Lecture 13: Principal Components Analysis

Statistical Learning (BST 263)

Jeffrey W. Miller

Department of Biostatistics
Harvard T.H. Chan School of Public Health

(Figures from *An Introduction to Statistical Learning*, James et al., 2013, and *The Elements of Statistical Learning*, Hastie et al., 2008)

Outline

Outline

Unsupervised Learning

- So far, we have focused on supervised learning.
- In supervised learning, we are given examples (x_i, y_i), and we try to predict y for future x's.

- In unsupervised learning, we are given only x_i's, with no outcome y_i.
- Unsupervised learning is less well defined, but basically consists of finding some structure in the x's.

- Two most common types of unsupervised learning:
  1. Finding lower-dimensional representations.
  2. Finding clusters / groups.

Unsupervised Learning

- Data: $x_1, \ldots, x_n$.
- Often, $x_i \in \mathbb{R}^p$, but the x_i's could be anything.
  - e.g., time-series data, documents, images, movies, mixed-type.

- Unsupervised learning can be used in many different ways:
  - Visualization of high-dimensional data
  - Exploratory data analysis
  - Feature construction for supervised learning
  - Discovery of hidden structure
  - Removal of unwanted variation (e.g., batch effects, technical biases, population structure)
  - Matrix completion or De-noising
  - Density estimation
  - Compression

Unsupervised Learning

- In supervised learning, we can use the outcome y to reliably evaluate performance.
- This enables us to:
  - choose model settings and
  - estimate test performance

  via cross-validation or similar train/test split approaches.

- However, we don't have this luxury in unsupervised learning.
- A challenge is that often there is no standard way to evaluate the performance of an unsupervised method.

Outline

Principal Components Analysis (PCA)

- PCA is an unsupervised method for dimension reduction.
- That is, finding a lower-dimensional representation.

- PCA is the oldest and most commonly used method in this class.
  - PCA goes back at least to Karl Pearson in 1901.

- Basic idea: Find a low-dimensional representation that approximates the data as closely as possible in Euclidean distance.

PCA example: Ad spending

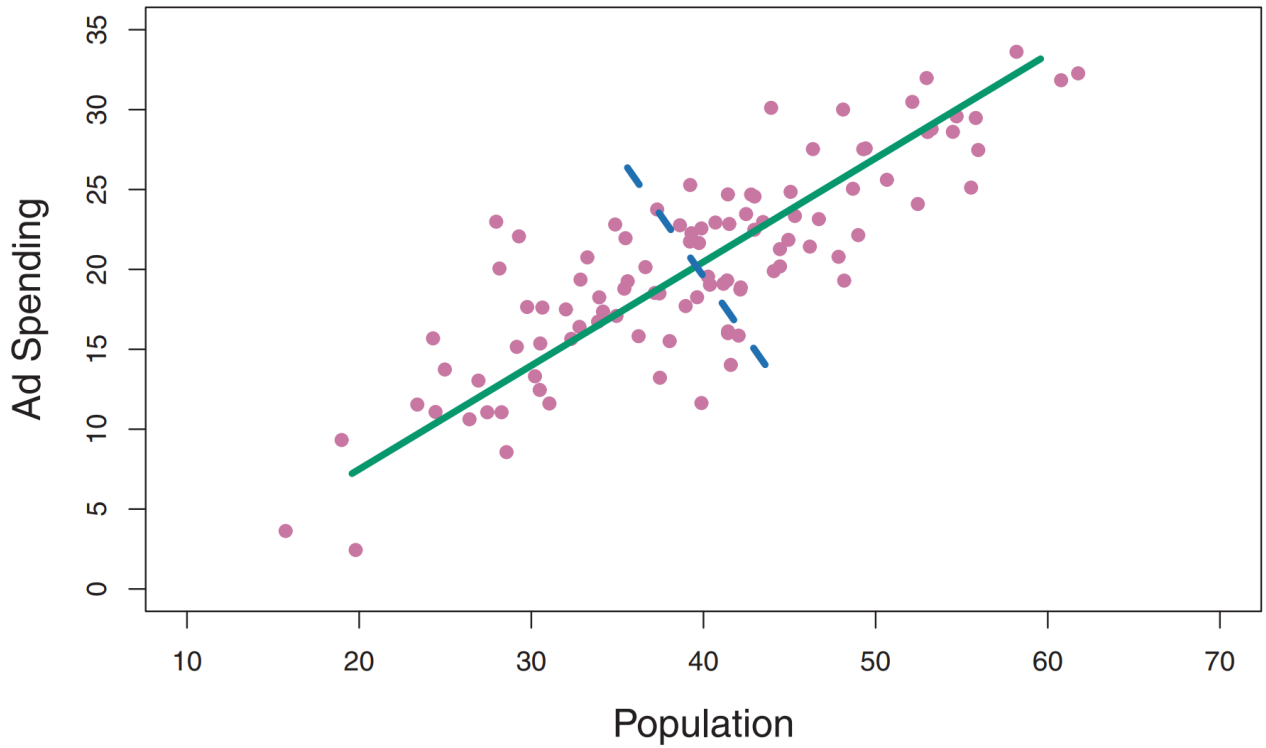


FIGURE 6.14. *The population size (*pop*) and ad spending (*ad*) for* 100 *different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.*

PCA example: Ad spending

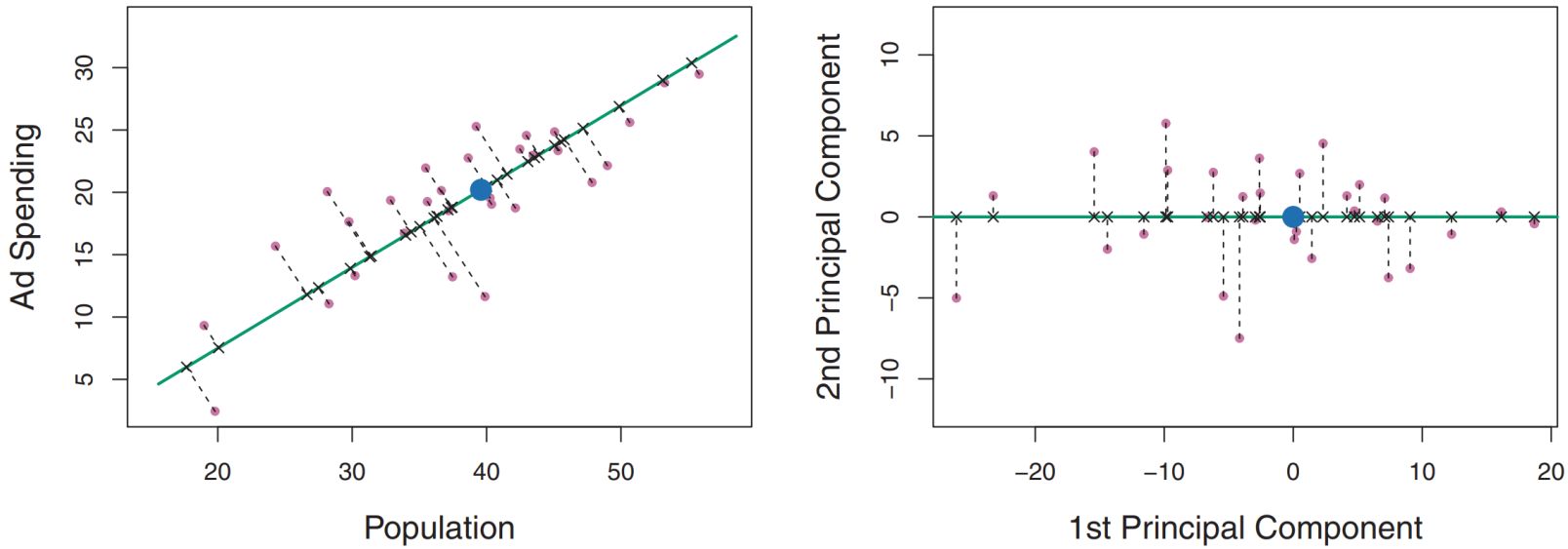


FIGURE 6.15. *A subset of the advertising data. The mean* pop *and* ad *budgets are indicated with a blue circle.* Left: *The first principal component direction is shown in green. It is the dimension along which the data vary the most, and it also defines the line that is closest to all* n *of the observations. The distances from each observation to the principal component are represented using the black dashed line segments. The blue dot represents* $(\overline{\texttt{pop}}, \overline{\texttt{ad}})$. Right: *The left-hand panel has been rotated so that the first principal component direction coincides with the* x*-axis.*

PCA example: Ad spending

PC1 and PC2 scores:

$$Z_1 = 0.839 \times (\texttt{pop} - \overline{\texttt{pop}}) + 0.544 \times (\texttt{ad} - \overline{\texttt{ad}}).$$

$$Z_2 = 0.544 \times (\texttt{pop} - \overline{\texttt{pop}}) - 0.839 \times (\texttt{ad} - \overline{\texttt{ad}}).$$

In this example, pop and ad are both highly correlated with PC1:

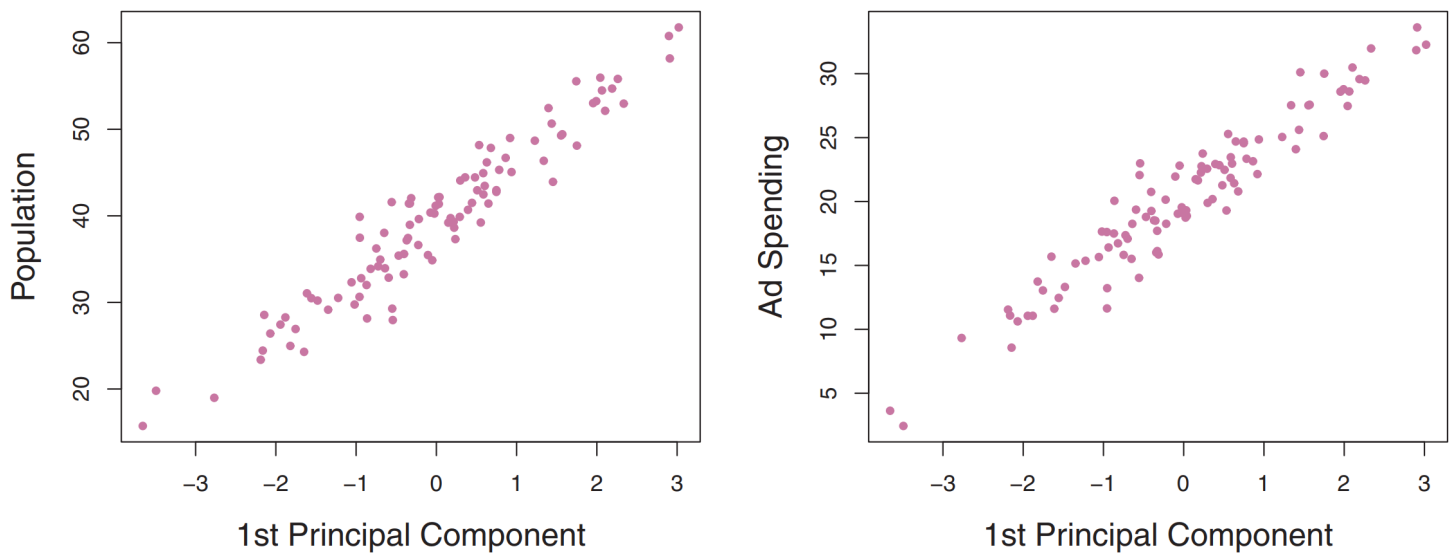


FIGURE 6.16. *Plots of the first principal component scores z_{i1} versus* pop *and* ad*. The relationships are strong.*

PCA example: Ad spending

PC1 and PC2 scores:

$$Z_1 = 0.839 \times (\texttt{pop} - \overline{\texttt{pop}}) + 0.544 \times (\texttt{ad} - \overline{\texttt{ad}}).$$

$$Z_2 = 0.544 \times (\texttt{pop} - \overline{\texttt{pop}}) - 0.839 \times (\texttt{ad} - \overline{\texttt{ad}}).$$

Pop and ad versus PC2:

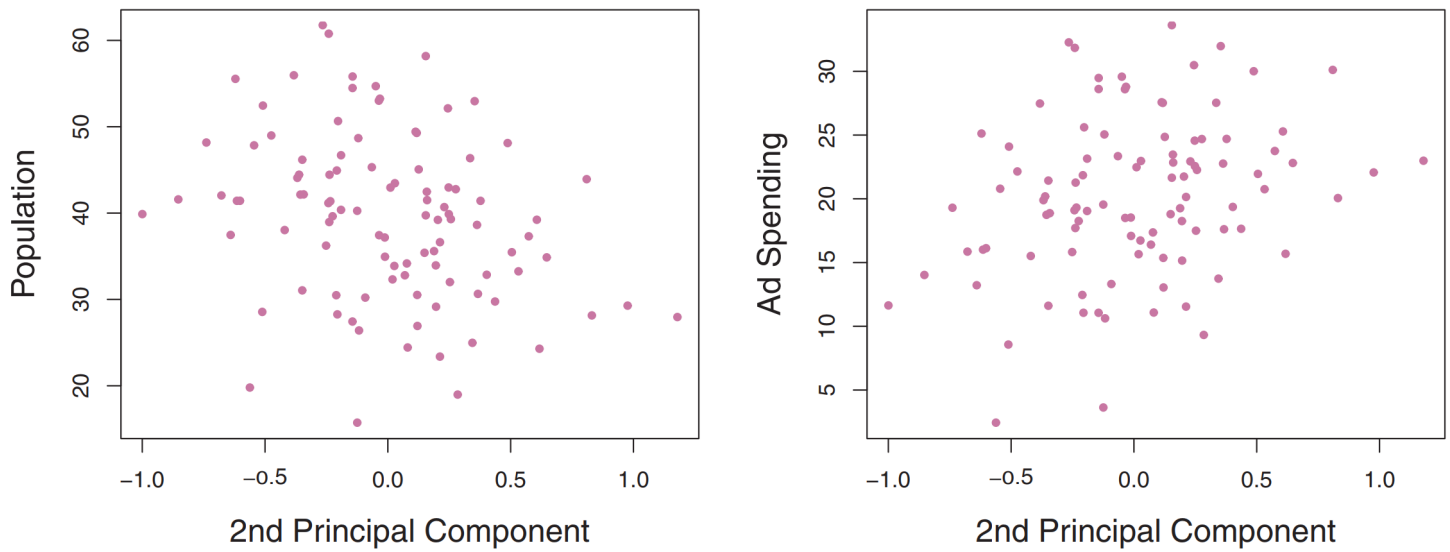


FIGURE 6.17. *Plots of the second principal component scores z_{i2} versus* pop *and* ad*. The relationships are weak.*

PCA example: Half-sphere simulation

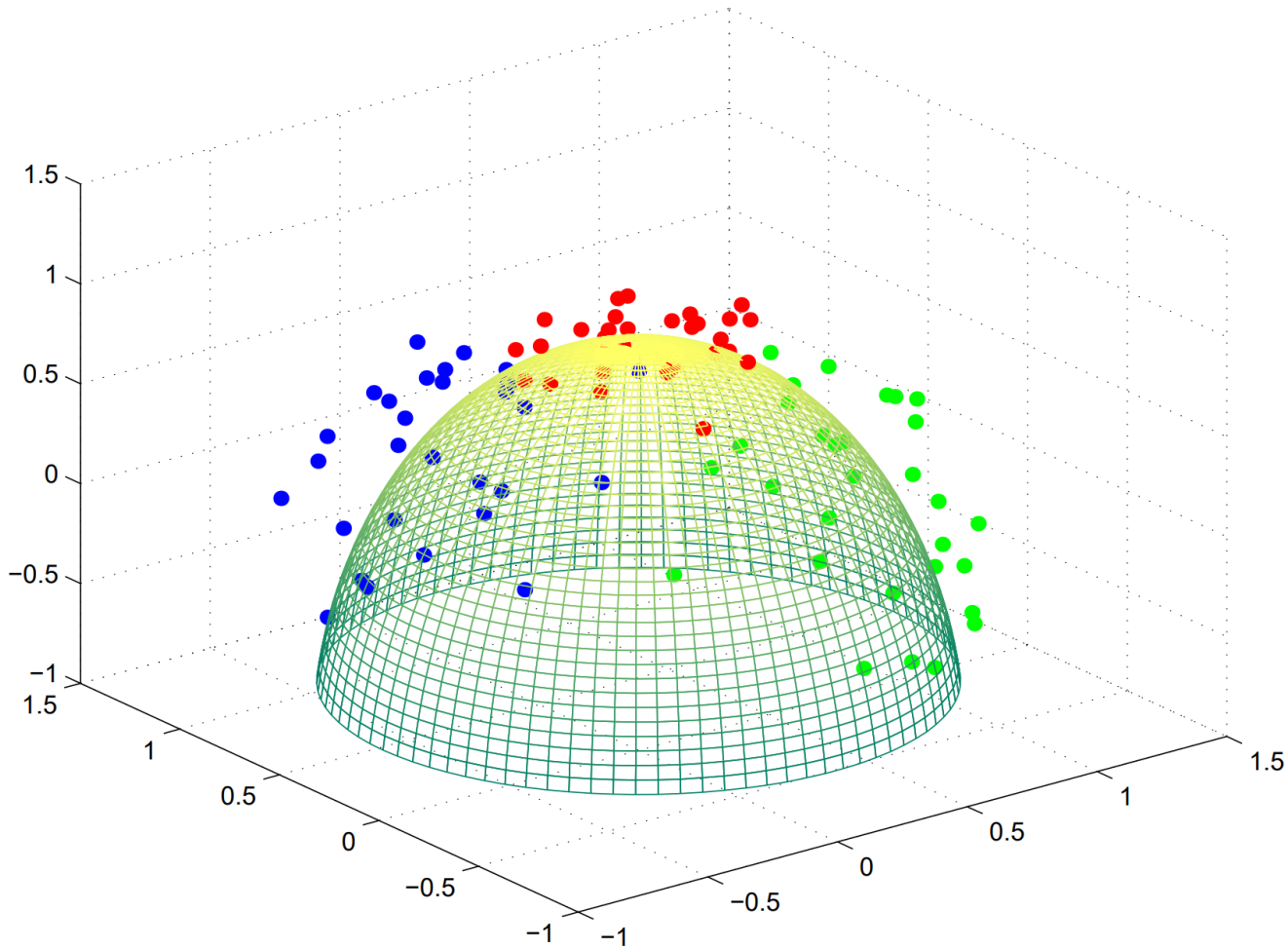


FIGURE 14.15. *Simulated data in three classes, near the surface of a half–sphere.*

PCA example: Half-sphere simulation

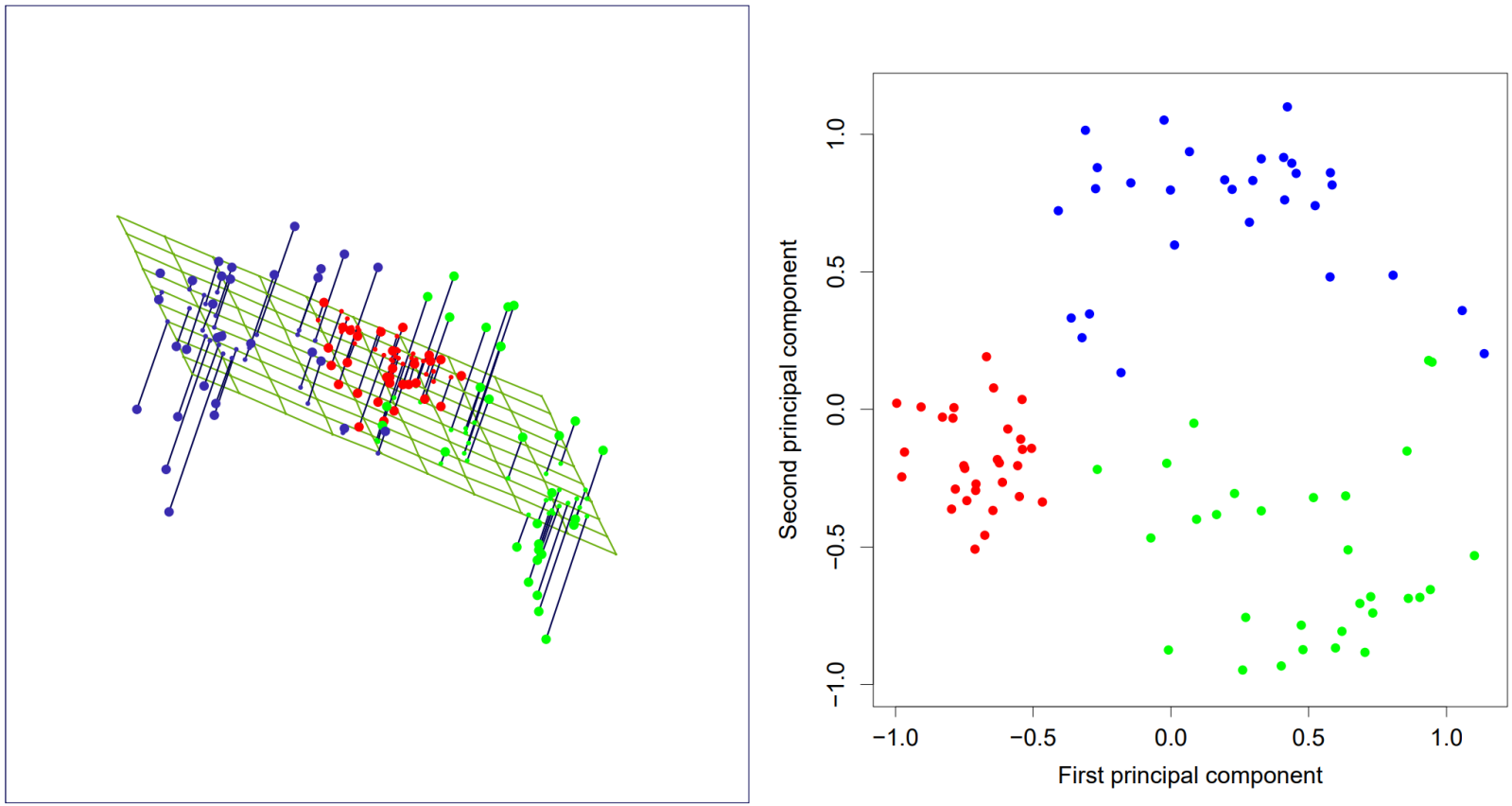


FIGURE 14.21. *The best rank-two linear approximation to the half-sphere data. The right panel shows the projected points with coordinates given by* $\mathbf{U}_2\mathbf{D}_2$*, the first two principal components of the data.*

PCA directions, scores, and scales

PC directions

- The *PC1 direction* is the direction along which the data has the largest variance.
- The *PCm direction* is the direction along which the data has the largest variance, among all directions orthogonal to the first $m-1$ PC directions.

PC scores

- The *PCm score* for point x_i is the position of x_i along the mth PC direction.
- Mathematically, the PCm score is the dot product of x_i with the PCm direction.

PC scales

- The *PCm scale* is the standard deviation of the data along the PCm direction.

Other interpretations of PCA

- Best approximation interpretation:
  - PC1 score $\times$ PC1 direction $=$ the best 1-dimensional approximation to the data in terms of MSE.
  - $\sum_{m=1}^{M}$ PCm score $\times$ PCm direction $=$ the best M-dimensional approximation to the data in terms of MSE.
- Eigenvector interpretation:
  - The PCm direction is the mth eigenvector (normalized to unit length) of the covariance matrix, sorting the eigenvectors by the size of their eigenvalues.

Outline

Computing the PCA directions and scores

- Covariance method
  - Simplest way of doing PCA.
  - Based on the eigenvector interpretation.
  - Slow when p is large.
- Singular value decomposition (SVD) method
  - Faster for large p.
  - Truncated SVD allows us to compute only the top PCs, which is much faster than computing all PCs when p is large.
  - More numerically stable.
- The SVD method is usually preferred.

Covariance method of computing PCA

- Data: $x_1, \ldots, x_n$ where $x_i \in \mathbb{R}^p$.
- For simplicity, assume the data has been centered so that $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$ for each j.

- Usually, it is a good idea to also scale the data to have unit variance along each dimension j.
- However, if the data are already in common units then it may be better to not standardize the scales.

Covariance method of computing PCA

- Put the data into an $n \times p$ matrix: $X = \begin{bmatrix} x_1^{\mathsf{T}} \\ \vdots \\ x_n^{\mathsf{T}} \end{bmatrix}$.
- Estimate the covariance matrix:

$$C = \tfrac{1}{n-1} X^{\mathsf{T}} X = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^{\mathsf{T}}.$$

- Compute the eigendecomposition: $C = V \Lambda V^{\mathsf{T}}$.
  - $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_p)$ where $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$ are the eigenvalues of C.
  - V is orthogonal and the mth column of V is the mth eigenvector of C.

- PCm direction is mth column of V.
- PCm scale is $\sqrt{\lambda_m}$.
- PC score vector (i.e., PC1-PCp scores) for x_i is $V^{\mathsf{T}} x_i$.
  - So XV gives us all the scores for all the x_i's.

Notes on the covariance method

- A matrix $V \in \mathbb{R}^{p \times p}$ is *orthogonal* if $VV^\mathsf{T} = V^\mathsf{T}V = I$.
- Since V is an orthogonal matrix,
  - the PC directions are orthogonal to one another,
  - the PC directions have unit length (Euclidean norm of 1), and
  - the PC scores XV are a rotated version of the data.
- Most languages have tools for computing the eigendecomposition, e.g., the `eigen` function in R.
- Notation: $\operatorname{diag}(\lambda_1, \ldots, \lambda_p) = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p \end{bmatrix}$.
- This follows from "block matrix multiplication":

$$X^\mathsf{T}X = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} x_1^\mathsf{T} \\ \vdots \\ x_n^\mathsf{T} \end{bmatrix} = \sum_{i=1}^{n} x_i x_i^\mathsf{T}.$$

PCA example: Hand-written digits

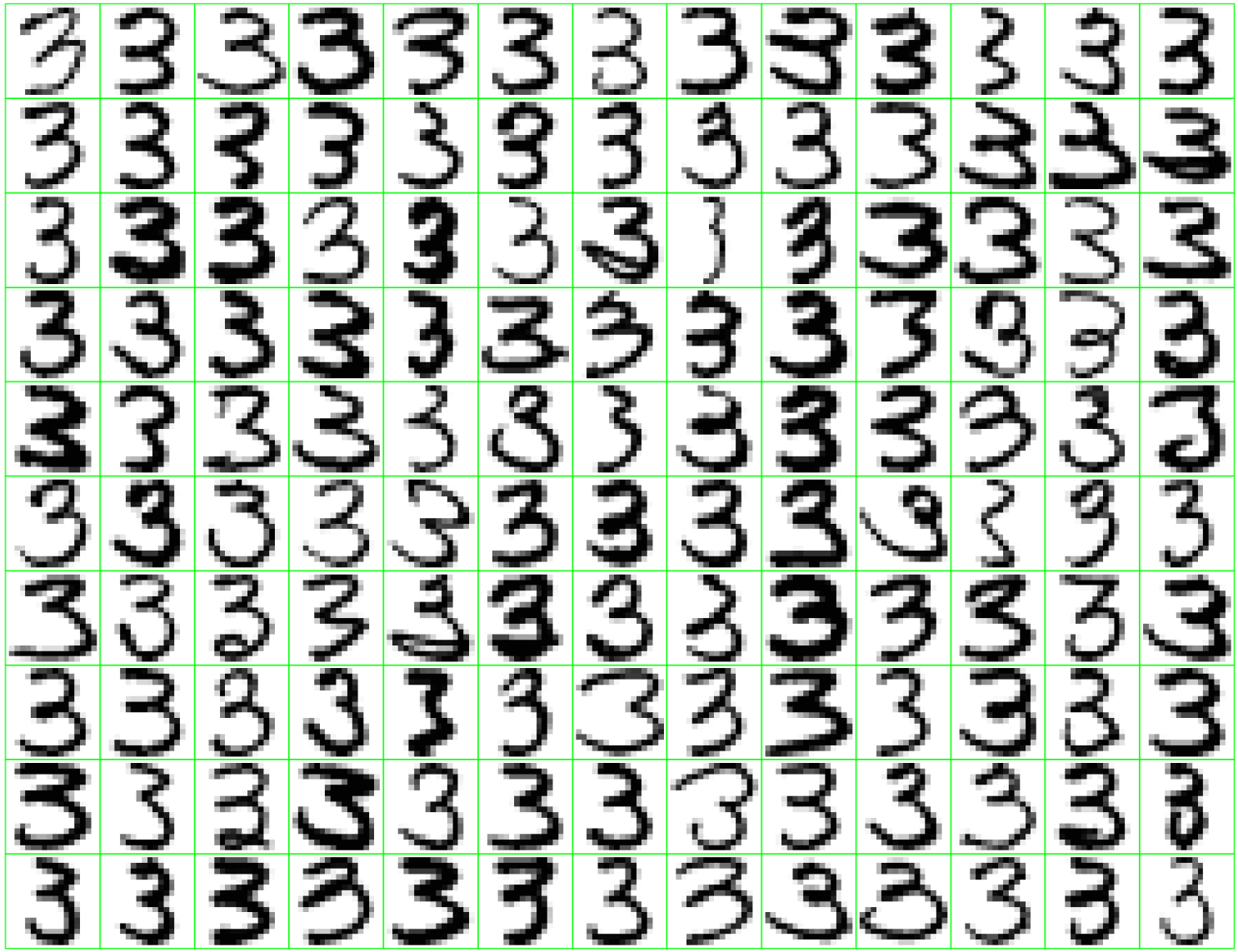
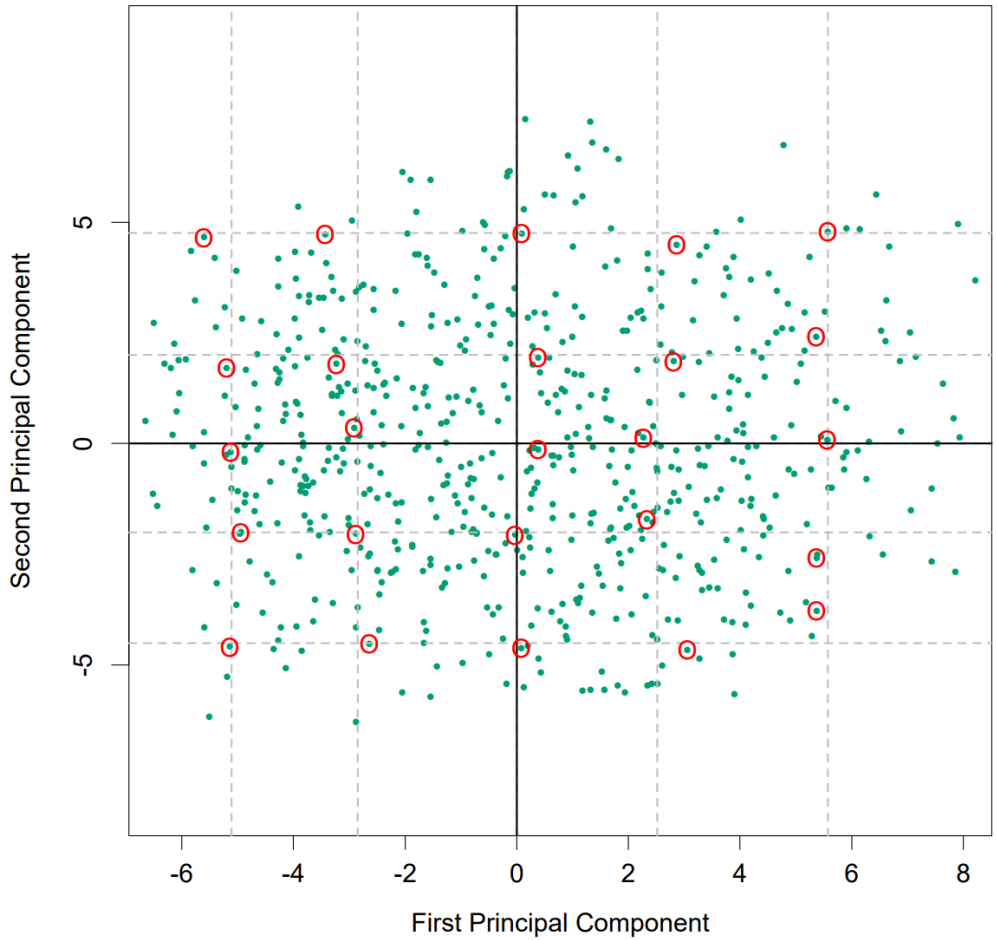


FIGURE 14.22. *A sample of* 130 *handwritten 3's shows a variety of writing styles.*

PCA example: Hand-written digits



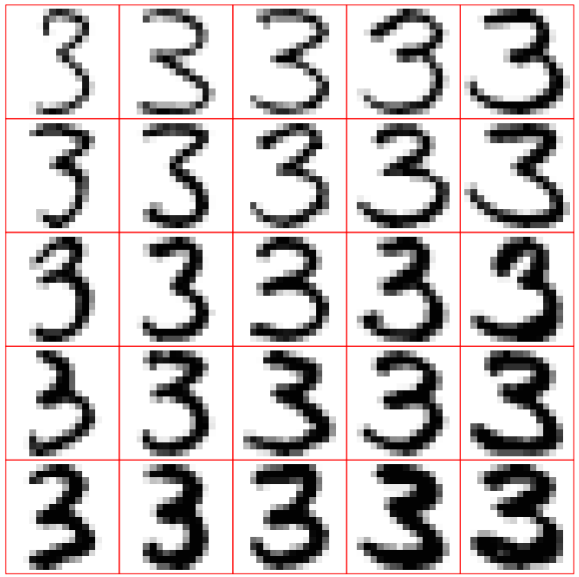


FIGURE 14.23. *(Left panel:) the first two principal components of the hand-written threes. The circled points are the closest projected images to the vertices of a grid, defined by the marginal quantiles of the principal components. (Right panel:) The images corresponding to the circled points. These show the nature of the first two principal components.*

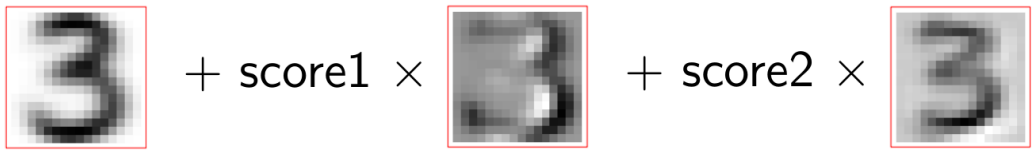
PCA example: Hand-written digits

- PCA can be used to make a low-dimensional approximation to each image.

- The PCA approximation is the sum of scores times directions, plus the sample mean since data is centered at 0:

$$\text{PC approx} = \text{sample mean} + \text{score1} \times \text{dir1} + \text{score2} \times \text{dir2}$$

$$= \quad + \text{score1} \times \quad + \text{score2} \times$$



- In this example, each PC direction is, itself, an image.

PCA example: Gene expression

PCA scores for gene expression of leukemia samples:

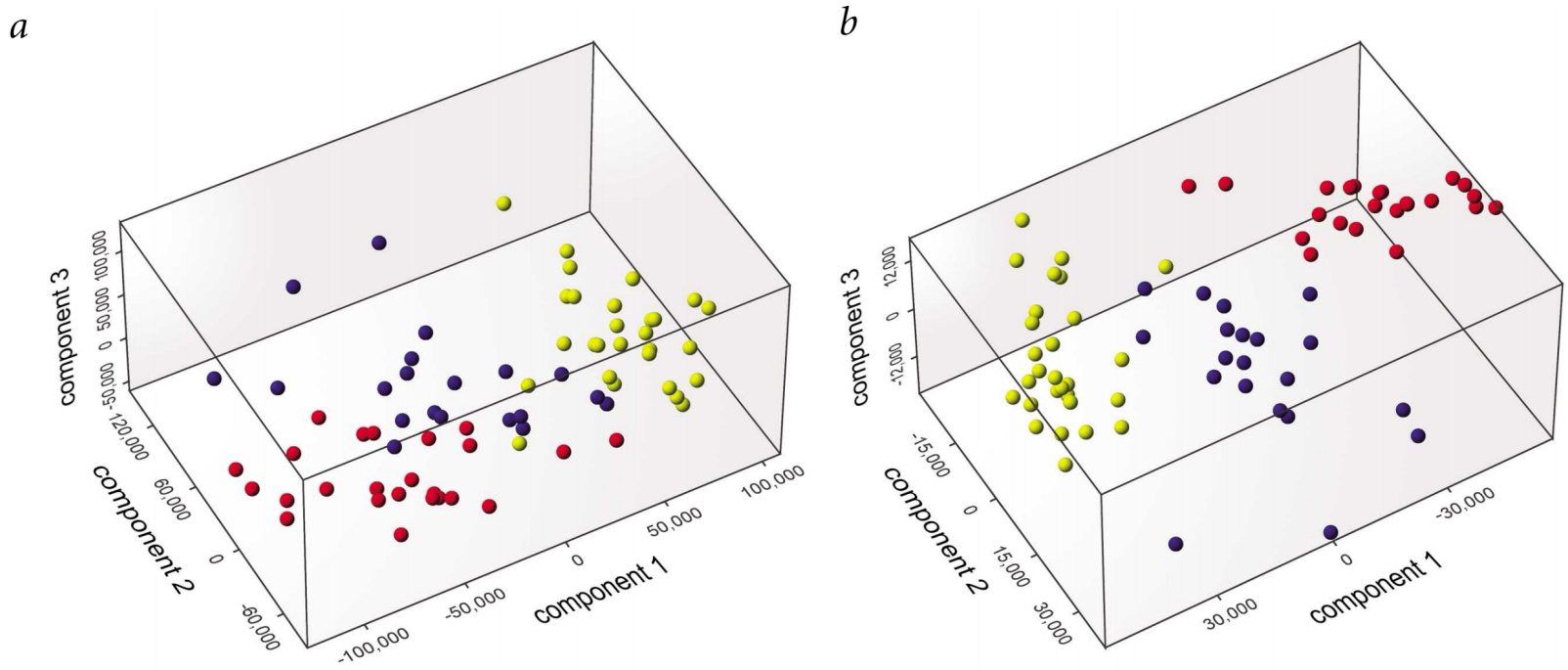


Fig. 4 Comparison of gene expression between ALL, MLL and AML. ***a***, Principal component analysis (PCA) plot of ALL (red), MLL (blue) and AML (yellow) carried out using 8,700 genes that passed filtering. ***b***, PCA plot comparing ALL (red), MLL (blue) and AML (yellow) using the 500 genes that best distinguished ALL from AML. Three-dimensional virtual reality modeling language (VRML) plots can be viewed at our web site (http://research.dfci.harvard.edu/korsmeyer/MLL.htm).

(figure from Armstrong et al., 2002, Nature Genetics)

Outline

SVD method of computing PCA

- The covariance method is simple, but slow when p is large.

- SVD = Singular Value Decomposition.
- The SVD method is faster for large p.
- Truncated SVD allows us to compute only the top PCs, which is much faster than computing all PCs when p is large.

- The *SVD* of $X \in \mathbb{R}^{n \times p}$ is $X = USV^{\mathrm{T}}$ where
  - $U \in \mathbb{R}^{n \times n}$ is orthogonal,
  - $V \in \mathbb{R}^{p \times p}$ is orthogonal, and
  - $S \in \mathbb{R}^{n \times p}$ is zero everywhere except $s_{11} \geq s_{22} \geq \cdots \geq 0$, which are called the *singular values*.

SVD method of computing PCA

- Put the data into an $n \times p$ matrix: $X = \begin{bmatrix} x_1^{\mathsf{T}} \\ \vdots \\ x_n^{\mathsf{T}} \end{bmatrix}$.
- Compute the SVD: $X = USV^{\mathsf{T}}$.

- PCm direction is the mth column of V.
- PCm scale is $\frac{1}{\sqrt{n-1}} s_{mm}$.
- PC score vector (i.e., PC1-PCp scores) for x_i is $V^{\mathsf{T}} x_i$.

- Connection to covariance method:

$$\begin{aligned} V\Lambda V^{\mathsf{T}} = C = \tfrac{1}{n-1} X^{\mathsf{T}} X = \tfrac{1}{n-1} V S^{\mathsf{T}} U^{\mathsf{T}} U S V^{\mathsf{T}} \\ = \tfrac{1}{n-1} V S^{\mathsf{T}} S V^{\mathsf{T}} = V (\tfrac{1}{n-1} S^{\mathsf{T}} S) V^{\mathsf{T}}. \end{aligned}$$

SVD method of computing PCA

- The truncated SVD allows us to compute only the top PCs.
- *Truncated SVD* computes $U[, 1:n_u]$, S, and $V[1:n_v,]$ for user-specified choices of n_u and n_v.
- This is much faster than computing all PCs when p is large.
- Usually, we only need the top few PCs anyway.

Outline

Principal Components Regression (PCR)

- We have seen methods of controlling variance by:
  - using a less flexible model, e.g., fewer parameters,
  - selecting a subset of predictors,
  - regularization / shrinkage.
- Another approach: transform the predictors to be lower dimensional.
  - Various transformations: PCA, ICA, principal curves.
- Combining PCA with linear regression leads to principal components regression (PCR).

Principal Components Regression (PCR)

- *PCR* = PCA + linear regression:
  - Choose how many PCs to use, say, M.
  - Use PCA to define a feature vector $\varphi(x_i)$ containing the PC1,...,PCM scores for x_i.
  - Use least-squares linear regression with this model:

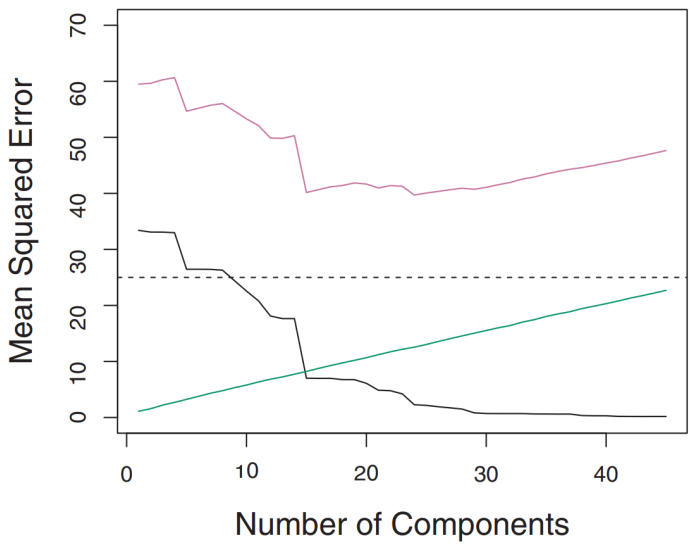
$$Y_i = \varphi(x_i)^\mathsf{T}\beta + \varepsilon_i.$$

- PCR works well when the directions in which the original predictors vary most are the directions that are predictive of the outcome.

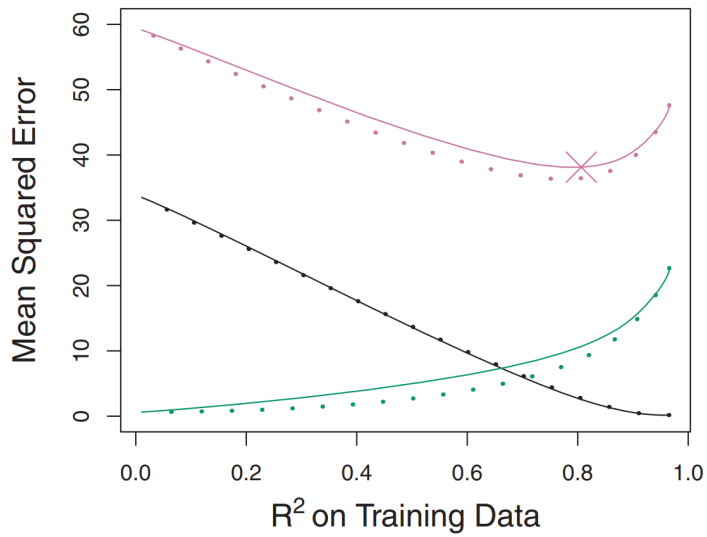
- PCR versus least-squares:
  - When $M = p$, PCR = least-squares.
  - PCR has higher bias but lower variance.
  - PCR can handle $p > n$.
  - PCR can handle some collinearity. (Why?)

Example #1: PCR versus Ridge and Lasso

Simulation example with $p = 45$ and $n = 50$.
True model is linear regression with all nonzero coefficients.



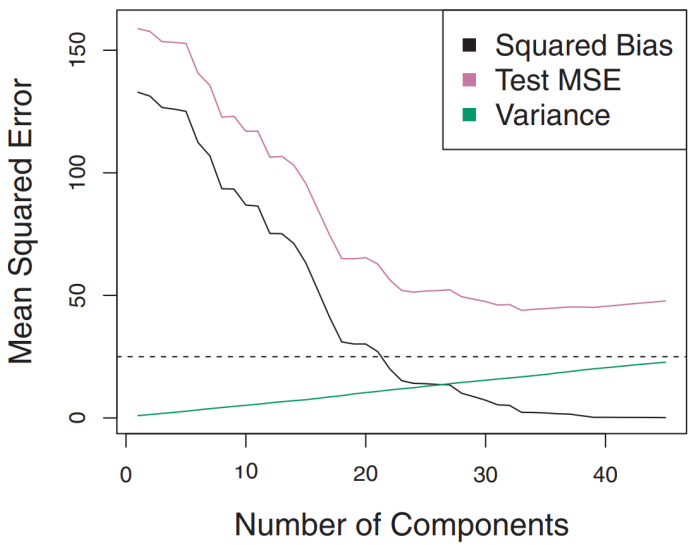
PCR



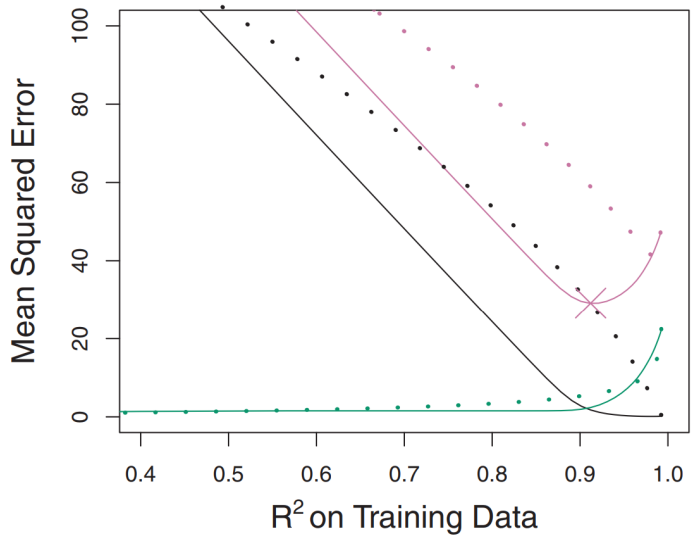
Ridge (dotted) & Lasso (solid)

Example #2: PCR versus Ridge and Lasso

Simulation example with $p = 45$ and $n = 50$.
True model is linear regression with 2 nonzero coefficients.



PCR



Ridge (dotted) & Lasso (solid)

Example #3: PCR versus Ridge and Lasso

Simulation example with $p = 45$.

True model is PCR with 5 nonzero PC coefficients.

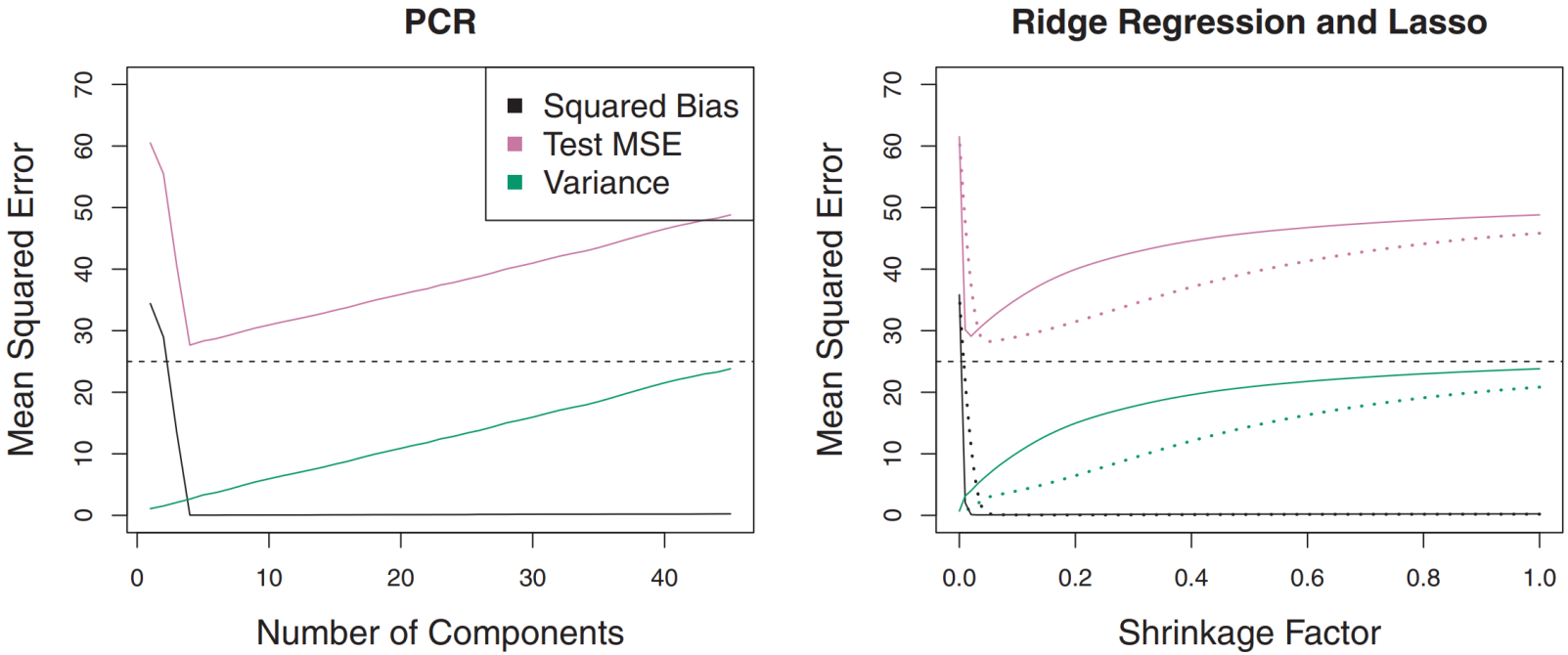


FIGURE 6.19. *PCR, ridge regression, and the lasso were applied to a simulated data set in which the first five principal components of X contain all the information about the response Y. In each panel, the irreducible error $Var(\epsilon)$ is shown as a horizontal dashed line.* Left: *Results for PCR.* Right: *Results for lasso (solid) and ridge regression (dotted). The x-axis displays the shrinkage factor of the coefficient estimates, defined as the ℓ_2 norm of the shrunken coefficient estimates divided by the ℓ_2 norm of the least squares estimate.*

PCR versus Ridge and Lasso

- PCR does not select a subset of predictors/features.
- PCR is more closely related to Ridge than Lasso.
- Ridge can be thought of as a continuous version of PCR.

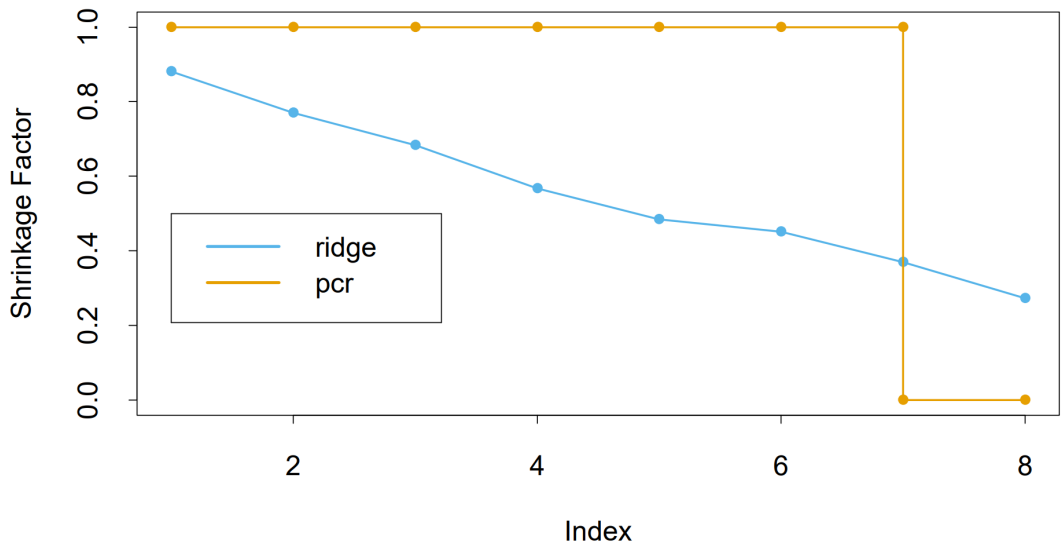


FIGURE 3.17. *Ridge regression shrinks the regression coefficients of the principal components, using shrinkage factors $d_j^2/(d_j^2 + \lambda)$ as in (3.47). Principal component regression truncates them. Shown are the shrinkage and truncation patterns corresponding to Figure 3.7, as a function of the principal component index.*

(See ESL Section 3.5 for more info.)

Cross-validation

Can choose PCR dimensionality M using cross-validation.

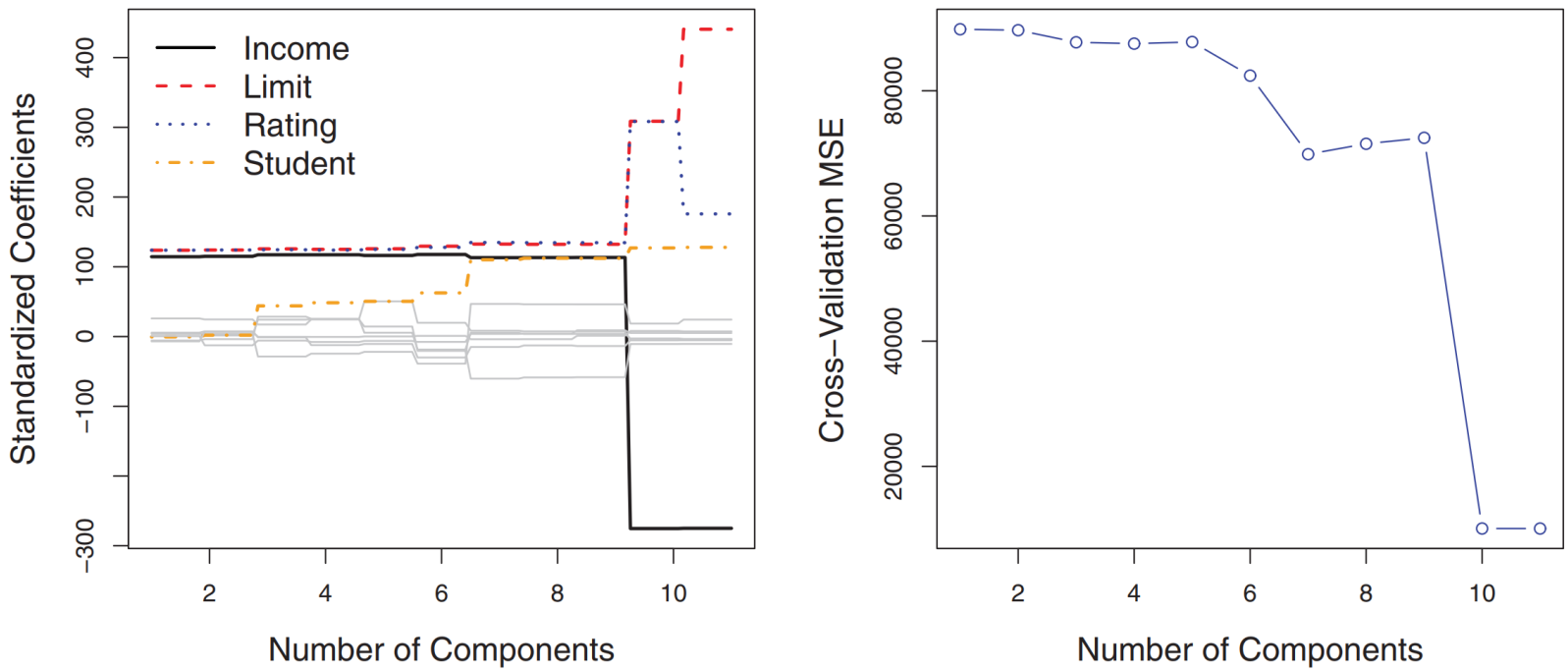


FIGURE 6.20. Left: *PCR standardized coefficient estimates on the* `Credit` *data set for different values of* M. Right: *The ten-fold cross validation MSE obtained using PCR, as a function of* M.

Outline

High-dimensional issues ($p \geq n$)

Overfitting:

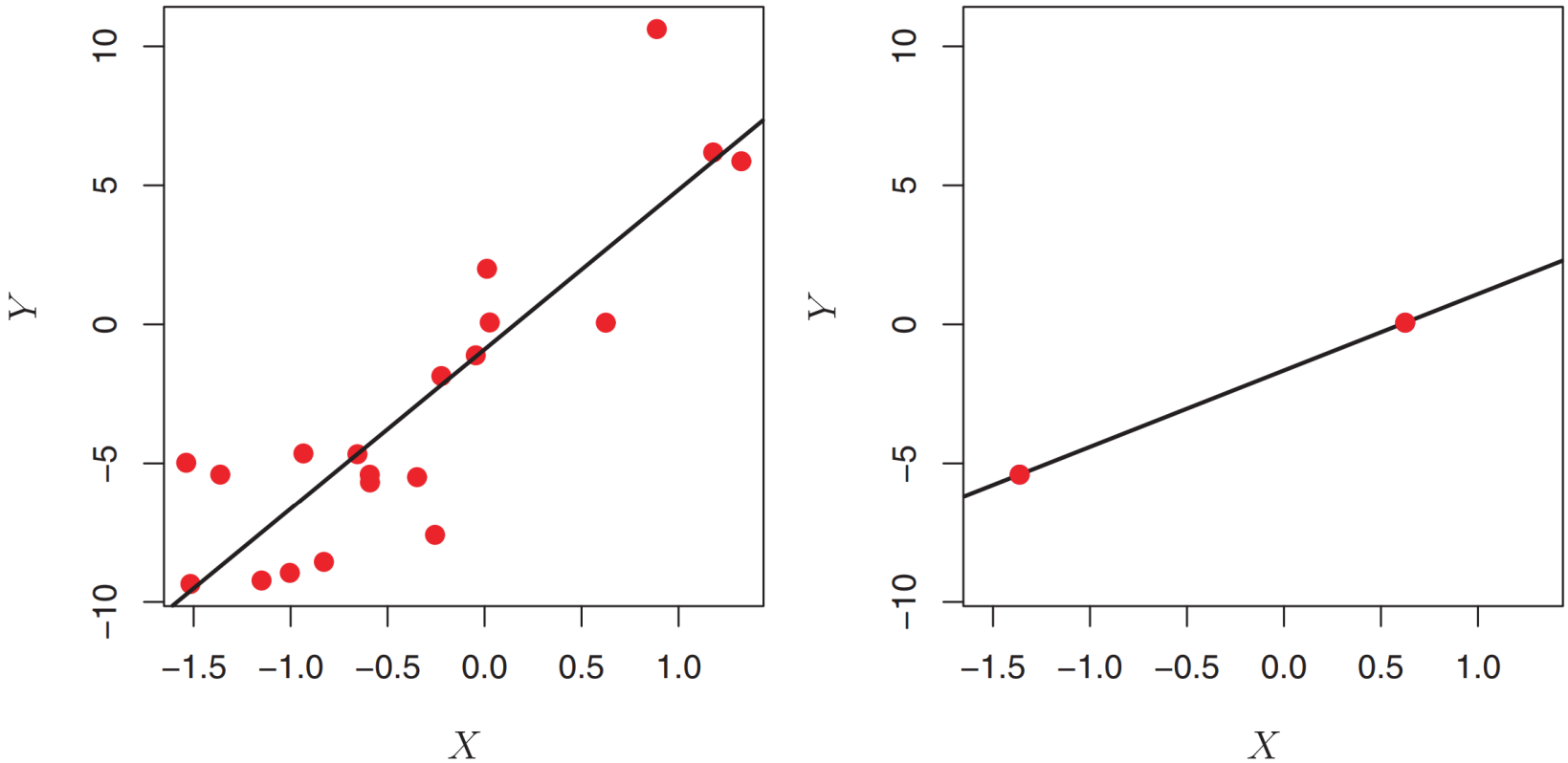


FIGURE 6.22. Left: *Least squares regression in the low-dimensional setting.* Right: *Least squares regression with $n = 2$ observations and two parameters to be estimated (an intercept and a coefficient).*

High-dimensional issues ($p \geq n$)

R^2, RSS, AIC, BIC often don't accurately assess fit (unless you reduce dimensionality first):

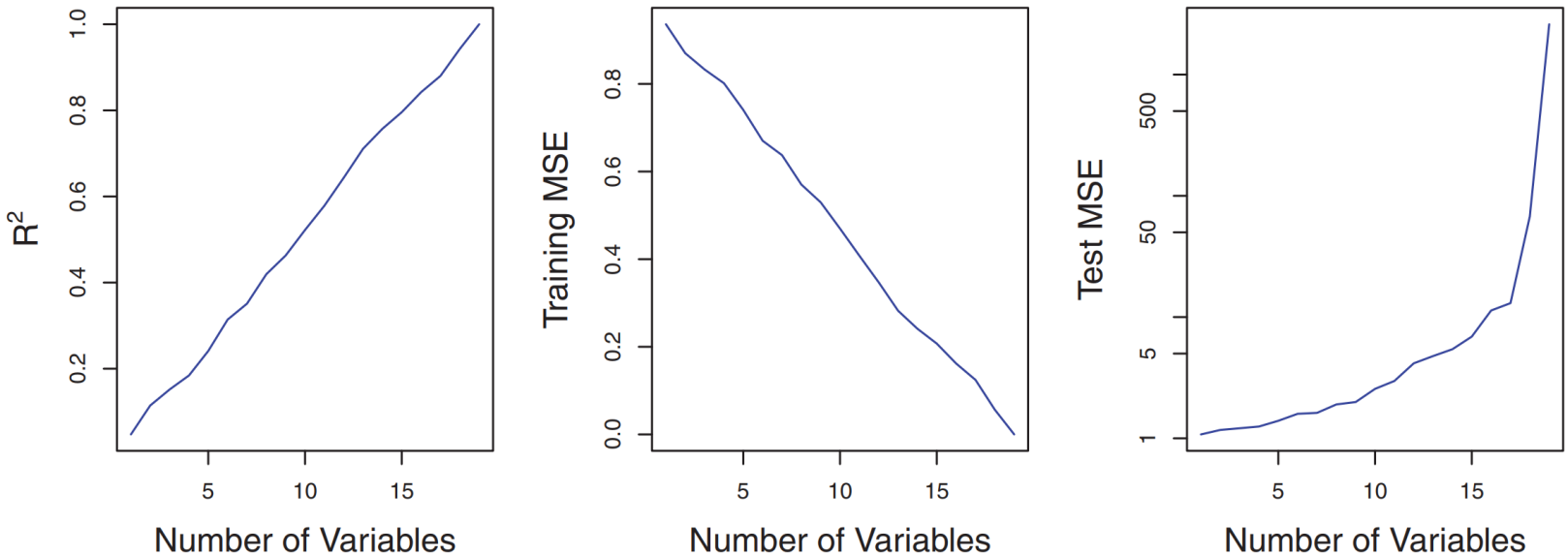


FIGURE 6.23. *On a simulated example with $n = 20$ training observations, features that are completely unrelated to the outcome are added to the model.* Left: *The R^2 increases to 1 as more features are included.* Center: *The training set MSE decreases to 0 as more features are included.* Right: *The test set MSE increases as more features are included.*

High-dimensional issues ($p \geq n$)

- Regularization or variable selection is key.
- Tuning the flexibility knob(s) is important.
- Adding uninformative features hurts performance.

- Assess performance using held-out test sets (e.g., cross-validation).

References

- James G, Witten D, Hastie T, and Tibshirani R (2013). *An Introduction to Statistical Learning*, Springer.
- Hastie T, Tibshirani R, and Friedman J (2008). *The Elements of Statistical Learning*, Springer.