Data Preprocessing and Analysis

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professor: Christian Gagné

Week 12

UNIVERSITÉ LAVAL

12.1 Data preprocessing

Importance of preprocessing

- Learning algorithms are sensitive to input values
  - Scales of variables must be comparable
    - Larger scale variables are dominant in measures of similarity (e.g. Gaussian kernel) and distance (e.g. Euclidean, Manhattan)
    - High input values cause saturation of sigmoid neurons
  - Variables may sometimes be missing
    - Defective sensor, omissions during data collection, measurements added along the way
  - High dimensionality
    - Sensitivity of algorithms to dimensionality
    - Redundancy in measurements
- Data preprocessing is essential in practice
  - Rarely have access to well formatted and complete data, ready to be used
  - Important to understand the nature of the data in order to process it properly

Scaling

- Scale adjustment of variables
  - Common approach: bring the range of possible values back into $[0, 1]$
  - Make scaling on each variable independently

$$x'_i = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}, \quad i = 1, \ldots, D$$

where:

$$x_i^{max} = \max_{t=1,\ldots,N} x_i^t, \quad i = 1, \ldots, D$$

$$x_i^{min} = \min_{t=1,\ldots,N} x_i^t, \quad i = 1, \ldots, D$$

- Scaling values calculated on a given dataset
  - New data could have value of variable X_i outside the domain $[x_i^{min}, x_i^{max}]$
- Simple approach that often does a reasonable job

Standardization

- Standardization: bring the distribution of each variable back to a reduced normal centered distribution, $x_i' \sim \mathcal{N}(0,1)$
  - Center the mean at zero and adjust for a unit standard deviation

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}, \quad i = 1, \ldots, D$$

  - Less sensitive to outliers than a scaling
- Independent variables treatment
  - Does not remove the covariance between the variables, $\mathbf{\Sigma} \neq \mathbf{I}$
  - Whitening transformation (presented later today) allows to obtain data according to a unit normal distribution, $\mathbf{x}' \sim \mathcal{N}_D(0,\mathbf{I})$

Imputation

- What to do if variable values are missing?
  - Remove data with missing values
    - Loss of data for learning
    - Possible bias in removed data
  - Mark missing variables for the learning algorithm
    - Some learning algorithms can handle missing variables
  - Assign a default value to the missing variables (typically zero)
  - Randomly select from the other data and assign its value to the missing variable
  - Assign mean value of the variable, $x_i' = \bar{x}_i$
    - Reduces the measured variance of the variable in the dataset

Regression for imputation

- Replacing missing variables can distort the data
  - How to assign a plausible value to missing values?
- Use supervised learning to fill in missing values
  - For each variable, learn regression model to impute missing values

$$x_i' = \mathrm{f}([x_1 \ \ldots \ x_{i-1} \, x_{i+1}, \ldots, x_D]^\top | \theta_i)$$

  - The targets r^t used to learn parameterization θ_i correspond to the values x_i for the data where they are not missing
- Values more representative of the data, but can still reduce the variance as regression will capture the most likely values

12.2 Feature selection

Dimensionality reduction

- Dimensionality reduction
  - Go from a space with D dimensions to a space with K dimensions, where $K < D$

$$X_1, \dots, X_D \mapsto X'_1, \dots, X'_K$$

- Possible approaches
  - Feature selection: choose K variables among the possible D variables

$$X_1, \dots, X_D \;\mapsto\; X_{v_1}, \dots, X_{v_K}$$
$$v_i \in \{1, \dots, D\} \mid v_i \neq v_j, \forall j \leq i$$

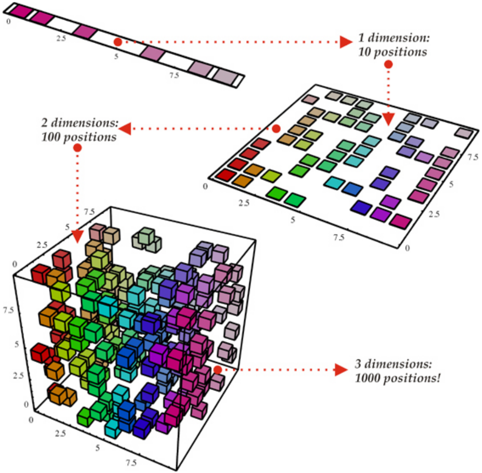
  - Feature extraction: generate K variables as transformations of the original D variables

$$X_1, \dots, X_D \;\mapsto\; \mathrm{f}_1(X_1, \dots, X_D), \dots, \mathrm{f}_K(X_1, \dots, X_D)$$

Reasons for reducing dimensionality

- Curse of dimensionality
    - Adding a dimension exponentially increases mathematical space
    - 100 points equidistant by 0.01 in one dimension ⇒ 10^{20} needed in 10 dimensions to keep the same density
    - High dimensionality: high computational and memory complexity
- Saving measurement costs
- The simpler a model is, the less variance there is
- Easier to explain with fewer variables: knowledge extraction
- Viewing data: analyzing results

Curse of dimensionality



Source: Y. Bengio, `http://www.iro.umontreal.ca/~bengioy/yoshua_en/research_files/CurseDimensionality.jpg`, accessed October 2, 2016.

Feature selection

- Objective: find a subset of K variables among $\{X_1, \ldots, X_D\}$, while preserving the performance
- Number of possible subsets: $\begin{pmatrix} D \\ K \end{pmatrix}$

$$\begin{pmatrix} 10 \\ 5 \end{pmatrix} = 252, \begin{pmatrix} 50 \\ 10 \end{pmatrix} \approx 10^{10}, \begin{pmatrix} 100 \\ 20 \end{pmatrix} \approx 10^{20}$$

- Heuristics: *the art of inventing, of making discoveries*
    - Algorithm that quickly provides (in polynomial time) a feasible, not necessarily optimal solution
    - As opposed to an exact algorithm that finds an optimal solution

Evaluations of subsets of features

- Filter approach
  - Calculate performance without a new training, with indirect measurement (*proxy*)
  - Not very demanding in calculation, but mixed results
- Wrapper approach
  - For each candidate set of features, train a new classifier
  - Empirical error assessment (training, validation, cross-validation, etc.)
  - Much more expensive in calculation time
- Embedded approach: feature selection integrated in model learning

Univariate selection

- Select according to performance measurement of individual features
  - Basic approach: select features for which variance exceeds a threshold
    - Assumes that the variance accurately describes the usefulness of each feature for classification
    - Good for filtering features of very low or zero variance (avoid singular covariance matrices)
- Selection according to other criteria
  - Correlation between features (keep set of decorrelated variables)
  - Mutual information between the feature and the target value

$$I(i) = \int_{X_i} \int_r p(X_i, r) \log \frac{p(X_i, r)}{p(X_i)\, p(r)} \, dr \, dX_i$$

  - Effect on empirical error, with imputation of unselected variables

Forward sequential selection

- Gradually build the feature set, adding the most promising variable
    1. Starting with an empty feature set
    2. Add the feature that improves the most (according to a certain criterion) the set of features
    3. Repeat step 2 as long as the stop criterion is not reached
- Greedy algorithm: making iterative local decisions
    - Does not account for complex relationships between variables
    - Example:
        - Variables X_a, X_b and X_c taken individually or in pairs $\Rightarrow$ low gain
        - The three variables taken together $\Rightarrow$ high gain
- Algorithmic complexity $O(KD)$

Forward Sequential Selection Algorithm

1. Initialize the algorithm:
   - Create the set of selected features:
   $$F^0 = \emptyset$$
   - Create the set of unselected features:
   $$G^0 = \{X_1, \ldots, X_D\}$$
2. For $t = 1, \ldots, D$, as long as the stop criterion is not reached:
   2.1 Determine the feature that reduces the most the error:
   $$X_j = \operatorname*{argmin}_{X_i \in G^{t-1}} E(F^{t-1} + \{X_i\})$$
   2.2 Select this feature by adding it to F and removing it from G:
   $$F^t = F^{t-1} + \{X_j\}, \quad G^t = G^{t-1} \setminus \{X_j\}$$
3. Return the final subset F of selected features

Stopping criteria

- Possible stopping criteria
    - Stop when K features are selected
    - Stop when all features are selected
        - Return the set of features that lead to minimal empirical error
    - Stop when error reduction is below a threshold

$$E(F^t) - E(F^{t+1}) < \epsilon$$

Backward sequential selection

- Reverse approach: start with all variables and iteratively remove those that contribute the least.

1. Create the set of selected features:

$$F^D = \{X_1, \ldots, X_D\}$$

2. For $t = D - 1, D - 2, \ldots, 1$, as long as the stop criterion is not reached:
    1. Determine the least contributing feature:

    $$X_j = \underset{X_i \in F^{t+1}}{\operatorname{argmin}} E(F^{t+1} \backslash \{X_i\})$$

    2. Remove this feature from F:

    $$F^t = F^{t+1} \backslash \{X_j\}$$

3. Return the final subset F of selected features

Other approaches for feature selection

- Add-*l*-remove-*r*
    - Hybrid between forward and backward sequential approaches, avoids some local minima
- *Branch-and-bound*
    - Organize features into trees, according to their similarities
    - Reduction by cutting into the tree to eliminate similar features
- Multi-objective evolutionary algorithm
    - Population-based stochastic optimization inspired by natural evolution
    - Global search: one individual = a subset of features
    - Optimization according to two objectives simultaneously: reducing the error and reducing the number of selected features

12.3 Principal component analysis

Feature extraction

- Feature selection
  - Advantage: allows to remove completely from the measurements
  - Drawback: sometimes several variables are poor in information when taken individually, but rich in information when taken collectively
    - Example: object recognition from image pixels
- Feature extraction
  - Projection from a space with D dimensions to a space with K dimensions
  - Advantage: allows to compress the information to a space of reduced dimensionality
  - Drawback: requires taking all original D measurements

Reminder: linear transformations

- Translation in a space

$$\mathbf{y} = \mathbf{x} + \mathbf{u}$$

- Linear transformation according to matrix $\mathbf{A}$ of size $K \times D$

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

  - Rotation in a space (example in 2D)

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- General formulation

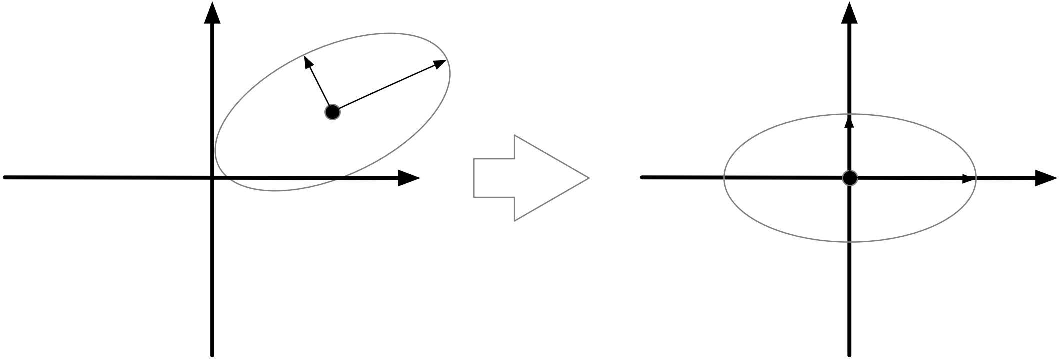
$$\mathbf{y} = \mathbf{A}(\mathbf{x} + \mathbf{u})$$

Principal component analysis

- Principal component analysis (PCA)
  - Linear projection in a space with K dimensions, with minimal loss of information
  - Variance = information
  - Consists in extracting vectors in the directions of maximum variances
  - Unsupervised: uses only measurements, not class labels
- 1st principal component: direction of maximum variance
- 2nd principal component: direction of maximum variance orthogonal to the first component
- Linear transformation, centered on the mean vector

$$\mathbf{z} = \mathbf{W}^\top (\mathbf{x} - \boldsymbol{\mu})$$

Illustration of PCA



12.4 PCA derivation

Lagrange multipliers

- Method for solving optimization problems under constraints
  - Example: maximize $f(\mathbf{x})$ under constraints that $g(\mathbf{x}) = 0$
  - There is a parameter $\lambda \neq 0$ so that

$$\nabla f + \lambda \nabla g = 0$$

  - Corresponding equation with Lagrange multiplier

$$L(\mathbf{x},\lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x})$$

- Maximum obtained by finding $\nabla L(\mathbf{x},\lambda) = 0$
  - If we are only interested in $\mathbf{x}$, we can eliminate λ without having to evaluate it

Example with the Lagrange multiplier

- Maximize $f(x_1,x_2) = 1 - x_1^2 - x_2^2$ subject to constraint $g(x_1,x_2) = x_1 + x_2 - 1 = 0$
- Formulation with Lagrange multiplier

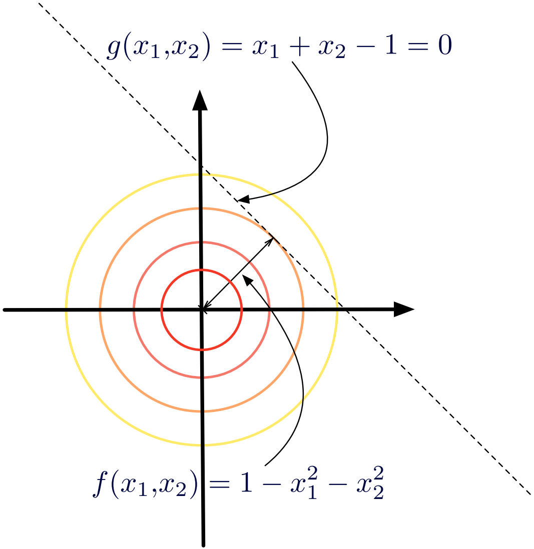
$$L(x_1,x_2,\lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$$

- Resolution of $\nabla L(x_1,x_2,\lambda) = 0$

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= -2x_1 + \lambda = 0 \\ \frac{\partial L}{\partial x_2} &= -2x_2 + \lambda = 0 \\ \frac{\partial L}{\partial \lambda} &= x_1 + x_2 - 1 = 0 \end{aligned}$$

- Solution to the system of equations: $x_1 = 0.5$, $x_2 = 0.5$ and $\lambda = 1$

Example with the Lagrange multiplier


$g(x_1,x_2) = x_1 + x_2 - 1 = 0$
$f(x_1,x_2) = 1 - x_1^2 - x_2^2$

PCA derivation

- First principal component $\mathbf{w}_1$: direction of the main variance

$$z_1 = \mathbf{w}_1^\top \mathbf{x}$$

- Only the direction is important, $\|\mathbf{w}_1\| = 1$
- If $\mathrm{Cov}(\mathbf{x}) = \mathbf{\Sigma}$ then $\mathrm{Var}(z_1) = \mathbf{w}_1^\top \mathbf{\Sigma} \mathbf{w}_1$

$$\begin{aligned}
\mathbb{E}[\mathbf{w}^\top \mathbf{x}] &= \mathbf{w}^\top \mathbb{E}[\mathbf{x}] = \mathbf{w}^\top \boldsymbol{\mu} \\
\mathrm{Var}(\mathbf{w}^\top \mathbf{x}) &= \mathbb{E}\left[(\mathbf{w}^\top \mathbf{x} - \mathbf{w}^\top \boldsymbol{\mu})^2\right] \\
&= \mathbb{E}\left[(\mathbf{w}^\top \mathbf{x} - \mathbf{w}^\top \boldsymbol{\mu})(\mathbf{w}^\top \mathbf{x} - \mathbf{w}^\top \boldsymbol{\mu})^\top\right] \\
&= \mathbb{E}\left[\mathbf{w}^\top (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{w}\right] \\
&= \mathbf{w}^\top \mathbb{E}\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top\right] \mathbf{w} \\
&= \mathbf{w}^\top \mathbf{\Sigma} \mathbf{w}
\end{aligned}$$

First principal component

- We look for the vector $\mathbf{w}_1$ which maximizes $\mathrm{Var}(z_1)$, with constraint $\mathbf{w}_1^\top \mathbf{w}_1 = 1$
- Resolution by Lagrange method

$$\begin{aligned} L(\mathbf{w}_1,\alpha) &= \mathbf{w}_1^\top \boldsymbol{\Sigma} \mathbf{w}_1 - \alpha \left(\mathbf{w}_1^\top \mathbf{w}_1 - 1 \right) \\ \frac{\partial L(\mathbf{w}_1,\alpha)}{\partial \mathbf{w}_1} &= 2\boldsymbol{\Sigma}\mathbf{w}_1 - 2\alpha\mathbf{w}_1 = 0 \\ \boldsymbol{\Sigma}\mathbf{w}_1 &= \alpha\mathbf{w}_1 \end{aligned}$$

- By definition, $\boldsymbol{\Sigma}\mathbf{w}_1 = \alpha\mathbf{w}_1$ is true when $\mathbf{w}_1$ is an eigenvector of $\boldsymbol{\Sigma}$ and that α is the associated eigenvalue
- We choose the eigenvector with the largest eigenvalue, $\alpha = \lambda_1$, given that:

$$\mathrm{Var}(\mathbf{w}_1^\top \mathbf{x}) = \mathbf{w}_1^\top \boldsymbol{\Sigma} \mathbf{w}_1 = \alpha \mathbf{w}_1^\top \mathbf{w}_1 = \alpha$$

Second principal component

- Vector $\mathbf{w}_2$ maximizes $\mathrm{Var}(z_2)$
  - Constraint 1: $\mathbf{w}_2$ is unitary, $\mathbf{w}_2^\top \mathbf{w}_2 = 1$
  - Constraint 2: $\mathbf{w}_2$ is orthogonal to $\mathbf{w}_1$, $\mathbf{w}_2^\top \mathbf{w}_1 = 0$
- Resolution by Lagrange method

$$\begin{aligned}
L(\mathbf{w}_1,\mathbf{w}_2,\alpha,\beta) &= \mathbf{w}_2^\top \mathbf{\Sigma} \mathbf{w}_2 - \alpha \left(\mathbf{w}_2^\top \mathbf{w}_2 - 1 \right) - \beta(\mathbf{w}_2^\top \mathbf{w}_1 - 0) \\
\frac{\partial L(\mathbf{w}_1,\mathbf{w}_2,\alpha,\beta)}{\partial \mathbf{w}_2} &= 2\mathbf{\Sigma}\mathbf{w}_2 - 2\alpha\mathbf{w}_2 - \beta\mathbf{w}_1 = 0 \\
\mathbf{w}_1^\top \frac{\partial L(\mathbf{w}_1,\mathbf{w}_2,\alpha,\beta)}{\partial \mathbf{w}_2} &= 2\mathbf{w}_1^\top \mathbf{\Sigma}\mathbf{w}_2 - 2\alpha\mathbf{w}_1^\top \mathbf{w}_2 - \beta\mathbf{w}_1^\top \mathbf{w}_1 = 0
\end{aligned}$$

- Given that $\mathbf{\Sigma}\mathbf{w}_1 = \lambda_1 \mathbf{w}_1$, then:

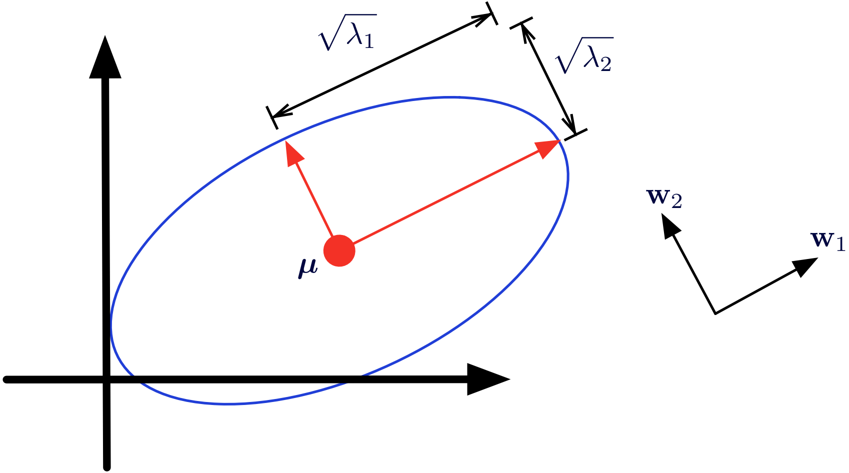
$$\begin{aligned}
\mathbf{w}_1^\top \mathbf{\Sigma}\mathbf{w}_2 = \mathbf{w}_2^\top \mathbf{\Sigma}\mathbf{w}_1 = \lambda_1 \mathbf{w}_2^\top \mathbf{w}_1 &= 0 \\
2\mathbf{w}_1^\top \mathbf{\Sigma}\mathbf{w}_2 - 2\alpha\mathbf{w}_1^\top \mathbf{w}_2 - \beta\mathbf{w}_1^\top \mathbf{w}_1 = -\beta\mathbf{w}_1^\top \mathbf{w}_1 = 0 &\Rightarrow \beta = 0
\end{aligned}$$

- So we simplify $2\mathbf{\Sigma}\mathbf{w}_2 - 2\alpha\mathbf{w}_2 - \beta\mathbf{w}_1 = 0$

Second principal component

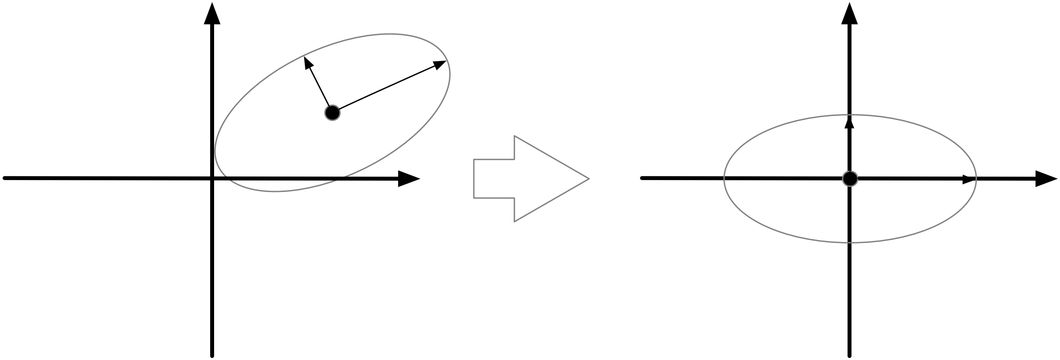
- $\boldsymbol{\Sigma}\mathbf{w}_2 = \alpha\mathbf{w}_2$ implies that $\mathbf{w}_2$ is also an eigenvector of $\boldsymbol{\Sigma}$
  - Since we want to maximize $\mathrm{Var}(\mathbf{w}_2^\top\mathbf{x})$, we choose the eigenvector associated with the second largest eigenvalue, $\alpha = \lambda_2$
- We proceed in the same way for the other dimensions, by choosing as $\mathbf{w}_i$ the eigenvectors, in decreasing order of associated eigenvalues
- Rotation matrix $\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_K]$ thus contains the $K \leq D$ first eigenvectors (with higher eigenvalues)
- Additional properties
  - Since $\boldsymbol{\Sigma}$ is symmetric, eigenvectors are orthogonal
  - Since $\mathbf{w}_i$ are unitary, they form an orthonormal base
  - If $\boldsymbol{\Sigma}$ is defined as positive ($\mathbf{x}^\top\boldsymbol{\Sigma}\mathbf{x} > 0,\ \forall\mathbf{x} \neq 0$), all eigenvalues are non-zero, $\lambda_i \neq 0,\ \forall\lambda_i$
  - Otherwise, the rank of $\boldsymbol{\Sigma}$ gives the number of non-zero eigenvalues

Eigenvalues/eigenvectors and PCA



ACP as a linear transformation

$$\mathbf{z} = \mathbf{W}^\top(\mathbf{x} - \mathbf{m})$$



12.5 Alternative PCA derivation

Alternative derivation

- Alternative PCA derivation
  - Search for a transformation $\mathbf{z} = \mathbf{W}^\top \mathbf{x}$, where variables of $\mathbf{z}$ are uncorrelated
  - Consists in looking for $\mathbf{W}$ so that $\mathrm{Cov}(\mathbf{z}) = \mathbf{D}'$ is diagonal
- Suppose $\mathbf{C}$, matrix $D \times D$, where column $\mathbf{c}_i$ is i-th eigenvector of $\mathbf{S}$, the estimator of $\boldsymbol{\Sigma}$.
  - So $\mathbf{C}\mathbf{C}^\top = \mathbf{C}^\top\mathbf{C} = \mathbf{I}$

$$\begin{aligned}
\mathbf{S} &= \mathbf{S}\mathbf{C}\mathbf{C}^\top \\
&= \mathbf{S}[\mathbf{c}_1\ \mathbf{c}_2\ \cdots\ \mathbf{c}_D]\mathbf{C}^\top \\
&= [\mathbf{S}\mathbf{c}_1\ \mathbf{S}\mathbf{c}_2\ \cdots\ \mathbf{S}\mathbf{c}_D]\mathbf{C}^\top \\
&= [\lambda_1\mathbf{c}_1\ \lambda_2\mathbf{c}_2\ \cdots\ \lambda_D\mathbf{c}_D]\mathbf{C}^\top \\
&= \lambda_1\mathbf{c}_1\mathbf{c}_1^\top + \lambda_2\mathbf{c}_2\mathbf{c}_2^\top + \cdots + \lambda_D\mathbf{c}_D\mathbf{c}_D^\top \\
&= \mathbf{C}\mathbf{D}\mathbf{C}^\top
\end{aligned}$$

- Matrix $\mathbf{D}$ is diagonal, with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_D$

Spectral decomposition

- $\mathbf{CDC}^\top$ is the spectral decomposition of $\mathbf{S}$
- Since $\mathbf{C}$ is orthogonal and $\mathbf{CC}^\top = \mathbf{C}^\top\mathbf{C} = \mathbf{I}$

$$\begin{aligned} \mathbf{S} &= \mathbf{CDC}^\top \\ \mathbf{C}^\top\mathbf{SC} &= \mathbf{C}^\top\mathbf{CDC}^\top\mathbf{C} \\ \mathbf{C}^\top\mathbf{SC} &= \mathbf{D} \end{aligned}$$

- We know that $\mathrm{Cov}(\mathbf{z}) = \mathbf{W}^\top\mathbf{SW}$ and that we want $\mathrm{Cov}(\mathbf{z})$ to be diagonal
  - We thus set $\mathbf{W} = \mathbf{C}$

12.6 PCA illustration

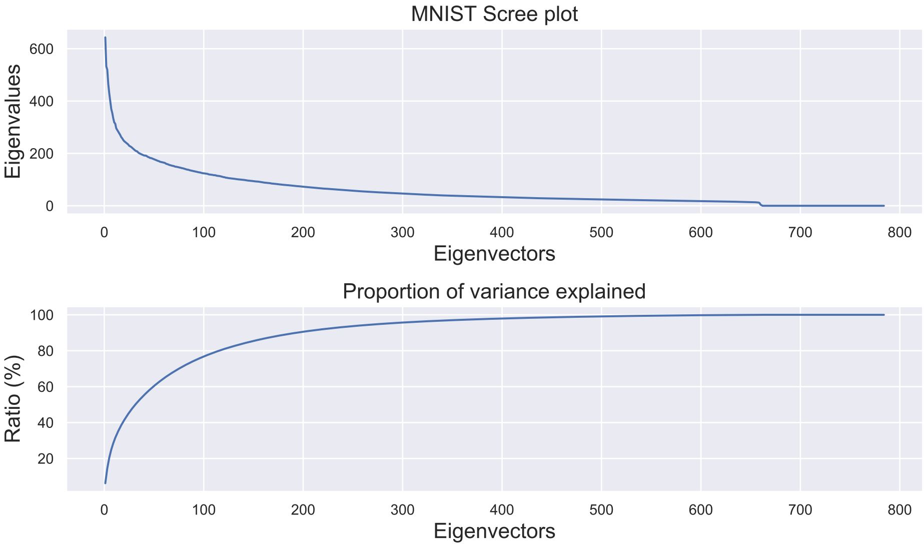
Proportion of variance

- Eigenvalue λ_i indicates the contribution of the component associated to the variance
- Proportion of the variance explained by the K principal components:

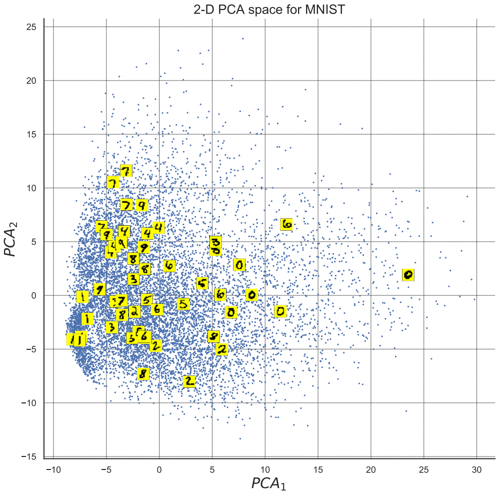
$$\mathrm{PoV} = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_K}{\lambda_1 + \lambda_2 + \cdots + \lambda_K + \cdots + \lambda_D}$$

- High correlation between variables $\Rightarrow$ few components with high eigenvalues
- Scree plot: plot of decreasing eigenvalue sorting

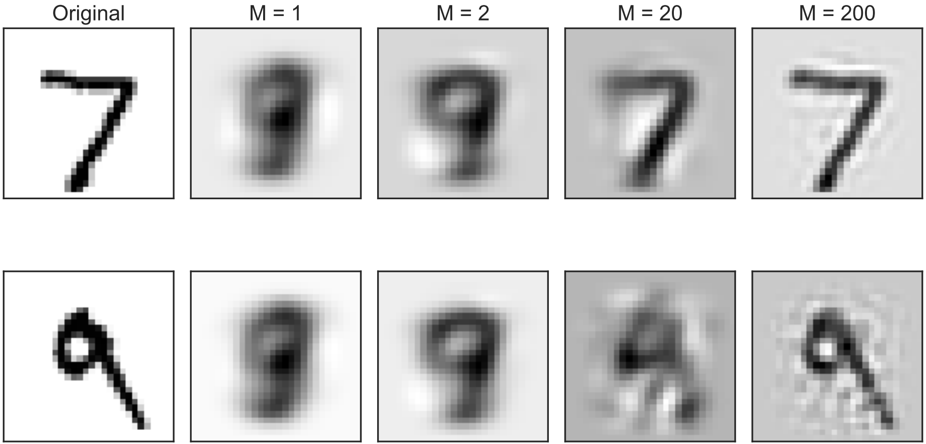
Scree plot


MNIST Scree plot
Eigenvalues
600
400
200
0
0
100
200
300
400
500
600
700
800
Eigenvectors
Proportion of variance explained
Ratio (%)
100
80
60
40
20
0
100
200
300
400
500
600
700
800
Eigenvectors

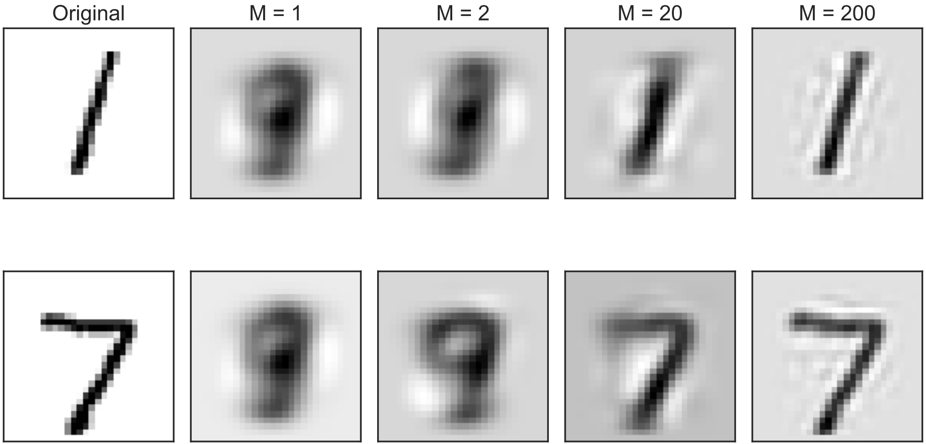
Example with PCA



Character reconstruction: 7 and 9



Character reconstruction: 1 and 7



PCA characteristics

- PCA explains the variance of datasets
  - However sensitive to outliers, which greatly influence the variance
- Very interesting to visualize data
- For high dimensionality (D large), calculations on $\mathbf{S}$ can be heavy ($O(D^2)$)
  - There are methods to reduce calculations to an order of $O(KD)$
- Loss of significance of variables
  - Construction of artificial variables corresponding to a linear combination of the original variables

Reconstruction error

- Data reconstruction
  - Projection in space of z

$$\mathbf{z}^t = \mathbf{W}^\top(\mathbf{x}^t - \boldsymbol{\mu})$$

  - Since $\mathbf{W}$ is orthogonal, $\mathbf{W}\mathbf{W}^\top = \mathbf{I}$

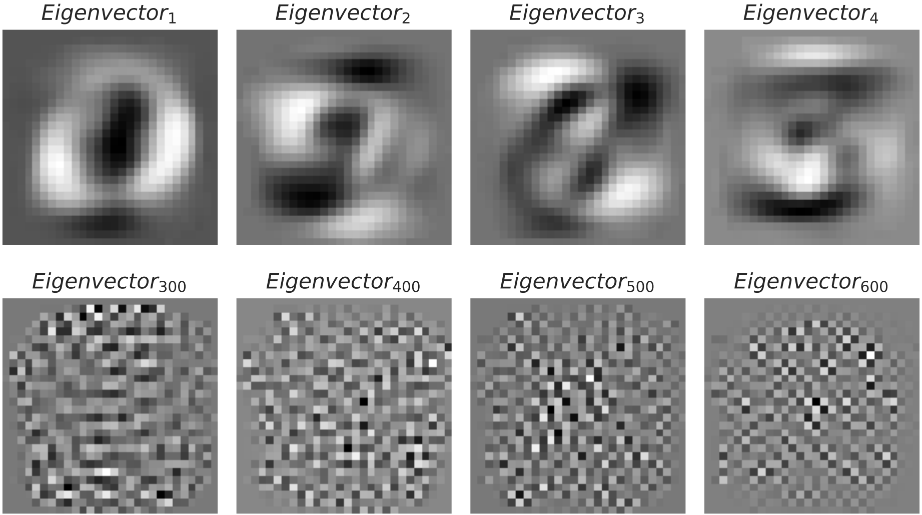
$$\begin{aligned} \mathbf{W}\mathbf{z}^t &= \mathbf{W}\mathbf{W}^\top(\mathbf{x}^t - \boldsymbol{\mu}) \\ \hat{\mathbf{x}}^t &= \mathbf{W}\mathbf{z}^t + \boldsymbol{\mu} \end{aligned}$$

- PCA minimizes reconstruction error

$$\mathrm{err}_{\mathrm{recon}} = \sum_t \|\hat{\mathbf{x}}^t - \mathbf{x}^t\|^2$$

- Reconstruction error depends directly on the number of components K used

Eigendigits



$Eigenvector_1$ $Eigenvector_2$ $Eigenvector_3$ $Eigenvector_4$

$Eigenvector_{300}$ $Eigenvector_{400}$ $Eigenvector_{500}$ $Eigenvector_{600}$

12.7 Whitening transformation

Whitening transformation

- Whitening transformation: center the mean of the data on the origin, remove all covariances and make the variance unitary.

$$\mathbf{x} \sim \mathcal{N}_D(\boldsymbol{\mu},\boldsymbol{\Sigma}) \overset{\text{whiten}}{\mapsto} \mathbf{z} \sim \mathcal{N}_D(0,\mathbf{I})$$

  - Linear transformation

$$\mathbf{z} = \boldsymbol{\Sigma}^{-0.5}(\mathbf{x} - \boldsymbol{\mu})$$

- Strong link with Mahalanobis distance

$$D_M(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$$

  - Mahalanobis distance corresponds to Euclidean distance squared in whitened space
- How to calculate $\boldsymbol{\Sigma}^{-0.5}$?

Spectral decomposition

- $\mathbf{CDC}^\top$ is the spectral decomposition of $\boldsymbol{\Sigma}$
- Since $\mathbf{C}$ is orthogonal and $\mathbf{CC}^\top = \mathbf{C}^\top\mathbf{C} = \mathbf{I}$

$$\begin{aligned} \boldsymbol{\Sigma} &= \mathbf{CDC}^\top \\ \mathbf{C}^\top\boldsymbol{\Sigma}\mathbf{C} &= \mathbf{C}^\top\mathbf{CDC}^\top\mathbf{C} \\ \mathbf{C}^\top\boldsymbol{\Sigma}\mathbf{C} &= \mathbf{D} \end{aligned}$$

- We know that $\mathrm{Cov}(\mathbf{z}) = \mathbf{W}^\top\boldsymbol{\Sigma}\mathbf{W}$ and that we want $\mathrm{Cov}(\mathbf{z})$ to be diagonal
  - We thus set $\mathbf{W} = \mathbf{C}$

Decomposition of the covariance matrix

- Decomposition of the covariance matrix

$$\boldsymbol{\Sigma} = \mathbf{W}\mathbf{D}\mathbf{W}^\top$$

- Eigenvectors of the covariance matrix

$$\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_D]$$

- Eigenvalues of the covariance matrix

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_D \end{bmatrix}$$

Square root of the covariance matrix

- $\mathbf{W}$ is orthogonal, so $\mathbf{W}^{-1} = \mathbf{W}^\top$
- Development of $\boldsymbol{\Sigma}^{0.5}$

$$\begin{aligned}
\boldsymbol{\Sigma} &= \mathbf{W}\mathbf{D}\mathbf{W}^\top = \mathbf{W}\mathbf{D}^{0.5}\mathbf{D}^{0.5}\mathbf{W}^\top \\
&= (\mathbf{W}\mathbf{D}^{0.5}\mathbf{W}^\top)(\mathbf{W}\mathbf{D}^{0.5}\mathbf{W}^\top) = \boldsymbol{\Sigma}^{0.5}\boldsymbol{\Sigma}^{0.5} \\
\boldsymbol{\Sigma}^{-0.5} &= (\mathbf{W}\mathbf{D}^{0.5}\mathbf{W}^\top)^{-1} = \mathbf{W}\mathbf{D}^{-0.5}\mathbf{W}^\top
\end{aligned}$$

- Matrix $\mathbf{D}$ is diagonal, so

$$\mathbf{D}^{-0.5} = \begin{bmatrix} \lambda_1^{-0.5} & 0 & \cdots & 0 \\ 0 & \lambda_2^{-0.5} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_D^{-0.5} \end{bmatrix}$$

Summary

$$\begin{aligned}
\mathbf{x} &\sim \mathcal{N}_D(\boldsymbol{\mu},\boldsymbol{\Sigma}) \\
\mathbf{z} &= \boldsymbol{\Sigma}^{-0.5}(\mathbf{x}-\boldsymbol{\mu}) \\
&= \mathbf{W}\mathbf{D}^{-0.5}\mathbf{W}^\top(\mathbf{x}-\boldsymbol{\mu}) \\
\text{where } \mathbf{W} &= [\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_D] \\
\text{and } \mathbf{D}^{-0.5} &= \begin{bmatrix} \lambda_1^{-0.5} & 0 & \cdots & 0 \\ 0 & \lambda_2^{-0.5} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_D^{-0.5} \end{bmatrix} \\
\mathbf{z} &\sim \mathcal{N}_D(0,\mathbf{I})
\end{aligned}$$

Illustration of a whitening transformation

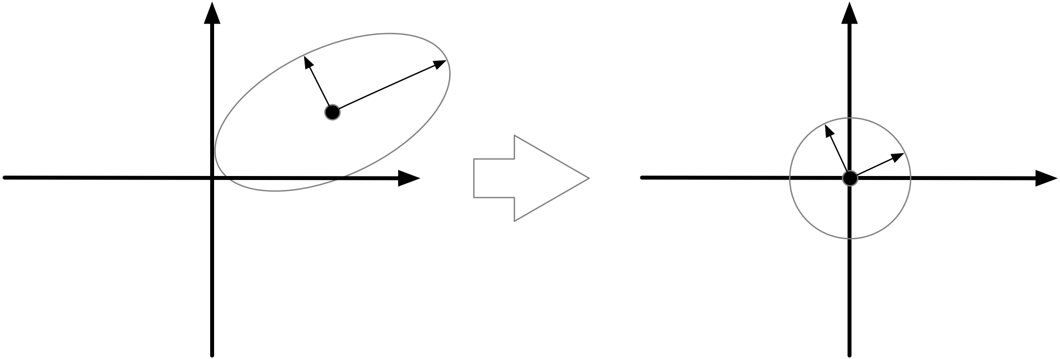
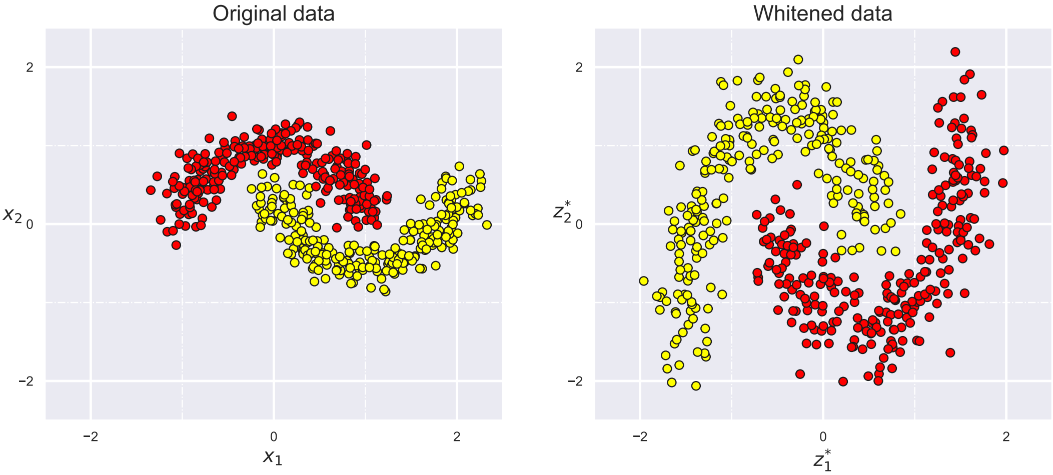


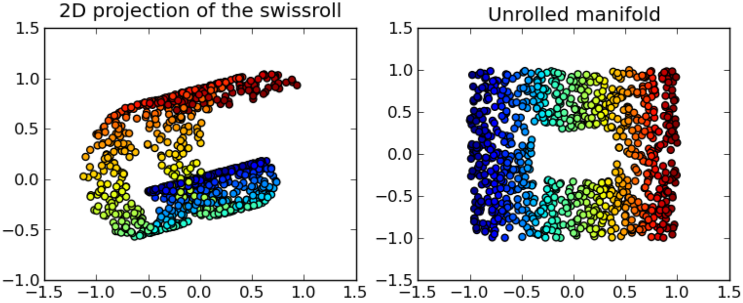
Illustration of a whitening transformation



12.8 Manifold learning

Manifold learning

- *Manifold* hypothesis: data are based on nonlinear space embedded in a higher dimensional space
  - Manifold learning aims at extracting this space
  - Non-linear methods of dimensionality reduction
- Example of the Swiss roll



By Olivier Grisel, CC-BY 3.0, `https://commons.wikimedia.org/wiki/File:Lle_hlle_swissroll.png`.

Multidimensional scaling

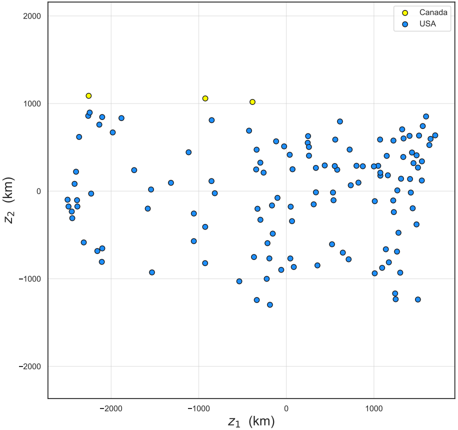
- Multidimensional scaling, MDS
  - Find projection to a space of lower dimensionality preserving as much as possible the values of distance $\|\mathbf{x}^i,\mathbf{x}^j\|$ between all the data pairs of the set $\mathcal{X} = \{\mathbf{x}^t\}_{t=1}^N$
- Sammon's method: determine nonlinear projection $\mathrm{g}(\mathbf{x}|\theta)$ that minimizes

$$E(\theta|\mathcal{X}) = \sum_{t=1,\ldots,N} \sum_{\substack{s=1,\ldots,N \\ s\neq t}} \frac{(\|\mathrm{g}(\mathbf{x}^t|\theta) - \mathrm{g}(\mathbf{x}^s|\theta)\| - \|\mathbf{x}^t - \mathbf{x}^s\|)^2}{\|\mathbf{x}^t - \mathbf{x}^s\|^2}$$

  - $\theta^* = \mathrm{argmin}_\theta\, E(\theta|\mathcal{X})$
  - $\mathrm{g}(\mathbf{x}|\theta)$ can be a polynomial regression, kernel regression, neural network, etc.
  - Measure of arbitrary distance $\|\cdot\|$, does not have to be Euclidean distance

Multidimensional scaling

- Position 128 North American cities based on road distances between them only



t-SNE (t-distributed Stochastic Neighbour Embedding) 1/2

- Determine projection of each data in low dimensionality by preserving the neighbourhood of the original space
  - In practice, useful to visualize data in a 2D or 3D space
- Determine probability to be neighbours between the pairs of the set $\mathcal{X} = \{\mathbf{x}^t\}_{t=1}^N$ in the original space
  - Probability $p_{j|i}$ of selecting $\mathbf{x}^j$ as neighbour of $\mathbf{x}^i$

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}^i - \mathbf{x}^j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}^i - \mathbf{x}^k\|^2/2\sigma_i^2)}$$

  - Probability $p_{i,j} = \frac{p_{i|j}+p_{j|i}}{2N}$ that $\mathbf{x}^j$ is selected as neighbour of $\mathbf{x}^i$ according to a normal law centered on $\mathbf{x}^i$ ($p_{i,i} = 0$)
  - σ_i^2 is adjusted locally for each data (bisection method)

t-SNE (t-distributed Stochastic Neighbour Embedding) 2/2

- Determining the probability of being neighbour between pairs of instances in low dimensional space
  - $\mathbf{z}^t$ is the projection of $\mathbf{x}^t$ in low dimensional space
  - Probability $q_{i,j}$ assuming a Student's Law

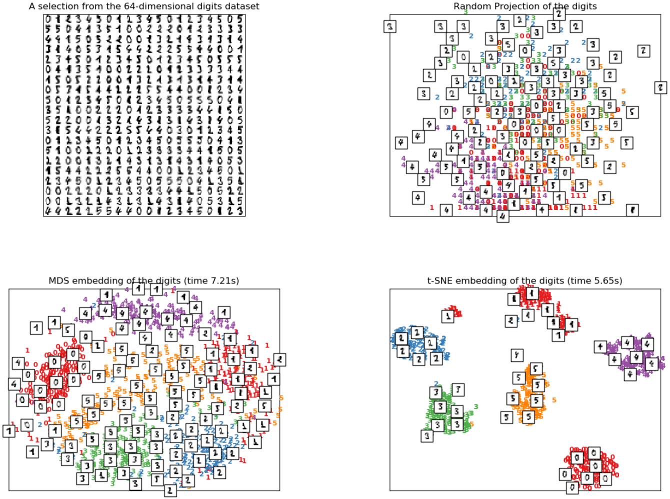
$$q_{i,j} = \frac{(1 - \|\mathbf{z}^i - \mathbf{z}^j\|^2)^{-1}}{\sum\limits_{\substack{k=1,\ldots,N \\ k \neq i}} (1 - \|\mathbf{z}^i - \mathbf{z}^k\|^2)^{-1}}$$

- Learn projections $\mathbf{z} = \mathrm{g}(\mathbf{x}|\theta)$ of the points in low dimensionality in order to minimize the divergence between these probabilities.

$$E(\theta|\mathcal{X}) = KL(P\|Q) = \sum_{t=1,\ldots,N} \sum_{\substack{k=1,\ldots,N \\ k \neq t}} p_{t,k} \log \frac{p_{t,k}}{q_{t,k}|\theta}$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, E(\theta|\mathcal{X})$$

Manifold learning comparison



A selection from the 64-dimensional digits dataset

Random Projection of the digits

MDS embedding of the digits (time 7.21s)

t-SNE embedding of the digits (time 5.65s)

Source: `https://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html`, accessed November 26, 2018, code licensed under BSD 3.

12.9 Preprocessing and data analysis with scikit-learn

Scikit-learn: scaling, standardization and imputation

- Scaling and standardization
  - `preprocessing.MinMaxScaler`: adjust the scale according to minimum/maximum values
  - `preprocessing.scale`: standardization so that variables follow a normal centered-reduced law
- Imputation
  - `impute.SimpleImputer`: imputing values to a fixed value for each variable
    - `strategy`: strategy used for simple imputation, either a mean value (`mean`), a median value (`median`), a more frequent value (`most_frequent`), or a constant (`constant`)
  - `impute.MissingIndicator`: get a mask indicating missing variables of a dataset

Scikit-learn: feature selection

- Univariate selection
  - `feature_selection.VarianceThreshold`: select feature with variance greater than a given threshold
  - `feature_selection.SelectKBest` (`SelectPercentile`): retains the best K (top percentile) features according to a given performance measure
    - `chi2`: χ^2 test between features
    - `f_classif`: ANOVA test between features
    - `mutual_info_classif`: mutual information criterion
- `feature_selection.RFE`: backward selection according to model coefficients
  - `estimator` (object): learning model used for selection
  - `n_features_to_select` (int): total number of features to be selected
  - `step` (int or float)
    - If $\geq$1, number of features removed at each iteration
    - If [0,1), ratio of the number of features removed at each iteration
- `feature_selection.SelectFromModel`: selection from a model (e.g. according to coefficients)

Scikit-learn: principal component analysis

- `decomposition.PCA`: principal component analysis
  - Parameters
    - `n_components` (int): number of components to keep, by default $K = \min(N,D)$
    - `whiten` (bool): normalizes by eigenvectors, thus performing a whitening transformation
  - Attributes
    - `components_` (array): vectors of the principal components (size $K \times D$)
    - `explained_variance_` (array): variance explained by each component (vector of size K)
    - `explained_variance_ratio_` (array): proportion of the variance explained by each component (vector of size K)

Scikit-learn: manifold learning

- `manifold.MDS`: multidimensional scaling
  - `n_components` (int): dimensionality of the destination space
  - `metric` (bool): metric or not
  - `dissimilarity`: measure of distance, i.e. `euclidean` (default) or `precomputed`
- `manifold.TSNE`: t-SNE
  - `n_components` (int): dimensionality of the destination space
  - `perplexity` (float): linked to the number of neighbours used (default: 30)
- Other non-linear manifold learning algorithms
  - `manifold.Isomap`: Isomap algorithm
  - `manifold.LocallyLinearEmbedding`: LLE algorithm
  - `manifold.SpectralEmbedding`: Laplacian eigenmaps algorithm