IFT 4030/7030,
Machine Learning for Signal Processing
Week1: Class Intro,
Linear Algebra Refresher

Cem Subakan

What is this class?

- What do you think this class is?

What is this class?

- What do you think this class is?
- Is it a Machine Learning class?
- Is it a Signal Processing class?

What is this class?

- What do you think this class is?
- Is it a Machine Learning class?
- Is it a Signal Processing class?
- What is Machine Learning?
- What is Signal Processing?

Signal Processing

- Here's the wikipedia definition:

  Signal processing is an electrical engineering subfield that focuses on analyzing, modifying and synthesizing *signals*, such as sound, images, potential fields, seismic signals, altimetry processing, and scientific measurements.[1] Signal processing techniques are used to optimize transmissions, digital storage efficiency, correcting distorted signals, subjective video quality and to also detect or pinpoint components of interest in a measured signal.[2]

Signal Processing

- Here's the wikipedia definition:

  > **Signal processing** is an electrical engineering subfield that focuses on analyzing, modifying and synthesizing *signals*, such as sound, images, potential fields, seismic signals, altimetry processing, and scientific measurements.[1] Signal processing techniques are used to optimize transmissions, digital storage efficiency, correcting distorted signals, subjective video quality and to also detect or pinpoint components of interest in a measured signal.[2]

- Hm, this kinda sounds like machine learning.

How are signals different than data?

- So, signals are just data?
- Yeah-(ish).

How are signals different than data?

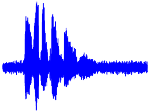
- So, signals are just data?
- Yeah-(ish).
- Why are we calling them signals then?

How are signals different than data?

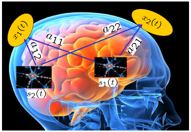
- So, signals are just data?
- Yeah-(ish).
- Why are we calling them signals then?
- When we speak of signals, we refer more to structured data. (Order matters)
- And, saying ‘signals’, ‘signal processing’ implies a more Electrical Engineering way to the approach.

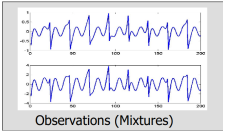
Example Signals

- Images, Audio/Speech

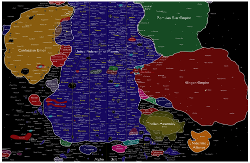


- Brains



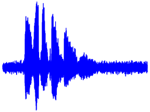


- Financial Time Series, Graphs

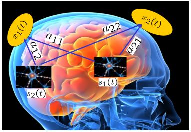


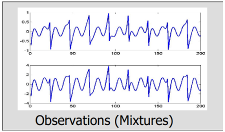
Example Signals

- Images, Audio/Speech

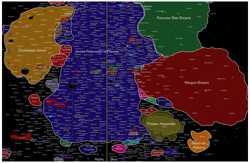



- Brains



- Financial Time Series, Graphs



- More?

But why bother? Isn't ML what's hip now?

- Yes, ML is extremely popular, and we should embrace that.

But why bother? Isn't ML what's hip now?

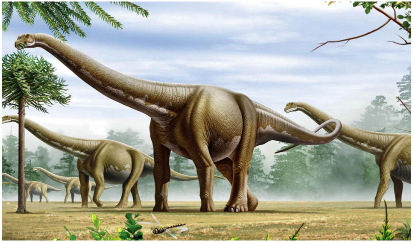
- Yes, ML is extremely popular, and we should embrace that.
- But, traditional ML isn't very friendly for signals.

But why bother? Isn't ML what's hip now?

- Yes, ML is extremely popular, and we should embrace that.
- But, traditional ML isn't very friendly for signals.
- What about signal processing, doesn't that cover what we need?

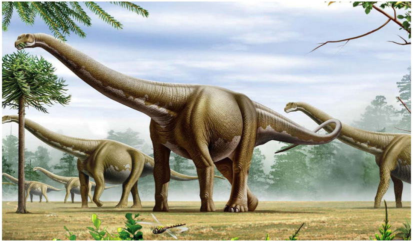
But why bother? Isn't ML what's hip now?

- Yes, ML is extremely popular, and we should embrace that.
- But, traditional ML isn't very friendly for signals.
- What about signal processing, doesn't that cover what we need?
  - No!



But why bother? Isn't ML what's hip now?

- Yes, ML is extremely popular, and we should embrace that.
- But, traditional ML isn't very friendly for signals.
- What about signal processing, doesn't that cover what we need?
  - No!



  - Traditional SP is typically **NOT** statistical, doesn't handle the statistical patterns of the signal well.
  - Traditional SP: Filtering, acquision, analog-digital-analog conversion, transmission
  - There is statistical signal processing also, but it doesn't go much beyond adaptive filtering.

MLSP: Machine Learning for Signal Processing

- How to build systems that would work with sequences and solve machine intelligence tasks on them?
  - Various tasks with Speech and Audio: ASR, Speech Enhancement, Music Transcription...

MLSP: Machine Learning for Signal Processing

- How to build systems that would work with sequences and solve machine intelligence tasks on them?
  - Various tasks with Speech and Audio: ASR, Speech Enhancement, Music Transcription...
  - Financial Time Series Prediction

MLSP: Machine Learning for Signal Processing

- How to build systems that would work with sequences and solve machine intelligence tasks on them?
  - Various tasks with Speech and Audio: ASR, Speech Enhancement, Music Transcription...
  - Financial Time Series Prediction
  - Understanding Biomedical Sequences

MLSP: Machine Learning for Signal Processing

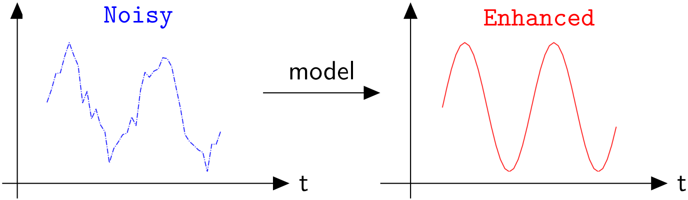
- How to build systems that would work with sequences and solve machine intelligence tasks on them?
  - Various tasks with Speech and Audio: ASR, Speech Enhancement, Music Transcription...
  - Financial Time Series Prediction
  - Understanding Biomedical Sequences
  - Generating Videos

MLSP: Machine Learning for Signal Processing

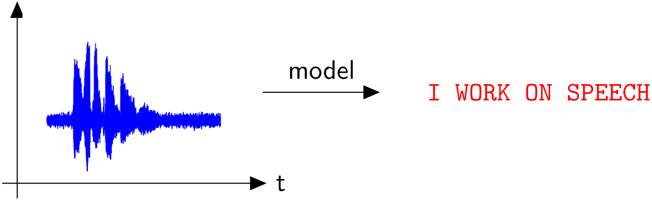
- How to build systems that would work with sequences and solve machine intelligence tasks on them?
  - Various tasks with Speech and Audio: ASR, Speech Enhancement, Music Transcription...
  - Financial Time Series Prediction
  - Understanding Biomedical Sequences
  - Generating Videos
  - More...

Speech and Audio Modeling

- Speech Enhancement

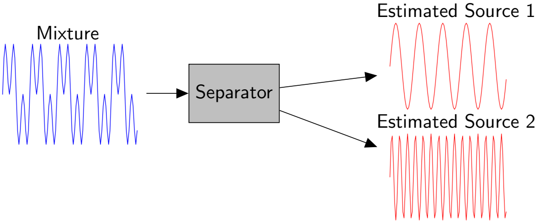


- Speech Recognition

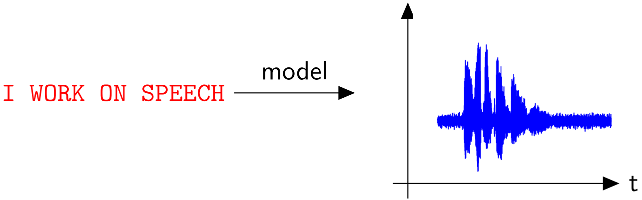


Speech and Audio Modeling

- Speech Separation

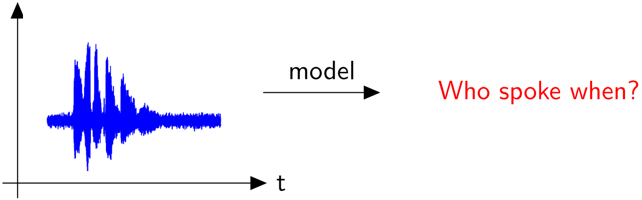


- Text-to-Speech

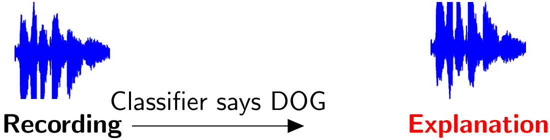


Speech and Audio Modeling

- Speaker Diarization



- Neural Network Explanation



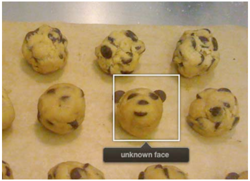
- Other problems: Generating Deep fakes, Detecting deep fakes, Music Source Separation, Music Transcription, Sound Event Detection/Classification...

Speech and Audio Modeling

- Field with huge economic value & job opportunities,
  - Speech Recognition (e.g. Siri)
  - Speech Enhancement (e.g. Google meet, Zoom)
  - Text-to-Speech
  - Speaker Verification, Spoof Detection(Banks)
  - Speaker Diarization for Meeting Analysis (Nuance, Microsoft)
  - Source Separation (e.g. Beatles Rock Band, Meeting Analysis)

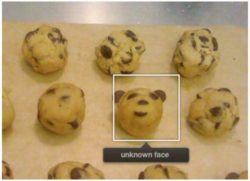
Other real-life applications

- Face recognition



Other real-life applications

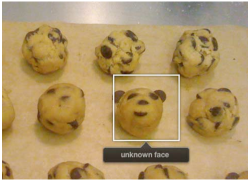
- Face recognition



- Brain-machine interfaces

Other real-life applications

- Face recognition



- Brain-machine interfaces

- Real time bio-signal analysis, learning generative models for bio/medical signals, condition monitoring (mining machines, production machines), Stock market, many more..

About this class

- This is class heavy on practice. How do we make things that work?
- We do not do deep theory in this class.
  - We will not prove things.
  - We will not stay Keras level either.
  - Our goal is to give useful insights, be useful.
- We go fast, our typical lecture could be a class.

Syllabus: Basics

- Linear Algebra
  - This class
- Probability
  - Probability Calculus, Random Variables, Bayesian vs Frequentist Principles
- Signal Processing
  - Signal Representations, Fourier Transform, Sampling

Syllabus: Machine Learning

- Decompositions
  - PCA, NMF, Linear Regression, Tensor Decompositions
- Classification
  - Logistic Regression, Maximum Margin, Kernels, Boosting
- Deep Learning
  - Deep Learning Firearms, Pytorch, Julia
- Optimization
  - Convex optimization
  - Gradient Descent and friends
  - Non-Convex optimization
- Clustering
  - Kmeans, Spectral Clustering, DBScan
- Unsupervised Non-linear learning
  - Manifold Learning, Deep Generative Models
- Time Series Models
  - HMMs, Kalman Filters

Syllabus: Fun Stuff

- Speech Recognition
- Speech Enhancement/Separation
- Text-to-speech
- Representation Learning Methods for Sequences
- Generative Models for Sequences
- Text prompted models (text prompted image / sound generation)
- Neural Network Interpretation Methods
- Graph Signal Processing / ML

Evaluation

- Homeworks (45%)
  - 3 homeworks, you need to work on these alone!
  - I would like you to typeset math in LaTeX. So if you don't know it, start learning it!
  - Do not use Generative AI, if you want to learn!
  - You will need to code. But we will reward good quality presentation of results.
- Weekly Labs (10%)
  - You will work on hands-on application of the things we talk about. TAs will lead the online sessions.
- Final Project (45%)

Final project

- This will be a mini-conference.
- Each paper will receive 3 peer-reviews (from you). We will evaluate the quality of your reviews (5% of your 45% project grade).
- You will work in teams of 2-3 (no more, no less)
- We will ask who did what in the project. So no freeriding!
- Start making friends!
- Mid-October, proposals are due
- Last 1-2 weeks, paper deadline.

Final project

- This will be a mini-conference.
- Each paper will receive 3 peer-reviews (from you). We will evaluate the quality of your reviews (5% of your 45% project grade).
- You will work in teams of 2-3 (no more, no less)
- We will ask who did what in the project. So no freeriding!
- Start making friends!
- Mid-October, proposals are due
- Last 1-2 weeks, paper deadline.
  - We will accept all the papers, and you will make a presentation.

Final project

- This will be a mini-conference.
- Each paper will receive 3 peer-reviews (from you). We will evaluate the quality of your reviews (5% of your 45% project grade).
- You will work in teams of 2-3 (no more, no less)
- We will ask who did what in the project. So no freeriding!
- Start making friends!
- Mid-October, proposals are due
- Last 1-2 weeks, paper deadline.
  - We will accept all the papers, and you will make a presentation.
  - However, you need to do a good job to get a good grade.
  - If it's a good paper, we can also work together to submit it to a real conference! We can work together towards that.

Communications

- We will have teams page where will have a forum, and you will submit your assignments.
- Be active on the forum, ask questions. Find friends for the project.
- We will do the announcements on teams, so sign-up for it!
- Check `https://ycemsubakan.github.io/mlsp.html` for class material.

Instructor: Who am I?

- **Instructor:** Cem Subakan
  - cem.subakan@ift.ulaval.ca
  - Assistant Prof. in Computer Science,
    Mila Associate Academic Member.
  - Just send me a message you if you want to meet.
- I work on machine learning for Speech and Audio.
  - Interpretability
  - Speech Separation & Enhancement
  - Multi-Modal Learning
  - Continual Learning
  - Probabilitic Machine/Deep Learning
- I review for many major conferences, involved in the organization of several MLSP workshops.
- I have written a lot of papers involving MLSP topics, worked with many people, also saw the industry side of things.

Who are the TAs?

- Sara Karami
  - sara.karami.1@ulaval.ca
- Mathieu Bazinet
  - mabaz21@ulaval.ca
- TAs will hold the online lab sessions (Fridays 15h00-16h50)
- The office hours will be on fridays (the second half of the lab sessions)
- Advice:
  - If you need help do not bombard them at the last minute. Seek help early.

Who are you?

- Name, department, grad/undergrad?
  - What are your interests?
  - Hint: Take notes, and contact the person if something picks your interest.

Table of Contents

Table of Contents

Scalars, Vectors, Matrices, Tensors

- Scalar, x, just a number.

Scalars, Vectors, Matrices, Tensors

- Scalar, x, just a number.
- Vector, x, of length L
  $$x = \begin{bmatrix} x_1 \\ \vdots \\ x_L \end{bmatrix}$$

Scalars, Vectors, Matrices, Tensors

- Scalar, x, just a number.

- Vector, x, of length L
$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_L \end{bmatrix}$$

- Matrix, x of size $L \times M$
$$x = \begin{bmatrix} x_{1,1} & \dots & x_{1,M} \\ \vdots & \vdots & \vdots \\ x_{L,1} & \dots & x_{L,M} \end{bmatrix} = \begin{bmatrix} x_1 & \dots & x_M \end{bmatrix}$$

Scalars, Vectors, Matrices, Tensors

- Scalar, x, just a number.
- Vector, x, of length L
$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_L \end{bmatrix}$$
- Matrix, x of size $L \times M$
$$x = \begin{bmatrix} x_{1,1} & \dots & x_{1,M} \\ \vdots & \vdots & \vdots \\ x_{L,1} & \dots & x_{L,M} \end{bmatrix} = \begin{bmatrix} x_1 & \dots & x_M \end{bmatrix}$$
- Tensor, x of size $L \times M \times N$
$$x = \begin{matrix} \begin{bmatrix} x_{1,1,1} & \dots & x_{1,M,1} \\ \vdots & \vdots & \vdots \\ x_{L,1,1} & \dots & x_{L,M,1} \end{bmatrix} \ddots \\ \ddots \begin{bmatrix} x_{1,1,N} & \dots & x_{1,M,N} \\ \vdots & \vdots & \vdots \\ x_{L,1,N} & \dots & x_{L,M,N} \end{bmatrix} \end{matrix}$$

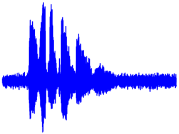
Scalars, Vectors, Matrices, Tensors

- Scalar, x, just a number. **0th order tensor.**
- Vector, x, of length L
  $$x = \begin{bmatrix} x_1 \\ \vdots \\ x_L \end{bmatrix}$$
  1th order tensor.
- Matrix, x of size $L \times M$
  $$x = \begin{bmatrix} x_{1,1} & \dots & x_{1,M} \\ \vdots & \vdots & \vdots \\ x_{L,1} & \dots & x_{L,M} \end{bmatrix} = \begin{bmatrix} x_1 & \dots & x_M \end{bmatrix}$$
  2nd order tensor.
- Tensor, x of size $L \times M \times N$
  $$x = \begin{bmatrix} x_{1,1,1} & \dots & x_{1,M,1} \\ \vdots & \vdots & \vdots \\ x_{L,1,1} & \dots & x_{L,M,1} \end{bmatrix} \ddots \begin{bmatrix} x_{1,1,N} & \dots & x_{1,M,N} \\ \vdots & \vdots & \vdots \\ x_{L,1,N} & \dots & x_{L,M,N} \end{bmatrix}$$
  3rd order tensor.

How do we represent signals as these?

- Sounds, Time Series

$$x^\top = \begin{bmatrix} x_1 & \dots & x_L \end{bmatrix} = [\quad]$$



- Images

$$X = \begin{bmatrix} x_{1,1}, & \dots & x_{1,M} \\ \vdots & \vdots & \vdots \\ x_{L,1} & \dots & x_{L,M} \end{bmatrix} = [\quad]$$

- Videos as tensors.. and so on..

Table of Contents

Index/Array Notation

- We need good ways to communicate operations on these objects.
- **Option 1:** Index Notation
  - Micro-level and detailed, but not very compact
- **Option 2:** Array Notation
  - Compact but abstracts away the details

Index Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c_i = a_i b_i$$

Index Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c_i = a_i b_i$$

  - Inner product of vectors

$$c = \sum_i a_i b_i$$

Index Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c_i = a_i b_i$$

  - Inner product of vectors

$$c = \sum_i a_i b_i$$

  - Outer product of vectors

$$c_{ij} = a_i b_j$$

Index Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c_i = a_i b_i$$

  - Inner product of vectors

$$c = \sum_i a_i b_i$$

  - Outer product of vectors

$$c_{ij} = a_i b_j$$

  - Matrix-vector product

$$c_i = \sum_j A_{ij} b_j$$

Index Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c_i = a_i b_i$$

  - Inner product of vectors

$$c = \sum_i a_i b_i$$

  - Outer product of vectors

$$c_{ij} = a_i b_j$$

  - Matrix-vector product

$$c_i = \sum_j A_{ij} b_j$$

  - Matrix multiplication

$$C_{ik} = \sum_j A_{ij} B_{jk}$$

Index Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c_i = a_i b_i$$

  - Inner product of vectors

$$c = \sum_i a_i b_i$$

  - Outer product of vectors

$$c_{ij} = a_i b_j$$

  - Matrix-vector product

$$c_i = \sum_j A_{ij} b_j$$

  - Matrix multiplication

$$C_{ik} = \sum_j A_{ij} B_{jk}$$

  - Some random tensor operations

$$C_{im} = \sum_{j,l,k} A_{ijlk} B_{mjlk}, \quad c = \sum_{i,j} A_{ij} B_{ij}$$

Array Notation

- We define the elements in index form.
  - Element-wise multiplication:

$$c = a \odot b, \;\; c \in \mathbb{R}^L$$

  - Inner product of vectors

$$c =< a, b >= a^\top b, \;\; c \in \mathbb{R}$$

  - Outer product of vectors

$$c = a \otimes b = ab^\top, \;\; c \in \mathbb{R}^{L \times M}$$

  - Matrix-vector product

$$c = Ab, \;\; c \in \mathbb{R}^L$$

  - Matrix multiplication

$$C = AB, \;\; C \in \mathbb{R}^{L \times M}$$

  - Some random tensor operations

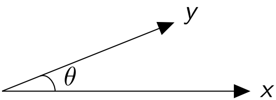
$$C = A \times_{jlk} B, \;\; C \in \mathbb{R}^{L \times M} \;\; c = A \times_{i,j} B, \;\; c \in \mathbb{R}$$

Index vs Array Notation

- Index Notation is very specific, not ambigous
- But the array notation makes it possible to manipulate the operations with ease. (E.g. gradient calculations)

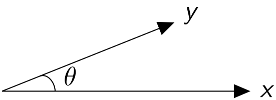
The dot product

- $c = \sum_i a_i b_i = a^\top b = \|a\| \|b\| cos\theta$



The dot product

- $c = \sum_i a_i b_i = a^\top b = \|a\| \|b\| cos\theta$



- Note that,

$$\theta = \arccos \left(\frac{a^\top b}{\|a\| \|b\|} \right)$$

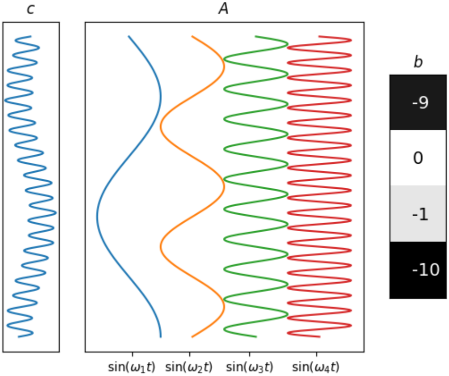
- So, dot product is a great tool to measure similarity.

Matrix-Vector Product

- $c = Ab$, or $c_i =< A_{i,:}, c >= \sum_j A_{ij} c_j$. A is a matrix, b is vector. c is a what?

Matrix-Vector Product

- $c = Ab$, or $c_i = < A_{i,:}, c >= \sum_j A_{ij} c_j$. A is a matrix, b is vector. c is a what?
- The resulting c vector is a linear combination of columns of c.

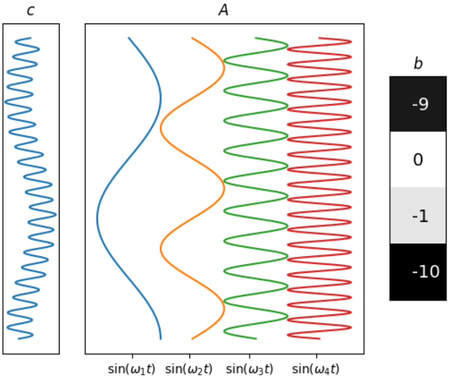


Matrix-Vector Product - 2nd interpretation

- It's a series of dot products.
- $c = Ab$, or $c_i =< A_{i,:}, c >= \sum_j A_{ij} c_j$. A is a matrix, b is vector. c is a what?

Matrix-Vector Product - 2nd interpretation

- It's a series of dot products.
- $c = Ab$, or $c_i =< A_{i,:}, c >= \sum_j A_{ij} c_j$. A is a matrix, b is vector. c is a what?
- The resulting c vector is a linear combination of columns of c.



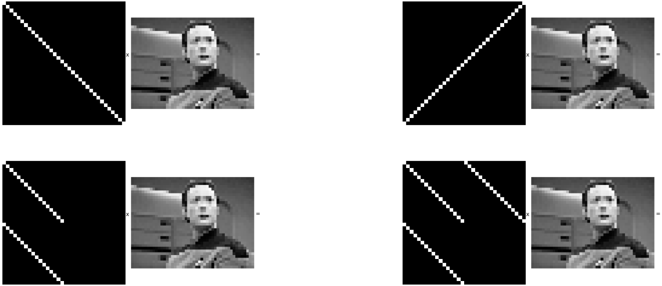
Matrix-Matrix Product

- It's a series of Matrix-vector products. (or series of inner products on a grid)
- $C = AB$, or $C_{ij} = \sum_k A_{ik} C_{kj}$, or $C_{ij} = A_{i,:}^\top C_{:,j}$
- $$C = \begin{bmatrix} A_{1,:}^\top \\ A_{2,:}^\top \\ A_{3,:}^\top \end{bmatrix} \begin{bmatrix} B_{:,1} & B_{:,2} & B_{:,3} \end{bmatrix} = \begin{bmatrix} A_1^\top B_1 & A_1^\top B_2 & A_1^\top B_3 \\ A_2^\top B_1 & A_2^\top B_2 & A_2^\top B_3 \\ A_3^\top B_1 & A_3^\top B_2 & A_3^\top B_3 \end{bmatrix}$$

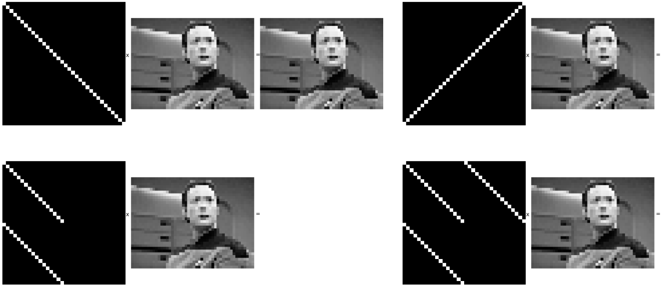
Matrix-Matrix Product

- It's a series of Matrix-vector products. (or series of inner products on a grid)
- $C = AB$, or $C_{ij} = \sum_k A_{ik} C_{kj}$, or $C_{ij} = A_{i,:}^\top C_{:,j}$
- $$C = \begin{bmatrix} A_{1,:}^\top \\ A_{2,:}^\top \\ A_{3,:}^\top \end{bmatrix} \begin{bmatrix} B_{:,1} & B_{:,2} & B_{:,3} \end{bmatrix} = \begin{bmatrix} A_1^\top B_1 & A_1^\top B_2 & A_1^\top B_3 \\ A_2^\top B_1 & A_2^\top B_2 & A_2^\top B_3 \\ A_3^\top B_1 & A_3^\top B_2 & A_3^\top B_3 \end{bmatrix}$$
- Not any pair of two matrices can be multiplied. You need to have equal number of columns from A, number rows from B.
- Master this, it will help! This has to become muscle memory.

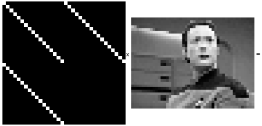
Visualize the matrix product



Visualize the matrix product



Visualize the matrix product

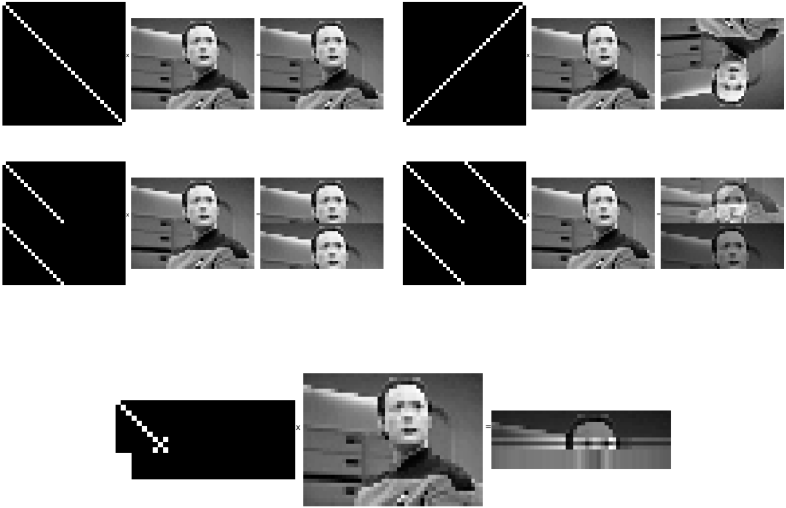


Visualize the matrix product

Visualize the matrix product

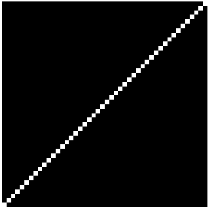
Visualize the matrix product

Visualize the matrix product



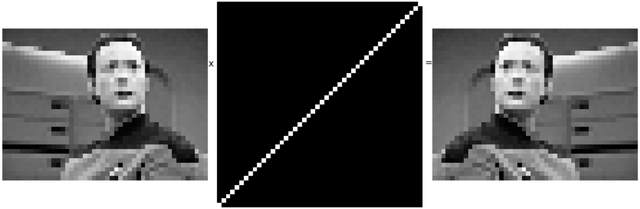
Multiplying from the other side

x



=

Multiplying from the other side



Reversing on the horizontal axis

Einstein Notation

- Let's go beyond matrices!
- $C_{i,j} = \sum_{l,k} A_{i,l,k} B_{l,j,k}$

Einstein Notation

- Let's go beyond matrices!
- $C_{i,j} = \sum_{l,k} A_{i,l,k} B_{l,j,k}$
- How about the Einstein notation?

$$A_{i,l,k}, w_{l,j,k} \to C_{i,j}$$

- You match the indices on the left. Whatever index that does not appear on the right gets summed over.

Einstein Notation

- Let's go beyond matrices!
- $C_{i,j} = \sum_{l,k} A_{i,l,k} B_{l,j,k}$
- How about the Einstein notation?

$$A_{i,l,k}, w_{l,j,k} \rightarrow C_{i,j}$$

- You match the indices on the left. Whatever index that does not appear on the right gets summed over.
- Can you express the matrix multiplication operation with Einstein notation?

Einstein Notation

- Let's go beyond matrices!
- $C_{i,j} = \sum_{l,k} A_{i,l,k} B_{l,j,k}$
- How about the Einstein notation?

$$A_{i,l,k}, w_{l,j,k} \rightarrow C_{i,j}$$

- You match the indices on the left. Whatever index that does not appear on the right gets summed over.
- Can you express the matrix multiplication operation with Einstein notation?
- $A_{i,l}, B_{l,j} \rightarrow C_{i,j}$

Let's do more Einstein stuff

- Element-wise multiplication:

$$c = a \odot b,\ c \in \mathbb{R}^L$$

- Inner product of vectors

$$c =< a, b >= a^\top b,\ c \in \mathbb{R}$$

- Outer product of vectors

$$c = a \otimes b = ab^\top,\ c \in \mathbb{R}^{L \times M}$$

Let's do more Einstein stuff

- Element-wise multiplication:

$$c = a \odot b, \; c \in \mathbb{R}^L$$

$$a_i, b_i \rightarrow c_i$$

- Inner product of vectors

$$c =< a, b >= a^\top b, \; c \in \mathbb{R}$$

$$a_i, b_i \rightarrow c$$

- Outer product of vectors

$$c = a \otimes b = ab^\top, \; c \in \mathbb{R}^{L \times M}$$

$$a_i, b_j \rightarrow c_{i,j}$$

Let's do more Einstein stuff

- Matrix-vector product

$$c = Ab, \; c \in \mathbb{R}^L$$

- Matrix multiplication

$$C = AB, \; C \in \mathbb{R}^{L \times M}$$

- Some random tensor operation

$$C = A \times_{jlk} B, \; C \in \mathbb{R}^{L \times M} \quad c = A \times_{i,j} B$$

Let's do more Einstein stuff

- Matrix-vector product

$$c = Ab,\ c \in \mathbb{R}^L$$

$$A_{i,k}, b_k \rightarrow c_i$$

- Matrix multiplication

$$C = AB,\ C \in \mathbb{R}^{L \times M}$$

$$A_{i,k}, B_{k,j} \rightarrow C_{i,j}$$

- Some random tensor operation

$$C = A \times_{jlk} B,\ C \in \mathbb{R}^{L \times M}\ \ c = A \times_{i,j} B$$

$$A_{ijlk}, B_{mjlk} \rightarrow C_{im}$$

Implementing Einstein products is easy in Python

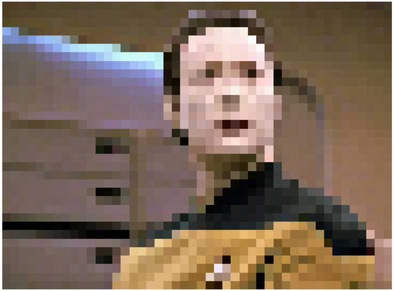
- Batch Matrix Multiplication

$$A_{bij}B_{bjk} \to C_{bik}$$

```
C = torch.einsum('bij,bjk->bik', A, B)
```

Application of Tensor Operations

- RGB images



- Let us apply a matrix multiplication to each channel, and then average over the channels.

Application of Tensor Operations

- In Index Notation

$$C_{ij} = \sum_{k,c} \underbrace{B_{ik}}_{\text{Matrix}} \underbrace{A_{kjc}}_{\text{image}} \underbrace{w_c}_{\text{WtOverCh.}}$$

- Notice that this notation can handle multilinear operations.

Application of Tensor Operations

- In Index Notation

$$C_{ij} = \sum_{k,c} \underbrace{B_{ik}}_{\text{Matrix}} \underbrace{A_{kjc}}_{\text{image}} \underbrace{w_c}_{\text{WtOverCh.}}$$

- In Einstein Notation:

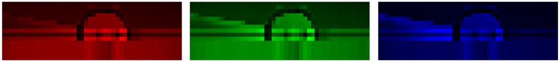
$$B_{ik}, A_{kjc}, w_c \to C_{ij}$$

- Notice that this notation can handle multilinear operations.

Application of Tensor Operations

- First step

$$B_{ik}, A_{kjc} \rightarrow T_{ijc}$$



- Second step

$$T_{ijc} w_c \rightarrow C_{ij}$$

Let's also see some reshaping operations

- Vectorization:

$$\text{vec}\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{12} \\ a_{22} \end{bmatrix}$$

- The 'Diag' Operation:

$$\text{Diag}\left(\begin{bmatrix} a_1 & a_2 \end{bmatrix}\right) = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

- The 'Reshape' Operation:

$$\text{Reshape}_{32}\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}\right) = \begin{bmatrix} a_{11} & a_{22} \\ a_{21} & a_{13} \\ a_{12} & a_{23} \end{bmatrix}$$

Kronecker Product

- It's sort of an outer product but has a specific shape,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix}$$

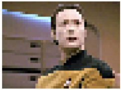
Kronecker Product

- It's sort of an outer product but has a specific shape,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix}$$

- Let's visualize this,

$$\begin{bmatrix} 1 & 4 \\ 0 & 2 \end{bmatrix} \otimes$$



$$=$$

Why Bother?

- Sometimes matrix algebra is compact and powerful.

Why Bother?

- Sometimes matrix algebra is compact and powerful.
- For instance, check this out:

$$C = \left(\operatorname{diag}\left(\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\right) \otimes I \otimes I\right)\operatorname{vec}(A)$$

Why Bother?

- Sometimes matrix algebra is compact and powerful.
- For instance, check this out:

$$C = \Big(\operatorname{diag}\left(\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\right) \otimes I \otimes I\Big)\operatorname{vec}(A)$$

- This is equivalent to:

$$A_{ijc}, w_c \to C_{ij}$$

Why Bother?

- Sometimes matrix algebra is compact and powerful.
- For instance, check this out:

$$C = \Big(\text{diag}\left(\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\right) \otimes I \otimes I\Big)\text{vec}(A)$$

- This is equivalent to:

$$A_{ijc}, w_c \to C_{ij}$$

- The matrix form could be helpful when calculating gradients, and coming up with efficient implementations.
- Einsum is not as optimized as matrix multiplication.

Table of Contents

Matrix inverse

- Let's think about a linear system,

$$\begin{aligned} Ax =& b \\ \rightarrow A^{-1}Ax =& x = A^{-1}b \end{aligned}$$

- Is A^{-1} always defined?

Matrix inverse

- Let's think about a linear system,

$$\begin{aligned} Ax =& b \\ \rightarrow A^{-1}Ax =& x = A^{-1}b \end{aligned}$$

- Is A^{-1} always defined?
- First, A needs to be square.
- Second, it needs to be full rank. Columns of A need to be linearly independent.

Matrix pseudoinverse

- Let's have the same linear system, but with a rectangular A matrix,

$$Ax = b$$

Matrix pseudoinverse

- Let's have the same linear system, but with a rectangular A matrix,

$$Ax = b$$

- We can not inverse A. However we can multiply from the left with $A^\top$,

$$\begin{aligned} A^\top A x &= A^\top b \\ \rightarrow (A^\top A)^{-1} A^\top A x = x &= \underbrace{(A^\top A)^{-1} A^\top}_{:= A^\dagger} b \end{aligned}$$

Matrix pseudoinverse

- Let's have the same linear system, but with a rectangular A matrix,

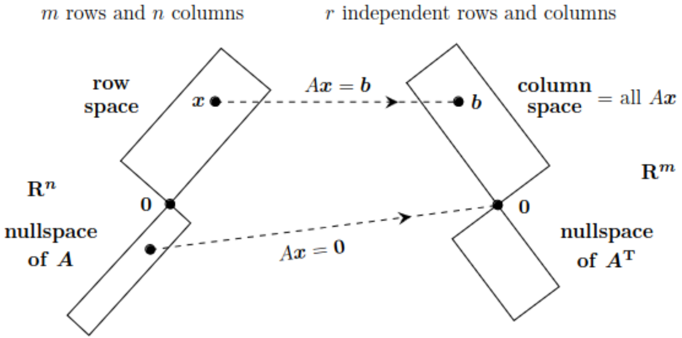
$$Ax = b$$

- We can not inverse A. However we can multiply from the left with $A^\top$,

$$\begin{aligned} A^\top Ax &= A^\top b \\ \rightarrow (A^\top A)^{-1} A^\top Ax = x &= \underbrace{(A^\top A)^{-1} A^\top}_{:= A^\dagger} b \end{aligned}$$

- $A^\dagger := (A^\top A)^{-1} A^\top$. This is known as the pseudo inverse.
- This is essentially least squares. (We will show that later)

Four Fundamental Subspaces in Linear Algebra



BIG PICTURE OF LINEAR ALGEBRA

row space ⊥ nullspace   column space of A ⊥ nullspace of A^T

row rank = column rank = r

Image Taken from Gilbert Strang's 'Introduction to Linear Algebra' book.

Norms, trace

- l_2 norm: $\|x\|_2 = \sqrt{\sum_j x_j^2}$. Also known as Euclidean Norm.
- l_1 norm: $\|x\|_1 = \sum_j |x_j|$.
- l_p norm: $\|x\|_p = \sqrt[p]{\sum_j |x_j|^p}$.
- $\mathrm{tr}(A) = \sum_i A_{ii}$, it's basically the sum of diagonal elements.
  Do not underestimate this.

Norms, trace

- l_2 norm: $\|x\|_2 = \sqrt{\sum_j x_j^2}$. Also known as Euclidean Norm.
- l_1 norm: $\|x\|_1 = \sum_j |x_j|$.
- l_p norm: $\|x\|_p = \sqrt[p]{\sum_j |x_j|^p}$.
- $\mathrm{tr}(A) = \sum_i A_{ii}$, it's basically the sum of diagonal elements.
  Do not underestimate this.
- Frobenius norm: $\|X\|_F = \sqrt{\sum_i \sum_j |X_{ij}|^2} = \sqrt{\mathrm{tr}(XX^\top)}$

Matrix Calculus

- $\frac{df(x)}{dx}$, gradient of a scalar wrt to a scalar is a scalar.

$$\frac{dx^2}{dx} = 2x$$

Matrix Calculus

- $\frac{df(x)}{dx}$, gradient of a scalar wrt to a scalar is a scalar.

$$\frac{dx^2}{dx} = 2x$$

- Gradient of a scalar function wrt to a vector is a vector.

$$\frac{db^\top Ax}{dx} = A^\top b$$

Matrix Calculus

- $\frac{df(x)}{dx}$, gradient of a scalar wrt to a scalar is a scalar.

$$\frac{dx^2}{dx} = 2x$$

- Gradient of a scalar function wrt to a vector is a vector.

$$\frac{db^\top Ax}{dx} = A^\top b$$

- Gradient of a vector wrt to a vector is a matrix

$$\frac{dAx}{dx} = A^\top$$

Matrix Calculus

- $\frac{df(x)}{dx}$, gradient of a scalar wrt to a scalar is a scalar.

$$\frac{dx^2}{dx} = 2x$$

- Gradient of a scalar function wrt to a vector is a vector.

$$\frac{db^\top Ax}{dx} = A^\top b$$

- Gradient of a vector wrt to a vector is a matrix

$$\frac{dAx}{dx} = A^\top$$

- Gradient of a vector wrt to a matrix is ?

Matrix Calculus

- $\frac{df(x)}{dx}$, gradient of a scalar wrt to a scalar is a scalar.

$$\frac{dx^2}{dx} = 2x$$

- Gradient of a scalar function wrt to a vector is a vector.

$$\frac{db^\top Ax}{dx} = A^\top b$$

- Gradient of a vector wrt to a vector is a matrix

$$\frac{dAx}{dx} = A^\top$$

- Gradient of a vector wrt to a matrix is ?
- Index notation helps to derive these. Otherwise you can just pattern match from the matrix cookbook.

Matrix Calculus

- $\frac{df(x)}{dx}$, gradient of a scalar wrt to a scalar is a scalar.

$$\frac{dx^2}{dx} = 2x$$

- Gradient of a scalar function wrt to a vector is a vector.

$$\frac{db^\top Ax}{dx} = A^\top b$$

- Gradient of a vector wrt to a vector is a matrix

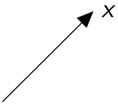
$$\frac{dAx}{dx} = A^\top$$

- Gradient of a vector wrt to a matrix is ?
- Index notation helps to derive these. Otherwise you can just pattern match from the matrix cookbook.
- We are just giving an idea here with simple examples. We will see these more in real action later. (hint: backprop)

Table of Contents

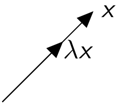
Eigenvalues / Eigenvectors

- $Ax = \lambda x$



Eigenvalues / Eigenvectors

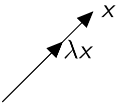
- $Ax = \lambda x$



- Note that x doesn't change its direction.

Eigenvalues / Eigenvectors

- $Ax = \lambda x$



- Note that x doesn't change its direction.
- Eigenvectors are 'characteristic' directions for the system described by A.

Finding the Eigenvectors

- The 'Linear Algebra Class Way':
- Let's have this matrix

$$A = \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix}$$

- Calculate the determinant (why?)

$$\det(A - \lambda I) = \begin{vmatrix} 0.8 - \lambda & 0.4 \\ 0.2 & 0.6 - \lambda \end{vmatrix} = \lambda^2 - 1.4\lambda + 0.40$$

Finding the Eigenvectors

- The 'Linear Algebra Class Way':
- Let's have this matrix

$$A = \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix}$$

- Calculate the determinant (why?)

$$\det(A - \lambda I) = \begin{vmatrix} 0.8 - \lambda & 0.4 \\ 0.2 & 0.6 - \lambda \end{vmatrix} = \lambda^2 - 1.4\lambda + 0.40$$

- Solve the characteristic equation for 0. $\lambda_1 = 1$, $\lambda_2 = 0.4$
- Then we find vectors in the null space of $A - \lambda I$

Finding the Eigenvectors

- The 'Linear Algebra Class Way':
- Let's have this matrix

$$A = \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix}$$

- Calculate the determinant (why?)

$$\det(A - \lambda I) = \begin{vmatrix} 0.8 - \lambda & 0.4 \\ 0.2 & 0.6 - \lambda \end{vmatrix} = \lambda^2 - 1.4\lambda + 0.40$$

- Solve the characteristic equation for 0. $\lambda_1 = 1$, $\lambda_2 = 0.4$
- Then we find vectors in the null space of $A - \lambda I$
- $A - I = \begin{bmatrix} -0.2 & 0.4 \\ 0.2 & -0.4 \end{bmatrix} v = 0$, find a non-zero vector v such that the equation is satisfied. $v = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

Finding the Eigenvectors

- For big matrices the method is untractable.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Finding the Eigenvectors

- For big matrices the method is untractable.
- But eigenvectors are attractor points. The recursion $v_{k+1} = \frac{Av_k}{\|Av_k\|}$ gets you the eigenvectors.
- Here are the power iterations starting from $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

- To get all the eigenvectors we can deflate the matrix. Just subtract v, and repeat the process..

Ok, but how is this a decomposition?

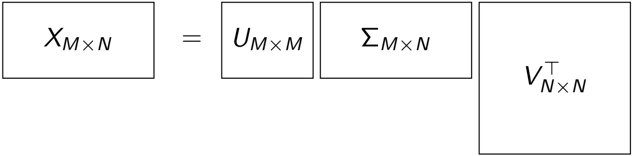
- $AV = V\Lambda$, where columns of V are the eigenvectors, and Λ is a diagonal matrix with eigenvalues on the diagonal.

Ok, but how is this a decomposition?

- $AV = V\Lambda$, where columns of V are the eigenvectors, and Λ is a diagonal matrix with eigenvalues on the diagonal.
- And here's the decomposition $A = V\Lambda V^{-1}$.
- But notice that this decomposition is only defined for square matrices.

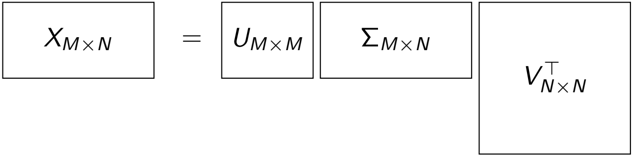
Singular Value Decomposition

- Let us given a matrix of size X in $\mathbb{R}^{M\times N}$.
- $X = U\Sigma V^\top$, $U \in \mathbb{R}^{M\times M}$ and is orthogonal $U^\top U = I$, $\Sigma \in \mathbb{R}^{M\times N}$ is a matrix with non-zero elements on the main diagonal, and $V \in \mathbb{R}^{N\times N}$, and is orthonal $VV^\top = I$.

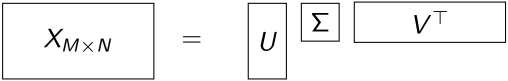


Singular Value Decomposition

- Let us given a matrix of size X in $\mathbb{R}^{M \times N}$.
- $X = U \Sigma V^\top$, $U \in \mathbb{R}^{M \times M}$ and is orthogonal $U^\top U = I$, $\Sigma \in \mathbb{R}^{M \times N}$ is a matrix with non-zero elements on the main diagonal, and $V \in \mathbb{R}^{N \times N}$, and is orthonal $VV^\top = I$.



- An alternative way of viewing it is $X = \sum_{k=1}^{M} \sigma_k u_k v_k^\top$. Note that we can cut the sum short, and keep the biggest singular values! (set $X = \sum_{k=1}^{K} \sigma_k u_k v_k^\top$, $K \leq M$)



Relationship between SVD and Eigenvalue decomposition

- $X = U\Sigma V^\top$, this is SVD.

Relationship between SVD and Eigenvalue decomposition

- $X = U\Sigma V^\top$, this is SVD.
- $XX^\top = U\Sigma \underbrace{V^\top V}_{I} \Sigma^\top U^\top = U\Sigma^2 U^\top$.

Relationship between SVD and Eigenvalue decomposition

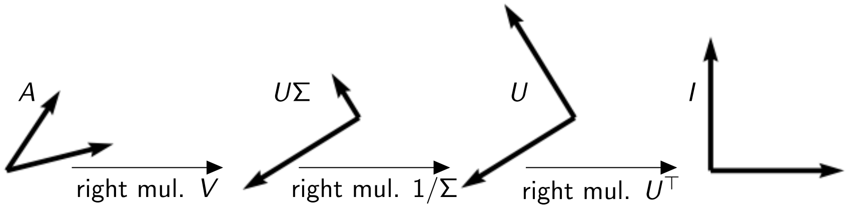
- $X = U\Sigma V^\top$, this is SVD.
- $XX^\top = U\Sigma \underbrace{V^\top V}_{I} \Sigma^\top U^\top = U\Sigma^2 U^\top$.
- Singular vectors U of X, are the eigenvectors of $XX^\top$.
- Singular values σ_k of X, are the square root of eigenvalues of $XX^\top$.

Relationship between SVD and Eigenvalue decomposition

- $X = U\Sigma V^\top$, this is SVD.
- $XX^\top = U\Sigma \underbrace{V^\top V}_{I} \Sigma^\top U^\top = U\Sigma^2 U^\top$.
- Singular vectors U of X, are the eigenvectors of $XX^\top$.
- Singular values σ_k of X, are the square root of eigenvalues of $XX^\top$.

- For positive semi-definite matrices, SVD and eigenvalue decomposition are equivalent.

Geometric Interpretation of SVD



List of Decompositions

- **LU decomposition:** $X = LU$, L is lower triangular, U is upper triangular.
- **QR decomposition:** $X = QR$, Q is a matrix with orthonormal columns, R is an upper triangular matrix.
- **Eigenvalue decomposition:** $X = U\Lambda U^{-1}$, columns of U are eigenvalues of X, which is square (diagonalizable) matrix.
- **Singular value decomposition:** $X = U\Sigma V^\top$, columns of U, and V have orthonormal columns. Defined for any matrix.

List of Decompositions

- **LU decomposition:** $X = LU$, L is lower triangular, U is upper triangular.
- **QR decomposition:** $X = QR$, Q is a matrix with orthonormal columns, R is an upper triangular matrix.
- **Eigenvalue decomposition:** $X = U\Lambda U^{-1}$, columns of U are eigenvalues of X, which is square (diagonalizable) matrix.
- **Singular value decomposition:** $X = U\Sigma V^\top$, columns of U, and V have orthonormal columns. Defined for any matrix.
- There's more, e.g. Cholesky, NMF, CR, ICA, ...

List of special type of matrices we'll see in this class

- **Rotation matrices**
- **Markov matrices** (Probability Transition Matrices)
- **Transform matrices** (Fourier Transform, Convolution,...)
- **Covariance matrices** (Define a Multivariable Random Variable)
- **Adjacency matrices** (Define a Graph)

Recap

- We saw how data/signals can be represented.

Recap

- We saw how data/signals can be represented.

- We saw how the data can be manipulated. (Vector, Matrix, Tensor Operations)

Recap

- We saw how data/signals can be represented.

- We saw how the data can be manipulated. (Vector, Matrix, Tensor Operations)

- We took a glimpse into how we can decompose signals.

Recap

- We saw how data/signals can be represented.

- We saw how the data can be manipulated. (Vector, Matrix, Tensor Operations)

- We took a glimpse into how we can decompose signals.
- We gave a crude summary into what we need from Linear Algebra.

Recommended Reading

- Gilbert Strang, Introduction to Linear Algebra, https://ocw.mit.edu/courses/18-06-linear-algebra-spring-2010/video_galleries/video-lectures/, https://math.mit.edu/~gs/linearalgebra/ila5/indexila5.html
- Trefethen and Bau, Numerical Linear Algebra, https://people.maths.ox.ac.uk/trefethen/text.html
- Matrix Cookbook, http://www2.imm.dtu.dk/pubdb/edoc/imm3274.pdf

What's Next

- Probability Calculus, Random Variables, Multi-dimensional Distributions
- Exponential Family Distributions
- Maximum Likelihood, MAP, Bayesian parameter estimation principles
- Labs are starting next week! (first one is Sept. 15)