Dimensionality reduction

Machine Learning for Process Engineers Workshop

Stellenbosch University

March 2022

Recap

- Goal of machine learning:

  provide accurate predictions on <u>new</u> data
- Model performance is a trade-off between bias and variance
- Bias can be introduced systematically through regularisation (sacrifice training error for testing error)

- Alternative approach to introducing bias: feature extraction

New system

- Linear state space system: inputs $u_1(t)$, $u_2(t)$, and measurement $y(t)$

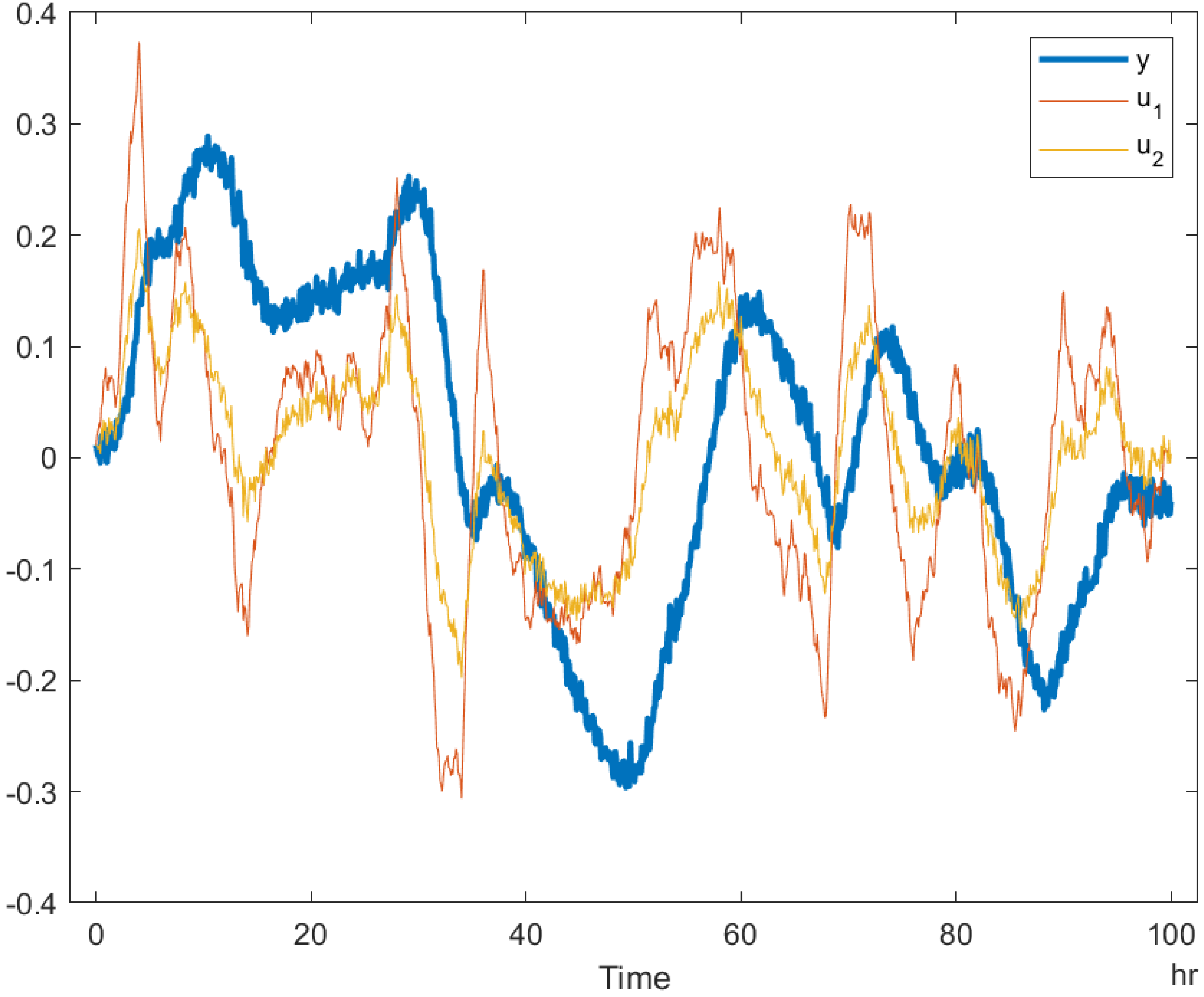
$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + B\mathbf{u}, \qquad \frac{d\mathbf{y}}{dt} = H\mathbf{x}$$

- The parameters are unknown: <u>systems identification problem</u>
- Simplified discrete form:

$$\begin{aligned}\hat{y}(k+1) = \beta_1 y(k) + \beta_2 y(k-1) \dots + \beta_L y(k-L) \\ +\beta_{L+1} u_1(k) + \beta_{L+2} u_1(k-1) \dots + \beta_{2L} u_1(k-L) \\ +\beta_{2L+1} u_2(k) + \beta_{2L+2} u_2(k-1) \dots + \beta_{3L} u_2(k-L)\end{aligned}$$

- System has L “lags”

New system



New system

$$\hat{y}(L+2) = [y(L+1) \dots y(1) \;\; u_1(L+1) \;\; \dots \;\; u_1(1) \;\; u_2(L+1) \;\; \dots \;\; u_2(1)]\boldsymbol{\beta}$$

$$\hat{y}(L+3) = [y(L+2) \dots y(2) \;\; u_1(L+2) \;\; \dots \;\; u_1(2) \;\; u_2(L+2) \;\; \dots \;\; u_2(2)]\boldsymbol{\beta}$$

$$\hat{y}(L+4) = [y(L+3) \dots y(3) \;\; u_1(L+3) \;\; \dots \;\; u_1(3) \;\; u_2(L+3) \;\; \dots \;\; u_2(3)]\boldsymbol{\beta}$$

$$\vdots$$

$$\mathbf{y}_{L+2:N} = \mathbf{X}\boldsymbol{\beta}$$

New system

We created a special MATLAB/Python function

```
[X,y] = CreateLaggedDesignMatrix(Data, L, f)
```

to create $\mathbf{y}$ and $\mathbf{X}$

given time data `Data`, number of lags `L` and the fraction of data to use `f`

$$\mathbf{y}_{L+2:N} = \mathbf{X}\boldsymbol{\beta}$$

Prediction vs simulation

- "One-step ahead" prediction problem:

$$\hat{y}(k+1) = \beta_1 y(k) + \beta_2 y(k-1) \ldots + \beta_L y(k-L)$$
$$+\beta_{L+1} u_1(k) + \beta_{L+2} u_1(k-1) \ldots + \beta_{2L} u_1(k-L)$$
$$+\beta_{2L+1} u_2(k) + \beta_{2L+2} u_2(k-1) \ldots + \beta_{3L} u_2(k-L)$$

- Simulation problem

$$\hat{\boldsymbol{y}}(\boldsymbol{k}+\mathbf{1}) = \beta_1 \hat{\boldsymbol{y}}(\boldsymbol{k}) + \beta_2 \hat{\boldsymbol{y}}(\boldsymbol{k}-\mathbf{1}) \ldots + \beta_L \hat{\boldsymbol{y}}(\boldsymbol{k}-\boldsymbol{L})$$
$$+\beta_{L+1} u_1(k) + \beta_{L+2} u_1(k-1) \ldots + \beta_{2L} u_1(k-L)$$
$$+\beta_{2L+1} u_2(k) + \beta_{2L+2} u_2(k-1) \ldots + \beta_{3L} u_2(k-L)$$

- Simulation is more difficult

Prediction vs simulation

We created a special MATLAB/Python function

`y = PredictTimeSeries(mdl, Data, L)`

to generate predictions `y`

given a model `mdl`, time data `Data`, and number of lags `L`

The model consists of `mdl.beta` containing the vector of coefficients $\boldsymbol{\beta}$ and `mdl.Q` containing a transformation matrix $\boldsymbol{Q}$ (to be discussed)

New system

- Simplified discrete form:

$$\boldsymbol{\hat{y}(k+1)} = \beta_1 \boldsymbol{\hat{y}(k)} + \beta_2 \boldsymbol{\hat{y}(k-1)} \ldots + \beta_L \boldsymbol{\hat{y}(k-L)}$$
$$+\beta_{L+1} u_1(k) + \beta_{L+2} u_1(k-1) \ldots + \beta_{2L} u_1(k-L)$$
$$+\beta_{2L+1} u_2(k) + \beta_{2L+2} u_2(k-1) \ldots + \beta_{3L} u_2(k-L)$$

- System above has L "lags"
  (often used when modelling time series data)
- Not sure how many lags are required
- Consecutive data points are highly correlated

In MATLAB/Python: Example 10

- Open file "`MLforProcEng_Workshop_3.m`" and run the **`%% Initialize`**
- Go to the cell
  `%% Example 10:  Linear model fit to timeseries data`

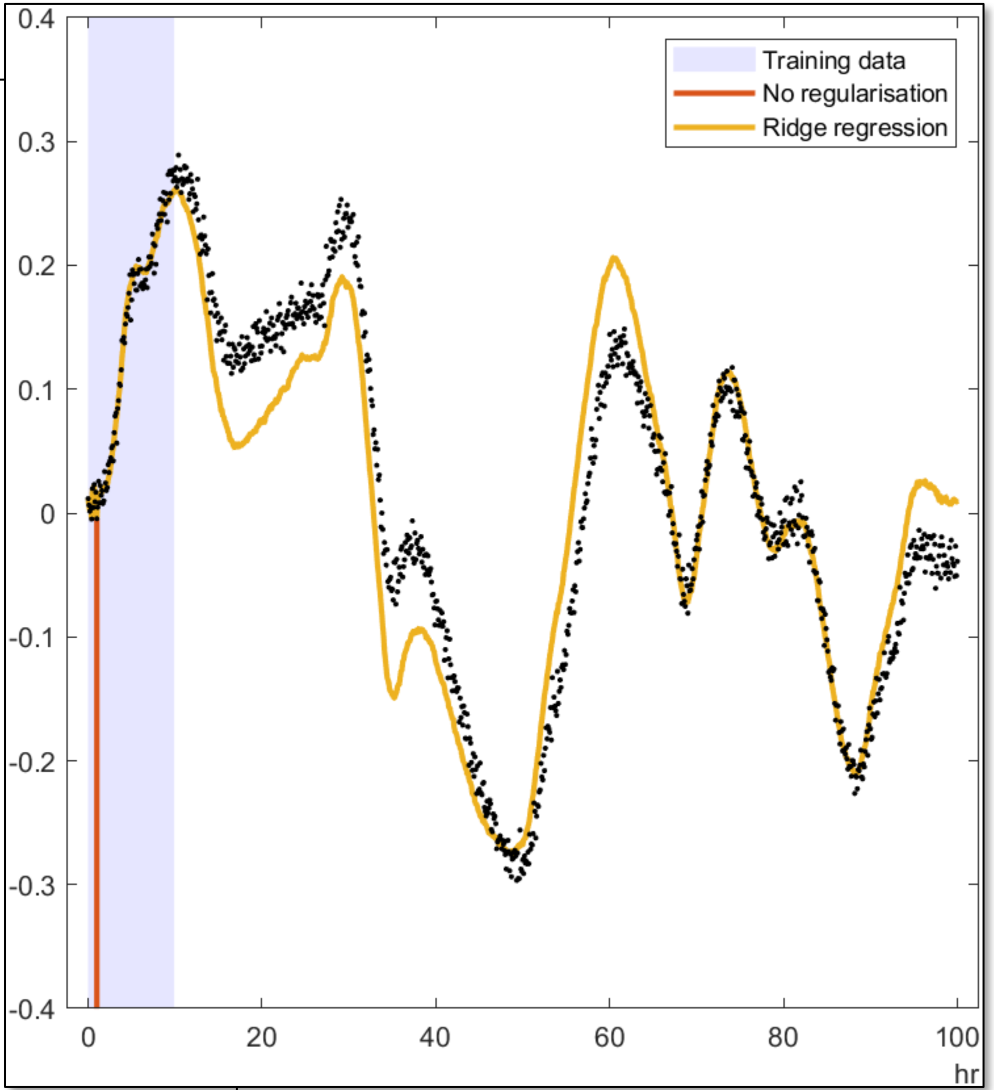
- The code is ready to run and needs no adjustment
- Note the construction of the "`mdl`" object for use in "`PredictTimeSeries`"

  `mdl.Q = 1` in both cases (for now)

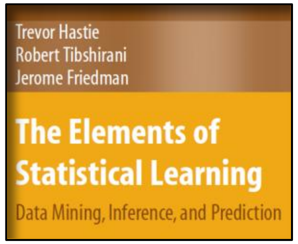
```matlab
%% Initialize
⋮
load ProcessData

% Prepare the design matrix "X"
L = 10;
⋮
[X, y] = CreateLaggedDesignMatrix(Data, L, 0.1);

%% Example 10: Linear model fit to timeseries data
% Fit a linear model without regularization
mdl = fitlm(X, y, 'Intercept',false);
linear_mdl.Q = 1;
linear_mdl.beta = mdl.Coefficients.Estimate;
y_linear = PredictTimeSeries(linear_mdl, Data, L);

% Fit a linear model with ridge regression
?
?
?
⋮
```

```matlab
%% Initialize
⋮
load ProcessData

% Prepare the design matrix "X"
L = 10;
⋮
[X, y] = CreateLaggedDesignMatrix(Data, L, 0.1);

%% Example 10: Linear model fit to timeseries data
% Fit a linear model without regularization
mdl = fitlm(X, y, 'Intercept',false);
linear_mdl.Q = 1;
linear_mdl.beta = mdl.Coefficients.Estimate;
y_linear = PredictTimeSeries(linear_mdl, Data, L);

% Fit a linear model with ridge regression
ridge_mdl.Q = 1;
ridge_mdl.beta = lasso(X, y, 'Alpha', 1e-6, 'Lambda', 0.1);
y_ridge = PredictTimeSeries(ridge_mdl, Data, L);
⋮
```

```
%% Initialize
⋮
load ProcessData

% Prepare the design matrix "X"
L = 10;
⋮
[X, y] = CreateLaggedDesignMatrix(Data, L, 0.1);

%% Example 10: Linear model fit to timeseries data
% Fit a linear model without regularization
mdl = fitlm(X, y, 'Intercept',false);
linear_mdl.Q = 1;
linear_mdl.beta = mdl.Coefficients.Estimate;
y_linear = PredictTimeSeries(linear_mdl, Data, L);

% Fit a linear model with ridge regression
ridge_mdl.Q = 1;
ridge_mdl.beta = lasso(X, y, 'Alpha', 1e-6, 'Lambda', 0.1);
y_ridge = PredictTimeSeries(ridge_mdl, Data, L);
⋮
```



Correlation in predictors



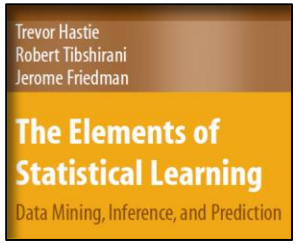
p 45

- The linear regression problem solves the following:

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

- If two predictors are perfectly correlated (e.g. $\mathbf{X}_2 = \gamma\mathbf{X}_1$) then $\mathbf{X}^T\mathbf{X}$ is singular
- If two predictors are closely correlated (e.g. $\mathbf{X}_2 \approx \gamma\mathbf{X}_1$) then $\mathbf{X}^T\mathbf{X}$ is ill-conditioned

Correlation in predictors



p 65

- The linear regression problem solves the following:

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

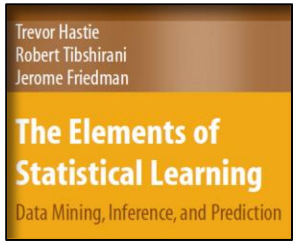
- If two predictors are perfectly correlated (e.g. $\mathbf{X}_2 = \gamma\mathbf{X}_1$) then $\mathbf{X}^T\mathbf{X}$ is singular
- If two predictors are closely correlated (e.g. $\mathbf{X}_2 \approx \gamma\mathbf{X}_1$) then $\mathbf{X}^T\mathbf{X}$ is ill-conditioned

- The ridge regression problem solves the following:

$$\mathbf{X}^T\mathbf{y} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\boldsymbol{\beta}$$

- Addition of the λ on the diagonal conditions the matrix

Correlation in predictors



p 65

- The linear regression problem solves the following:

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

- If two predictors are perfectly correlated (e.g. $\mathbf{X}_2 = \gamma\mathbf{X}_1$) then $\mathbf{X}^T\mathbf{X}$ is singular
- If two predictors are closely correlated (e.g. $\mathbf{X}_2 \approx \gamma\mathbf{X}_1$) then $\mathbf{X}^T\mathbf{X}$ is ill-conditioned

- The ridge regression problem solves the following:

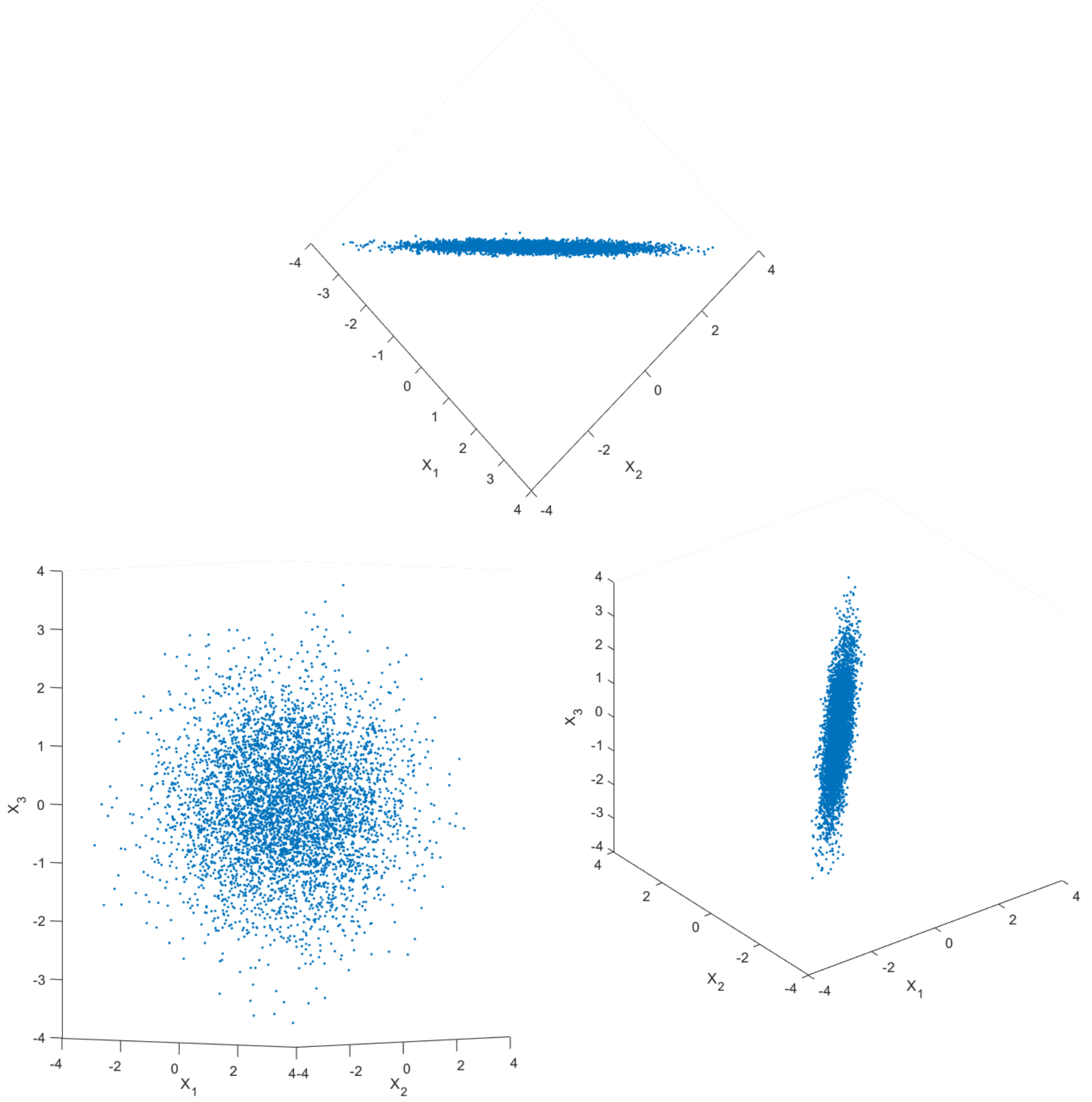
$$\mathbf{X}^T\mathbf{y} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\boldsymbol{\beta}$$

- Addition of the λ on the diagonal conditions the matrix

Interestingly "`lasso(X, y, 'Lambda', 0);`" yields *much* better results than "`fitlm(X, y);`" due to the underlying numerical method

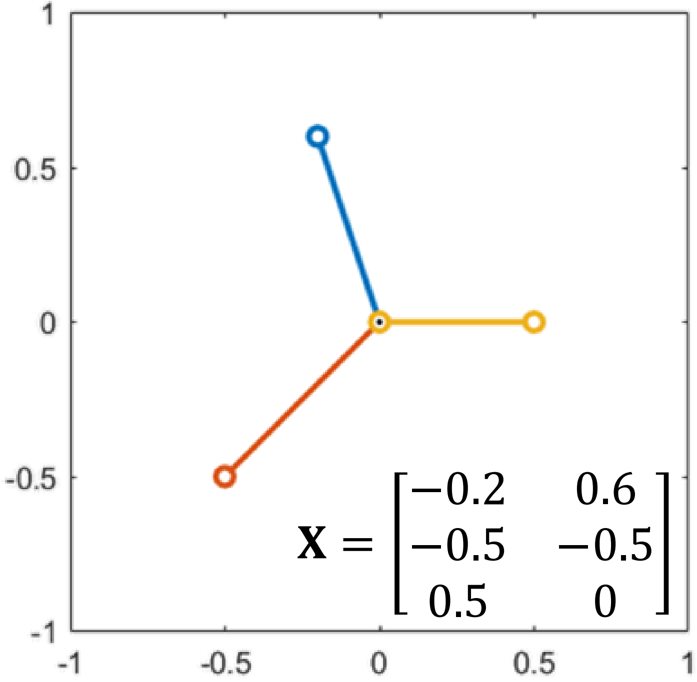
Projection of predictors

- Can we “combine” correlated variables into single components?



Orthonormal matrix as "rotation"

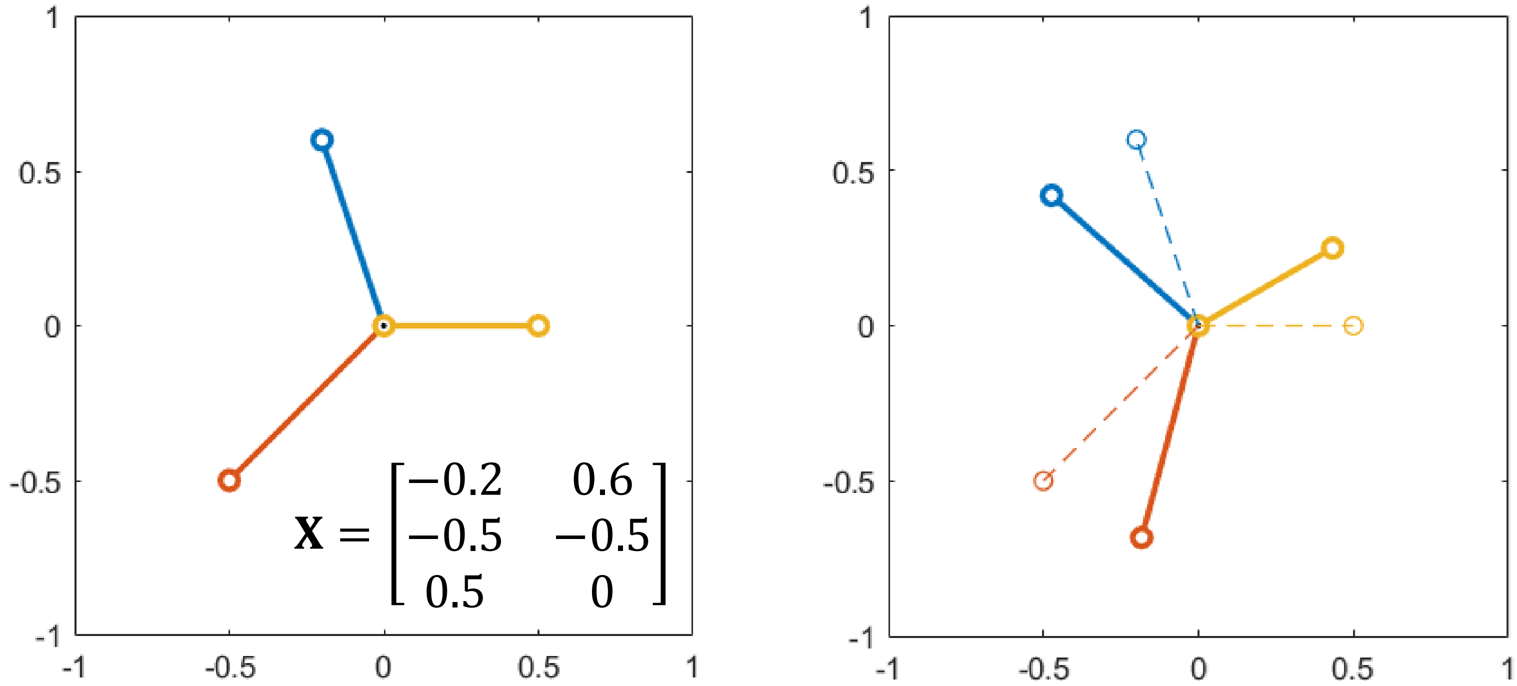
- Matrices can project vectors to a new set of basis functions
- Orthonormal matrices amounts to a rotation of the basis vectors



$$\mathbf{X} = \begin{bmatrix} -0.2 & 0.6 \\ -0.5 & -0.5 \\ 0.5 & 0 \end{bmatrix}$$

Orthonormal matrix as "rotation"

- Matrices can project vectors to a new set of basis functions
- Orthonormal matrices amounts to a rotation of the basis vectors

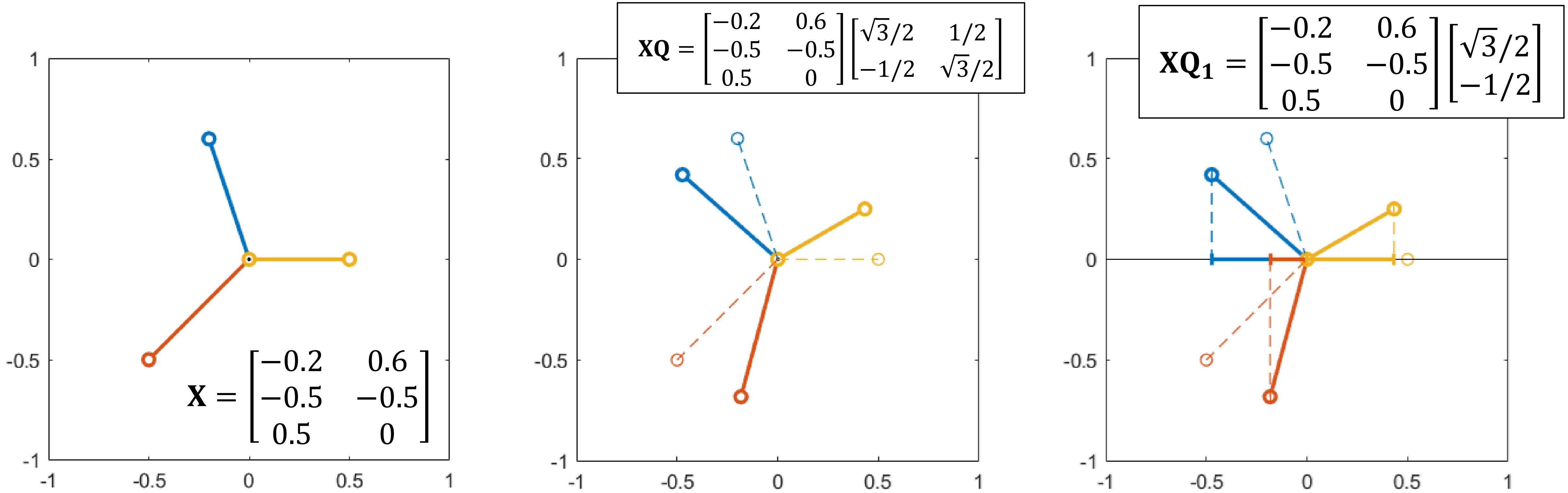


$$\mathbf{X} = \begin{bmatrix} -0.2 & 0.6 \\ -0.5 & -0.5 \\ 0.5 & 0 \end{bmatrix}$$

$$\mathbf{XQ} = \begin{bmatrix} -0.2 & 0.6 \\ -0.5 & -0.5 \\ 0.5 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 & 1/2 \\ -1/2 & \sqrt{3}/2 \end{bmatrix}$$

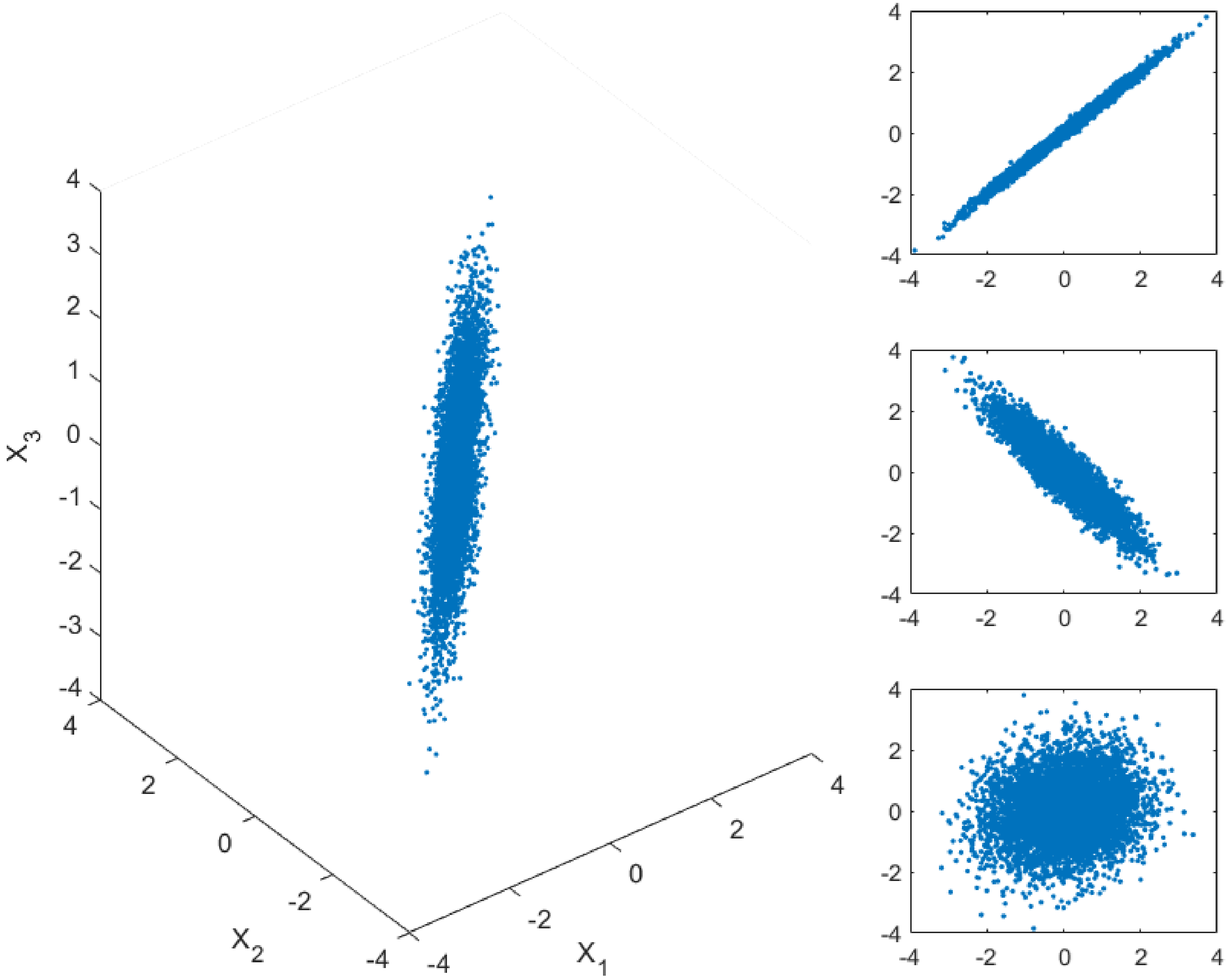
Orthonormal matrix as “rotation”

- Matrices can project vectors to a new set of basis functions
- Orthonormal matrices amounts to a rotation of the basis vectors

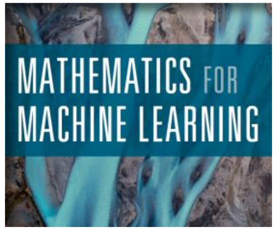


Orthonormal matrix as "rotation"

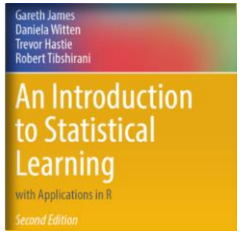
- What "projection" (rotation, eliminating dimensions) of the data yields a reduced dimension data set with the greatest *variance*?



Orthonormal matrix as "rotation"



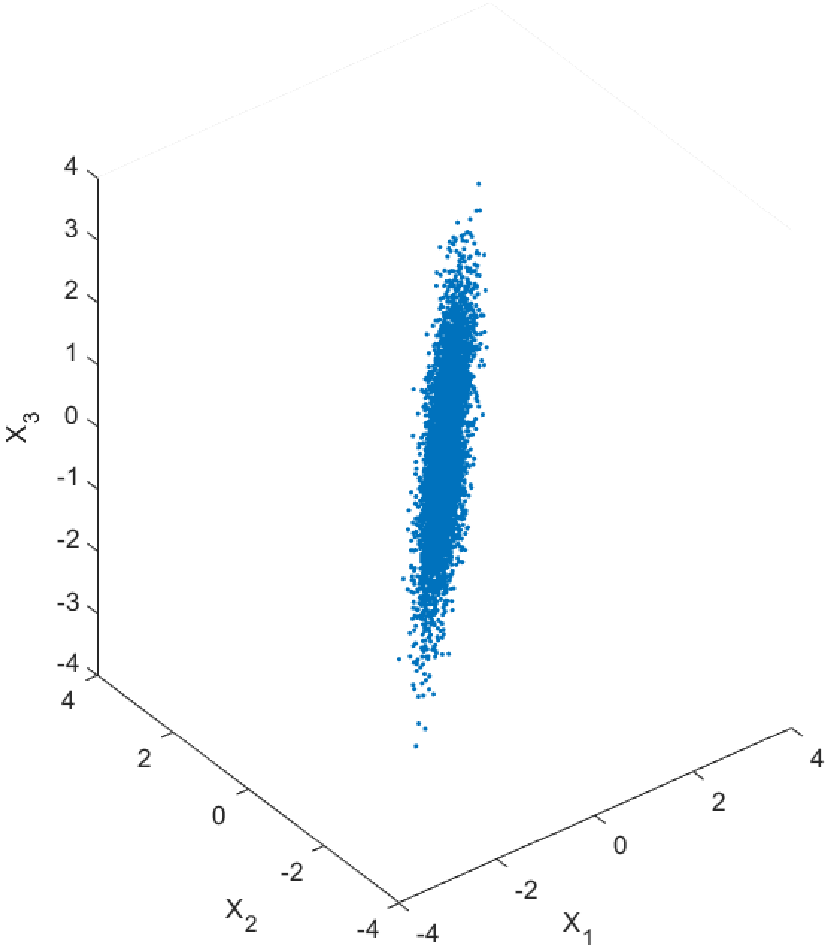
p 115

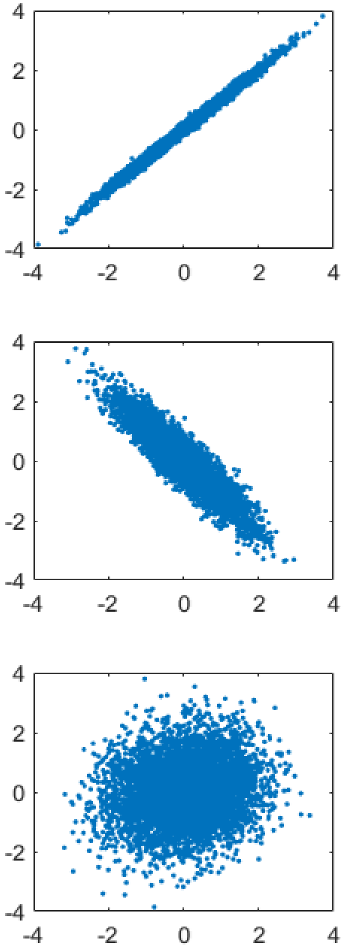


p 498

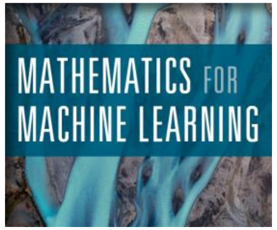
- What "projection" (rotation, eliminating dimensions) of the data yields a reduced dimension data set with the greatest *variance*?

- **Principal Component Analysis (PCA)**: Use the *eigenvectors* $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ … of the *covariance matrix* $\mathbf{X}^T\mathbf{X}$ corresponding to the largest *eigenvalues* $\lambda_1, \lambda_2, \lambda_3$ …

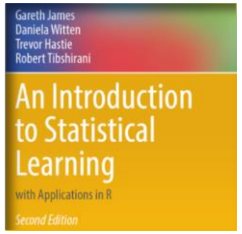




Orthonormal matrix as “rotation”

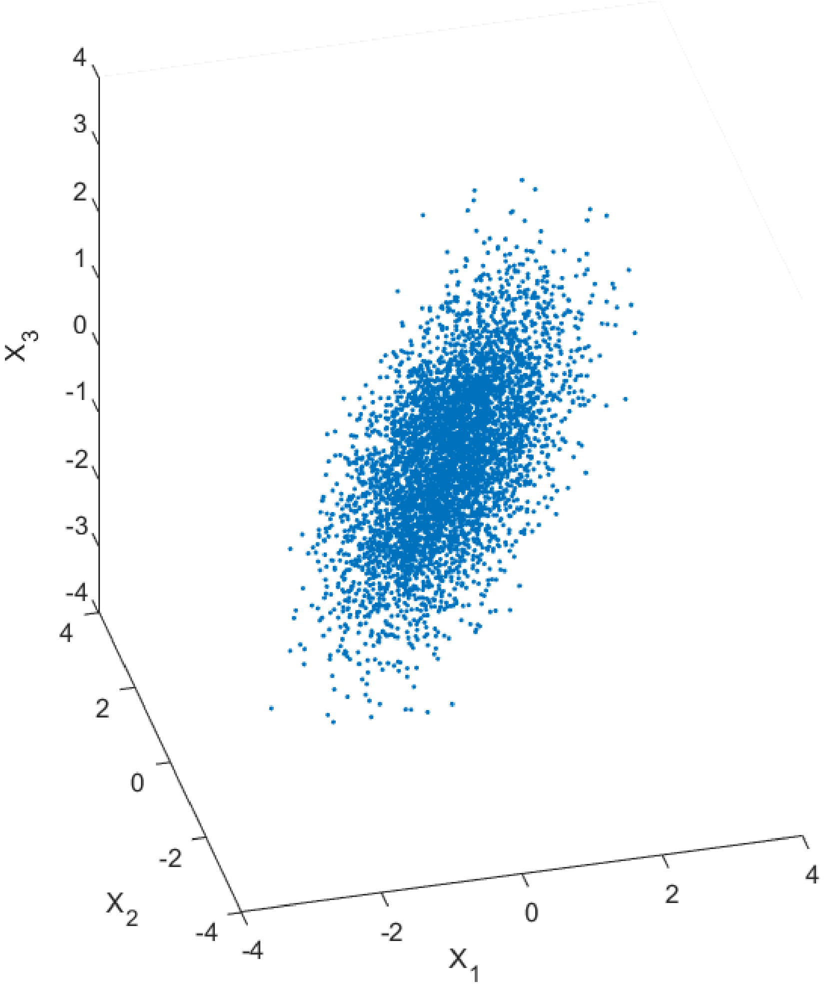


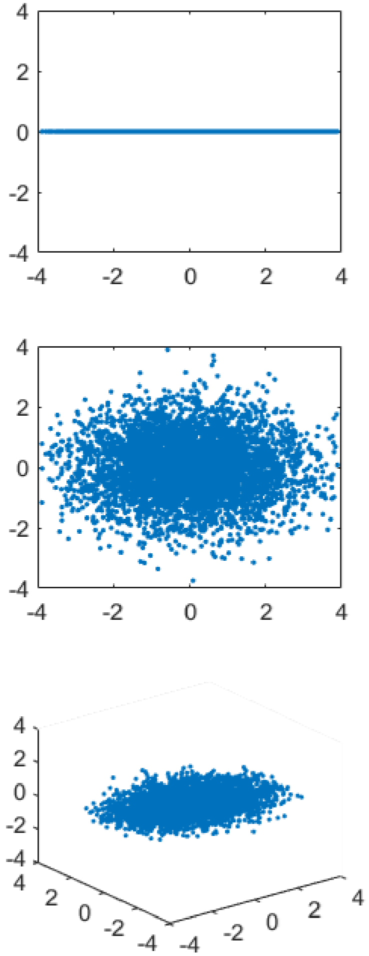
p 115



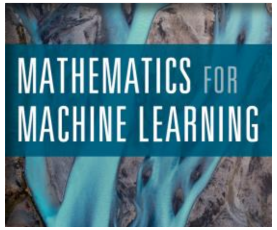
p 498

- **Principal Component Analysis (PCA)**: Use the *eigenvectors* $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ … of the *covariance matrix* $\mathbf{X}^T\mathbf{X}$ corresponding to the largest *eigenvalues* $\lambda_1, \lambda_2, \lambda_3$ …

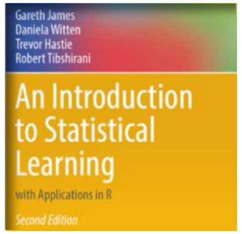




Orthonormal matrix as "rotation"



p 115



p 498

- **Principal Component Analysis (PCA)**: Use the *eigenvectors* $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ … of the *covariance matrix* $\mathbf{X}^T\mathbf{X}$ corresponding to the largest *eigenvalues* $\lambda_1, \lambda_2, \lambda_3$ …

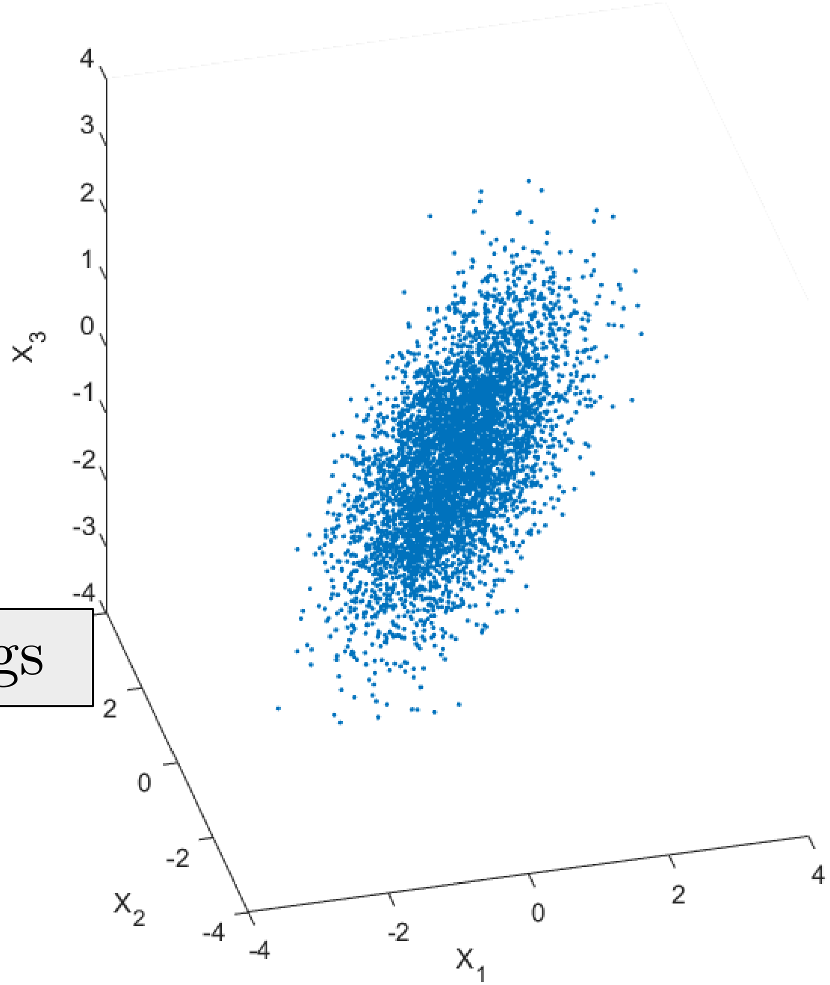
$$\mathbf{T} = \mathbf{X}\mathbf{Q}_{1:m} = \mathbf{X}[\mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \dots \mathbf{q}_m]$$

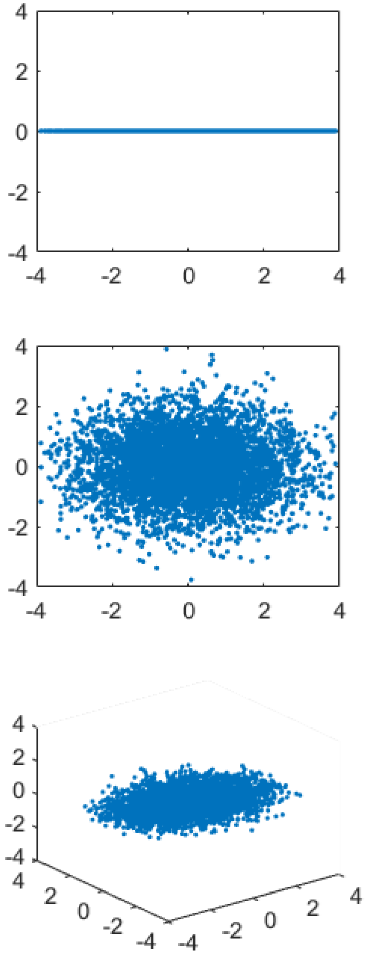
PCA scores

Design matrix

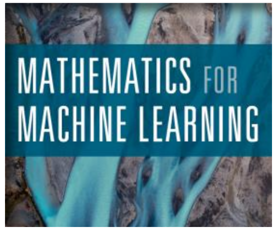
PCA loadings

$$\{N \times m\} = \{N \times p\}\{p \times m\}$$

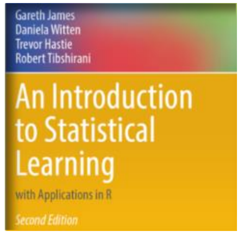




Orthonormal matrix as “rotation”



p 115



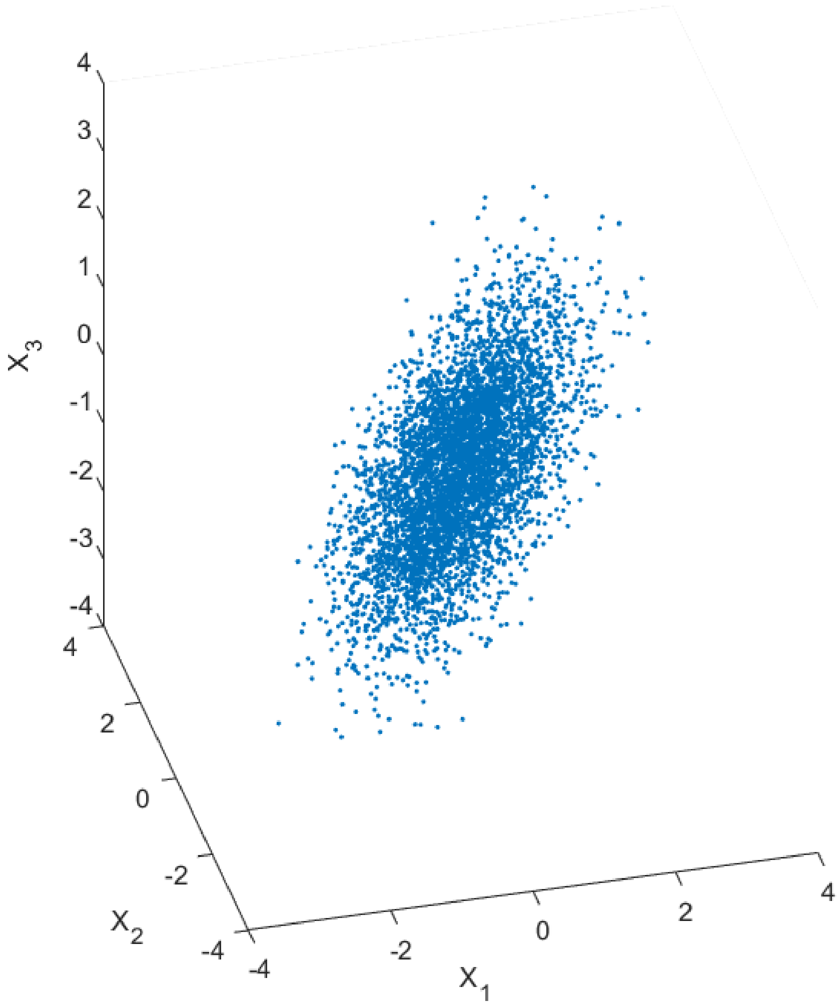
p 498

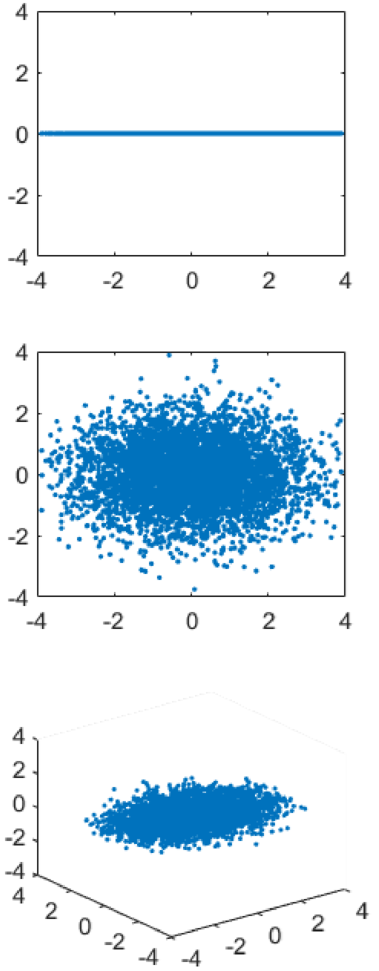
- **Principal Component Analysis (PCA)**: Use the *eigenvectors* $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ … of the *covariance matrix* $\mathbf{X}^T\mathbf{X}$ corresponding to the largest *eigenvalues* $\lambda_1, \lambda_2, \lambda_3$ …

$$\mathbf{T} = \mathbf{X}\mathbf{Q}_{1:m} = \mathbf{X}[\mathbf{q}_1 \ \ \mathbf{q}_2 \ \ \dots \mathbf{q}_m]$$

- Because the covariance matrix $\mathbf{X}^T\mathbf{X}$ is symmetric, the eigenvalues are orthonormal:

$$\mathbf{X}\mathbf{Q} \rightarrow \textit{rotation}$$





Orthonormal matrix as “rotation”

$$\mathbf{T} = \mathbf{X}[\mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \dots \mathbf{q}_m]$$

$$\begin{bmatrix} \vdots & \vdots \\ t_{i-1,1} & t_{i-1,2} \\ t_{i,1} & t_{i,2} \\ t_{i+1,1} & t_{i+1,1} \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ x_{i-1,1} & x_{i-1,2} & x_{i-1,3} \\ x_{i,1} & x_{i,2} & x_{i,3} \\ x_{i+1,1} & x_{i+1,2} & x_{i+1,3} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} q_{1,1} & q_{2,1} \\ q_{1,2} & q_{2,2} \\ q_{1,3} & q_{2,3} \end{bmatrix}$$

Orthonormal matrix as "rotation"

$$\mathbf{T} = \mathbf{X}[\mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \dots \mathbf{q}_m]$$

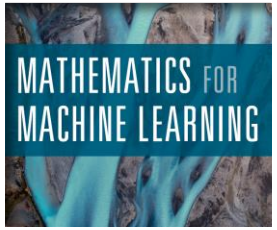
$$\begin{bmatrix} \vdots & \vdots \\ t_{i-1,1} & t_{i-1,2} \\ t_{i,1} & t_{i,2} \\ t_{i+1,1} & t_{i+1,1} \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ x_{i-1,1} & x_{i-1,2} & x_{i-1,3} \\ x_{i,1} & x_{i,2} & x_{i,3} \\ x_{i+1,1} & x_{i+1,2} & x_{i+1,3} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} q_{1,1} & q_{2,1} \\ q_{1,2} & q_{2,2} \\ q_{1,3} & q_{2,3} \end{bmatrix}$$

Orthonormal matrix as "rotation"

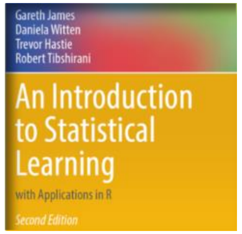
$$\mathbf{T} = \mathbf{X}[\mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \dots \mathbf{q}_m]$$

$$\begin{bmatrix} \vdots & \vdots \\ t_{i-1,1} & t_{i-1,2} \\ t_{i,1} & t_{i,2} \\ t_{i+1,1} & t_{i+1,1} \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ x_{i-1,1} & x_{i-1,2} & x_{i-1,3} \\ x_{i,1} & x_{i,2} & x_{i,3} \\ x_{i+1,1} & x_{i+1,2} & x_{i+1,3} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} q_{1,1} & q_{2,1} \\ q_{1,2} & q_{2,2} \\ q_{1,3} & q_{2,3} \end{bmatrix}$$

Principal component regression



p 317



p 251

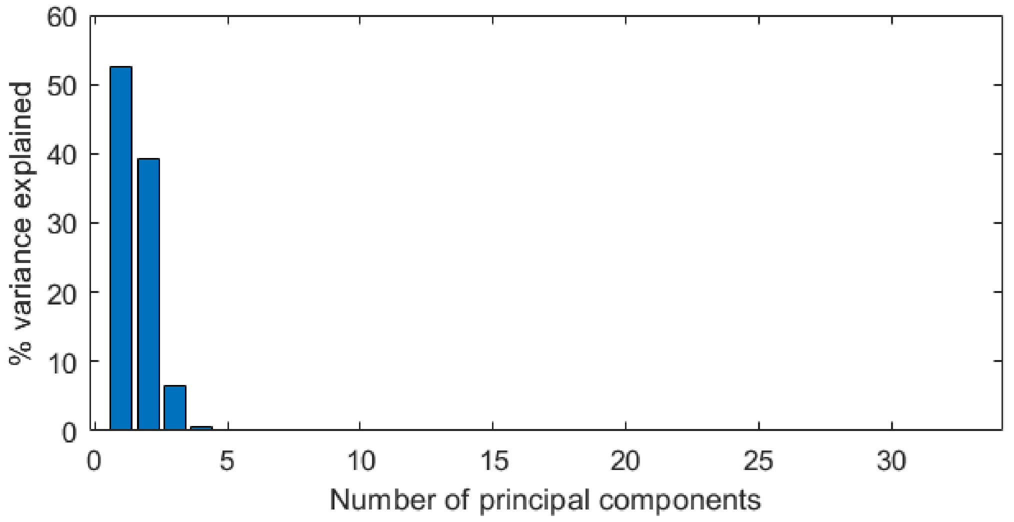
1. Project data to a lower dimensional space using the first $m < p$ principal component loadings, $\mathbf{Q}_{1:m}$, with $\mathbf{q}_i$ ordered according to $\lambda_1 > \lambda_2 > \lambda_3 \ldots$
2. Regress response variables onto m principal component scores

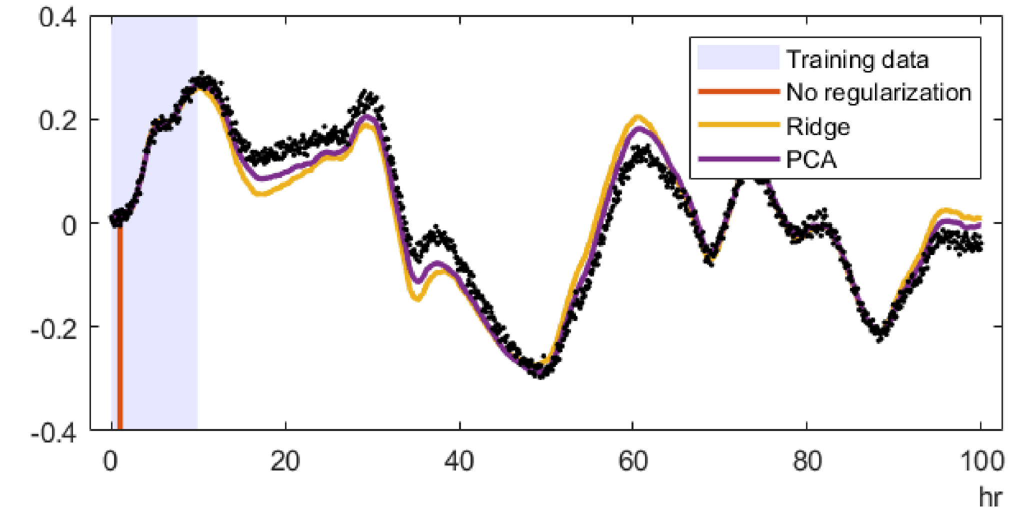
$$\mathbf{y} = \mathbf{X}\mathbf{Q}_{1:m}\boldsymbol{\beta}$$

- Correlated variables are combined into single features
- Fewer predictors, fewer parameters, decrease model variance

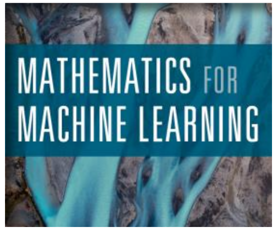
```
%% Example 11: Use PCA regression to predict time series
[loadings, ~, ~, ~, explained]  = pca(X, 'NumComponents',20);

% Plot the variance explained…
clf
subplot(2,1,1)
bar(explained);
⋮

% Fit the linear model to the reduced
% set of predictors X*Q
PCA_mdl.Q = loadings(:, 1:4);
T = X*PCA_mdl.Q;
mdl = fitlm(T, y, 'Intercept', false);
PCA_mdl.beta = mdl.Coefficients.Estimate;

% Simulate the model response
y_PCA = PredictTimeSeries(PCA_mdl, Data, L);

% Plot and compare the results
⋮
```

```
%% Example 11: Use PCA regression to predict time series
[loadings, ~, ~, ~, explained]  = pca(X, 'NumComponents',20);

% Plot the variance explained…
clf
subplot(2,1,1)
bar(explained);
⋮

% Fit the linear model to the reduced
% set of predictors X*Q
PCA_mdl.Q = loadings(:, 1:4);
T = X*PCA_mdl.Q;
mdl = fitlm(T, y, 'Intercept', false);
PCA_mdl.beta = mdl.Coefficients.Estimate;

% Simulate the model response
y_PCA = PredictTimeSeries(PCA_mdl, Data, L);

% Plot and compare the results
⋮
```

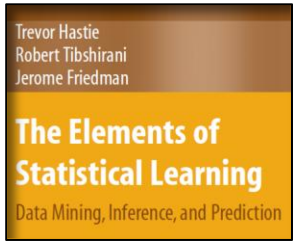




Principal component regression



p 317



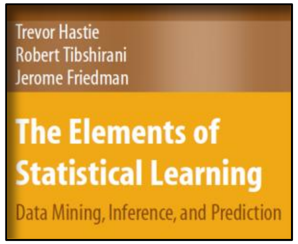
p 79

1. Project data to a lower dimensional space using the first $m < p$ principal component loadings, $\mathbf{Q}_{1:m}$, with $\mathbf{q}_i$ ordered according to $\lambda_1 > \lambda_2 > \lambda_3 \ldots$
2. Regress response variables onto m principal component scores

$$\mathbf{y} = \mathbf{X}\mathbf{Q}_{1:m}\boldsymbol{\beta}$$

- Correlated variables are combined into single features
- Fewer predictors, fewer parameters, decrease model variance

Principal component regression



p 81

Find components $\mathbf{q}_i$ such that:

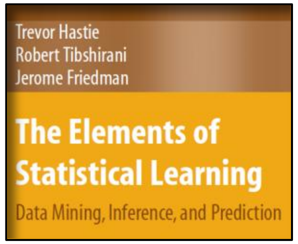
$$\mathbf{q}_i = \arg\max_{\mathbf{v}}\{\mathrm{Var}(\mathbf{X}\mathbf{v})\} = \arg\max_{\mathbf{v}}\{(\mathbf{X}\mathbf{v}) \cdot (\mathbf{X}\mathbf{v})\},$$

$$\text{s.t.}\ \ |\mathbf{v}| = 1\ \ (\mathbf{X}\mathbf{v}) \cdot \left(\mathbf{X}\mathbf{q}_j\right) = \mathbf{0}\ \ \forall\, j < i$$

The input is selected to maximize variance in the input

The response $\mathbf{y}$ is not considered in constructing the input directions

Partial least squares regression



p 81

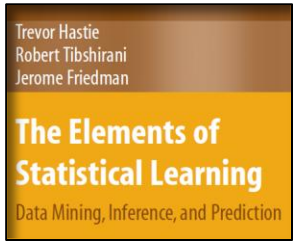
Find components $\mathbf{q}_i$ such that:

$$\mathbf{q}_i = \arg\max_{\mathbf{v}}\left\{\mathrm{Corr}^2(\mathbf{y},\mathbf{X}\mathbf{v})\mathrm{Var}(\mathbf{X}\mathbf{v})\right\}$$
$$= \arg\max_{\mathbf{v}}\{(\mathbf{y})\cdot(\mathbf{X}\mathbf{v})\times(\mathbf{X}\mathbf{v})\cdot(\mathbf{X}\mathbf{v})\},$$

$$\text{s.t. } |\mathbf{v}| = 1 \;\; (\mathbf{X}\mathbf{v})\cdot\left(\mathbf{X}\mathbf{q}_j\right) = \mathbf{0} \;\; \forall\, j < i$$

The input is selected to maximize correlation with the response $\mathbf{y}$ as well as variance in the input

Partial least squares regression



p 81

1. Set $i = 0$
2. Let $\widetilde{\mathbf{X}}^{(0)} = \mathbf{X}$
3. Let $i \leftarrow i + 1$
4. Obtain loading direction:

$$\mathbf{q}_i = \sum_{j=1}^{p} \left(\widetilde{\mathbf{X}}_j^{(i-1)} \cdot \mathbf{y} \right) \widetilde{\mathbf{X}}_j^{(i-1)}$$

5. Regress coefficient:

$$\beta_i \leftarrow \mathbf{y} = \widetilde{\mathbf{X}}^{(i-1)} \mathbf{q}_i \beta_i$$

6. Generate design matrix orthogonal to $\mathbf{q}_i$:

$$\widetilde{\mathbf{X}}_j^{(i)} = \widetilde{\mathbf{X}}_j^{(i-1)} - \left(\frac{\mathbf{q}_i \cdot \widetilde{\mathbf{X}}_j^{(i-1)}}{\mathbf{q}_i \cdot \mathbf{q}_i} \right) \mathbf{q}_i$$

7. Repeat steps 3-7 until $i = m$

Partial least squares regression

- MATLAB implementation relies on slightly different algorithm (SIMPLS)

https://doi.org/10.1016/0169-7439(93)85002-X

ELSEVIER

Chemometrics and Intelligent Laboratory Systems

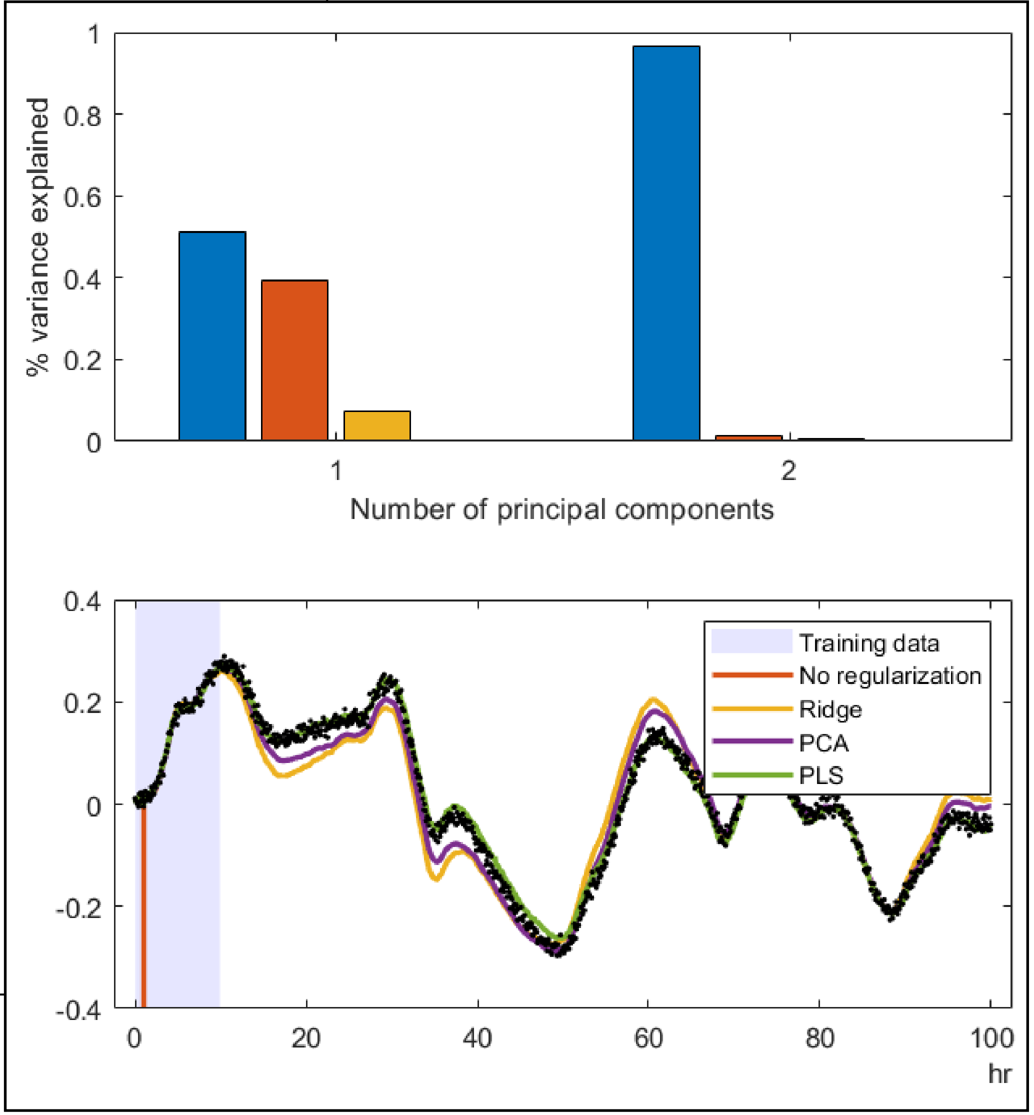
Volume 18, Issue 3, March 1993, Pages 251-263

SIMPLS: An alternative approach to partial least squares regression

Sijmen de Jong

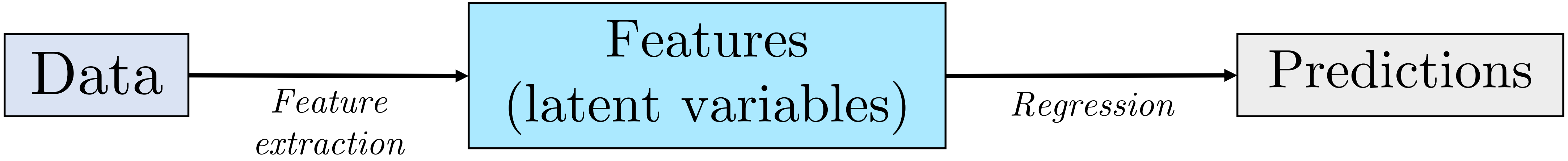
```
%% Example 12: Use PLS regression to predict time series
[loadings, ~, ~, ~, ~, explained]  = plsregress(X, y,4);

% Plot the variance explained…
?
?
?

% Fit the linear model…
?
?
?
?

% Simulate the model response
?

% Plot and compare the results
⋮
```

```
%% Example 12: Use PLS regression to predict time series
[loadings, ~, ~, ~, ~, explained]  = plsregress(X, y,4);

% Plot the variance explained…
clf
subplot(2,1,1)
bar(explained);

% Fit the linear model…
PLS_mdl.Q = loadings(:, 1:4);
T = X*PLS_mdl.Q;
mdl = fitlm(T, y, 'Intercept', false);
PLS_mdl.beta = mdl.Coefficients.Estimate;

% Simulate the model response
y_PLS = PredictTimeSeries(PLS_mdl, Data, L);

% Plot and compare the results
:
```



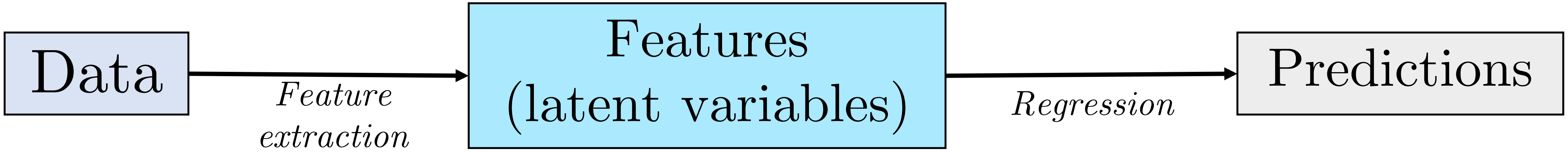
Recap

- Model variance can be reduced by extracting the most important features, and discarding those that do not contribute



Recap

- Model variance can be reduced by extracting the most important features, and discarding those that do not contribute



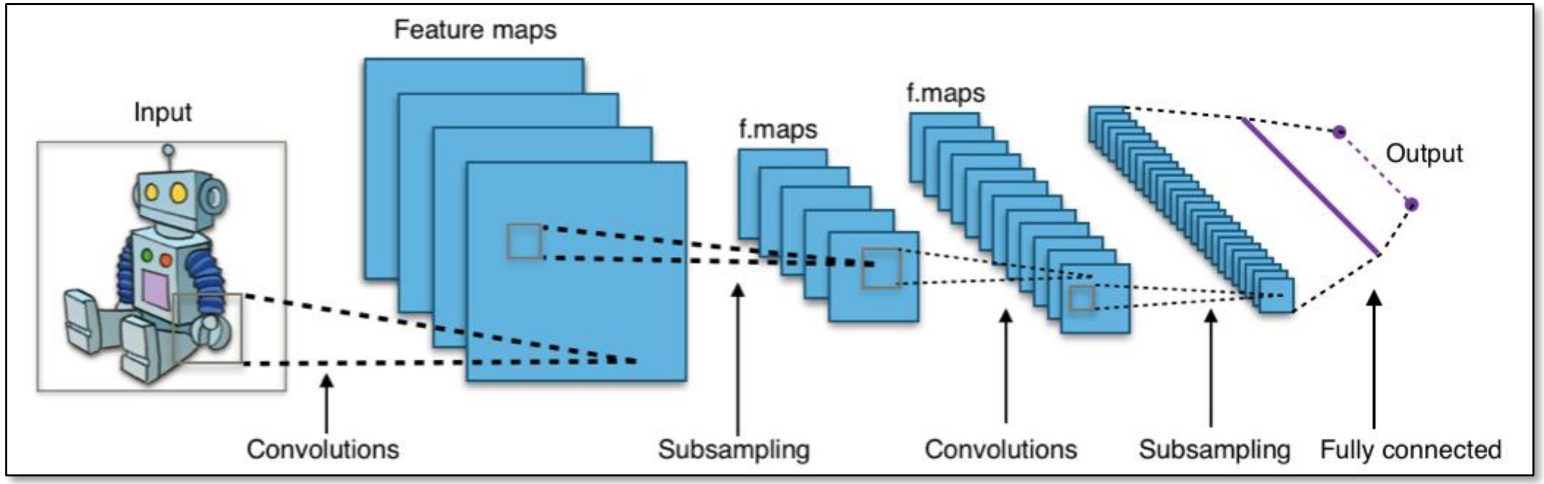
$\mathbf{X}$ $\quad$ $\boldsymbol{\Sigma} = \mathbf{Q}^T \boldsymbol{\Lambda} \mathbf{Q}$ $\quad$ $\mathbf{T} = \mathbf{X}\mathbf{Q}$ $\quad$ $\mathbf{T}^T\mathbf{y} = (\mathbf{T}^T\mathbf{T})\boldsymbol{\beta}$ $\quad$ $\hat{\mathbf{y}} = \mathbf{T}\boldsymbol{\beta}$

Recap

- Model variance can be reduced by extracting the most important features, and discarding those that do not contribute

Data $\xrightarrow{\textit{Feature extraction}}$ Features (latent variables) $\xrightarrow{\textit{Regression}}$ Predictions

$\mathbf{X}$ $\quad$ $\boldsymbol{\Sigma} = \mathbf{Q}^T \boldsymbol{\Lambda} \mathbf{Q}$ $\quad$ $\mathbf{T} = \mathbf{XQ}$ $\quad$ $\mathbf{T}^T\mathbf{y} = (\mathbf{T}^T\mathbf{T})\boldsymbol{\beta}$ $\quad$ $\hat{\mathbf{y}} = \mathbf{T}\boldsymbol{\beta}$



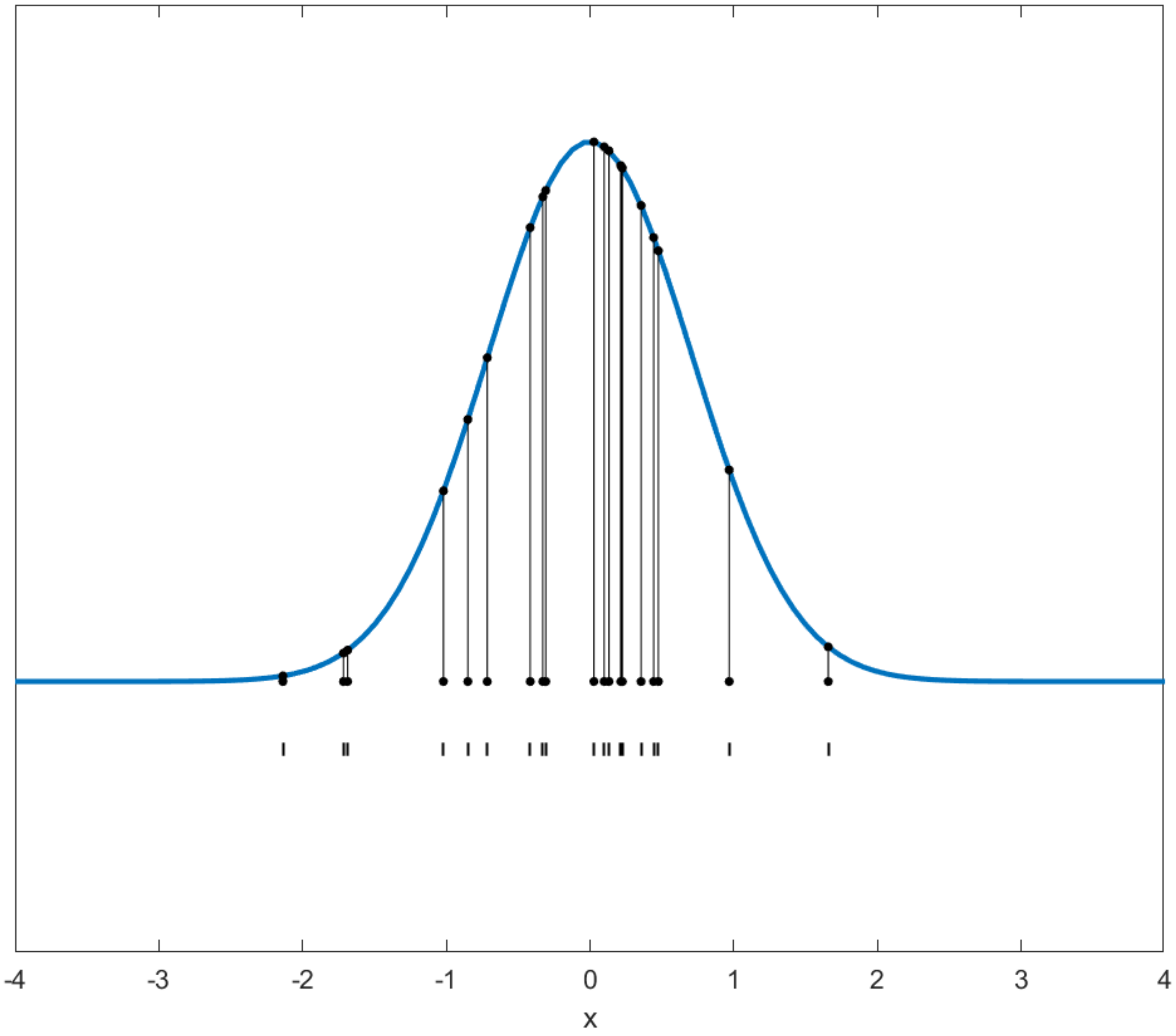
Recap

- Feature extraction provides an alternative approach to decrease model variance and improve prediction accuracy
- PCA and PLS are examples of linear feature extraction
- Common in system identification: Canonical Correlation Analysis (CCA)
- Non-linear feature extraction / latent variable modelling is at the heart of many modern machine learning methods (deep learning, GP-LVM, etc)

Why are the eigenvectors of the covariance matrix also the principal components?

- Consider data distributed normally with zero mean and unit variance
- The probability density of an observation x_i is given by:

$$p(x_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-x_i^2\right)$$



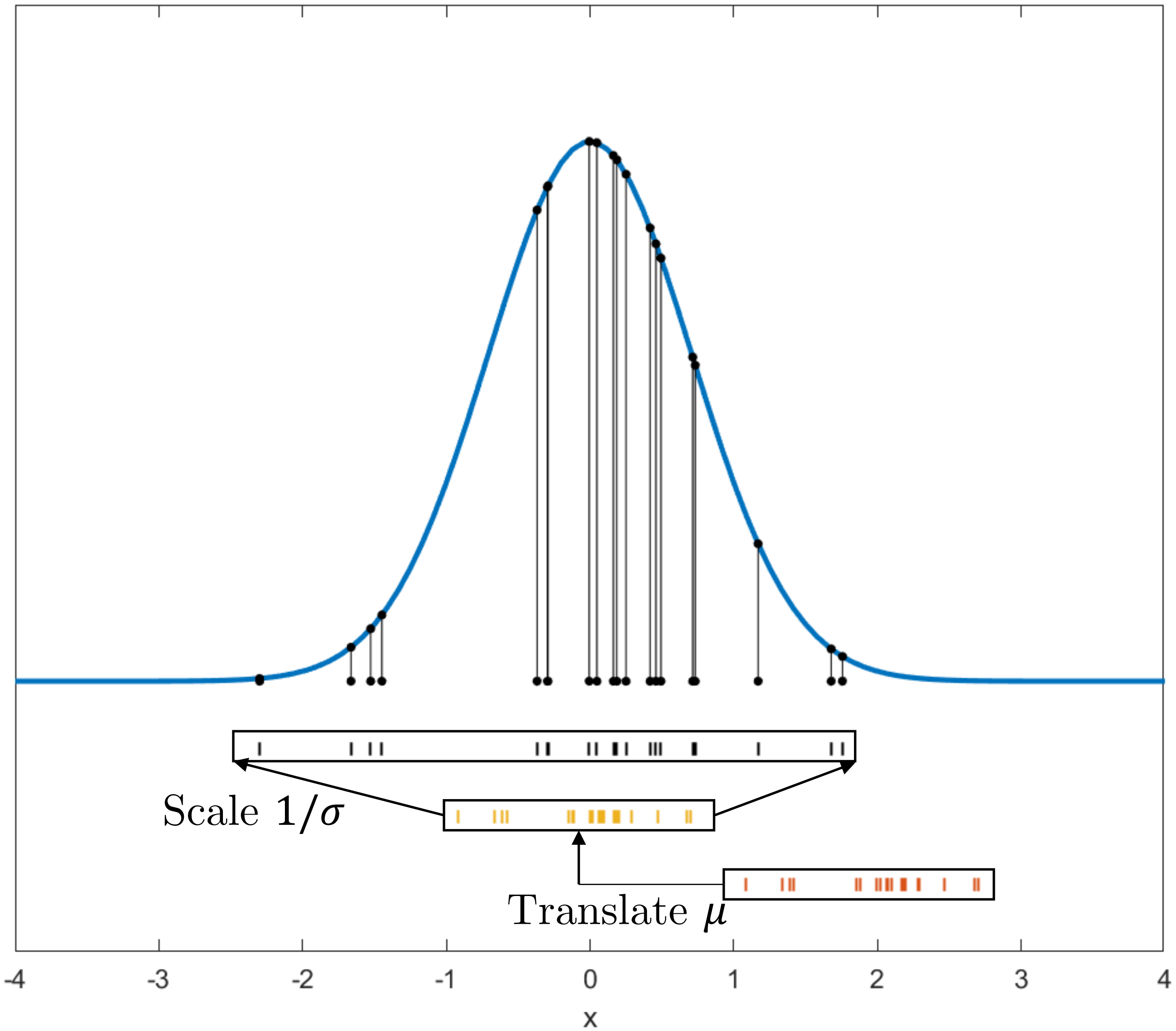
Why are the eigenvectors of the covariance matrix also the principal components?

- Consider data distributed with mean μ and variance σ^2
- The probability density is found by translating each x_i by μ, then scaling by σ

$$\tilde{x}_i = \frac{x_i - \mu}{\sigma}$$

- The probability density of an observation x_i is given by:

$$p(x_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\tilde{x}_i^2\right)$$



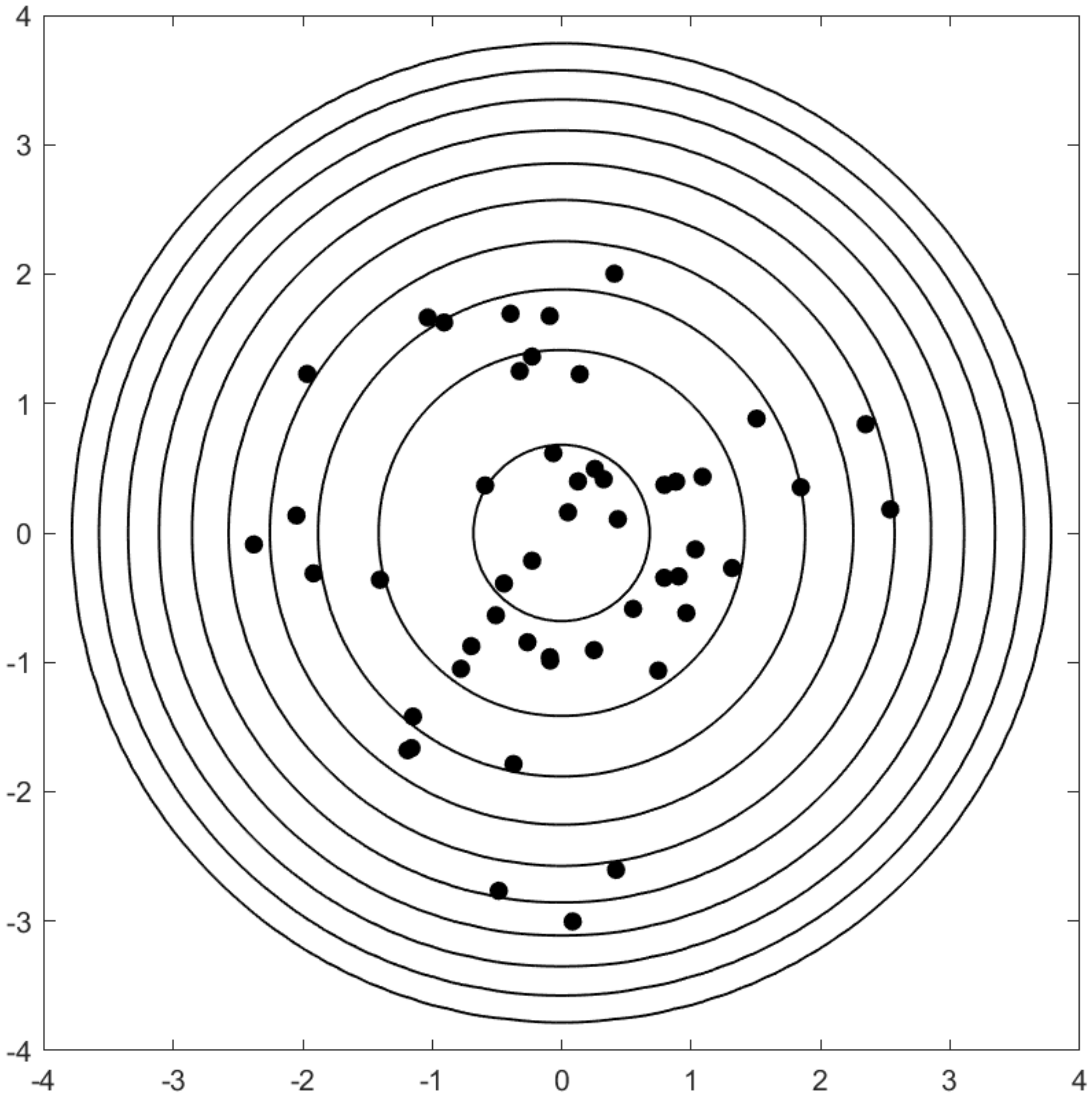
Why are the eigenvectors of the covariance matrix also the principal components?

- Consider 2-dimensional data distributed normally with zero mean and unit variance
- The probability density of an observation $(x_{i,1}, x_{i,2})$ is given by:

$$p(\mathbf{x}_i) = \frac{1}{2\pi}\exp\left(-\left(x_{i,1}^2 + x_{i,2}^2\right)\right)$$

$$= \frac{1}{2\pi}\exp\left(-\begin{bmatrix} x_{i,1} & x_{i,2}\end{bmatrix}\begin{bmatrix} x_{i,1} \\ x_{i,2}\end{bmatrix}\right)$$

$$= \frac{1}{2\pi}\exp\left(-\mathbf{x}_i\mathbf{x}_i^T\right)$$



Why are the eigenvectors of the covariance matrix also the principal components?

- Consider 2-dimensional data distributed normally with mean $\boldsymbol{\mu}$ and covariance $\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$
- The data is translated and scaled by

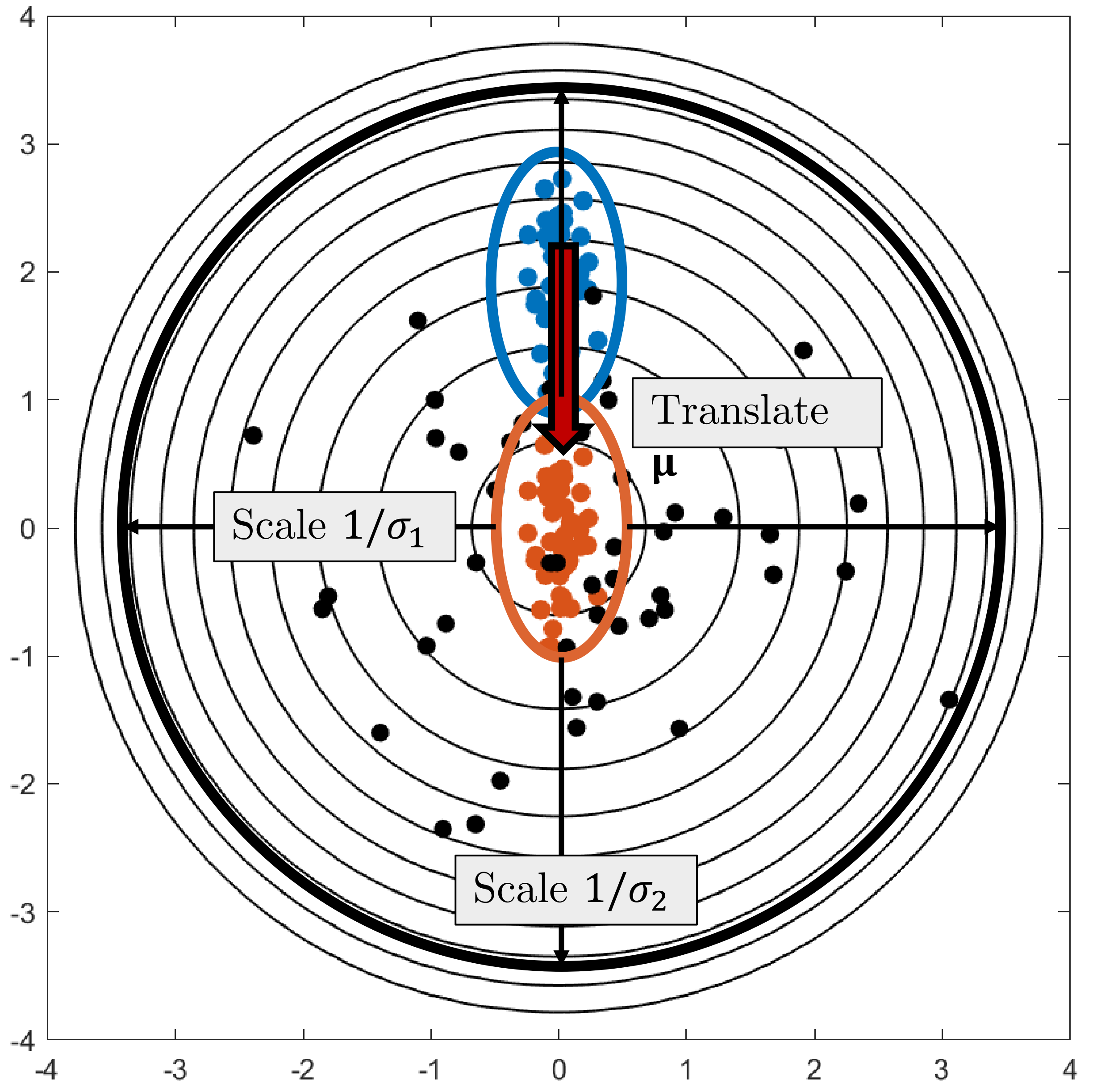
$$[\tilde{x}_{i,1} \quad \tilde{x}_{i,2}] = \left[\frac{x_{i,1} - \mu_1}{\sigma_1} \quad \frac{x_{i,2} - \mu_2}{\sigma_2}\right]$$

$$= (\mathbf{x}_i - \boldsymbol{\mu}) \begin{bmatrix} 1/\sigma_1 & 0 \\ 0 & 1/\sigma_2 \end{bmatrix}$$

$$= (\mathbf{x}_i - \boldsymbol{\mu})\Sigma^{-1/2}$$

- The probability density of an observation $(x_{i,1}, x_{i,2})$ is given by:

$$p(\mathbf{x}_i) = \frac{1}{2\pi} \exp\left(-\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T\right)$$



Why are the eigenvectors of the covariance matrix also the principal components?

- Consider 2-dimensional data distributed normally with mean $\boldsymbol{\mu}$ and covariance $\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12}^2 \\ \sigma_{12}^2 & \sigma_2^2 \end{bmatrix}$

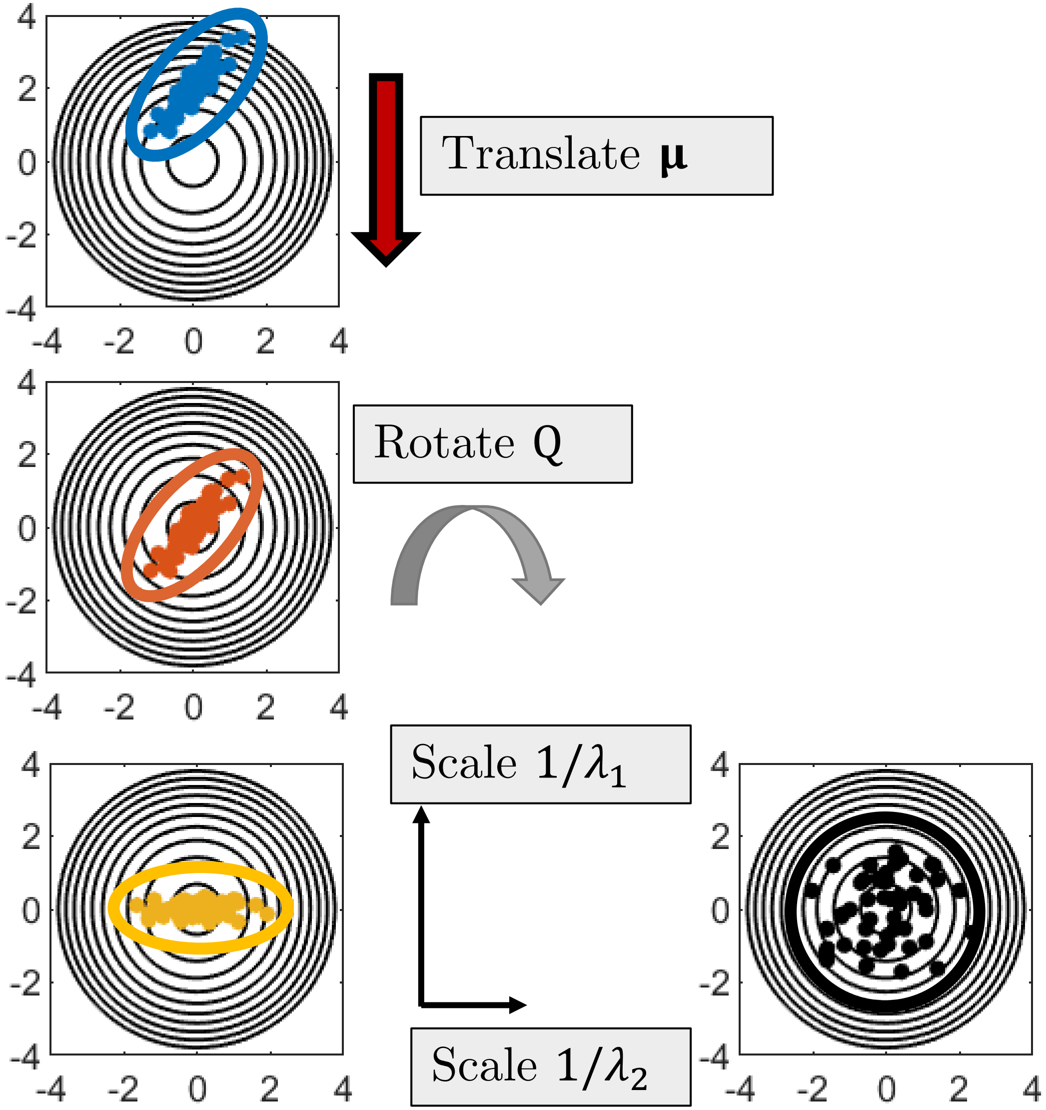
- The data is translated, rotated, and scaled by

$$[\tilde{x}_{i,1} \quad \tilde{x}_{i,2}] = (\mathbf{x}_i - \boldsymbol{\mu}) \begin{bmatrix} q_{11} & q_{21} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} 1/\Lambda_1 & 0 \\ 0 & 1/\Lambda_1 \end{bmatrix}$$

$$= (\mathbf{x}_i - \boldsymbol{\mu}) \mathrm{Q} \Lambda^{-1/2} = (\mathbf{x}_i - \boldsymbol{\mu}) \Sigma^{-1/2}$$

- The probability density of an observation $(x_{i,1}, x_{i,2})$ is given by:

$$p(\mathbf{x}_i) = \frac{1}{2\pi} \exp\left(-\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T\right)$$



Why are the eigenvectors of the covariance matrix also the principal components?

- The eigendecomposition of the covariance matrix yields:

$$\Sigma = \mathrm{Q}\Lambda\mathrm{Q}^T = \left(\mathrm{Q}\Lambda^{1/2}\right)\left(\mathrm{Q}\Lambda^{1/2}\right)^T$$

$$\Sigma^{-1} = \mathrm{Q}\Lambda\mathrm{Q}^T = \left(\mathrm{Q}\Lambda^{-1/2}\right)\left(\mathrm{Q}\Lambda^{-1/2}\right)^T$$

$$p(\mathbf{x}_i) \;\; = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu})\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu})^T\right)$$

$$= \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left[(\mathbf{x}_i - \boldsymbol{\mu})\left(\mathrm{Q}\Lambda^{-1/2}\right)\right]\left[(\mathbf{x}_i - \boldsymbol{\mu})\left(\mathrm{Q}\Lambda^{-1/2}\right)\right]^T\right)$$