SM2001 Study Problems

1. ① Def full SVD; ② Expression of econ SVD, ③ Labeling size, name

Given a Dataset $\underset{\sim}{X} \in \mathbb{R}^{n \times m}$

$\underset{\sim}{X} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix}$ snapshots, $x_i \in \mathbb{R}^{n \times 1}$

- Definition of SVD: A unique matrix decomposition exists for every complex-valued Matrix $\underset{\sim}{X} \in \mathbb{C}^{n \times m}$

- Full SVD:

$$U \Sigma V^* = X$$

- $\underset{\sim}{U} \in \mathbb{C}^{n \times n}$, $V \in \mathbb{C}^{m \times m}$, $\begin{cases} \underset{\sim}{U} \cdot \underset{\sim}{U}^* = 1 \\ \underset{\sim}{V} \cdot \underset{\sim}{V}^* = 1 \end{cases}$

unitary matrix with orthonormal columns. Right columns of $\underset{\sim}{U}$: left singular vectors, V: Right singular vectors

- $\underset{\sim}{\Sigma} \in \mathbb{R}^{n \times m}$

Real, non-negative Entries on digonal and zeros off diag

Ordered from Largest → Smallest

Diagonal Value: Singular Value

- Economic SVD: when $n \gg m$, $\underset{\sim}{\Sigma}$ has most in non-zero elements on digonal $\hat{\underset{\sim}{\Sigma}}$

Therefore: $\underset{\sim}{X} = \underset{\sim}{U} \underset{\sim}{\Sigma} \underset{\sim}{V}^* = \begin{bmatrix} \hat{\underset{\sim}{U}} & \hat{\underset{\sim}{U}}^{\perp} \end{bmatrix} \begin{bmatrix} \hat{\underset{\sim}{\Sigma}} \\ 0 \end{bmatrix} V^*$

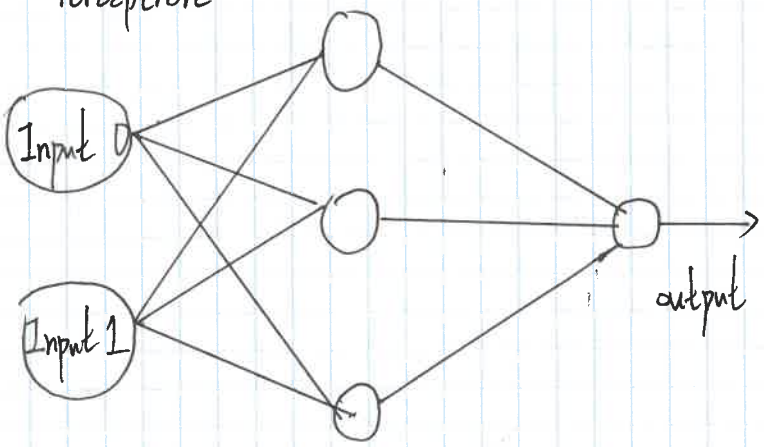
$\underset{\sim}{X} = \hat{\underset{\sim}{U}} \cdot \hat{\underset{\sim}{\Sigma}} \cdot \underset{\sim}{V}^*$ $\hat{\underset{\sim}{U}}$ and $\hat{\underset{\sim}{U}}^{\perp}$ are complementary

2. Schematic Representation of MLP, LSTM

Show Inputs & Outputs, strengths & weakness.

2-1 MLP: Multi-Layer Perceptron

• Schematic:



• Strengths. 1. Good performance of linear regression/ classification

Gives a good prediction of point data (no time information)

2. Easier to interpret and understand Algorithm

• Weakness: 1. Can not treat data as sequence. Can not predict data in time series.

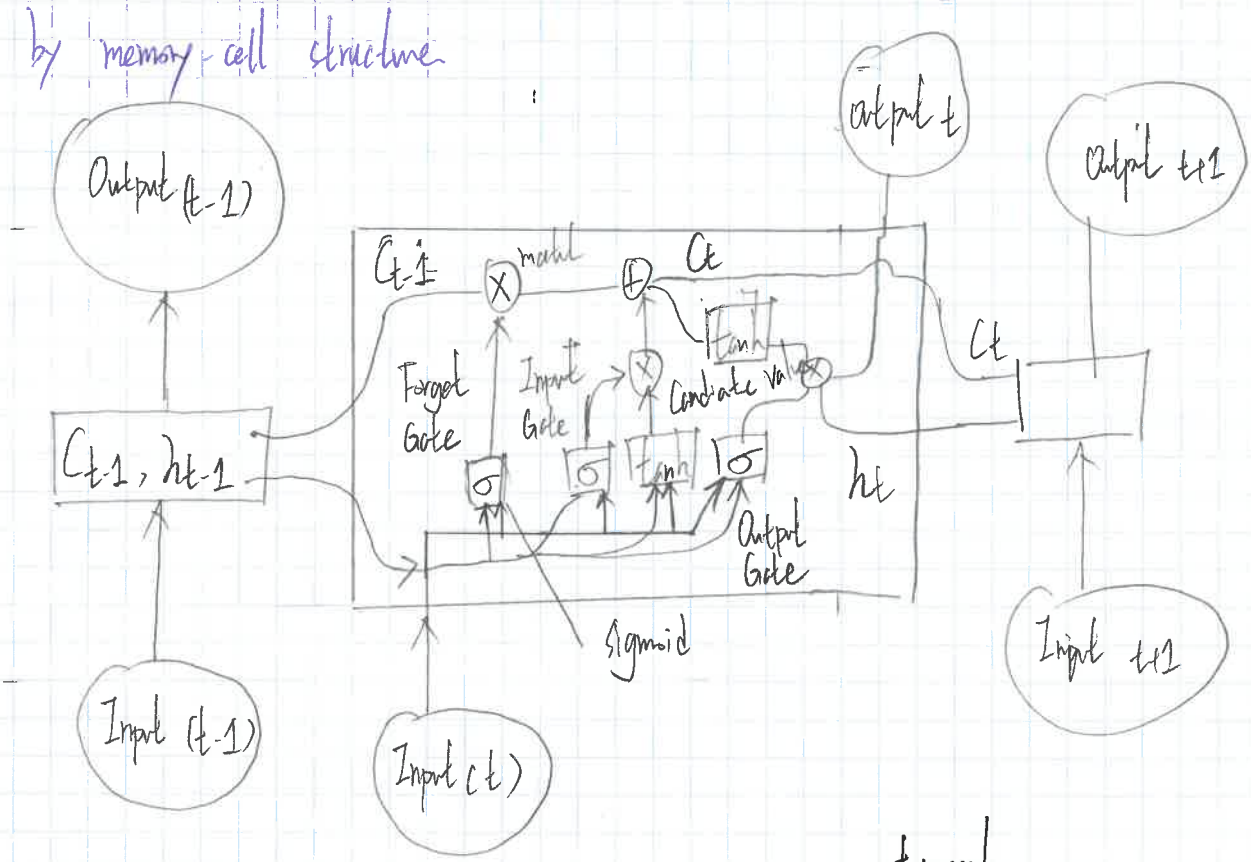
2. Not able to get spatial information between data point. eg Image Data, thus it's diffiult to use MLP deal with high-dimensial data.

2.2 LSTM: Long Short term Memory

LSTM is a kind of Recurrent Neural Network (RNN)

RNN can deal with sequence by iterating through sequence elements and maintaining a state that contains information about what it has "seen" so far. Map a sequence of data points to another sequence

Compared with simple RNN LSTM solved Gradient Vanish Problem by memory-cell structure

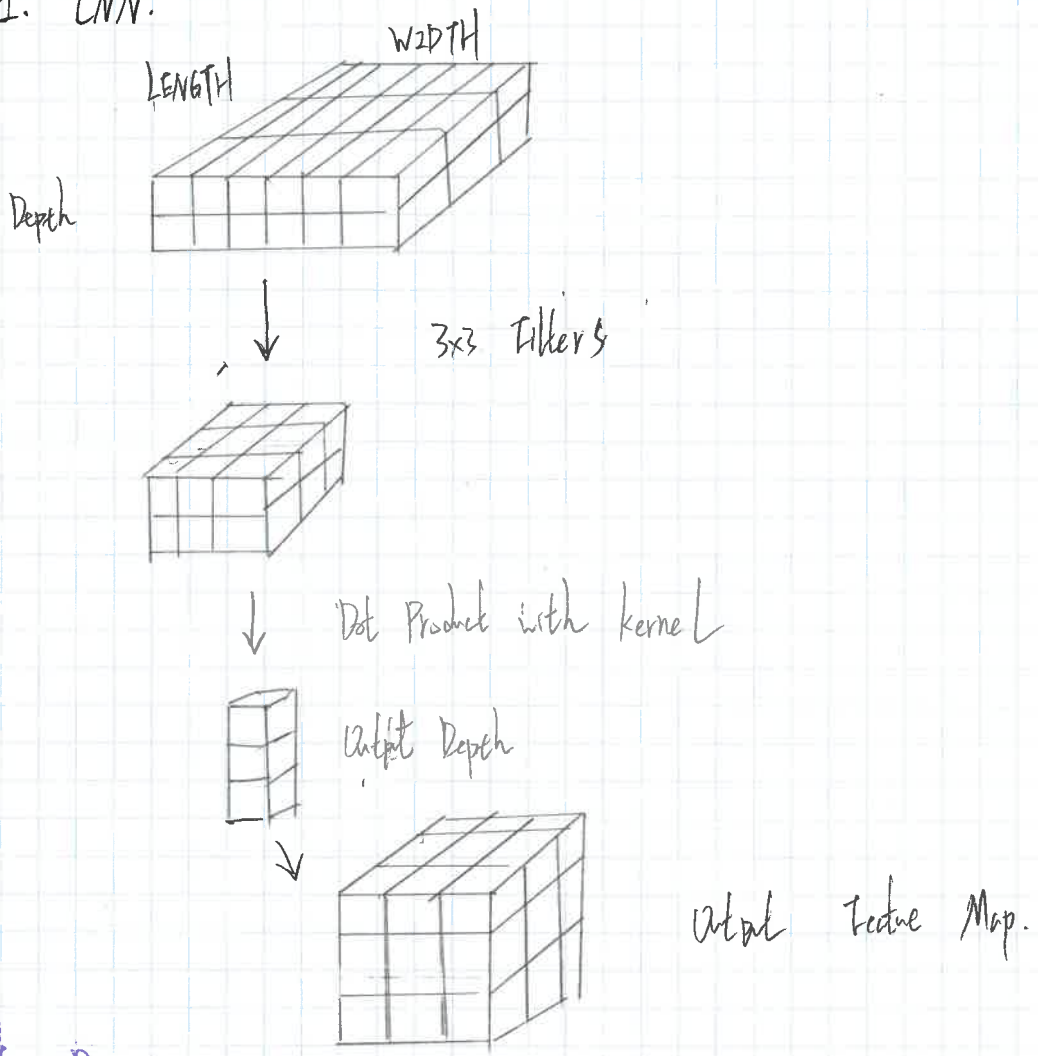


Strength:
- Treat data as sequential, thus avoiding temporal information loss
  - ⇒ it will be useful when predicting time-series data.
- Also, since it treats data as sequence, bi-direction LSTM is performs well in NLP

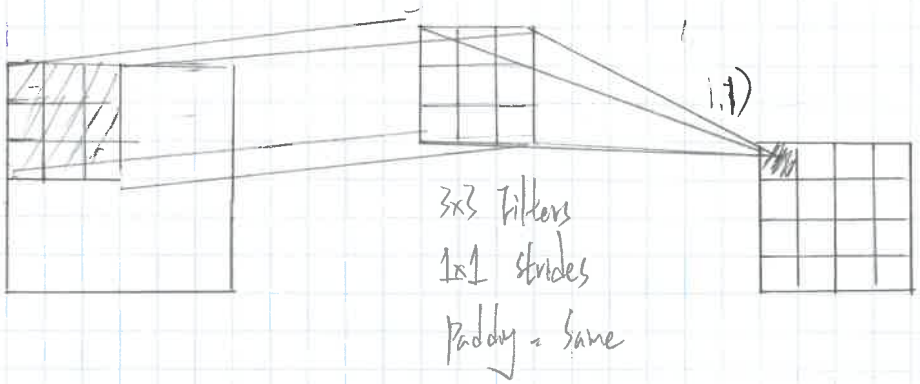
Weakness:
1. Easier to Overfitting. Need dropout (which built-in in tensorflow)
2. At tensorflow, LSTM compute via CPU instead of GPU.

3. Schematic a LSTM, a CNN, an AE, List Application

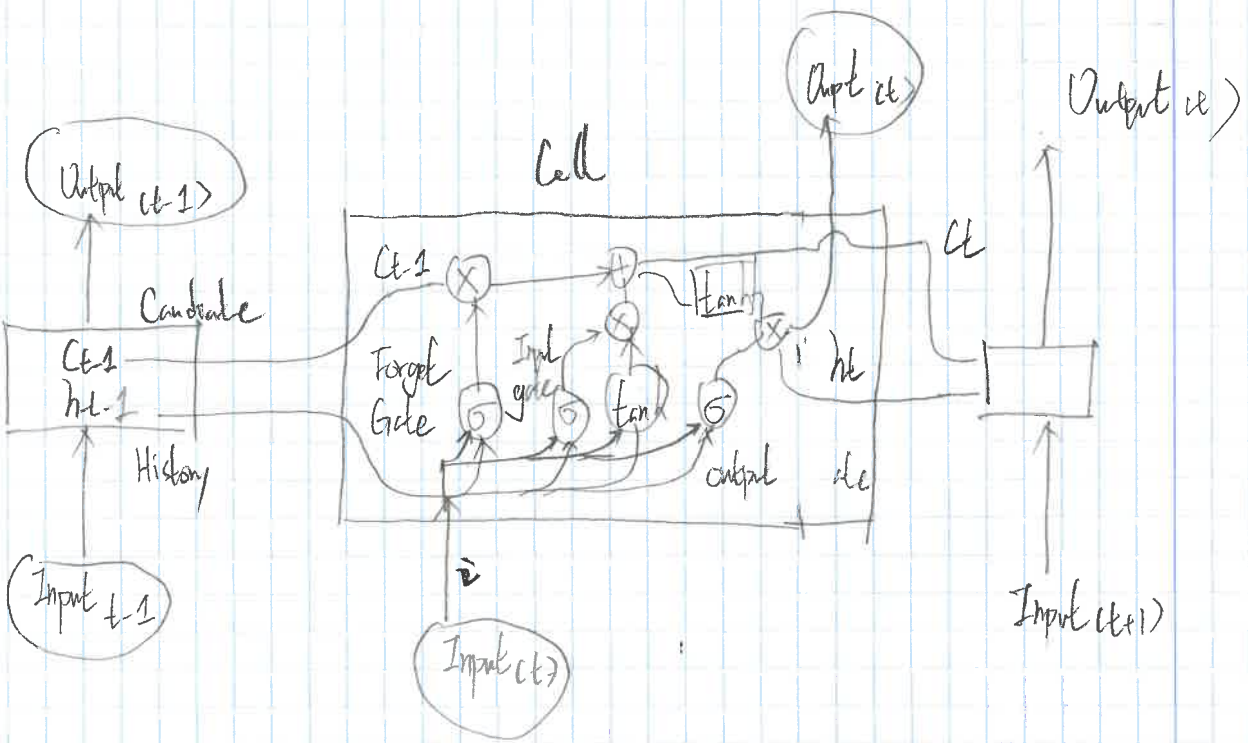
3.1. CNN.



In 2D:

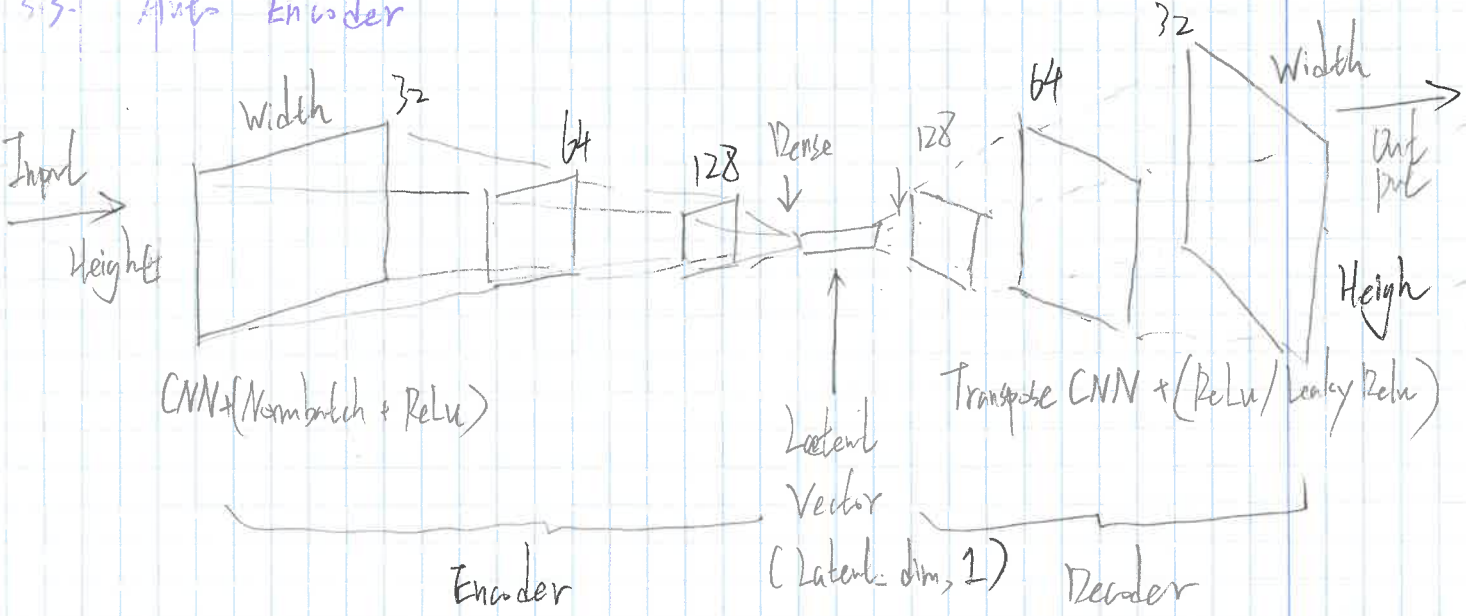


3.2. LSTM



Time series prediction Weather Prediction

3.3. Auto Encoder



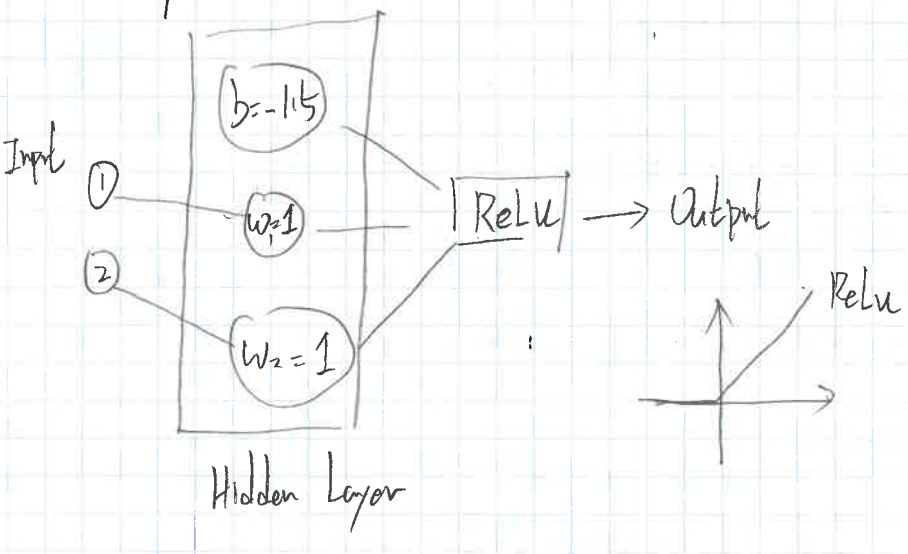
Mode decomposition and reconstruction

4. A neuron, 2 inputs, 1 output, Activation = "ReLu"

weights $w_1 = 1$, $w_2 = 1$, Bias $b = -1.5$

If Input = (0,0), (1,0), (0,1), (1,1)

Output ?



(a) Input (0,0) → ReLu(0×1 + 0×1 + -1.5) = 0

(b) (1,0) ReLu(1×1 + 0×1 - 1.5) = 0

(c) (0,1) ReLu(0×1 + 1×1 - 1.5) = 0

d (1,1) ReLu(1 + 1 - 1.5) = 0.5

5. Consider Loss $E(\underline{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x,\underline{w}) - t_n\}^2$

• y : prediction, $\underline{w}$: weights, t_n : targets

• For Prediction $y(x,\underline{w}) = w_0 + w_1 x + w_2 x^2 \cdots = \sum_{j=0}^{M} w_j x^j$

Show that: $\underline{w} = \{w_i\}$ minimize loss function $E(\underline{w})$ are given by the solutions to the following Linear functions

$$\boxed{\sum_{j=0}^{M} A_{ij} w_j = T_i} \quad (1)$$

where $A_{ij} = \sum_{n=1}^{N} x_n^{i+j}$, $T_i = \sum_{n=1}^{N} x_n^i t_n$ (2)

i, j denote index of component

At iteration i snapshot of X^i is $x_n^i = [x_1^{i,0}, x_1^{i,1} \cdots x_1^{i,M}]$

$$X^i = \begin{bmatrix} - x_1^i - \\ - x_n^i - \\ - x_N^i - \end{bmatrix} \quad \updownarrow N \qquad \leftrightarrow M$$

$X^i \in R^{N \times M}$

For this in (1), $LHS = \sum_{j=0}^{M} \sum_{n=1}^{N} x_n^{i+j} \cdot w_j$

For A_{ij}, it means at certain iter, a certain x_n^i summed through N

$$X^i = \left[\begin{bmatrix} x_1^{i,0} \\ \vdots \\ x_N^{i,0} \end{bmatrix} \cdots \begin{bmatrix} x_1^{i,j} \\ \vdots \\ x_N^{i,j} \end{bmatrix} \cdots \begin{bmatrix} x_1^{i,M} \\ \vdots \\ x_N^{i,M} \end{bmatrix} \right] \updownarrow N$$

Sum A_{i0} $\quad A_{ij}$ $\quad A_{iM}$

$$\underline{w}^i = \begin{bmatrix} w_0^i \\ \vdots \\ w_M^i \end{bmatrix} \updownarrow M$$

$$LHS = [A_{i0} \cdots A_{iM}] \cdot \begin{bmatrix} w_0^i \\ \vdots \\ w_M^i \end{bmatrix}$$

(7)

Solution for #5

Substitute (2) into (1)

$$\sum_{j=0}^{M} \sum_{n=1}^{N} x_n^{i+j} w_j = \sum_{n=1}^{N} x_n^i t_n \quad (3)$$

For (3). $LHS = \sum_{j=0}^{M} \left(x_1^{i+j} + x_2^{i+j} + \cdots + x_N^{i+j} \right) w_j$

$$= \underbrace{x_1^i \sum_{j=0}^{M} x_1^j w_j + x_2^i \sum_{j=0}^{M} x_2^j w_j + \cdots + x_N^i \sum_{j=0}^{M} x_N^j w_j}_{N}$$

$$\Rightarrow \left\{ \sum_{j=0}^{M} x_n w_j = y_n \right\}$$

$$= x_1^i y_1 + \cdots x_N^i y_n = \sum_{n=1}^{N} x_n^i y_n$$

$$LHS = RHS$$

$$\sum_{n=1}^{N} x_n^i y_n = \sum_{n=1}^{N} x_n^i t_n$$

$$\sum_{n=1}^{N} y_n = \sum_{n=1}^{N} t_n \quad (4)$$

(4) It shows that for optimal $\underline{w}$, prediction is identical to target @ n

In another word, the $\underline{w}$ minize the loss function are given by solution (1).

6- Show that derivative of loss function

$$E(\underline{w}) = -\sum_{n=1}^{N} \{ t_n \ln(y_n) + (1-t_n) \ln(1-y_n) \}$$

with respect to activation value a_k (which is argument of activator function)

for an output unit having a logistic sigmoid activation

Sigmoid activation function satisfies

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

Note that sigmoid is $\sigma(a) = \{1 + \exp(-a)\}^{-1}$

$$\frac{\partial E}{\partial a_k} = -\left[\frac{t_k}{y_k} \times \frac{\partial \sigma(a_k)}{\partial a_k} - (1-t_k)\left(\frac{1}{1-y_k}\right) \times \frac{\partial \sigma_k(a_k)}{\partial a_k} \right] \quad (1)$$

For sigmoid.

$$\frac{\partial \sigma(a)}{\partial a} = -1 \left[1 + \exp(-a)\right]^{-2} \times \left[-\exp(-a)\right]$$

$$= \frac{e^{-a}}{(1+e^{-a})^2} = \frac{e^{-a}}{1+e^{-a}} \times \frac{1}{1+e^{-a}} = \frac{1}{1+e^{-a}}\left(1 - \frac{1}{1+e^{-a}}\right)$$

$$\boxed{\frac{\partial \sigma(a)}{\partial a} = \sigma(a) \cdot [1 - \sigma(a)]}$$

Substitute into (1)

$$\frac{\partial E}{\partial a_k} = -\left\{ \frac{t_k}{\sigma(a_k)} \cdot \sigma(a_k)[1-\sigma(a_k)] - \frac{1-t_k}{[1-\sigma(a_k)]} \sigma(a_k)[1-\sigma(a_k)] \right\}$$

$$\frac{\partial E}{\partial a_k} = -1 \times [\, t_k - t_k \cdot y_k - (1-t_k) y_k \,]$$

$$\boxed{\frac{\partial E}{\partial a_k} = y_k - t_k} \quad \checkmark \quad (4)$$

7. Show that sigmoid $\sigma(a)$, $\frac{d\sigma}{da}$ can be expressed as function of σ, $\sigma(a) = \{1 + \exp(a)\}^{-1}$

$$\frac{d\sigma}{da} = -1 \cdot \{1 + \exp(-a)\}^{-2} \times -1 \times \exp(-a)$$

$$= \frac{e^{-a}}{1+e^{-a}} \times \frac{1}{1+e^{-a}}$$

$$= \left(1 - \frac{1}{1+e^{-a}}\right)\left(\frac{1}{1+e^{-a}}\right)$$

where $\frac{1}{1+e^{-a}} = \sigma(a)$

thus $\boxed{\frac{d\sigma}{da} = (1-\sigma)\sigma}$ ✓

8. Show that for activation function $\tanh(a) = g(a)$.

dg/da can be a function of g.

$$\tanh(a) = \left\{ \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} \right\}$$

$$\frac{dy}{da} = \frac{[\exp(a) - \exp(-a)]'[\exp(a) + \exp(-a)] - [\sim][\sim]'}{[\exp(a) + \exp(-a)]^2}$$

$$= \frac{(e^a + e^{-a})^2 - (e^a - e^{-a})^2}{[\exp(a) + \exp(-a)]^2}$$

$$= 1 - \left[\frac{e^a - e^{-a}}{e^a + e^{-a}}\right]^2$$

$$\boxed{\frac{dy}{da} = 1 - y^2} \quad \checkmark$$

9. A matrix $\underline{\underline{X}}$, following eigenvector equation

$$\underline{\underline{X}} \cdot \underline{u_i} = \lambda_i \underline{u_i}$$

λ_i Eigvalue

$\underline{u_i}$ Eigvectors

Eigenvectors : Orthonormal set

- therefore, an arbitrary vector $\underline{v}$ we have:

$$\underline{v}^T \underline{\underline{X}} \cdot \underline{v} = \Sigma_i c_i^2 \lambda_i$$

— c_i are coefficients of $\underline{v}$ in basis of eigenvectors

- By setting $\underline{v}$ equal to each $\underline{u_i}$, show that $\underline{\underline{X}}$ is positive define, if and only if all eigenvalues are positive (ie. if & only if $\underline{v}^T \cdot \underline{\underline{X}} \cdot \underline{v} > 0$ for all $\underline{v}$)

Since $\underline{u_i}$ are orthonormal, which means they are orthogonal & all of unit length mutually.

thus $\underline{u_i}^T \cdot \underline{\underline{X}} \underline{u_i} = \lambda_i$ for all u_i & λ_i

If $\underline{\underline{X}}$ is positive-defined, if and only if all $\lambda_i > 0$

$\underline{u_i}^T \underline{\underline{X}} \underline{u_i}$ since $\underline{\underline{X}} \cdot \underline{u_i} = \lambda_i \underline{u_i}$

$\Rightarrow \underline{u_i}^T \cdot \lambda_i \underline{u_i}$ the eigenvectors are orthonormal,

$\underline{u_i}^T \cdot \underline{u_i} = 1$

thus $\underline{u_i}^T \cdot \underline{\underline{X}} \underline{u_i} = \lambda_i$

If $\underline{\underline{X}}$ is positive definite, if and only if all eigenvectors are positive

10. Consider a quadratic loss function defined by

$$E(\underline{w}) = E(\underline{w}^*) + \frac{1}{2}(\underline{w}-\underline{w}^*)^T \underline{\underline{H}} (\underline{w}-\underline{w}^*)$$

where $\underline{\underline{H}}$ is an known operator evaluated at the set of weights $\underline{w}^*$

if $\underline{\underline{H}}$ has a eigenvalue equation $\underline{\underline{H}}\,\underline{u}_i = \lambda_i \underline{u}_i$

Show that the contour of constant error C are ellipses. the axes are aligned with eigenvectors $\underline{u}_i$

(Hint: you can start by expressing $(\underline{w}-\underline{w}^*)$ as a linear combination of eigenvectors)

Express the linear combination of $(\underline{w}-\underline{w}^*)$ with eigenvectors $\underline{u}_j$

$$(\underline{w}-\underline{w}^*) = \sum_{j=0}^{N} k_j \underline{u}_j \qquad \left[\begin{array}{c|c|c|c} | & | & | & | \\ u_{0k_0} & \cdots & u_{jk_j} & u_{Nk_N} \\ | & | & | & | \end{array}\right]$$

Thus the $\frac{1}{2}(\underline{w}-\underline{w}^*)\underline{\underline{H}}(\underline{w}-\underline{w}^*)$ can expressed as

$$C = \frac{1}{2}\sum_{i=0}^{N} k_i \underline{u}_i \sum_{j=0}^{N} \lambda_j k_j \underline{u}_j \qquad \text{since } \underline{\underline{H}}\underline{u}_i = \lambda_i \underline{u}_i$$

~~thus $C = \frac{1}{2}\sum_{i=0}^{N} \lambda_i$~~

Since $\underline{u}_i$ and $\underline{u}_j$ are orthogonal, and all the eigenvectors are normalized.

thus $\underline{u}_i \cdot \underline{u}_j = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{else } i \neq j \end{cases}$

$$C = \frac{1}{2}\sum_{i=0}^{N} k_i^2 \lambda_i$$

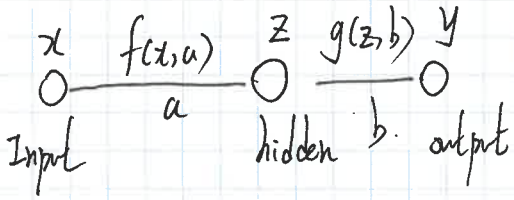
which is ellipses contour in hyperplane, and aligned with $\underline{u}_i$

12. Consider One-node, one-hidden-layer network

Assume linear activation function $f(\xi, \alpha) = g(\xi, \alpha) = \alpha\xi$

For loss function $E = \frac{1}{2}(y_0 - y)^2$

Calculate expressions for $\frac{\partial E}{\partial a}$ and $\frac{\partial E}{\partial b}$



y_0 : target

a, b. weight

x: input, z : hidden layer output, y output

$$\frac{dE}{da} = \frac{dE}{dy}\frac{dy}{dz}\frac{dz}{da} = \frac{dE}{dy} \cdot \frac{dy(z,b)}{dz} \cdot \frac{df(x,a)}{da}$$

$$= -(y_0 - y) \cdot b \cdot x = (y - y_0)\, b\, x$$

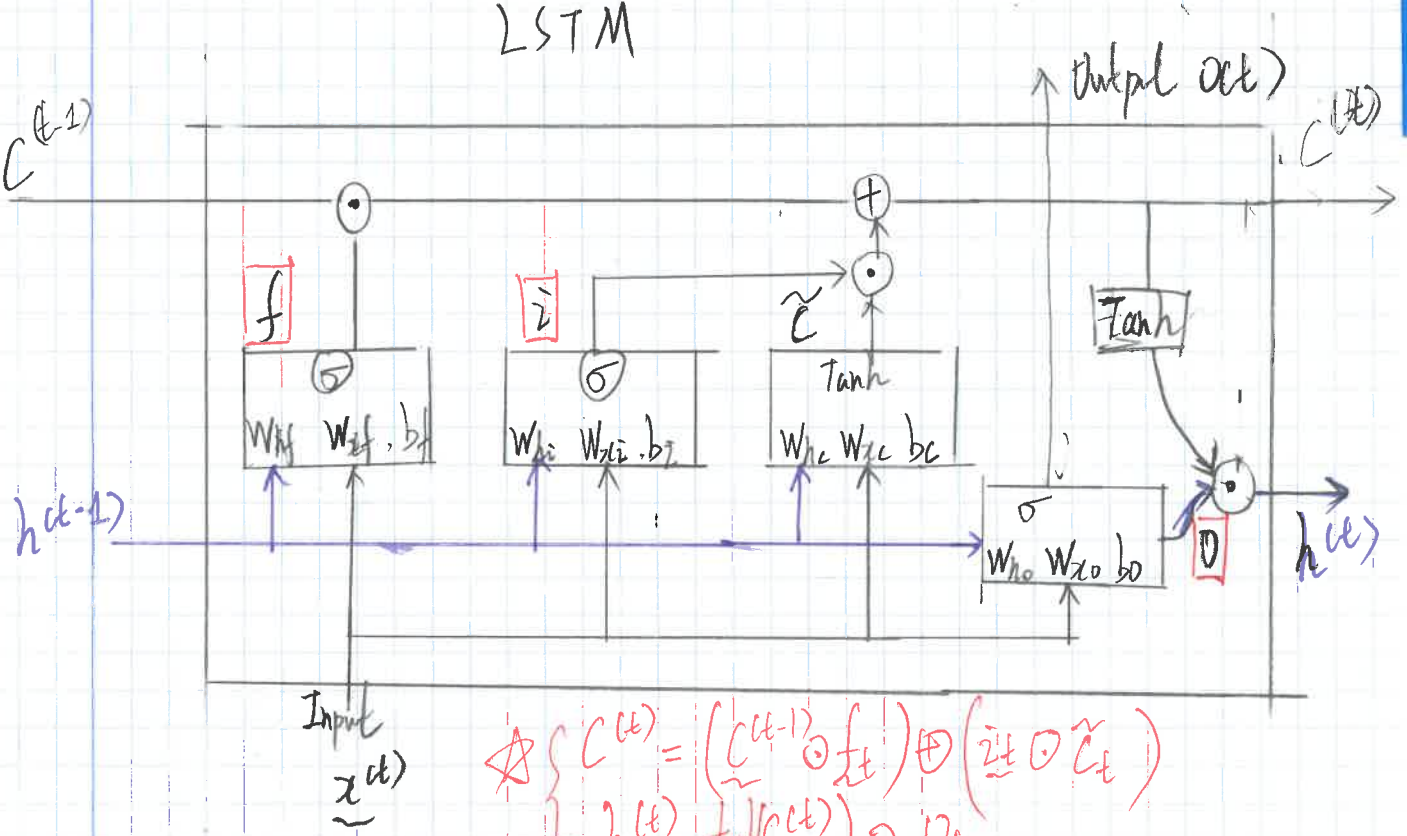
$$\frac{dE}{db} = \frac{dE}{dy}\frac{dy}{db} = \frac{dE}{dy} \cdot \frac{dg(z,b)}{db}$$

$$= -(y_0 - y) \cdot z$$

$$= (y - y_0) \cdot a \cdot x$$

12. Describle different gates in LSTM, write algorithm for output

LSTM



$$\begin{cases} C^{(t)} = (C^{(t-1)} \odot f_t) \oplus (i_t \odot \tilde{C}_t) \\ h^{(t)} = \tanh(C^{(t)}) \odot O_t \end{cases}$$

Operator $\odot$: Element-wise Product. , $\oplus$ element-wise Summary

Activation fun σ Sigmoid. Tanh, tanh,

Variable state C: Cell state. h: history

1. f: forget gate: Decide which information is allowed to go through reset cell state not going infinitely.

$$f_t = \sigma(W_{hf} \cdot h^{(t-1)} + W_{xf} x^{(t)} + b_f)$$

2. i: Input Gate, $\tilde{C}$: Candidate Value } updating cell state

$$\begin{cases} i_t = \sigma(W_{hi} h^{(t-1)} + W_{xi} x^{(t)} + b_i) \\ \tilde{C} = \tanh(W_{hc} h^{(t-1)} + W_{xc} x^{(t)} + b_c) \end{cases}$$

3. Output Gate

$$O_t = \sigma(W_{ho} h^{t-1} + W_{xo} x^{(t)} + b_o)$$

13. Consider a CNN where we want Global output
Do we need any-fully-connected layer?
Motivate your answer.

- It depends on the problem CNN is going to solve.
- In theory, once we use Global Max Pooling layer

  OR Global Average Pooling layer, we do not need Fully-Connected layer anymore, if we just need a global output.

  it is suitable for AutoEncoder or GAN

- However, if suppose we are dealing with image classification

  We need linear output, say with sigmoid activation function

  thus loss function Binary Crossentropy / Softmax can be implemented.

  In this case, we do need fully connected layer providing a linear output.

12. Consider on CNN with 5 input channels of 256^2 grids point each and a kernel size (5×5).

If we apply total 64 filters in first layers

How many parameters will each of filters have in that layer?

Parameters in CNN Layer totally will be

$$(\text{Num\_filter}) \times \left[(\text{filter Height}) \times (\text{filter width}) \times \text{Num\_channels} + 1\right]_{\text{Bias}}$$

=

For each filter

$$(\text{Height} \times \text{width} \times \text{Num\_channels} + 1)_{\text{Bias}}$$

$$= 5 \times 5 \times 5 + 1 = \boxed{126 \text{ paras}}$$

15. In a snapshot matrix $\underset{\sim}{X} \in R^{n \times m}$ $n >> m$.

If we want to apply method of snapshot to compute SVD which correlation matrix is more convient to use?

- The $\underset{\sim}{X}^* \cdot \underset{\sim}{X}$ is more convent since $\underset{\sim}{X}^* \cdot \underset{\sim}{X} \in R^{m \times m}$ is very low-dimensional matrix. , m = no. of snapshots
- Then by eigenvalue decomposition of $\underset{\sim}{X}^* \cdot \underset{\sim}{X}$ we can get right singular vectors $\underset{\sim}{V_j}$ & singular values $\underset{\sim}{\sigma_j}$
- Then by definition of SVD:

$$\underset{\sim}{X} = \underset{\sim}{U} \cdot \underset{\sim}{\Sigma} \cdot \underset{\sim}{V}^*$$

$$\underset{\sim}{X}\, \underset{\sim}{V_j} = \underset{\sim}{U_j} \cdot \sigma_j \text{ , } \underset{\sim}{U_j} \text{ is left singular vector}$$

$$\underset{\sim}{U_j} = \underset{\sim}{X}\, \underset{\sim}{V_j} / \sigma_j$$

get first m left singular vector, m corresponds to number of snapshot

alternative, get, maintain $\hat{\underset{\sim}{V}}$ & $\hat{\underset{\sim}{\Sigma}}$ which only has non-zero value

$$\underset{\sim}{U} = \underset{\sim}{X}\, \hat{\underset{\sim}{V}} \cdot \hat{\underset{\sim}{\Sigma}}^{-1}$$

16. Compute Eigenvalues & Eigenvectors of Matrices

$$\underline{\underline{A}} = \begin{bmatrix} 5 & 7 \\ -2 & -4 \end{bmatrix}, \quad \underline{\underline{B}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \underline{\underline{C}} = \begin{bmatrix} 1 & 2 & 1 \\ 6 & -1 & 0 \\ -1 & 2 & 1 \end{bmatrix}$$

① $\underline{\underline{A}}$: $\det(\underline{\underline{A}} - \lambda \cdot \underline{\underline{I}}^{2\times 2}) = 0$

$$\det\left(\begin{bmatrix} 5 & 7 \\ -2 & -4 \end{bmatrix} - \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\right) = 0$$

$$\begin{bmatrix} (5-\lambda) & 7 \\ -2 & -4-\lambda \end{bmatrix} = 0$$

$$(5-\lambda)(-4+\lambda_2) + 14 = 0$$

$$-20 - 5\lambda + 4\lambda + \lambda^2 + 14 = 0$$

$$\lambda^2 - \lambda - 6 = 0 \Rightarrow \begin{cases} \lambda_1 = 3 \\ \lambda_2 = -2 \end{cases}$$

$$\begin{bmatrix} 5 & 7 \\ -2 & -4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 3 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\begin{cases} 5v_1 + 7v_2 = 3v_1 \\ -2v_1 - 4v_2 = 3v_2 \end{cases} \quad \begin{cases} -2v_1 = 7v_2 \\ -7v_2 = 2v_1 \end{cases}$$

$$v_1 = \begin{bmatrix} -1 \\ \frac{2}{7} \end{bmatrix}$$

$$\begin{cases} 5v_2 + 7v_2 = -2v_1 \\ -2v_1 - 4v_2 = -2v_2 \end{cases} \Rightarrow \begin{cases} v_1 = -v_2 \\ v_1 = -v_2 \end{cases}$$

$$v_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\underline{\underline{Q}} = \begin{bmatrix} -1 & 1 \\ \frac{2}{7} & -1 \end{bmatrix} \quad \underline{\underline{\Lambda}} = \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix} \quad \underline{\underline{A}}\,\underline{\underline{Q}} = \underline{\underline{Q}}\,\underline{\underline{\Lambda}} \checkmark$$

17. Compute full SVD & Pseudo inverse of $\underline{\underline{A}}$

$$\underline{\underline{A}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

① Full SVD $\quad A = \underline{\underline{U}} \cdot \underline{\underline{\Sigma}} \cdot \underline{\underline{V}}^*$

$$\underline{\underline{A}}^* = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \in R^{3\times 2}$$

let $\underline{\underline{X}} = \underline{\underline{A}}^* \cdot \underline{\underline{A}} \in R^{3\times 3}$ to find $\underline{\underline{V}}$ & $\underline{\underline{\Sigma}}$

$$\underline{\underline{X}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad \underline{\underline{X}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

eigenvalue of $\underline{\underline{X}}$ $\begin{cases} \lambda_1 = 1 \\ \lambda_2 = 1 \\ \lambda_3 = 0 \end{cases}$

eigenvectors of $\underline{\underline{X}}$ $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

thus $\boxed{\underline{\underline{V}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in R^{3\times 3}}$

$$\boxed{\underline{\underline{\Sigma}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \in R^{2\times 3}}$$

Then Let $\underline{\underline{X}}_2 = \underline{\underline{A}} \cdot \underline{\underline{A}}^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

eigenvectors of $\underline{\underline{X}}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$\boxed{\underline{\underline{U}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in R^{2\times 2}}$$

$\underline{\underline{A}} = \underline{\underline{U}} \cdot \underline{\underline{\Sigma}} \cdot \underline{\underline{V}}^*$ checked!

② Pesudo Inverse $\underline{\underline{A}}^{+}$

$$\underline{\underline{A}}^{+} = \underline{\underline{V}}\,\underline{\underline{\Sigma}}^{-1}\,\underline{\underline{U}}^{*}$$ where $\tilde{\underline{\underline{V}}}$, $\tilde{\underline{\underline{\Sigma}}}$ & $\tilde{\underline{\underline{U}}}$ are truncated

$$\underline{\underline{\Sigma}}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\underline{\underline{U}}^{*} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A^{+} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boxed{A^{+} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}}$$

check. $A^{+}A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = I^{2\times 2}$

checked ✓

2. Mention strengths & weakness of POD & DMD, motivate your answer.

	Proper - Orthogonal Decomposition POD	Dynamic Mode Decomposition DMD
Pros:	• Fast Convergence "Optimal Orthogonality" ⇒ Good reconstruction with relative low noise	• Modes have coherent behavior in time • Modes are orthogonal in time • Each mode can be interpret
Cons:	• Not always interpret physical meaning • Temporal & Spatial info. are always mixed together	• Result are less optimal • Oscillation appears @ certain frequency • the growth will decay with time

2|. Describe STLS algorithm in SINDy. Does this algorithm lead to a parsimonious model? Motivate your answer.

for SINDy, it solve coefficients for ODE equation $\dot{X} = \Theta(X)\,\Xi$

STLS is sequential-threshold least square method.

Once the matrix of ODEs coefficient was obtained through SINDy.

(Ξ)

Assume $\Xi \in \mathbb{R}^{n \times m}$, n means the number of terms in library $\Theta(X)$

m is the number of ODE we want

thus for $i = 1$ to k: self defined iteration

for: $j = 1 : m$

small = where $|\Xi[:, j]| < \lambda$ (threshold, self defined)

$\Xi[small, j] = 0$

$big = \sim small$

$\Xi[big, j] = least\_square\_regression(\Theta(X)[:, big], \dot{X}[:, j])$

end

end

By iteration and setting threshold, it can ensure most of coefficient will be filtered without affect the result.

In this way, the Ξ matrix is obtained as sparse as possible which means the ODE will has least coefficients

it lead to a parsimonious model

22. Describe LASSO, elastic_net & Ridge ~~regression~~ regularization
Write corresponding equations and Norms.

When finding the spares vector in compress sensing problem, in order to improve the accuray & reduce noise as much as possible, the l_2-norm regularization may not sufficient.
thus we consider adding hyper parameter λ to strengeh the pentalty from regularization of $\underline{\underline{s}}$. l_1 norm

then we introduce LASSO : Least absolute shrinkage and selection Operator

$$\hat{\underline{\underline{s}}} = \underset{\underline{\underline{c}}}{\text{Argmin}} \| \underline{\underline{c}} \cdot \underline{\underline{\Psi}} \underline{\underline{s}} - \underline{\underline{y}} \|_2 + \lambda \| \underline{\underline{s}} \|_1$$

where λ is the hyperparameter.
It can effectively regulares the model and avoid overfitting and increase sparsity.

Elastic-net

Elastic Net

$$\hat{\underline{\underline{s}}} = \underset{\underline{\underline{c}}}{\text{Argmin}} \| \underline{\underline{c}} \cdot \underline{\underline{\Psi}} \underline{\underline{s}} - \underline{\underline{y}} \|_2 + \lambda_1 \| \underline{\underline{s}} \|_2 + \lambda_2 \| \underline{\underline{s}} \|_2$$

LASSO

Ridge

23. Describe the DMD algorithm

DMD is able to discover dynamic systems from high-dimensional data. The data can be decomposed into a simple representation based on temporal-spatial features

$$\underset{\sim}{u}(x,t) = \sum_{m=1} a_m \cdot \underset{\sim}{u}_m(x) \exp[(\delta_m + i\omega_m)\cdot t]$$

(a_m: Amplitude, $\underset{\sim}{u}_m(x)$: mode, δ_m: growth, ω_m: Frequency.)

Algorithm: We define Matrix $\underline{X} = \begin{bmatrix} | & & | \\ x_0 & \cdots & x_{n-1} \\ | & & | \end{bmatrix}$ } m

$$\underline{X}' = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} \} m$$

For exact DMD, a representation of $\underline{A}$ is

$$\underline{A}\underline{X} = \underline{X}' \Leftrightarrow \underline{A} = \underline{X}' \cdot \underline{X}^{+}, \quad A \in R^{m\times m}$$

① We take truncated SVD on $\underline{X}$ with $r \ll m$

$$\underline{X} = \underline{U}_r \cdot \underline{\Sigma}_r \underline{V}_r^{*}$$

According to definition, $\underline{X}^{+} = \underline{V}_r \cdot \underline{\Sigma}_r^{-1} \cdot \underline{U}_r^{*}$

② $\underline{A} = \underline{X}' \cdot \underline{V}_r \cdot \underline{\Sigma}_r^{-1} \cdot \underline{U}_r^{*}$, $A \in R^{m\times m}$

However, $\underline{A}$ could be truncated as $r \times r$ via POD-reduced state ☆

$$\hat{\underline{A}} = \underline{U}_r^{*} \underline{A} \cdot \underline{U}_r$$

$$= \underline{U}_r^{*} \cdot \underline{X}' \cdot \underline{V}_r \cdot \underline{\Sigma}_r^{-1} \cdot \underline{U}_r^{*} \cdot \underline{U}_r, \quad \& \ \underline{U}_r^{*} \cdot \underline{U}_r = \underline{I}$$

thus $\hat{\underline{A}} = \underline{U}_r^{*} \cdot \underline{X}' \cdot \underline{V}_r \cdot \underline{\Sigma}_r^{-1}$

<24>

④ Solve eigenvalue decomposition on $\tilde{\underline{A}}$

$$\tilde{\underline{A}} \cdot \underline{W} = \underline{\Lambda} \cdot \underline{W} \quad \begin{cases} \underline{W}: \text{eigenvectors} \\ \underline{\Lambda}: \text{Eigenvalues} \end{cases}$$

⑤ Compute DMD modes

* That's why not use $\underline{V}_r^*$ in projection

modes are orthogonal in temporal space not in spatial

Eigenvectors $\underline{W}$ have been calculated in diffent bias

thus we need to project $\underline{b}$ onto original basis

DMD modes: $\underline{\Phi} = \underline{X}' \cdot \underline{V} \cdot \underline{\Sigma}^{-1} \cdot \underline{W}$, $\underline{b} = \underline{\Phi}^{+} \underline{x}_0$

⑤ DMD Expansion

No input:

$$\underline{u}(\underline{x}, t) = \underline{\Phi}(\underline{x}) \exp(\underline{\Lambda} t) \cdot \underline{b}$$

$$\Downarrow$$

$$\underline{u}(\underline{x}, t) = \sum_{k=1}^{r} \phi_k \cdot b_k(\underline{x}) \exp(\lambda_k \cdot t)$$

24. Describe snapshot method to compute SVD.

When we are dealing with Data $\underline{\underline{X}} \in R^{n \times m}$ $n \gg m$

$$\underline{\underline{X}} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix} \Big\} n$$

$\underbrace{\quad\quad}_{m}$

each x_i for i in m is called a snapshot

We find $X^* \in R^{m \times n}$

- thus $X^* \cdot X \in R^{m \times m}$, which is a very low-dimensional matrix

it means for finding right singular vectors & singular value cost much less computational power and time

- thus we do eigenvalue decomposition on $X^* \cdot X$ first to obtain $\underline{\underline{V}} \in R^{m \times m}$ & $\underline{\underline{\Sigma}} \in R^{n \times m}$

Then for each snapshot(j), the left singular vectors can be yield by

$\star$ $u_j = \underline{\underline{X}} \cdot v_j / \sigma_j$ since for each j,

$$\underline{\underline{X}} = u_j \cdot \sigma_j v_j^*$$

$$\underline{\underline{X}} \cdot v_j = u_j \cdot \sigma_j \cdot 1$$

$$u_j = \underline{\underline{X}} \cdot v_j / \sigma_j$$

25. Describle calculation of POD in case of non-uniform sampling in space & time. Consider that $\underline{\underline{M}}$ & $\underline{\underline{N}}$ are mass matrix and matrix containing temporal weights respectively.

The POD (proper-Orthogonal Decomposition) is used for get deterministic & Orthogonal basis function $\underline{\underline{\Phi}}$ and their time collections $\underline{\underline{A}}$ to decompose snapshot matrix $\underline{\underline{X}} \in R^{n \times m}$

① ∵ According to the definition

$$\underline{\underline{X}} = \underline{\underline{\Phi}} \cdot \underline{\underline{A}} = \underline{\underline{\Phi}} \cdot \underline{\underline{\Sigma}} \cdot \underline{\underline{V}}^* \quad (*)$$

② We defined that inner product for $\underline{x_j}$ snapshot

$\langle \underline{x_j}, \underline{x_j} \rangle_{\underline{\underline{M}}} = \underline{x_j}^\dagger \cdot \underline{\underline{M}} \cdot \underline{x_j}$ where $\underline{\underline{M}}$ is matrix of mass (diagnol) for non-uniform meshes, also valid for matrix $\underline{\underline{N}} \in R^{m \times m}$

③ In (*), $\| \underline{\underline{\Phi}} \|_M = \underline{\underline{I}}^{n \times n}$

$\| \underline{\underline{V}} \|_N = \underline{\underline{I}}^{m \times m}$

④ Denote $\underline{\underline{M}}^{\frac{1}{2}}$, $\underline{\underline{N}}^{\frac{1}{2}}$ as normalized $\underline{\underline{M}}$ & $\underline{\underline{N}}$ in spatial & temporal

$\hat{\underline{\underline{\Phi}}}$ & $\hat{\underline{\underline{V}}}$ as normalized $\underline{\underline{\Phi}}$, $\underline{\underline{V}}$ $\quad \hat{\underline{\underline{\Phi}}}\hat{\underline{\underline{\Phi}}}^* = \underline{\underline{I}}^{n \times n}$, $\hat{\underline{\underline{V}}}\hat{\underline{\underline{V}}}^* = \underline{\underline{I}}^{m \times m}$

then (*) is

$$SVD\left(\underline{\underline{M}}^{\frac{1}{2}} \cdot \underline{\underline{X}} \cdot \underline{\underline{N}}^{\frac{1}{2}}\right) = \hat{\underline{\underline{\Phi}}} \cdot \underline{\underline{\Sigma}} \cdot \hat{\underline{\underline{V}}}^*$$

26. Derive the Galerkin Projection after apply POD on a partial differential equation

For a dynamic equation $\dot{\underline{x}} = f(\underline{x})$ $\underline{x} \in R^n$ (1)

Suppose we have original data of $\underline{x}$.

$\underline{\underline{X}} \in R^{n \times m}$ obtain m snapshots.

① first we apply economic / truncated SVD on $\underline{\underline{X}}$

$$\underline{\underline{X}} = \underline{\underline{V}}_r \underline{\underline{\Sigma}}_r \underline{\underline{V}}_r^*$$ which $r << n$

② then we can obtain low-dimensional approximation of $\underline{\underline{X}}$

$$\underline{\underline{V}}_r^* \underline{\underline{X}} = \underline{\underline{Z}} \Leftrightarrow \underline{\underline{X}} = \underline{\underline{V}}_r \cdot \underline{\underline{Z}}$$

Substitute (2) into (1)

$$\frac{\partial \underline{x}}{\partial t} = f(\underline{x})$$

$$\frac{\partial}{\partial t} \frac{d}{dt} (\underline{\underline{V}}_r \underline{\underline{Z}}) = f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}})$$

since $\underline{\underline{V}}_r$ is the spatial modes, thus it is not related to time!

$$\underline{\underline{V}}_r \frac{d}{dt} \frac{\partial}{\partial t} (\underline{\underline{Z}}) = f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}})$$

$$\underline{\underline{V}}_r^* \cdot \underline{\underline{V}}_r \frac{d}{dt} (\underline{\underline{Z}}) = \underline{\underline{V}}_r^* f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}})$$

$$\dot{\underline{\underline{Z}}} = \underline{\underline{V}}_r^* \cdot f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}}) = g(\underline{\underline{Z}}) \quad (3)$$

(3) is called Galerkin Projection onto a POD subspace.

26. Derive the Galerkin Projection after apply PoD on a partial differential equation

For a dynamic equation $\dot{\underline{x}} = f(\underline{x})$ $\underline{x} \in R^n$ (1)

Suppose we have original data of $\underline{x}$.

$\underline{\underline{X}} \in R^{n \times m}$ obtain m snapshots.

① first we apply economic / truncated SVD on $\underline{\underline{X}}$

$$\underline{\underline{X}} = \underline{\underline{V}}_r \underline{\underline{\Sigma}}_r \underline{\underline{V}}_r^*$$ which $r << n$

② then we can obtain low-dimensional approximation of $\underline{\underline{X}}$

$$\underline{\underline{V}}_r^* \underline{\underline{X}} = \underline{\underline{Z}} \Leftrightarrow \underline{\underline{X}} = \underline{\underline{V}}_r \cdot \underline{\underline{Z}}$$

Substitute (2) into (1)

$$\frac{d\underline{x}}{dt} = f(\underline{x})$$

$$\frac{\partial}{\partial t}\frac{d}{dt}(\underline{\underline{V}}_r \underline{\underline{Z}}) = f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}})$$

since $\underline{\underline{V}}_r$ is the spatial modes, thus it is not related to time!

$$\underline{\underline{V}}_r \frac{d}{dt}(\underline{\underline{Z}}) = f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}})$$

$$\underline{\underline{V}}_r^* \cdot \underline{\underline{V}}_r \frac{d}{dt}(\underline{\underline{Z}}) = \underline{\underline{V}}_r^* f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}})$$

$$\dot{\underline{\underline{Z}}} = \underline{\underline{V}}_r^* \cdot f(\underline{\underline{V}}_r \cdot \underline{\underline{Z}}) = g(\underline{\underline{Z}}) \quad (3)$$

(3) is called Galerkin Projection onto a PoD subspace.

27. Discuss the difference between variational autoencoder, denoising autoencoder and a sparse autoencoder.

- Variational AutoEncoder, instead of turn data into a latent vector, it assumes data generated by normal distribution and compress input as its mean and variance parameters in latent space.
  Then the mean & variance are generated statistically with randomness into a vector, finally the decoder will reconstruct data by this vector

- Denoising encoder holds the same architecture as AutoEncoder, the difference is, the inputs of denoising encoder is added with some random noise, while the target is original data. In this way, a trained denoising autoencoder can elimate noise of input data

- Sparse autoencoder: Holds the sparse information when it is compressed into latent vector. the difference mainly at the loss function.
  Sparse autoencoder use L_1 norm regularization /penalty when computing reconstruction loss.
  In this way, some coefficient can be shrink to ZERO, but L_2 norm can only let coefficient close to ZERO, which ensure the sparsity.

28. Derive ROM for Navier-Stokes Equation by POD and Galerkin Projection

A

For compressible Navier-Stokes Equation

Mass: $\nabla \cdot \underset{\sim}{u} = 0$

Momentum: $\frac{\partial \underset{\sim}{u}}{\partial t} + (\underset{\sim}{u} \cdot \nabla)\underset{\sim}{u} = -\nabla \cdot p + \frac{1}{Re}\nabla^2 \underset{\sim}{u}$

① First, we do POD on $\underset{\sim}{u}(\underset{\sim}{x}, t)$, which can be expressed as:

$$\underset{\sim}{u}(\underset{\sim}{x}, t) = \sum_{k=0}^{R} a_k(t)\,\underset{\sim}{u}_k(\underset{\sim}{x})$$

$$= \sum_{k=1}^{R} a_k(t)\,\underset{\sim}{u}_k(\underset{\sim}{x}) + \underset{\sim}{u}_0(\underset{\sim}{x})$$

where $R \ll M$

② Apply Galerkin Projection by the form of inner product for each $\underset{\sim}{u}_j$ in $\underset{\sim}{U}^*$ $\underset{\sim}{U}^* \in R^{R \times N}$

$$\dot{\underset{\sim}{a}}(\underset{\sim}{x}, t) = \sum_{k=0}^{R} b_{ik}\, a_k(t) + \sum_{k=1}^{R}\sum_{j=0}^{R} c_{ik}\, a_k(t)\cdot a_j(t)$$

where $b_{ik} = \langle \nabla \underset{\sim}{u}_i(\underset{\sim}{x}), \sum \underset{\sim}{u}_k(\underset{\sim}{x}) \rangle$

$c_{ik} = \langle \underset{\sim}{u}_i(\underset{\sim}{x}), [\underset{\sim}{u}_i(\underset{\sim}{x}) \nabla] \underset{\sim}{u}_k(\underset{\sim}{x}) \rangle$

29. Describe different kind of learning covered in this course, provide examples

Mainly talking about Supervised & Unsupervised learning in this course.

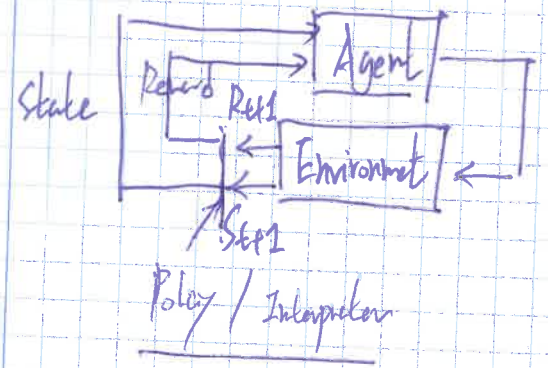
- Supervised learning is the learning algorithms are trained through labeled target. Such as Linear Regression, logistic Regression, SVM are belongs to supervised learning

- Unsupervised learning are learning algorithms are trained without target or labeled data. The k-Means, Decision trees are belongs to unsupervised learn.

- In the course, since autoencoder is introduced as well, I suppose it could be categorated as Semi-Supervised learning to some extent.

3. Describe Reinforcement learning and provide a schematic representation with the various interacting part.

- Reinforcement learning is a kind of machine learning that enables an agent to learn an interactive environment by trail & error using feed back from its own actions & experience
- Unlike Supervise learning, it does not correct the actions it: use reward & punishment for Pos / neg behavior
- Unline Unsupervised learning, it aims at maximize total cumulative reward of agent



1. Environt: Physical word where agent operates
2. State: Current state of agent
3. Reword: feed back from environment
4. Policy: map state & Action
5. Value: Future reward that an agent would receive by taking action