MODEL SELECTION

Prof. Mehrdad Pirnia

This content is based on concepts from "Introduction to Statistical Learning" by James, Witten, Hastie, and Tibshirani.

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Lecture outcomes

By the end of this lecture, you will be able to:

- **Perform** Subset Selection methods to choose the important features of the models.
- **Perform** Shrinkage methods to reduce variance and help with variable selection.
- **Use** Python to do model selection

Three classes of methods

- **Subset Selection**. We identify a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.

- **Shrinkage**. We fit a model involving all p predictors, but the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance and can also perform variable selection.

- **Dimension Reduction**. We project the p predictors into a M-dimensional subspace, where M < p. This is achieved by computing M different linear combinations, or projections, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squares.

Subset Selection

1.Let *Mo* denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.

2. For $k = 1, 2, \ldots p$:

   1. (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   2. (b) Pick the best among these $\binom{p}{k}$ models, and call it M_k. Here *best* is defined as having the smallest *RSS*, or equivalently largest R^2.

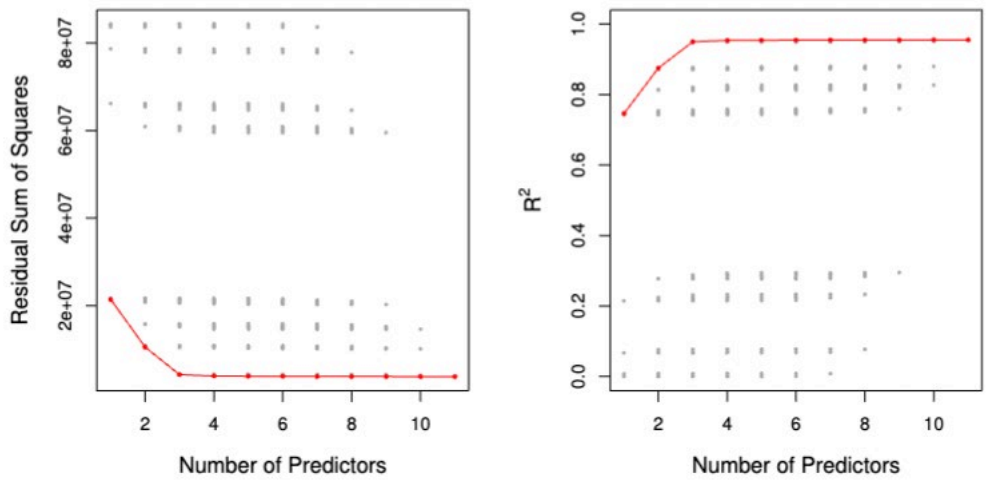
3. Select a single best model from among $M_0, \ldots, M_p$ using metrics such as cross validation.

Best Subset Selection

Finds the best model by evaluating all possible subsets of predictors.

```python
from itertools import combinations
import statsmodels.api as sm
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load dataset
data = pd.read_csv("dataset.csv")
X = data.drop(columns=["target"])
y = data["target"]

# Function to evaluate all subsets
def best_subset_selection(X, y):
    best_model = None
    best_score = float("inf")

    for k in range(1, len(X.columns) + 1):
        for subset in combinations(X.columns, k):
            X_subset = X[list(subset)]
            model = sm.OLS(y, sm.add_constant(X_subset)).fit()
            if model.aic < best_score:
                best_model, best_score = model, model.aic

    return best_model.summary()

print(best_subset_selection(X, y))
```

Example - Credit data set



For each possible model containing a subset of the ten predictors in the Credit data set, the RSS and R^2 are displayed. The red frontier tracks the best model for a given number of predictors, according to RSS and R^2. Though the data set contains only ten predictors, the x-axis ranges from 1 to 11, since one of the variables is categorical and takes on three values, leading to the creation of two dummy variables.

Extensions to other models

• Although we have presented best subset selection here for least squares regression, the same ideas apply to other types of models, such as logistic regression.

• The deviance — negative two times the maximized log-likelihood — plays the role of RSS for a broader class of models.

Stepwise Selection

• For computational reasons, best subset selection cannot be applied with very large p. Why not?

• Best subset selection may also suffer from statistical problems when p is large: larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data.

• Thus an enormous search space can lead to overfitting and high variance of the coefficient estimates.

• For both of these reasons, stepwise methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

Forward Stepwise Selection

• Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model.

• In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model.

In Detail
Forward Stepwise Selection

Let M_0 denote the null model, which contains no predictors.

For $k = 0, \ldots, p - 1$:
1. Consider all $p - k$ models that augment the predictors in M_k with one additional predictor.
2. Choose the best among these $p - k$ models, and call it M_{k+1}. Here best is defined as having the smallest RSS or highest R^2.

Select a single best model from among $M_0, \ldots, M_p$ using metrics such as cross-validation.

More on Forward Stepwise Selection

Computational advantage over best subset selection is clear.
It is not guaranteed to find the best possible model out of all 2^p models containing subsets of the p predictors.
Why not?

Forward Stepwise Selection

Starts with no predictors and adds the most significant one at each step.

```
def forward_selection(X, y):
    selected_features = []
    remaining_features = list(X.columns)
    best_score = float("inf")

    while remaining_features:
        scores = {}
        for feature in remaining_features:
            model = LinearRegression().fit(X[selected_features + [feature]], y)
            score = mean_squared_error(y, model.predict(X[selected_features + [feature]]))
            scores[feature] = score

        best_feature = min(scores, key=scores.get)
        if scores[best_feature] < best_score:
            selected_features.append(best_feature)
            remaining_features.remove(best_feature)
            best_score = scores[best_feature]
        else:
            break

    return selected_features

print(forward_selection(X, y))
```

Credit Data Example

# Variables	Best subset	Forward stepwise
One	rating	rating
Two	rating, income	rating, income
Three	rating, income, student	rating, income, student
Four	cards, income	rating, income,
	student, limit	student, limit

The first four selected models for best subset selection and forward stepwise selection on the Credit data set. The first three models are identical, but the fourth models differ.

Backward Stepwise Selection

Like forward stepwise selection, backward stepwise selection provides an efficient alternative to best subset selection.

However, unlike forward stepwise selection, it begins with the full least squares model containing all p predictors and then iteratively removes the least useful predictor, one at a time.

Backward Stepwise Selection: Details

Let M_p denote the full model, which contains all p predictors.

For k = p, p - 1, ..., 1:
1. Consider all k models that contain all but one of the predictors in M_k, for a total of k - 1 predictors.
2. Choose the best among these k models, and call it M_{k-1}. Here best is defined as having the smallest RSS or highest R^2.
Select a single best model from among M_0, ..., M_p using cross validation.

More on Backward Stepwise Selection

Like forward stepwise selection, the backward selection approach searches through only $1 + p(p + 1)/2$ models and so can be applied in settings where p is too large to apply best subset selection.

Like forward stepwise selection, backward stepwise selection is not guaranteed to yield the best model containing a subset of the p predictors.

Backward selection requires that the number of samples n is larger than the number of variables p (so that the full model can be fit).

In contrast, forward stepwise can be used even when $n < p$, and so it is the only viable subset method when p is very large.

Backward Stepwise Selection

Starts with all predictors and removes the least significant one at each step.

```
def backward_selection(X, y):
    selected_features = list(X.columns)
    best_score = float("inf")

    while len(selected_features) > 1:
        scores = {}
        for feature in selected_features:
            temp_features = selected_features.copy()
            temp_features.remove(feature)
            model = LinearRegression().fit(X[temp_features], y)
            score = mean_squared_error(y, model.predict(X[temp_features]))
            scores[feature] = score

        worst_feature = max(scores, key=scores.get)
        if scores[worst_feature] < best_score:
            selected_features.remove(worst_feature)
            best_score = scores[worst_feature]
        else:
            break

    return selected_features

print(backward_selection(X, y))
```

Choosing the Optimal Model

The model containing all of the predictors will always have the smallest RSS and the largest R^2, since these quantities are related to the training error.
We wish to choose a model with low test error, not a model with low training error.
Recall that training error is usually a poor estimate of test error.
Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

Shrinkage Methods

Ridge regression and *Lasso*

- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all p predictors using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero.
- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance.

Ridge Regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize:

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \, x_{ij} \right)^2$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \, x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

Ridge Regression: Continued

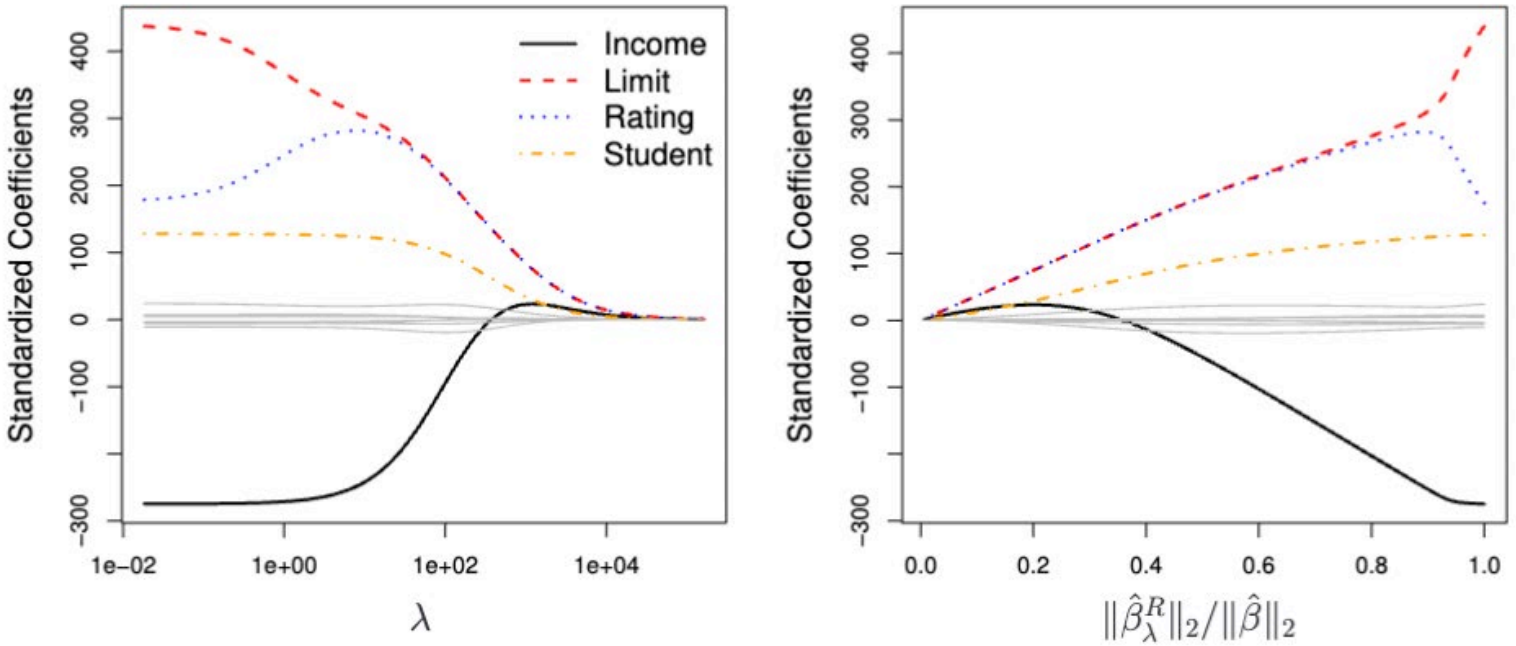
- As with least squares, ridge regression seeks coefficient estimates that fit the data well by making the RSS small.
- However, the second term, $\lambda \sum_j \beta_j^2$, called a *shrinkage penalty*, is small when $\beta_1, \ldots, \beta_p$ are close to zero, and so it has the effect of *shrinking* the estimates of β_j towards zero.
- The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates.
- Selecting a good value for λ is critical; cross-validation is used for this.

Ridge Regression

Ridge regression applies **L2 regularization**, which shrinks coefficients but does **not** set them to zero.

```
from sklearn.linear_model import Ridge

# Train Ridge model
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)

# Predict and evaluate
y_pred = ridge.predict(X_test)
print("MSE:", metrics.mean_squared_error(y_test, y_pred))

# Display coefficients
print(pd.DataFrame({'Feature': X.columns, 'Coefficient': ridge.coef_})
```

Credit data example



Details of Previous Figure

- In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of λ.
- The right-hand panel displays the same ridge coefficient estimates as the left-hand panel, but instead of displaying λ on the x-axis, we now display $\frac{||\hat{\beta}_\lambda^R||_2}{||\hat{\beta}||_2}$, where $\hat{\beta}$ denotes the vector of least squares coefficient estimates.
- The notation $||\ \beta\ ||_2$ denotes the ℓ_2 norm (pronounced "ell 2") of a vector, and is defined as:

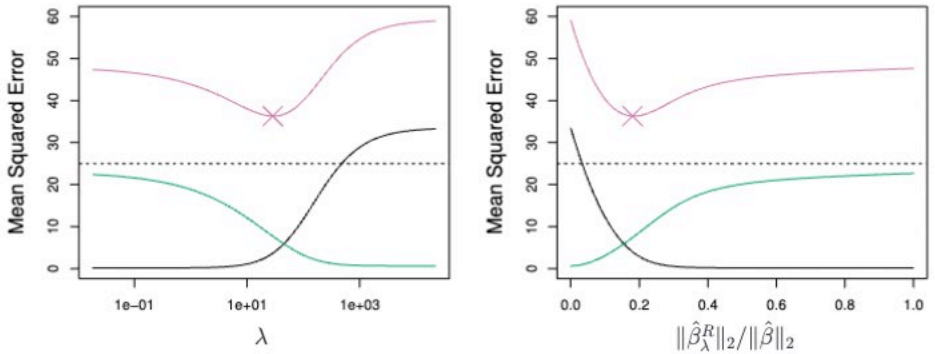
$$||\ \beta\ ||_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$$

Ridge Regression: Scaling of Predictors

- The standard least squares coefficient estimates are *scale equivariant*: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same.
- In contrast, the ridge regression coefficient estimates can change *substantially* when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- Therefore, it is best to apply ridge regression after *standardizing* the predictors, using the formula:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij} - \bar{x}_j\right)^2}}$$

Why Does Ridge Regression Improve Over Least Squares?



The Bias-Variance Tradeoff

- Simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients.
- Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of λ and $\frac{||\hat{\beta}^R_\lambda||_2}{||\hat{\beta}||_2}$.
- The horizontal dashed lines indicate the minimum possible MSE.
- The purple crosses indicate the ridge regression models for which the MSE is smallest.

The Lasso

- Ridge regression does have one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model.
- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j\, x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\,\beta_j\,|$$

- In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by:

$$||\,\beta\,||_1 = \sum_{j=1}^{p}|\,\beta_j\,|$$

The Lasso: Continued

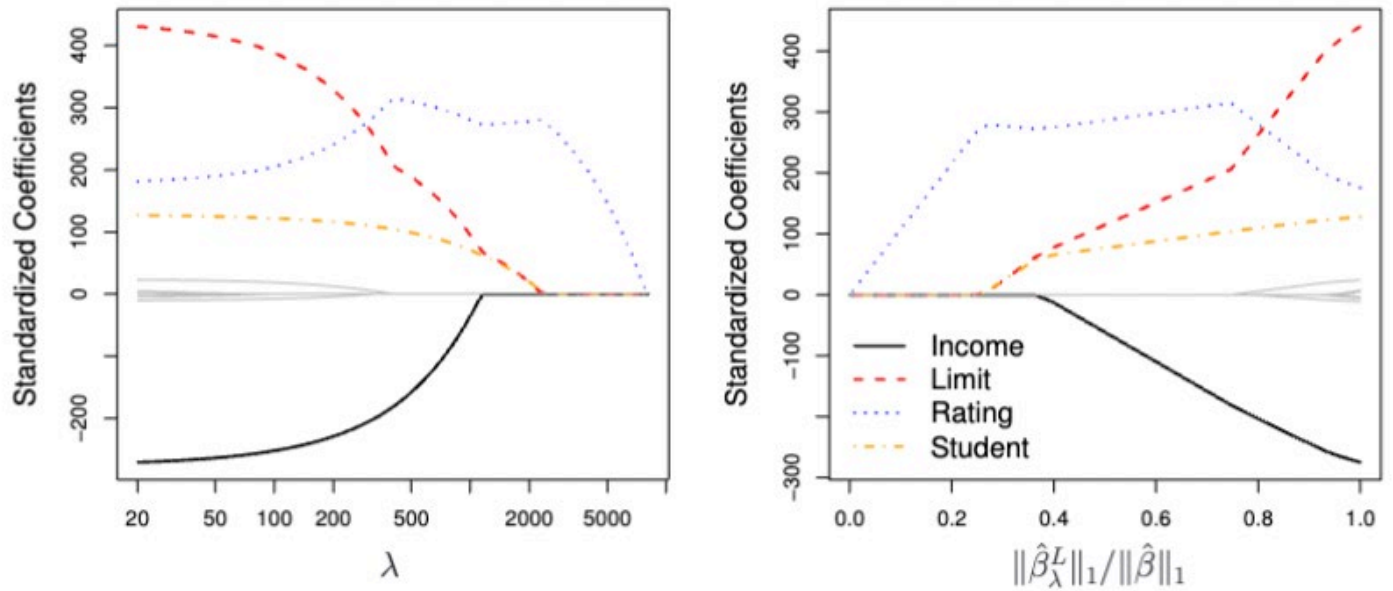
- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, much like best subset selection, the lasso performs *variable selection*.
- We say that the lasso yields *sparse* models — that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of λ for the lasso is critical; cross-validation is again the method of choice.

Lasso Regression Example

Lasso regression applies **L1 regularization**, which can shrink some coefficients to zero, effectively performing feature selection.

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pandas as pd

# Load dataset
advertising = pd.read_csv('advertising.csv', usecols=[1,2,3,4])
X = advertising.iloc[:, :-1]
y = advertising.iloc[:, -1]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train Lasso model
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)

# Predict and evaluate
y_pred = lasso.predict(X_test)
print("MSE:", metrics.mean_squared_error(y_test, y_pred))

# Display coefficients
print(pd.DataFrame({'Feature': X.columns, 'Coefficient': lasso.coef_}))
```

Credit data example



The Variable Selection Property of the Lasso

Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero?

One can show that the lasso and ridge regression coefficient estimates solve the problems:

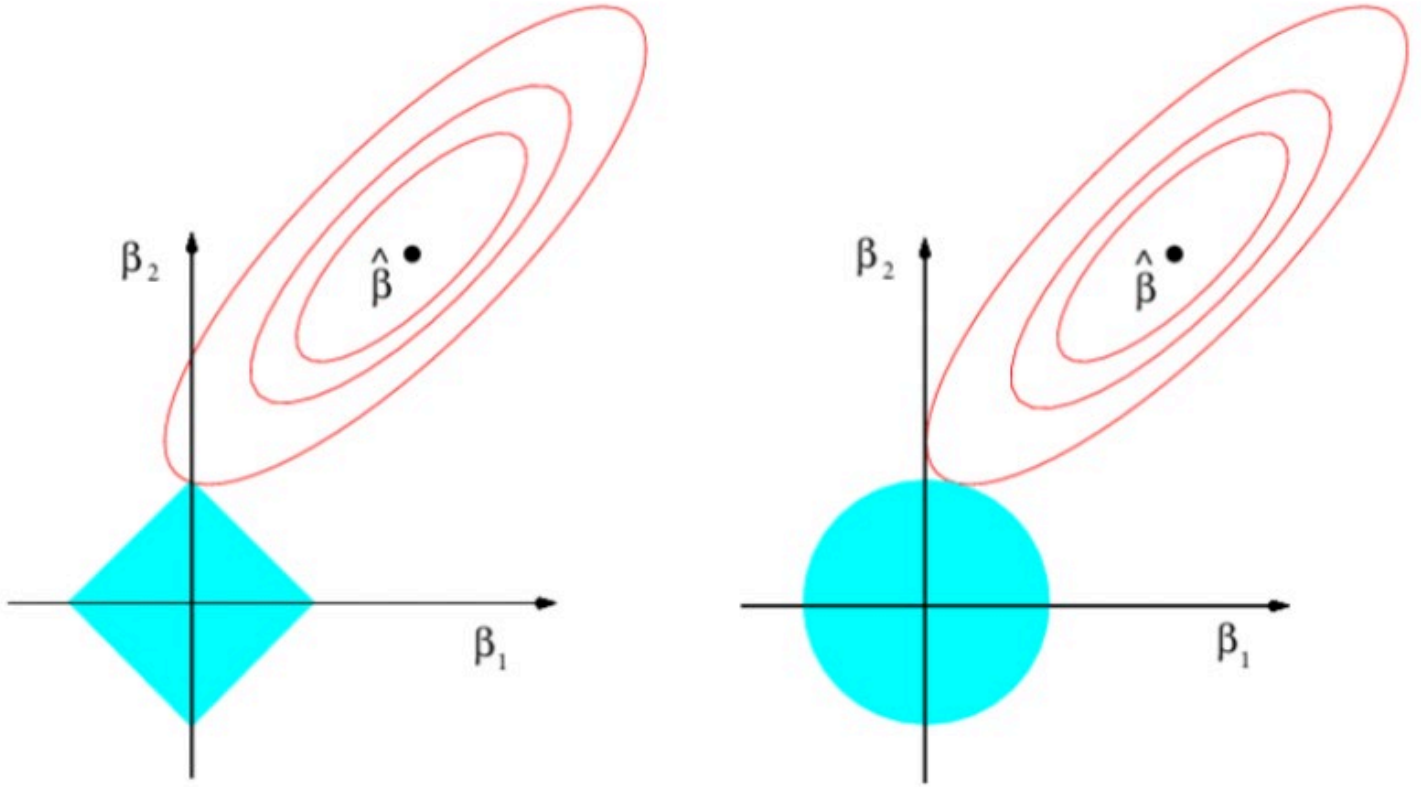
$$\text{minimize}_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

and

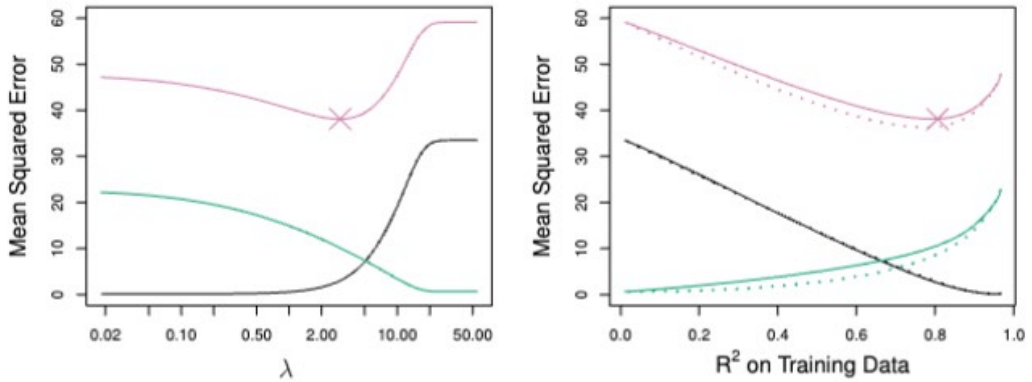
$$\text{minimize}_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

respectively.

The Lasso Picture



Comparing the Lasso and Ridge Regression



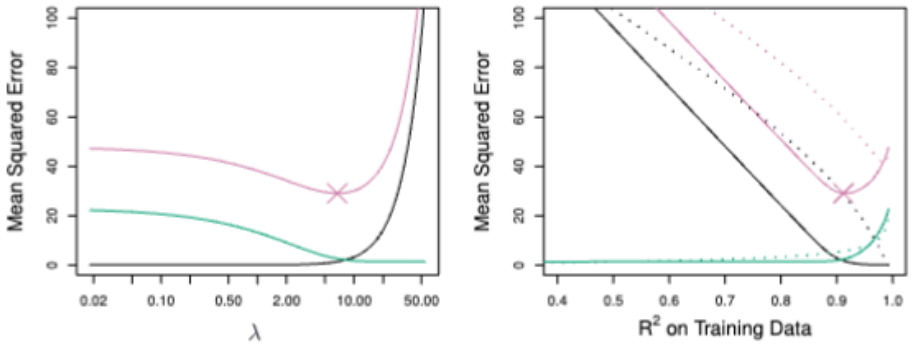
Left:

Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on a simulated data set.

Right:

Comparison of squared bias, variance, and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

Comparing the Lasso and Ridge Regression: Continued



Left:

Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to the previous example, except that now only two predictors are related to the response.

Right:

Comparison of squared bias, variance, and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

Lasso vs Ridge Regression: Example

Question:

Lasso regression applies **L1 regularization**, while Ridge regression applies **L2 regularization**.

Given a simple linear regression model:

$$y = 3x_1 + 2x_2 + \epsilon$$

Assume that after applying **Lasso regression**, the coefficient of x_2 shrinks to **zero**. How does this affect **model interpretability** compared to **Ridge regression**, where coefficients only **shrink** but do **not become zero**?

- **Lasso regression** removes x_2 by setting its coefficient to **zero**, simplifying the model to:

$$y = 3x_1 + \epsilon$$

Improves interpretability by eliminating less important features.

- **Ridge regression** shrinks but **keeps all features**, leading to:

$$y = 3x_1 + 0.8x_2 + \epsilon$$

Retains all predictors, but model is **less interpretable**.

Method	Effect on Coefficients	Interpretability
Lasso (L1)	Some coefficients become zero, removing features.	Simpler model, easier to interpret.
Ridge (L2)	Coefficients shrink but never reach zero.	More complex, retains all features.

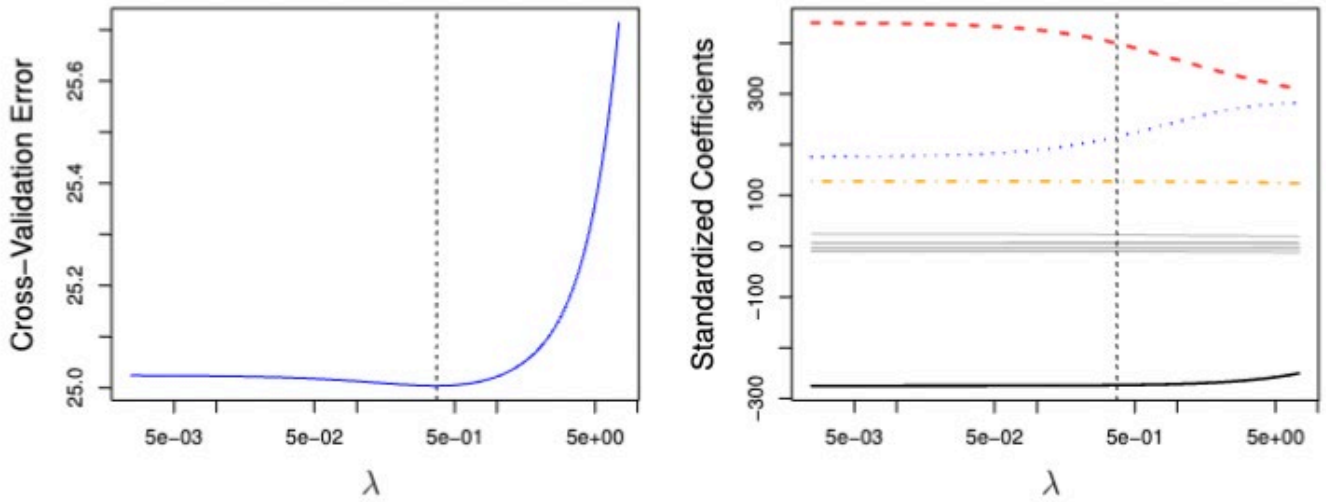
Conclusions

- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better when the response is a function of only a relatively small number of predictors.
- However, the number of predictors that is related to the response is never known *a priori* for real data sets.
- A technique such as cross-validation can be used to determine which approach is better on a particular data set.

Selecting the Tuning Parameter for Ridge Regression and Lasso

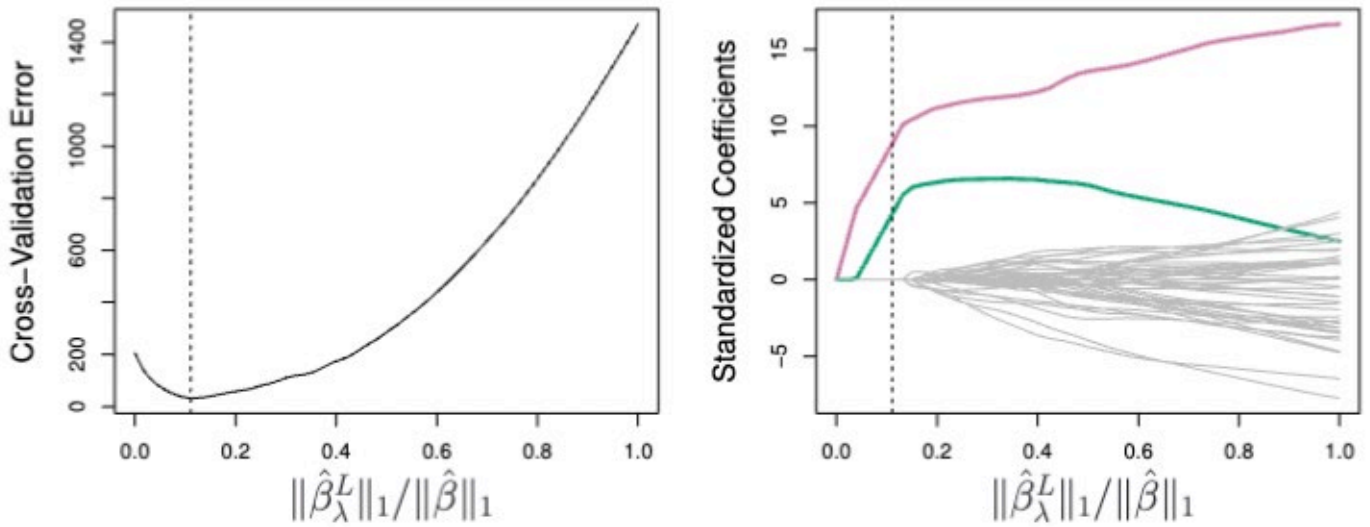
- As with subset selection, for ridge regression and lasso we require a method to determine which of the models under consideration is best.
- That is, we require a method for selecting a value for the tuning parameter λ or equivalently, the value of the constraint s.
- *Cross-validation* provides a simple way to tackle this problem. We choose a grid of λ values and compute the cross-validation error rate for each value.
- We then select the tuning parameter value for which the cross-validation error is smallest.
- Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter.

Credit Data Example



- **Left**: Cross-validation errors that result from applying ridge regression to the *Credit* data set with various values of λ.
- **Right**: The coefficient estimates as a function of λ. The vertical dashed lines indicate the value of λ selected by cross-validation.

Simulated Data Example



- **Left**: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set.
- **Right**: The corresponding lasso coefficient estimates are displayed. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

Dimension Reduction Methods

- The methods discussed so far have involved fitting linear regression models via least squares or a shrinkage approach, using the original predictors $X_1, X_2, \ldots, X_p$.
- We now explore a class of approaches that *transform* the predictors and then fit a least squares model using the transformed variables.
- These techniques are referred to as *dimension reduction* methods.

Dimension Reduction Methods: Details

- Let $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ *linear combinations* of our original p predictors. That is:

$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j$$

  for some constants $\phi_{m1}, \ldots, \phi_{mp}$.

- We then fit the linear regression model:

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \quad i = 1, \ldots, n$$

  using ordinary least squares.

- In model (2), the regression coefficients are given by $\theta_0, \theta_1, \ldots, \theta_M$.
- If the constants $\phi_{m1}, \ldots, \phi_{mp}$ are chosen wisely, then dimension reduction approaches can often outperform OLS regression.

Mathematical Reformulation of Dimension Reduction

- From definition (1),

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \sum_{j=1}^{p} \theta_m \phi_{mj} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{mj} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij}$$

  where

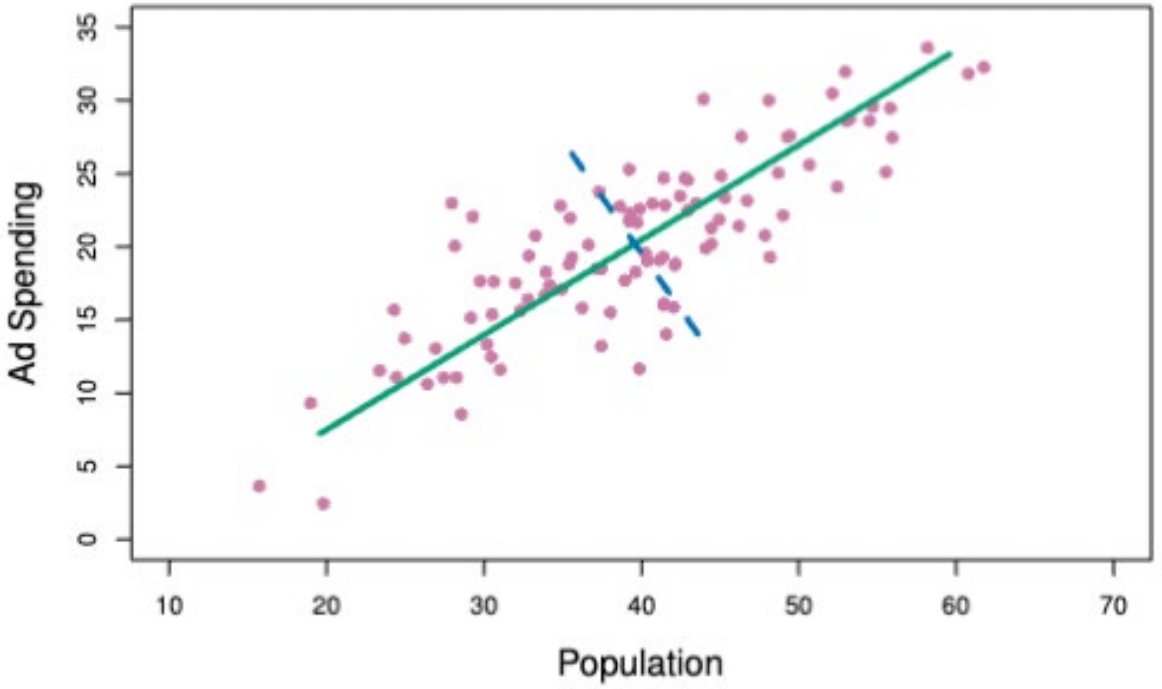
$$\beta_j = \sum_{m=1}^{M} \theta_m \phi_{mj}$$

- Hence, model (2) can be thought of as a special case of the original linear regression model.
- Dimension reduction constrains the estimated β_j coefficients, forcing them to take the form given above.
- It can provide advantages in the *bias-variance tradeoff*.

Principal Components Regression

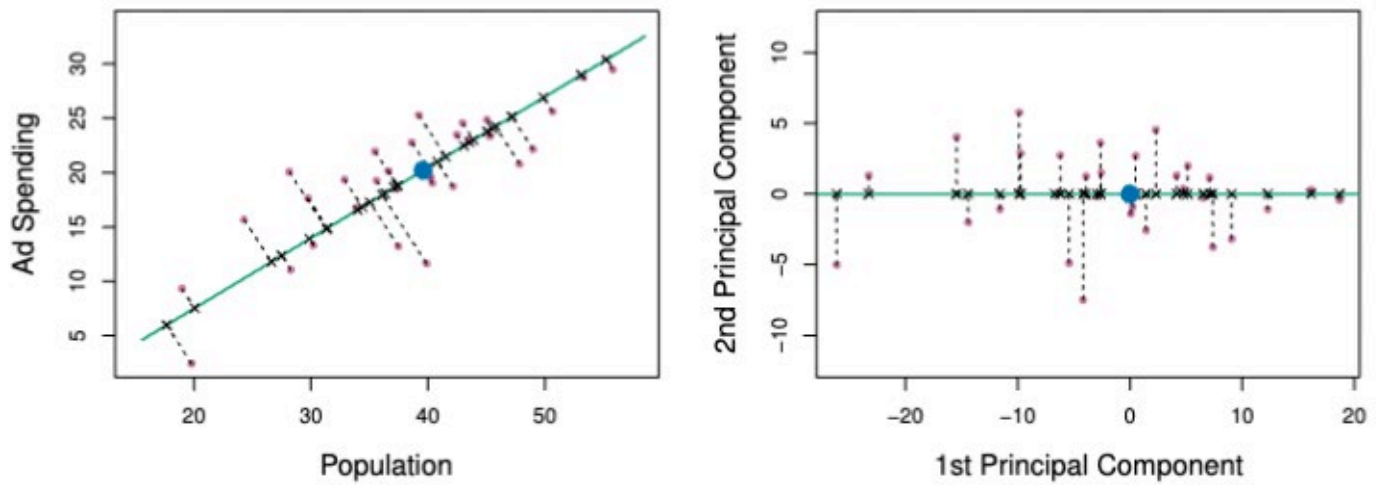
- Principal Components Analysis (PCA) is applied to define linear combinations of predictors for regression.
- The *first principal component* is the (normalized) linear combination of variables with the *largest variance*.
- The *second principal component* has the largest variance, subject to being uncorrelated with the first.
- This process continues iteratively.
- When dealing with highly correlated variables, PCA allows us to replace them with a small set of principal components that capture their joint variation.

Pictures of PCA



- The population size (*pop*) and ad spending (*ad*) for 100 different cities are shown as purple circles.
- The **green solid line** indicates the *first principal component*.
- The **blue dashed line** indicates the *second principal component*.

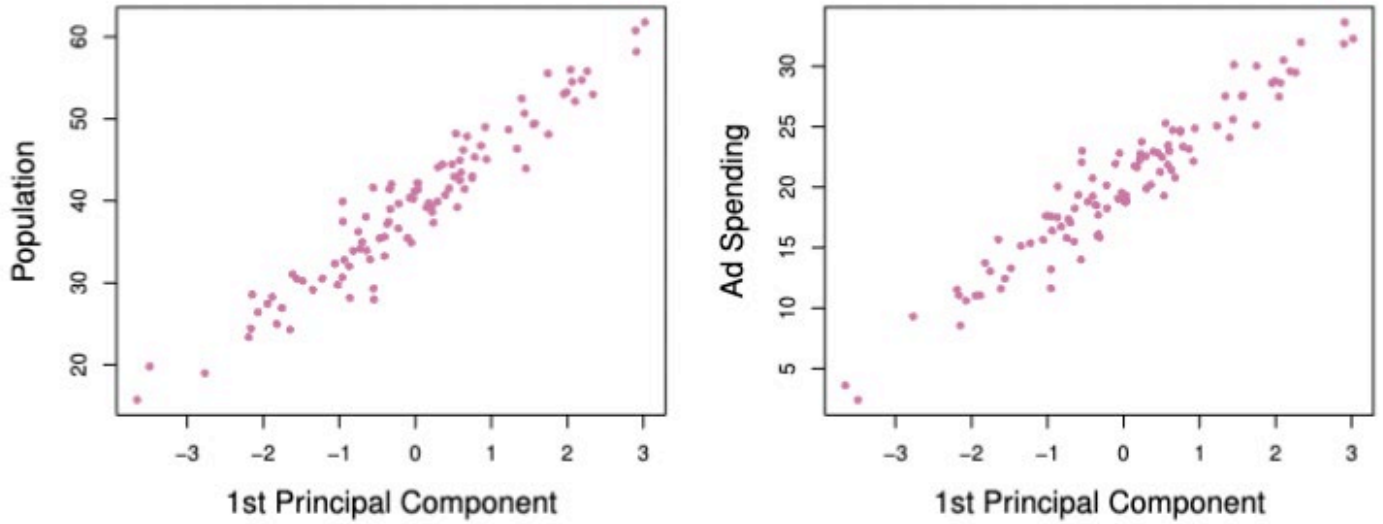
Pictures of PCA: Continued



A subset of the advertising data.

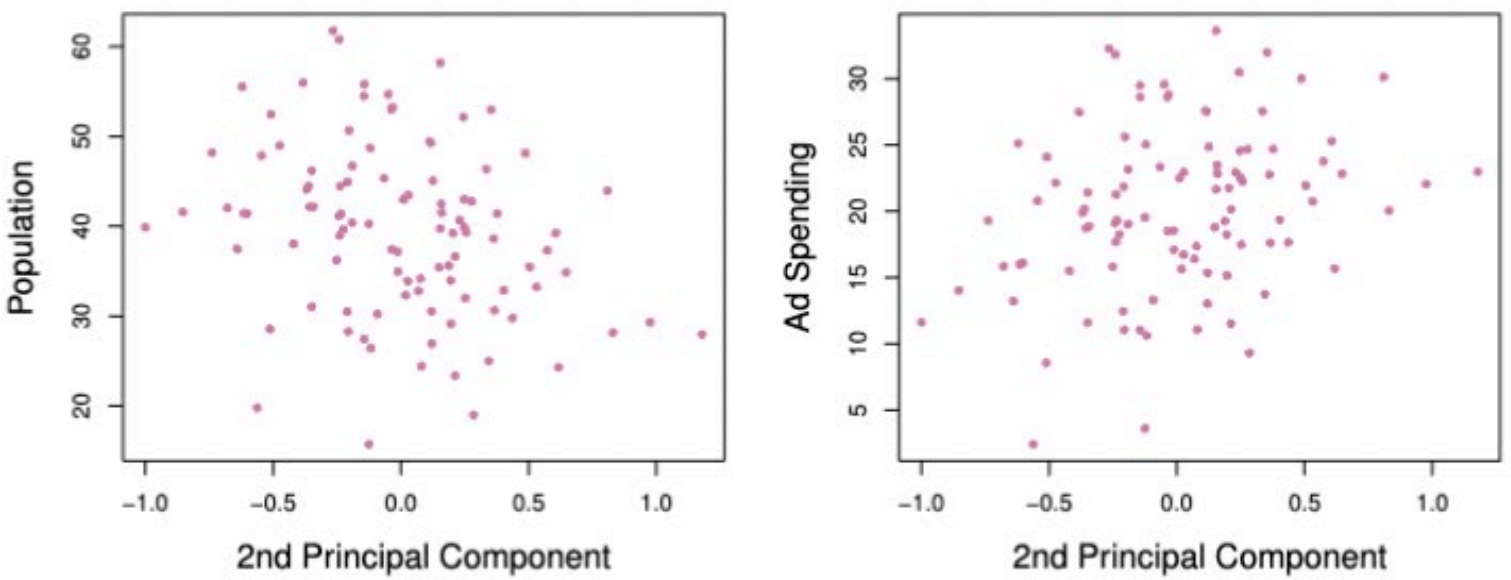
Left: The first principal component, chosen to minimize the sum of the squared perpendicular distances to each point, is shown in green.
These distances are represented using the black dashed line segments.
Right: The left-hand panel has been rotated so that the first principal component lies on the x-axis.

Pictures of PCA: Continued



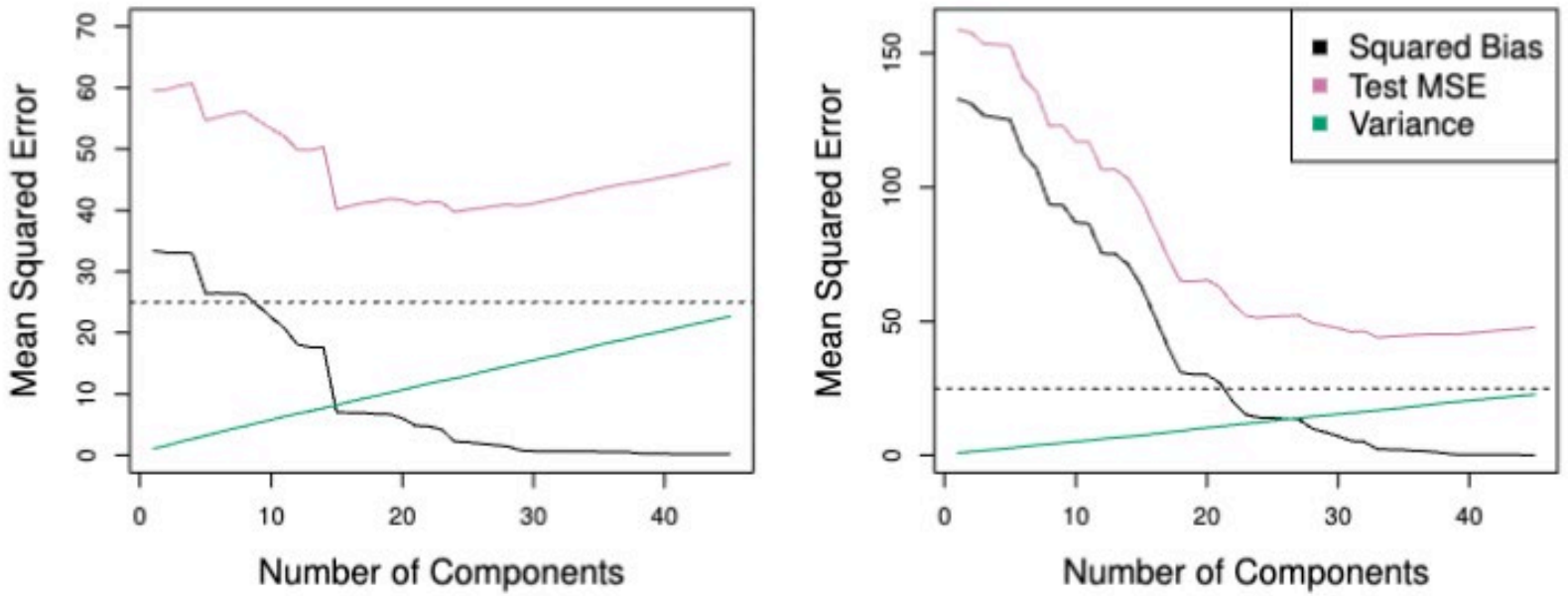
- **Plots of the first principal component scores** z_{i1} versus **pop** and **ad**.
  - The relationships are **strong**.

Pictures of PCA: Continued



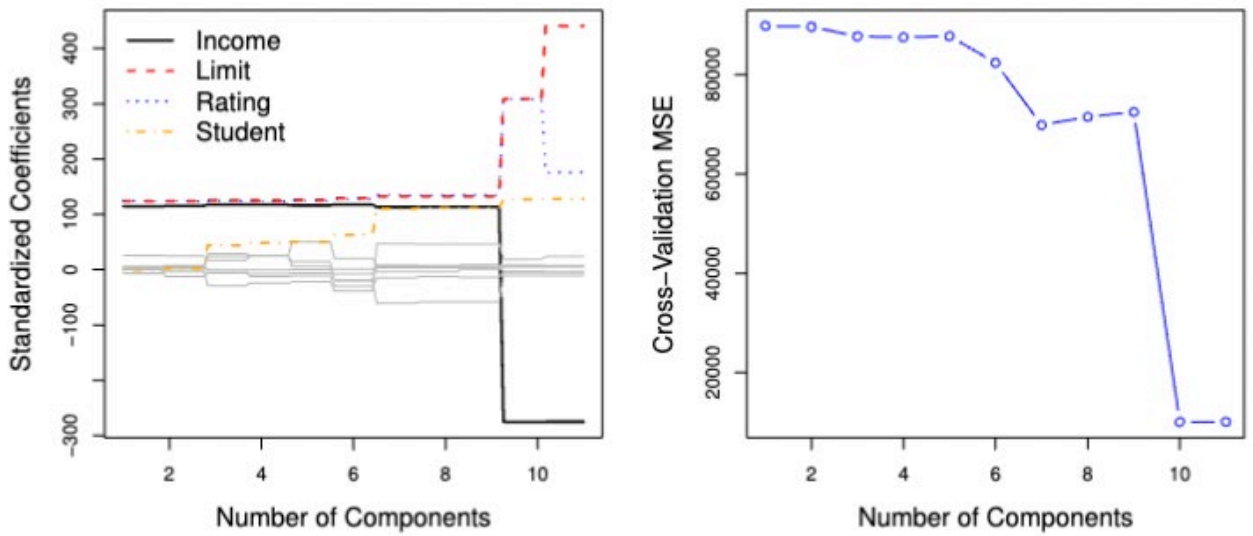
- **Plots of the second principal component scores** z_{i2} versus **pop** and **ad**.
  - The relationships are **weak**.

Application to Principal Components Regression



- **PCR was applied to two simulated data sets.**
  - The **black**, **green**, and **purple** lines correspond to: **Squared bias**, **Variance**, **Test mean squared error** (MSE)
  - **Left**: Simulated data from slide 32.
  - **Right**: Simulated data from slide 39.

Choosing the Number of Directions M



- **Left**: PCR standardized coefficient estimates on the *Credit* data set for different values of M.
- **Right**: The 10-fold cross-validation MSE obtained using PCR, as a function of M.

Partial Least Squares

- **PCR identifies** linear combinations, or *directions*, that best represent the predictors $X_1, \ldots, X_p$.
- These directions are identified in an **unsupervised** way, since the response Y is not used to determine the principal component directions.
- That is, the response **does not supervise** the identification of the principal components.
- Consequently, **PCR has a serious drawback**:
  - There is no guarantee that the directions that best explain the predictors will also be the best directions for predicting the response.

Partial Least Squares: Continued

- Like PCR, **PLS is a dimension reduction method**, which:
  - First identifies a new set of features $Z_1, \ldots, Z_M$ that are linear combinations of the original features.
  - Then fits a linear model via **OLS** using these M new features.
- **Difference between PCR and PLS:**
  - PCR identifies features in an **unsupervised way**.
  - PLS identifies features in a **supervised way**, meaning:
    - It uses the response Y to find features that **not only approximate the old features well** but are also **related to the response**.

PLS approach finds directions that explain **both the response and the predictors.**

Details of Partial Least Squares

- After **standardizing** the p predictors, **PLS computes the first direction** Z_1 by:
  - Setting each ϕ_{1j} equal to the coefficient from the simple linear regression of Y onto X_j.
- One can show that this coefficient is **proportional** to the correlation between Y and X_j.
- **Computation of** Z_1:

$$Z_1 = \sum_{j=1}^{p} \phi_{1j} X_j$$

  - PLS places the **highest weight** on variables **most strongly related** to the response.
- **Subsequent directions** are found by taking residuals and **repeating the procedure**.

Summary

- **Model selection methods** are essential for data analysis, particularly for large datasets with many predictors.
- **Research into sparsity methods** (e.g., *Lasso*) is a **hot area** in data science.
- Later, we will explore **sparsity methods** in more detail, including **Elastic Net**.

Right: The left-hand panel has been rotated so that the *first principal component* lies on the x-axis.


UNIVERSITY OF
WATERLOO
FACULTY OF ENGINEERING