# Mathematical techniques in data science

Lecture 10: Linear regression

March 11th, 2019

## Schedule

| **Week** | **Chapter** |
|---|---|
| 1 | Chapter 2: Intro to statistical learning |
| 3 | Chapter 4: Classification |
| 4 | Chapter 9: Support vector machine and kernels |
| 5, 6 | Chapter 3: Linear regression |
| 7 | Chapter 8: Tree-based methods + Random forest |
| 8 | |
| 9 | Neural network |
| 12 | PCA $\rightarrow$ Manifold learning |
| 11 | Clustering: K-means $\rightarrow$ Spectral Clustering |
| 10 | Bootstrap + Bayesian methods + UQ |
| 13 | Reinforcement learning/Online learning/Active learning |
| 14 | Project presentation |

## Chapter 3 & 6: Topics on Linear regression

- Linear regression
- Subset selection
- Shrinkage methods
- *Dimension reduction*

## Supervised learning: standard setting

- Given: a sequence of label data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ sampled (independently and identically) from an unknown distribution $P_{X,Y}$
- Goal: predict the label of a new instance $x$
- In a regression problem, the outputs are not categorical

## Linear regression

Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \dots \beta^{(p)} X^{(p)} + \epsilon$$

- $p$: number of variables ($X \in \mathbb{R}^p$)
- $n$: number of observations

## Classical setting

Linear model

$$Y = \beta^{(0)} + \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \ldots \beta^{(p)}X^{(p)}$$

- $n \gg p$ ($n$ much larger than $p$). With enough observations, we hope to be able to build a good model
- even if the true relationship between the variables is not linear, we can include transformations of variables

$$X^{(p+1)} = [X^{(1)}]^2, \quad X^{(p+2)} = X^{(1)}X^{(3)}, \ldots$$

- adding transformed variables can increase $p$ significantly

## Trade-off: complexity vs. interpretability

Linear model

$$Y = \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \ldots \beta^{(p)}X^{(p)} + \epsilon$$

- Higher values of $p$ lead to more complex model $\rightarrow$ increases prediction power/accuracy
- Higher values of $p$ make it more difficult to interpret the model: It is often the case that some or many of the variables regression model are in fact not associated with the response

## Moderns settings

Linear model

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \ldots \beta^{(p)} X^{(p)} + \epsilon$$

- it is often the case that $n \ll p$
- requires supplementary assumptions (e.g. sparsity)
- can still build good models with very few observations.

## Linear regression by least squares

## Settings

- $Y \in \mathbb{R}^{n\times 1}, \quad X \in \mathbb{R}^{n\times (p+1)}$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} 1 & | & | & \dots & | \\ \dots & x^{(1)} & x^{(2)} & \dots & x^{(p)} \\ 1 & | & | & \dots & | \end{bmatrix}$$

  where $x^{(1)}, x^{(2)}, \dots, x^{(p)} \in \mathbb{R}^{n\times 1}$ are the observations of $X^{(1)}, X^{(2)}, \dots, X^{(p)}$.
- We want

$$Y = \beta^{(0)} + \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \dots \beta^{(p)}X^{(p)}$$

## Settings

- We want

$$Y = \beta^{(0)} + \beta^{(1)} X^{(1)} + \beta^{(2)} X^{(2)} + \dots \beta^{(p)} X^{(p)}$$

- Equivalent to

$$Y = X\beta, \qquad \beta = \begin{bmatrix} \beta^{(0)} \\ \beta^{(1)} \\ \dots \\ \beta^{(n)} \end{bmatrix}$$

## Least squares

$$Y = X\beta$$

- In general, the system has no solution ($n \gg p$) or infinitely many solutions ($n \ll p$)
- The most popular estimation method is least squares, in which we pick the coefficients to minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

## Least squares

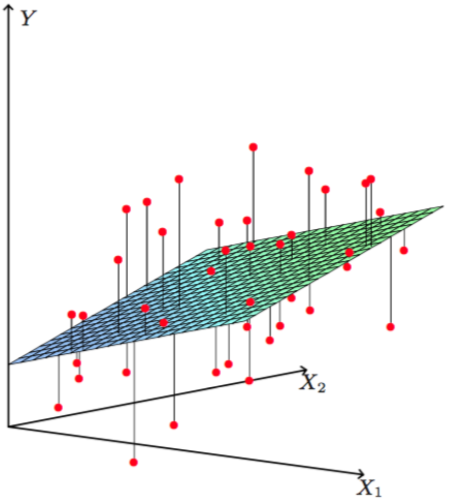


**FIGURE 3.1.** *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of $X$ that minimizes the sum of squared residuals from $Y$.*

## Least squares

- Minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- Or alternatively,

$$\hat{\beta} = \min_{\beta} \| Y - X\beta \|_2^2$$

## Least squares

- Minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$
$$= \sum_{i=1}^{n} \left( y_i - \beta^{(0)} - \beta^{(1)} x_i^{(1)} - \beta^{(2)} x_i^{(2)} - \ldots - \beta^{(p)} x_i^{(p)} \right)^2$$

- Taking derivative

$$\frac{\partial RSS}{\partial \beta^{(j)}} = \sum_{i=1}^{n} 2(y_i - x_i\beta)x_i^{(j)} = 2[x^{(j)}]^T (Y - X\beta)$$

## Least squares

- Set derivatives to zero

$$X^T(Y - X\beta) = 0$$

- If $X^TX$ is invertible

$$\hat{\beta} = (X^TX)^{-1}X^TY$$

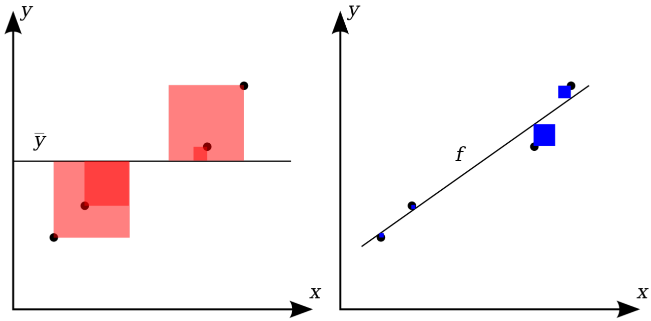
- Predicted values

$$\hat{Y} = X_{test}\hat{\beta} = X_{test}(X_{train}^T X_{train})^{-1}X_{train}^T Y_{train}$$

## The coefficient of determination

- The coefficient of determination, called “R squared” and denoted by
$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$
where $\bar{y}$ is the average of $y_1, \dots, y_n$
- Often used to measure the quality of a linear model
- A model with a $R^2$ close to 1 fit the data well.

## The coefficient of determination



In some sense, the $R^2$ measures how much better is the prediction compared to a constant prediction

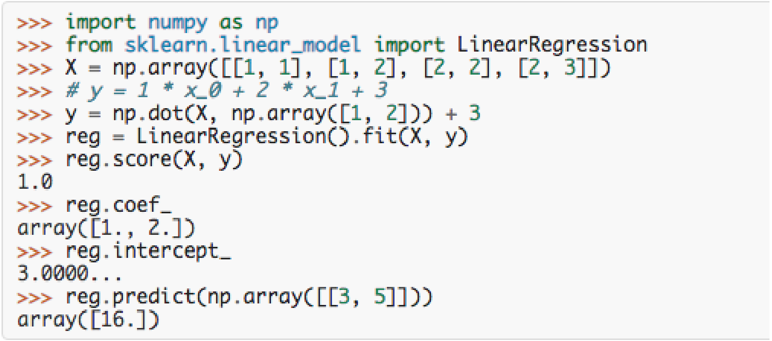
## The adjusted coefficient of multiple determination

- It is desirable to adjust $R^2$ to take account of the fact that its value may be quite high just because many predictors were used relative to the amount of data
- The adjusted coefficient of multiple determination

$$R_a^2 = 1 - \frac{\frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

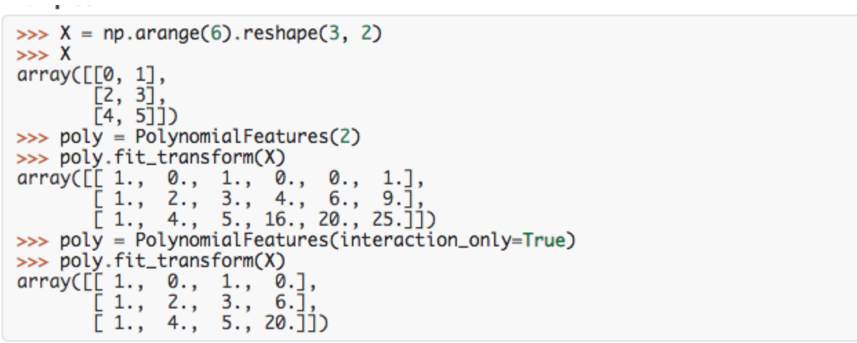
where $\bar{y}$ is the average of $y_1, \ldots, y_n$

## sklearn.linear_model.LinearRegression



```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0000...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

## sklearn.preprocessing.PolynomialFeatures



```
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
>>> poly = PolynomialFeatures(interaction_only=True)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.],
       [ 1.,  2.,  3.,  6.],
       [ 1.,  4.,  5., 20.]])
```

## Subset selection

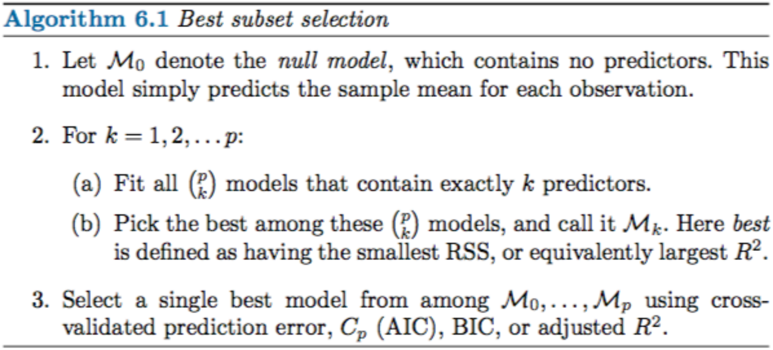
## Trade-off: complexity vs. interpretability

Linear model

$$Y = \beta^{(1)}X^{(1)} + \beta^{(2)}X^{(2)} + \ldots \beta^{(p)}X^{(p)} + \epsilon$$

- Higher values of $p$ lead to more complex model $\rightarrow$ increases prediction power/accuracy
- Higher values of $p$ make it more difficult to interpret the model
- Ideally, we would like to try out a lot of different models, each containing a different subset of the predictors, then select the best model
- Problem: there are $2^p$ models that contain subsets of $p$ variables

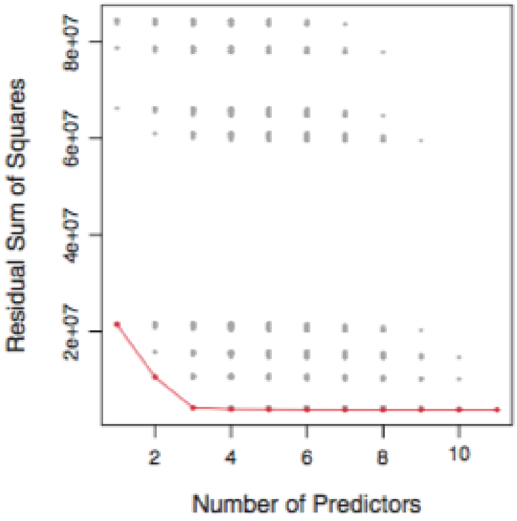
## Best subset selection



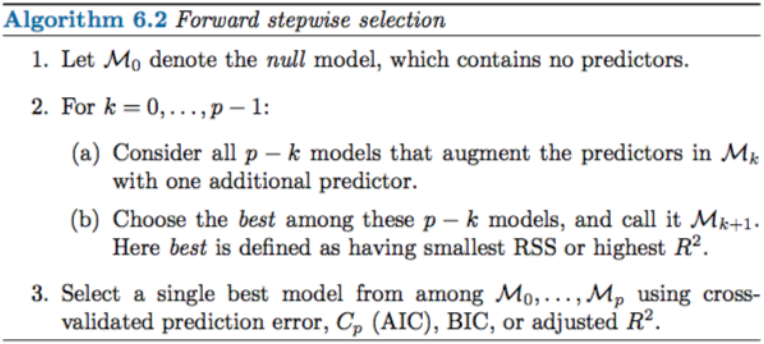
**Algorithm 6.1** *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \dots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly $k$ predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$ using cross-validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.

## Best subset selection



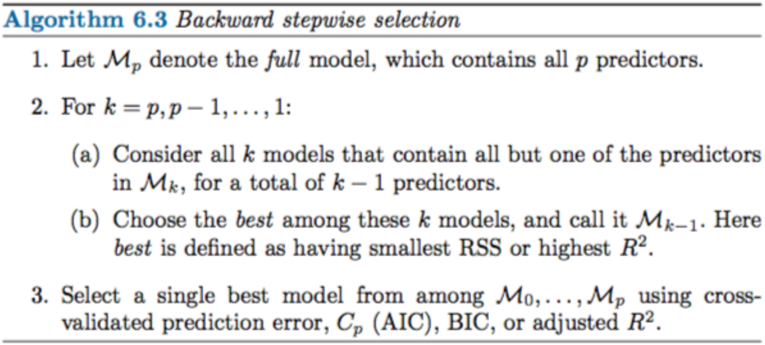
## Forward stepwise selection



**Algorithm 6.2** *Forward stepwise selection*

```
1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
    (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
    (b) Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.
```

## Backward stepwise selection



**Algorithm 6.3** *Backward stepwise selection*

```
1. Let $\mathcal{M}_p$ denote the *full* model, which contains all $p$ predictors.
2. For $k = p, p-1, \ldots, 1$:
    (a) Consider all $k$ models that contain all but one of the predictors
        in $\mathcal{M}_k$, for a total of $k-1$ predictors.
    (b) Choose the *best* among these $k$ models, and call it $\mathcal{M}_{k-1}$. Here
        *best* is defined as having smallest RSS or highest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-
   validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.
```

## Hybrid approach

- Hybrid versions of forward and backward stepwise selection are available
- variables are added to the model sequentially
- after adding each new variable, the method may also remove any variables that no longer provide an improvement in the model fit

## Choosing the optimal model

- in order to implement these methods, we need a way to determine which of these models is best
- we wish to choose a model with a low test error
    - indirectly estimate test error by making an adjustment to the training error to account for the bias due to overfitting
    - indirectly estimate the test error, using either a validation set approach or a cross-validation approach

## Adjusted training errors

- Adjusted $R^2$
- Mallow's $C_p$

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

  where $\hat{\sigma}^2$ is an estimate of the variance of the error, $d$ is the number of predictors

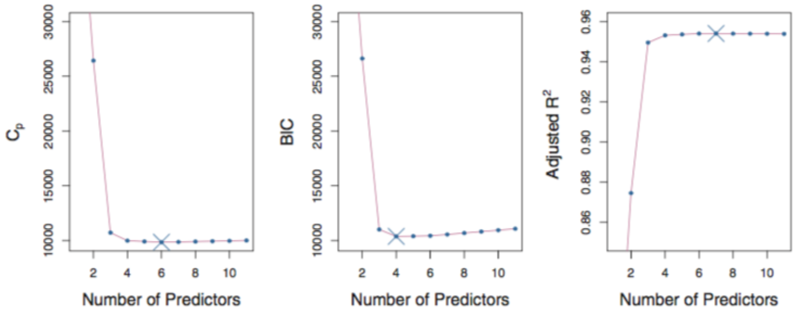
- AIC (Akaike information criterion)

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

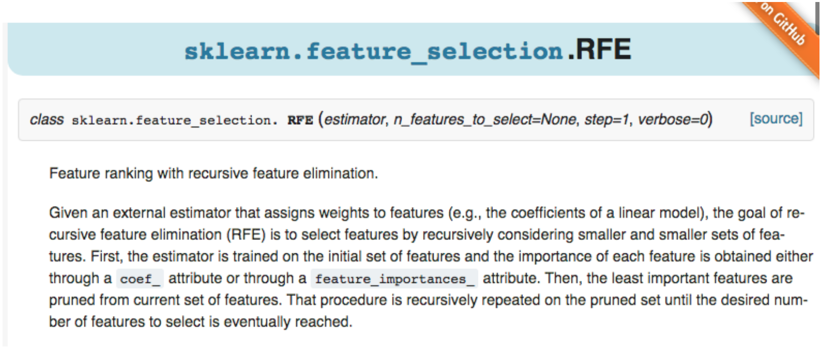
- BIC (Bayesian information criterion)

$$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + \log(n)\hat{\sigma}^2)$$

## Choosing the optimal model



## sklearn does not support subset selection



sklearn.feature_selection.RFE

*class* sklearn.feature_selection. **RFE** (*estimator*, *n_features_to_select=None*, *step=1*, *verbose=0*) [source]

Feature ranking with recursive feature elimination.

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a `coef_` attribute or through a `feature_importances_` attribute. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

## Shrinkage methods

## Least squares

- Least squares regression

$$\hat{\beta}^{LS} = \min_{\beta} \| Y - X\beta \|_2^2$$

- Penalizing the coefficients:
  - restrict the number of the regression coefficients
  - stabilize the estimator to prevent overfitting
  - add a penalty for including a non-zero coefficient

## $\ell_0$ regularization

- $\ell_0$ regularization

$$\hat{\beta}^0 = \min_{\beta} \|Y - X\beta\|_2^2 + \lambda \sum_{i=1}^{p} \mathbf{1}_{\beta(i) \neq 0}$$

  where $\lambda > 0$ is a parameter
- pay a fixed price $\lambda$ for including a given variable into the model
- variables that do not significantly contribute to reducing the error are excluded from the model (i.e., $\beta_i = 0$)
- problem: difficult to solve (combinatorial optimization). Cannot be solved efficiently for a large number of variables.

## $\ell_2$ (Tikhonov) regularization

- Ridge regression/ Tikhonov regularization

$$\hat{\beta}^0 = \min_{\beta} \|Y - X\beta\|_2^2 + \lambda \sum_{i=1}^{p} [\beta^{(i)}]^2$$

  where $\lambda > 0$ is a parameter
- shrinks the coefficients by imposing a penalty on their size
- penalty is a smooth function.
- easy to solve (solution can be written in closed form)
- can be used to regularize a rank deficient problem ($n < p$)

## $\ell_2$ (Tikhonov) regularization

$$\frac{\partial \left( \|Y - X\beta\|_2^2 + \lambda \|\beta\|^2 \right)}{\partial \beta} = 2X^T(Y - X\beta) + 2\lambda\beta$$

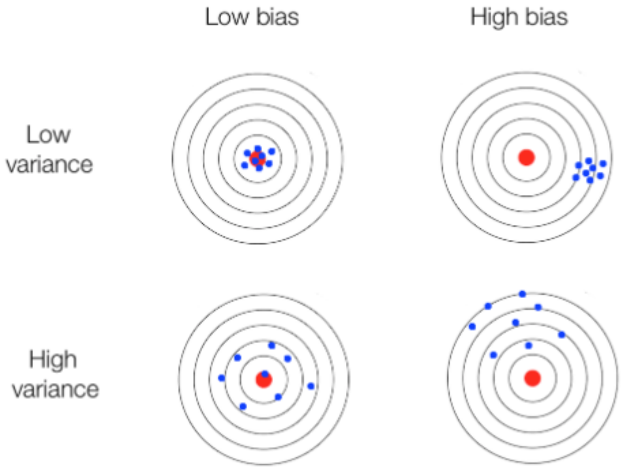
- The critical point satisfies

$$(X^TX + \lambda I)\beta = X^TY$$

- Note: $(X^TX + \lambda I)$ is positive definite, and thus invertible
- Thus

$$\hat{\beta}^{RIDGE} = (X^TX + \lambda I)^{-1}X^TY$$

## Bias-variance decomposition



$$MSE(\hat{\theta}) = Var(\hat{\theta}) + (bias)^2$$

## Stein's phenomenon

- Given i.i.d. $X_1, \dots, X_n$ samples from $\mathcal{N}_p(\mu, I_p)$ ($p \geq 3$), we wish to estimate $\mu$
- The accuracy of an estimator is measured by the risk function

$$MSE(\hat{\mu}) = E[\|\hat{\mu} - \mu\|^2]$$

- The standard estimate is

$$\bar{X} = \frac{X_1 + \ldots + X_n}{n}$$

which minimizes

$$\min_c \sum_{i=1}^n \|X_i - c\|^2$$

## Stein's phenomenon

- The standard estimate is

$$\bar{X} = \frac{X_1 + \ldots + X_n}{n}$$

which minimizes

$$\min_c \sum_{i=1}^{n} \|X_i - c\|^2$$

- James-Stein's estimator

$$\mu^{JS} = \left(1 - \frac{p-2}{n\|\bar{X}\|^2}\right)\bar{X}$$

is a strictly better estimator than the sample mean $\bar{X}$

## $\ell_2$ (Tikhonov) regularization

$$\hat{\beta}^{RIDGE} = (X^T X + \lambda I)^{-1} X^T Y$$

- When $\lambda > 0$, the estimator is defined even when $n < p$
- When $\lambda > 0$ and $n = p$, we recover the usual least squares solution

## $\ell_2$ (Tikhonov) regularization

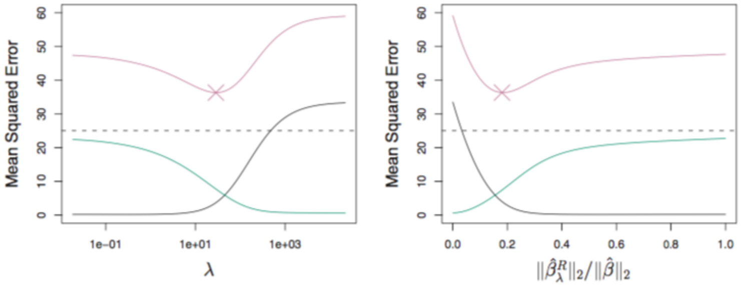


**FIGURE 6.5.** *Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of $\lambda$ and $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$. The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.*