Machine Learning 2018 – Dimension Reduction

Trung Le

December 18, 2018

How to visualize this dataset ?

Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Setosa
7.0	3.2	4.7	1.4	Versicolor
6.4	3.2	4.5	1.5	Versicolor
6.3	2.9	5.6	1.8	Virginica
5.9	3.0	5.1	1.8	Virginica

Dimensionality Reduction

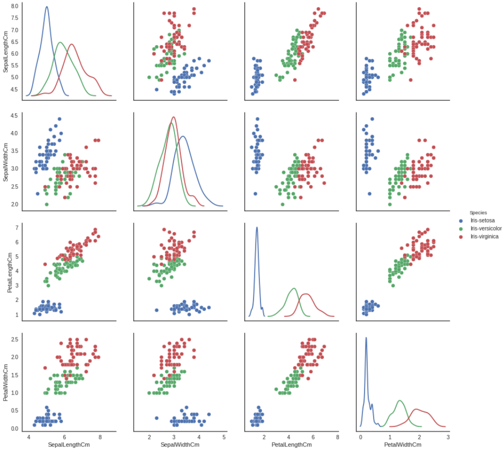


Figure: *Pair plot of iris dataset, source: kaggle.com*

- The number of such plots required for such visualizing data of n variables is $O(n^2)$
- The simplest way to reduce the dimension is by taking a random projection of the data.
- Though random projection allows some degree of visualization of the data structure, it is likely that the more interesting structure within the data will be lost.

Dimensionality Reduction

- Dimensionality Reduction tries to express the data in lower dimension without losing too much information.
- Why dimensionality reduction ?
  - Reduce the dimensions of data to 2D or 3D to visualize it precisely.
  - Help in data compressing and reducing the storage space.
  - Remove redundant features, if any.
- Some of dimension reduction methods: PCA, t-SNE, LDA,...

Dimensionality Reduction

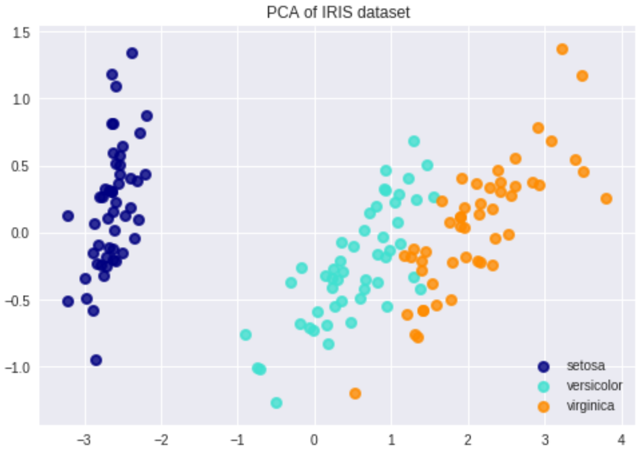


Figure: *PCA of IRIS dataset, source: scikit-learn.org*

Given a dataset $X \in \mathbb{R}^{N \times D}$

- Mean: $\bar{X}_i = \frac{1}{N} \sum_{j=1}^{N} X_{ij}$
- Variance: $Var(X_i) = \frac{1}{N} \sum_{j=1}^{N} (X_{ij} - \bar{X}_i)^2$
- Covariance:
  $Cov(X_i, X_k) = Cov(X_k, X_i) = \frac{1}{N} \sum_{j=1}^{N} (X_{ij} - \bar{X}_i)(X_{kj} - \bar{X}_k)$

Covariance

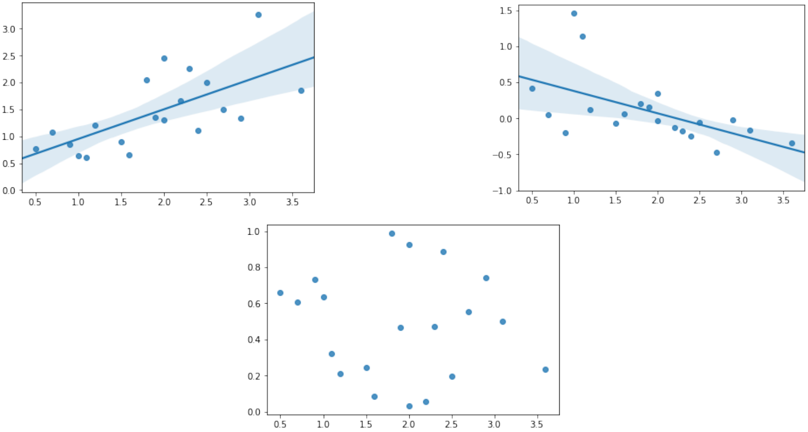


Figure: *Positive, negative and zero covariance*

Covariance Matrix

- Covariance matrix of $X \in \mathbb{R}^{N \times D}$

$$\Sigma = \begin{pmatrix} Var(X_1) & Cov(X_1, X_2) & \dots & Cov(X_1, X_D) \\ Cov(X_2, X_1) & Var(X_2) & \dots & Cov(X_1, X_D) \\ \vdots & \vdots & \vdots & \dots \\ Cov(X_D, X_1) & Cov(X_D, X_2) & \dots & Var(X_D) \end{pmatrix} \tag{1}$$

- For centered data: $\Sigma = \frac{1}{N} X X^T$ where X is re-constructed by subtracting every column by it's mean $X_i = X_i - \bar{X}_i$.

Principal Component Analysis (PCA)

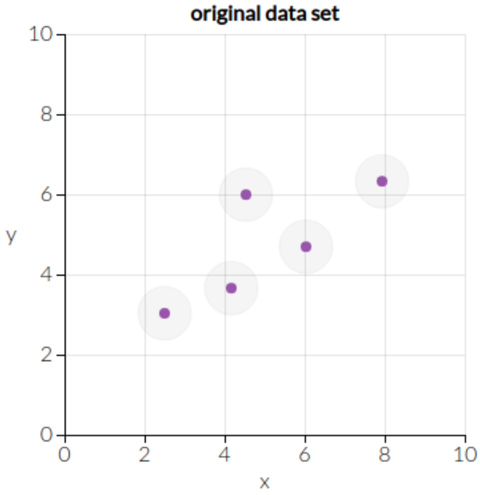


Figure: *Sample data points in 2D*

Principal Component Analysis (PCA)

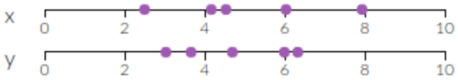


Figure: *Sample data points in 2D*

Principal Component Analysis (PCA)

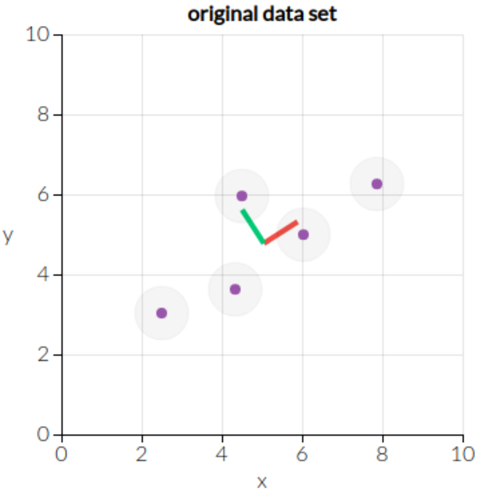


Figure: *PCA of sample data points in 2D*

Principal Component Analysis (PCA)

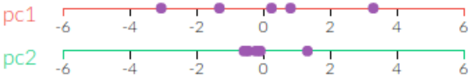


Figure: *PCA of sample data points in 2D*

Principal Component Analysis (PCA)

- Algebraically, principal components are particular linear combinations of the D random variables $X_1, X_2, ..., X_D$.
- Geometrically, these linear combinations represent the selection of a new coordinate system obtained by rotating the original system.

Principal Component Analysis (PCA)

- Let $X \in \mathbb{R}^{N \times D}$ is the original data matrix with N samples and D measurements.
- Consider the linear combinations:

$$
\begin{aligned}
Y_1 = w_1^T X &= w_{11} X_1 + w_{12} X_2 + ... + w_{1D} X_D \\
Y_2 = w_2^T X &= w_{21} X_1 + w_{22} X_2 + ... + w_{2D} X_D \\
\vdots \\
Y_D = w_D^T X &= w_{D1} X_1 + w_{D2} X_2 + ... + w_{DD} X_D
\end{aligned}
\tag{2}
$$

where $w \in \mathbb{R}^{D \times D}$

Principal Component Analysis (PCA)

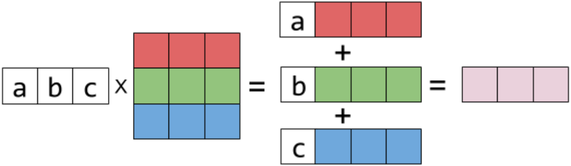


Figure: *Matrix multiplication visualization, source: eli.thegreenplace.net*

Principal Component Analysis (PCA)

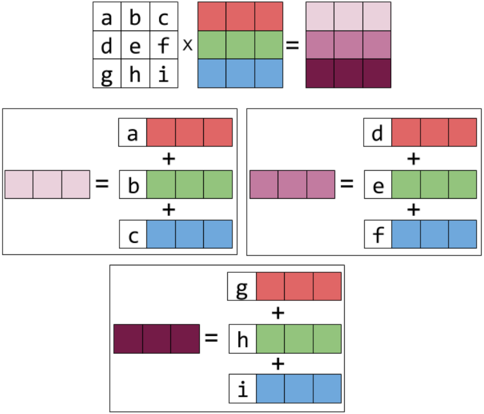


Figure: *Matrix multiplication visuallization, source: eli.thegreenplace.net*

The important point to note is that the variance of any linear combination can be computed using the covariance matrix of the data:

$$\begin{aligned} Var(Y_i) &= \frac{1}{N}\sum_j (X_j w_i)^2 \\ &= \frac{1}{N}(Xw_i)^T(Xw_i) \\ &= \frac{1}{N} w_i^T X^T X w_i \\ &= w_i^T \frac{X^T X}{N} w_i \\ &= w_i^T \Sigma w_i \end{aligned} \tag{3}$$

Principal Component Analysis (PCA)

- Principal components are those linear combinations $Y_1, Y_2, ..., Y_D$ whose variances are as large as possible.
- First principal component: Linear combination $w_1^T X$ that maximize $Var(w_1^T X)$ subject to $||w_1||_2^2 = 1$
- Second principal component: Linear combination $w_2^T X$ that maximize $Var(w_2^T X)$ subject to $||w_2||_2^2 = 1$ and $w_2^T w_1 = 0$
- ith principal component: Linear combination $w_i^T X$ that maximize $Var(w_i^T X)$ subject to $||w_i||_2^2 = 1$ and $w_i^T w_k = 0$ for $k < i$

First Principal Component Analysis (PC1)

For the first principal component, we maximize:

$$Var(Y_1) = w_1^T \Sigma w_1 \tag{4}$$

subject to:

$$w_1^T w_1 = 1 \tag{5}$$

First Principal Component Analysis (PC1)

Using the Lagrange function:

$$\mathcal{L} = w_1^T \Sigma w_1 + \lambda_1 (1 - w_1^T w_1) \tag{6}$$

Taking the partial derivative of $\mathcal{L}$ with respect to w_1, λ_1:

$$\frac{\partial}{\partial w_1} \mathcal{L}(w_1, \lambda_1) = 2\Sigma w_1 - 2\lambda_1 w_1 = 0 \tag{7}$$

$$\frac{\partial}{\partial \lambda_1} \mathcal{L}(w_1, \lambda_1) = 1 - w_1^T w_1 = 0 \tag{8}$$

First Principal Component (PC1)

From 7, we've got:

$$\Sigma w_1 = \lambda w_1 \tag{9}$$

This implies w_1 is an eigenvector of Σ and λ_1 is the corresponded eigenvalue.

Multiply each side of 9 to w_1^T, we've got:

$$w_1^T \Sigma w_1 = Var(w_1^T X) = \lambda_1 w_1^T w_1 = \lambda_1 \tag{10}$$

So $Var(w_1^T X)$ is maximized when λ_1 is the largest eigenvalue of Σ.

Second Principal Component (PC2)

For the second principal component, we maximize:

$$Var(Y_2) = w_2^T \Sigma w_2 \tag{11}$$

subject to:

$$w_2^T w_2 = 1 \tag{12}$$

$$w_1^T w_2 = 0 \tag{13}$$

Second Principal Component (PC2)

Lagrangian of the problem 11:

$$\mathcal{L} = w_2^T \Sigma w_2 + \lambda_2 (1 - w_1^T w_1) + \beta w_1^T w_2 \tag{14}$$

Taking the partial derivative of $\mathcal{L}$ with respect to w_1, λ_1, β:

$$\frac{\partial}{\partial w_2} \mathcal{L}(w_2, \lambda_2, \beta) = 2\Sigma w_2 - 2\lambda_2 w_2 + \beta w_1 = 0 \tag{15}$$

$$\frac{\partial}{\partial \lambda_2} \mathcal{L}(w_2, \lambda_2, \beta) = 1 - w_2^T w_2 = 0 \tag{16}$$

$$\frac{\partial}{\partial \beta} \mathcal{L}(w_2, \lambda_2, \beta) = w_1^T w_2 = 0 \tag{17}$$

Second Principal Component

Multiply each side of 15 with w_1^T:

$$\begin{aligned} 2w_1^T \Sigma w_2 + \beta &= 0 \\ \leftrightarrow 2w_1^T \Sigma w_2 + \beta &= 0 \\ \leftrightarrow 2(\Sigma w_1)^T w_2 + \beta &= 0 \\ \leftrightarrow 2\lambda_1 w_1^T w_2 + \beta &= 0 \\ \rightarrow \beta &= 0 \end{aligned} \tag{18}$$

Second Principal Component (PC2)

- Equation 15 now becomes:

$$\Sigma w_2 = \lambda_2 w_2 \tag{19}$$

- This implies w_2 is an eigenvector of Σ and λ_2 is the corresponded eigenvalue.
- Multiply each side of 9 to w_2^T, we've got:

$$w_2^T \Sigma w_2 = Var(w_2^T X) = \lambda_2 w_2^T w_2 = \lambda_2 \tag{20}$$

- So $Var(w_2^T X)$ is maximized when λ_2 is the second largest eigenvalue of Σ.
- The ith principal component turns out to be obtained by the ith largest eigenvector of Σ.

PCA step by step

1. Compute mean of each column:

$$\bar{X}_i = \frac{1}{N}\sum_{j=1}^{N} X_{ij} \tag{21}$$

2. Subtract mean:

$$X_i = X_i - \bar{X}_i \tag{22}$$

3. Compute covariance matrix:

$$\Sigma = \frac{1}{N}XX^T \tag{23}$$

4. Compute eigenvectors and eigenvalues of Σ: $(\lambda_1, w_1), .., (\lambda_D, w_D)$, $\lambda_1 > \lambda_2 > ... > \lambda_D$
5. Pick K eigenvectors with highest eigenvalues as a matrix: U_K
6. Project original data to selected eigenvectors:

$$\tilde{X} = U_K^T X \tag{24}$$

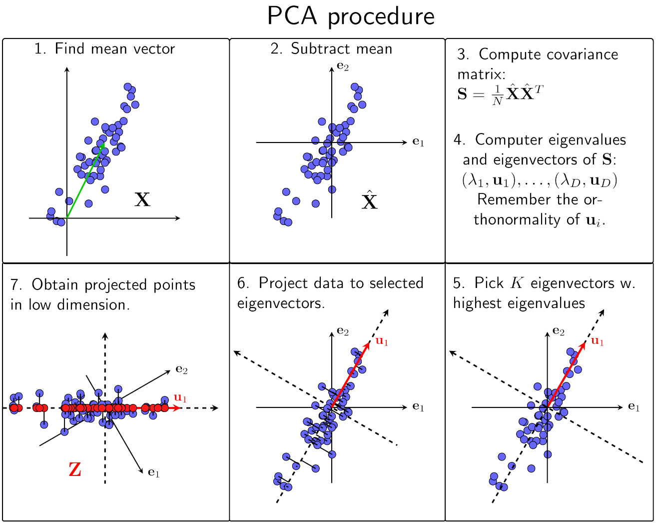


Figure: *PCA procedure, source: machinelearningcoban.com*

Applications

Figure: *Eigenfaces in face recognition.*

t-Distributed Stochastic Neighbor Embedding (t-SNE)

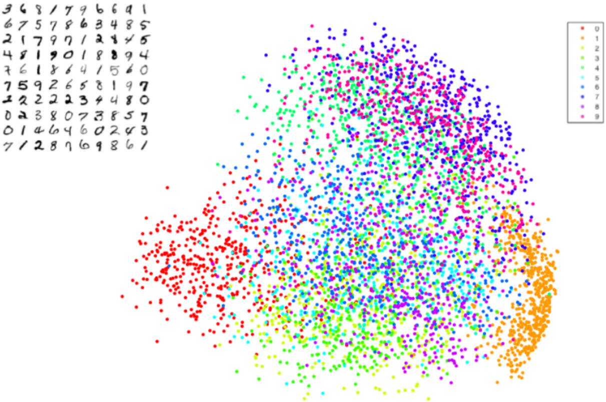


Figure: *PCA on MNIST dataset*

t-Distributed Stochastic Neighbor Embedding (t-SNE)

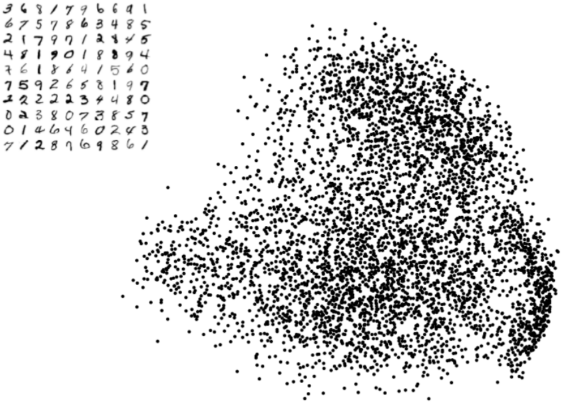
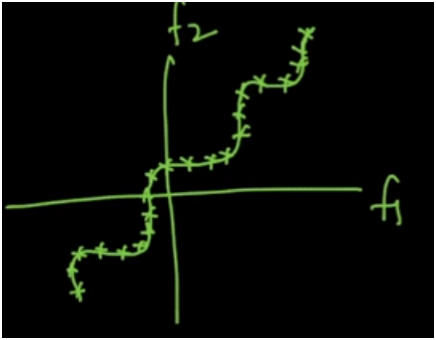


Figure: *PCA on MNIST dataset*

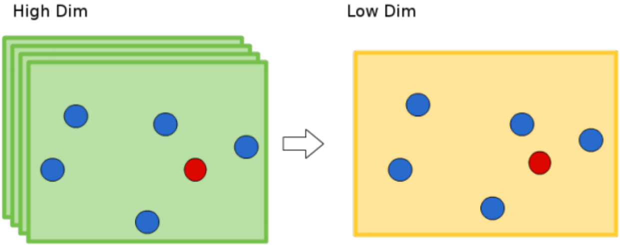
t-Distributed Stochastic Neighbor Embedding (t-SNE)

- How about non-linear data?

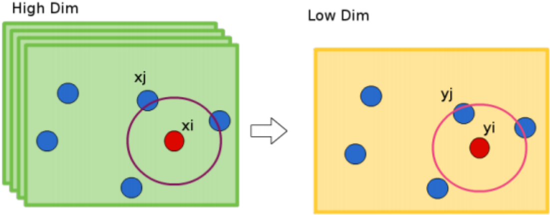


- t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.
- The t-SNE algorithm models the probability distribution of neighbors around each point, so it preserve the local structure (neigborhood) of data.

- Preserve the neighborhood



- Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities



$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{j' \neq i} \exp(-||x_i - x_j'||^2/2\sigma^2)} \tag{25}$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities.

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{j' \neq i} \exp(-||x_i - x_j'||^2/2\sigma_i^2)} \tag{26}$$

- Since each point has different density, we'd use the symmetrized conditional:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{27}$$

- We set the bandwidth σ_i such that the conditional has fixed *perplexity*.

- Similarity in low dimension is measured as:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^-1}{\sum_k \sum_{l \neq k} (1 + ||y_k - y_l||^2)^-1} \tag{28}$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Cost function: Kullback Leibler (KL) divergence:

$$KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{29}$$

  - Large p_{ij} modeled by small q_{ij}: $\rightarrow$ Big penalty.
  - Small p_{ij} modeled by large q_{ij}: $\rightarrow$ Small penalty.
  - t-SNE mainly preserves local similarity structure of data.
- Gradient:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-}1(y_i - y_j) \tag{30}$$

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $X = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=*1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
  end
end

Figure: *t-SNE learning algorithm*

t-Distributed Stochastic Neighbor Embedding (t-SNE)

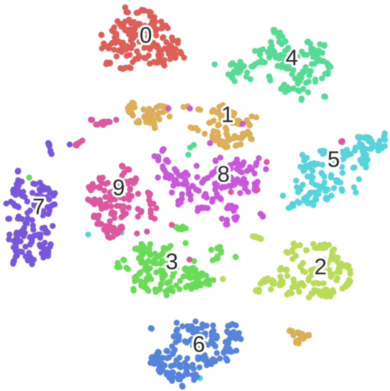


Figure: *t-SNE on MNIST dataset*

References

[1] Richard Johnson et al, Applied Multivariate Statistical Analysis 6th Edition.
[2] Tiep H. Vu, Principal Component Analysis, https://machinelearningcoban.com/2017/06/15/pca/
[3] Laurens van der Maaten & Geoffrey Hinton, Visualizing Data using t-SNE.
[4] Manifold learning, https://scikit-learn.org/stable/modules/manifold.html.