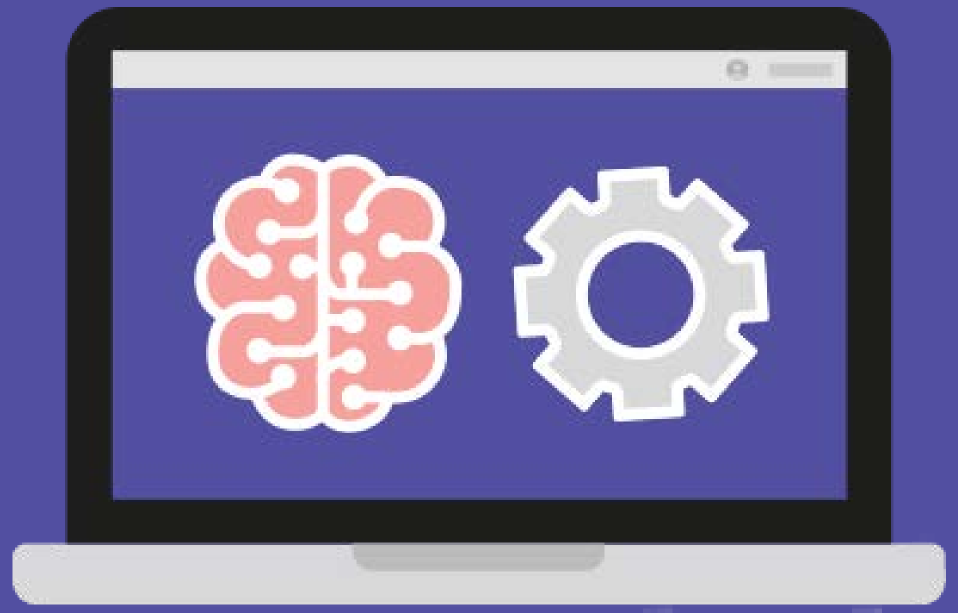
/* elice */

양재 AI School 인공지능 캠프

Lecture 9

차원 축소(Dimensionalitiy Reduction), PCA, tSNE



김도경 선생님

수업 목표

1 차원의 저주 및 차원 축소

1. 차원의 저주(Curse of Dimensionality)
2. 차원 축소(Dimensionality Reduction)

2 주성분 분석(PCA)

1. 선형대수 정리
2. 주성분 분석(PCA)
3. EigenFace

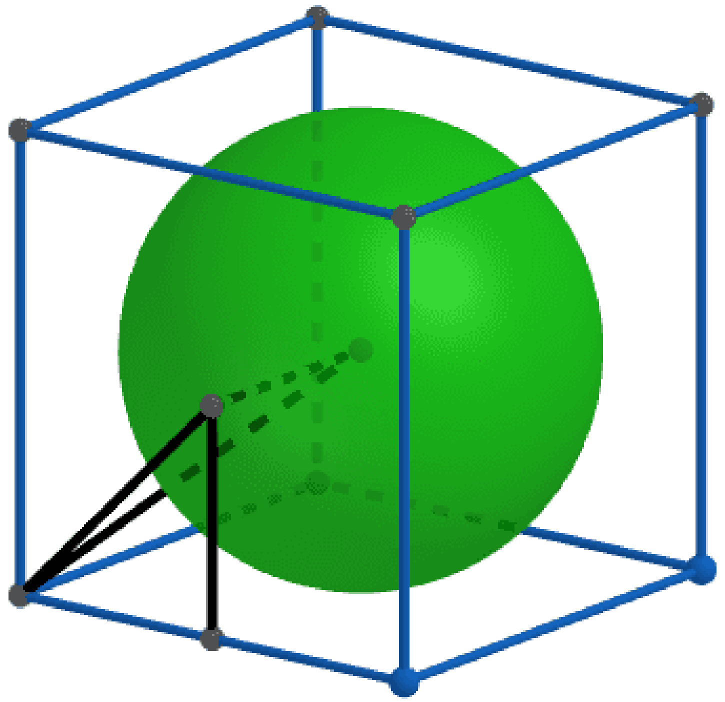
3 tSNE

1. 차원의 저주 및 차원 축소

1-1. 차원의 저주

※ 저차원 vs. 고차원: 저차원에서의 직관이 성립하지 않음

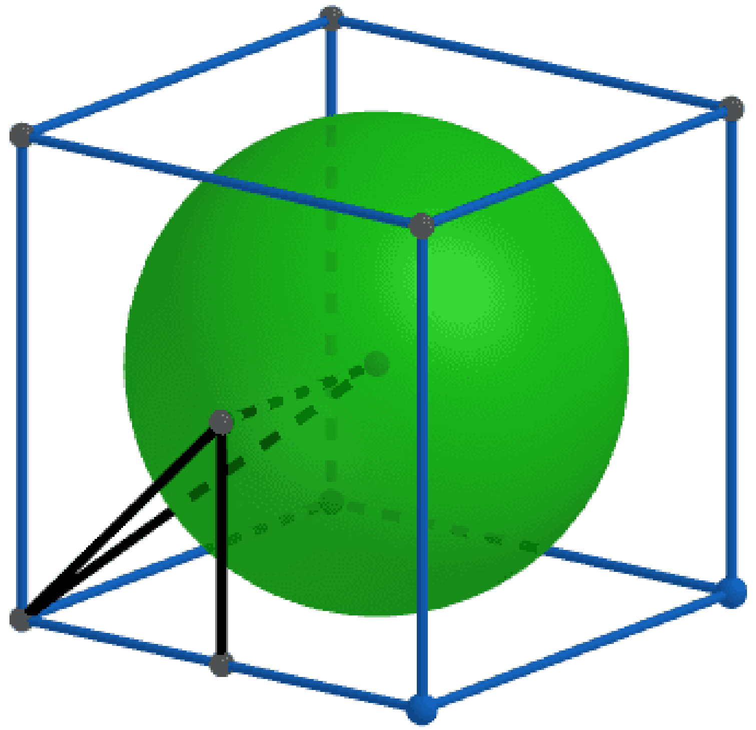
- 2차원의 단위면적을 가진 정사각형 안에 있는 점을 무작위로 선택할 때 가장자리에 있는 점을 선택할 가능성은 매우 낮음
- 10,000차원의 단위면적을 가진 초입방체(hyper cube)에서는 이 가능성이 99.99% 이상



- 2차원의 단위 정사각형에서 임의의 두 점을 선택하면 두 점 사이의 평균거리는 0.52
- 1,000,000차원의 단위 초입방체에서 임의의 두 점을 선택하면 두 점 사이의 평균거리는 428.25

1-1. 차원의 저주

- 한변의 길이가 2r인 초입방체의 부피는 $V_{n-cube} = (2r)^n$
- 한변의 길이가 r인 초구(hyper sphere)의 부피는



$$V_{n-sphere} = \frac{2r^n}{n\,\Gamma(\frac{n}{2})}\pi^{\frac{n}{2}}$$

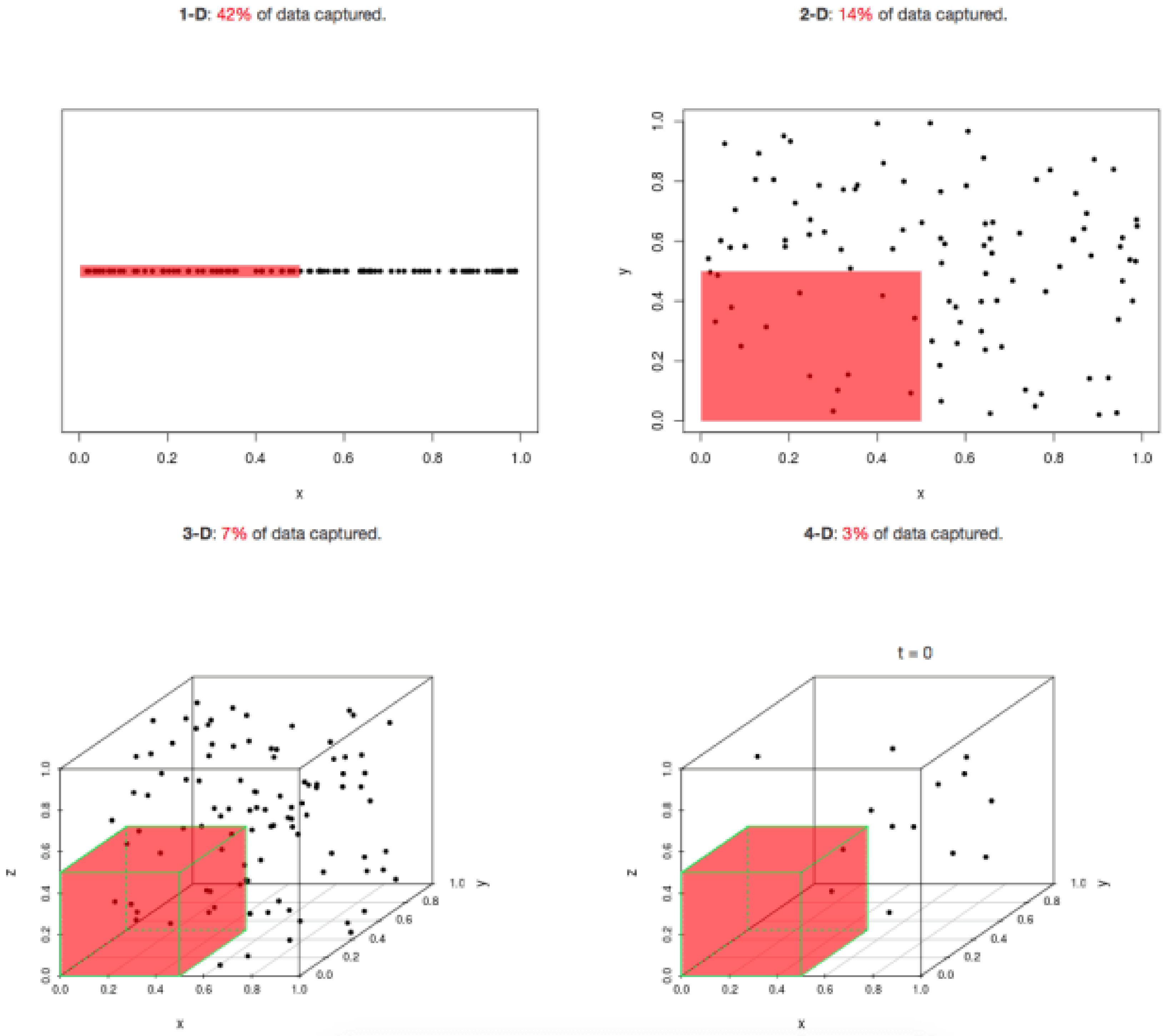
- 차원이 커지면 초입방체에 내접한 초구의 부피의 비율은 0으로 수렴

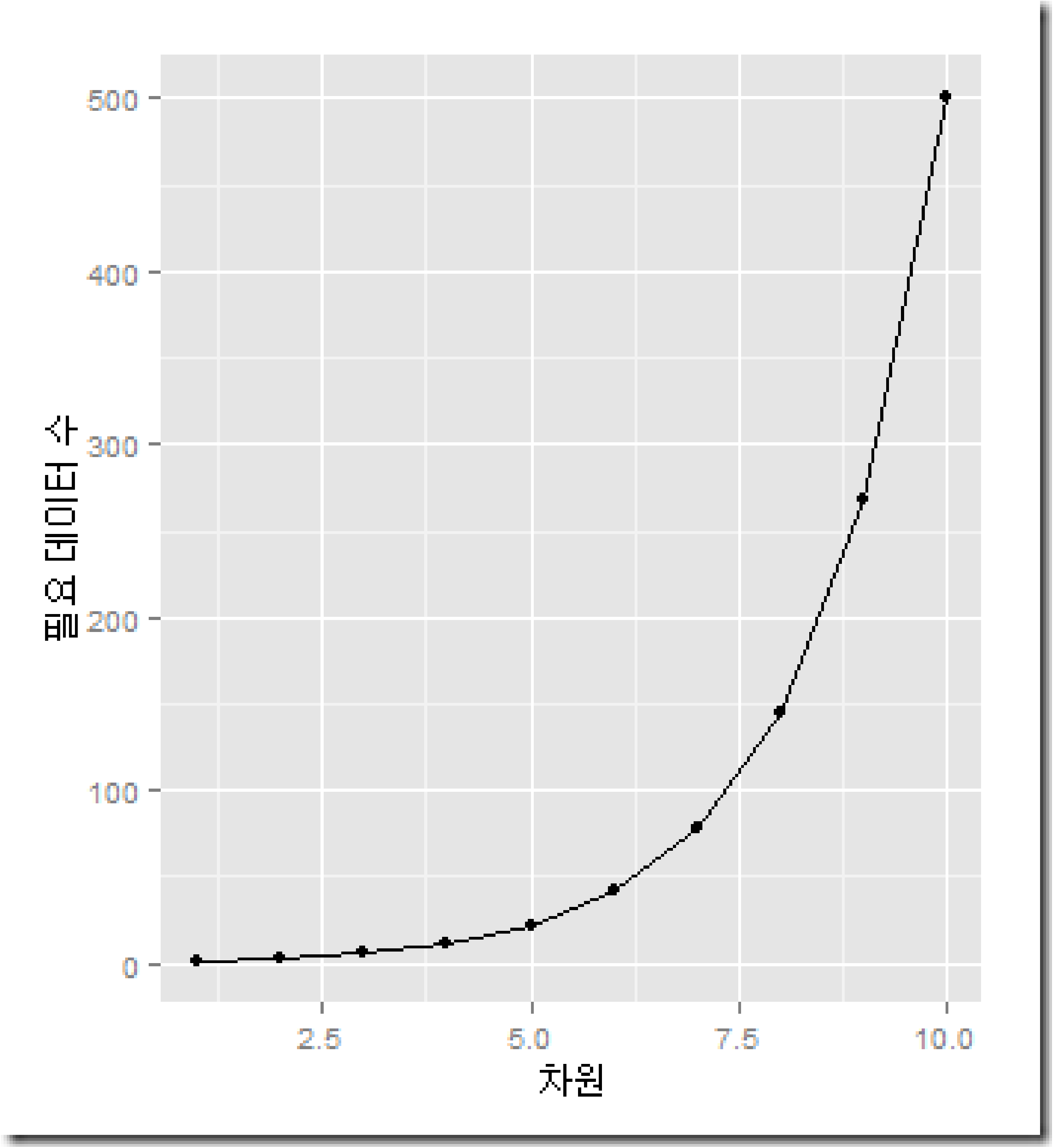
$$\frac{V_{n-sphere}}{V_{n-cube}} = \frac{2}{n\,\Gamma\left(\frac{n}{2}\right)} \times (\frac{\sqrt{\pi}}{2})^n \rightarrow 0$$

1-1. 차원의 저주

- 고차원일수록 전체에서 데이터가 차지하는 공간이 매우 적어짐

  => 필요한 데이터 양이 기하급수적으로 증가



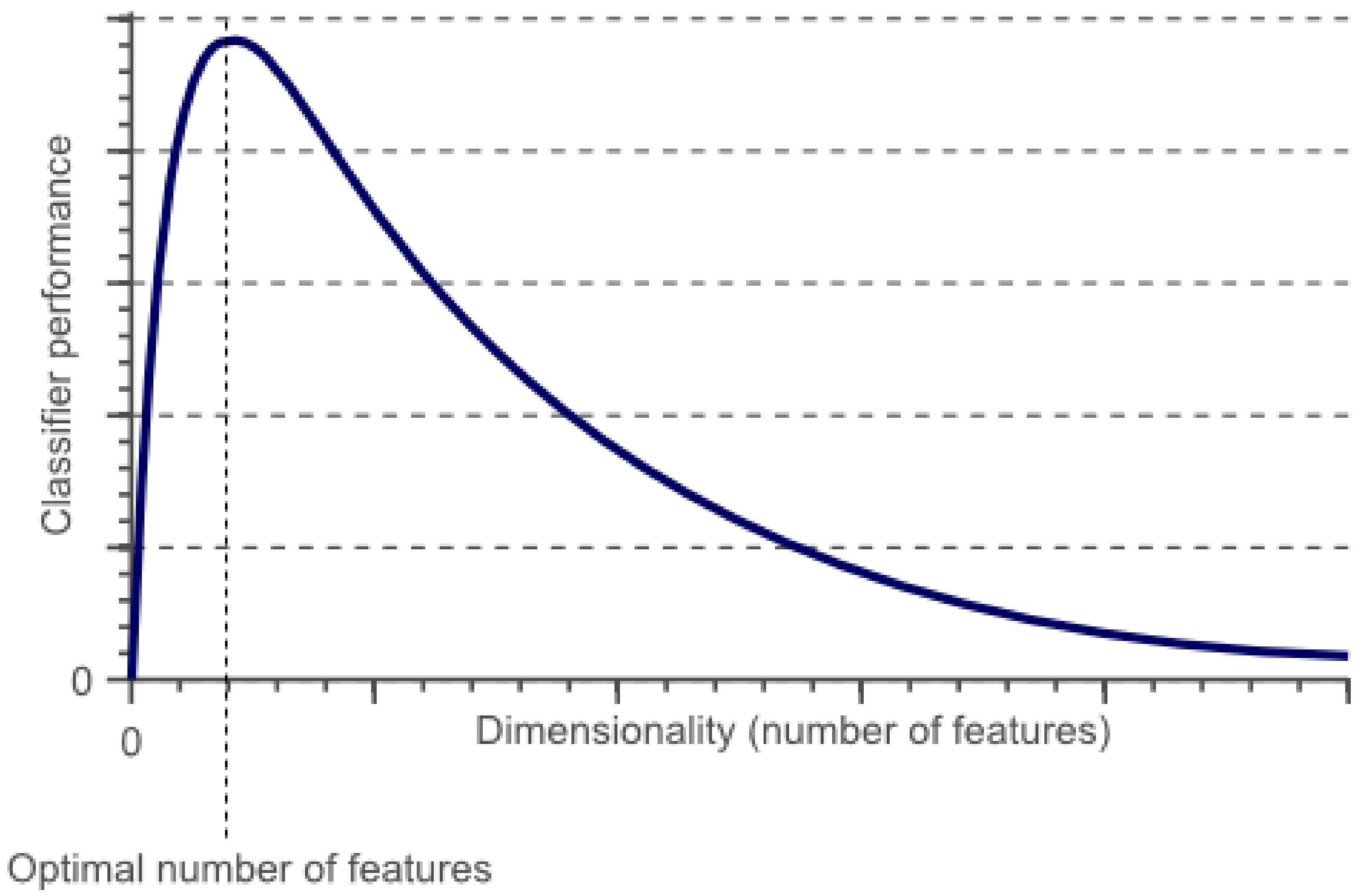


1-1. 차원의 저주

- **차원의 저주(Curse of Dimensionality)**

훈련샘플 각각이 엄청나게 많은(Ex. 수백만 개) 특성을 가지고 있을 때 훈련이 느려질 뿐만 아니라, 최적의 솔루션을 찾기 어려워지는 현상

예) 일정 차원을 넘으면 분류기의 성능은 점점 떨어져 0으로 수렴



1-2. 차원 축소(Dimensionality Reduction)

- 미스코리아들의 얼굴에는 비슷한 점들이 많음

  => 굳이 모든 픽셀을 다 보지 않고도 중요한 특징을 잡아낼 수 있음

1-2. 차원 축소(Dimensionality Reduction)

- 관찰 대상들을 잘 설명할 수 있는 잠재 공간(latent space)은 실제 관찰 공간(observation space)보다 작을 수 있음

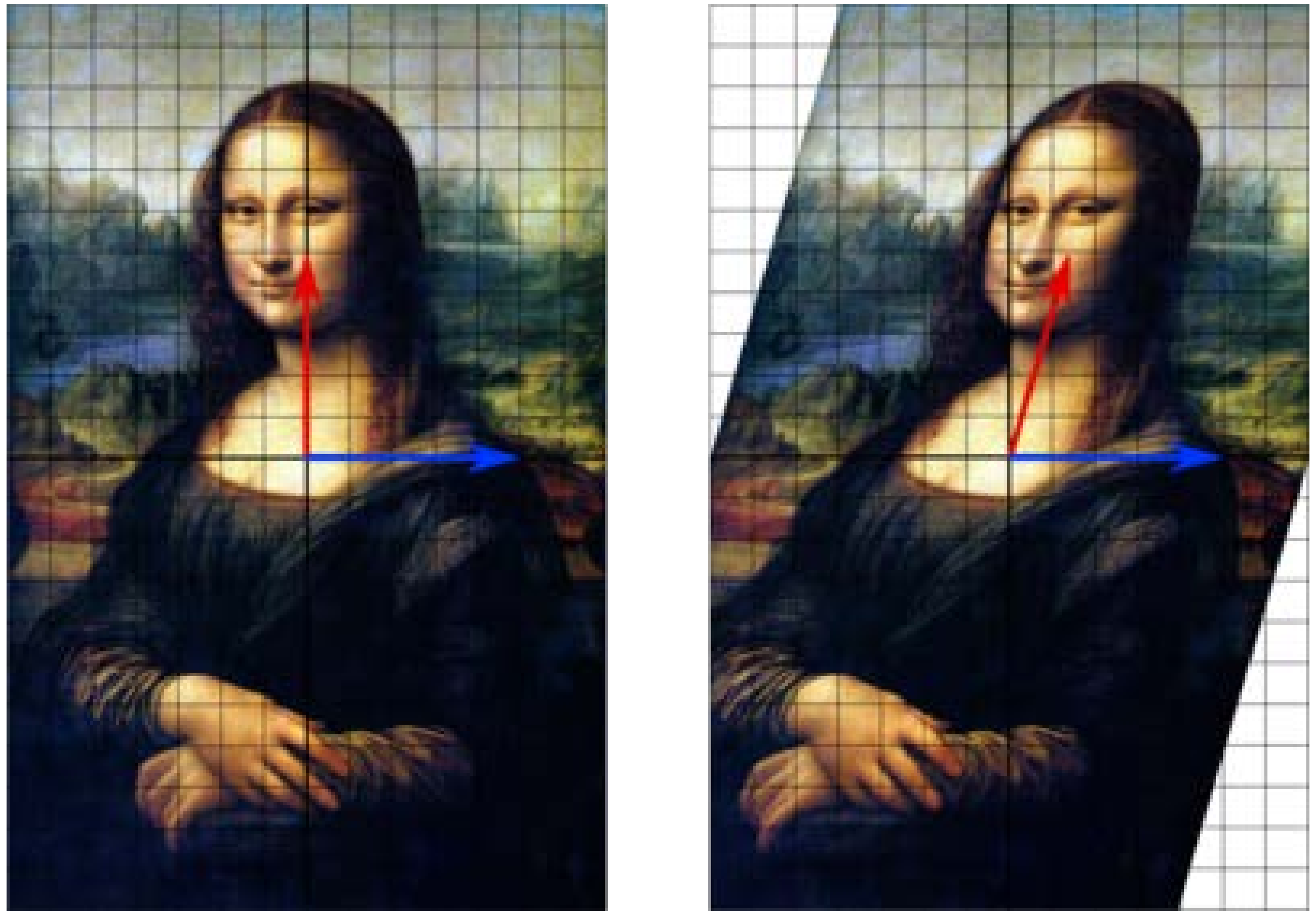
- **차원 축소**

  관찰 공간 위의 샘플들에 기반으로 잠재 공간을 파악하는 것

2. 주성분분석(PCA: Principal Component Analysis)

2-1. 선형대수 정리

- 아래와 같은 선형변환에 의해 파란색 벡터는 방향이 변하지 않음



2-1. 선형대수 정리

- **고유값(Eigenvalue)과 고유벡터(Eigenvector)**

  정사각행렬 A 에 대해 영벡터가 아닌 벡터 x에 대해 $Ax = \lambda x$일 때,

  λ를 고유값, x 를 고유벡터라 함

- **특성방정식(Characteristic Equation)**

  $(\lambda I - A)x = 0$의 영공간(Null space)이 영벡터가 아닌 벡터를

  포함해야 하므로 최고차항 계수가 1인 n차 방정식

  $\det(\lambda I - A) = 0$가 성립해야 한다. 이를 A 의 특성방정식이라 한다.

2-1. 선형대수 정리

- **대각화(Diagonalization)**

  고유값들을 대각성분으로 갖는 행렬을 D, 고유값들에 대응하는 고유벡터들을 열벡터로 갖는 행렬을 Q라 하면

  $$A = QDQ^{-1}$$

  과 같이 표현가능하고, 이를 대각화라 한다.

- **정리(Theorem)**

  대칭(real Symmetric)행렬은 항상 직교(orthogonal)행렬로 대각화할수 있다.

2-1. 선형대수 정리

Example 6.1.1 Find the eigenvalues and eigenvectors of

$$A = \begin{bmatrix} 2 & \sqrt{2} \\ \sqrt{2} & 1 \end{bmatrix}.$$

Solution: The characteristic polynomial is

$$\det(\lambda I - A) = \det \begin{bmatrix} \lambda - 2 & -\sqrt{2} \\ -\sqrt{2} & \lambda - 1 \end{bmatrix} = \lambda^2 - 3\lambda = \lambda(\lambda - 3).$$

Thus the eigenvalues are $\lambda_1 = 0$ and $\lambda_2 = 3$. To determine the eigenvectors belonging to λ_i's, we should solve the homogeneous system of equations $(\lambda_i I - A)\mathbf{x} = \mathbf{0}$ for each λ_i's.

2-1. 선형대수 정리

For $\lambda_1 = 0$, the system of equations $(\lambda_1 I - A)\mathbf{x} = \mathbf{0}$ becomes

$$\begin{cases} -2\,x_1 & - & \sqrt{2}\,x_2 & = & 0, \\ -\sqrt{2}\,x_1 & - & x_2 & = & 0, \end{cases} \quad \text{or} \quad x_2 = -\sqrt{2}\,x_1.$$

Hence, $\mathbf{x}_1 = (x^1,\ x^2) = (-1,\ \sqrt{2})$ is an eigenvector belonging to $\lambda_1 = 0$, and $E_0 = \{t\mathbf{x}_1\ :\ t \in \mathbb{R}\}$.

For $\lambda_2 = 3$, an eigenvector belonging to $\lambda_2 = 3$, as the solutions of the system of equations $(\lambda_2 I - A)\mathbf{x} = \mathbf{0}$, is $\mathbf{x}_2 = (\sqrt{2},\ 1)$, and so $E_3 = \{t\mathbf{x}^2\ :\ t \in \mathbb{R}\}$. Note that the eigenvectors $\mathbf{x}_1$ and $\mathbf{x}_2$ belonging to the eigenvalues λ_1 and λ_2 respectively are linearly independent. $\square$

2-1. 선형대수 정리(SVD, 생략가능)

Let $A \in \mathcal{M}_{m\times n}(\mathbb{C})$ be a matrix of rank k. Then $A^H A \in \mathcal{M}_n(\mathbb{C})$ is Hermitian with real eigenvalues and has n orthonormal eigenvectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ forming a basis for $\mathbb{C}^n$ so that

$$(A^H A)\mathbf{u}_i = \lambda_i \mathbf{u}_i, \qquad \mathbf{u}_i \cdot \mathbf{u}_j = \mathbf{u}_i{}^H \mathbf{u}_j = \delta_{ij}.$$

(When A is a real matrix, we take A^T and $\mathbf{u}_i^T$ instead of A^H and $\mathbf{u}_i^H$.) Since

$$\|A\mathbf{u}_i\|^2 = \mathbf{u}_i^H (A^H A)\mathbf{u}_i = \lambda_i \mathbf{u}_i^H \mathbf{u}_i = \lambda_i,$$

$\lambda_i \geq 0$ for all $i = 1, \ldots, n$. Thus, we may assume that $\lambda_1, \ldots, \lambda_k$ are positive and the remaining $n-k$ eigenvalues $\lambda_{k+1}, \ldots, \lambda_n$ are zero, so that the last $n-k$ vectors $\mathbf{u}_{k+1}, \ldots, \mathbf{u}_n$ form an orthonormal basis for $E_0(A) = \mathcal{N}(A)$, and so the first k vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ orthogonal to $\mathcal{N}(A)$ form an orthonormal basis for $\mathcal{R}(A) = \mathcal{N}(A)^\perp$. Let $\sigma_i = \sqrt{\lambda_i}$ for $i \leq k$, and set

$$\mathbf{v}_i \equiv \frac{1}{\sigma_i} A\mathbf{u}_i, \quad \text{or } A\mathbf{u}_i \equiv \sigma_i \mathbf{v}_i \in \mathcal{C}(A) \subseteq \mathbb{C}^m, \quad \text{for } i \leq k.$$

Then $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is an orthonormal basis for $\mathcal{C}(A)$ in $\mathbb{C}^m$ since

$$\mathbf{v}_i^H \mathbf{v}_j = \frac{\mathbf{u}_i^H A^H A \mathbf{u}_j}{\sigma_i \sigma_j} = \frac{\lambda_j \mathbf{u}_i{}^H \mathbf{u}_j}{\sigma_i \sigma_j} = \delta_{ij}.$$

Extend this to an orthonormal basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_k, \mathbf{v}_{k+1}, \ldots, \mathbf{v}_m\}$ for $\mathbb{C}^m$ by using the Gram-Schmidt process. Let Q_1 and Q_2 be the unitary matrices defined as:

$$Q_1 = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_n], \quad Q_2 = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_m],$$

2-1. 선형대수 정리(SVD, 생략가능)

Definition 6.8.1 The nonzero diagonal entries $\sigma_i = \sqrt{\lambda_i}$'s of the $k \times k$ diagonal matrix Σ_k on the upper left part of E are called the **singular values** of A.

Theorem 6.8.1 (Singular value decomposition) *Any matrix $A \in \mathcal{M}_{m\times n}(\mathbb{C})$ of rank k has a* **singular value decomposition***:*

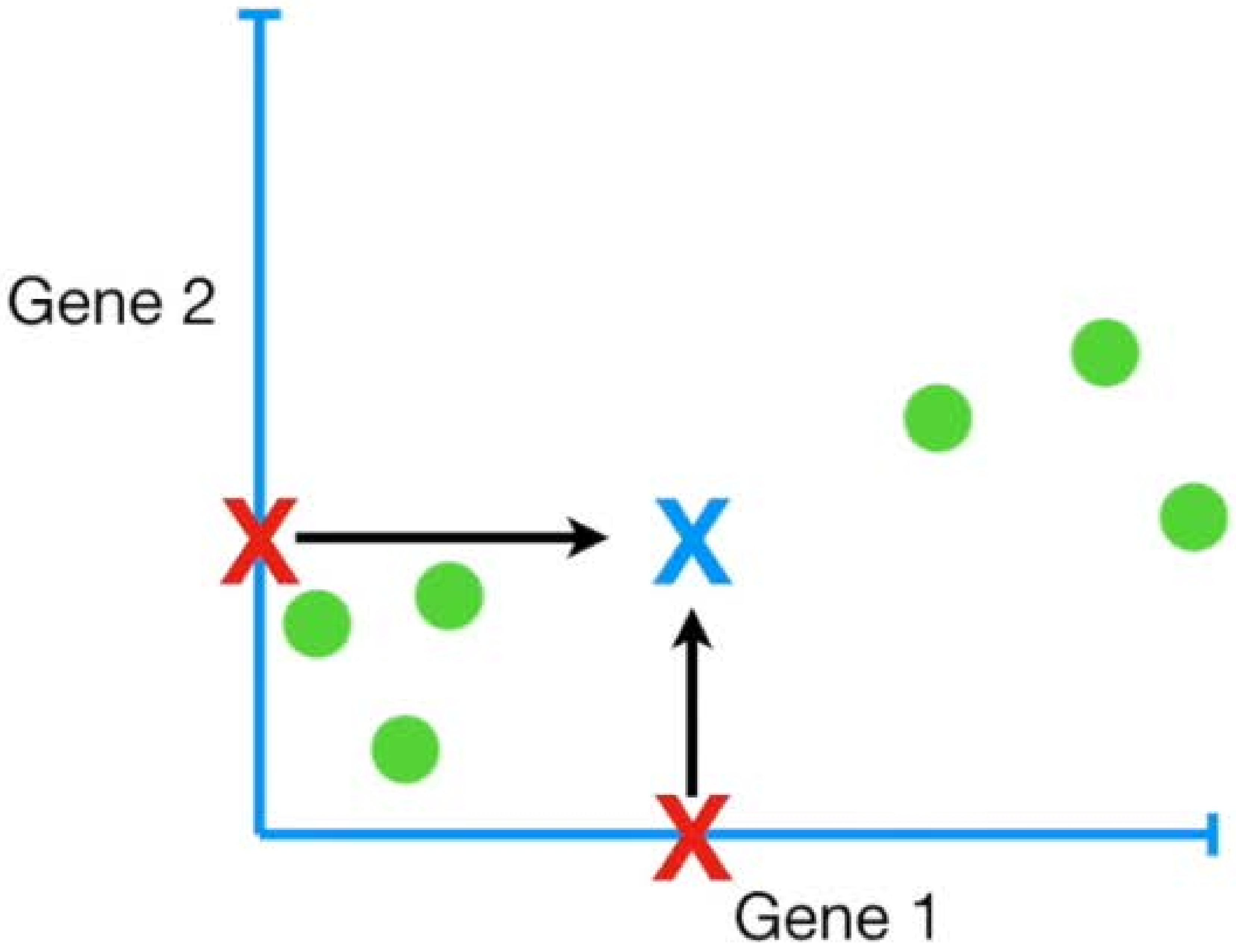
$$A = Q_2 E Q_1^H = \sigma_1 \mathbf{v}_1 \mathbf{u}_1^H + \cdots + \sigma_k \mathbf{v}_k \mathbf{u}_k^H,$$

where the first k columns $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of a unitary matrix Q_1 form an orthonormal basis of eigenvectors of $A^H A$ for $\mathcal{R}(A) \subseteq \mathbb{C}^n$, the first k columns $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of a unitary matrix Q_2 form an orthonormal basis of eigenvectors of AA^H for $\mathcal{C}(A) \subseteq \mathbb{C}^m$ with $A\mathbf{u}_i = \sigma_i \mathbf{v}_i$ for $i \le k$, and the k nonzero diagonals $\sigma_i = \sqrt{\lambda_i}$ of $m \times n$ matrix E are the singular values of A.

2-2. 주성분 분석(PCA)

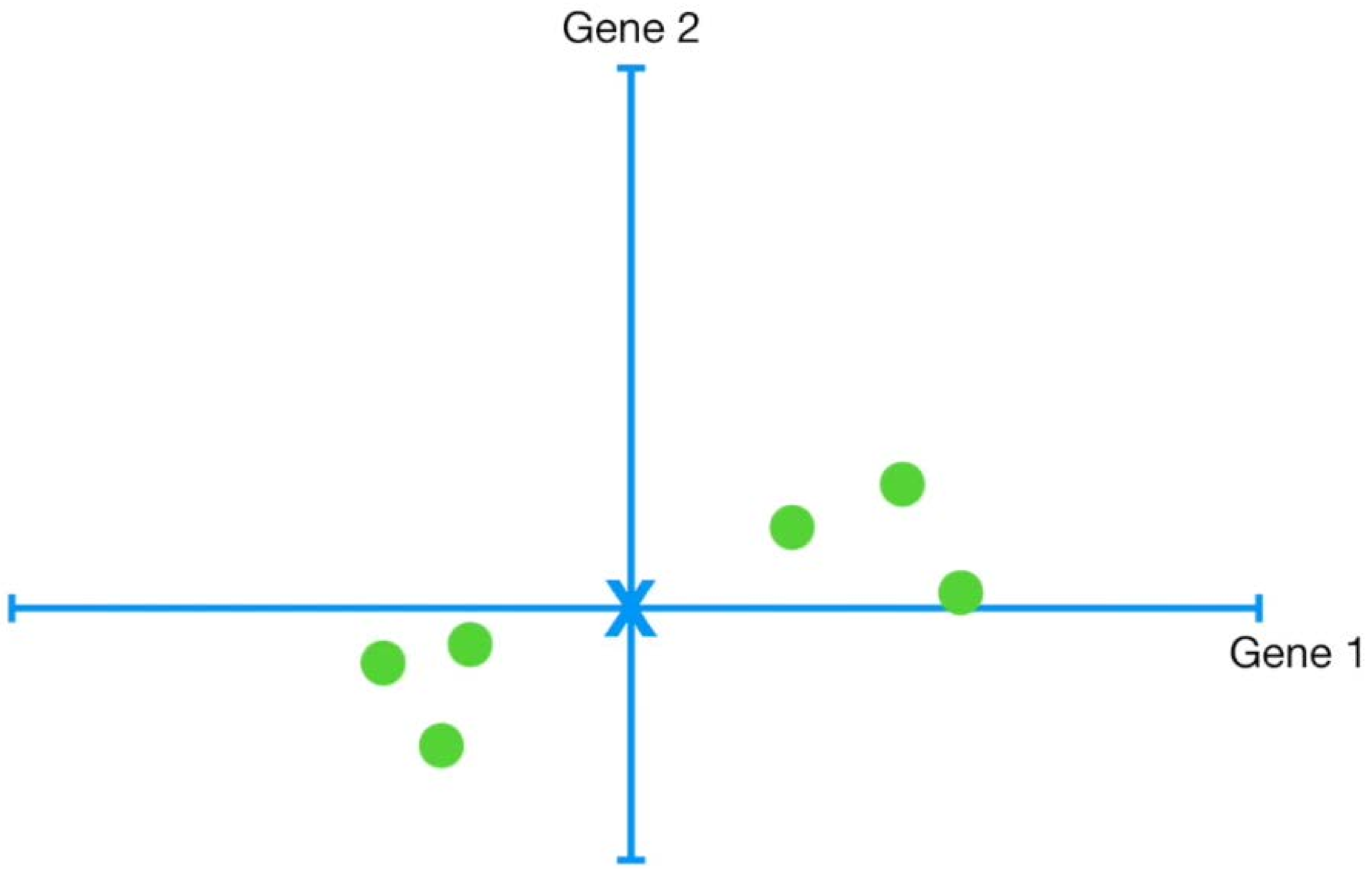
- 피쳐가 2개, 샘플의 개수가 6개인 데이터를 시각화하고 평균을 표시

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



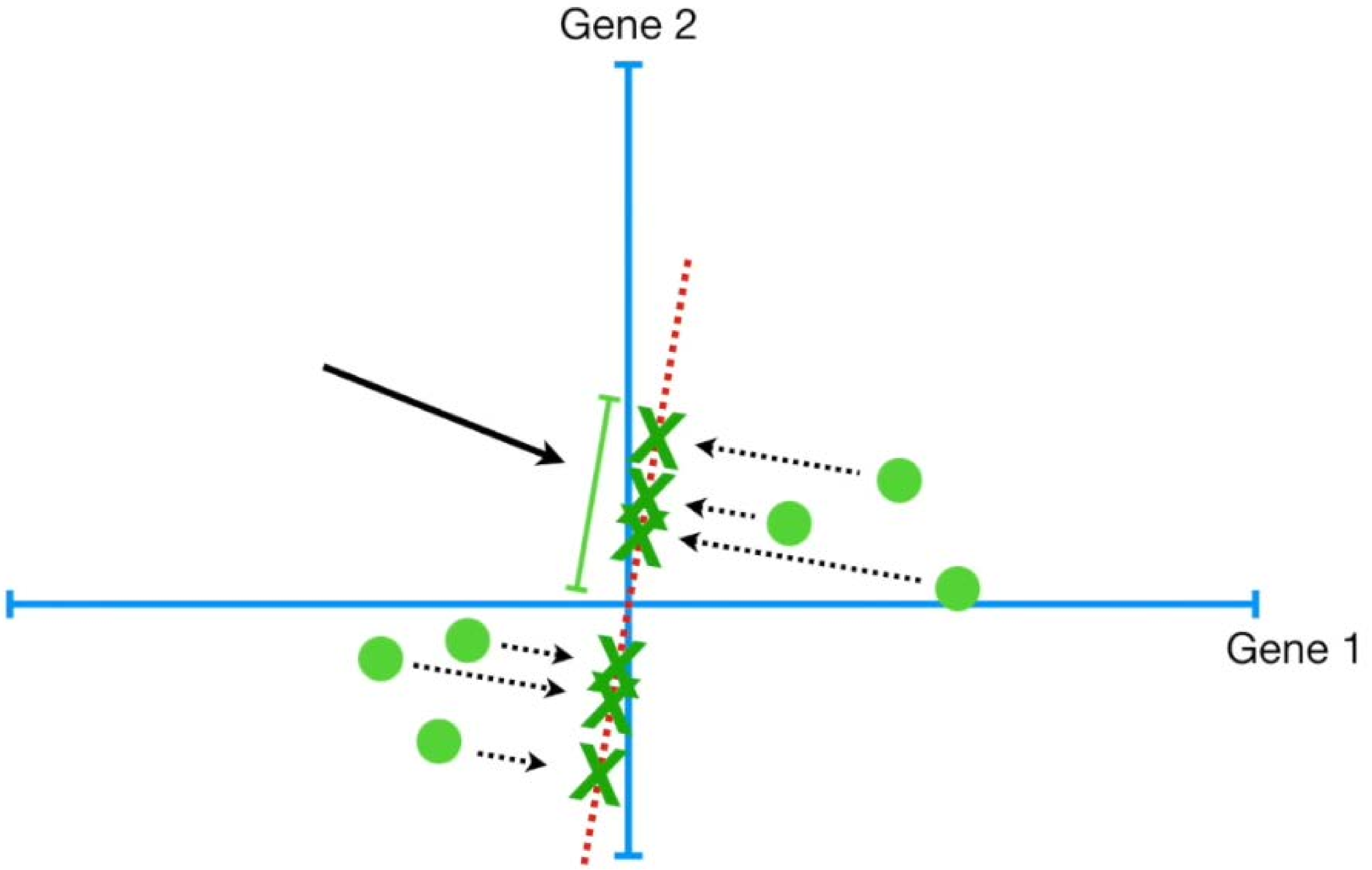
2-2. 주성분 분석(PCA)

- 모든 데이터에서 각 행의 평균을 빼서 모든 행의 평균이 0이 되게 함



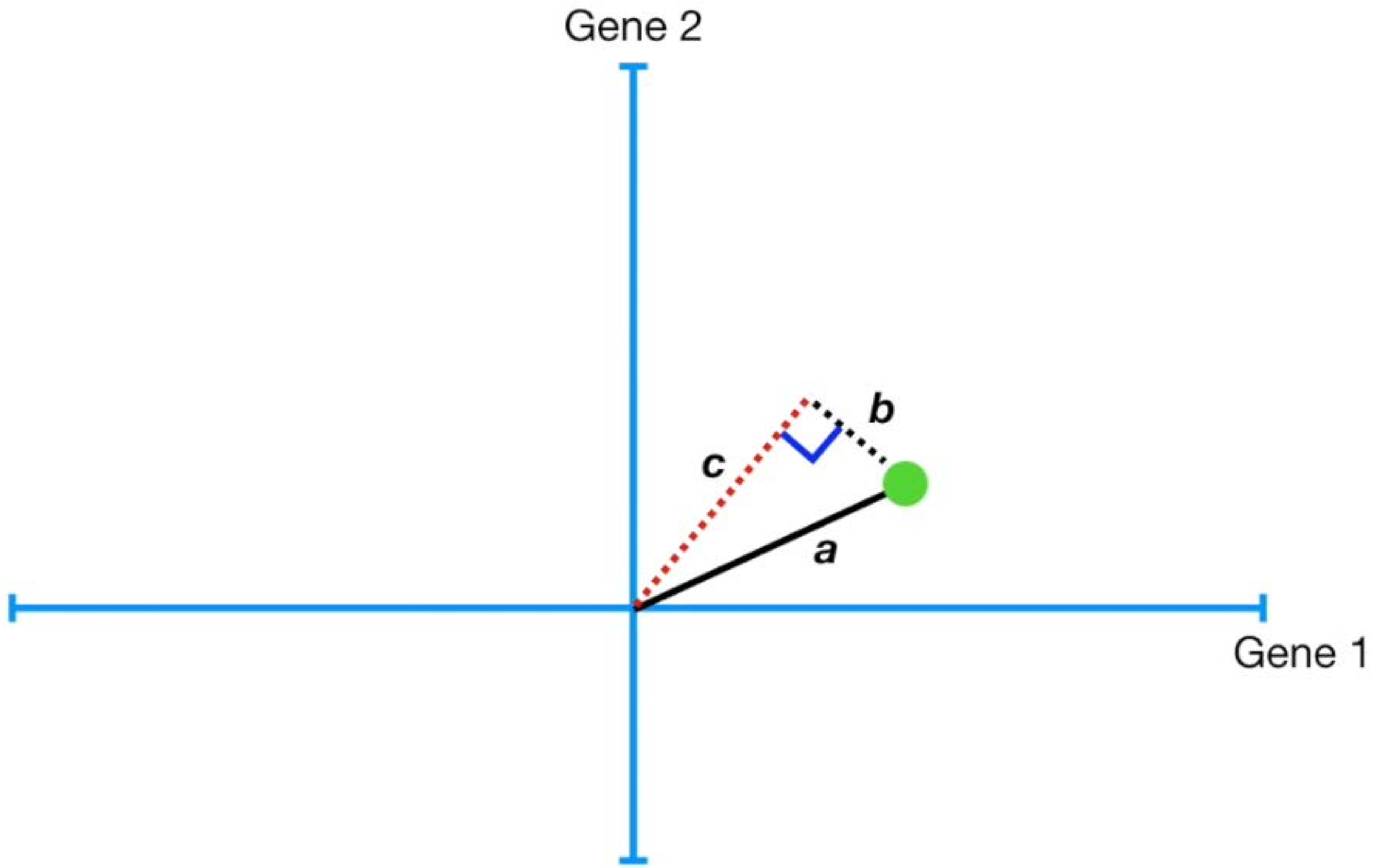
2-2. 주성분 분석(PCA)

- 원점을 지나는 직선 중에서 데이터들을 정사영 했을 때의 분산을 최대로 하는 직선을 찾고자 함



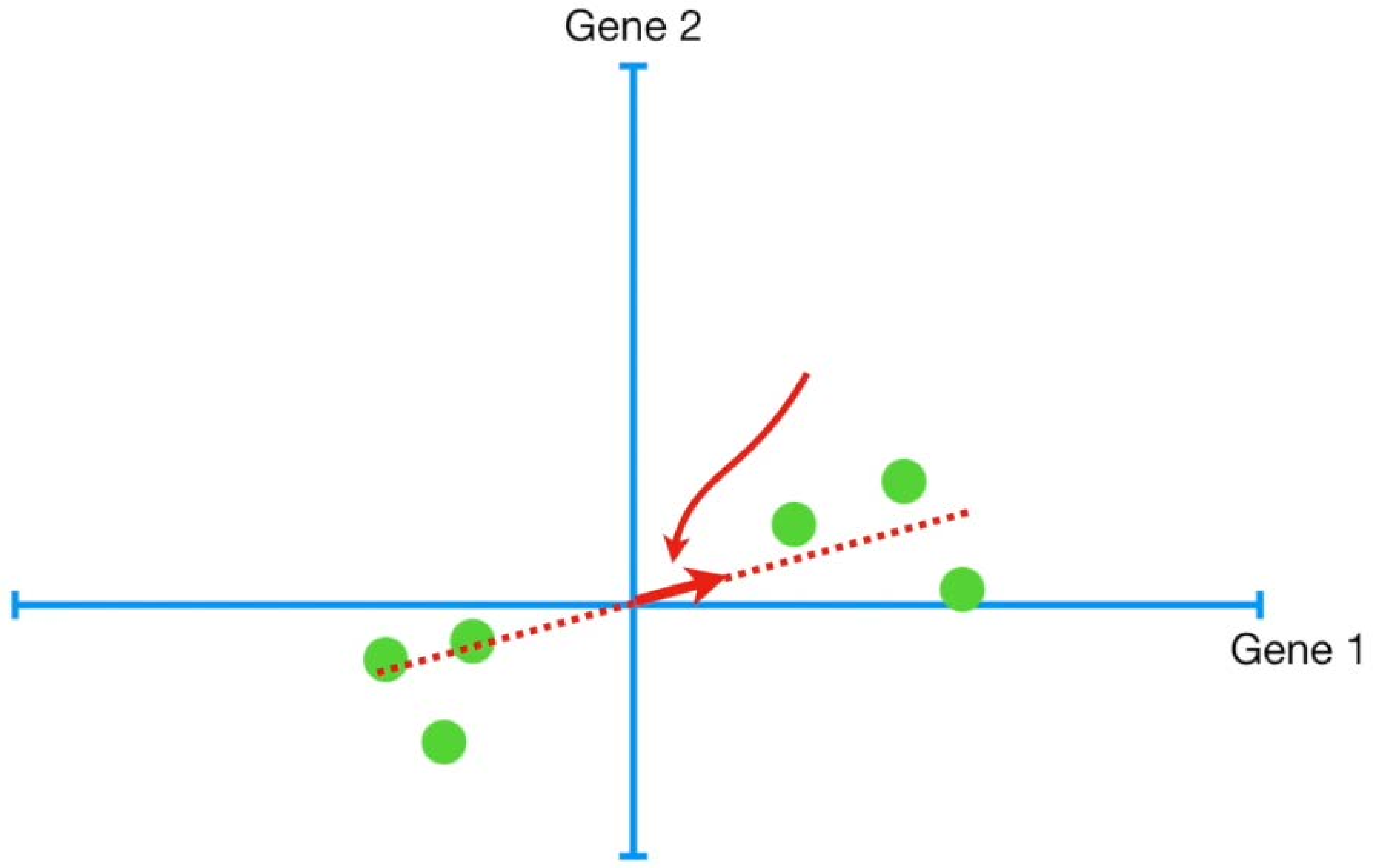
2-2. 주성분 분석(PCA)

- 피타고라스 정리에 의해 $a^2 = b^2 + c^2$이므로 결국 정사영했을 때의 분산을 최대화하는 것은 각 점에서 직선까지의 거리제곱의 합을 최소화하는 것과 같음



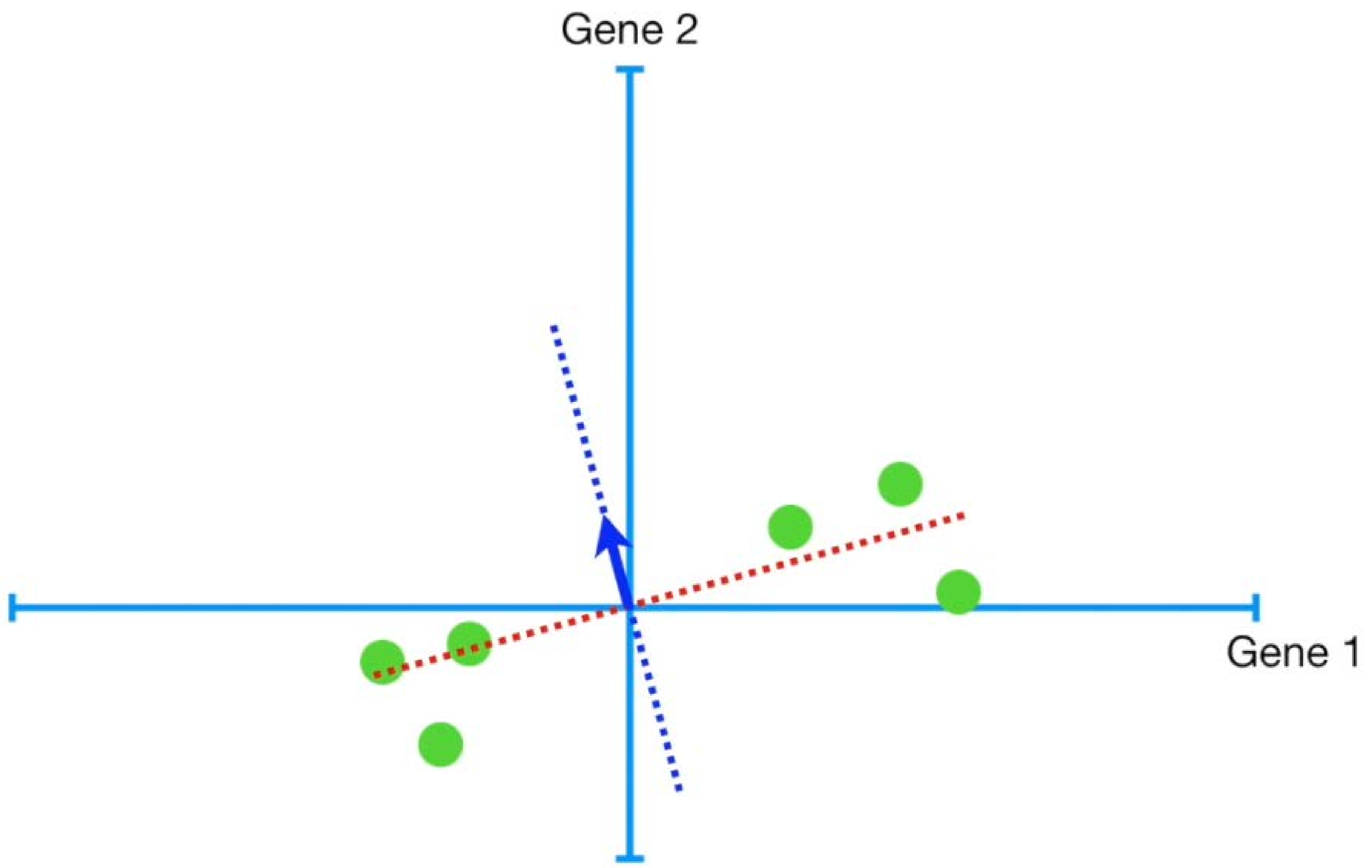
2-2. 주성분 분석(PCA)

- 그렇게 찾은 직선을 첫번째 주성분(PC_1), 빨간색 벡터를 PC_1의 싱귤러(singular) 벡터라고 함
- PC_1의 싱귤러 벡터는 공분산 행렬의 가장 큰 고유값(λ_1)에 대한 고유벡터가 됨 (사실 대각화가 아니라 SVD를 이용해서 접근 가능)



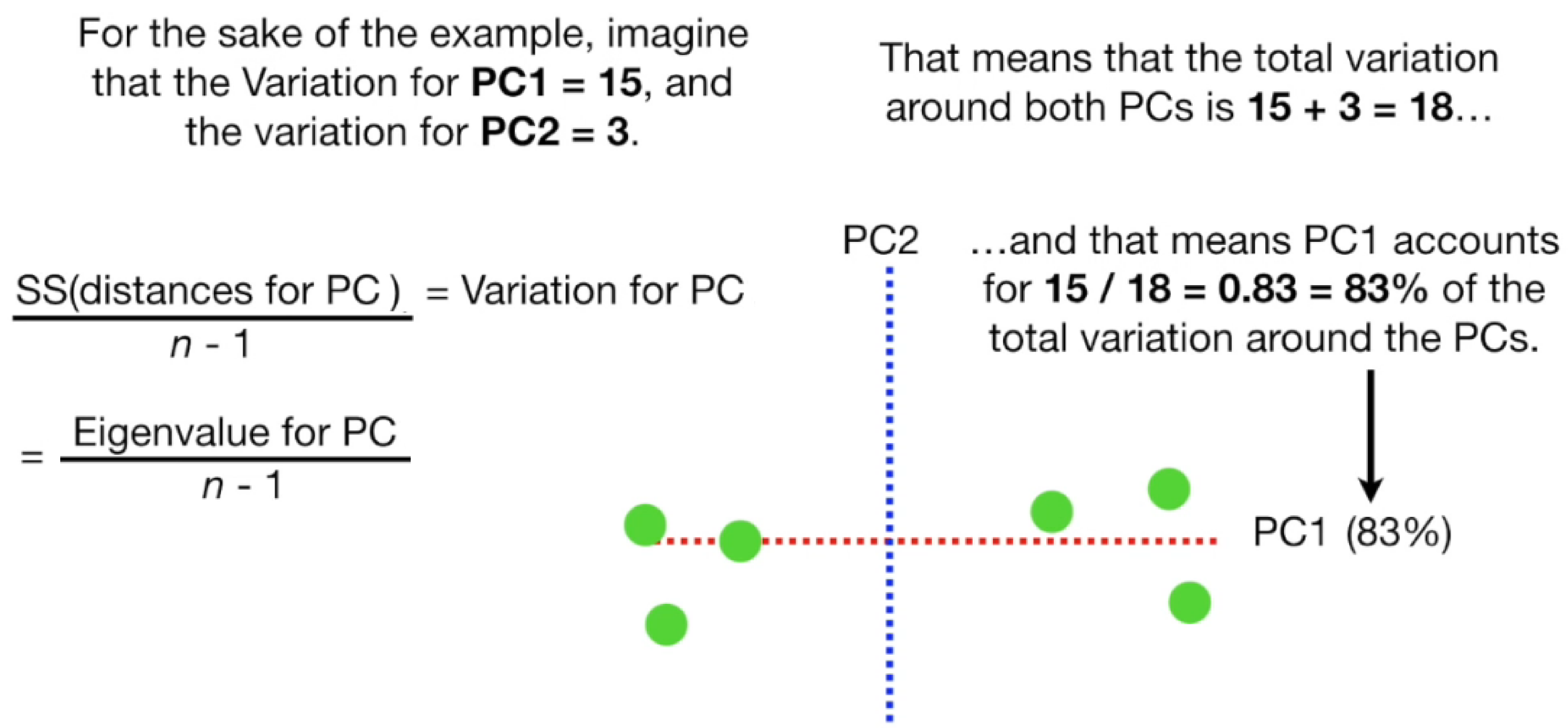
2-2. 주성분 분석(PCA)

- PC_2는 PC_1과 수직인 직선 중 정사영했을 때의 분산이 가장 큰 직선
- PC_2의 싱귤러 벡터는 공분산 행렬의 두번째 큰 고유값(λ_2)에 대한 고유벡터



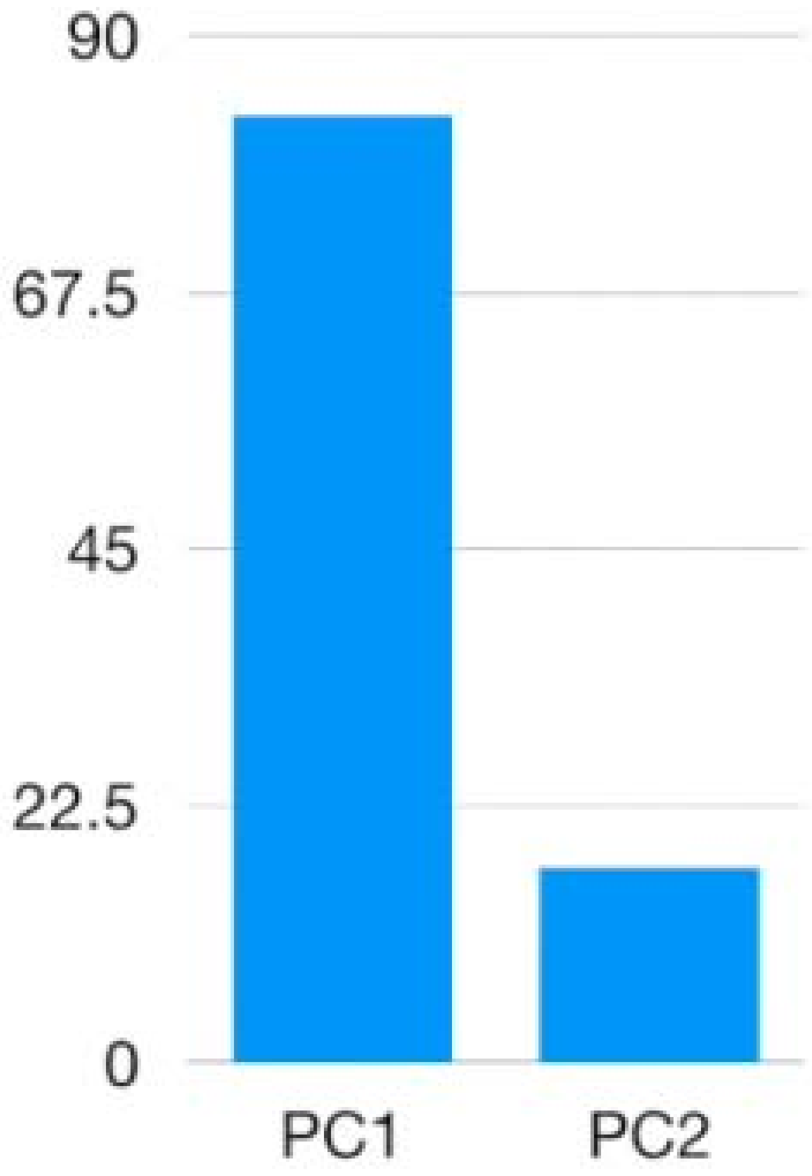
2-2. 주성분 분석(PCA)

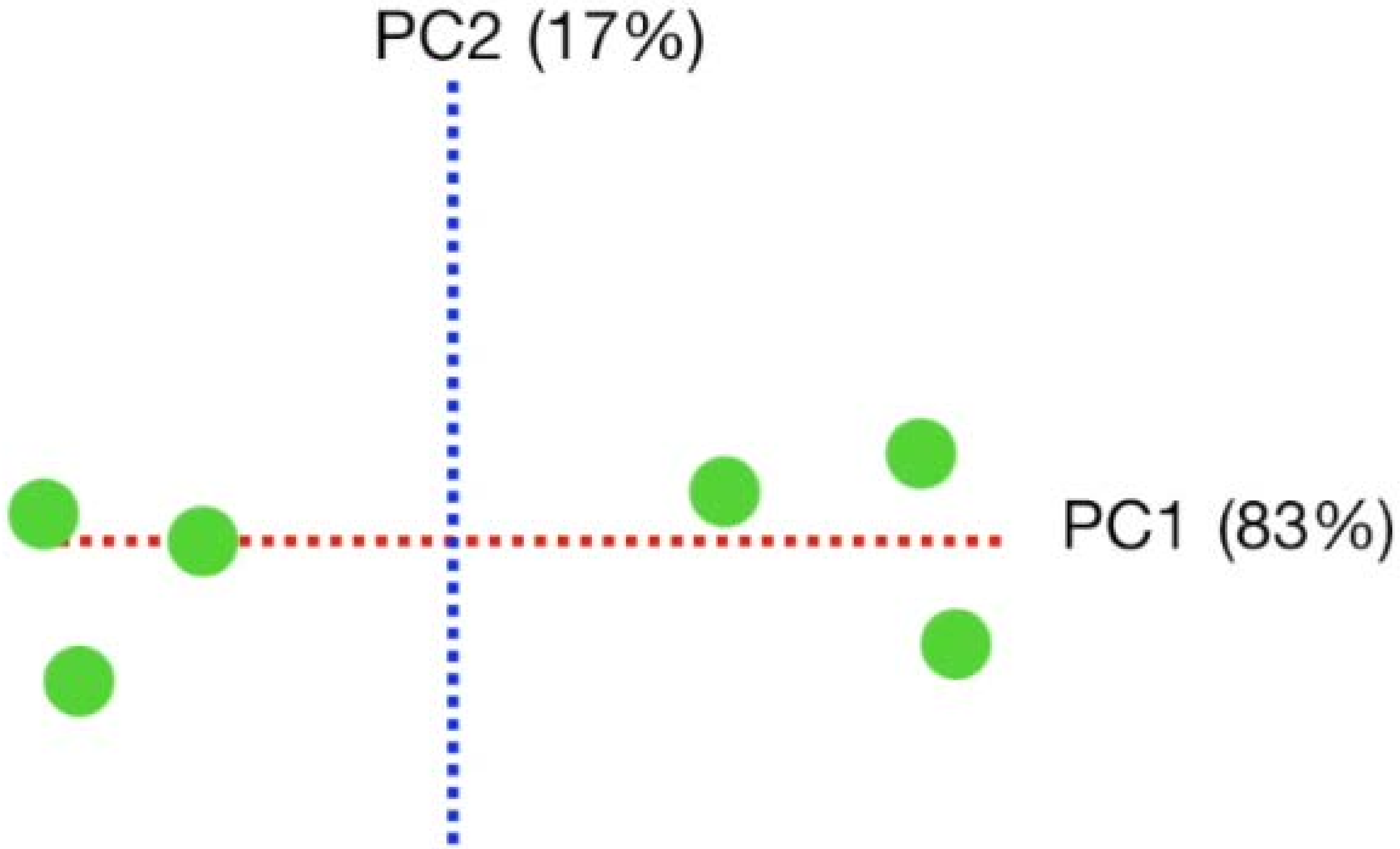
- PC_1과 PC_2가 각각 얼마나 중요한지 알아보기 위해 비율을 계산



2-2. 주성분 분석(PCA)

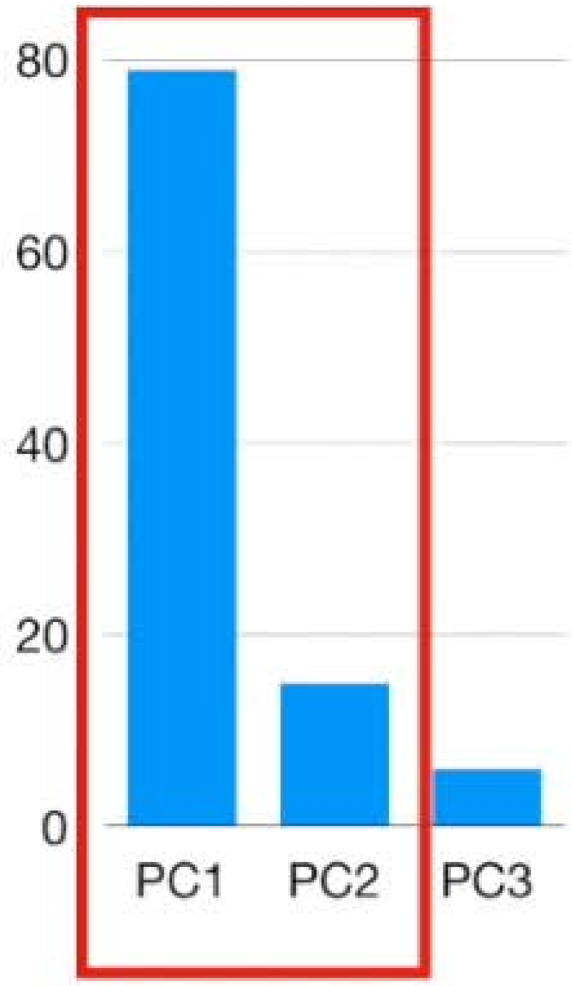
- 앞에서 계산한 비율을 히스토그램으로 나타낸 것을 스크리 플랏(Scree Plot)이라 함

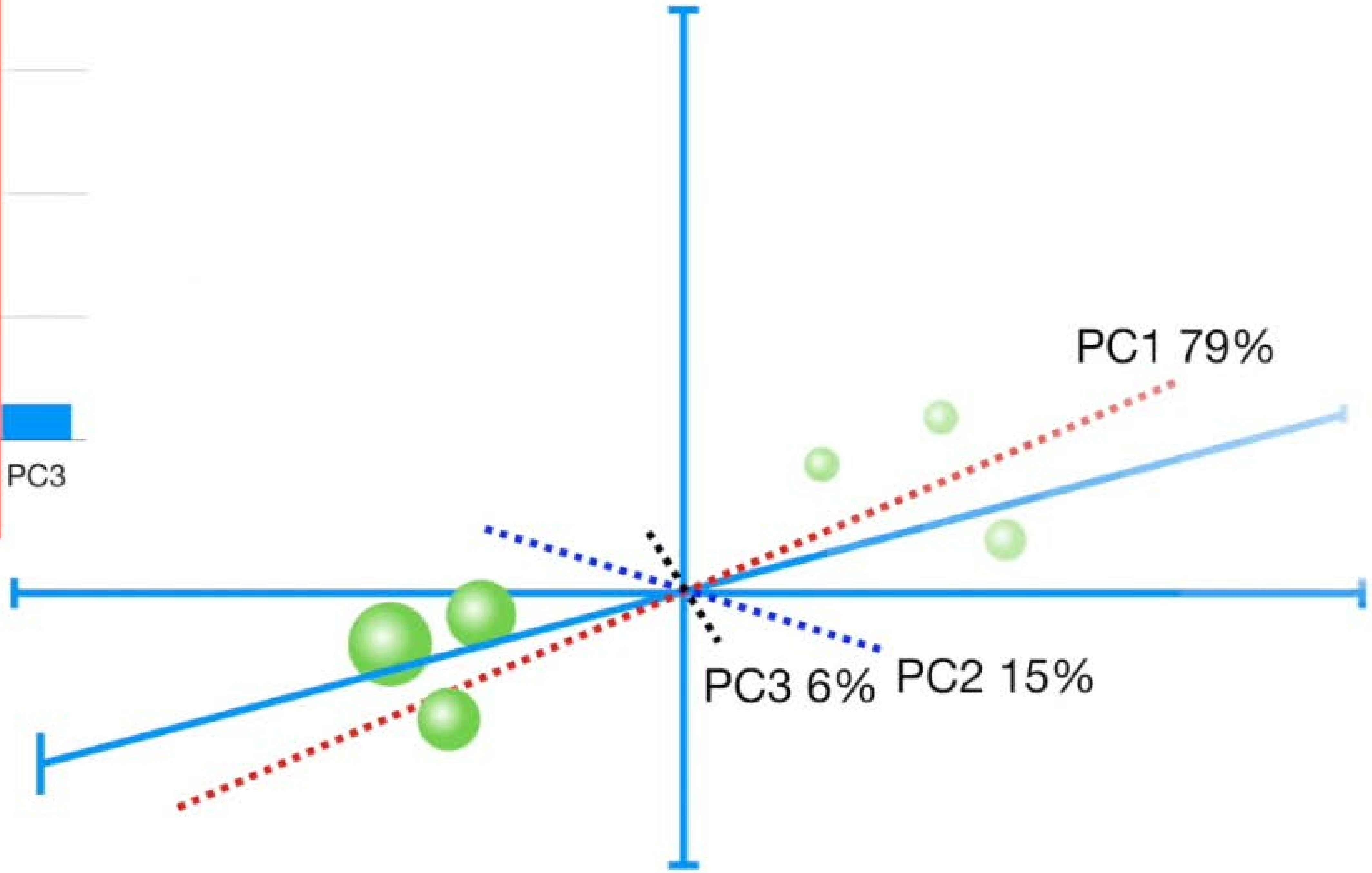




2-2. 주성분 분석(PCA)

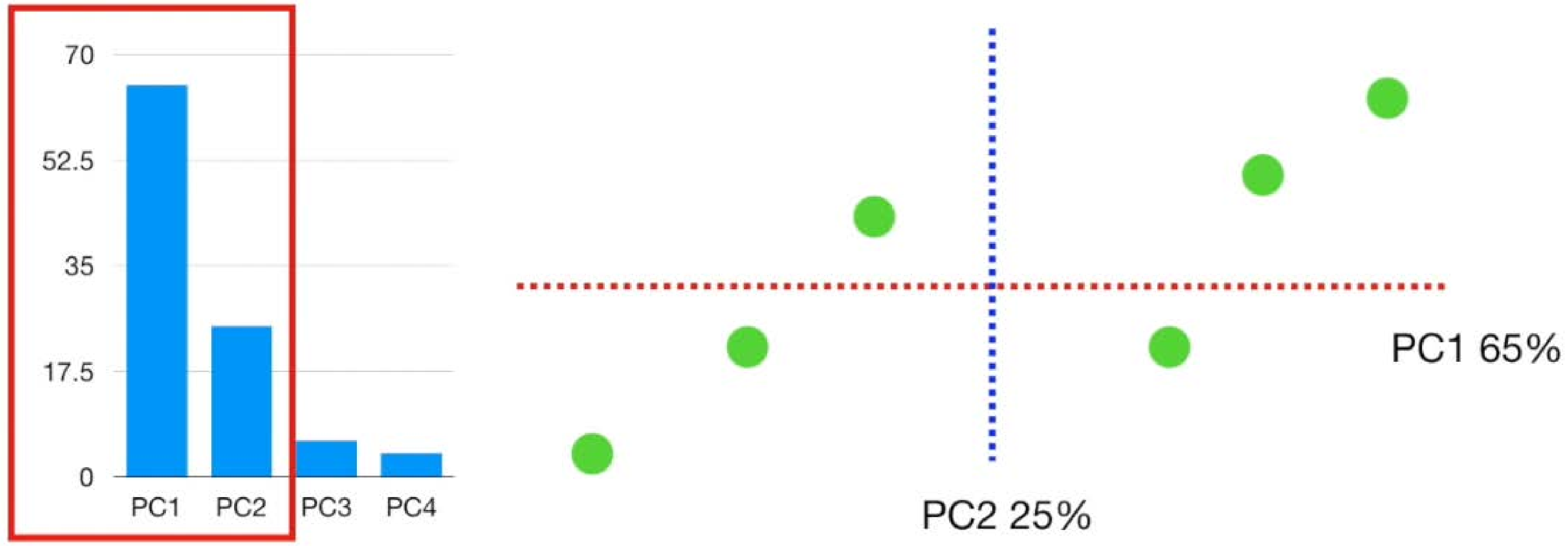
피쳐가 3개인 경우 스크리 플랏에서 기여도를 보고 2개만 선택할 수 있음



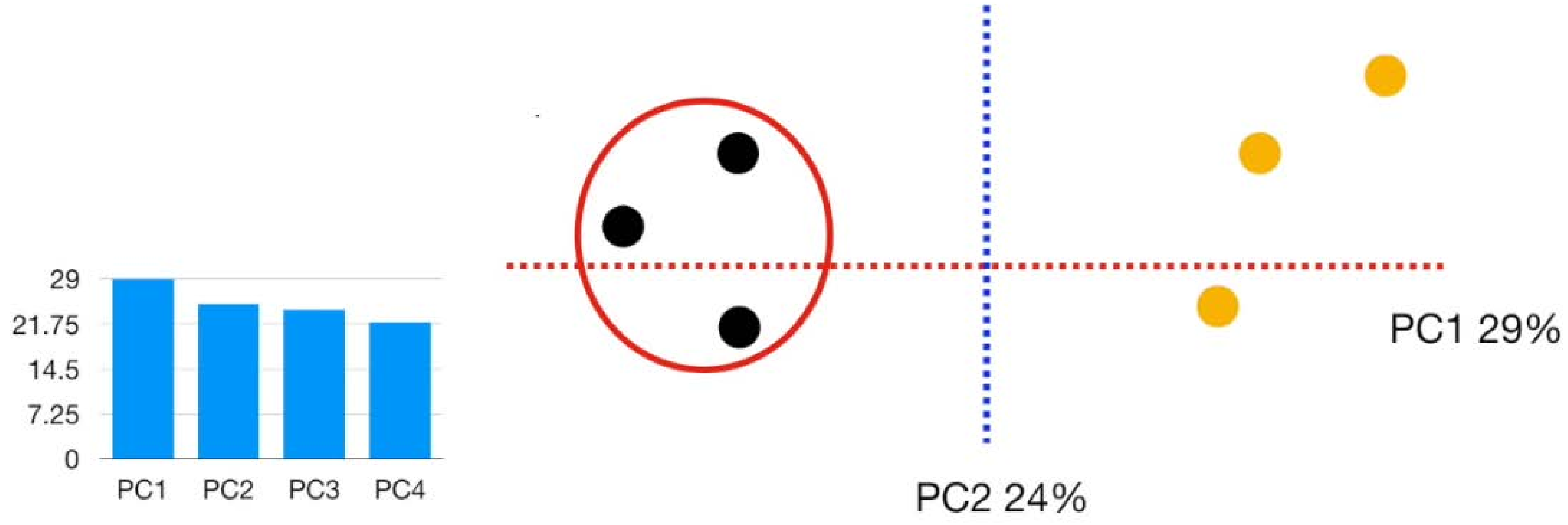


2-2. 주성분 분석(PCA)

- 피쳐가 4개인 경우 다음과 같이 기여도가 큰 2개를 선택하면 좋음



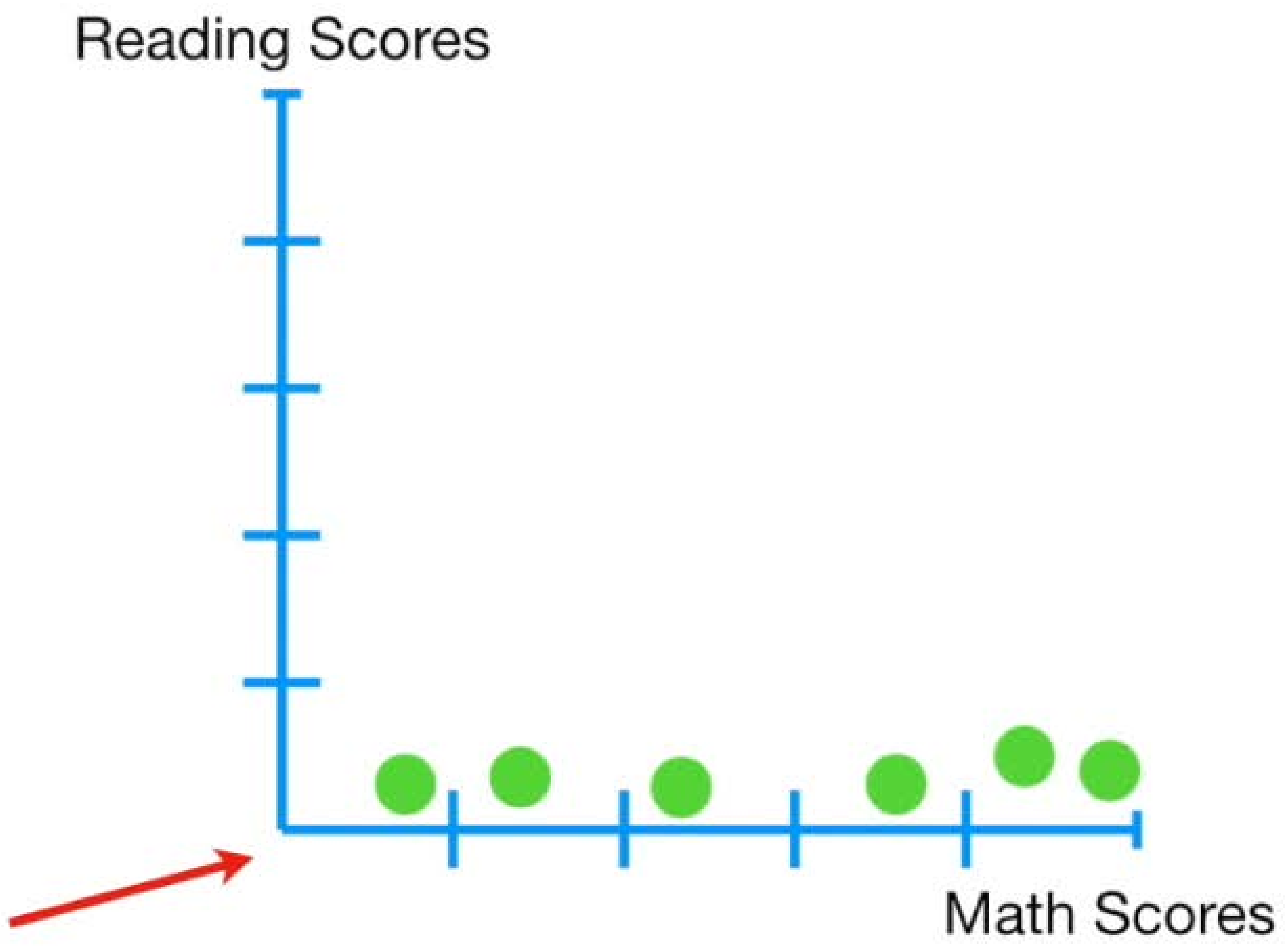
- 다음 그림과 같이 기여도 차이가 크지 않은 경우에도 2개를 선택해서 clustering을 하는데 사용할수 있다.(원래 데이터 복원에는 좋지 않음)



2-2. 주성분 분석(PCA)

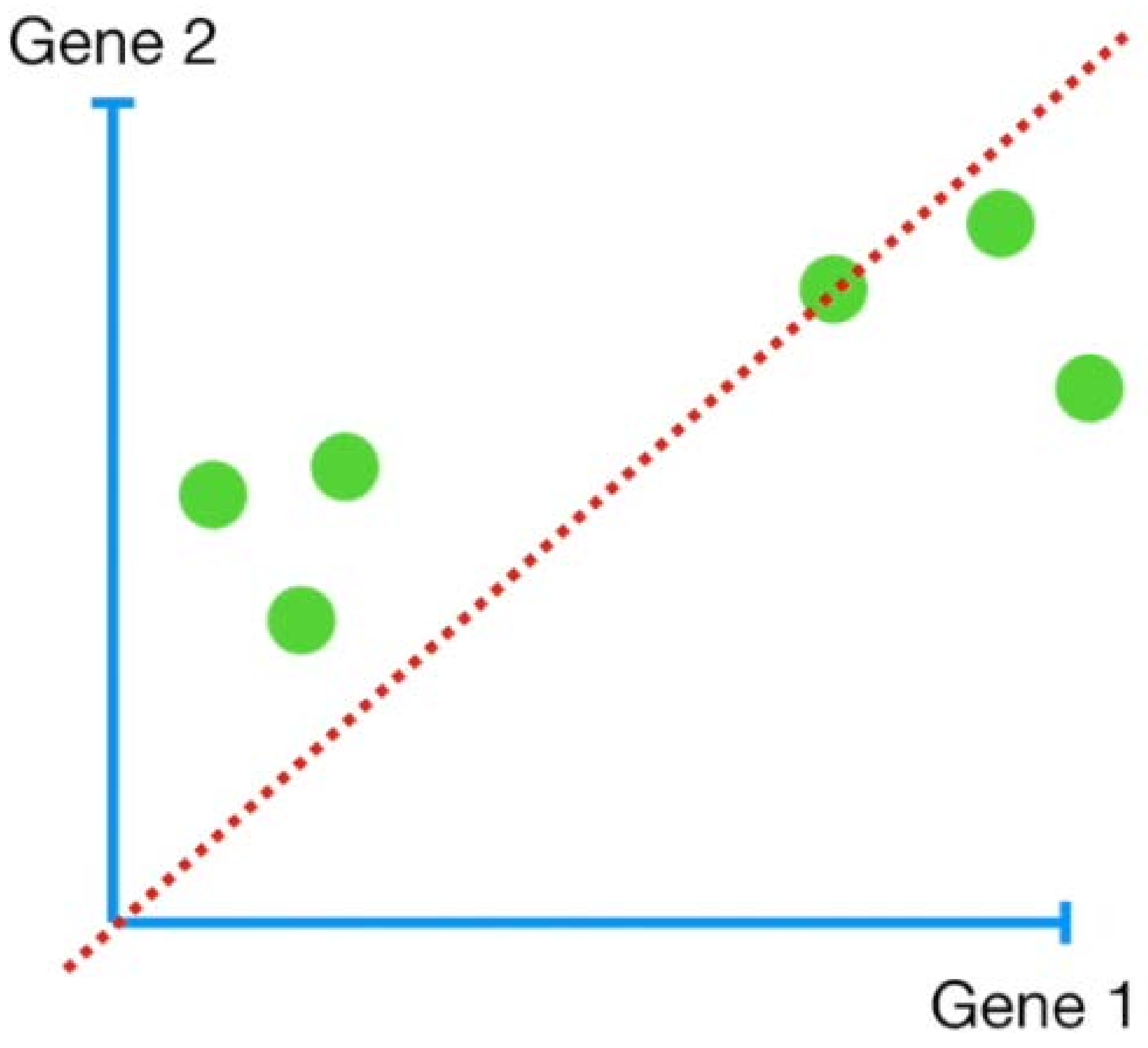
Tip1. 아래의 경우 PCA를 실행하면 수학 점수가 읽기 점수보다 10배 중요한 것으로 나오는데 이는 단지 Scaling이 되어있지 않기 때문

	Student 1	Student 2	Student 3	Student 4	...
Math	95	88	93	75	...
Reading	9	8	10	7	...



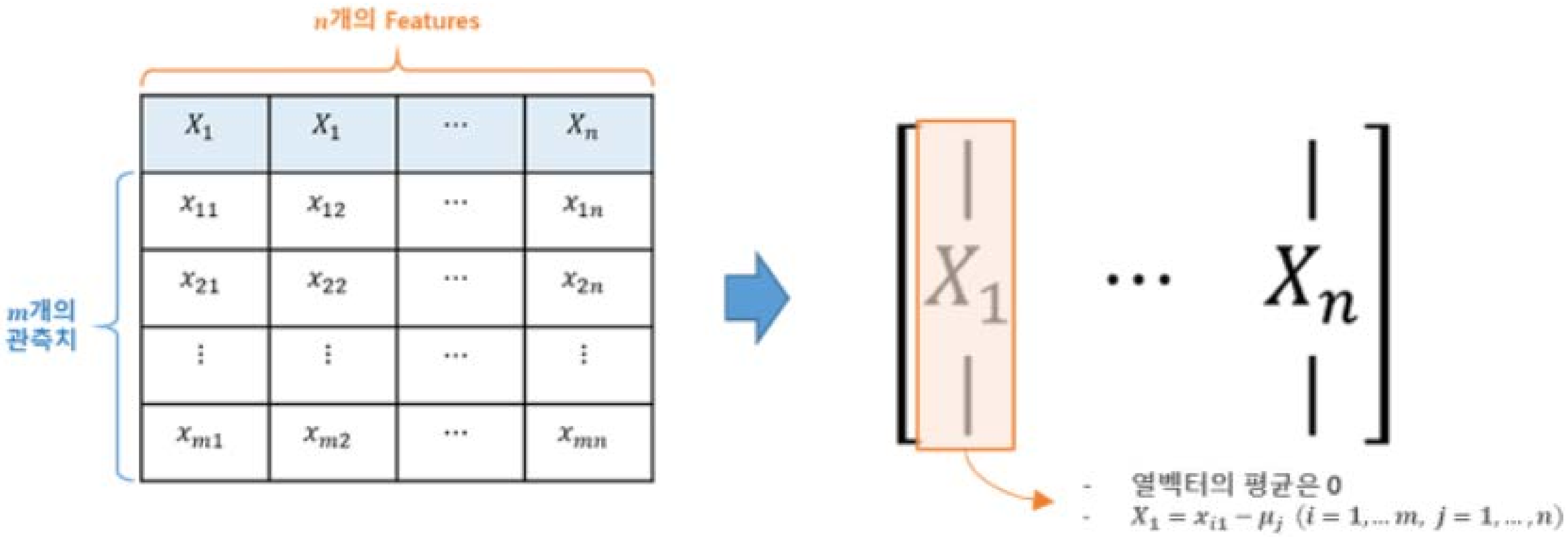
2-2. 주성분 분석(PCA)

Tip2. PCA를 수행하는 프로그램들 중에 평균을 0으로 바꾸는 작업을 해주지 않는 경우가 있는데, 그럴 경우 원치 않는 결과가 나올수 있으므로 미리 확인하는 것이 바람직



2-2. 주성분 분석(PCA)

- PCA 요약 1



$$cov(\mathbf{X}) = \frac{1}{m-1}\mathbf{X}^T\mathbf{X}$$

2-2. 주성분 분석(PCA)

- PCA 요약 2

PCA의 목적은 원 데이터(original data)의 분산을 최대한 보존하는 축을 찾아 투영(projection)하는 것이다. 예를 들어, 평균 0으로 조정한(편차를 구한) 데이터셋 $\mathbf{X}$를 단위벡터 $\vec{e}$인 임의의 축 P에 투영한다고 했을 때, $\mathbf{X}$의 투영된 결과는 $\mathbf{X}\vec{e}$로 표현할 수 있다. 이때의 분산은 다음과 같이 나타낼 수 있다.

$$\begin{aligned}
Var\left[\mathbf{X}\vec{e}\right] &= \frac{1}{m-1}\sum_{i=1}^{m}\left[X\vec{e} - E\left(X\vec{e}\right)\right]^2 \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left[X\vec{e} - E(X)\vec{e}\right]^2, \quad (E(X)=0) \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left(X\vec{e}\right)^2 \quad = \frac{1}{m-1}\left(\mathbf{X}\vec{e}\right)^T\left(\mathbf{X}\vec{e}\right) \\
&= \frac{1}{m-1}\vec{e}^T\mathbf{X}^T\mathbf{X}\vec{e} \quad = \vec{e}^T\left(\frac{\mathbf{X}^T\mathbf{X}}{m-1}\right)\vec{e} \\
&= \vec{e}^T\mathbf{C}\vec{e}
\end{aligned}$$

따라서, PCA는 $Var\left[\mathbf{X}\vec{e}\right] = \vec{e}^T\mathbf{C}\vec{e}$를 목적함수로 하는 최대화 문제이며 이때 제약조건은 $\|\vec{e}\|^2 = 1$ 이다.

$$\begin{aligned}
&\text{maximize} \quad \vec{e}^T\mathbf{C}\vec{e} \\
&\text{s.t.} \quad \|\vec{e}\|^2 = 1
\end{aligned}$$

2-2. 주성분 분석(PCA)

- PCA 요약 3

라그랑제 승수법을 이용하여 계산할 수 있다. 위의 식을 라그랑지안 함수 L로 나타내면 다음과 같다.

$$L(\vec{e}, \lambda) = \vec{e}^T \mathbf{C} \vec{e} - \lambda \left(\vec{e}^T \vec{e} - 1 \right)$$

라그랑지안 함수 L을 $\vec{e}$에 대해 편미분 하면 다음과 같다.

$$\begin{aligned} \frac{\partial L}{\partial \vec{e}} &= \left(\mathbf{C} + \mathbf{C}^T \right) \vec{e} - 2\lambda \vec{e} \\ &= 2\mathbf{C}\vec{e} - 2\lambda \vec{e} = 0 \end{aligned}$$

$$\therefore \mathbf{C}\vec{e} = \lambda \vec{e}$$

$$\therefore \mathbf{C} = \vec{e} \lambda \vec{e}^T$$

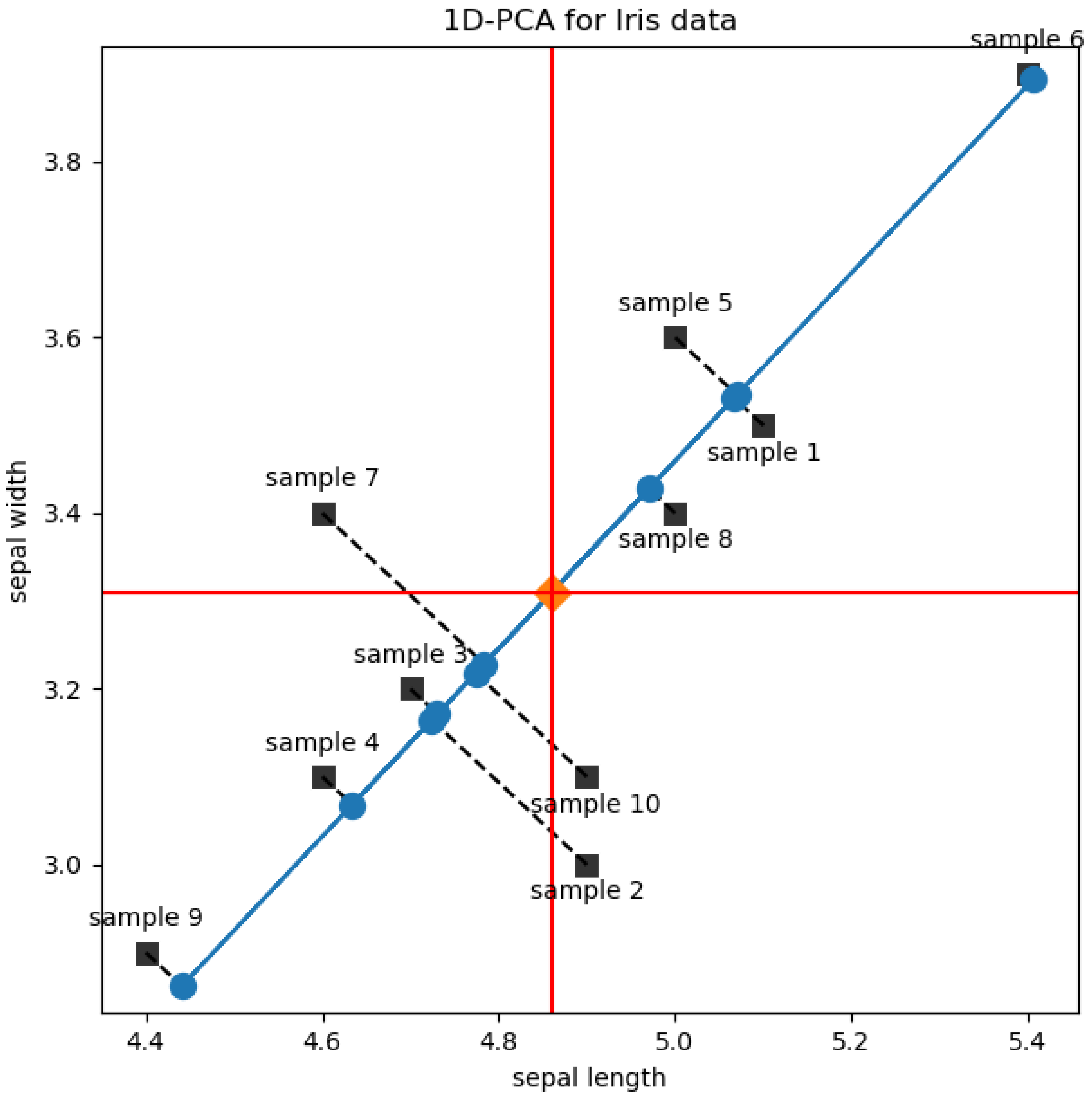
즉, $\mathbf{C}\vec{e} = \lambda \vec{e}$를 만족하는 $\vec{e}$가 바로 분산 $Var[\mathbf{X}\vec{e}]$를 최대화한다.

위의 식에서 $\vec{e}$는 공분산 $\mathbf{C}$의 **고유벡터**(eigenvector)이며, λ는 $\mathbf{C}$의 **고유값**(eigenvalue)이자 eigenvector로 투영했을 때의 **분산**(variance)이다. 이때, 고유벡터의 열벡터를 **주성분**(PC, principal component)이라고 한다. 따라서 고유벡터(eigenvector)에 투영하는 것이 분산이 최대가 된다.

2-2. 주성분 분석(PCA)

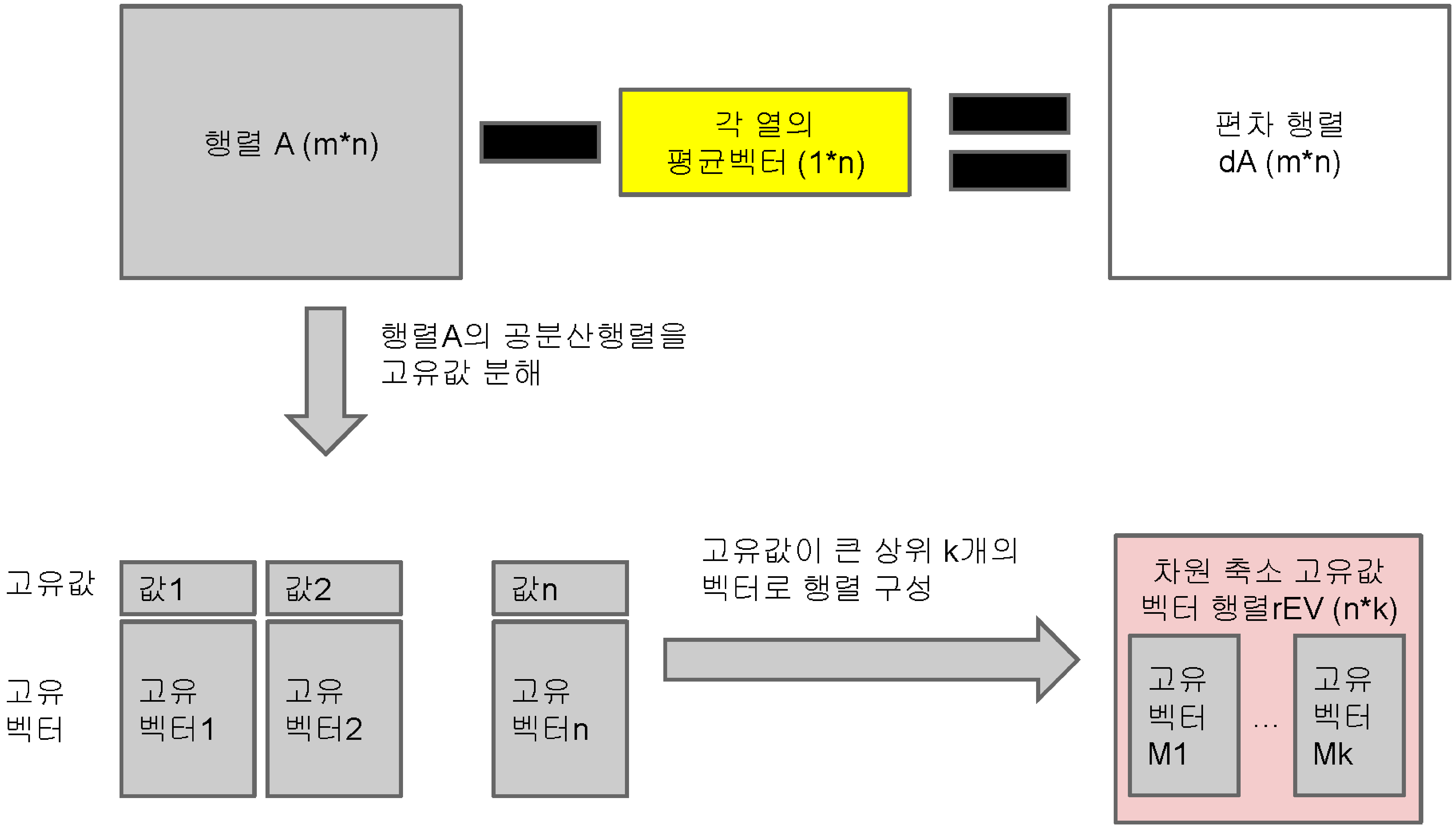
```
iris = load_iris()
N = 10  # 앞의 10송이만 선택
X = iris.data[:N, :2]  # 꽃받침 길이와 꽃받침 폭만 선택
pca1 = PCA(n_components=1)
X_low = pca1.fit_transform(X)
X2 = pca1.inverse_transform(X_low)

plt.figure(figsize=(7, 7))
ax = sns.scatterplot(0, 1, data=pd.DataFrame(X), s=100, color=".2", marker="s")
for i in range(N):
    d = 0.03 if X[i, 1] > X2[i, 1] else -0.04
    ax.text(X[i, 0] - 0.065, X[i, 1] + d, "sample {}".format(i + 1))
    plt.plot([X[i, 0], X2[i, 0]], [X[i, 1], X2[i, 1]], "k--")
```

2-2. 주성분 분석(PCA)


1D-PCA for Iris data
sample 6
sample 5
sample 1
sample 7
sample 8
sample 3
sample 4
sample 10
sample 2
sample 9
3.8
3.6
3.4
3.2
3.0
sepal width
4.4
4.6
4.8
5.0
5.2
5.4
sepal length

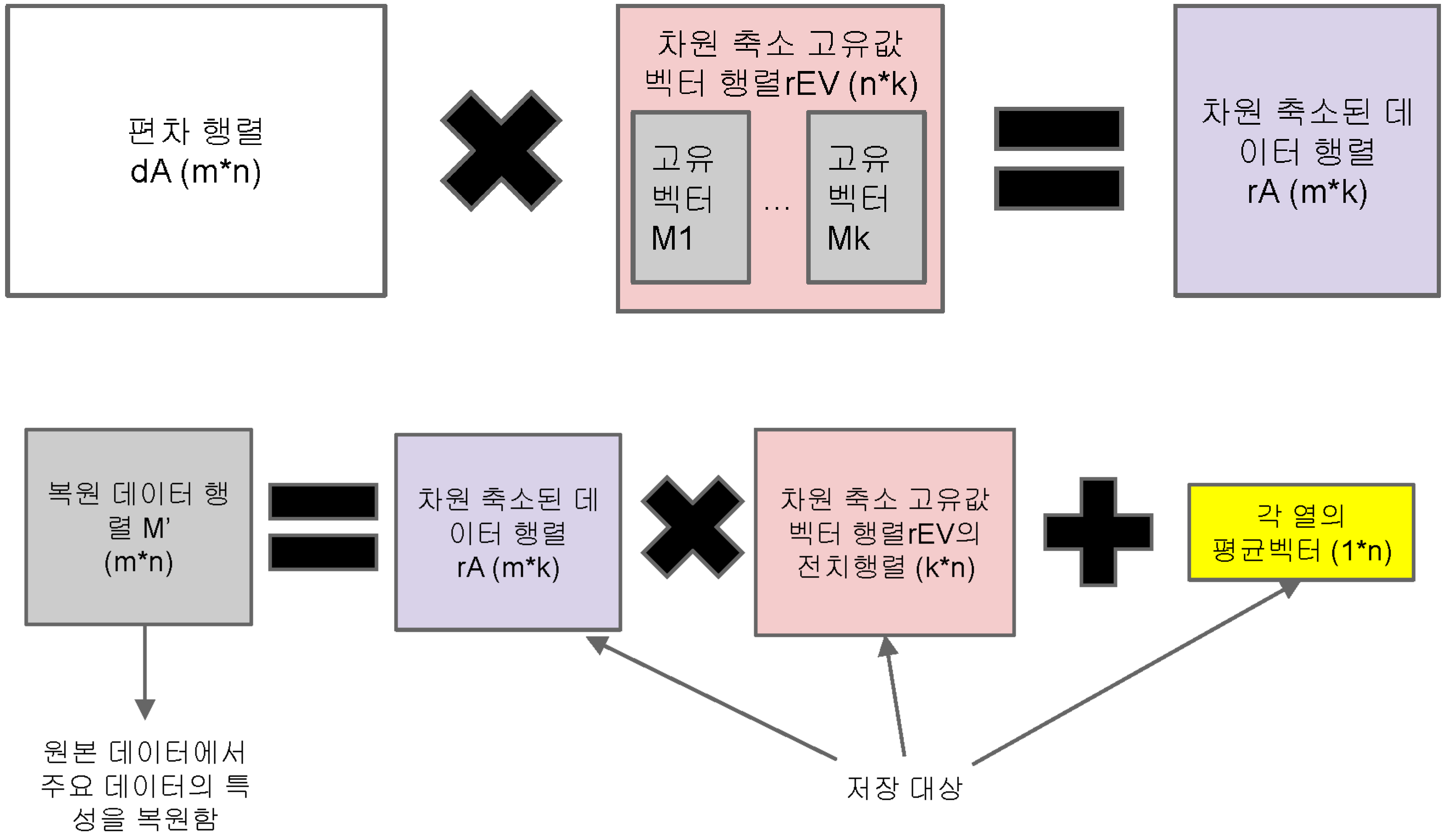
2-2. 주성분 분석(PCA)

- **PCA를 이용한 데이터 압축 및 복원**



2-2. 주성분 분석(PCA)

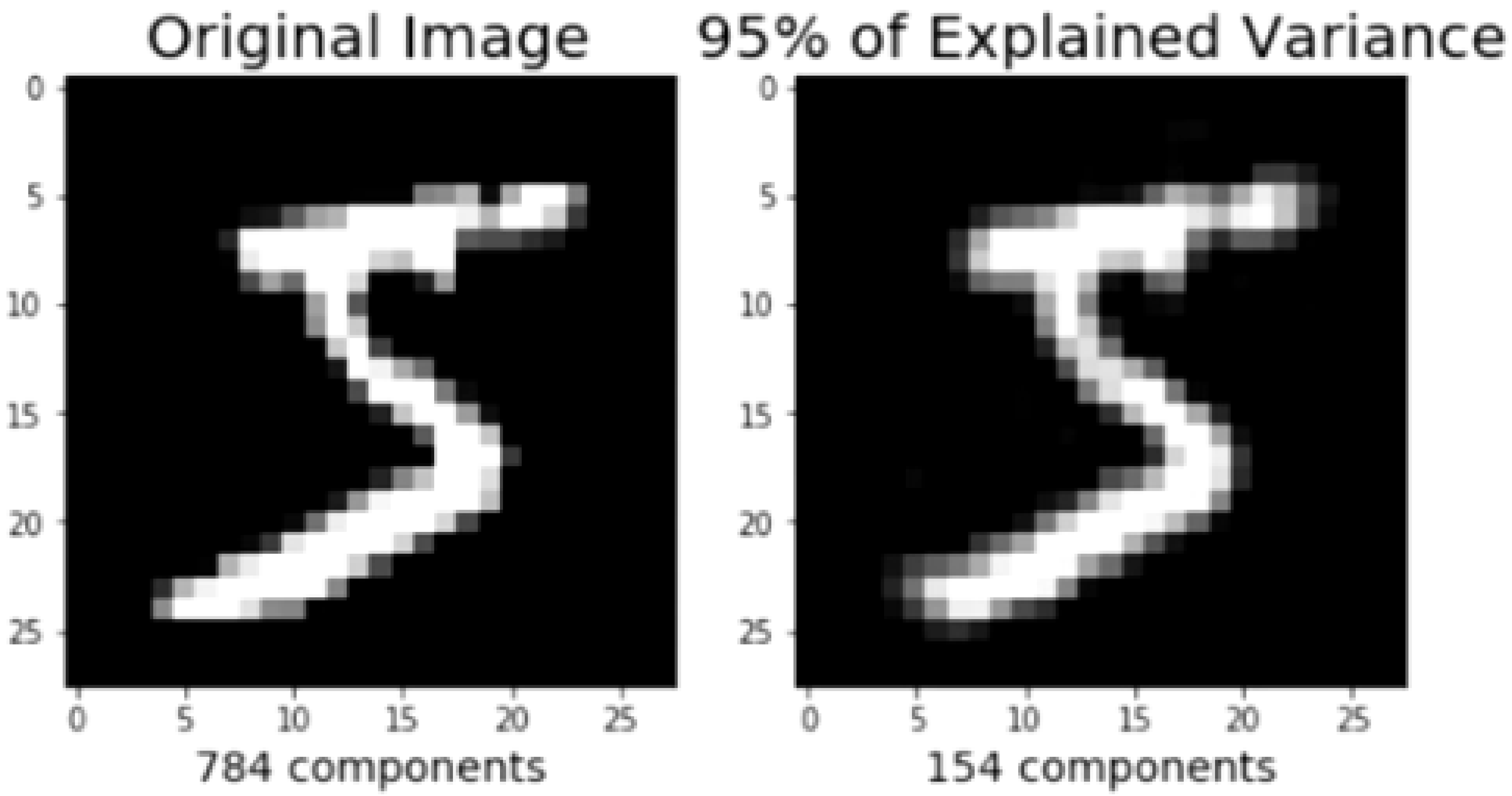
- **PCA를 이용한 데이터 압축 및 복원**



2-2. 주성분 분석(PCA)

- **PCA를 이용한 데이터 압축 및 복원**

  $28 \times 28 = 784$개의 피쳐를 PCA를 사용해 95% 분산인 154개의 피쳐로 줄인뒤 다시 복원



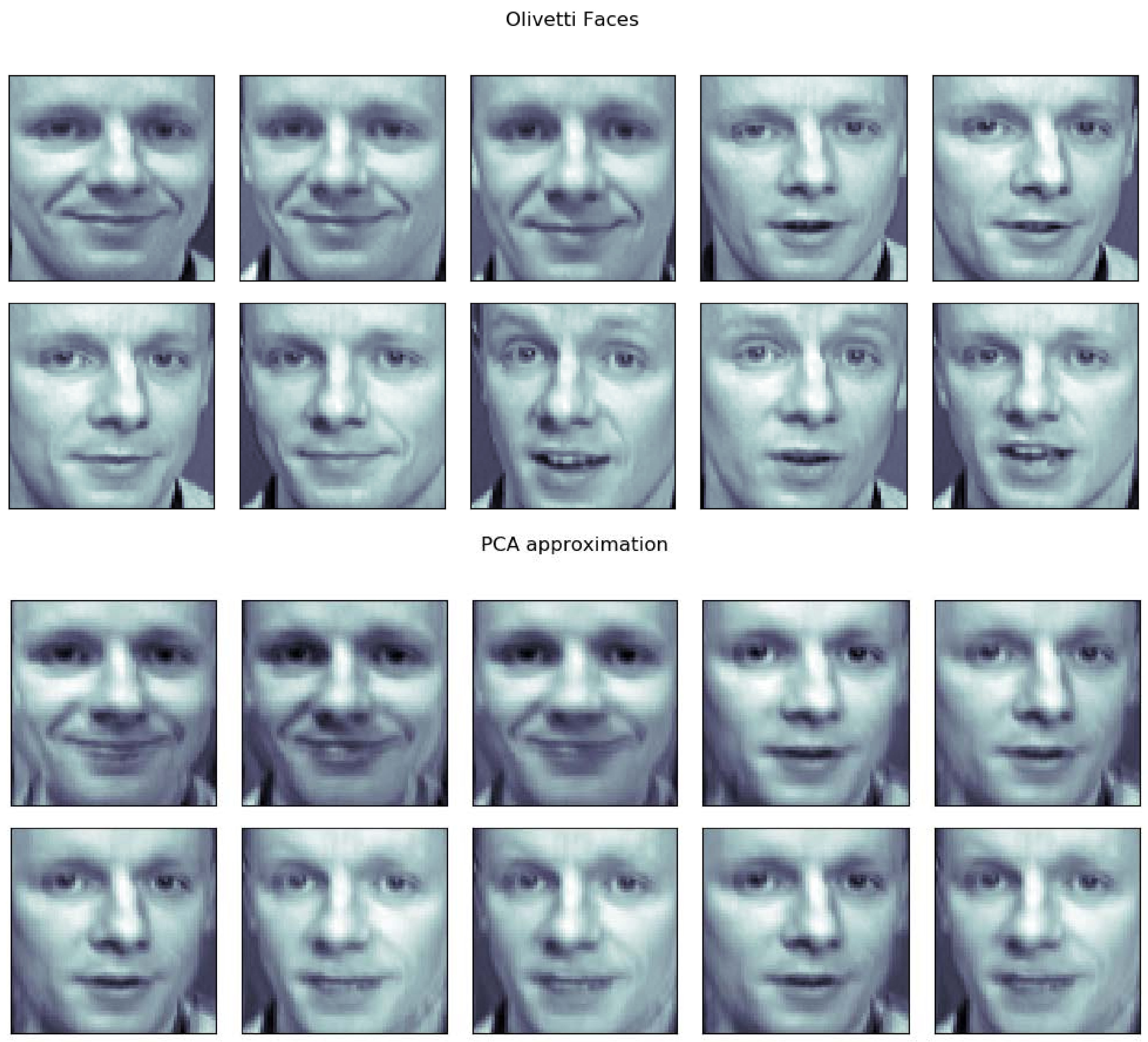
2-2. 주성분 분석(PCA)

- **PCA를 이용한 데이터 압축 및 복원**

```
pca3 = PCA(n_components=2)
X3 = faces_all.data[faces_all.target == K]
W3 = pca3.fit_transform(X3)
X32 = pca3.inverse_transform(W3)
fig = plt.figure(figsize=(10, 5))
plt.subplots_adjust(top=1, bottom=0, hspace=0, wspace=0.05)
for i in range(N):
    for j in range(M):
        k = i * M + j
        ax = fig.add_subplot(N, M, k+1)
        ax.imshow(X32[k].reshape(64, 64), cmap=plt.cm.bone)
```

2-2. 주성분 분석(PCA)

- PCA를 이용한 데이터 압축 및 복원

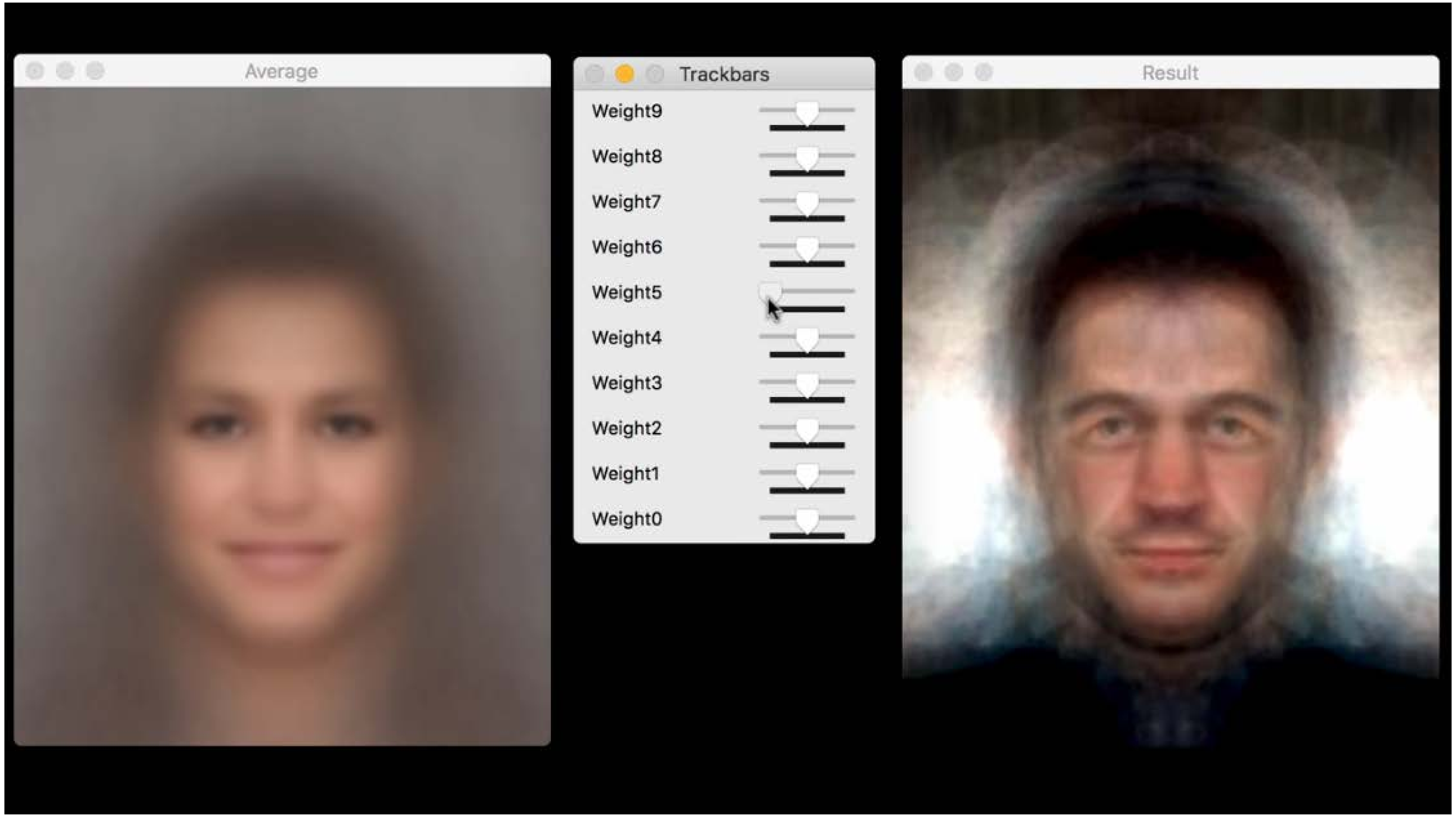


2-3. EigenFace

```
# Create data matrix for PCA.
data = createDataMatrix(images)

# Compute the eigenvectors from the stack of images created
print("Calculating PCA ", end="...")
mean, eigenVectors = cv2.PCACompute(data, mean=None, maxComponents=NUM_EIGEN_FACES)
print ("DONE")

averageFace = mean.reshape(sz)
eigenFaces = [];
for eigenVector in eigenVectors:
    eigenFace = eigenVector.reshape(sz)
    eigenFaces.append(eigenFace)
```

2-3. EigenFace

- https://www.youtube.com/watch?v=J0arU2PAMIs



3. tSNE(t-Stochastic Neighborhood Embedding)

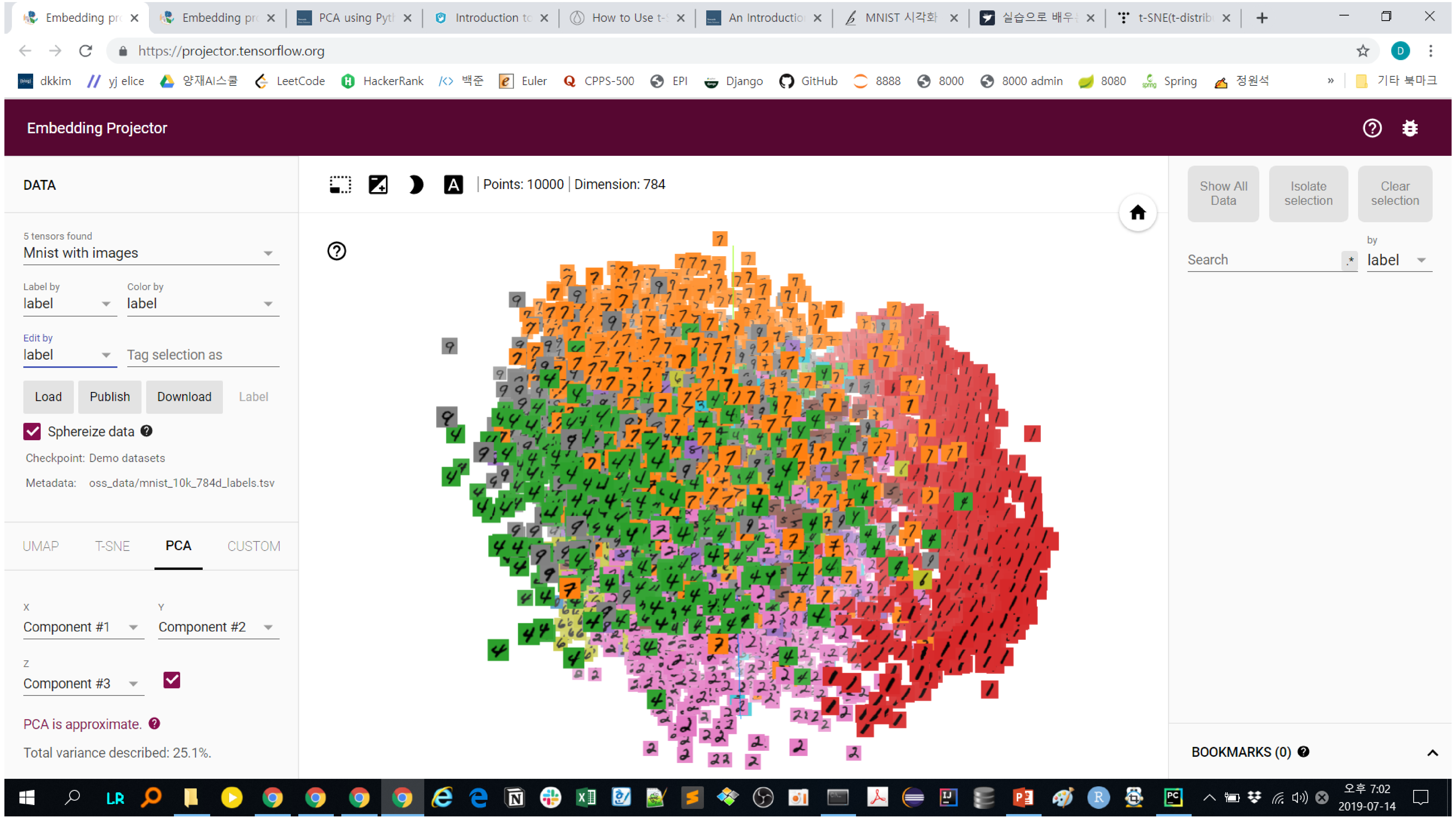
3. tSNE

- SNE: 고차원 공간에서 유클리드 거리를 데이터 포인트의 유사성을 표현하는 조건부 확률로 변환하는 방식으로 저차원 공간에 표시
  1. KL(Kullback-Leibler) divergence 사용

- 2008년 Laurens van der Maaten과 Geoffrey Hinton가 기존의 SNE의 단점을 보완하는 tSNE(t-Stochastic Neighbor Embedding) 제안
  1. 손실함수를 대칭 버전으로 변경
  2. 정규분포 대신 Student-t 분포 사용

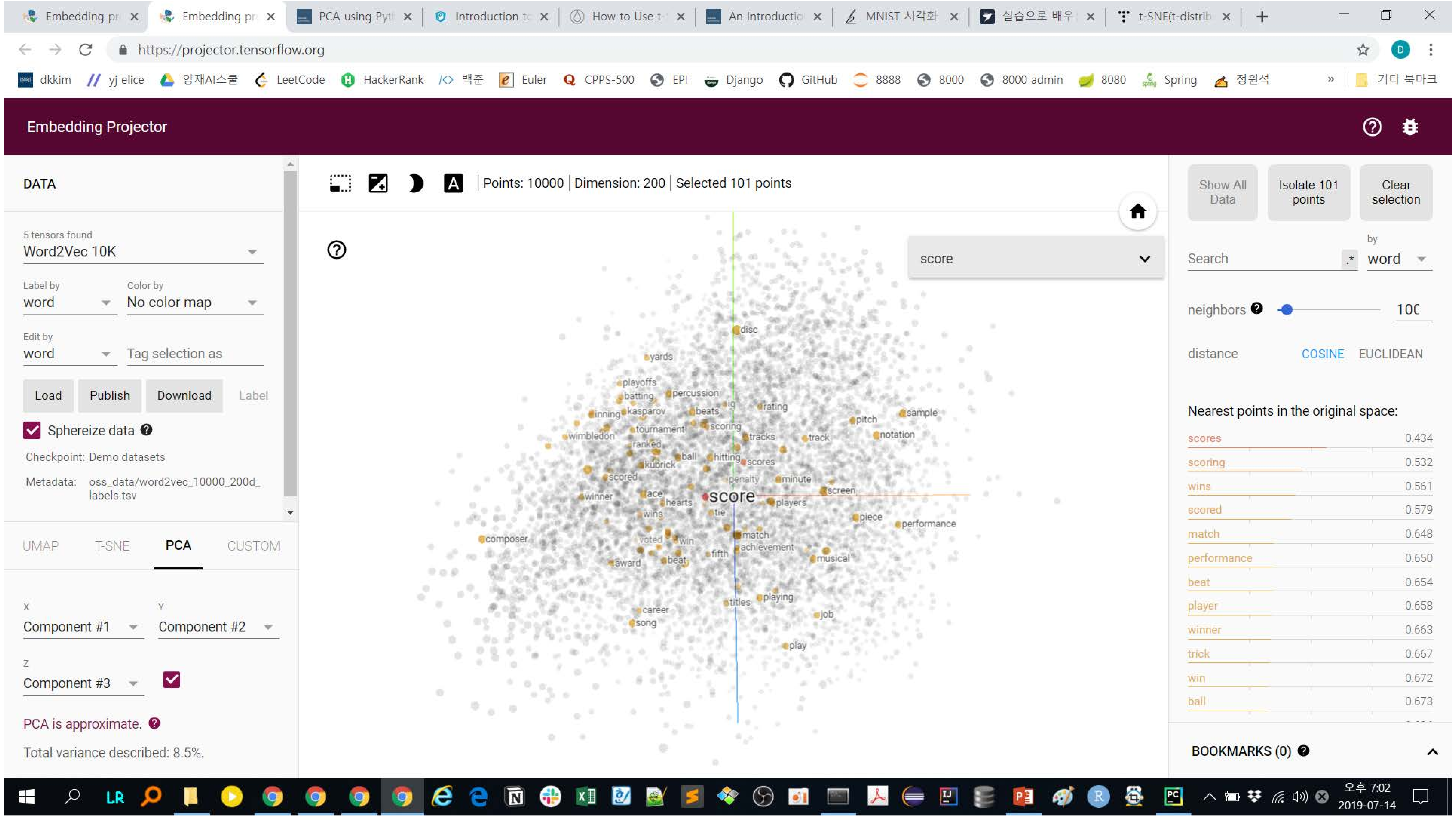
3. tSNE

- MNIST 시각화 (https://projector.tensorflow.org/)



3. tSNE

Word2Vec 시각화



참고자료

1. StatQuest 동영상
    1. https://www.youtube.com/watch?v=FgakZw6K1QQ
    2. https://www.youtube.com/watch?v=oRvgq966yZg
2. PCA
    1. https://m.blog.naver.com/PostView.nhn?blogId=sw4r&logNo=221031465518&proxyReferer=https%3A%2F%2Fwww.google.com%2F
    2. https://excelsior-cjh.tistory.com/167
3. EigenFace 동영상 및 코드
    https://www.learnopencv.com/eigenface-using-opencv-c-python/
4. PCA Scikit-learn 코드
    https://datascienceschool.net/view-notebook/f10aad8a34a4489697933f77c5d58e3a/
5. Projector(시각화 툴)
    https://projector.tensorflow.org/

/* elice */

문의 및 연락처

academy.elice.io
contact@elice.io
facebook.com/elice.io
medium.com/elice