ECS231

PCA, *revisited*

May 28, 2019

Outline

1. PCA for lossy data compression
2. PCA for learning a representation of data
3. Extra: learning XOR

1. PCA for lossy data compression[1]

- Data compression:

  given data points $\{x^{(1)}, ..., x^{(m)}\} \in \mathbb{R}^n$, *for each* $x^{(i)} \in \mathbb{R}^n$, *find the code vector* $c^{(i)} \in \mathbb{R}^\ell$, *where* $\ell < n$.

- Encoding function $f: \ x \to c$
- Lossy decoding function $g: \ c \rightsquigarrow x$
- Reconstruction: $x \approx g(c) = g(f(x))$
- PCA is defined by choicing decoding function:

$$g(c) = Dc$$

  where $D \in \mathbb{R}^{n \times \ell}$ defines the decoding and is constrained to have column orthonormal, i.e., $D^T D = I_\ell$.

- Questions:
  1. How to generate optimal code point c^* for each input point x?
  2. How to choose the decoding matrix D?

[1] Section 2.12 of I. Goodfellow, Y. Bengio and A. Courville, **Deep Learning**, `deeplearningbook.org`

1. PCA for lossy data compression, cont'd

Question 1: How to generate optimal code point c^* for each input point x?
i.e., solve

$$c_* = \underset{c}{\operatorname{argmin}} \|x - g(c)\|_2^2.$$

- By vector calculus and the first-order necessary condition for optimality, we conclude

$$c_* = D^T x.$$

- To encode x, we just need the mat-vec product

$$f(x) = D^T x$$

- PCA reconstruction operation

$$r(x) = g(f(x)) = g(D^T x) = DD^T x.$$

1. PCA for lossy data compression, cont'd

Question 2: How to choose the decoding matrix D?

- Idea: minimize the L^2 distance between inputs and reconstructions:

$$\begin{cases} D_* & = & \text{argmin}_D \sqrt{\sum_{i,j} (x_j^{(i)} - r(x^{(i)})_j)^2} \\ \text{s.t.} & & D^T D = I_\ell \end{cases}$$

- **For simplicity**, consider $\ell = 1$ and $D = d \in \mathbb{R}^n$, then

$$\begin{cases} d_* & = & \text{argmin}_d \sum_i \|x^{(i)} - dd^T x^{(i)}\|_2^2 \\ \text{s.t.} & & d^T d = 1. \end{cases}$$

- Let $X \in \mathbb{R}^{m \times n}$ with $X_{(i,:)} = (x^{(i)})^T$, then

$$\begin{cases} d_* & = & \text{argmin}_d \|X - Xdd^T\|_F^2 \\ \text{s.t.} & & d^T d = 1. \end{cases}$$

1. PCA for lossy data compression, cont'd

- Equivalently,

$$\begin{cases} d_* & = & \text{argmax}_d \text{tr}(X^T X d d^T) = \text{argmax}_d \|Xd\|_2^2 \\ \text{s.t.} & & d^T d = 1. \end{cases}$$

- Let $(\sigma_, u_1, v_1)$ be the largest singular triplet of X, i.e.,

$$Xv_1 = \sigma_1 u_1.$$

  Then we have

$$d_* = \text{argmax}_d \|Xd\|_2^2 = v_1.$$

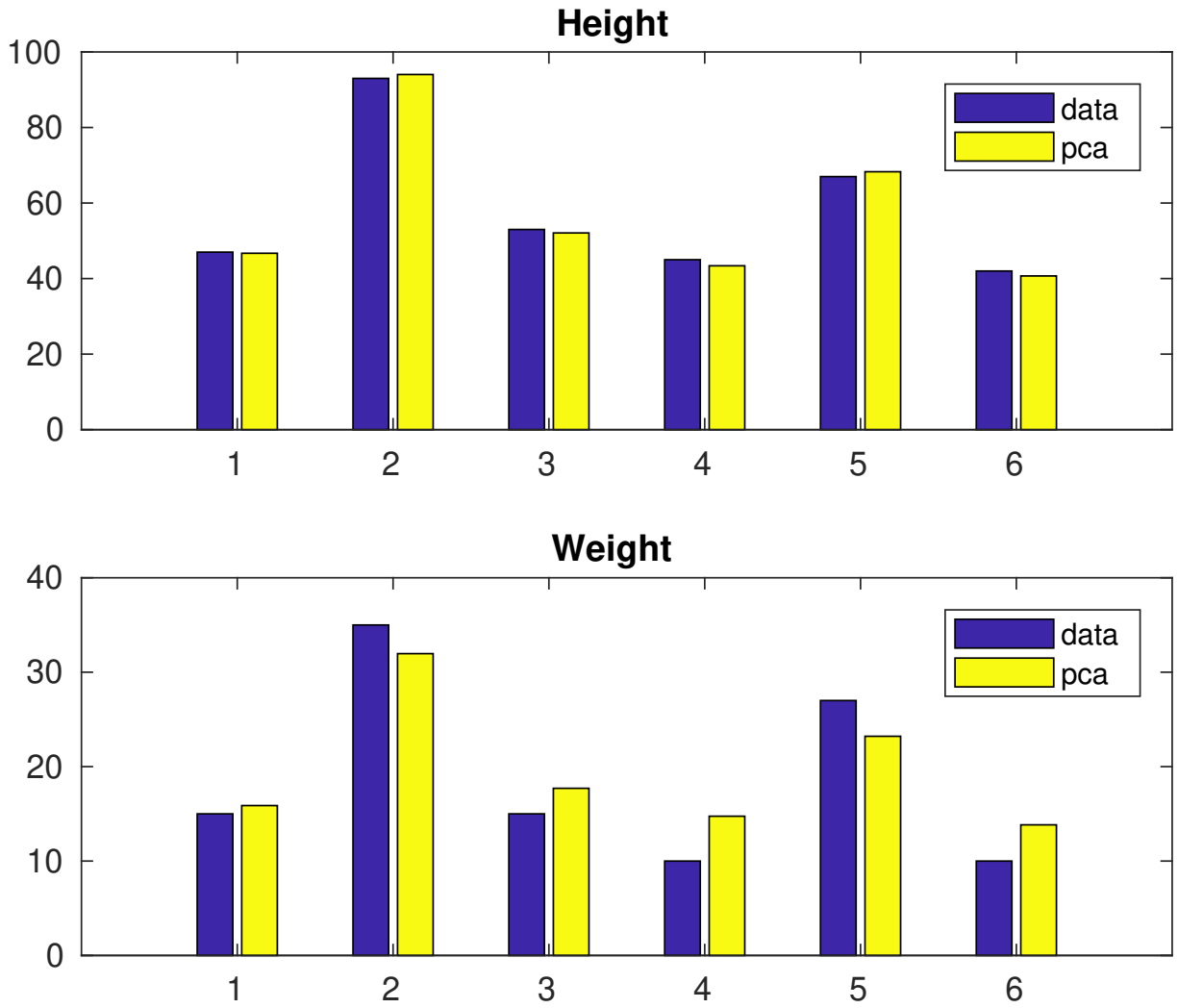
- In the general case, when $\ell > 1$, the matrix D is given by the ℓ right singular vectors of X corresponding to the ℓ largest singular values of X. (Exercise: write out the proof.)

1. PCA for lossy data compression, cont'd

MATLAB demo code: `pca4ldc.m`

```
>> ...
>> % SVD
>> [U,S,V] = svd(X,0);
>> %
>> % Decode matrix D = V(:,1)
>> %
>> % PCA reconstruction
>> %   Xpca = (X*V(:,1))*V(:,1)' = sigma(1)*U(:,1)*V(:,1)';
>> %
>> Xpca = (X*V(:,1))*V(:,1)'
>> ...
```

1. PCA for lossy data compression, cont'd



22. PCA for learning a repesentation of data[2]

- PCA as an unsupervised learning algorithm that learns a representation of data:
  - learns a representation that has lower dimensionality than the original input.
  - learns a representation whose element have no linear correlation with each other (but may still have nonlinear relationships between variables).
- Consider $m \times n$ "design" matrix X of data x with

$$\mathbb{E}[x] = 0$$

$$\text{Var}[x] = \frac{1}{m-1} X^T X.$$

- PCA finds a representation of x via an orthogonal linear transformation

$$z = x^T W$$

such that

$$\text{Var}[z] = \text{diag},$$

where the transformation matrix W satisfying $W^T W = I$.

[2] Section 5.8.1 of I. Goodfellow, Y. Bengio and A. Courville, **Deep Learning**, `deeplearningbook.org`

2. PCA for learning a repesentation of data, cont'd

Question: how to find W?

- Let $X = U\Sigma W^T$ be the SVD of X
- Then

$$\begin{aligned}
\text{Var}[x] &= \frac{1}{m-1} X^T X \\
&= \frac{1}{m-1} (U\Sigma W^T)^T U\Sigma W^T \\
&= \frac{1}{m-1} W^T \Sigma^T U^T U \Sigma W^T \\
&= \frac{1}{m-1} W^T \Sigma^T \Sigma W^T
\end{aligned}$$

2. PCA for learning a repesentation of data, cont'd

- Therefore, if we take

$$z = x^T W$$

Then

$$\begin{aligned}
\text{Var}[z] &= \frac{1}{m-1} Z^T Z \\
&= \frac{1}{m-1} W^T X^T X W \\
&= \frac{1}{m-1} W^T W \Sigma^T \Sigma W^T W \\
&= \frac{1}{m-1} \Sigma^T \Sigma
\end{aligned}$$

2. PCA for learning a repesentation of data, cont'd

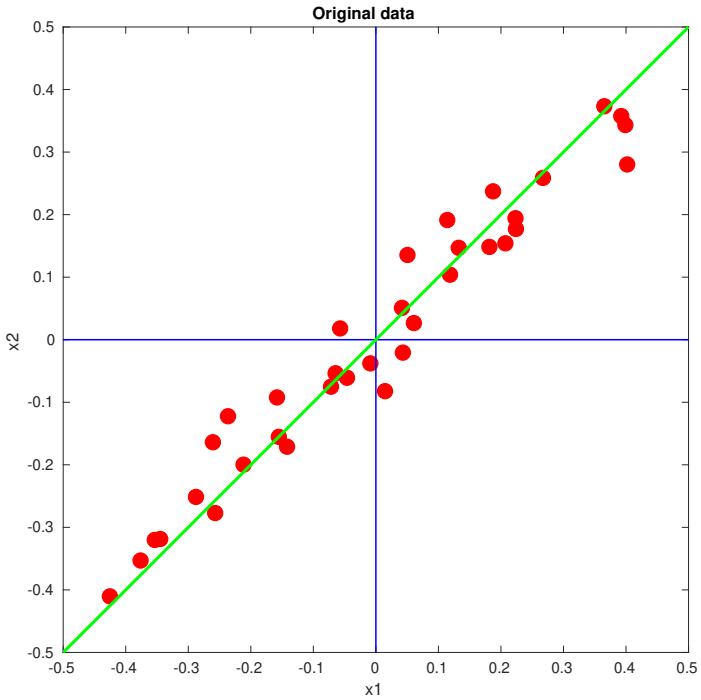
- The individual elements of z are mutually uncorrelated — **disentangle the unknown factors of variation** underlying the data.
- While correlation is an important category of dependency between element of data, we are also interested in learning more representation that disentangle more complicated forms of feature dependencies. For this, we will need to more than what can be done with a simple linear transformation.

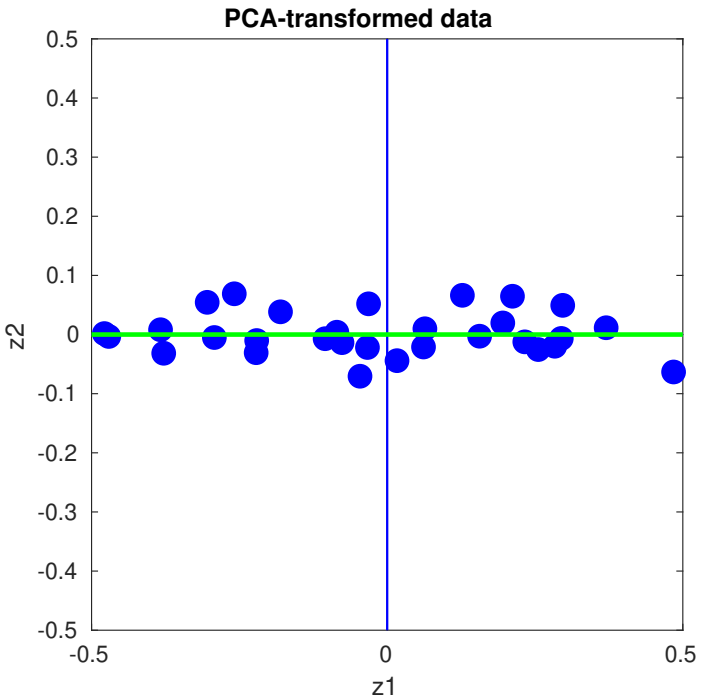
2. PCA for learning a repesentation of data, cont'd

MATLAB demo code: `pca4dr.m`

```
>> ...
>> % make E(x) = 0
>> X1 = X - ones(m,1)*mean(X);
>> %
>> % SVD
>> [U,S,W] = svd(X1);
>> %
>> %PCA
>> Z = X1*W;
>> %
>> % covariance of the new variable z
>> var_z = Z'*Z
>> ...
```

2. PCA for learning a repesentation of data, cont'd





Topic: extra

Learning XOR[3]

- The first (simplest) example of "Deeping Learning"
- The XOR function ("exclusive or")

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

- Task: find function f^* such that
  $y = f^*(x)$ for $x \in \mathbb{X} = \{(0,0),(1,0),(0,1),(1,1)\}$.
- Model: $\hat{y} = f(x;\theta)$, where θ are parameters
- Measure: MSE loss function

$$J(\theta) = \frac{1}{4}\sum_{x\in\mathbb{X}}(f^*(x) - f(x;\theta))^2.$$

[3]Section 6.1 of I. Goodfellow, Y. Bengio and A. Courville, **Deep Learning**, `deeplearningbook.org`

Learning XOR, cont'd

- Linear model:

$$f(x;\theta) = f(x;w,b) = x^T w + b$$

- Solution of the minimization of the MSE loss function

$$w = 0 \quad \text{and} \quad b = \frac{1}{2}.$$

- *A linear model is not able to represent the XOR function*

Learning XOR, cont'd

- Two-layer model:

$$f(x;\theta) = f^{(2)}\left(f^{(1)}(x;W,c);w,b\right)$$

where $\theta \equiv \{W, c, w, b\}$ and

$$f^{(1)}(x;W,c) = \max\{0, W^T x + c\} \equiv h$$
$$f^{(2)}(h;w,b) = w^T h + b,$$

$\max\{0, z\}$ is called an "activation function".

- Then by taking

$$\theta_* = \left\{W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, b = 0\right\}$$

we can verify that the two-layar model ("neural network") obtains the correct answer for any $x \in \mathbb{X}$.

- Question: how to find θ_*?