3. Principal Component Analysis

See: Machine Learning: An Algorithmic Perspective by Stephen Marsland

Data Compression:

- Reduce data from 2D to 1D

$$
\begin{aligned}
x^{(1)} &\in R^2 \rightarrow z^{(1)} \\
x^{(2)} &\in R^2 \rightarrow z^{(2)} \\
&\vdots \\
x^{(m)} &\in R^2 \rightarrow z^{(m)}
\end{aligned}
$$

  e.g.: (work skill, work enjoyment) $\rightarrow$ work aptitude

- Reduce data from 3D to 2D

$$x^{(i)} \in R^3 \rightarrow z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

- Reduce data from 1000D to 100D.
- Reduce data from 1000D to 2D for data visualization.

Example: Going from 2d $\rightarrow$ 1d

Fig. from Marsland (2014)

x	y
2.00	-1.43
2.37	-2.80
1.00	-3.17
0.63	-1.80

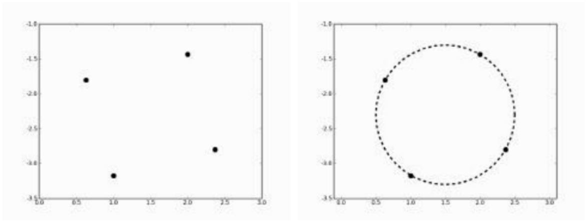


FIGURE 6.1 Three views of the same four points. *Left:* As numbers, where the links are unclear. *Centre:* As four plotted points. *Right:* As four points that lie on a circle.

PCA — an example

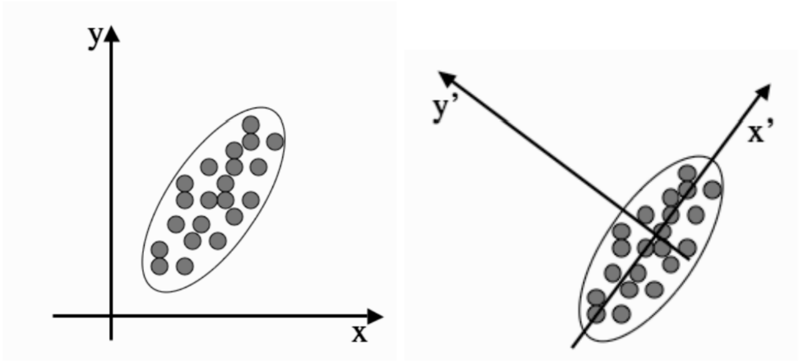


FIGURE 6.6 Two different sets of coordinate axes. The second consists of a rotation and translation of the first and was found using Principal Components Analysis.

→ **Idea: Remove dimensions with little variability**

Principal Component Analysis (PCA) problem formulation

- Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathrm{R}^2$) onto which to project the data so as to minimize the projection error.
- Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \cdots, u^{(k)} \in \mathbb{R}^n$ onto which to project the data so as to minimize the projection error.
- 3D $\rightarrow$ 2d, $K = 2$
- PCA is not linear regression $x \rightarrow y$

Regression:	vertical distance to y	a special variable y
PCA:	a shortest distance to projection surface	all variables $x_1, x_2, \cdots, x_n$ are treated symmetrically

Principal Component Analysis (PCA)

- By finding particular sets of coordinates axes $u^{(1)}, u^{(2)}, \cdots, u^{(k)} \in \mathbb{R}^n$, it will became clear that some of the dimensions are not required.
- In fact, it can make the results better, since we are often removing some of the noise in the data.
- The question is how to choose the directions $u^{(1)}, u^{(2)}, \cdots, u^{(k)} \in \mathbb{R}^n$
- The idea of the principal component is that it is a direction in the data with the largest variation.

Data preprocessing

- Training set: $x^{(1)}, x^{(2)}, \cdots, x^{(k)}$
- Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

- Replace each $x_j^{(i)}$ with $x_j^{(i)} - \mu_j$.
- If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values:

$$x_j^{(i)} \text{ with } x_j^{(i)} - \mu_j.$$

s_j is some measure of deviation like range $=$ max $-$ min or standard deviation.

The PCA Algorithm

- Reduce data from n dimensions to k dimensions
- Write m data points $x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\right)$ as row vectors.
- Matrix X will have size $m \times n$.
- Center the data by subtracting off the mean of each column.

The PCA Algorithm

- Compute **covariance matrix**:

$$C = \frac{1}{m} \sum_{i=1}^{n} \left(x^{(i)}\right)^{T} \left(x^{(i)}\right)$$

$\left(x^{(i)}\right)^{T}$ is a $n \times 1$ vector, $x^{(i)}$ is a $1 \times n$ vector, then C will be a $n \times n$ matrix.

The PCA Algorithm

- Rotate the data $Y = P^T X$.
- P^T is a rotation matrix and P is choose so that the covariance matrix of Y is diagonal:

$$\mathrm{cov}(Y) = \mathrm{cov}\left(P^T X\right) = \begin{pmatrix} \lambda_1 & & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & & 0 & \cdots & \lambda_N \end{pmatrix}$$

- Thus $\lambda P = CP$

The PCA Algorithm

- Compute the eigenvalues and eigenvectors of C, so $V^{-1}CV = D$, where V holds the eigenvectors of C and D is the $n \times n$ diagonal eigenvalue matrix:

$$\text{evecs} = U = \left[\ u^{(1)}; \quad u^{(2)}; \quad \cdots ; \quad u^{(n)}\ \right]$$

- Sort the columns of D into order of decreasing eigenvalues, and apply the same order to the columns of V.
- **Leave k dimensions in the data:**

$$U_{\text{reduce}} = \left[\ u^{(1)}; \quad u^{(2)}; \quad \cdots ; \quad u^{(k)}\ \right]$$

The PCA Algorithm

- The new components $z^(i)$ will be calculated as:

$\left(z^{(i)}\right)^T = (U_{\text{reduce}})^T \left(x^{(i)}\right)^T$ or $\quad z^{(i)} = x^{(i)} U_{\text{reduce}}$

U_{reduce} is a $n \times k$ matrix, $x^{(i)}$ is a $1 \times n$ vector, then $z^{(i)}$ will be a $1 \times k$ vector.

$\rightarrow$ demo/Principal_ComponentAnalysis_01.ipynb

Alternative derivation of PCA: Minimum error

We introduce a complete orthonormal set of D-dimensional basis vectors $\{u_i\}$ where $i = 1, \ldots, D$ that satisfy

$$\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = \delta_{ij}. \quad \alpha_{nj} = \mathbf{x}_n^{\mathrm{T}}\mathbf{u}_j \quad \mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni}\mathbf{u}_i$$

$$\mathbf{x}_n = \sum_{i=1}^{D} \left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i$$

$$\widetilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i$$

Alternative derivation of PCA: Minimum error

Minimize

$$J = \frac{1}{N}\sum_{n=1}^{N} \left\| \mathbf{x}_n - \widetilde{\mathbf{x}}_n \right\|^2 .$$

$$z_{nj} = \mathbf{x}_n^{\mathrm{T}}\mathbf{u}_j \quad \text{where } j = 1, \ldots, M.$$

setting the derivative of J with respect to b_i to zero

$$\longrightarrow \quad b_j = \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_j \quad \text{where } j = M+1, \ldots, D$$

$$\mathbf{x}_n - \widetilde{\mathbf{x}}_n = \sum_{i=M+1}^{D} \left\{ \left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^{\mathrm{T}} \mathbf{u}_i \right\} \mathbf{u}_i$$

Alternative derivation of PCA: Minimum error

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}\left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)^2 = \sum_{i=M+1}^{D}\mathbf{u}_i^{\mathrm{T}}\mathbf{S}\mathbf{u}_i.$$

constrained minimization otherwise we will obtain the vacuous result $\mathbf{u}_i = 0$. minimize $J = \mathbf{u}_2^{\mathrm{T}}\mathbf{S}\mathbf{u}_2$, subject to the normalization constraint $\mathbf{u}_2^{\mathrm{T}}\mathbf{u}_2 = 1$

$$\widetilde{J} = \mathbf{u}_2^{\mathrm{T}}\mathbf{S}\mathbf{u}_2 + \lambda_2\left(1 - \mathbf{u}_2^{\mathrm{T}}\mathbf{u}_2\right)$$

Setting the derivative with respect to $\mathbf{u}_2$ to zero, we obtain $\mathbf{S}\mathbf{u_2} = \lambda_2\mathbf{u}_2$ $\mathbf{u}_2$ is an eigenvector of $\mathbf{S}$ with eigenvalue λ_2.

Alternative derivation of PCA: Minimum error

The general solution to the minimization of J for arbitrary D and arbitrary $M < D$ is obtained by choosing the $\{\mathbf{u}_i\}$ to be eigenvectors of the covariance matrix given by

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

where $i = 1, \ldots, D$, and as usual the eigenvectors $\{\mathbf{u}_i\}$ are chosen to be orthonormal. The corresponding value of the distortion measure is then given by

$$J = \sum_{i=M+1}^{D} \lambda_i$$

Alternative derivation of PCA: Minimum error

- This is simply the sum of the eigenvalues of those eigenvectors that are orthogonal to the principal subspace.
- We therefore obtain the minimum value of J by selecting these eigenvectors to be those having the D - M smallest eigenvalues, and hence the eigenvectors defining the principal subspace are those corresponding to the **M** largest eigenvalues.
- Although we have considered $M < D$, the PCA analysis still holds if $M = D$, in which case there is no dimensionality reduction but simply a rotation of the coordinate axes to align with principal components.

Reconstruction from compressed representation

$\left(z^{(i)}\right)^T = (U_{\text{reduce}})^T \left(x^{(i)}\right)^T$ or $z^{(i)} = x^{(i)} U_{\text{reduce}}$ U_{reduce} is a $n \times k$ matrix, $x^{(i)}$ is a $1 \times n$ vector, then $z^{(i)}$ will be a $1 \times k$ vector.

Choosing k (number of principal components)

- Average squared projection error: $\frac{1}{m}\sum_{i=1}^{n}\left\|x^{(i)} - x_{\text{approx}}^{(i)}\right\|^2$
- Total variation in the data: $\frac{1}{m}\sum_{i=1}^{n}\left\|x^{(i)}\right\|^2$
- Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{n}\left\|x^{(i)} - x_{\text{approx}}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{n}\left\|x^{(i)}\right\|^2} \leq \eta = 0.01 \quad (1\%)$$

- 99% of variance is retained
- If $\eta = 0.05(5\%)$, then 95% of variance is retained.
- If $\eta = 0.1(10\%)$, then 90% of variance is retained.

Choosing k (number of principal components)

1. Try PCA with $k = 1$
2. Compute U_{reduce} , $z^{(1)}, z^{(2)}, \cdots , z^{(m)}, x^{(1)}_{\text{approx}} , \cdots , x^{(m)}_{\text{approx}}$
3. Check if

$$\frac{\frac{1}{m}\sum_{i=1}^{n} \left\| x^{(i)} - x^{(i)}_{\text{approx}} \right\|^2}{\frac{1}{m}\sum_{i=1}^{n} \left\| x^{(i)} \right\|^2} \leq 0.01?$$

4. If not, then try PCA with $k = 2$, with $k = 3$ and so on until, for example, for $k = 17$ check 3 will be satisfied

Choosing k (number of principal components)

- Another algorithm to choose k is to use a matrix

$$D = \begin{pmatrix} \lambda_1 & & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & & 0 & \cdots & \lambda_N \end{pmatrix}$$

- For given k the retained variance is

$$1 - \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

- Pick smallest value of k for which

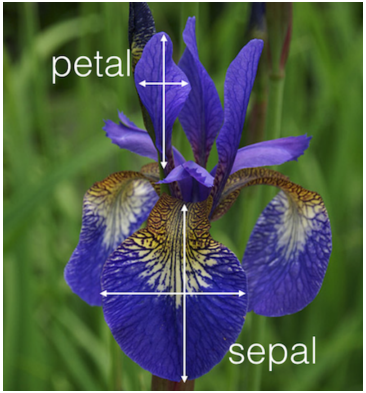
$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 0.99$$

(99% of variance retained)

Example for Principal Component Analysis (PCA): Iris data

`demo/iris/iris_pca.ipynb`

- Downloaded the file from here: `https://archive.ics.uci.edu/ml/datasets/iris`
- `https://en.wikipedia.org/wiki/Iris_flower_data_set`
-
- We have 150 iris flowers.
- For each flower we have 4 measurements
- sepal length, sepal width, petal length, petal width giving 150 points $x^{(1)}, \ldots, x^{(150)} \in \mathbb{R}^4$
- The flowers belong to three different species: 0 : setosa, 1: versicolor, 2: virginica



Kernel PCA (read at home)

- One problem with PCA is that it assumes that the directions of variation are all straight lines.
- This is often not true.
- The Kernel PCA uses the kernel trick to get around this problem.
- We apply a (possible non-linear) function $\Phi(\bullet)$ to each data point x that transform the data into the kernel space, and then perform normal linear PCA in that space.

Kernel PCA (read at home)

- The covariance matrix is defined in the kernel space:

$$C = \frac{1}{N} \sum_{n=1}^{N} \Phi\left(x_n\right) \Phi\left(x_n\right)^T ,$$

- which produces the eigenvector equation:

$$\lambda \left(\Phi\left(x_i\right) P\right) = \left(\Phi\left(x_i\right) CP\right) \quad i = 1 \cdots N,$$

- Where

$$P = \sum_{j=1}^{N} \alpha_j \Phi\left(x_j\right)$$

- are the eigenvectors of the original problem and the α_j will turn out to be the eigenvectors of the kernelized problem. The projection of a new point x into the kernel PCA space:

$$\left(P_k \Phi(x)\right) = \sum_{i=1}^{N} \alpha_i^k \left(\Phi\left(x_i\right) \Phi\left(x_j\right)\right)$$