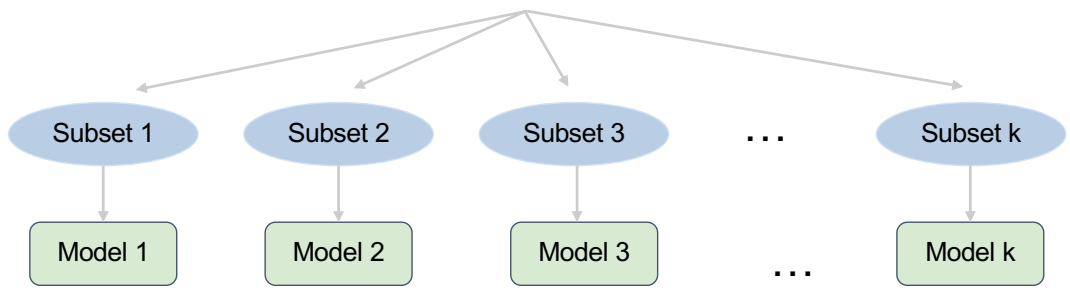
Bias-Variance Tradeoff



2

Bias-Variance Tradeoff: an example

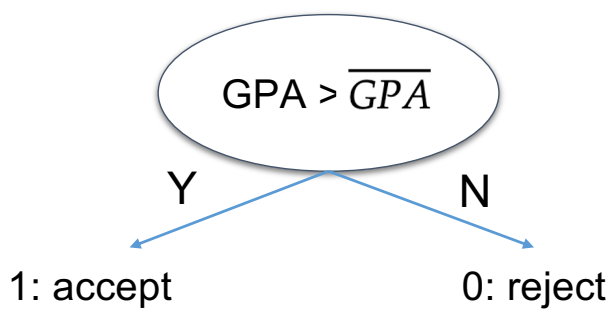
Student ID	GPA	Working experiences	…	Demographics
0	…	…	…	…
…	…	…	…	…



Task: given a new student, predict if accepted or rejected.

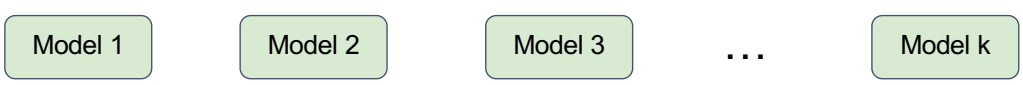
3

Option 1: overly simple model



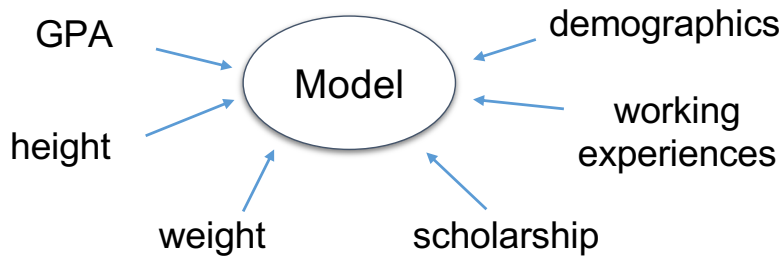
Models are similar but wrong on average:
- Bias high
- Variance low

$\overline{GPA_i}$ is similar for i = 1, 2, … k



4

Option 2: overly complex model



Models are different but right on average:

- Bias low
- Variance high

small changes in training set → big changes in the model

Model 1 | Model 2 | Model 3 | … | Model k

5

Regularized Least Squares

6

Ordinary least-squares (OLS) estimation (no regularizer),

$$w^{\text{OLS}} = \arg\min_{w} \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2$$

L2 norm: $\|w\| = \sqrt{\sum_{d=1}^{D} w_d^2}$

L1 norm: $\|w\|_1 = \sum_{d=1}^{D} |w_d|$

L2-regularized Least-Squares (Ridge)

$$w^{\text{L2}} = \arg\min_{w} \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2 + \lambda \|w\|^2$$

Convention: Just saying 'RLS' means L2-RLS

6

Constrained Optimization Viewpoint

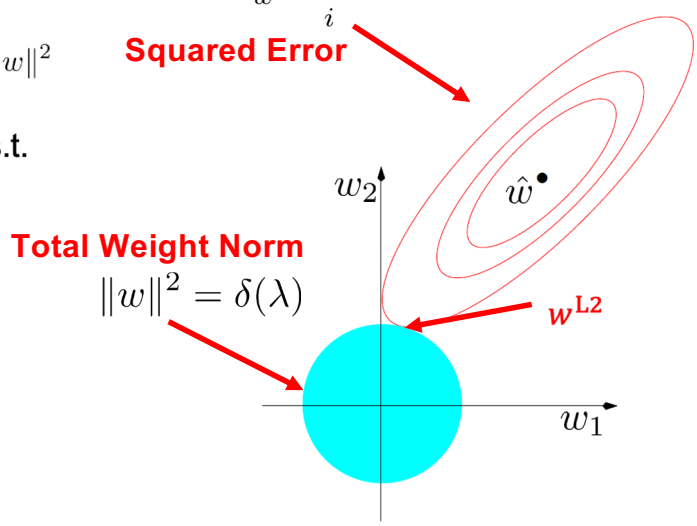
(Theorem) If

$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2 + \lambda \|w\|^2$$

then there exists a function $\delta(\lambda)$ s.t.

$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2 \quad \text{subject to} \quad \|w\|^2 \le \delta(\lambda)$$

$$\hat{w} = \arg\min_w \sum_i (y^{(i)} - w^T x^{(i)})^2$$



[Source: Hastie et al. (2001)]

7

Regularized Least Squares

8

Ordinary least-squares (OLS) estimation (no regularizer),

$$w^{\mathrm{OLS}} = \arg\min_w \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2$$

L2 norm: $\|w\| = \sqrt{\sum_{d=1}^{D} w_d^2}$

L1 norm: $\|w\|_1 = \sum_{d=1}^{D} |w_d|$

L2-regularized Least-Squares (Ridge)

$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2 + \lambda \|w\|^2$$

Convention: Just saying 'RLS' means L2-RLS

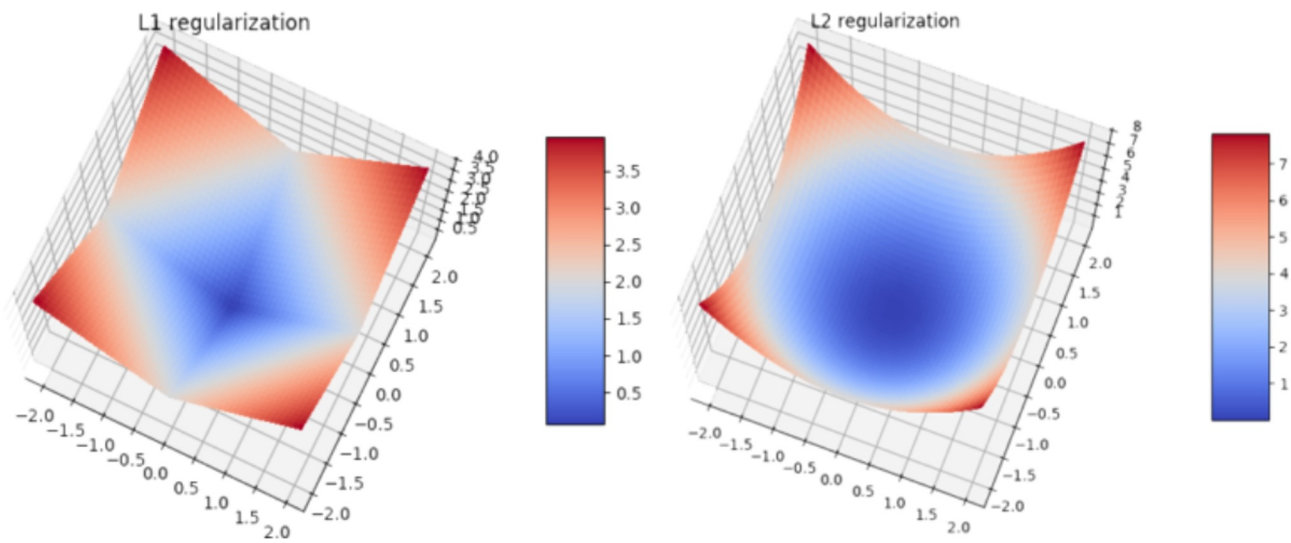
L1-regularized Least-Squares (LASSO) LASSO: Least Absolute Shrinkage and Selection Operator

$$w^{\mathrm{L2}} = \arg\min_w \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2 + \lambda \|w\|_1$$

8

L1 vs L2

9



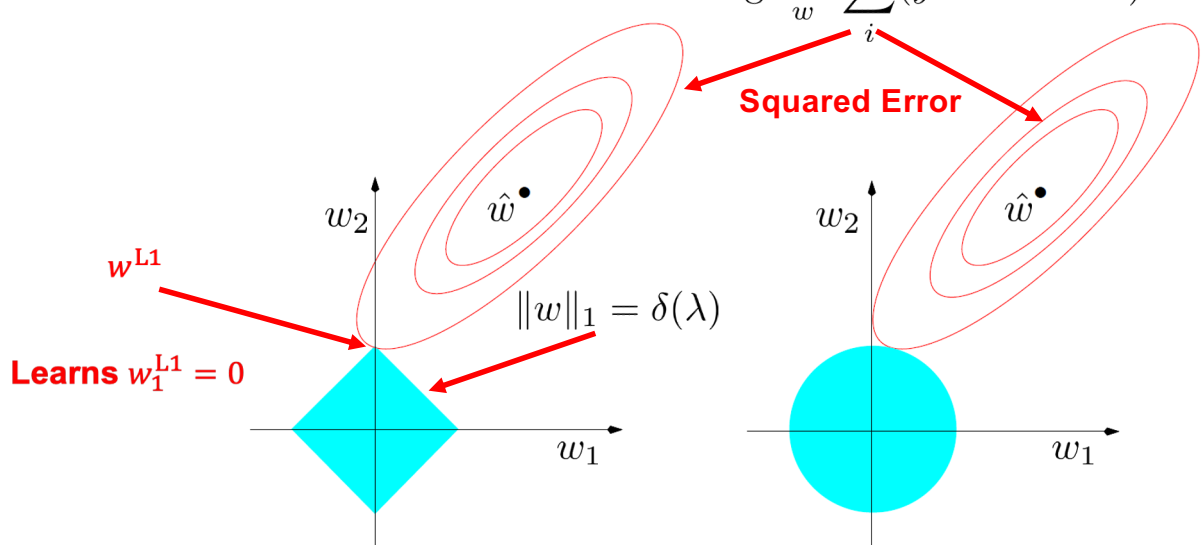
Looks subtle but L2-RLS and L1-RLS are often quite different!

9

L1 Regularized Least-Squares

10

$$\hat{w} = \arg\min_{w} \sum_{i} (y^{(i)} - w^T x^{(i)})^2$$

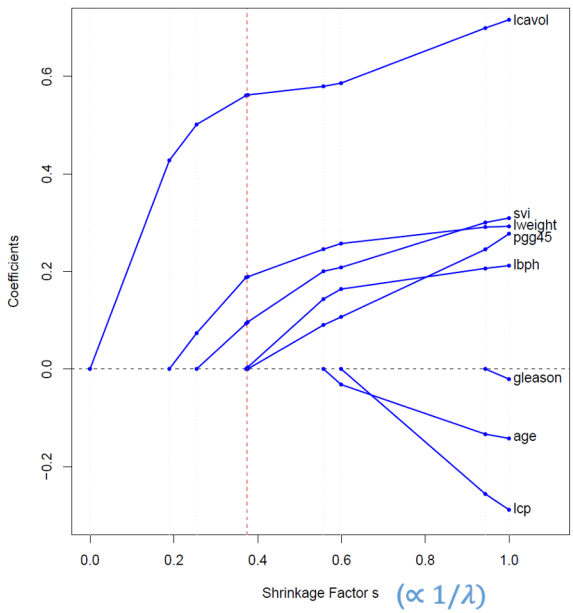


Quite often, zero-out coefficients that are not predictive…

10

Feature Weight Profiles

11



Varying regularization parameter adjusts *shrinkage factor*

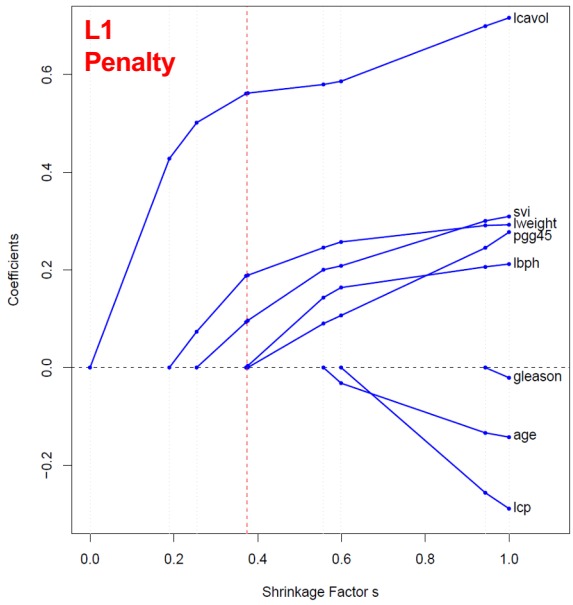
For moderate regularization strength weights for many features go to zero

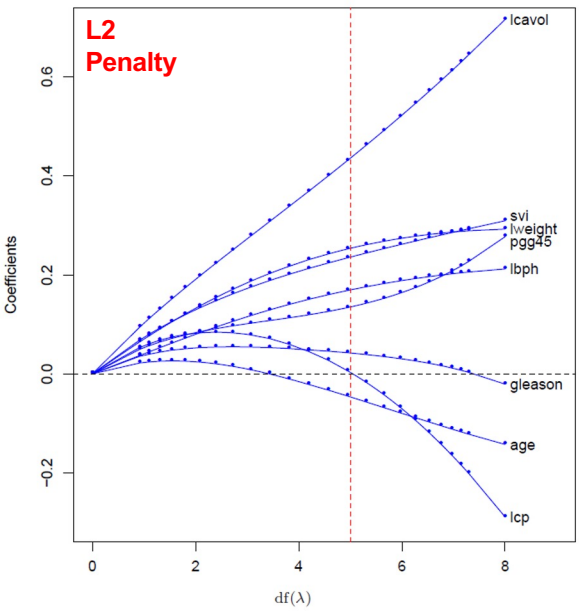
- Induces *feature sparsity*
- Ideal for high-dimensional settings since it reduced variance from having too many features!
- Gracefully handles D>m case, for D features and m training data

11

Feature Weight Profiles

12

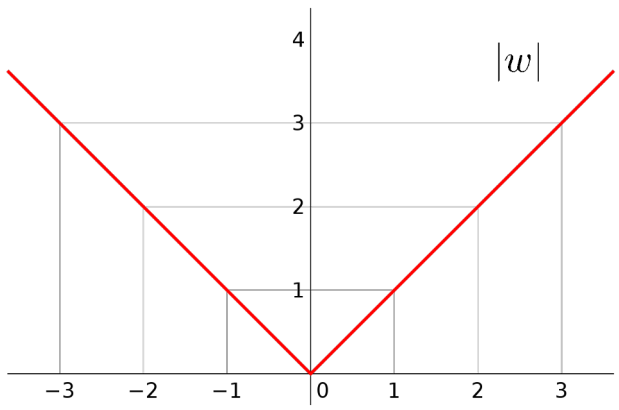




12

Learning L1 Regularized Least-Squares 13

$$w^{\mathrm{L2}} = \arg\min_{w} \sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2 + \lambda \|w\|_1$$



Not differentiable…

$$\frac{d}{dx}|x|$$

…doesn't exist at x=0

Can't set derivatives to zero as in the L2 case!

13

Learning L1 Regularized Least-Squares 14

- **<u>Not differentiable</u>**, no closed-form solution. => Need to use iterative methods
- But it is **<u>convex</u>**!
  - Global minimum can be found!
  - Efficient optimization algorithms exist
- *Least Angle Regression* (LAR) computes full solution path for a range of values λ

14

15

sklearn.linear_model.Lasso

class sklearn.linear_model.Lasso(*alpha=1.0, *, fit_intercept=True, normalize='deprecated', precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*) [source]

Parameters:

alpha : *float, default=1.0* ← this is λ

Constant that multiplies the L1 term. Defaults to 1.0. `alpha = 0` is equivalent to an ordinary least square, solved by the `LinearRegression` object. For numerical reasons, using `alpha = 0` with the `Lasso` object is not advised. Given this, you should use the `LinearRegression` object.

fit_intercept : *bool, default=True*

Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

precompute : *'auto', bool or array-like of shape (n_features, n_features), precompute*

Whether to use a precomputed Gram matrix to speed up calculations. The Gram matrix can also be passed as argument. For sparse input this option is always `False` to preserve sparsity.

copy_X : *bool, default=True*

If `True`, X will be copied; else, it may be overwritten.

15

16

Specialized methods for cross-validation…

sklearn.linear_model.LassoCV

class sklearn.linear_model.LassoCV(**, eps=0.001, n_alphas=100, alphas=None, fit_intercept=True, normalize='deprecated', precompute='auto', max_iter=1000, tol=0.0001, copy_X=True, cv=None, verbose=False, n_jobs=None, positive=False, random_state=None, selection='cyclic'*) [source]

Tries out a range of α values and reports the best, but maintains other values of α as well.

16

L1 Regression Cross-Validation

17

Perform L1 Least Squares (LASSO) 20-fold cross-validation,

```
model = LassoCV(cv=20).fit(X, y)
```

Plot the error for range of alphas,

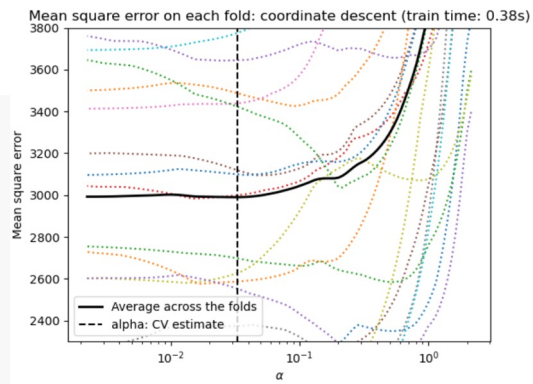
```
plt.figure()
ymin, ymax = 2300, 3800
plt.semilogx(model.alphas_ + EPSILON, model.mse_path_, ":")
plt.plot(
    model.alphas_ + EPSILON,
    model.mse_path_.mean(axis=-1),
    "k",
    label="Average across the folds",
    linewidth=2,
)
plt.axvline(
    model.alpha_ + EPSILON, linestyle="--", color="k", label="alpha: CV estimate"
)
```

all these colored dotted lines for each test fold

all alphas_

adds vertical line

the best alpha



17

Least Angle Regression (LAR)

If 20 fold:

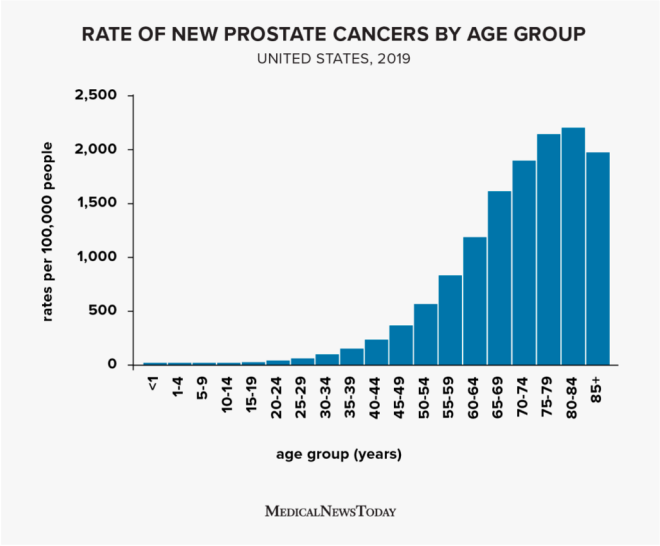
```
from sklearn.linear_model import LassoLarsCV, LassoCV

l1 = LassoLarsCV(cv=20, normalize=False).fit(X_train, Y_train)

# compute stats
# get mean mse for each fold
mean_mse = l1.mse_path_.mean(axis=-1)
# get standard error of mse for each fold
std_mse = l1.mse_path_.std(axis=-1)
# get best alpha
best_alpha_l1 = l1.alpha_
```

18

Feature Selection

19

Rate of Prostate Cancer



https://www.medicalnewstoday.com/articles/age-range-for-prostate-cancer

20

Example: Prostate Cancer Dataset

21

Term	LS	Ridge	Lasso
Intercept	2.465	2.452	2.468
lcavol	0.680	0.420	0.533
lweight	0.263	0.238	0.169
age	−0.141	−0.046	
lbph	0.210	0.162	0.002
svi	0.305	0.227	0.094
lcp	−0.288	0.000	
gleason	−0.021	0.040	
pgg45	0.267	0.133	

Task: predict logarithm of prostate specific antigen (PSA).

Best LASSO model learns to ignore several features (age, lcp, gleason, pgg45).

Wait…Is **age** really not a significant predictor of prostate cancer? What's going on here?

Age is highly correlated with other factors and thus *not significant* in the presence of those factors

21

Best-Subset Selection

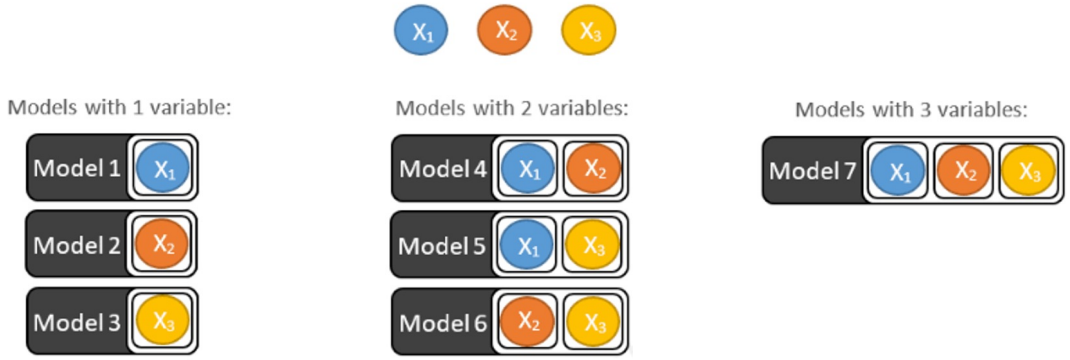
22

The optimal strategy for p features looks at models over *all possible combinations* of features,

```
For k in 1,…,p:
  subset = Compute all subset of k-features (p-choose-k)
  For kfeat in subset:
    model = Train model on kfeat features
    score = Evaluate model using cross-validation
Choose the model with best cross-validation score
```

22

Best-Subset Selection



23

Feature Selection: Prostate Cancer Dataset

24

Best subset works well!

reasonably good test error, low standard deviation, and only based on two features!

Term	LS	Best Subset	Ridge	Lasso
Intercept	2.465	2.477	2.452	2.468
lcavol	0.680	0.740	0.420	0.533
lweight	0.263	0.316	0.238	0.169
age	−0.141		−0.046	
lbph	0.210		0.162	0.002
svi	0.305		0.227	0.094
lcp	−0.288		0.000	
gleason	−0.021		0.040	
pgg45	0.267		0.133	
Test Error	0.521	0.492	0.492	0.479
Std Error	0.179	0.143	0.165	0.164

[Source: Hastie et al. (2001)]

24

Best-Subset Selection : Prostate Cancer Dataset 25

Time complexity

- Data have 8 features, there are 8-choose-k subsets for each k=1,…,8 for a total of 255 models
- Using 10-fold cross-val requires 10 x 255 = 2,550 training runs!
- In general, $O(2^p)$ time complexity

… who can afford exponential time complexity?

25

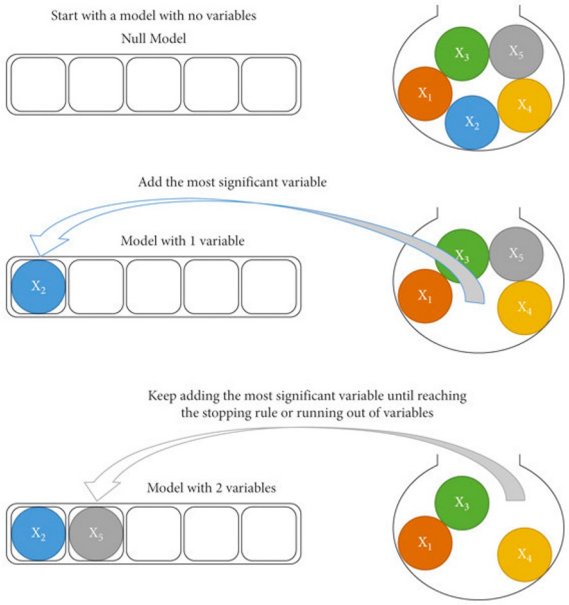
Forward Sequential Selection 26

An efficient method adds the most predictive feature one-by-one

```
featSel = empty
featUnsel = All features
For iter in 1,…,p:
  For kfeat in featUnsel:
   thisFeat = featSel + kfeat
    model = Train model on thisFeat features
    score = Evaluate model using cross-validation
  featSel = featSel + best scoring feature
  featUnsel = featUnsel – best scoring feature
Choose the model with best cross-validation score
```

26

Forward Sequential Selection



27

Backward Sequential Selection

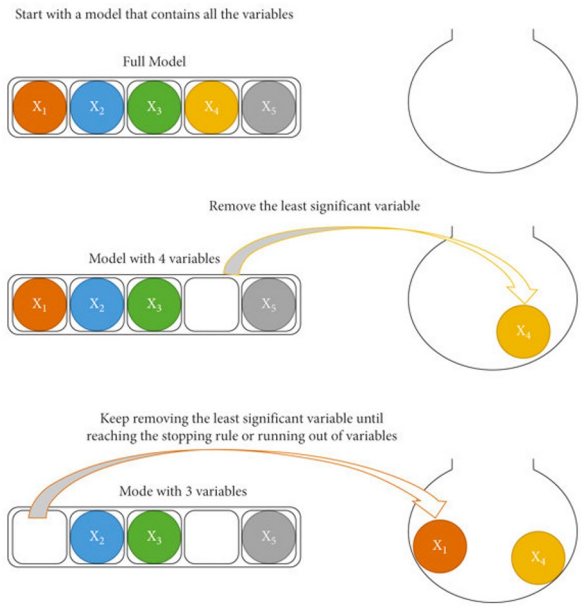
28

Backwards approach starts with *all* features and removes one-by-one

```
featSel = All features
For iter in 1,…,p:
  For kfeat in featSel:
    thisFeat = featSel - kfeat
     model = Train model on thisFeat features
     score = Evaluate model using cross-validation
  featSel = featSel – worst scoring feature
Choose the model with best cross-validation score
```

28

Backward Sequential Selection

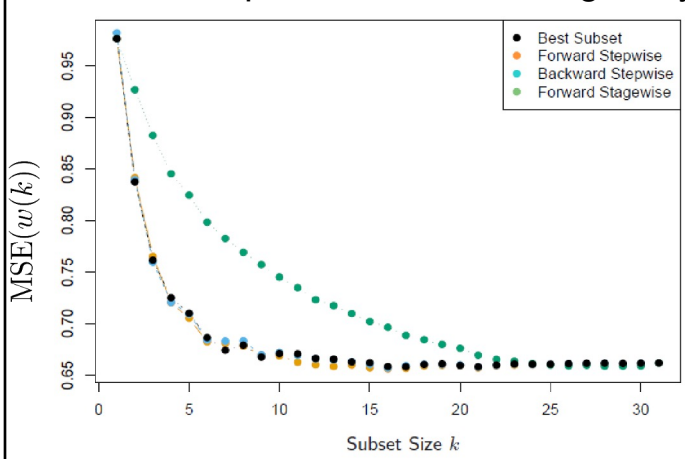


29

Comparing Feature Selection Methods

30

Sequential selection is greedy, but often performs well…



Example Feature selection on synthetic model with p=30 features with pairwise correlations (0.85). True feature weights are all zero except for 10 features, with weights drawn from N(0,6.25).

Sequential selection with p features takes $O(p^2)$ time, compared to exponential time for best subset

Sequential feature selection available in Scikit-Learn under:
`feature_selection.SequentialFeatureSelector`

30

General Principles of Regularization

31

- From the loss function point of view

$$\text{Model} = \min_{\text{model}} \text{Loss}(\text{Model}, \text{Data}) + \lambda \cdot \text{Regularizer}(\text{Model})$$

Regularization Strength **Regularization Penalty**

- We will see more examples of loss functions going forward.

31