University of Amsterdam
Informatics Institute

Machine Learning 1

Lecture 10.1 - Unsupervised Learning
Principal Component Analysis - Variance Maximization

Erik Bekkers

(Bishop 12.1.1)

Image credit: Kirillm | Getty Images

Slide credits: Thomas Kipf and Giorgio Patrini

Continuous latent space

Goal

- Dimensionality reduction: model the data in a low dim. space

- Example: take one grey-scale image of “3” and make multiple copies by translation and rotation

Figure: Synthetic “3” dataset (Bishop 12.1)

- Pixel space dimension: 100x100 pixels

- Latent space dimension: 3 = 2 (translations) + 1 rotation

- From the 3 latent variables we could generate all 100x100 pixels!

Example continued

- A more realistic dataset of images will have more degrees of freedom in the latent space, such as:
    - Scaling
    - Digits from 0-9
    - Colors
    - Different hand-writing styles
    - Etc.

  … but still much fewer than 100x100!

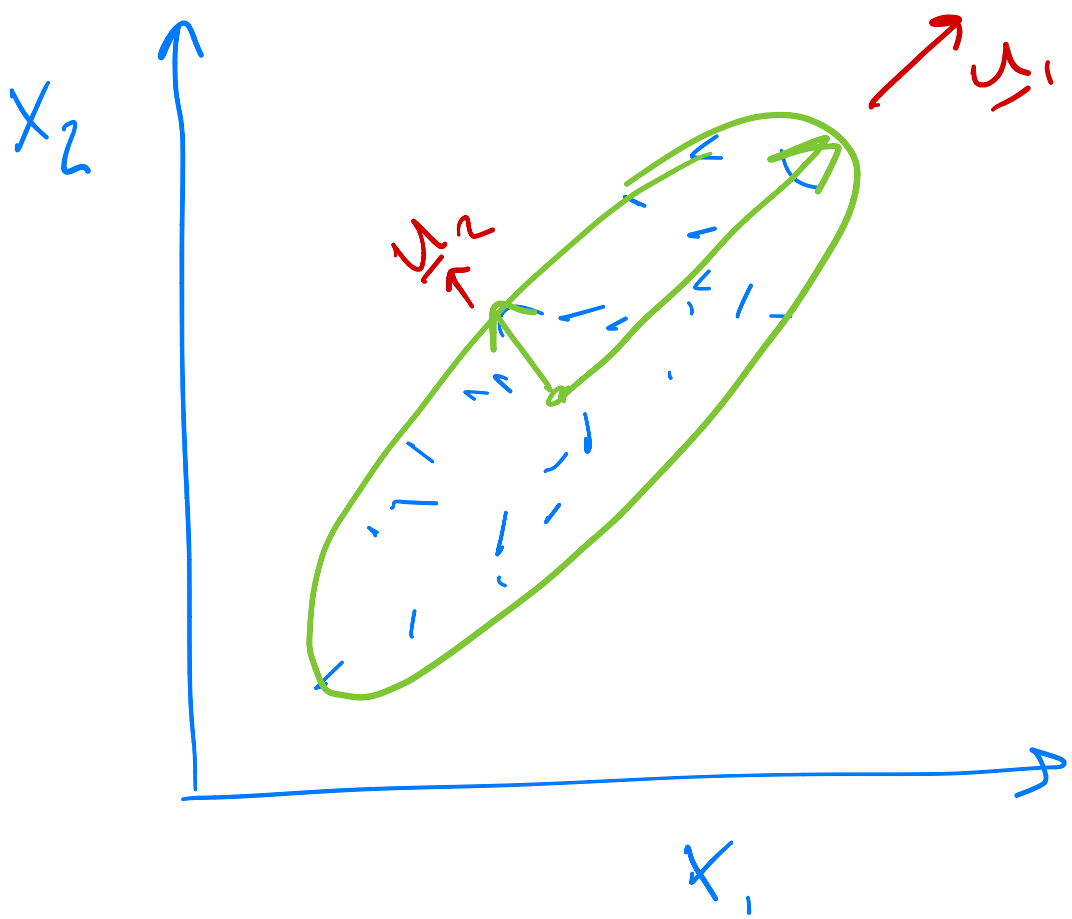
- In this example, the latent subspace is a non-linear transformation of the images

- We first study linear latent spaces with PCA and later consider generalizations to the non-linear case

Principal Component Analysis (PCA)

- Find a linear projection of the data such that the variance in the projected space is maximal

- PCA captures the axes of maximal variation in the data, called **principal components**



Principal Component Analysis (PCA)

- Data: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \ \mathbf{x}_n \in \mathbb{R}^D$
- Goal: project data into a $M < D$ dimensional space while **maximizing the variance** of the projected data
- M is given
- Mean and covariance defined by

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$$

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T$$

- $\mathbf{S}$ is symmetric and positive definite

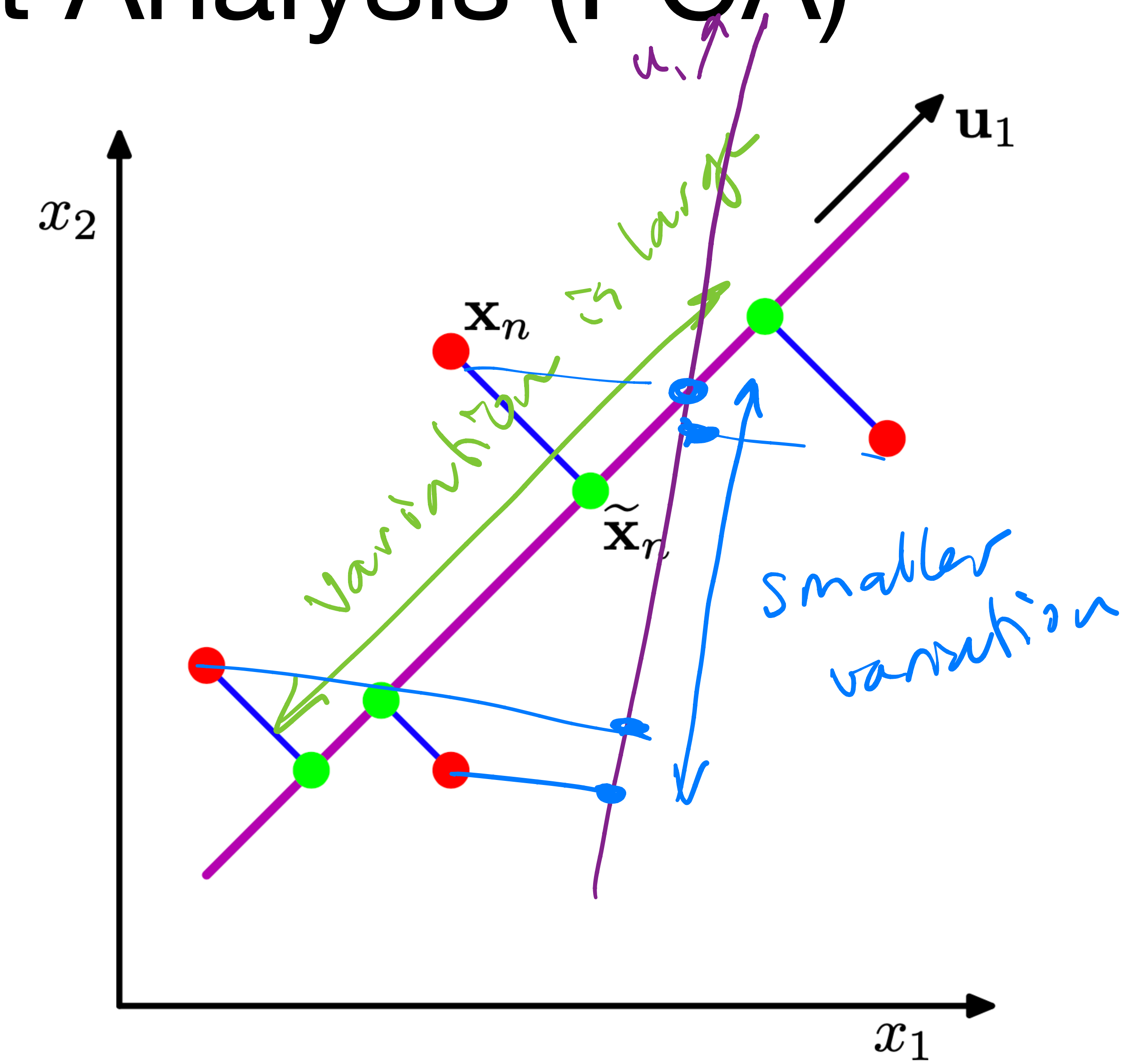


Figure: Maximizing variance of projections (Bishop 12.2)

1D Projection

$\mathbb{E}[z] = \mathbb{E}[u_1^T x]$
$= u_1^T \mathbb{E}[x]$

- Project data int the first latent dimension by a vector $\mathbf{u}_1 \in \mathbb{R}^D$
- The projection gives the scalar $\underbrace{\mathbf{u}_1^T\mathbf{x}_n}_{z_{n1} \in \mathbb{R}}$, the mean of the projection is $\mathbf{u}_1^T\overline{\mathbf{x}}$
- We only need its direction, so normalize this component: $\|\mathbf{u}_1\|^2 = \mathbf{u}_1^T\mathbf{u}_1 = 1$
- The variance of the projected data is

$$\boxed{\mathrm{var}[z_1]} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}})^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T(\mathbf{x}_n - \overline{\mathbf{x}}))^2$$

$$= \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_1^T(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T\mathbf{u}_1$$

$$= \mathbf{u}_1^T\left(\underbrace{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T}_{\mathrm{cov}[x] = S}\right)\mathbf{u}_1 = \boxed{\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1}$$

Maximizing the variance of 1 component

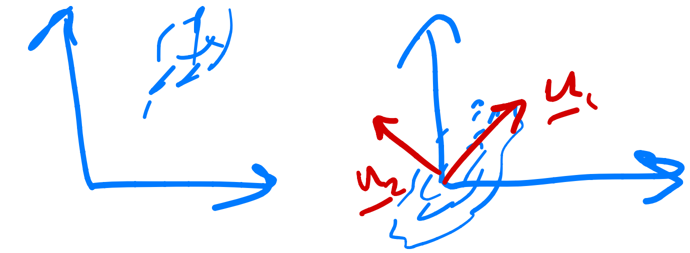
Need this constraint else objective is infinite

- Solve $\underset{\mathbf{u}_1}{\text{argmax}}\ \underbrace{\mathbf{u}_1^T \mathbf{S}\, \mathbf{u}_1}_{f(u_1)}$ subject to $\underbrace{\mathbf{u}_1^T \mathbf{u}_1}_{g(u_1) = c} = 1$
- Method of Lagrange multipliers
  - Define Lagrangian $L(\mathbf{u}_1, \lambda_1) = \overbrace{\mathbf{u}_1^T \mathbf{S}\, \mathbf{u}_1}^{f(u_1)} + \overbrace{\lambda_1(\mathbf{u}_1^T \mathbf{u}_1 - 1)}^{\lambda_1 (g(u_1) - c)}$
  - Solving for $\mathbf{u}_1$ means $\frac{\partial}{\partial \mathbf{u}_1} L(\mathbf{u}_1, \lambda_1) = \mathbf{S}\, \mathbf{u}_1 - \lambda_1 \mathbf{u}_1 = 0$
  - We need to solve eigensystem $\boxed{\mathbf{S}\, \mathbf{u}_1 = \lambda_1 \mathbf{u}_1}$
  - So $\mathbf{u}_1$ and λ_1 are respectively an eigenvector and eigenvalue of $\mathbf{S} \in \mathbb{R}^{D \times D}$!
- The $\mathbf{u}_1$ is called a **principal component**.

$$\underline{u}_1^T \lambda_1 \underline{u}_1 = \lambda_1 \underline{u}_1^T \underline{u}_1$$
$$= \lambda_1$$

- The variance of the projected data is $\mathbf{u}_1^T \mathbf{S}\, \mathbf{u}_1 = \lambda_1$
- Maximizing variance means we search for the eigenvecor with **largest eigenvalue**

PCA via maximum variance



- We repeat the procedure for M orthogonal vectors and get a projection defined by $U_M = [\mathbf{u}_1, \ldots, \mathbf{u}_M] \in \mathbf{R}^{D\times M}$

$$\begin{pmatrix} z_{n1} \\ z_{n2} \\ \vdots \end{pmatrix} = \begin{pmatrix} u_1^T x_n \\ u_2^T x_n \\ \vdots \end{pmatrix}$$

- PCA: compute $\bar{\mathbf{x}}$ and the eigen-decomposition of $\mathbf{S}$. The **projection** then is $\mathbf{z} = \mathbf{U}_M^T(\mathbf{x} - \bar{\mathbf{x}})$

- Those are M eigenvectors of $\mathbf{S}$, **the principal components**. The eigenvalues are $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$

- The matrix $\mathbf{S}$ is positive semi-definite, thus $\forall_j : \lambda_j \geq 0$

- The (total) variance of the projected data is $\mathrm{Tr}[\mathrm{Cov}[\mathbf{z}]] = \sum_{j=1}^{M} \lambda_j$

Reminder: eigen-decomposition

$u_i^T u_j = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$

- When the matrix is **symmetric positive semi-definite**:

$$\mathbf{S} = \mathbf{U}\,\Lambda\,\mathbf{U}^T \quad \text{with} \quad \Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_D\}$$

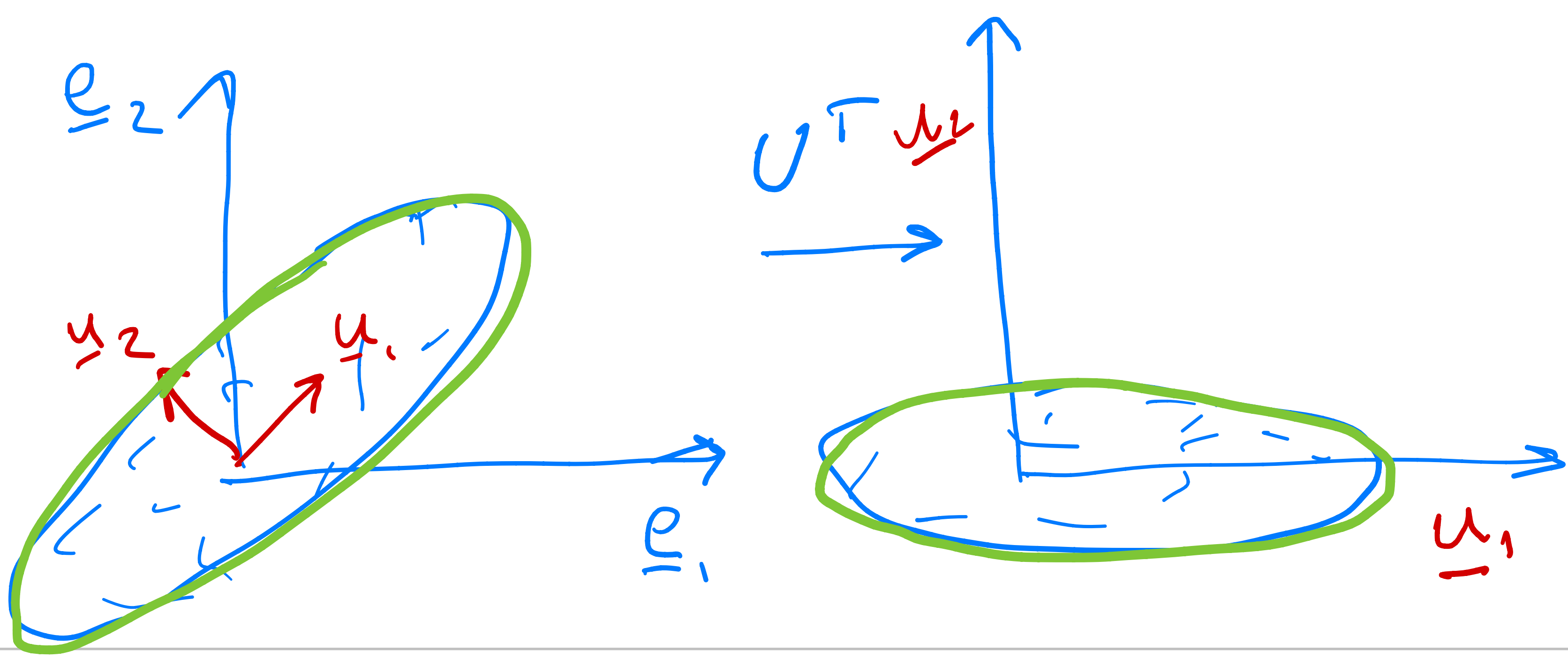
- The eigenvectors are **orthonormal** and are stored in $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_D]$

Change of basis → rotation

- All eigenvalues are **non-negative** and are the elements of the diagonal matrix Λ

$= \mathrm{Tr}(\Lambda U^T U) = \mathrm{Tr}(\Lambda I)$

- Total variance given by $\mathrm{Tr}(\mathbf{S}) = \mathrm{Tr}(\mathbf{U}\,\boldsymbol{\Lambda}\,\mathbf{U}^T) = \mathrm{Tr}(\boldsymbol{\Lambda}) = \sum_{i=1}^{D} \lambda_i$



Getting the eigenvectors in practice

- Full eigenvalue decomposition is expensive: $O(D^3)$
- Only need up to the M^{th} component: $O(MD^2)$
- In python:

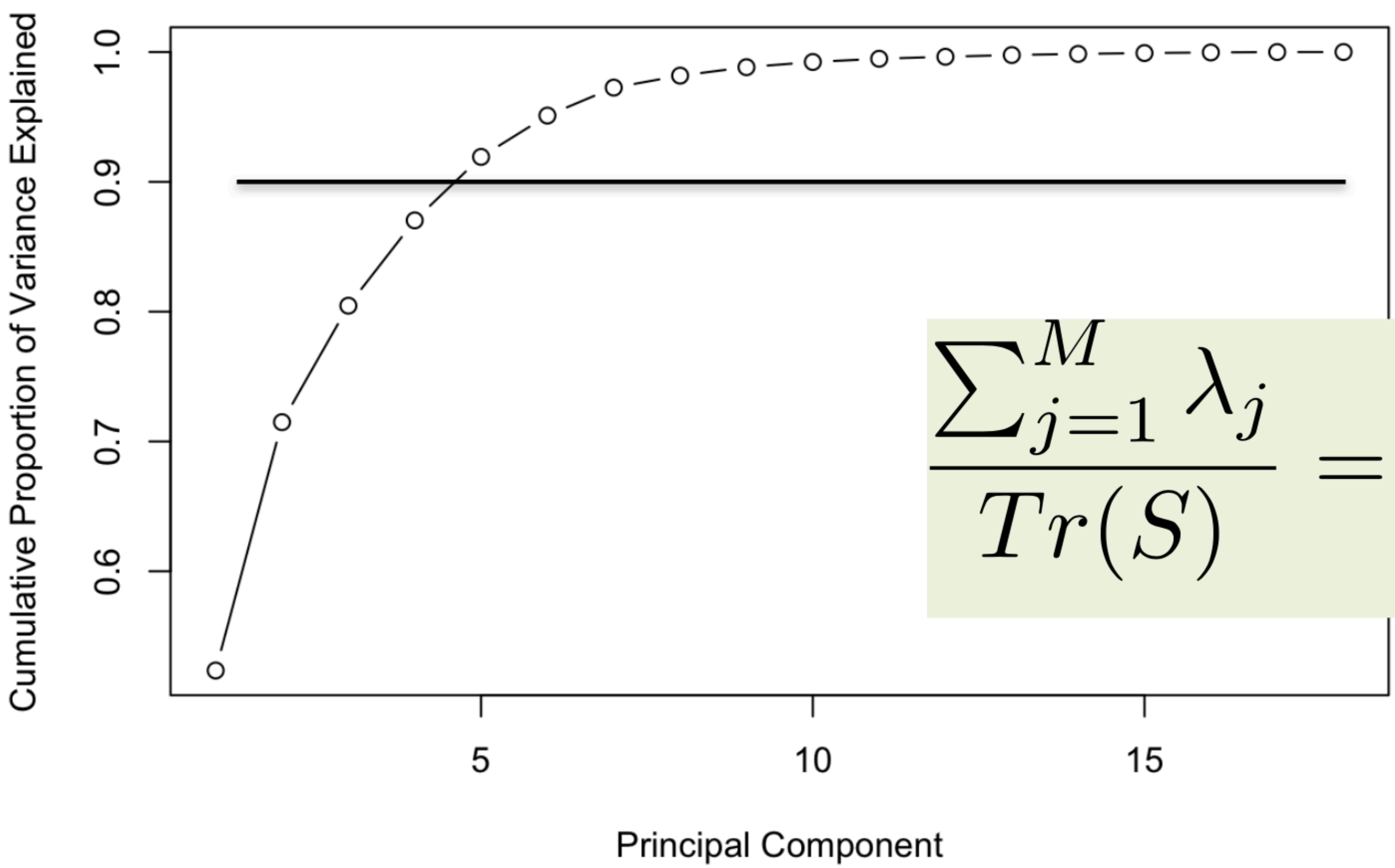
```
M = 10
S = np.cov(X)
Um, Lm, Vm = scipy.sparse.linalg.svds(S, k=M)
```

For symmetric positive definite matrices such as $\mathbf{S}$*, the SVD decomposition is equivalent to the eigen-decomposition*

How to choose M?

- We can measure the discarded variance

- For example to preserve 90% of the variance, pick M such that



The proportion of explained variance

$$\frac{\sum_{j=1}^{M} \lambda_j}{Tr(S)} = \frac{\sum_{j=1}^{M} \lambda_j}{\sum_{j=1}^{D} \lambda_i} > 0.9$$

Applications: dimensionality reduction

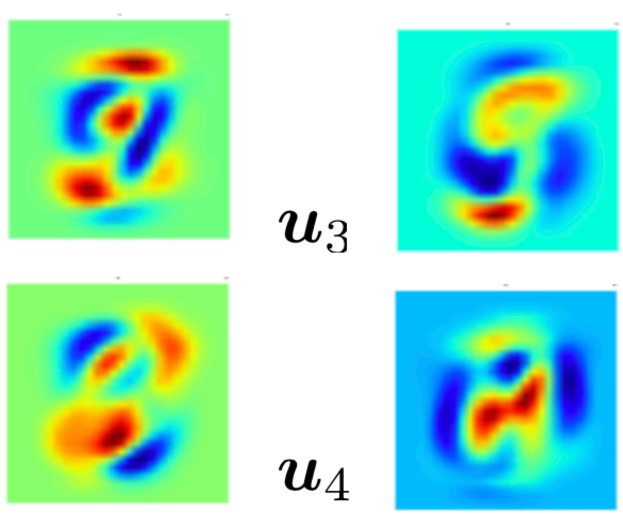
- When data is defined in high dimension (large D) we want to project down to lower dimension because:
  - Reduce time and storage space required
  - For classification/regression: our model **will have less parameters**, thus we need less data points for learning

- Other methods (not covered): **feature selection**. PCA is known as a **feature extraction** method instead.

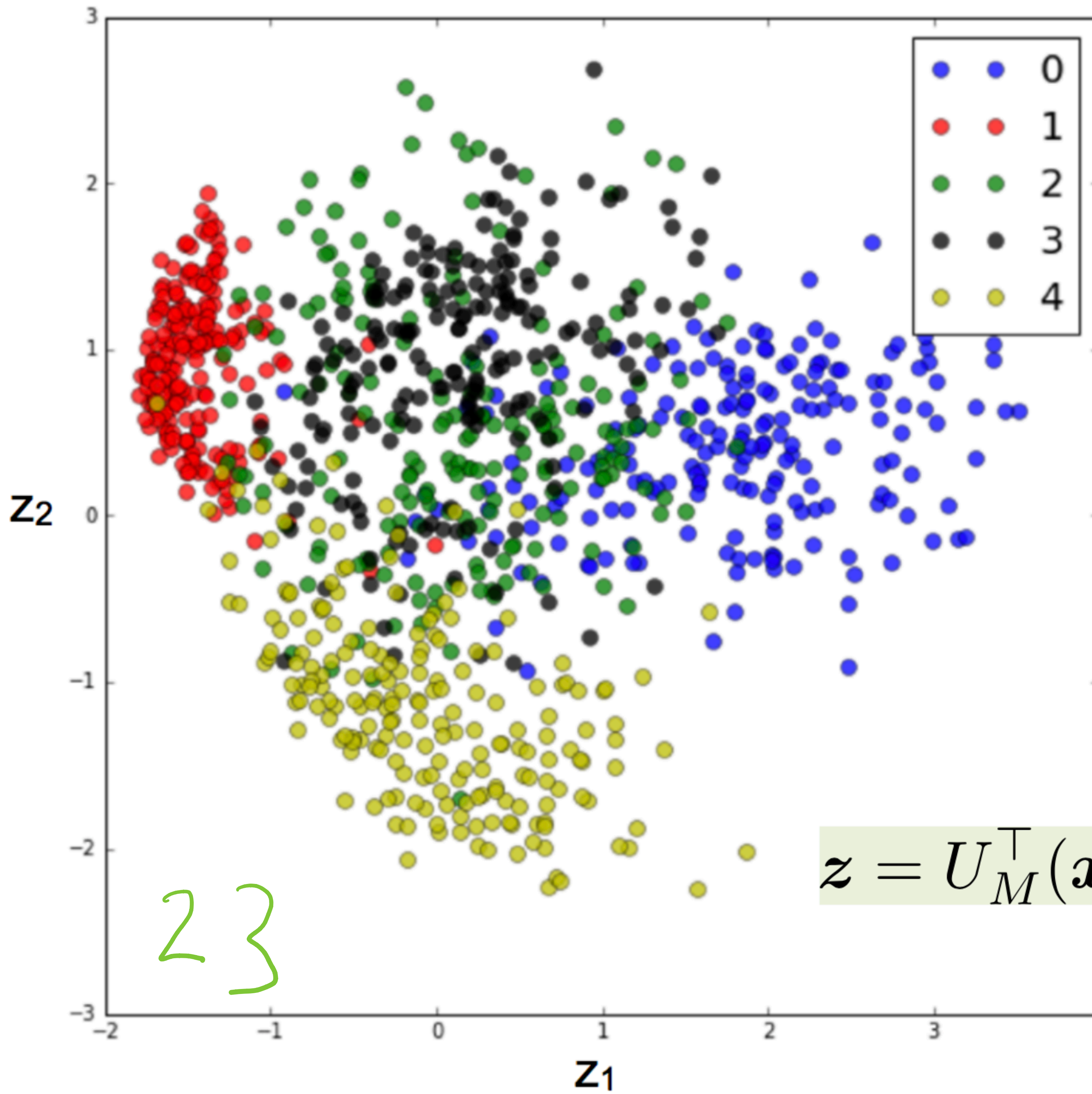
Applications: 2D Visualization (MNIST)

MNIST: 24 x 24 pixels

u_1 u_3 u_2 u_4



Eigenvectors



$$z = U_M^\top (x - \overline{x})$$

Feature Decorrelation

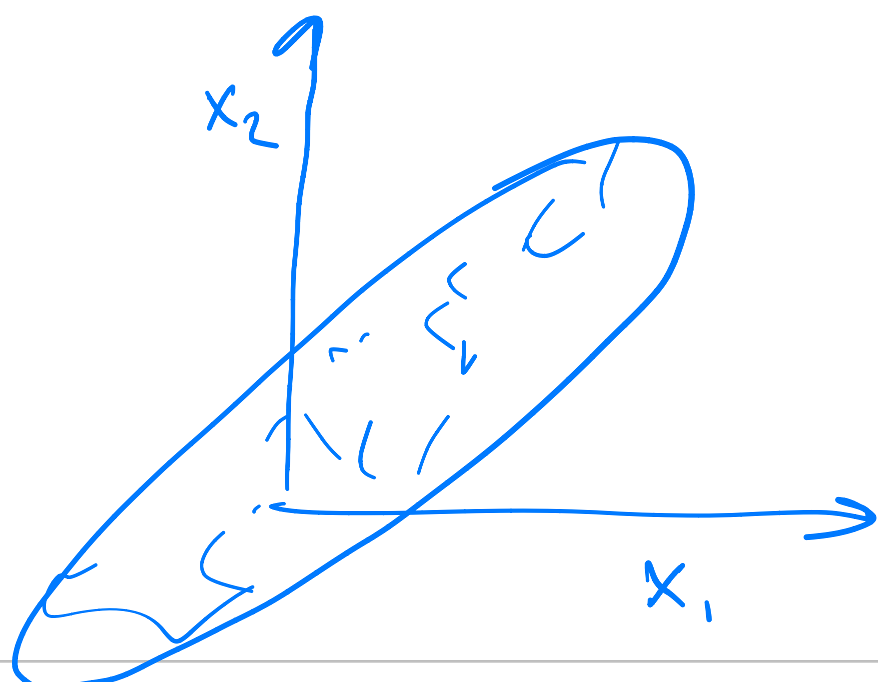
- Good side effect of PCA: features have **no correlation** in the projected space.

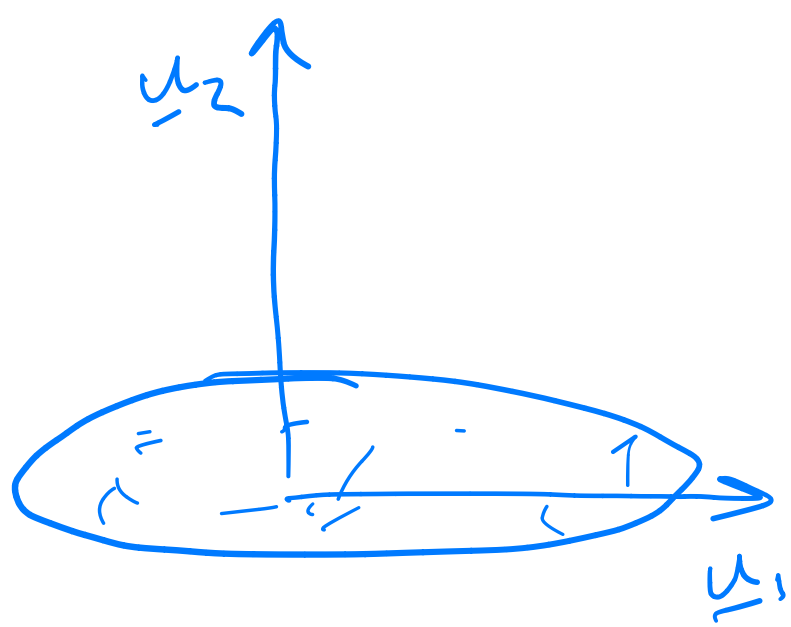
- The covariance matrix of the projected data is **diagonal**

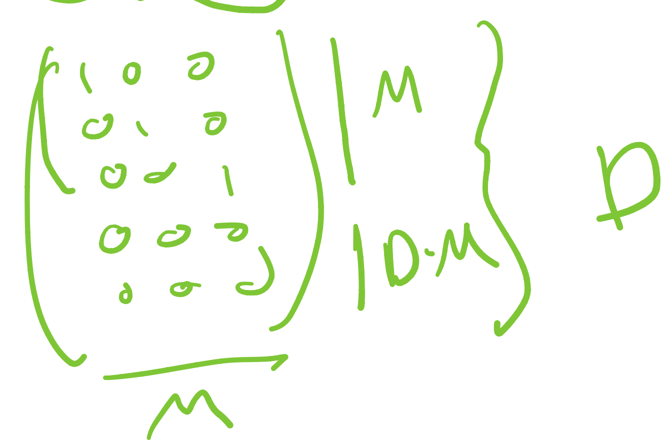
$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{z}_n\mathbf{z}_n^T = \frac{1}{N}\sum_{n=1}^{N}\mathbf{U}_M^T(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{U}_M$$

$u_i^T S u_i = \lambda_i$

$$= \mathbf{U}_M^T\,\mathbf{S}\,\mathbf{U}_M = \mathbf{U}_M^T\,\mathbf{U}\,\mathbf{\Lambda}\,\underbrace{\mathbf{U}^T\,\mathbf{U}_M} = \Lambda_M$$







Applications: whitening (or sphering)

- Before applying learning algorithms we usually do some pre-processing:
  - e.g. **standardization**: subtract the mean and divide by the standard deviation

- With PCA we can **whiten** the data, one step more:
  - **Centre** and **de-correlate** the features:

$$\mathbf{z} = \mathbf{U}_M^T(\mathbf{x} - \bar{\mathbf{x}})$$

  - Cast features to **unit standard deviation** by rescaling:

$$\mathbf{z} = \Lambda_M^{-1/2}\mathbf{U}_M^T(\mathbf{x} - \bar{\mathbf{x}})$$