Discussion 5

EE599: Deep Learning

Keith M. Chugg

Spring 2020

USC University of Southern California

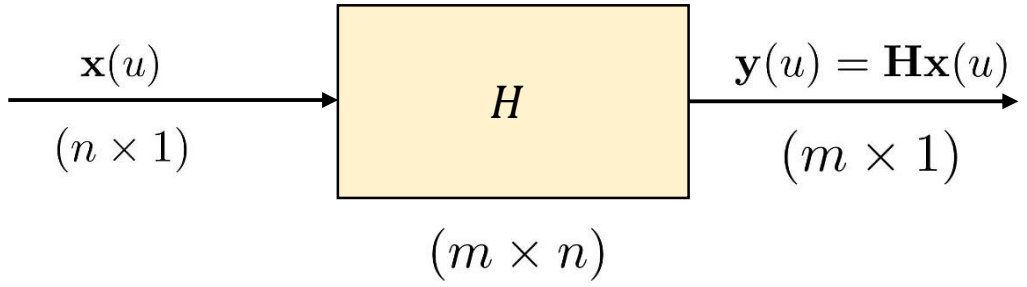
Contributors: Olaoluwa Adigun

Outline

- Principal Component Analysis

- Filter Design

Random Vectors



$$\mathbf{m}_y = \mathbf{H}\mathbf{m}_x$$

$$\mathbf{R}_y = \mathbf{H}\mathbf{R}_x\mathbf{H}^T$$

$$\mathbf{K}_y = \mathbf{H}\mathbf{K}_x\mathbf{H}^T$$

Correlation

$$\begin{aligned}
\mathbf{R}_y &= \mathbb{E}\left[\mathbf{y}(u)\mathbf{y}^T(u)\right] \\
&= \mathbb{E}\left[\mathbf{H}(\mathbf{x}(u))(\mathbf{H}\mathbf{x}(u))^T\right] \\
&= \mathbb{E}\left[\mathbf{H}\mathbf{x}(u)\mathbf{x}(u)^T\mathbf{H}^T\right] \\
&= \mathbf{H}\mathbb{E}\left[\mathbf{x}(u)\mathbf{x}(u)^T\right]\mathbf{H}^T \\
&= \mathbf{H}\mathbf{R}_x\mathbf{H}^T
\end{aligned}$$

Special case

$$y(u) = \mathbf{b}^T\mathbf{x}(u) \qquad (1 \times 1)$$

$$m_y = \mathbf{b}^T\mathbf{m}_x$$

$$\mathbb{E}\left[y(u)^2\right] = \mathbf{b}^T\mathbf{R}_x\mathbf{b}$$

$$\sigma_y^2 = \mathbf{b}^T\mathbf{K}_x\mathbf{b}$$

Note that covariance / correlation matrices are symmetric, non-negative definite.

Karhunen-Loéve (KL) Expansion

Can always find orthonormal set of eigen-vectors of **K**.

These are an alternate coordinate systems (rotations, reflections)

in this eigen-coordinate system, the components are uncorrelated

(principle components)

Eigen-values are the variant (energy) in each of these principle directions

(can be used to reduce dimensions by throwing out components with low energy)

KL-Expansion

$$\mathbf{K}_x \mathbf{e}_k = \lambda_k \mathbf{e}_k \qquad k = 0, 1, ..., N-1 \qquad \text{(Eigen equation)}$$

$$\mathbf{e}_k^T \mathbf{e}_l = \delta[k-l] \qquad \lambda_k \geq 0 \qquad \text{(Orthonormal eigen-vectors)}$$

$$\mathbf{x}(u) = \sum_{k=0}^{N-1} X_k(u) \mathbf{e}_k \qquad \text{(Change of coordinates)}$$

$$X_k(u) = \mathbf{e}_k^T \mathbf{x}(u)$$

$$\mathbb{E}\big[X_k(u) X_l(u)\big] = \mathbf{e}_k^T \mathbf{K}_x \mathbf{e}_l = \lambda_k \delta(k-l) \qquad \text{(Uncorrelated components)}$$

$$\mathbf{K}_x = \sum_{k=0}^{N-1} \lambda_k \mathbf{e}_k \mathbf{e}_l^T \qquad \text{(Mercer's Theorem)}$$

$$\mathbb{E}\big[||\mathbf{x}(u)||^2\big] = tr(\mathbf{K}_x) = \sum_{k=0}^{N-1} \lambda_k \qquad \text{(Total Energy)}$$

Always exists because **K** is positive semi-definite (PSD) and symmetric

KL-Expansion

$$d_k(u) = \mathbf{e}_k^T \mathbf{x}(u) \qquad k = 0, 1, ..., D-1$$

$$\mathbf{d}(u) = \mathbf{E}^T \mathbf{x}(u)$$

$$\mathbf{K}_d = \mathbf{E}^T \mathbf{K}_x \mathbf{E}$$

$$= \mathbf{E}^T (\mathbf{E} \mathbf{\Lambda} \mathbf{E}^T) \mathbf{E}$$

$$= \mathbf{\Lambda} = \mathbf{diag}(\lambda_k)$$

Multiplying by $\mathbf{E}^T$ makes the component uncorrelated

$$\mathbf{E} = \left[\mathbf{e}_0 \mid \mathbf{e}_1 \mid \mathbf{e}_2 \mid ... \mid \mathbf{e}_{D-1} \right]$$

KL-Expansion: Relation to Whitening

$$w_k(u) = \frac{X_k(u)}{\sqrt{\lambda_k}} = \frac{\mathbf{e}_k^T \mathbf{x}(u)}{\sqrt{\lambda_k}} \qquad k = 0, 1, ..., D-1$$

$$\begin{aligned}
\mathbf{w}(u) &= \mathbf{\Lambda}^{-1/2}\mathbf{E}^T\mathbf{x}(u) \\
\mathbf{K}_w &= \mathbf{\Lambda}^{-1/2}\mathbf{E}^T\mathbf{K}_x\mathbf{E}\mathbf{\Lambda}^{-1/2} \\
&= \mathbf{\Lambda}^{-1/2}\mathbf{\Lambda}\mathbf{\Lambda}^{-1/2} \\
&= \mathbf{I}
\end{aligned}$$

For any orthogonal matrix **U**, this whitening matrix also works:

$$\mathbf{G} = \mathbf{U}\mathbf{\Lambda}^{-1/2}\mathbf{E}^T$$

KL- Expansion: Relation to PCA

$$\tilde{x}_k(u) = \mathbf{e}_k^K \mathbf{x}(u) \qquad k = 0, 1, ..., K-1$$

$$\tilde{\mathbf{x}}(u) = \mathbf{E}_{[:K]} \mathbf{x}(u) \qquad \text{First } K \text{ componenets}$$

$$\mathbf{K}_{\tilde{\mathbf{x}}} = \mathbf{\Lambda}_{[:K]} \qquad \text{Assumes ordered eigen-values: } \lambda_0, \lambda_1, ..., \lambda_{D-1}$$

$$\mathbb{E}\left[||\tilde{\mathbf{x}}(u)||^2\right] = \sum_{k=0}^{K-1} \lambda_k$$

$$\mathbb{E}\left[||\mathbf{x}(u) - \tilde{\mathbf{x}}(u)||^2\right] = \sum_{k=K}^{D-1} \lambda_k \qquad \text{minimizes approximation error (lossy compression)}$$

PCA is simply taking only the ***K*** most important Eigen-directions or principal components

$$\mathbf{E}_{[:K]} = \left[\mathbf{e}_0 \mid \mathbf{e}_1 \mid \mathbf{e}_2 \mid ... \mid \mathbf{e}_{K-1} \right]$$

KL- Expansion / PCA for Data

Everything is the same, except that we use the data averaging instead of the expectation $\mathbb{E}[\cdot]$

$$\begin{aligned}\hat{\mathbf{R}}_x &= \left\langle \mathbf{x}\mathbf{x}^T \right\rangle_{\mathcal{D}} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{x}_n \mathbf{x}_n^T \\ &= \frac{1}{N}\mathbf{X}^T\mathbf{X}\end{aligned}$$

Both KL and PCA can be applied to **R** or **K**. Center x if you want to use **K**

$$\boldsymbol{x \leftarrow x - m}$$

(same if mean is zero)

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0^T \\ \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_{N-1}^T \end{bmatrix} \qquad \mathbf{X}^T = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 \ldots & \mathbf{x}_{N-1} \end{bmatrix}$$

"stacked" data matrix

KL- Expansion / PCA for Data

PCA for data

$$\tilde{\mathbf{x}}_n = \mathbf{E}_{[:K]}^T \mathbf{x}_n \qquad \text{First } K \text{ components} \qquad \mathbf{E}_{[:K]} = \left[\mathbf{e}_0 \mid \mathbf{e}_1 \mid \mathbf{e}_2 \mid \ldots \mid \mathbf{e}_{K-1} \right]$$

Apply to the "stacked" data matrix

$$\tilde{\mathbf{X}} = \begin{bmatrix} (\mathbf{E}_{[:K]}^T \mathbf{x}_0)^T \\ (\mathbf{E}_{[:K]}^T \mathbf{x}_1)^T \\ \vdots \\ (\mathbf{E}_{[:K]}^T \mathbf{x}_{N-1})^T \end{bmatrix} = \mathbf{X}\mathbf{E}_{[:K]}$$

$$\underset{N \times K}{\tilde{\mathbf{X}}} = \underset{N \times D}{\mathbf{X}} \quad \underset{D \times K}{\mathbf{E}_{[:K]}}$$

Dimension reduced from D to K

$$\tilde{\mathbf{X}}^T = \left[\mathbf{E}_{[:K]}^T \mathbf{x}_0 \;\; \mathbf{E}_{[:K]}^T \mathbf{x}_1 \ldots \;\; \mathbf{E}_{[:K]}^T \mathbf{x}_{N-1} \right] = \mathbf{E}_{[:K]}^T \mathbf{X}^T$$

PCA reduces the number of features from D to K

KL / PCA for Data: Singular Value Decomposition (SVD)

SVD for an arbitrary matrix $\mathbf{A}$

$$\underset{m\times n}{\mathbf{A}} = \underset{m\times m}{\mathbf{U}}\ \underset{m\times n}{\mathbf{\Sigma}}\ \underset{n\times n}{\mathbf{V}^T}$$

The **SVD** for $\mathbf{A}$ provides the KL factorization for the non-negative definition, symmetric $\mathbf{A^T A}$

Use **SVD** to expand matrix $\mathbf{A^T A}$

$$\begin{aligned}\mathbf{A}^T\mathbf{A} &= (\mathbf{U\Sigma V})^T\mathbf{U\Sigma V}\\ &= \mathbf{V\Sigma}^T\mathbf{U}^T\mathbf{U\Sigma V}\\ &= \underset{n\times n}{\mathbf{V}}\ \underset{n\times n}{\mathbf{\Sigma\Sigma}^T}\ \underset{n\times n}{\mathbf{V}^T}\\ &= \mathbf{E\Lambda E}^T\end{aligned}$$

U and **V** orthogonal matrices, **Σ** is the diagonal with singular values on diagonal

Note that this is also the **SVD** for $\mathbf{A^T A}$

KL-Expansion / PCA for Data: Singular Value Decomposition (SVD)

SVD for a stacked data matrix **X**

$$\underset{N\times D}{\mathbf{X}} = \underset{N\times N}{\mathbf{U}} \quad \underset{N\times D}{\mathbf{\Sigma}} \quad \underset{D\times D}{\mathbf{V}^T}$$

$$\underset{D\times D}{\mathbf{X}^T\mathbf{X}} = \underset{D\times D}{\mathbf{V}} \quad \underset{D\times D}{\mathbf{\Sigma}\mathbf{\Sigma}^T} \quad \underset{D\times D}{\mathbf{V}^T}$$

$$= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$$

Equivalent approaches:

1. Find the **SVD** of $\mathbf{X}$. Take $\mathbf{V}$
2. Find the Eigen decomposition of $\mathbf{X^T X}$. Take $\mathbf{E} = \mathbf{V}$
3. Find the **SVD** of $\mathbf{X^T X}$. Take $\mathbf{V} = \mathbf{U} = \mathbf{E}$

$$\underset{N\times K}{\tilde{\mathbf{X}}} = \underset{N\times D}{\mathbf{X}} \quad \underset{D\times K}{\mathbf{V}_{[:K]}}$$

KL-Expansion / PCA for Data: Singular Value Decomposition (SVD)

Equivalent approaches:

1. Find the **SVD** of $\mathbf{X}$. Take $\mathbf{V}$
2. Find the Eigen decomposition of $\mathbf{X}^\mathbf{T}\mathbf{X}$. Take $\mathbf{E} = \mathbf{V}$
3. Find the **SVD** of $\mathbf{X}^\mathbf{T}\mathbf{X}$. Take $\mathbf{V} = \mathbf{U} = \mathbf{E}$

May want to use method 3, with numpy.linalg.svd, instead of method 2, with numpy.linalg.eig, since the **SVD** returns the Eigen-vectors in sorted order and eig does not.

$$\underset{N \times K}{\tilde{\mathbf{X}}} = \underset{N \times D}{\mathbf{X}} \quad \underset{D \times K}{\mathbf{V}_{[:K]}}$$

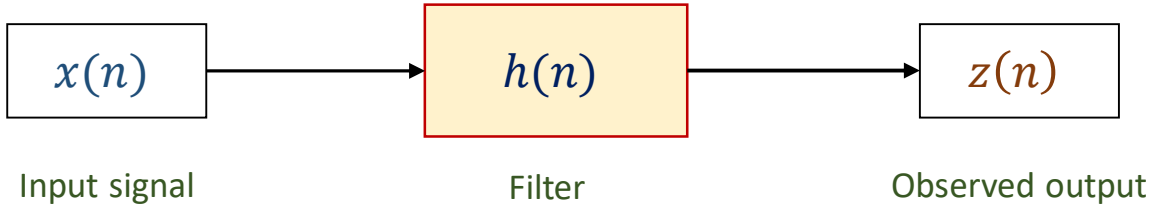
Outline

- Principal Component Analysis

- Filter Design

Filter Design

This involves creating a signal processing system that satisfies a pre-defined set of requirements.



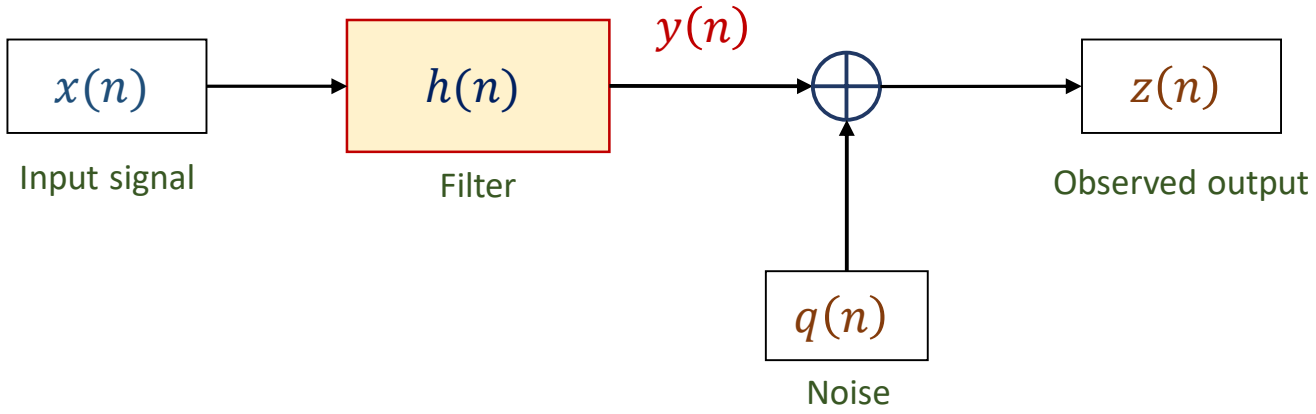
Example of Design requirements

- Frequency Response : *Low-pass, high-pass, band-pass, or band-stop*
- Impulse Response : *Finite (FIR) or infinite (IIR)*
- Causality : *Causal or non-causal*
- Phase and group delay

The goal is to find $\boldsymbol{h(n)}$ that minimizes the error between the observed output $\boldsymbol{z(n)}$ and the expected output $\boldsymbol{y(n)}$

Homework 2: Filter Design for Additive Noise

Input signal $x(n)$ goes through a linear system and is observed in additive Gaussian noise.



$$z(n) = y(n) + q(n)$$

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2)$$

We need a system that minimize the mean squared error between $\boldsymbol{y(n)}$ and $\boldsymbol{z(n)}$

Wiener Filter

This is a *linear* and *time invariant* filter that separates original signal $y_n(u)$ from a corrupted signal.

Assumptions:

- Original signal and (additive) noise are stationary linear stochastic processes.
- Known spectral characteristics or known autocorrelation and cross-correlation.
- It is a minimum mean-square error (MMSE) estimator.

$$\hat{z}_n(u) = w_{lmmse}^T \, x_n(u)$$

$$E(w) = \mathbb{E}[\|y_n(u) - w^T x_n(u)\|^2]$$

$$w_{lmmse} = \arg max_w \; E(w) = R_x^{-1} \, r_{xz}$$

Homework 2: Wiener Filter

$$w_{lmmse} = \arg max_w \; E(w) = R_v^{-1} \, r_{vz} \qquad (1)$$

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2)$$

$$E(w) = \mathbb{E}[\|y_n(u) - \boldsymbol{w}^T x_n(u)\|^2]$$

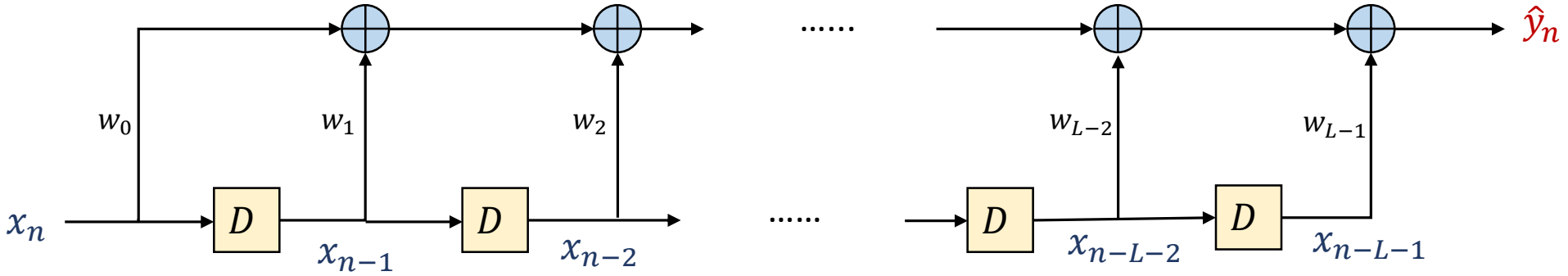
$$v_n(u) = [x_n(u), x_{n-1}(u), x_{n-1}(u)]^T$$

$$z(n) = y(n) + q(n)$$

- Show that : $r_{xz}[m] = \mathbb{E}[x_n(u) x_{n+m}(u)] = h_m$
- Compute auto-correlation R_{v_n}: $R_{v_n} = \mathbb{E}[v_n(u) v_n^T(u)]$
- Compute cross-correlation r_{vz} : $r_{vz} = \mathbb{E}[v_n(u) z_n(u)]$
- Compute MMSE : $E(w) = \mathbb{E}[\|y_n(u) - w^T x_n(u)\|^2]$

Least Mean Squares (LMS) Algorithm

This is a method for designing adaptive filters (finding the coefficients) that minimizes the mean square error.



$$\boldsymbol{v_n} = [x_n, x_{n-1}, \dots, x_{n-L-2}, x_{n-L-1}]^T$$

$$\hat{y}_n = \sum_{l=0}^{L-1} w_l x_{n-l} = \boldsymbol{w}^T \boldsymbol{v_n}$$

$$E(w) = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{y}^{(i)} - \boldsymbol{w}^T \boldsymbol{v}^{(i)} \right\|^2$$

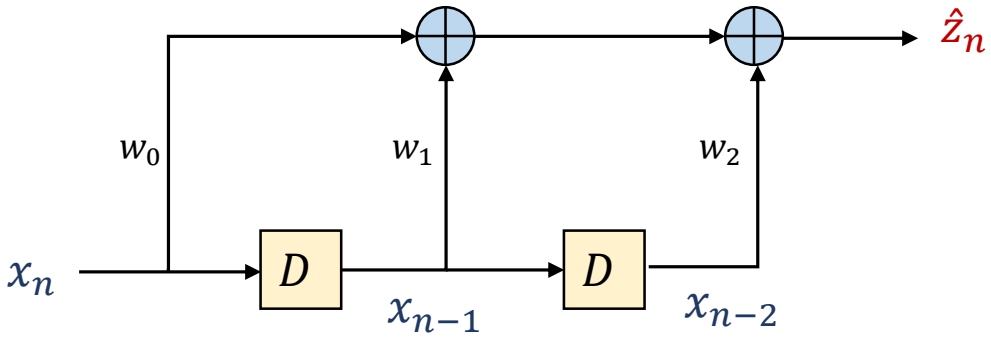
- Filter is adaptive
- It uses stochastic gradient descent (update rule)
- We need real samples $\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$ to use the LMS algorithm.

Update rule with single point stochastic gradient descent

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta \nabla_{\boldsymbol{w}} E(w) = \boldsymbol{w}_n + \eta \left(\boldsymbol{y}^{(i)} - \boldsymbol{w}_n^T \boldsymbol{v}^{(i)} \right) \boldsymbol{v}^{(i)}$$

Homework 2 : LMS Algorithm

- Dataset : $\{\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\}_{i=1}^{N}$
- Number of sequence N: 600
- Length of sequence : $n \in \{0,1,2,\ldots,500\}$
- Vary the learning rate η.



$$\boldsymbol{v_n} = [x_n, x_{n-1}, x_{n-2}]^T$$

$$\hat{z}_n = \sum_{l=0}^{2} w_l x_{n-l} = \boldsymbol{w}^T \boldsymbol{v_n}$$

$$E(w) = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{z}^{(i)} - \boldsymbol{w}^T \boldsymbol{v}^{(i)} \right\|^2$$

Update rule with single point stochastic gradient descent

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta \nabla_{\boldsymbol{w}} E(w) = \boldsymbol{w}_n + \eta \left(\boldsymbol{y}^{(i)} - \boldsymbol{w}_n^T \boldsymbol{v}^{(i)} \right) \boldsymbol{v}^{(i)}$$

Code for the LMS algorithm and PCA !!!