AUTOENCODERS

Michele Rossi

michele.rossi@dei.unipd.it

Dept. of Information Engineering

University of Padova, IT

Outline

- Why? What? How?
- Review of some matrix algebra
  - Singular Value Decomposition (SVD)
  - Frobenious norm - def and main properties
  - **Revisiting PCA**: minimum error *vs* maximum variance formulations
- Unsupervised learning with FFNN
- Autoencoders (AE)
  - Definition, training objective
  - Equivalence between linear AE and PCA
  - Nonlinear AE
  - Learning strategies
- Denoising AE
- Application example
  - The ECG signal case

Why? What? How?

- What we are going to see in this lesson
  - We are interested in (for now) i.i.d. data sequences
- Data points (samples)
  - Are generated one at a time
  - Can be either i.i.d. or time correlated
  - Are sequentially fed to an algorithm
  - To capture some key features of the data
- Learning objectives
  - i.i.d. seqs: meaningful and compact descriptors (feature vectors)
  - Correlated seqs: capture temporal evolution (RNN)
    - Will be treated in another lesson

Unsupervised learning

- We are interested in
  - Unsupervised learning algorithms
  - That automatically extract useful structure from data
- Useful to what?
  - To **(i)** compress, **(ii)** classify, **(ii)** predict (or interpolate)

"We expect unsupervised learning to become *far more important in the longer term*. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object." [LeCun15]

[LeCun15] Yann LeCun, Yoshua Bengio, Gaoffrey Hinton, "Deep Learning," *Nature*, 2015.

MATRIX ALGEBRA: LOW-RANK APPROXIMATIONS

Singular Value Decomposition (SVD)

- It is a factorization of a real or complex matrix **M**
- It is a generalization of *eigenvalue decomposition* (which holds for a square matrix)

SVD decomposition:

$$\boxed{\boldsymbol{M} = \boldsymbol{U\Sigma V}^{\dagger}} \quad (1)$$

$(\cdot)^{\dagger}$ means transpose conjugate, if **M** is a real matrix, the same relation holds with all matrices real and using the transpose

Singular Value Decomposition (SVD)

SVD Theorem: Let **M** be a complex mxn matrix with

$$\mathrm{rank}(\boldsymbol{M}) = r \leq \min(m, n)$$

Then **M** can be factorized as

$$\boxed{\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\dagger}}$$

- $\boldsymbol{\Sigma}$ is an mxn *diagonal* matrix with *non-negative* real numbers σ_i on the diagonal, called the *singular values* of **M**

$$\sigma_i = \Sigma_{ii} = \sqrt{\lambda_i} \geq 0\,,\ i = 1, \ldots, m$$

- The number of non-zero (and non-negative) singular values is r
- λ_i are the eigenvalues of $\boldsymbol{M}^{\dagger}\boldsymbol{M}$

Singular Value Decomposition (SVD)

SVD Theorem: Let **M** be a complex mxn matrix with

$$\operatorname{rank}(\boldsymbol{M}) = r \leq \min(m, n)$$

Then **M** can be factorized as

$$\boxed{\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\dagger}}$$

matrix **U**

- $\boldsymbol{U}$ is an mxm complex *unitary matrix*

$$\boldsymbol{U}^{\dagger}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^{\dagger} = \boldsymbol{U}\boldsymbol{U}^{-1} = \boldsymbol{I}$$

- Its columns are called the left-singular vectors of $\boldsymbol{M}$
- They form an orthonormal basis $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$
- They are the eigenvectors of $\boldsymbol{M}\boldsymbol{M}^{\dagger}$

Singular Value Decomposition (SVD)

SVD Theorem: Let **M** be a complex mxn matrix with

$$\mathrm{rank}(\boldsymbol{M}) = r \leq \min(m, n)$$

Then **M** can be factorized as

$$\boxed{\boldsymbol{M} = \boldsymbol{U\Sigma V}^{\dagger}}$$

matrix **V**

- $\boldsymbol{V}$ is an nxn complex *unitary matrix* $\boldsymbol{V}^{\dagger}\boldsymbol{V} = \boldsymbol{I}$
- Its columns are called the right-singular vectors of $\boldsymbol{M}$
- They form an orthonormal basis $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$
- They are the eigenvectors of $\boldsymbol{M}^{\dagger}\boldsymbol{M}$

Frobenius norm

- For matrix **M** real, we define (see Appendix 1)

$$\|\boldsymbol{M}\|_F \triangleq \sqrt{\sum_{i,j} |M_{ij}|^2} \quad (2)$$

- if $\mathbf{r}_i$ and $\mathbf{c}_i$ are respectively the rows and columns of **M**
- It holds

$$\|\boldsymbol{M}\|_F^2 = \sum_i \|\boldsymbol{r}_i\|^2 = \sum_j \|\boldsymbol{c}_j\|^2 \quad (3)$$

- using norm-2 of a vector

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T \rightarrow \|\boldsymbol{x}\|^2 \triangleq \sum_{i=1}^{n} x_i^2$$

Frobenius norm and matrix trace

- For matrix **M** real, we define

$$\|\boldsymbol{M}\|_F \triangleq \sqrt{\sum_{i,j} |M_{ij}|^2}$$

- if $\mathbf{r}_i$ and $\mathbf{c}_i$ are respectively the rows and columns of **M**
- Consider $\mathbf{M}^T \mathrm{x} \mathbf{M}$
  - On the main diagonal of this product, we have: $\boldsymbol{c}_i^T \boldsymbol{c}_i = \|\boldsymbol{c}_i\|^2$
  - It follows that

$$\|\boldsymbol{M}\|_F^2 = \sum_i \|\boldsymbol{c}_i\|^2 = \mathrm{trace}(\boldsymbol{M}^T \boldsymbol{M}) = \mathrm{trace}(\boldsymbol{M}\boldsymbol{M}^T) \quad (4)$$

Element-wise inner product

- Let **X** and **Y** be two matrices
- Let $\mathbf{x}_i$ be column i of **X**, $\mathbf{y}_j^T$ be row j of **Y**

$$< \boldsymbol{X}, \boldsymbol{Y} >_e \triangleq \sum_{i,j} x_{ij} y_{ij}$$ "e" = element-wise

- Given this, it holds

$$\|\boldsymbol{X}\|_F^2 =< \boldsymbol{X}, \boldsymbol{X} >_e = \sum_{i,j} x_{i,j}^2$$

- Moreover, it holds

$$\boldsymbol{XY} = \sum_i \boldsymbol{x}_i \boldsymbol{y}_i^T \qquad \|\boldsymbol{XY}\|_F^2 =< \boldsymbol{XY}, \boldsymbol{XY} >_e$$

$col_i(X) \times row_i(Y)$

Another property

- For two matrices **A** and **B** (e.g., 2x2) with, it holds:

$$\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} y_1 & y_2 \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 \\ x_2 y_1 & x_2 y_2 \end{bmatrix}$$

$$\boldsymbol{B} = \boldsymbol{u}\boldsymbol{v}^T = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \end{bmatrix}$$

$$< \boldsymbol{A}, \boldsymbol{B} >_e = x_1 y_1 u_1 v_1 + x_1 y_2 u_1 v_2 + x_2 y_1 u_2 v_1 + x_2 y_2 u_2 v_2$$

Another property

- From the following expression, we can collect $y_i v_i$

$$< \boldsymbol{A}, \boldsymbol{B} >_e = x_1 y_1 u_1 v_1 + x_1 y_2 u_1 v_2 + x_2 y_1 u_2 v_1 + x_2 y_2 u_2 v_2 =$$

$$= (x_1 u_1 + x_2 u_2)(y_1 v_1 + y_2 v_2) =< \boldsymbol{x}, \boldsymbol{u} > (y_1 v_1 + y_2 v_2)$$

and rewrite:

$$< \boldsymbol{x}, \boldsymbol{u} > \sum_i y_i v_i =< \boldsymbol{x}, \boldsymbol{u} >< \boldsymbol{y}, \boldsymbol{v} >=$$

$$= (x_1 u_1 + x_2 u_2)(y_1 v_1 + y_2 v_2) =$$

$$=< \boldsymbol{A}, \boldsymbol{B} >_e =< \boldsymbol{x}\boldsymbol{y}^T, \boldsymbol{u}\boldsymbol{v}^T >_e$$

$$\Rightarrow \quad < \boldsymbol{x}\boldsymbol{y}^T, \boldsymbol{u}\boldsymbol{v}^T >_e =< \boldsymbol{x}, \boldsymbol{u} >< \boldsymbol{y}, \boldsymbol{v} >$$

holds in general

Frobenius norm of product of matrices

- Let **X** and **Y** be two matrices
- Let $\mathbf{x}_i$ be column i of **X**, $\mathbf{y}_j^T$ be row j of **Y**

$$\|\boldsymbol{XY}\|_F^2 = <\boldsymbol{XY}, \boldsymbol{XY}>_e = <\sum_i \boldsymbol{x}_i \boldsymbol{y}_i^T, \sum_j \boldsymbol{x}_j \boldsymbol{y}_j^T>_e$$

$$= \sum_{i,j} <\boldsymbol{x}_i, \boldsymbol{x}_j><\boldsymbol{y}_i, \boldsymbol{y}_j>$$

applying property in the previous slide

$$= \sum_i \|\boldsymbol{x}_i\|^2 \|\boldsymbol{y}_i\|^2 + \sum_{i \neq j} <\boldsymbol{x}_i, \boldsymbol{x}_j><\boldsymbol{y}_i, \boldsymbol{y}_j>$$

Orthonormal matrices

- We have obtained that

$$\|\boldsymbol{XY}\|_F^2 = \sum_i \|\boldsymbol{x}_i\|^2 \|\boldsymbol{y}_i\|^2 + \sum_{i \neq j} < \boldsymbol{x}_i, \boldsymbol{x}_j >< \boldsymbol{y}_i, \boldsymbol{y}_j >$$

- If, e.g., **Y** is an orthonormal matrix, it holds

$$\begin{cases} \|\boldsymbol{y}_i\| = 1 & \forall\, i \\ < \boldsymbol{y}_i, \boldsymbol{y}_j >= 0 & i \neq j \end{cases}$$

- and thus

$$\|\boldsymbol{XY}\|_F^2 = \sum_i \|\boldsymbol{x}_i\|^2 = \|\boldsymbol{X}\|_F^2 \quad (5)$$

the same result holds if **X** is orthonormal

Bound for Frobenius norm of **XY**

- Another useful result:

plain matrix product   from Cauchy-Schwarz inequality

$$\|\boldsymbol{XY}\|_F^2 = \sum_i \sum_j \Big| \underbrace{\sum_k X_{ik} Y_{kj}}^{(\boldsymbol{XY})_{ij}} \Big|^2 \le \sum_i \sum_j \left(\sum_k |X_{ik}|^2 \sum_k |Y_{kj}|^2 \right) =$$

$$= \sum_i \sum_j \left(\sum_{k,\ell} |X_{ik}|^2 |Y_{\ell j}|^2 \right) = \sum_{i,k} |X_{ik}|^2 \sum_{\ell,j} |Y_{\ell j}|^2 =$$

$$= \|\boldsymbol{X}\|_F^2 \|\boldsymbol{Y}\|_F^2$$

$$\boxed{\|\boldsymbol{XY}\|_F \le \|\boldsymbol{X}\|_F \|\boldsymbol{Y}\|_F} \qquad (6)$$

Frobenious norm

- Consider generic matrix **M** real
- From SVD it holds

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \rightarrow \boldsymbol{M}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}$$

- Which means that (using Eq. (5))

apply F-norm to both sides $$||\boldsymbol{M}\boldsymbol{V}||_F^2 = ||\boldsymbol{U}\boldsymbol{\Sigma}||_F^2 \qquad (7)$$

- Both **U** and **V** are *orthonormal*, hence it follows

$$||\boldsymbol{M}||_F^2 = ||\boldsymbol{\Sigma}||_F^2 = \sum_{i,j} |\Sigma_{ij}|^2 = \sum_i \sigma_i^2 \qquad (8)$$

singular values

Lower rank approximations

- Assume that **M** is an mxn real matrix of rank r, with singular values $$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$$
- And with singular value decomposition $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Find the best approximation for **M**, among all real matrices $\mathbf{X}_k$ of size mxn of lower rank

$$\operatorname{rank}(\boldsymbol{X}_k) = k \leq r$$

- The best approximation means

$$\|\boldsymbol{M} - \boldsymbol{X}_k\|_F = \min_{\boldsymbol{X} \in \mathcal{M}_{m,n}} \{\|\boldsymbol{M} - \boldsymbol{X}\|_F \text{ s.t. } \operatorname{rank}(\boldsymbol{X}) = k\} \qquad (9)$$

Lower rank approximations

- Assume that **M** is an mxn real matrix of rank r, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$
- And with singular value decomposition $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Then, among all real matrices $\mathbf{X}_k$ of size mxn of lower rank

$$\operatorname{rank}(\boldsymbol{X}_k) = k \leq r$$

- The best low-rank approximation is $\boldsymbol{X}_k = \boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^T$ (10)
- Where $\boldsymbol{\Sigma}_k$ is a *diagonal matrix* with singular values $\sigma_1, \sigma_2, \ldots, \sigma_k$

Lower rank approximations

- Proof.
  - Take generic matrix **X** of rank k (with k≤r), size mxn
  - Writing the Frobenious norm and left- and righ-multiplying inside of it by $\mathbf{U}^T$ and **V** (the F-norm is invariant), respectively, we obtain

$$\|\boldsymbol{M} - \boldsymbol{X}\|_F = \|\boldsymbol{U\Sigma V}^T - \boldsymbol{X}\|_F = \|\boldsymbol{\Sigma} - \boldsymbol{U}^T\boldsymbol{XV}\|_F$$

  - Denoting $\mathbf{N=U^TXV}$, an mxn matrix of rank k, we write:

1. diagonal terms up to r

2. Diagonal terms after r

$$\|\boldsymbol{\Sigma} - \boldsymbol{N}\|_F^2 = \sum_{i,j} |\Sigma_{ij} - N_{ij}|^2 = \sum_{i=1}^{r} |\sigma_i - N_{ii}|^2 + \sum_{i>r} |N_{ii}|^2 + \sum_{i\neq j} |N_{ij}|^2$$

due to structure of Σ

off-diagonal terms

Lower rank approximations

Proof.

- Up to now, we have found:

$$\|\boldsymbol{M}-\boldsymbol{X}\|_F^2 = \|\boldsymbol{\Sigma}-\boldsymbol{N}\|_F^2 = \sum_{i,j} |\Sigma_{ij} - N_{ij}|^2 =$$

$$= \sum_{i=1}^{r} |\sigma_i - N_{ii}|^2 + \cancel{\sum_{i>r} |N_{ii}|^2} + \cancel{\sum_{i\neq j} |N_{ij}|^2}$$

- that is minimal if second and third term are zero, and first term is minimized, i.e.,

$$\begin{cases} N_{ii} = \sigma_i & i = 1, \ldots, k \ (k \leq r) \\ N_{ii} = 0 & i > k \\ N_{ij} = 0 & i \neq j \end{cases} \qquad (11)$$

Lower rank approximations

Discussion

$$\|\boldsymbol{M}-\boldsymbol{X}\|_F^2=\sum_{i=1}^{r}|\sigma_i-N_{ii}|^2+\sum_{i>r}|N_{ii}|^2+\sum_{i\neq j}|N_{ij}|^2$$

- Is this really the best thing we could do for the first term?
- Maybe, can we make it equal to zero as well?
- By taking

$$\begin{cases} N_{ii}=\sigma_i & i=1,\ldots,\boxed{r} \\ N_{ii}=0 & i>\boxed{r} \\ N_{ij}=0 & i\neq j \end{cases}$$

NO, in this case matrix **N** would have rank r (and the approximating matrix would be equal to the original one **X**) → NOT permitted, we are looking for a low-rank k approx.

Lower rank approximations

- Proof. (continued)
  - Using $\mathbf{X}_k$ as in the theorem statement, $\boldsymbol{X}_k = \boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^T$
  - From the definition of **N**, we get

$$\boldsymbol{N} = \boldsymbol{U}^T\boldsymbol{X}_k\boldsymbol{V} = \boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{\Sigma}_k$$

  - Which proves that the **N** that minimizes the F-norm is exactly equal to the rank k approx. provided by $\mathbf{X}_k$
  - It holds

$$\begin{cases} (\Sigma_k)_{ii} = N_{ii} = \sigma_i & i = 1, \ldots, k \\ (\Sigma_k)_{ii} = N_{ii} = 0 & i > k \\ (\Sigma_k)_{ij} = N_{ij} = 0 & i \neq j \end{cases}$$

QED

Lower rank approximations

- Discussion
  - Using $\mathbf{X}_k$
  - Quantify the F-norm of the difference between
    - **M** : original matrix and $\mathbf{X}_k$: its **low rank approximation**
  - From the previous calculations, it descends – approximation error

$$\|\boldsymbol{M}-\boldsymbol{X}\|_F^2 = \|\boldsymbol{\Sigma}-\boldsymbol{N}\|_F^2 = \sum_{i=k+1}^{r} |\sigma_i|^2$$

  - Are the terms not contained in the **N** matrix
  - Moreover, since $\sigma_i = \sqrt{\lambda_i}$ (λ_i are the eigenvalues of $\mathbf{M}^T\mathbf{M}$)
  - It also holds

$$\|\boldsymbol{M}-\boldsymbol{X}\|_F^2 = \sum_{i=k+1}^{r} \lambda_i$$

Divide by n and you get the average distortion provided by PCA

PCA – minimum error formulation

- Setup
  - Let $\boldsymbol{X}$ be the mxn data matrix
  - And $\boldsymbol{X}'$ be the zero mean data matrix
    - Obtained by removing the mean vector from all vectors in $\boldsymbol{X}$

PCA – minimum error formulation

- Setup
  - Let **P** be the PCA transform matrix $\boldsymbol{Y}' = \boldsymbol{P}\boldsymbol{X}'$
  - We define **P** by stacking the first p<m eigenvectors (in the rows of **P**), related to the p largest eigvenvalues of: $\mathrm{Cov}(\boldsymbol{X}') = \frac{1}{n}\boldsymbol{X}'(\boldsymbol{X}')^T$
  - Since the number of rows of matrix **P** is p<m → information is lost
  - The approximated (reconstructed) data from **Y**' is obtained as

$$\tilde{\boldsymbol{X}}' = \boldsymbol{P}^T\boldsymbol{Y}' = \boldsymbol{P}^T\boldsymbol{P}\boldsymbol{X}' \quad (12)$$

  - Now, define $\boldsymbol{W}^T \triangleq \boldsymbol{P} \in \mathbb{R}^{p\times m}$
  - Given all this, the reconstruction error J is

$$J = \|\boldsymbol{X}' - \tilde{\boldsymbol{X}}'\|_F^2 = \|\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'\|_F^2 \quad (13)$$

PCA – minimum error formulation

- Reconstruction error J (using (4))

$$J = \|\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'\|_F^2 = \mathrm{trace}((\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}')(\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}')^T) =$$

$$= \mathrm{trace}((\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}')((\boldsymbol{X}')^T - (\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T) =$$

$$= \mathrm{trace}(\boldsymbol{X}'(\boldsymbol{X}')^T) - 2\mathrm{trace}(\boldsymbol{X}'(\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T) + \mathrm{trace}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'(\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T)$$

using 1) $(A + B)^T = A^T + B^T$ 2) $tr(sum_i A_i) = sum\ tr(A_i)$ 3) cyclic prop. tr(ABC) = tr(CAB) = tr(BCA)

- Moreover,

$$\mathrm{trace}(\boldsymbol{X}'(\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T) = \mathrm{trace}((\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}')$$ cyclic prop. tr(ABC) = tr(CAB) = tr(BCA)

$$\mathrm{trace}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'(\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T) = \mathrm{trace}((\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}') = \mathrm{trace}((\boldsymbol{X}')^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}')$$

cyclic property $\quad$ $\mathbf{W}^T \mathbf{W} = I_p$

$$\Rightarrow \|\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'\|_F^2 = \mathrm{trace}(\boldsymbol{X}'(\boldsymbol{X}')^T) - \mathrm{trace}(\boldsymbol{W}^T\boldsymbol{X}'(\boldsymbol{X}')^T\boldsymbol{W})$$

reconstruction error

constant
(does not depend on **W)**

projected variance
(cyclic prop. again)

Minimum error *vs* projected variance

$$J = \|\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'\|_F^2 = \mathrm{trace}(\boldsymbol{X}'(\boldsymbol{X}')^T) - \mathrm{trace}(\boldsymbol{W}^T\boldsymbol{X}'(\boldsymbol{X}')^T\boldsymbol{W}) \quad (14)$$

- This relation says that minimizing the **reconstruction error J** (F-norm, optimization variable is $\mathbf{W}^T$) is equivalent to maximizing the projected variance (second, negative term on the right)
- The PCA tranformation matrix $\mathbf{P}=\mathbf{W}^T$ minimizes J and, at the same time, maximizes the projected variance
- **Note:** if p=m
  - **W** is square, invertible, $\mathbf{WW}^T = I_m$ and J=0

PCA formulation with minimum error

- The PCA transform $\boldsymbol{W}^T \triangleq \boldsymbol{P} \in \mathbb{R}^{p \times m}$
- where $\boldsymbol{Y}' = \boldsymbol{P}\boldsymbol{X}'$

is also a solution to:

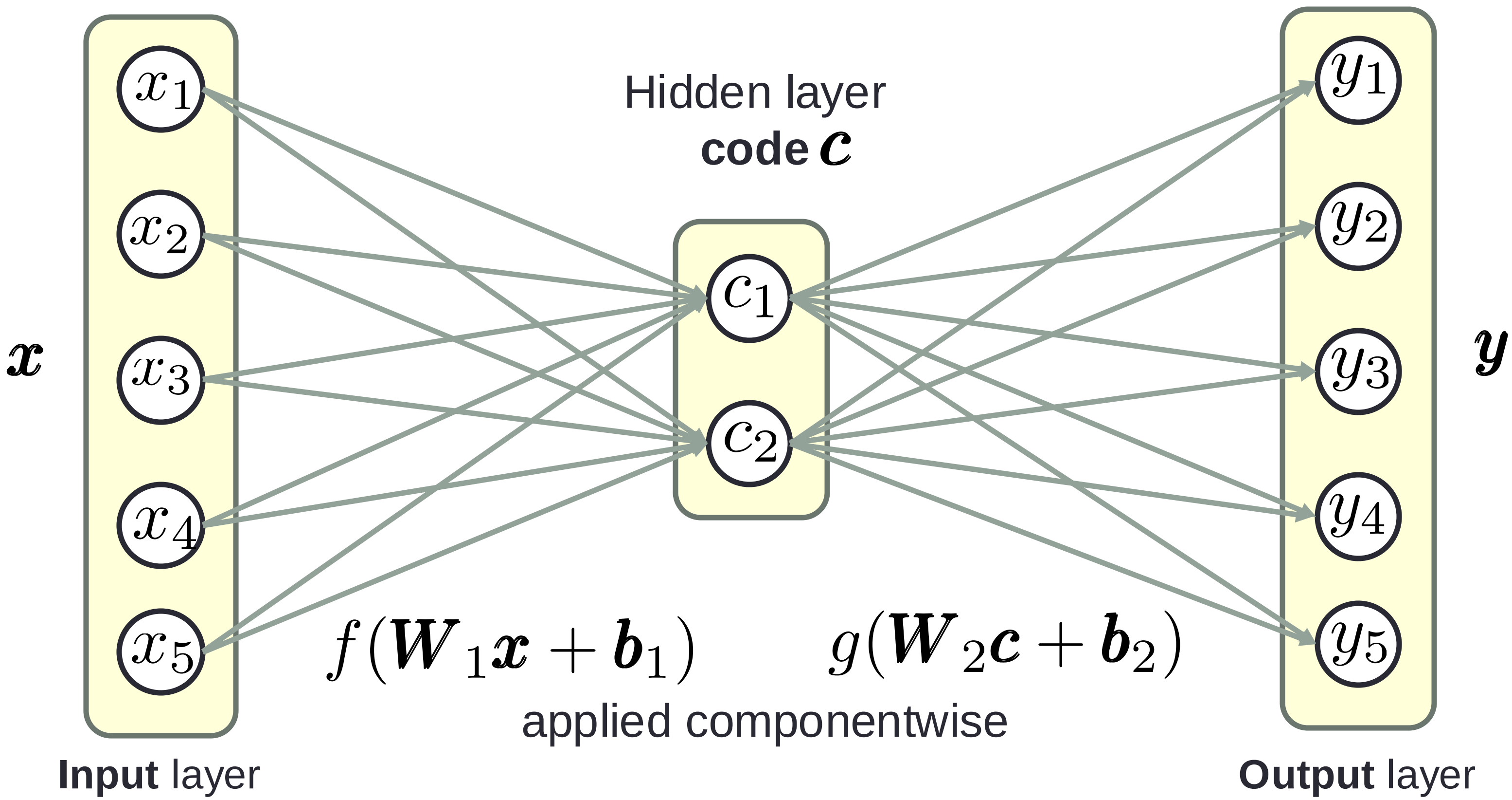
$$\min_{\boldsymbol{W} \in \mathbb{R}^{m \times p}} \|\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'\|_F^2 \text{ , subject to: } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_p \quad (15)$$

minimum error formulation of PCA

AUTOENCODERS

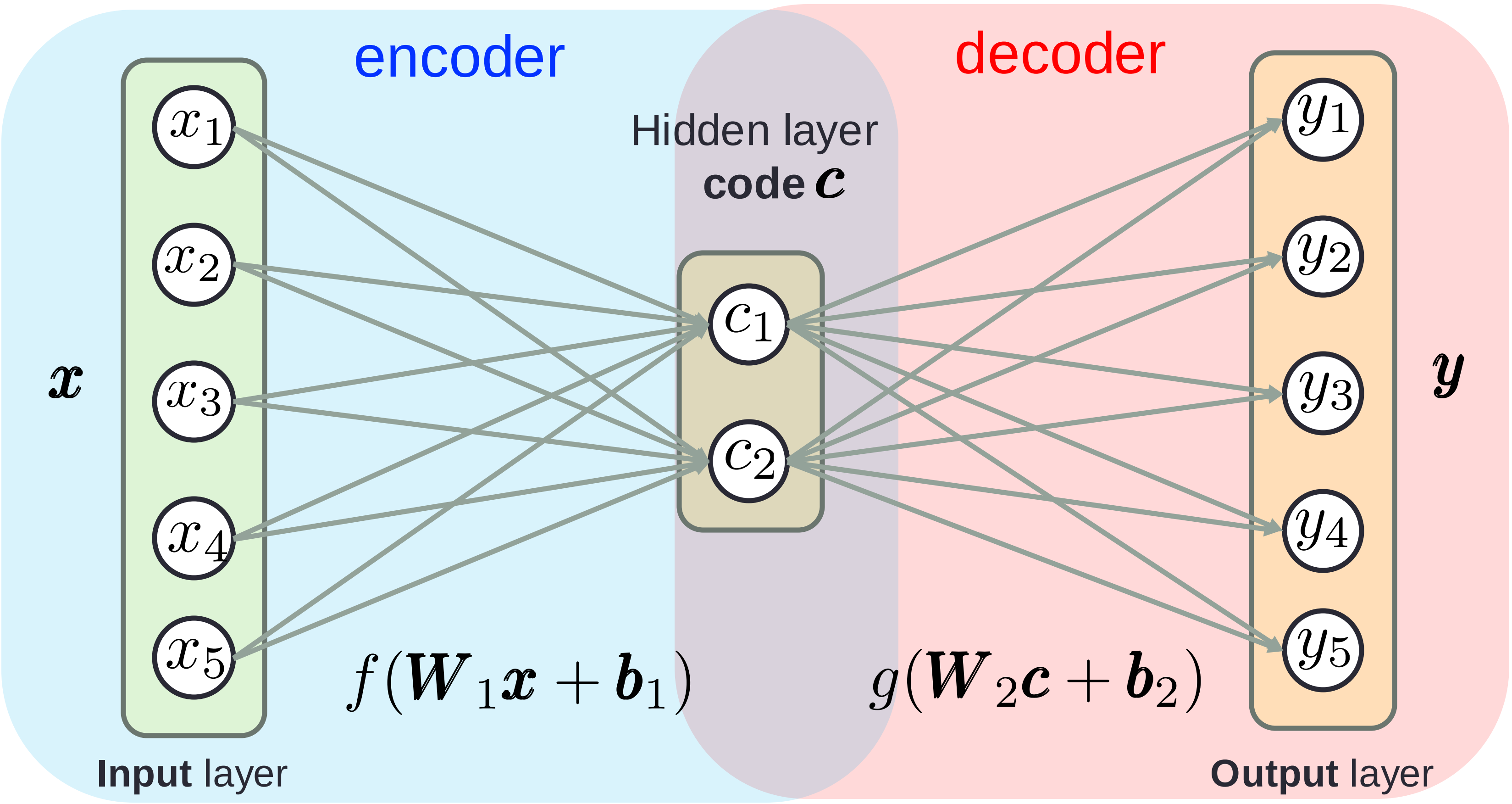
Autoencoder through FFNNs

- Feed Forward Neural Networks (FFNN)
  - Implement functions, do not have internal states (memory cells)
  - Artificial neurons organized in layers, signals flow from input (left) to output (right)
  - Structure is fully connected (i.e., dense)



Autoencoder through FFNNs

- Encoder-decoder FFNN architecture
  - Intended to reproduce the input



The FFNN autoencoder

- Input vectors $\boldsymbol{x}_k \in \mathbb{R}^m$, $k = 1, 2, \ldots, n$
- Data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$
- Encoder output

$$\boldsymbol{c}_k = f(\boldsymbol{W}_1 \boldsymbol{x}_k + \boldsymbol{b}_1)$$ code vector

  - $f(\cdot)$: componentwise nonlinearity
  - Encoder weights: $\boldsymbol{W}_1 \in \mathbb{R}^{p \times m}, \boldsymbol{b}_1 \in \mathbb{R}^p$

- Decoder output

$$\boldsymbol{y}_k = g(\boldsymbol{W}_2 \boldsymbol{c}_k + \boldsymbol{b}_2)$$

  - $g(\cdot)$: componentwise nonlinearity
  - Decoder weights: $\boldsymbol{W}_2 \in \mathbb{R}^{m \times p}, \boldsymbol{b}_2 \in \mathbb{R}^m$

First setup – the linear autoencoder

- With f() and g() identities
- Input vectors $\boldsymbol{x}_k \in \mathbb{R}^m$, $k = 1, 2, \ldots, n$
- Input data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$
- Output matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$
- Code matrix $\boldsymbol{C} \in \mathbb{R}^{p \times n}$

$$\boldsymbol{C} = \boldsymbol{W}_1 \boldsymbol{X} + \boldsymbol{b}_1 \mathbf{1}^T$$

$$\boldsymbol{Y} = \boldsymbol{W}_2 \boldsymbol{C} + \boldsymbol{b}_2 \mathbf{1}^T \quad (16)$$

with f() and g() identities

$\mathbf{1}$: column vector of all 1s

Autoencoder – objective J

- Objective: make each output vector $\mathbf{y}_k$ as close as possible to the corresponding input vector $\mathbf{x}_k$
- Squared error norm corresponds to (cost function)

$$J = \sum_{k=1}^{n} \|\boldsymbol{x}_k - \boldsymbol{y}_k\|^2$$

- It can be compactly rewritten using the F-norm (see Eq. (3) in this slide set), as follows

$$J = \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 = \|\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C} - \boldsymbol{b}_2\boldsymbol{1}^T\|_F^2 \quad (17)$$

Autoencoder – objective J

- Squared error norm

$$J = \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 = \|\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C} - \boldsymbol{b}_2\mathbf{1}^T\|_F^2$$

- Objective: finding $\mathbf{b}_2$ that minimizes J:

$$\boldsymbol{b}_2^\star = \operatorname*{argmin}_{\boldsymbol{b}_2} \|\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C} - \boldsymbol{b}_2\mathbf{1}^T\|_F^2$$
$$= \operatorname*{argmin}_{\boldsymbol{b}_2} [\operatorname{trace}(\boldsymbol{A}\boldsymbol{A}^T)]$$

- with (using Eq. (4))

$$\begin{cases} \|\boldsymbol{A}\|_F^2 = \operatorname{trace}(\boldsymbol{A}\boldsymbol{A}^T) \\ \boldsymbol{A} \triangleq \boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C} - \boldsymbol{b}_2\mathbf{1}^T \end{cases}$$

Autoencoder – optimal bias $\mathbf{b}_2$

$$\boldsymbol{b}_2^\star = \underset{\boldsymbol{b}_2}{\operatorname{argmin}} \|\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C} - \boldsymbol{b}_2\mathbf{1}^T\|_F^2$$

- Since

$$\|\boldsymbol{A}\|_F^2 = \operatorname{trace}(\boldsymbol{A}\boldsymbol{A}^T)$$

$$\nabla_{\boldsymbol{b}_2}\left(\operatorname{trace}(\boldsymbol{A}\boldsymbol{A}^T)\right) = \mathbf{0}$$

- Leads to

$$\boldsymbol{b}_2^\star = \frac{1}{n}(\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C})\mathbf{1} \qquad (18)$$

Autoencoder – objective J

- Replacing optimal $\mathbf{b}_2$* (Eq. (18)), we obtain

$$J = \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 = \|\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C} - \boldsymbol{b}_2^{\star}\mathbf{1}^T\|_F^2 =$$

$$= \|\boldsymbol{X}' - \boldsymbol{W}_2\boldsymbol{C}'\|_F^2 \qquad (19)$$

- with

$$\boldsymbol{X}' = \boldsymbol{X}\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) \qquad (20)$$

$$\boldsymbol{C}' = \boldsymbol{C}\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) \qquad (21)$$

zero mean matrices

Autoencoder – effect of $\mathbf{b}_2^*$

- Note: average vectors for input, hidden and output units are

$$\bar{\boldsymbol{x}} = \frac{\boldsymbol{X}\mathbf{1}}{n} \qquad \bar{\boldsymbol{c}} = \frac{\boldsymbol{C}\mathbf{1}}{n} \qquad \bar{\boldsymbol{y}} = \frac{\boldsymbol{Y}\mathbf{1}}{n}$$

- From these, it descends

$$\boldsymbol{X}' = \boldsymbol{X}\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) = \boldsymbol{X} - \bar{\boldsymbol{x}}\mathbf{1}^T \qquad (22)$$

$$\boldsymbol{C}' = \boldsymbol{C}\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) = \boldsymbol{C} - \bar{\boldsymbol{c}}\mathbf{1}^T \qquad (23)$$

- The optimal bias vector $\mathbf{b}_2$ reduces the training problem (17) to zero-average patterns

Autoencoder – effect of **b**$_2^*$

- Moreover: recalling (17) and (18)

$$\boldsymbol{Y} = \boldsymbol{W}_2\boldsymbol{C} + \boldsymbol{b}_2^\star \mathbf{1}^T \quad \boldsymbol{b}_2^\star = \frac{1}{n}(\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C})\mathbf{1} \quad \overline{\boldsymbol{x}} = \frac{\boldsymbol{X}\mathbf{1}}{n}$$

$$\Rightarrow \quad \boldsymbol{Y} = \boldsymbol{W}_2\boldsymbol{C} + \left(\overline{\boldsymbol{x}} - \frac{\boldsymbol{W}_2\boldsymbol{C}\mathbf{1}}{n}\right)\mathbf{1}^T \quad (24)$$

$$\boldsymbol{Y}\mathbf{1} = \boldsymbol{W}_2\boldsymbol{C}\mathbf{1} + \left(\overline{\boldsymbol{x}} - \frac{\boldsymbol{W}_2\boldsymbol{C}\mathbf{1}}{n}\right)\mathbf{1}^T\mathbf{1}$$

$$(\mathbf{1}^T\mathbf{1} = n)$$

$$\boldsymbol{Y}\mathbf{1} = n\overline{\boldsymbol{x}}$$

Autoencoder – effect of **b**$_2^*$

- Moreover: recalling (17) and (18)

$$\boldsymbol{Y} = \boldsymbol{W}_2\boldsymbol{C} + \boldsymbol{b}_2^\star\boldsymbol{1}^T \quad \boldsymbol{b}_2^\star = \frac{1}{n}(\boldsymbol{X} - \boldsymbol{W}_2\boldsymbol{C})\boldsymbol{1} \quad \overline{\boldsymbol{x}} = \frac{\boldsymbol{X}\boldsymbol{1}}{n}$$

$$\Rightarrow \quad \boldsymbol{Y} = \boldsymbol{W}_2\boldsymbol{C} + \left(\overline{\boldsymbol{x}} - \frac{\boldsymbol{W}_2\boldsymbol{C}\boldsymbol{1}}{n}\right)\boldsymbol{1}^T \quad (24)$$

$$\boldsymbol{Y}\boldsymbol{1} = \boldsymbol{W}_2\boldsymbol{C}\boldsymbol{1} + \left(\overline{\boldsymbol{x}} - \frac{\boldsymbol{W}_2\boldsymbol{C}\boldsymbol{1}}{n}\right)\boldsymbol{1}^T\boldsymbol{1}$$

$$\overline{\boldsymbol{y}} = \frac{\boldsymbol{Y}\boldsymbol{1}}{n} = \overline{\boldsymbol{x}} \quad (25)$$

Optimal bias scales input and output vectors to the same average value

Minimizing J

- So far, we have obtained

$$J = \|\boldsymbol{X}' - \boldsymbol{W}_2\boldsymbol{C}'\|_F^2 \Rightarrow J_{\min} = \min_{\boldsymbol{W}_2\boldsymbol{C}'} \|\boldsymbol{X}' - \boldsymbol{W}_2\boldsymbol{C}'\|_F^2$$

?

- Let the SVD of **X'** be

$$\boldsymbol{X}' = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

- Assume that $\mathbf{W}_2$ has low rank p<m (usually verified)
- Then, from (9) and (10), it descends that the best p-rank approximation is

$$\boxed{\boldsymbol{W}_2\boldsymbol{C}' = \boldsymbol{U}\boldsymbol{\Sigma}_p\boldsymbol{V}^T} \qquad (26)$$

optimal approx.

Let’s look at the first linear layer

- For the first layer, it holds $\boldsymbol{C} = \boldsymbol{W}_1 \boldsymbol{X} + \boldsymbol{b}_1 \mathbf{1}^T$
- Multiplying both sides by $\left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n} \right)$
- Using (22) and (23), leads to (as. $\mathbf{1}^T \mathbf{1} = n$)

$$\boldsymbol{C}' = \boldsymbol{W}_1 \boldsymbol{X}' + \boldsymbol{b}_1 \mathbf{1}^T \left(\boldsymbol{I} - \frac{\mathbf{1}\mathbf{1}^T}{n} \right) = \boldsymbol{W}_1 \boldsymbol{X}' \quad (27)$$

- which shows that **$\mathbf{b}_1$** is arbitrary

Finding the optimal matrices

- Hence, the error function for the linear AE becomes

$$J = \|\boldsymbol{X}' - \boldsymbol{W}_2\boldsymbol{C}'\|_F^2 = \|\boldsymbol{X}' - \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{X}'\|_F^2 \quad (28)$$

- J is minimized with (from (26) and (27))

$$\boldsymbol{W}_2\boldsymbol{C}' = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{X}' = \boldsymbol{U}\Sigma_p\boldsymbol{V}^T \quad (29)$$

- The following is a solution (i.e., minimizes J)

$$\begin{cases} \boldsymbol{W}_2 = \boldsymbol{U}\boldsymbol{T}^{-1} \\ \boldsymbol{C}' = \boldsymbol{W}_1\boldsymbol{X}' = \boldsymbol{T}\Sigma_{\boldsymbol{p}}\boldsymbol{V}^T \end{cases} \quad (30)$$

- For an *arbitrary* orthogonal pxp matrix **T**
- Hence, the solution to is not unique

Putting it all together

- The error function for the linear AE becomes

$$J = \|\boldsymbol{X}' - \boldsymbol{W}_2\boldsymbol{C}'\|_F^2 = \|\boldsymbol{X}' - \boxed{\boldsymbol{W}_2}\boxed{\boldsymbol{W}_1}\boldsymbol{X}'\|_F^2$$

going from latent space back to original space

moving input to latent space

  - multiple solutions exist for the two matrices, back propagation finds one

- With PCA we have

$$J = \|\boldsymbol{X}' - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}'\|_F^2$$

  - Optimal **W** is uniquely determined by SVD applied to Cov(**X**')
  - The columns of **W** are the principal vectors (eigenvalues of Cov(**X**')), ordered according to the amplitude of the eigenvalues of Cov(**X**')

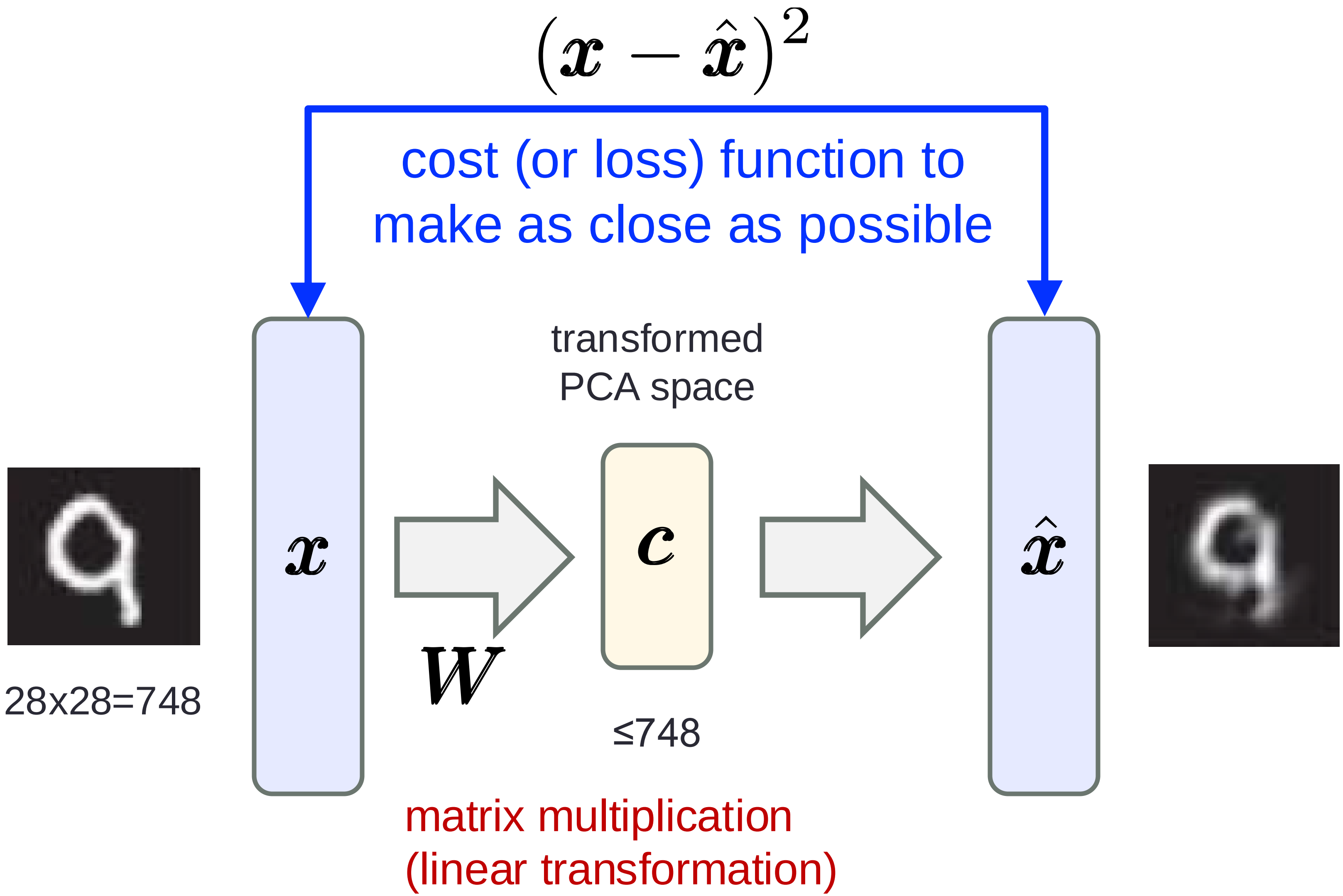
Take home message

- The linear AE behaves as a PCA, in the sense that they minimize the same error between original and reconstructed data vectors
- They both behave as data compressors – the compressed representation is the inner vector **c** for the AE (spanning the same subspace)
- However, the AE
  - unlike PCA, the coordinates of **c** can be correlated and not necessarily sorted in descending order of variance (eigenvalues)
  - does not ensure that entries of vector **c** are uncorrelated
  - matrix $\mathbf{W}_2$ is, in general, equal to

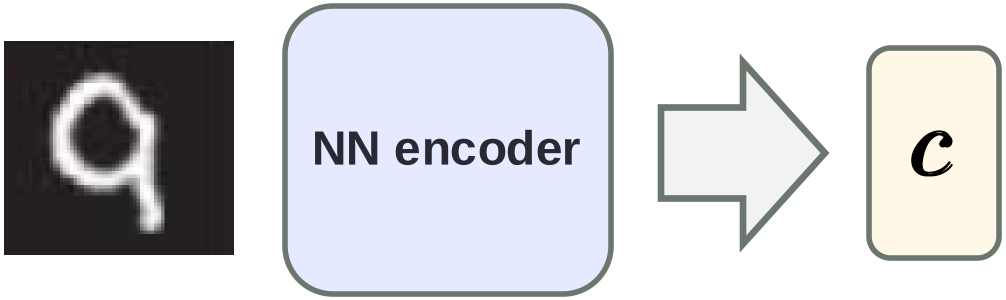
$$\boldsymbol{W}_2 = \boldsymbol{W}\boldsymbol{O}$$

  - where **W** is the matrix found by PCA, **O** is an orthogonal matrix

Principal Component Analysis (PCA)



Autoencoder [Hinton06]

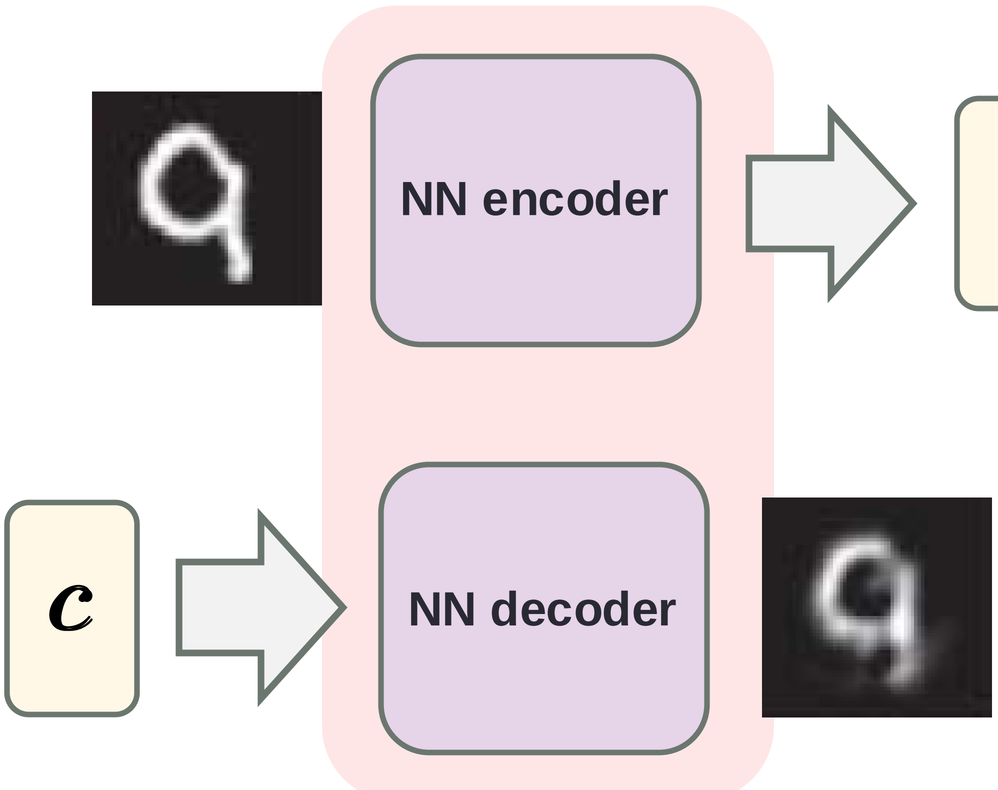


the code: a compact representation of the input object

the code: can be used to reconstruct the object (up to some accuracy)

[Hinton06] G. E. Hinton, R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” Science, Vol. 313, July 2006.

Autoencoder



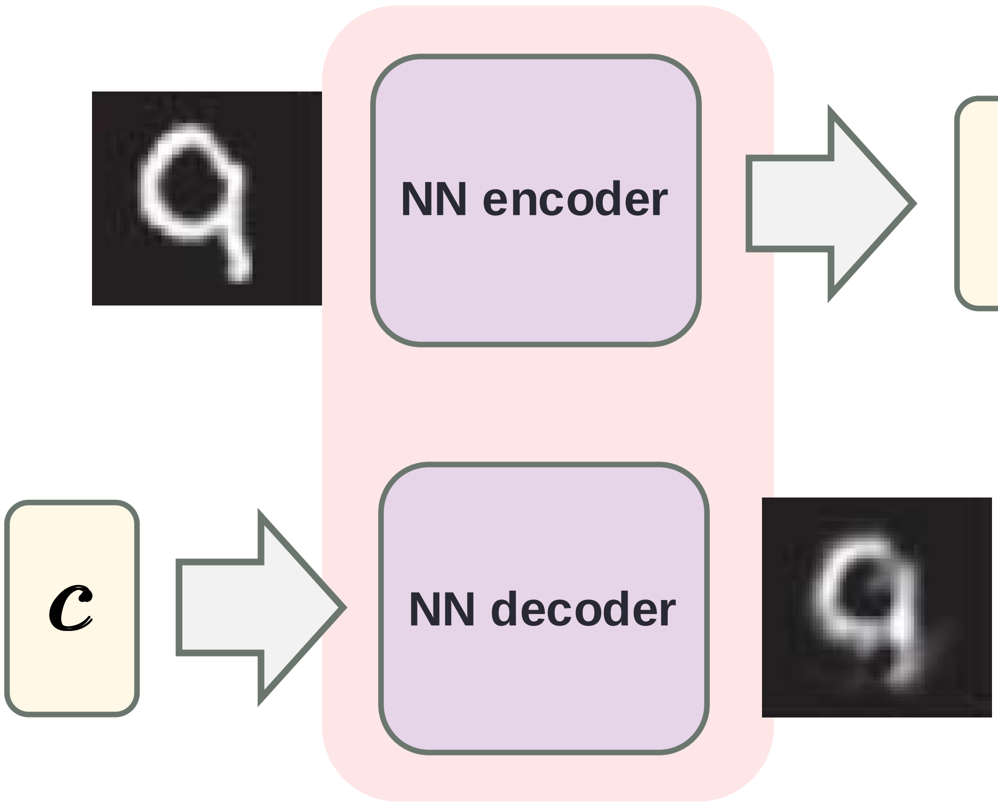
the code: a compact representation of the input object

the code: can be used to reconstruct the object (up to some accuracy)

Key point 1: encoder and decoder

- are jointly trained

Autoencoder



the code: a compact representation of the input object

the code: can be used to reconstruct the object (up to some accuracy)

Key point 2: encoder and decoder

- are non linear (usually via sigmoid, tanh, ReLU, activations)

Autoencoder caveats

- If after training $\boldsymbol{x} = \boldsymbol{y}$ everywhere
  - This is useless and must be avoided !!!
- Autoencoders (AE) should be trained such that
  - They are unable to learn to copy perfectly
  - They only copy approximately
  - And only copy input that resembles ***valid*** input data
- In this way AE
  - Must prioritize *which aspects* of the input should be copied
  - Usually learn useful properties of the data

Example: linear multiplication by identity matrix copies perfectly every input vector but is useless

AE training

- Unsupervised
  - Label (target output) is the input data itself
  - Are trained using gradient descent as any FFNN
- Standard gradient descent techniques
  - Batch-mode gradient descent: all data points considered to compute the true error derivative wrt the FFNN weights
  - Stochastic gradient descent (SGD): 1) one input vector at a time is fed to the FFNN, 2) gradient computed solely based on it, 3) network weights are updated using gradient descent of this pointwise gradient, 4) reiterate for all points
  - Mini-batches: between batch-mode and SGD

Learning strategies

- Objective
  - Prevent the AE from *just copying* the data
- Solutions
  - Limit the AE approximation *capacity*
- Popular strategies
  - Undercomplete AEs
  - Sparse AEs
  - Denoising AEs

Undercomplete AE

- Input $\boldsymbol{x}$
- Output $\boldsymbol{y} = g(f(\boldsymbol{x}))$
- Loss (error) $\mathcal{L}(\boldsymbol{x}, g(f(\boldsymbol{x})))$

- Under completeness
  - Code dimension p << m input dimension
  - Forces the AE to capture the most salient data features

- Special case
  - Linear encoder/decoder and quadratic loss: the AE learns to span the same subspace as PCA (they are equivalent)

Sparse AE

- Input $\boldsymbol{x}$
- Output $\boldsymbol{y} = g(f(\boldsymbol{x}))$
- Loss (error) $\mathcal{L}(\boldsymbol{x}, g(f(\boldsymbol{x})))$
- Training criterion $\mathcal{L}(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{c})$

Sparsity penalty:

- It is a regularizer term
- Expresses a preference over functions
- For instance (Laplacian prior), we have:

$$\Omega(\boldsymbol{c}) = \lambda \sum_i |c_i|$$

Laplacian prior (1/2)

- Idea
  - See the sparse AE framework as approximating ML training of a *generative model* that has latent variables (code **c**)
  - Explicit joint distribution (*input* **x** and *latent* variable **c**)

$$p_{\text{model}}(\boldsymbol{x}, \boldsymbol{c}) = p_{\text{model}}(\boldsymbol{c}) p_{\text{model}}(\boldsymbol{x} | \boldsymbol{c})$$

$$\log p_{\text{model}}(\boldsymbol{x}, \boldsymbol{c}) = \log p_{\text{model}}(\boldsymbol{c}) + \log p_{\text{model}}(\boldsymbol{x} | \boldsymbol{c})$$

- Laplacian prior over c_i (assume c_i i.i.d.)

$$p_{\text{model}}(c_i) = \frac{\lambda}{2} e^{-\lambda |c_i|}$$

Laplacian prior (2/2)

- Laplacian prior over c_i

i.i.d. assumption

$$p_{\text{model}}(c_i) = \frac{\lambda}{2} e^{-\lambda |c_i|} \qquad p_{\text{model}}(\boldsymbol{c}) = \prod_i p_{\text{model}}(c_i)$$

- As a training cost, we take –log of the prior

$$-\log p_{\text{model}}(\boldsymbol{c}) = \sum_i \left(\lambda |c_i| - \log \frac{\lambda}{2} \right) = \Omega(\boldsymbol{c}) + \text{constant}$$

  - Minimizing the cost: amounts to maximizing the prior pdf
  - Other priors are possible: lead to different penalties
  - This show why the features learned by an AE are useful: they describe the latent variables that explain the input

Denoising AE (1/3)

- Rather than constrain the representation (the code)
  - Train the AE for a more challenging task:
  - cleaning partially corrupted input (denoising)
- From [Vincent10]:

"a good representation is one that can be obtained robustly ***from a corrupted input*** *and that will be* ***useful for recovering the corresponding clean input****..."*

[Vincent10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, 2010.

Denoising AE (1/3)

- Rather than constrain the representation (the code)
  - Train the AE for a more challenging task:
  - cleaning partially corrupted input (denoising)
- From [Vincent10]:

"...our goal is not the task of denoising per se. Rather, ***denoising is advocated and investigated as a training criterion for learning to extract useful features*** *that will constitute better higher level representations..."*

[Vincent10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, 2010.

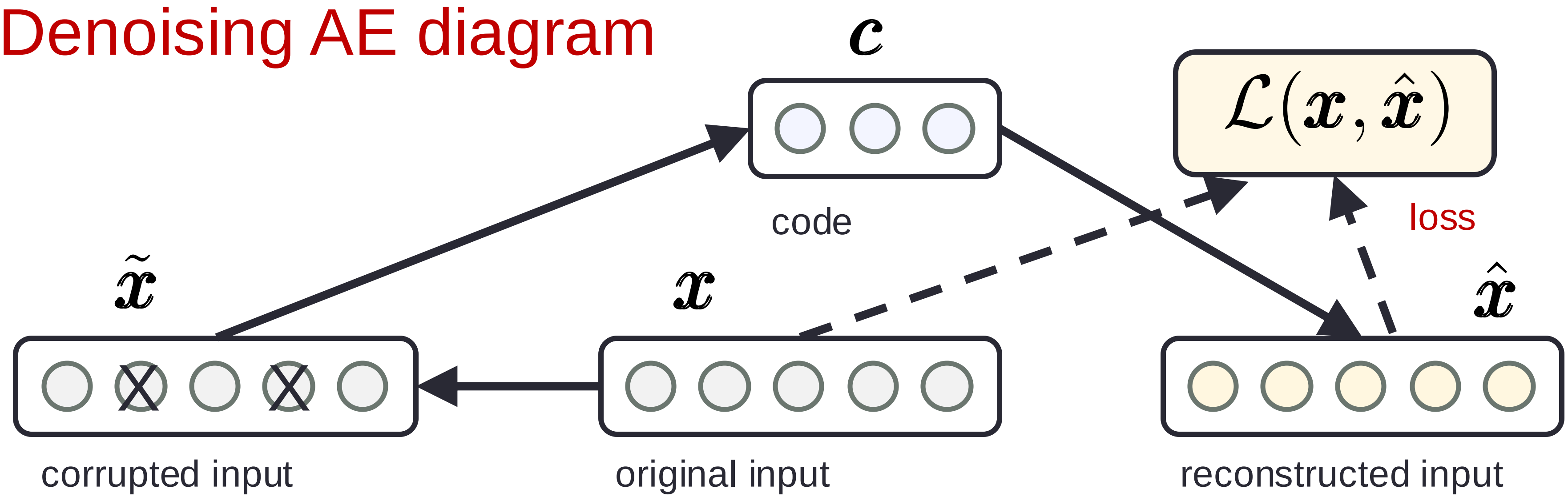
Denoising AE (2/3)

- Rationale

  1) It is expected that a high-level representation should be rather stable under a corruption of the input

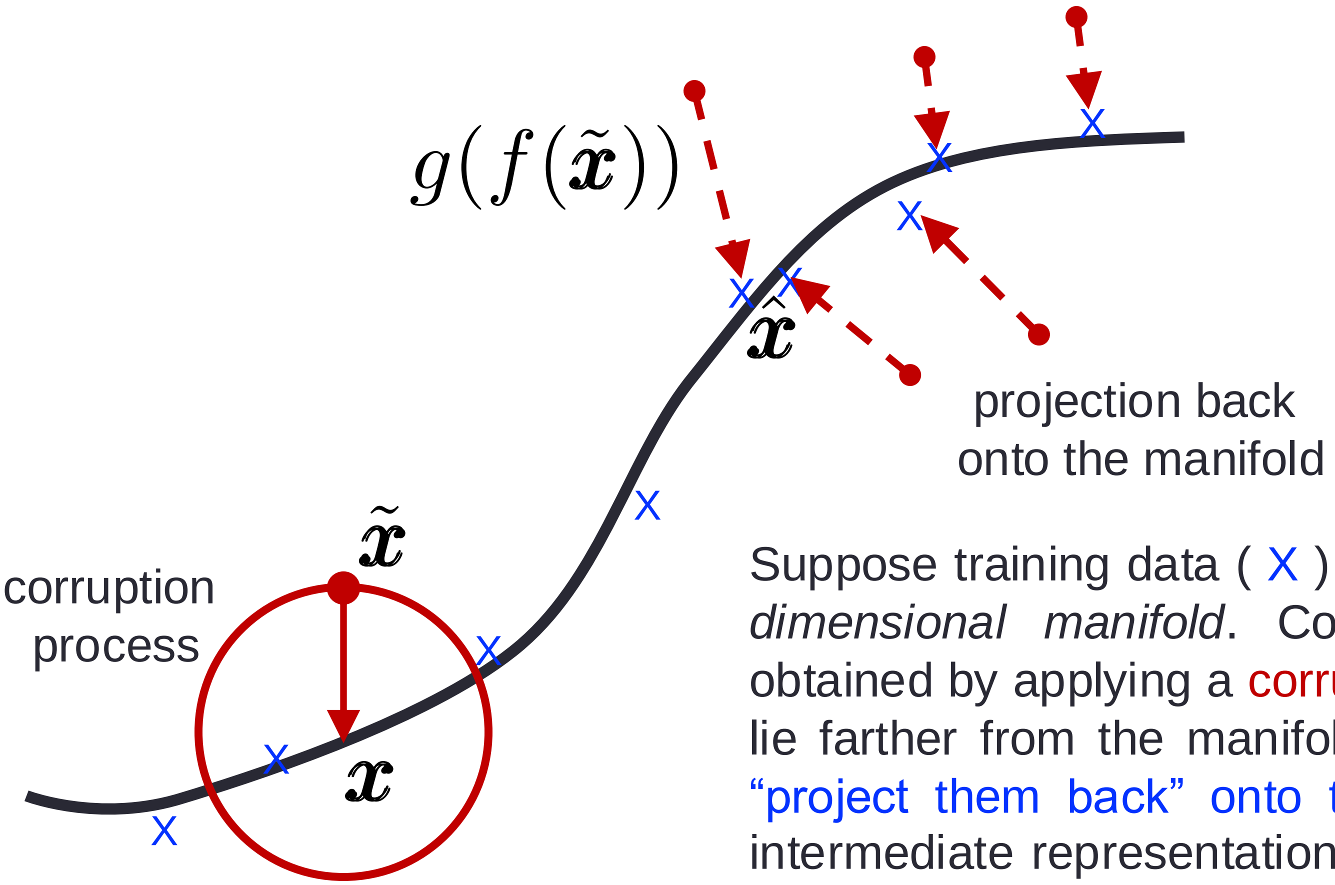
  2) It is expected that performing the denoising task requires extracting features that capture useful features in the input distribution

- Denoising AE diagram



Denoising AE (3/3)

Geometrical interpretation - manifold learning



Suppose training data (X) concentrate near a *low-dimensional manifold*. Corrupted examples (•) obtained by applying a corruption process generally lie farther from the manifold. The model learns to "project them back" onto the manifold. Thus, the intermediate representation **c**=f(**x**) may be seen as a coordinate system for points on the manifold

AE for Missing Data Imputation

Algorithms' Family	Algorithm & Article	Score
Matrix Completion△	Singular Value Thresholding (Tran et al., 2017)	4
	SoftImpute (Tran et al., 2017)	4
	OptSpace (Tran et al., 2017)	4
Evolutionary△□	Genetic Algorithms (GA) (Tran et al., 2017)	4
	Genetic Algorithms (Malek et al., 2018)	4
Connectionist△‖	Denoising Autoencoder (Tran et al., 2017)	4
	Stacked Denoising Autoencoder (Tran et al., 2017)	4
	Multi-modal Autoencoder (Tran et al., 2017)	4
	Deep Canonically Correlated Autoencoders (Tran et al., 2017)	3
	Variational Autoencoder (Ma et al., 2019)	4
Other□	Orthogonal Matching Pursuit (Malek et al., 2018)	4
	Basis Pursuit (Malek et al., 2018)	4

Imputation in images

1. AE has worse results
2. AE has same results
3. AE has marginally better results
4. AE has significantly better results

[Pereira2020] R. C. Pereira, M. S. Santos, P. P. Rodriguez, P. H. Abreu, "Reviewing Autoencoders for Missing Data Imputation: Technical Trends, Applications and Outcomes," *Journal of Artificial Intelligence Research*, Vol. 69, 2020.

Master's degree ICT Internet Multimedia Engineering

VECTOR QUANTIZATION AS AUTOENCODER-BASED COMPRESSION – THE CASE OF ECG SIGNALS

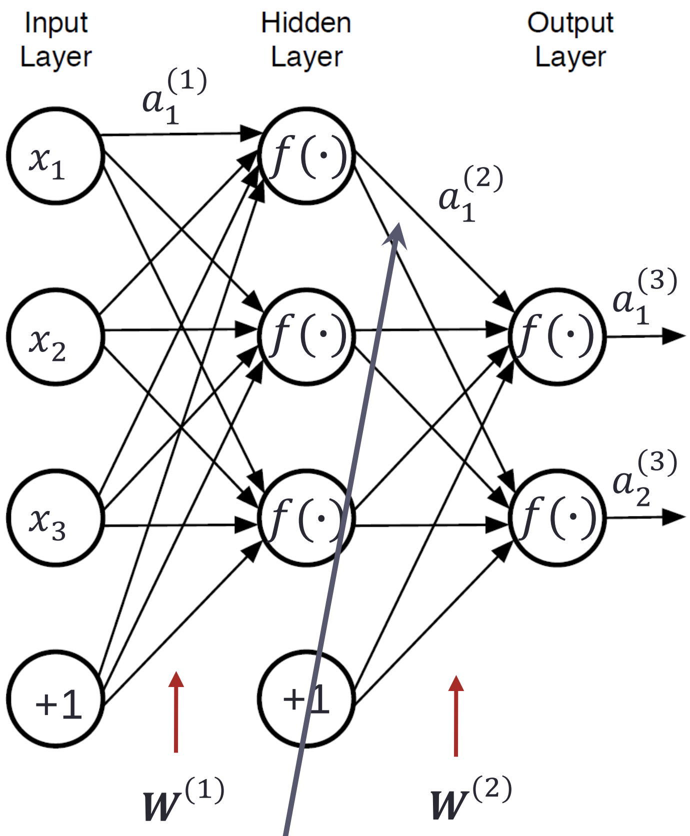
[DelTesta2015] Davide Del Testa, Michele Rossi, “Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders,” *IEEE Signal Processing Letters*, 2015.

DIPARTIMENTO DI INGEGNERIA DELL’INFORMAZIONE

DIPARTIMENTO MATEMATICA

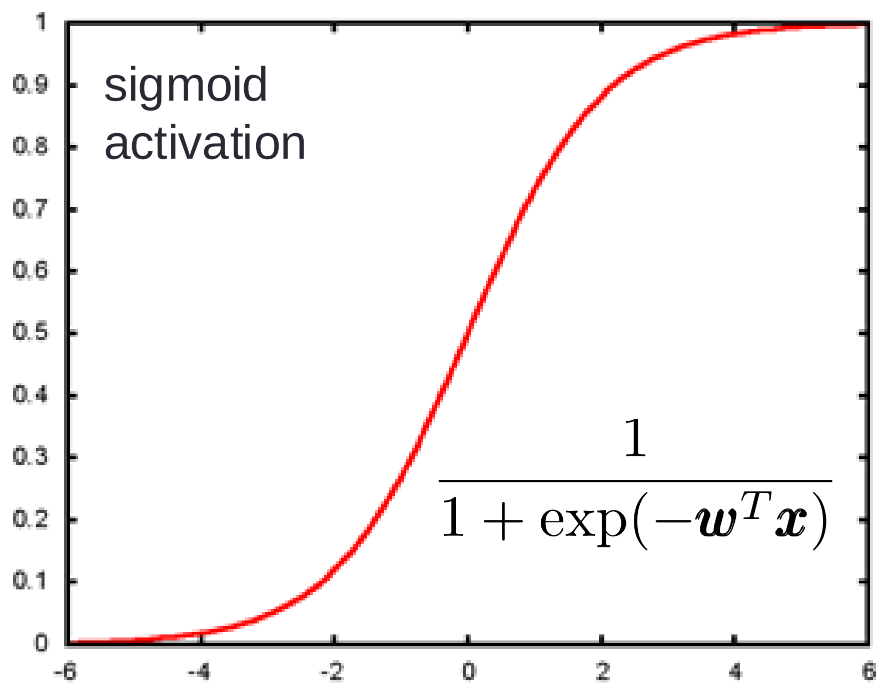
UNIVERSITÀ DEGLI STUDI DI PADOVA

Neural networks

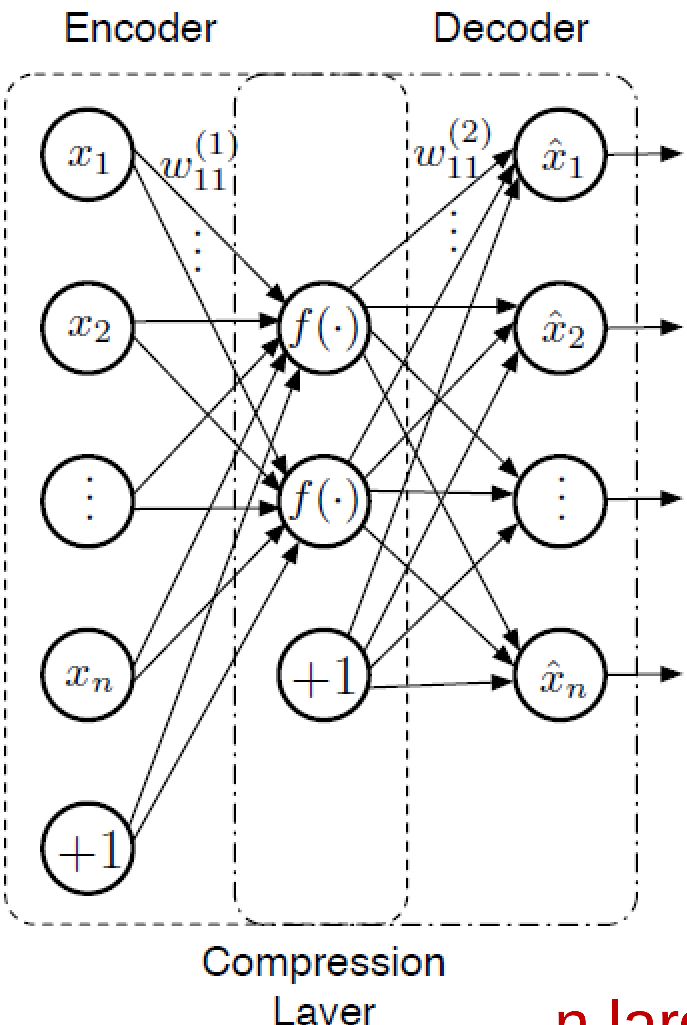


$$a_i^{(n+1)}(\boldsymbol{x}) = f(\boldsymbol{w}^T\boldsymbol{x})$$

- Feed Forward NN
- Layers of neurons
- Non-linear activation functions
- Set of weights



Autoencoders



- Unsupervised learning
- Same no. of input & output neurons
- Target values = input
- Goal: learn to reconstruct the input

$\boldsymbol{x} \in \mathbb{R}^n$

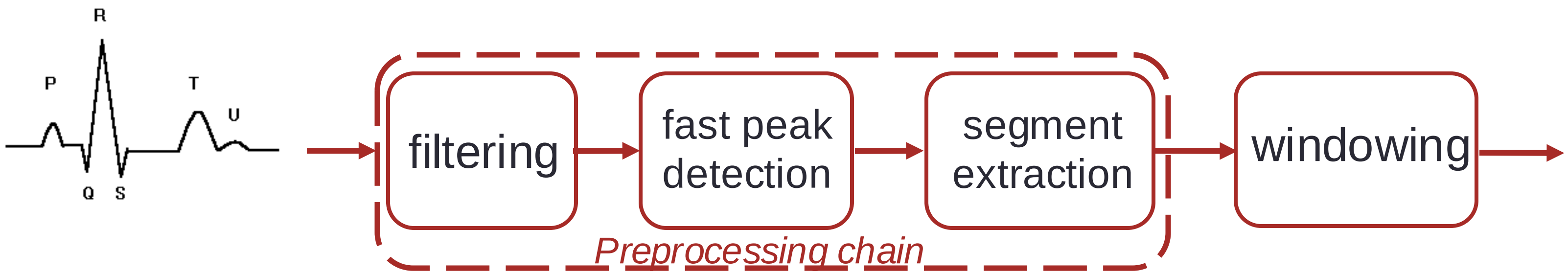
mapped onto $\boldsymbol{x}' \in \mathbb{R}^c$, $c < n$

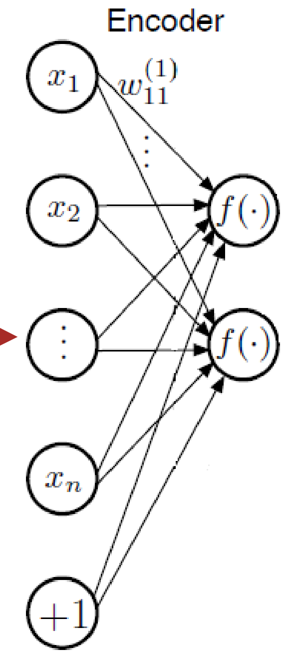
c hidden units

n larger than 250 ECG samples / segment

Compression architecture

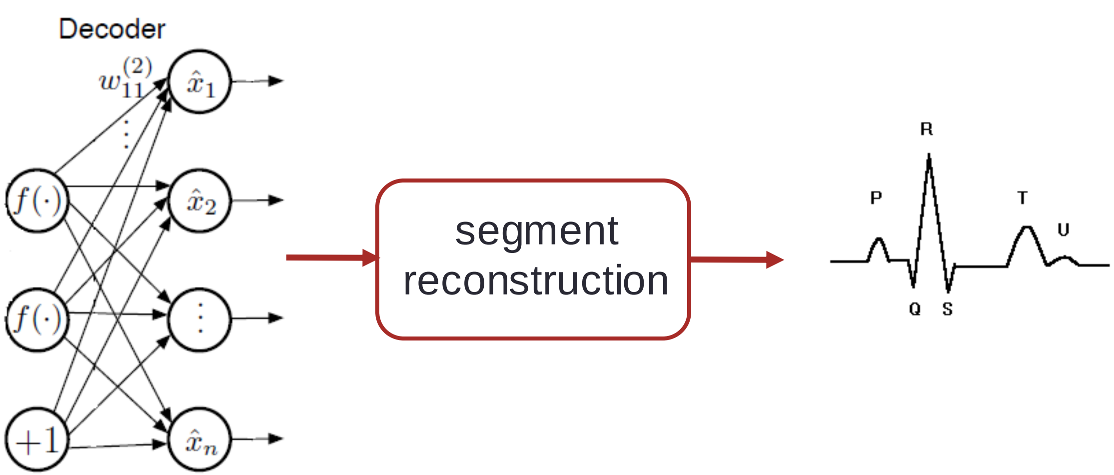
Compressor (transmitter)





c values TX

Decompressor (receiver)



weight set $\mathbf{W}$
computed offline
(training examples)

Performance metrics (1/2)

E1 - Energy associated with compression

- We count the number of operations (divisions, additions, comparisons)
- Translate them into the corresponding number of clock cycles
- From clock cycles → energy consumption
- MCU: ARM Cortex M4

E2 - Energy associated with transmission / reception

- Consider the compressed data stream
- Compute the energy consumption associated with TX / RX
- Radio: Texas Instruments CC2541 (Bluetooth SoC)

⇒ total energy E1+E2

Performance metrics (2/2)

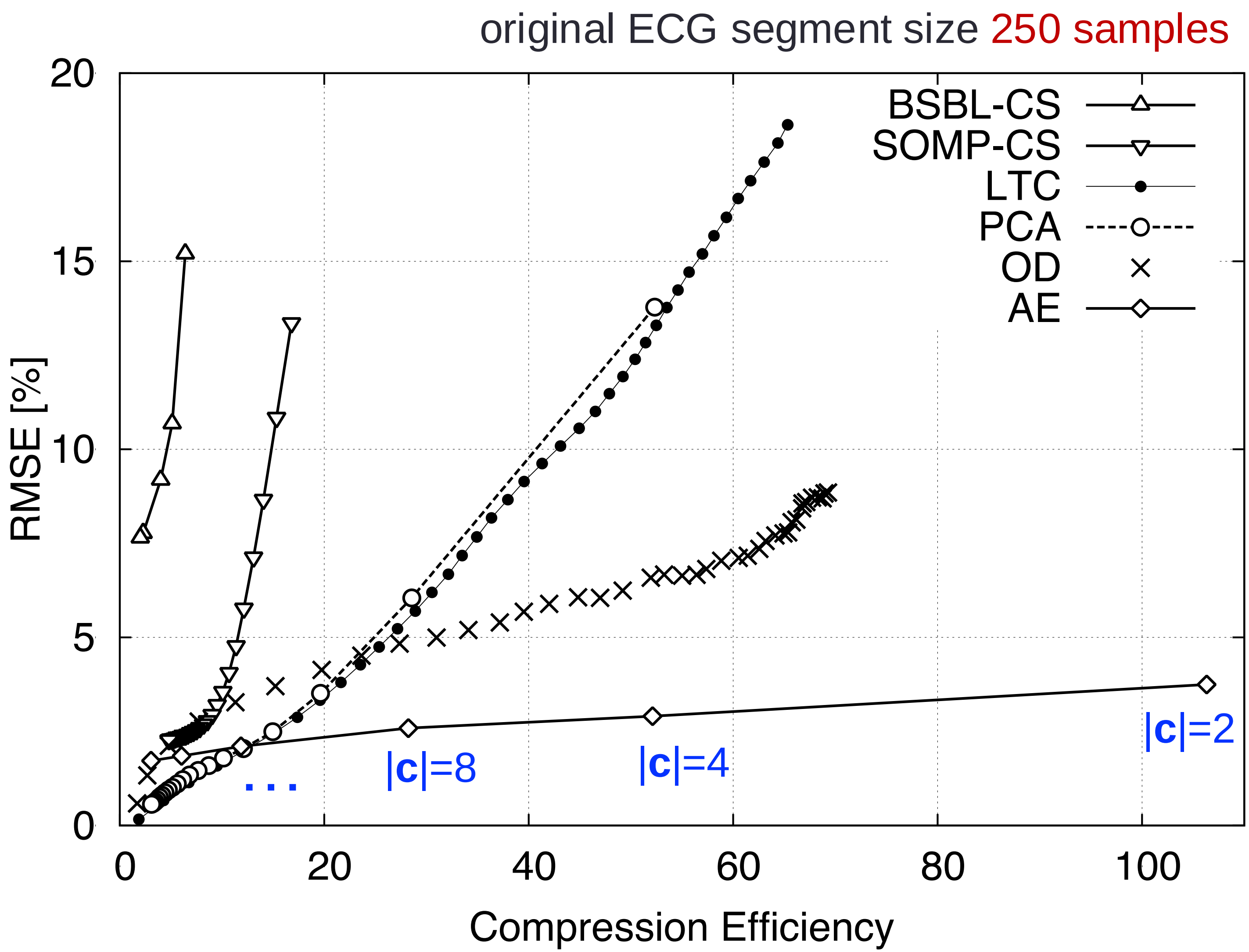
Representation accuracy

- Root Mean Square Error (RMSE)
- Expressed as % of the p2p signal's amplitude

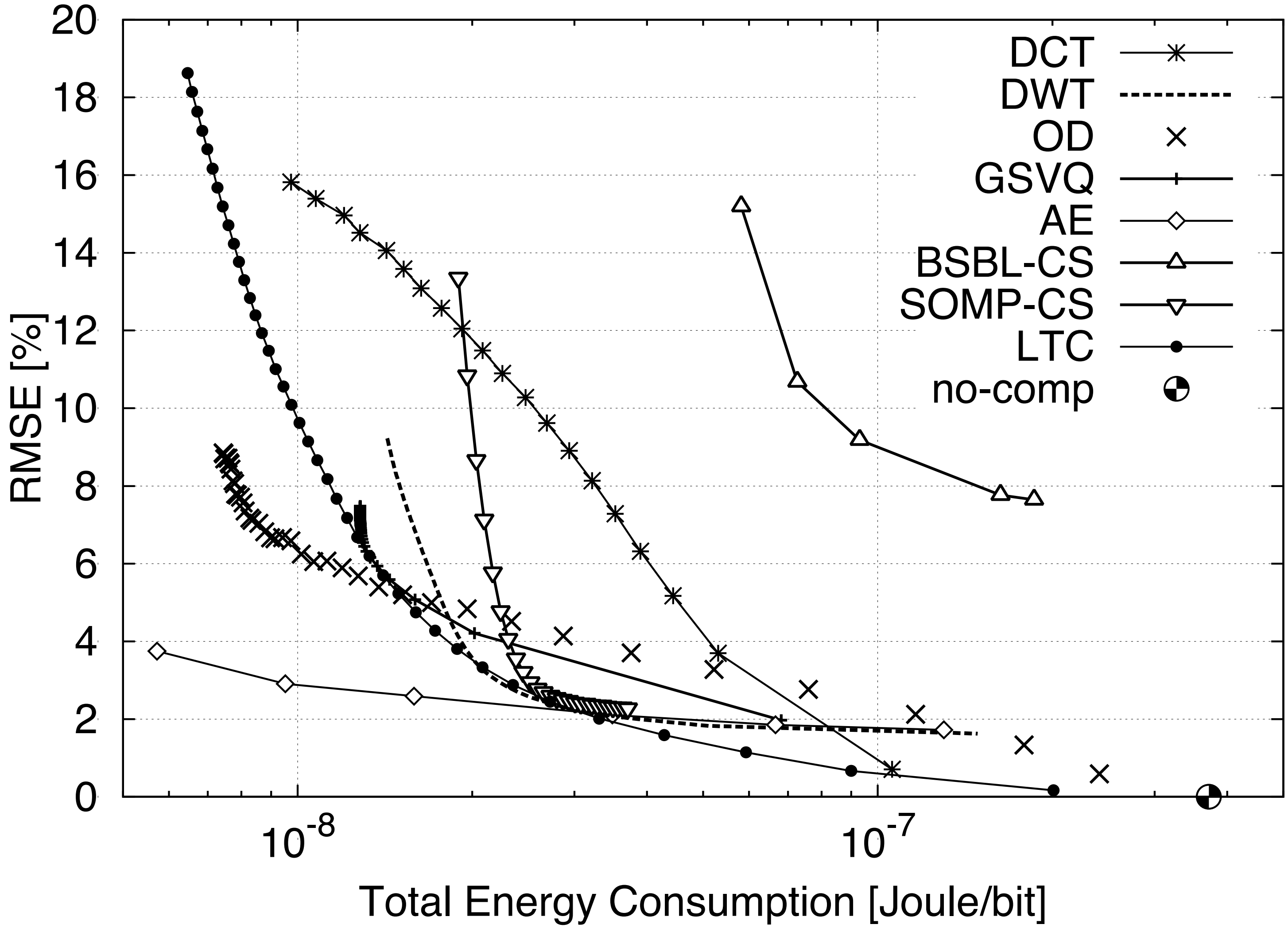
Compression efficiency

$$CE = \frac{\#\text{bits in the original stream}}{\#\text{bits in the compressed stream}}$$

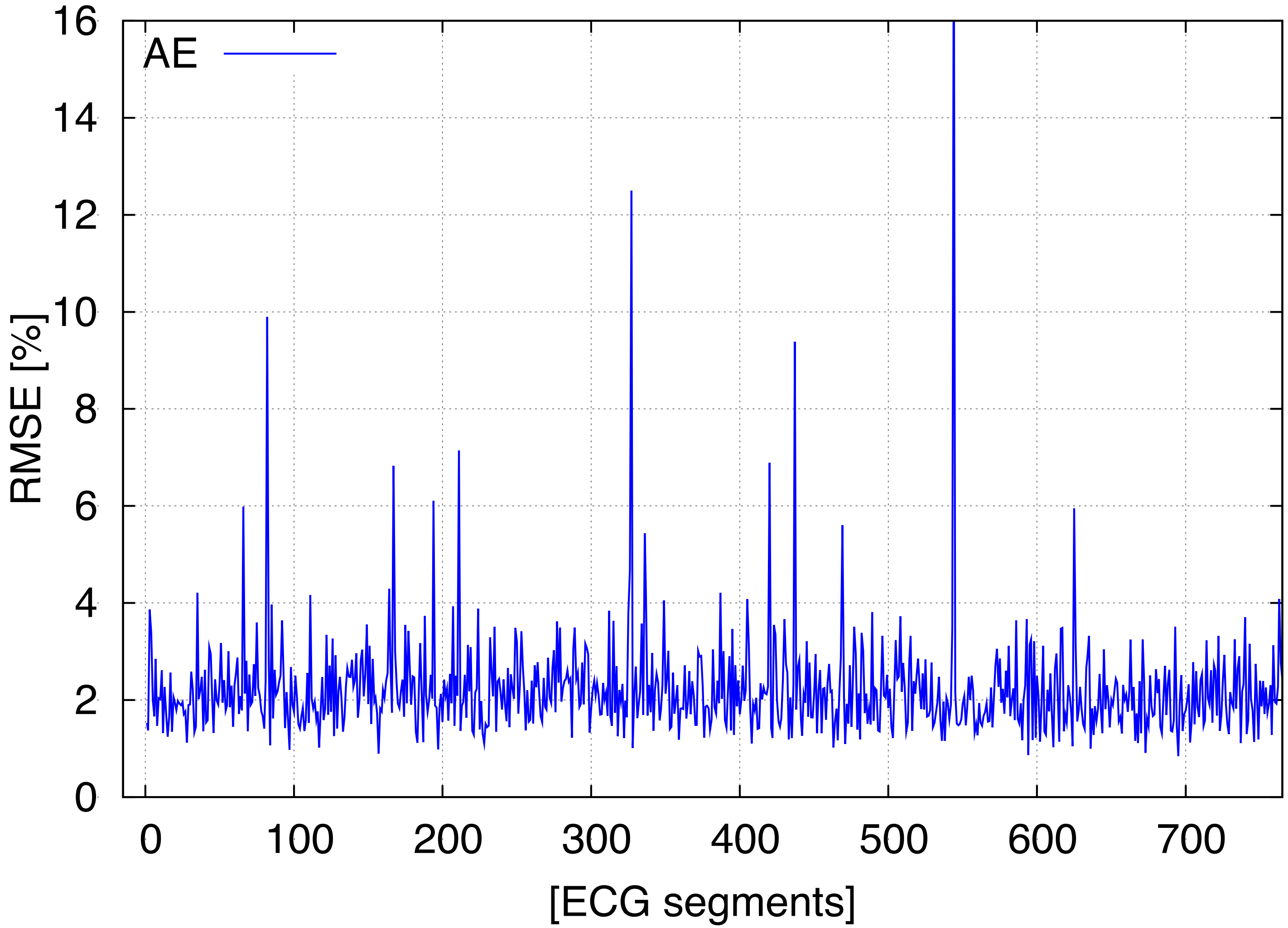
Denoising AE – example results (ECG)



Results: RMSE *vs* Energy



Where AE fails



Bibliography

[Amhed1975] N. Ahmed, K. R. Rao, “Orthogonal transforms for digital signal processing,” Springer, New York Berlin Heidelberg, 1975.

[Baldiand1989] P. Baldiand, K. Hornik,“Neural networks and principal component analysis: Learning from examples without local minima,” Neural Networks, vol. 2(1), pp. 53–58, 1989.

[Hinton06] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” Science, July 2006.

[Vincent04] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” Journal of Machine Learning Research, 2010.

[DelTesta15] D. Del Testa, M. Rossi, ”Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders,” *IEEE Signal Processing Letters*, Vol. 22, No. 12, September 2015.

[Plaut2018] E. Plaut, “From Principal Subspaces to Principal Components with Linear Autoencoders,” Technical Report, arXiv:1804.10253 [stat.ML], [v3] December 2018.

AUTOENCODERS

Michele Rossi

michele.rossi@dei.unipd.it

Dept. of Information Engineering

University of Padova, IT

APPENDIX 1 – WHY SQUARE ERRORS?

Square *vs* absolute errors

- Absolute is often "what we care about"
  - Example: you buy a stock of items to sell them into the future and gain some money from it. If P_{paid} is the price paid to buy the stock and P_{pred} is the future predicted cost, the money you gain is

$$P_{\text{pred}} - P_{\text{paid}}$$

(absolute) and **not** the square difference

  - The same holds for many other practical examples…
- However, the square error has many appealing properties which make its use convenient mathematically
- Let's see some of them…

What does not hold for the absolute error

- If X is a r.v.
  - The estimator of X that minimizes the square norm is the mean E[X], whereas the estimator that minimizes the absolute difference is the median m(X). The mean has much nicer properties than the median, e.g., if Y is also a r.v., it holds E[X+Y] = E[X] + E[Y]
- If **x**=(x_1,x_2) is a vector
  - If you have a vector **x**=$(x_1,x_2)^T$ estimated by **w**=$(w_1,w_2)^T$. For the squared error it does not matter whether you consider the components separately or together, i.e.,

$$\|\boldsymbol{x} - \boldsymbol{w}\|^2 = (x_1 - w_1)^2 + (x_2 - w_2)^2$$

  - You cannot do this with the absolute error. Moreover, this means that the squared error is independent of reparameterization, i.e.,
  - If we define a new vector **y**=$(x1+x2,x1-x2)^T$ the minimum squared deviance estimators for **x** and **y** are the same

What does not hold for the absolute error

- Independent r.v.s. X and Y
  - Variances (expected squared errors) add, i.e.,

$$\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$$

- Differentiability
  - The absolute error is $\mathrm{MAE} \triangleq |x_{\mathrm{pred}} - x_{\mathrm{true}}|$

$$\frac{d\mathrm{MAE}}{dx_{\mathrm{pred}}} = \begin{cases} +1 & x_{\mathrm{pred}} > x_{\mathrm{true}} \\ -1 & x_{\mathrm{pred}} < x_{\mathrm{true}} \\ \text{undefined} & x_{\mathrm{pred}} = x_{\mathrm{true}} \end{cases}$$

  - The squared error is differentiable everywhere, the MAE derivative does not exist in 0, this complicates analysis and numerical computations

Two further and important reasons

- The above are *mathematically convenient reasons*, but there are two more profound “mathematical coincidences” for which the norm-2 is a better choice…
  - When fitting a Gaussian pdf to a dataset, the maximum likelihood fit is the one *minimizing the squared error*, not the abolute error. For a precise account of this result, see, e.g., Chapter 3 “Linear models for regression” of [Bishop2006]
  - When doing dimensionality reduction, finding the basis that minimizes the norm-2 yields PCA. PCA has a natural interpretation in terms of multivariate Gaussian distribution, i.e., finding the axes of the ellipse of the pdf makes. There is a “robust PCA” variant that minimizes the MAE, but it is hard to compute and not very popular…

[Bishop2006] C. Bishop, “Pattern recognition and machine learning,” Springer, 2006.

APPENDIX 2 – INPUT VS OUTPUT VARIANCE OF PCA

Linear AE: property P1

Property P1: We **prove** that the covariance of the output vectors $\mathbf{y}_k$ is the best p-rank approximation of the covariance of the input vectors $\mathbf{x}_k$, k=1,2,…,n

- Since (zero mean data)

$$\boldsymbol{X}' = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

- For the covariance of the input vectors, it holds

$$n\mathrm{Cov}(\boldsymbol{X}') = \boldsymbol{X}'(\boldsymbol{X}')^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T$$

- For the output covariance (from def. of covariance)

$$n\mathrm{Cov}(\boldsymbol{Y}') = (\boldsymbol{Y} - \overline{\boldsymbol{y}}\mathbf{1}^T)(\boldsymbol{Y} - \overline{\boldsymbol{y}}\mathbf{1}^T)^T =$$

$$= (\boldsymbol{Y} - \overline{\boldsymbol{y}}\mathbf{1}^T)(\boldsymbol{Y}^T - \mathbf{1}\overline{\boldsymbol{y}}^T) = ?$$

Proof of Property P1

- From (24)

$$\boldsymbol{Y} = \boldsymbol{W}_2\boldsymbol{C} + \left(\overline{\boldsymbol{x}} - \frac{\boldsymbol{W}_2\boldsymbol{C}\mathbf{1}}{n}\right)\mathbf{1}^T$$

- Using $\overline{\boldsymbol{y}} = \overline{\boldsymbol{x}}$, we have

$$\boldsymbol{Y} - \overline{\boldsymbol{y}}\mathbf{1} = \boldsymbol{Y} - \overline{\boldsymbol{x}}\mathbf{1} = \boldsymbol{W}_2\boldsymbol{C} - \frac{\boldsymbol{W}_2\boldsymbol{C}\mathbf{1}\mathbf{1}^T}{n} = \boldsymbol{W}_2\boldsymbol{C}'$$

- Using

$$\boldsymbol{W}_2\boldsymbol{C}' = \boldsymbol{U}\boldsymbol{\Sigma}_p\boldsymbol{V}^T$$

$$\begin{aligned}\mathrm{Cov}(\boldsymbol{Y}) &= (\boldsymbol{Y} - \overline{\boldsymbol{y}}\mathbf{1}^T)(\boldsymbol{Y} - \overline{\boldsymbol{y}}\mathbf{1}^T)^T = \boldsymbol{W}_2\boldsymbol{C}'(\boldsymbol{W}_2\boldsymbol{C}')^T \\ &= \boldsymbol{U}\boldsymbol{\Sigma}_p\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}_p^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Sigma}_p^2\boldsymbol{U}^T\end{aligned}$$

QED

Linear AE is equivalent to PCA

- From property P1, this means that the linear autoencoder is an indirect way of performing the Karhunen-Loève transform (PCA) on zero average data [Ahmed1975]
- Hence, the linear autoencoder applies PCA to the input data in the sense that its code **c** is a projection of the data into the low-dimensional principal subspace
- However, unlike PCA, the coordinates of **c** are correlated and not necessarily sorted in descending order of variance (eigenvalues)

[Amhed1975] N. Ahmed, K. R. Rao, "Orthogonal transforms for digital signal processing," Springer, New York Berlin Heidelberg, 1975.