UNIVERSITY OF SAN FRANCISCO

Master of Science in Analytics

Model Selection & Regularization

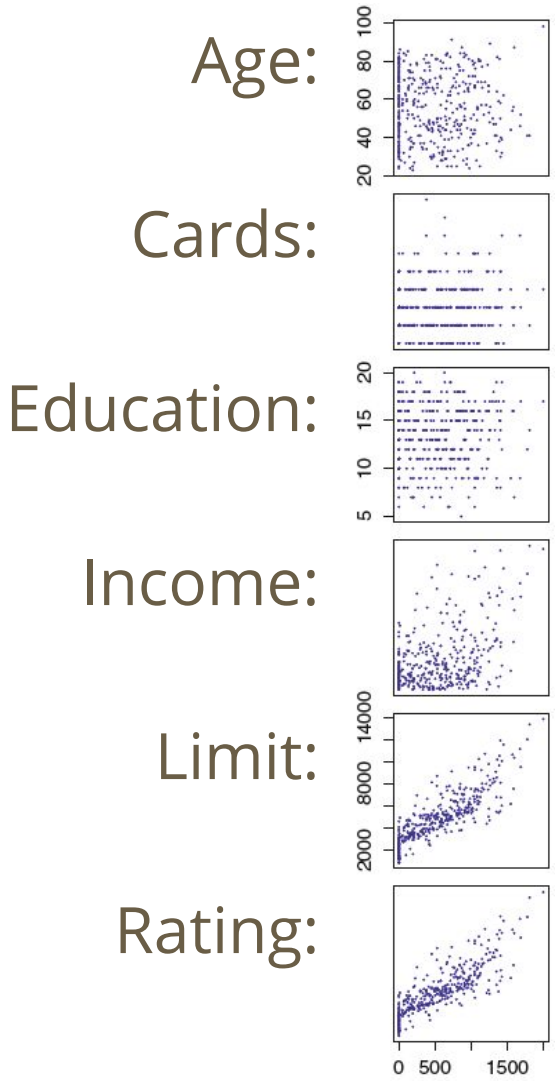
Machine Learning 1

Goal & definitions

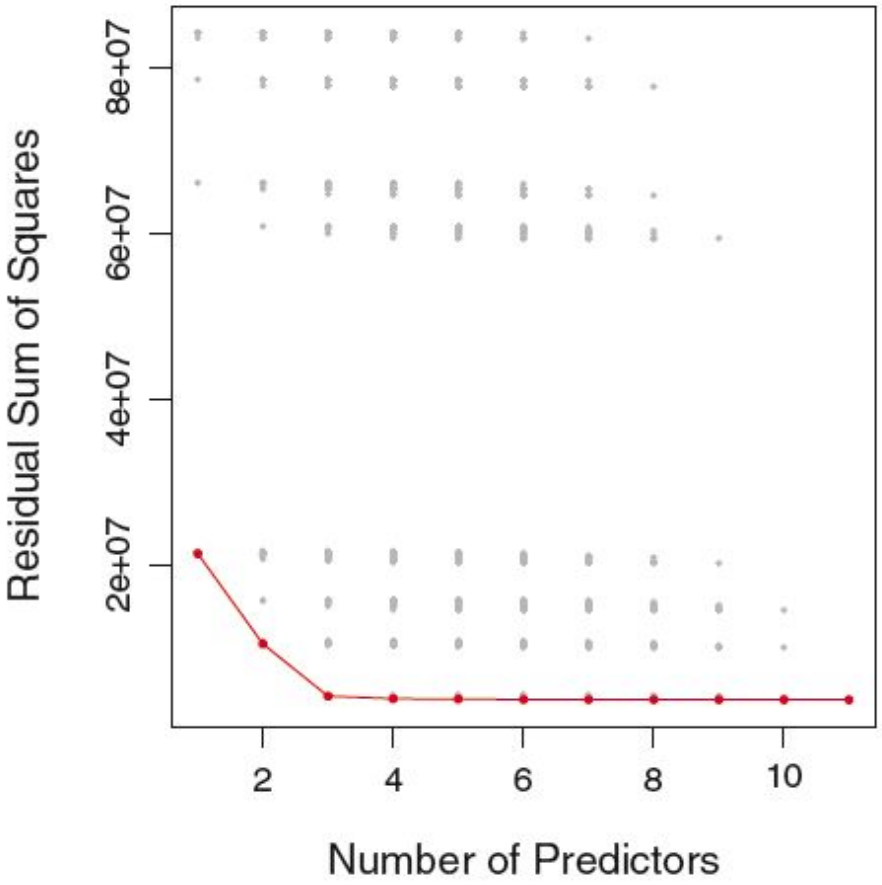
- Goals
  - Reduce the complexity [number of features] in a model
  - Why?
- Model Selection
  - Select optimal feature set for model
  - Useful for [linear] regression and classification
- Regularization
  - Shrink feature coefficients of least squares to (near) zero
  - In theory: reduces variance
- Dimension Reduction
  - Use fewer features by projecting into M-dimensional space

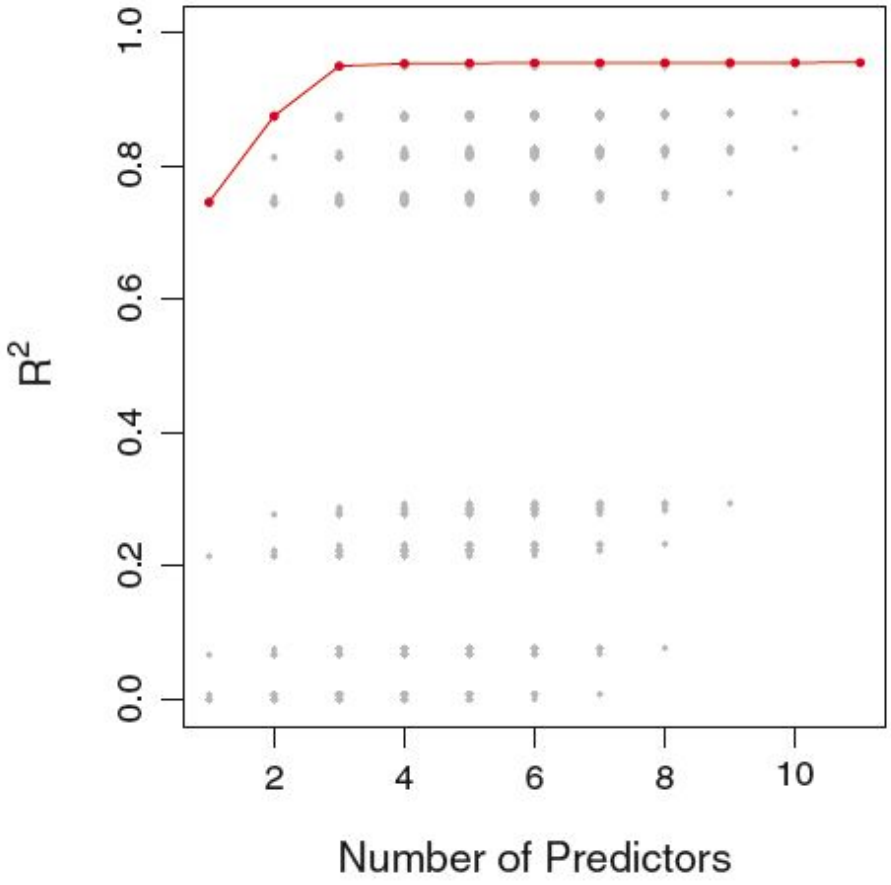
Example data - Credit.csv

credit
ID primary_key: INTEGER
income: DECIMAL(6,2)
limit: DECIMAL(6,2)
rating: INTEGER
cards: INTEGER
age: INTEGER
education: INTEGER
gender: VARCHAR(5)
student: BOOLEAN
married: BOOLEAN
ehnicity: VARCHAR(20)
balance: DECIMAL(5,2)



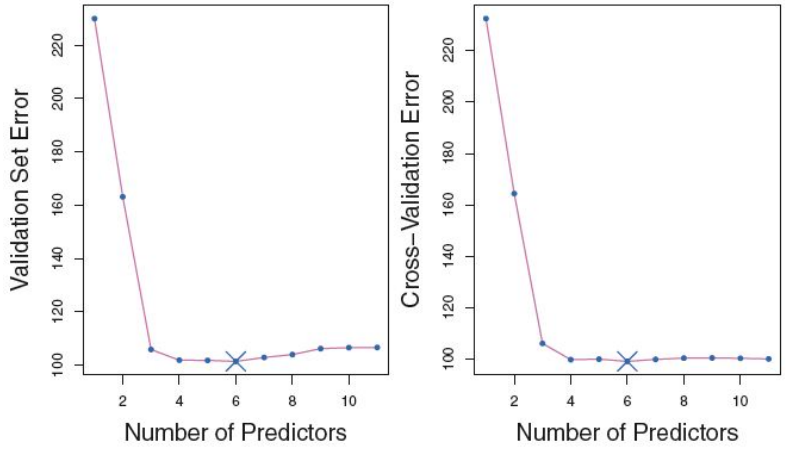
Predictors vs. errors





Evaluating models

- Problem:
  - Models with more features always* have lower RSS, higher R^2
  - Cannot compare models with difference in numbers of features
  - Must determine test error across models
- Estimating test error
  - Directly: use validation set / cross validation



  - Indirectly: make adjustment to training error, account for overfitting

Evaluating models - C_p

- Estimate:

$$C_p = \frac{1}{n}\left(\mathrm{RSS} + 2d\hat{\sigma}^2\right)$$

  - ... where *d* = number of features in model
- What it's doing
  - Adds $2d\sigma^2$ penalty to training set RSS (i.e. lower C_p is better score)
  - Accounts for higher test set error?

Evaluating models - AIC

- Akaike information criterion
- Estimate:

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}\left(\mathrm{RSS} + 2d\hat{\sigma}^2\right)$$

- What it's doing
  - Also adds "additional features" penalty
  - Equal to C_p for (Gaussian) linear regression
- Comparison (example)
  - Can compare two AIC models, x & y, using exp((x-y)/2)
  - Example: aic(model 1) = 61; aic(model 2) = 65
  - Model 1 is 54.6 more probable than model 2 to be a better model

Evaluating models - BIC

- Bayesian information criterion
- Estimate:

$$\mathrm{BIC} = \frac{1}{n}\left(\mathrm{RSS} + \log(n) d \hat{\sigma}^2\right)$$

  - ... where *n* is number of observations
- What it's doing
  - Also adds a large "additional features" penalty, based also on *n*
  - Tends to create smaller models than C_p

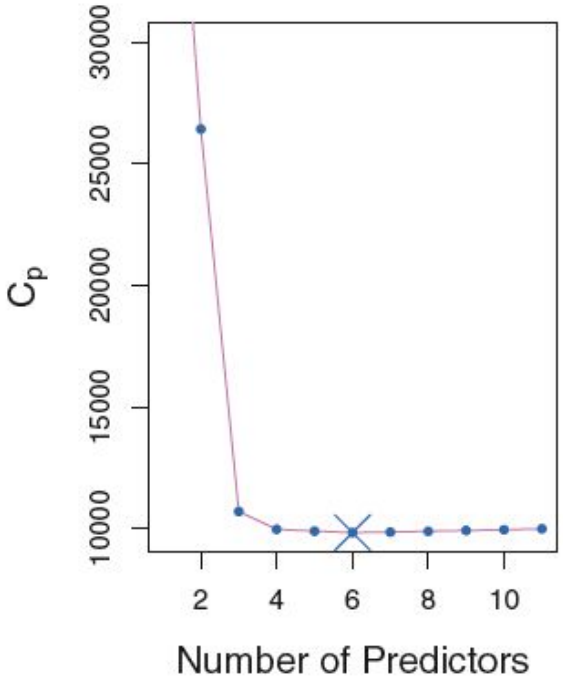
Evaluating models - Adjusted R^2

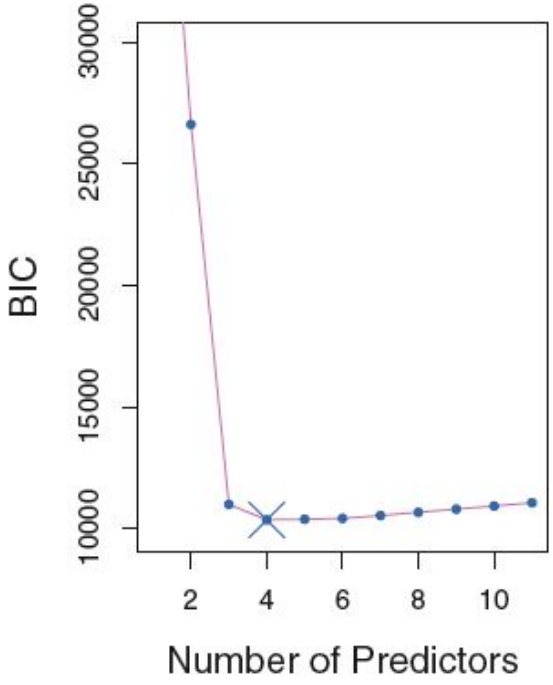
- Recall R^2
  - R^2 = 1 - RSS/TSS
  - RSS always decreases with more features, so R^2 always increases
- Estimate:

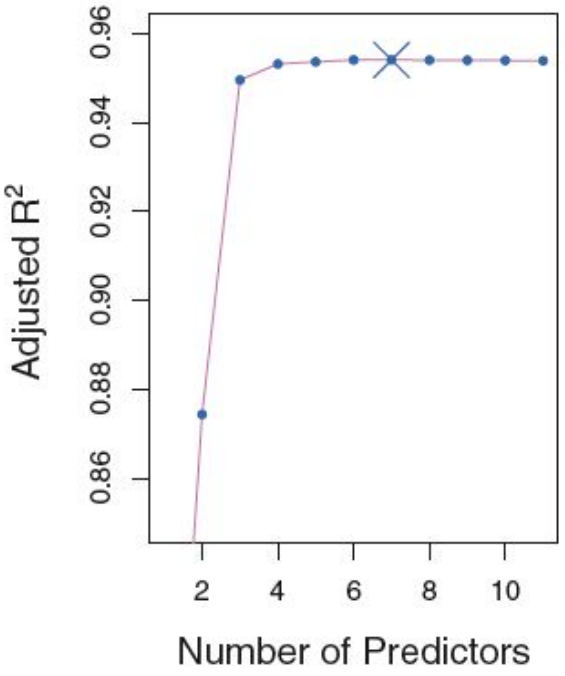
$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

  - ... where *n* is number of observations
- Intuition
  - Adding features with high variance increases *d*
  - i.e. also adds an “additional features” penalty
  - Best possible score is 1.0

Comparison (credit.csv)







Implementation in scikit-learn

1) Import data
   a) If necessary, split data into train, test sets
2) Coerce data into:
   a) X = List-of-lists / numpy matrix: all features
   b) Y = List / numpy vector: all targets

```
# 3a) BIC / AIC

from sklearn.linear_model import LassoLarsIC

model = LassoLarsIC(criterion='bic')    # ... or 'aic'
model.fit(X, Y)
alpha = model.alpha_
```

```
# 3b) R2
# Needs hypotheses for Y

from sklearn.metrics import r2_score

r2 = r2_score(Y, hypotheses)
```

4) Compare the result with other models

Subset selection

- Fit a model for each subset of features
- Algorithm:

$\mathcal{M}_0 \leftarrow \varnothing$
For k = 1, 2, …, p:
    Fit (p choose k) models, each with k predictors
    $\mathcal{M}_k \leftarrow$ model with lowest RSS (highest R^2?)
Output model from [$\mathcal{M}_0$ - $\mathcal{M}_p$] with lowest error

- This approach:
  - Uses cross-validation to avoid training set bias
  - Is impractical: running time comparable to SAT

Forward stepwise selection

- One tractable variation of subset selection
- Algorithm:

$\mathcal{M}_0 \leftarrow \varnothing$
For k = 0, 1, …, p - 1:
    Consider adding best predictor to each model $\mathcal{M}_k$
    Choose best among $p - k$ models (lowest RSS?)
Output model from [$\mathcal{M}_0$ - $\mathcal{M}_p$] with lowest error

- Problem: does not account for feature combinations

# 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

Backward stepwise selection

- Another tractable variation of subset selection
- Algorithm:

$\mathcal{M}_p$ ← all features
For k = p, p-1, …, 1:
    Consider *k* models with 1 less feature than $\mathcal{M}_{k-1}$
    Choose best among *k* models (lowest RSS?)
Output model from [$\mathcal{M}_0$ - $\mathcal{M}_p$] with lowest error

- Problem: fails when n < p

Regularization

- Recall:
  - Regularization / shrinkage: may establish low (zero?) coefficients
  - Effect: decreases variance for noisy features
  - Good in high-dimensional settings
- Techniques:
  - Ridge regression
  - The lasso

Ridge regression

- Also known as Tikhonov regularization (β^R)
- Evaluation

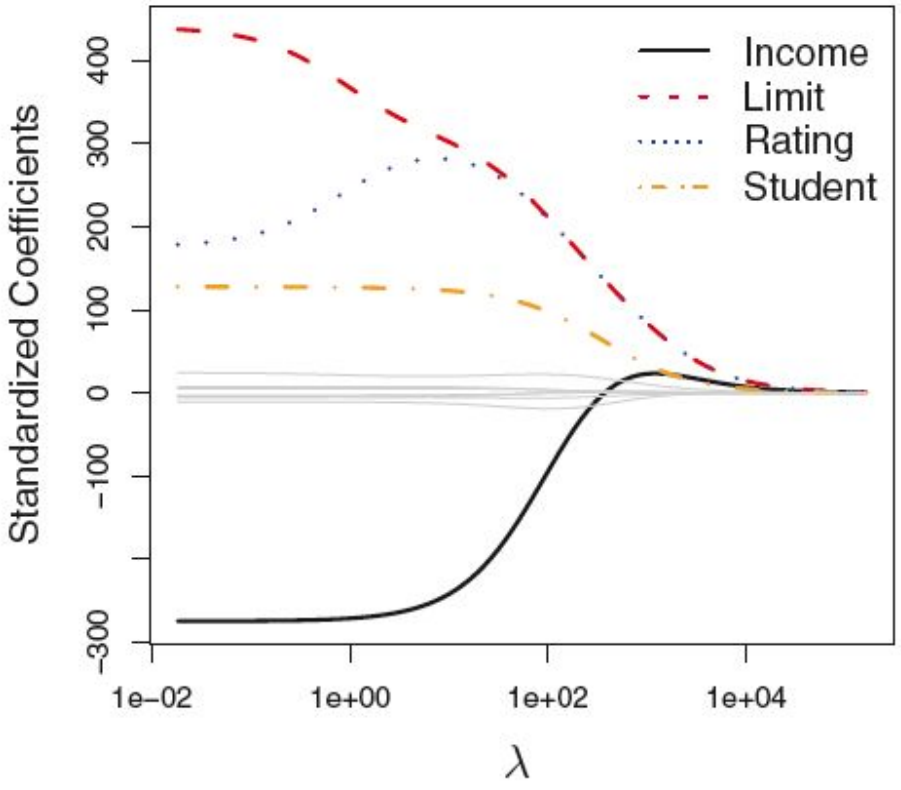
$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

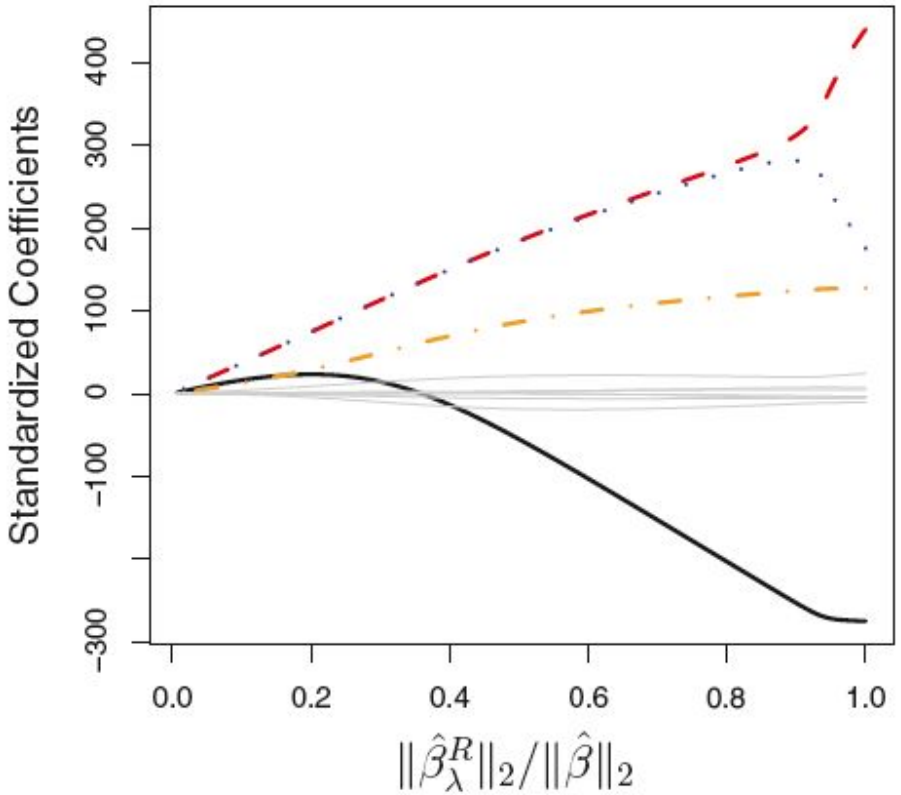
- Recall

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

- What it's doing:
  - Adds a shrinkage penalty (to linear regression) when β_j is small
  - When $\lambda = 0$, no penalty

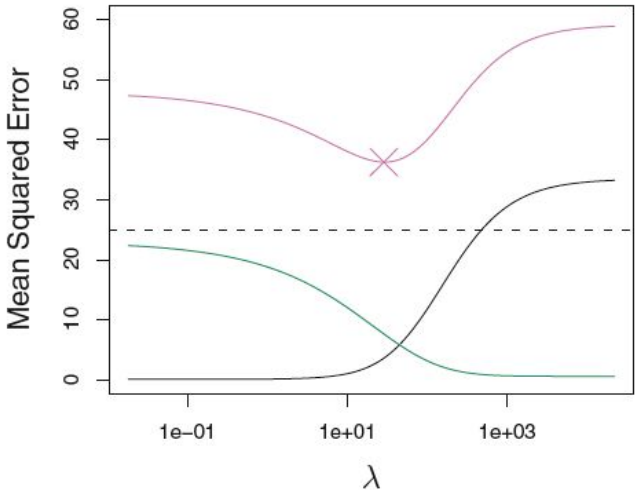
Ridge regression vs. credit

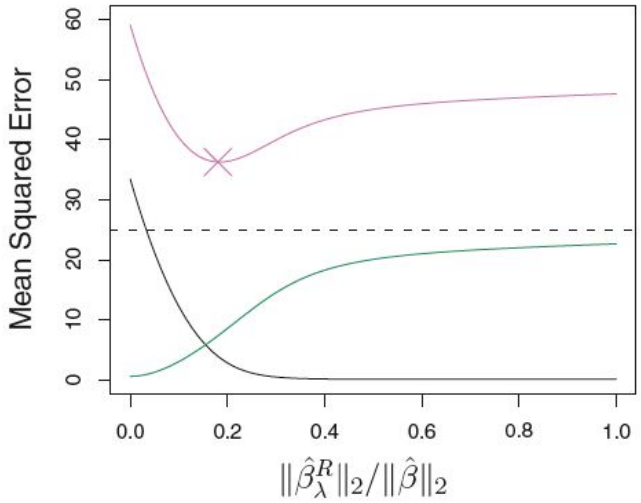




Why does ridge regression work?

- Bias-variance trade-off
  - Increasing λ decreases flexibility of ridge regression
  - Alternately: bias increases, variance decreases
- We can find optimal λ by looking at MSE





Simulated dataset

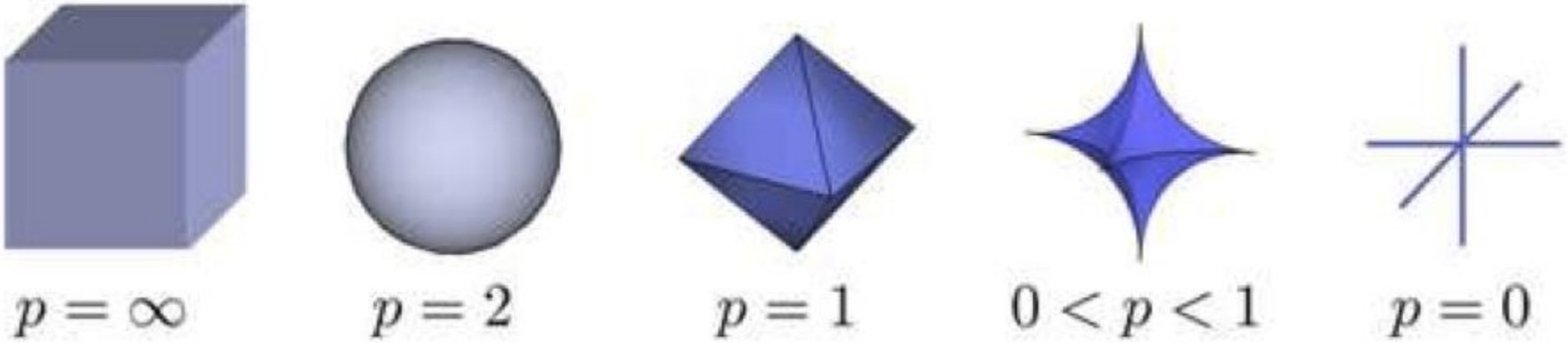
Black: squared bias
Green: variance
Purple: test MSE
Dashed line: min MSE

The lasso

- Evaluation

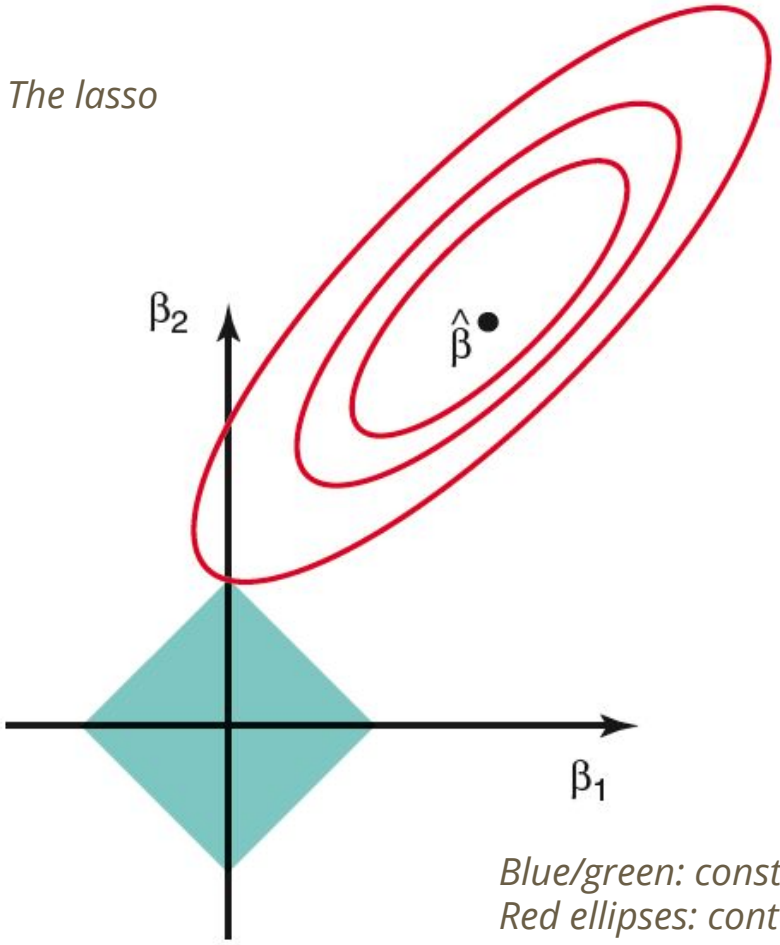
$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

  - ... in other words, similar to ridge regression
  - Difference is penalty ($|\beta_j|$) —> ℓ_1 penalty instead of ℓ_2 penalty
- Comparisons to the ball (cf. L1 vs. L2 on quora):

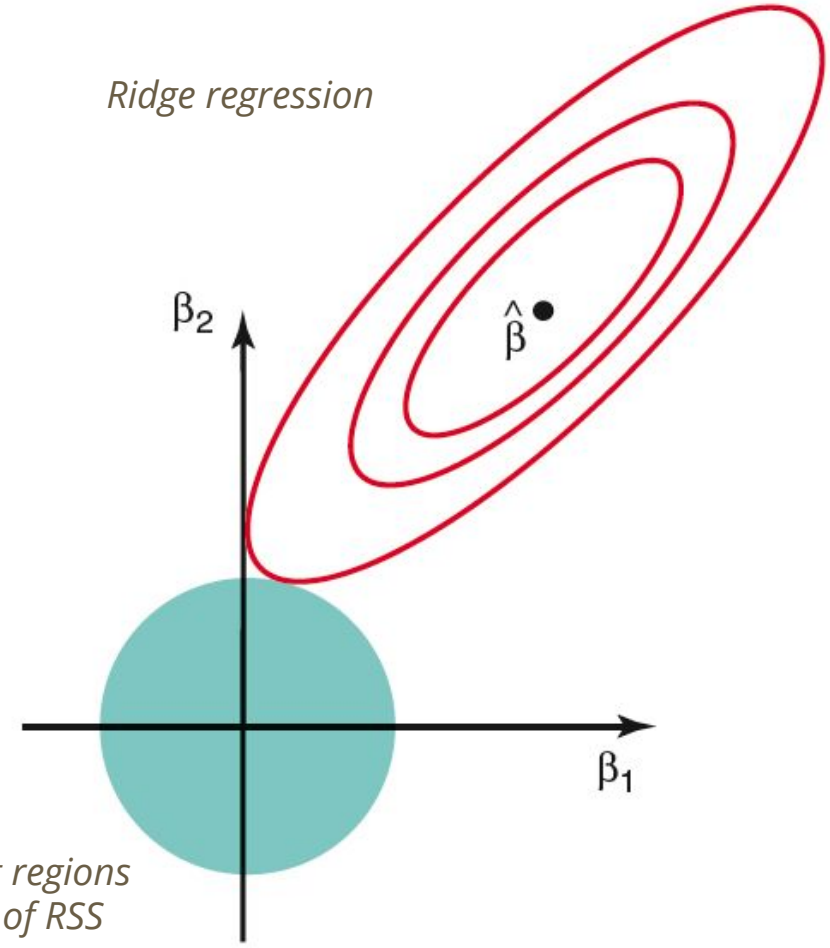


Error & constraint - the lasso vs. ridge regression

The lasso



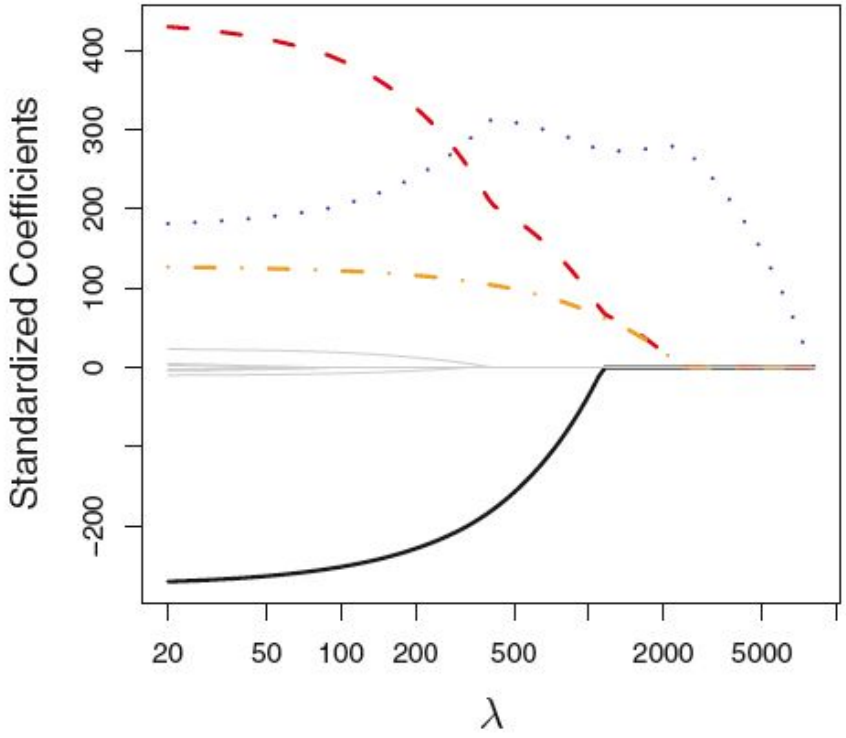
Ridge regression

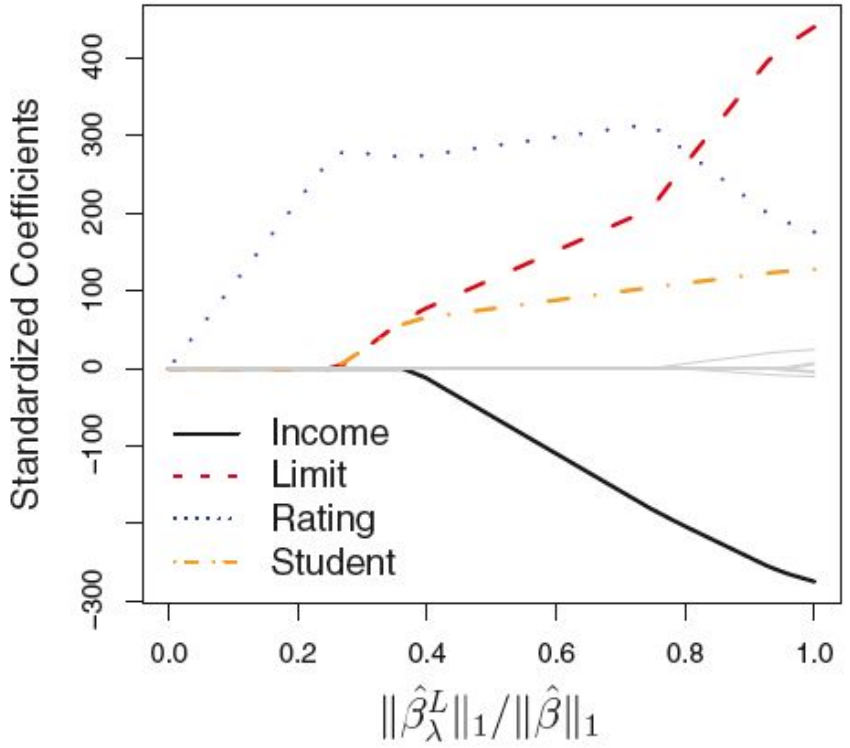


Blue/green: constraint regions
Red ellipses: contours of RSS

The effects of the lasso

- The lasso vs. credit





- May set feature coefficients to zero

Selecting the tuning parameter λ

- Procedure for a number of λ values
  - Use k-fold CV to estimate average MSE
  - Choose λ with lowest MSE
- We can find optimal λ by looking at MSE

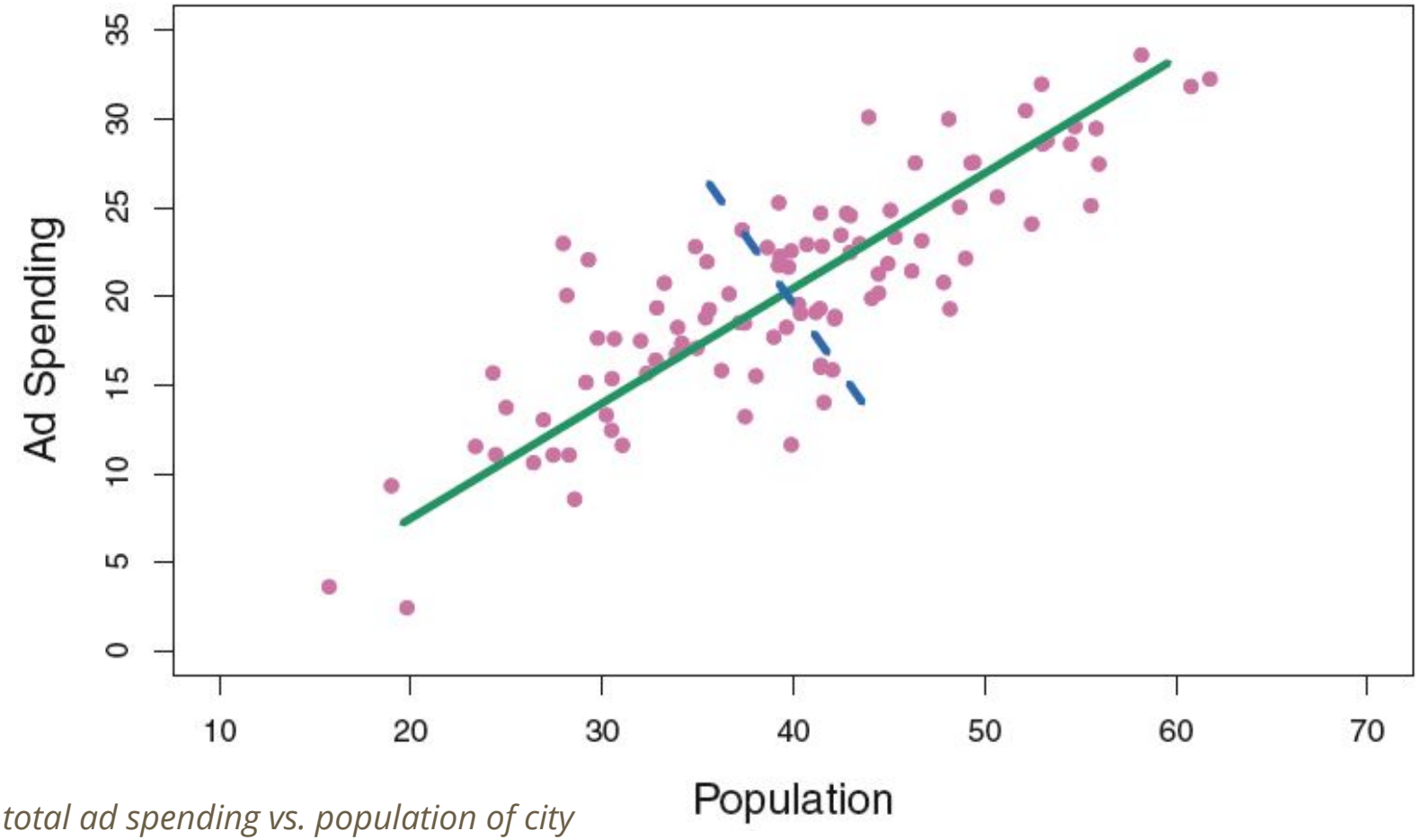
Dimension Reduction

- Recall that data is described in p dimensions
- Goal
  - Re-purpose linear regression model
  - Reduce variance for noisy features
  - Describe data using M+1 coefficients (M < p)
- Techniques:
  - Principal Components
  - Partial Least Squares

Principal Components Analysis

- Principal Component (intuitive)
  - Identify feature with maximum variance
  - Fit (linear) regression to that feature
  - Describe each data point with distance to regression line
- Rinse, wash, repeat:
  - Second Principal Component is created on feature with second most variance
  - Additional Principal Components are (necessarily) orthogonal to others
- … for dimensional reduction
  - There are always *p* principal components (Why?)
  - Use “description” of points as features
  - Fit model on first M descriptions

Population vs. ad spending data



Data: total ad spending vs. population of city

Green: first principal component
Blue: second principal component

PCR

- Formally
  - Principal component is a normalized linear combination of the original predictors
  - First principal component captures the maximum variance
- Caution
  - Features' should be in some comparable form — i.e. normalized
  - Features must be numeric; must convert categorical data

Thoughts on PCA

- May choose M through cross-validation
- Terminology
  - PCR: Principal Components Regression
  - PCA-transformation is formally: $X_1, ..., X_P$ —> $Z_1, ..., Z_M$
  - We can mitigate overfitting using PCA
- Assumptions
  - We can mitigate overfitting using PCR
  - Using $Z_1, ..., Z_M$ describes data better than $X_1, ..., X_P$
  - $Z_1, ..., Z_M$ is related to outcome (Y) — i.e. PCA is *unsupervised*

Implementation in scikit-learn

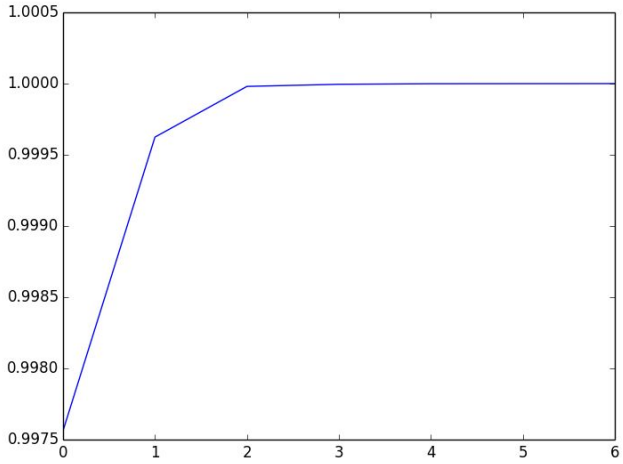
1) Import data
    a) If necessary, split data into train, test sets
2) Coerce data into: X (numpy matrix)

```
# 3a) BIC / AIC

from sklearn.decomposition import PCA
from sklearn.preprocessing import scale # For normalizing

components = 50   # Must be less than len(features)
pca = PCA(n_components=components)
pca.fit(X)
# Optionally, if you need variance
variance = pca.explained_variance_ratio_
```

Lab

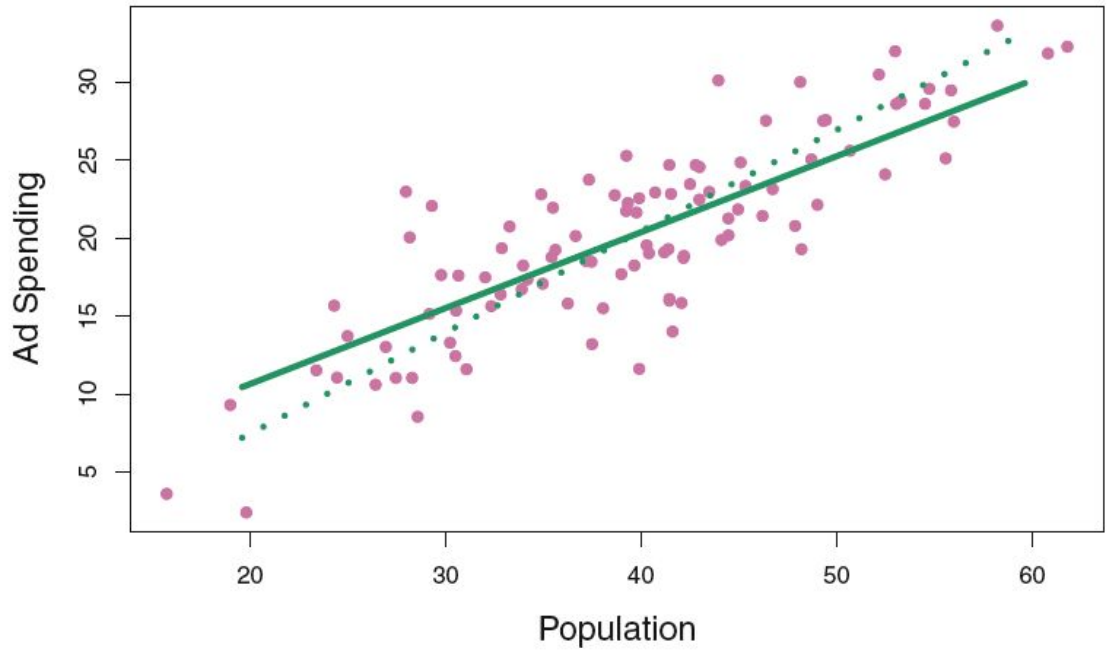
- Input
  - Use auto.csv [1:-1] in https://github.com/dbrizan/MSAN621-data
  - Field [1] = MPG (outcome); field[-1] is the auto make/model
  - This is a clean modification of auto data at UCI
- What to do
  - Get the explained variance list using PCA
  - Plot the cumulative explained variance:

```
import matplotlib.pyplot as plt
plt.plot(cumulative_explained_variance)
plt.show()
```



Partial Least Squares

- PLS
  - PCR identifies maximum variance without considering response
  - PLS identifies Z_1 as max variance with respect to outcome (Y)



Data: total ad spending vs. population of city

PCR: green dotted line
PLS: green solid line

- Theoretically: better fit to data