Computational Intelligence Laboratory

Lecture 1: Course Overview & Linear Algebra Primer

Thomas Hofmann

TAs: Martin Jaggi & Aurelien Lucchi

Besmira Nushi, Octavian Ganea, Hadi Daneshmand

Dwarikanath Mahapatra & Dmitry Laptev

ETH Zurich – `cil.inf.ethz.ch`

20 February 2015

CIL Team

Week 1-5

Besmira Nushi

Aurelien Lucchi

Week 6-9

Martin Jaggi

Dwarikanath Mahapatra

Week 10-13

Hadi Daneshmand

Octavian Ganea

Machine Learning Laboratory

Credit

- Laboratory has been developed by Prof. Joachim Buhmann and various research associates and assistants
- Some changes and amendments have been made (by TH) compared to 2014, yet the overall structure and the majority of material has been kept.

Table of Contents

Organization

Overview

- Course Information
- Theory
- Applications

Linear Algebra Primer

- Vectors and Matrices
- Vector Spaces
- Range and Nullspace
- Basis and Dimension
- Inverse and Eigenvalues

Learning Goals

- Acquire fundamental **theoretical concepts** of machine learning related to **matrix factorization**.
- Implement and compare **alternative techniques** to solve the same application problem:

Collaborative Filtering

Inpainting

Compression

- Combine and extend techniques $\Rightarrow$ **novel solution**
  - compare to baselines developed during the course
- Produce a **write up** of your findings = scientific short paper
  - we will emphasize an ethically sound way to conduct research.

Course Structure

Method/application matrix structure:

- basic methods are useful for several applications
- moderate adaption to use case (e.g. in class projects)

Applications

Theory	Collaborative Filtering	Inpainting	Image Compression
Dimension Reduction	✔	✔	✔
Clustering	✔	✔	✔
Sparse Coding		✔	✔
Robust PCA			✔

Organizational Form

Weekly Schedule: 2 + 2 + 1

- Lecture: Fri 10-12, CAB G61
- Exercises: Thu 15-17, CAB G51 **or** Thu 16-18, CAB G61 **or** Fri 15-17, CAB G61
  - all three exercise sessions are "identical"
  - first hour: pen-and-paper **exercise**, immediate discussions
  - second hour: group work on programming **assignment**
  - we reserve the right to "load balance", if the distribution is highly skewed
- Voluntary presence time: Mo 11-12, CAB H53
  - TAs help completing programming assignments
- **Webpage:** `http://cil.inf.ethz.ch/`
  - only accessible from the ETH network

Organizational Form

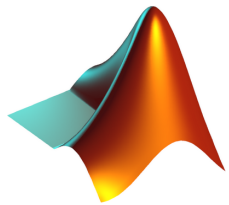
Programming Assignments

- Joint work in groups of **two or three** students
- Solving problems by applying techniques learned in class
- Submitting solution to the CIL online evaluation and ranking system

Been here before? :-/

- You can re-submit previous year's project. The competitive part will be re-graded!
- You have to let us know that you are a "one person group"
- You have the choice to redo the project and join a group (as anyone else)

First Week



Coming week: exercise session on efficient **Matlab** programming

- Bring laptops with Matlab installed
- "Hands on" exercises in groups of 2-3 students

Reading material

- Linear algebra background – on the course website

Grading Criteria

Semester Work

- Develop a **novel solution** for one of the application problems
- **Compare** with two baseline techniques already implemented in the weekly programming assignments
- Write up in the form of a short **paper**
- **Competitive criteria:** run-time, accuracy, ..
- **Non-competitive criteria:** paper review, creativity of solution, quality of implementation

Written Exam

- 120 minutes written exam
- up to 5 problems in the spirit of the pen-and-paper exercises

Grading Formula

You need to satisfy two **requirements** to pass this course:

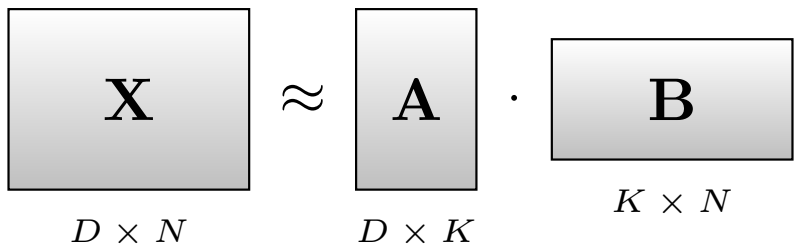
1. Your **average grade** (2/3 written exam, 1/3 group project) is greater or equal to 4.
2. Your **written exam grade** is greater or equal to 3.5.

Therefore, your **final grade** will be:

1. Your average grade, if your written exam grade is greater or equal to 3.5.
2. Your written exam grade if it is below 3.5.

Matrix Factorization

Core problem: **matrix factorization** (MF) of a given data matrix into two or three unknown matrices, e.g.



Many important data analysis techniques can be written as MF problems by specifying:

- **Type of data:** e.g. Boolean: $x_{d,n} \in \{0, 1\}$ or non-negative real: $x_{d,n} \in \mathbb{R}_+$, i.e., $x_{d,n} \geq 0$
- **Approximation:** e.g. exact: $\mathbf{X} = \mathbf{A} \cdot \mathbf{B}$ or minimal Frobenius norm: $\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X} - \mathbf{A} \cdot \mathbf{B}\|_F$
- **Constraints on unknown matrices:** e.g. $\mathbf{A}$ has to be a basis or $\mathbf{B}$ must be sparse.

Matrix Factorization

For many variants, MF is computationally NP-hard, therefore we need **efficient** and **low-error** approximation algorithms.

Techniques studied in this course:

1. Dimension reduction
2. Clustering
3. Sparse coding
4. Robust PCA

Theory Part I: Dimension Reduction

Given N vectors in D dimensions, find the K most important axes to project them.

Why dimensionality reduction?

- some features may be irrelevant, projection may reduce noise
- visualize high dimensional data, e.g., in 2 or 3 dimensions.
- "intrinsic" dimensionality may be smaller than D

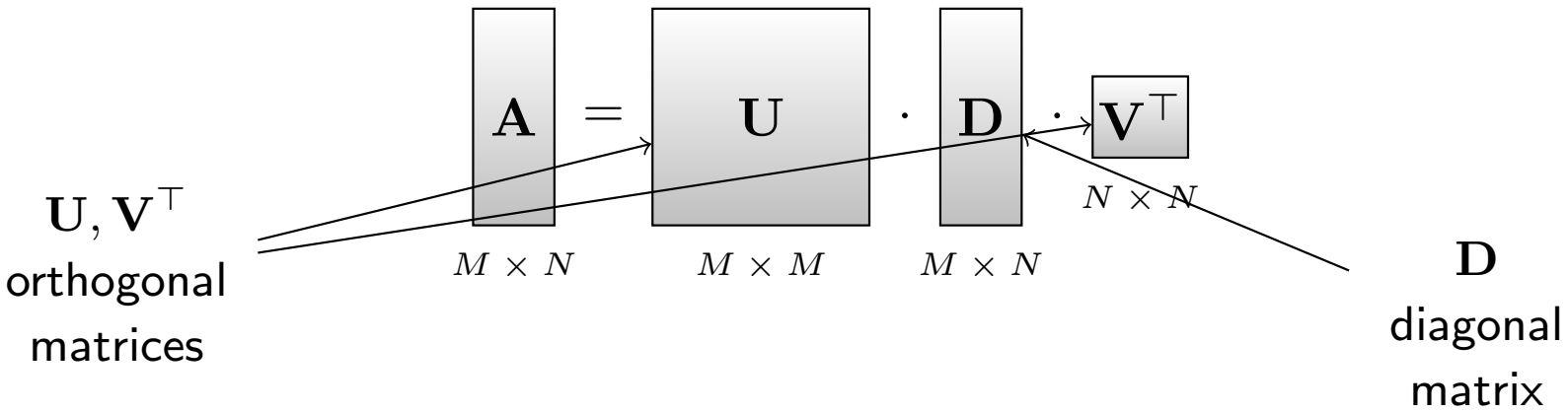
Many important dimensionality reduction techniques are based on the **eigen-decomposition** of the data matrix.

In this course we review the singular value decomposition (SVD) and principal component analysis (PCA).

Theory Part I: Dimension Reduction

SVD

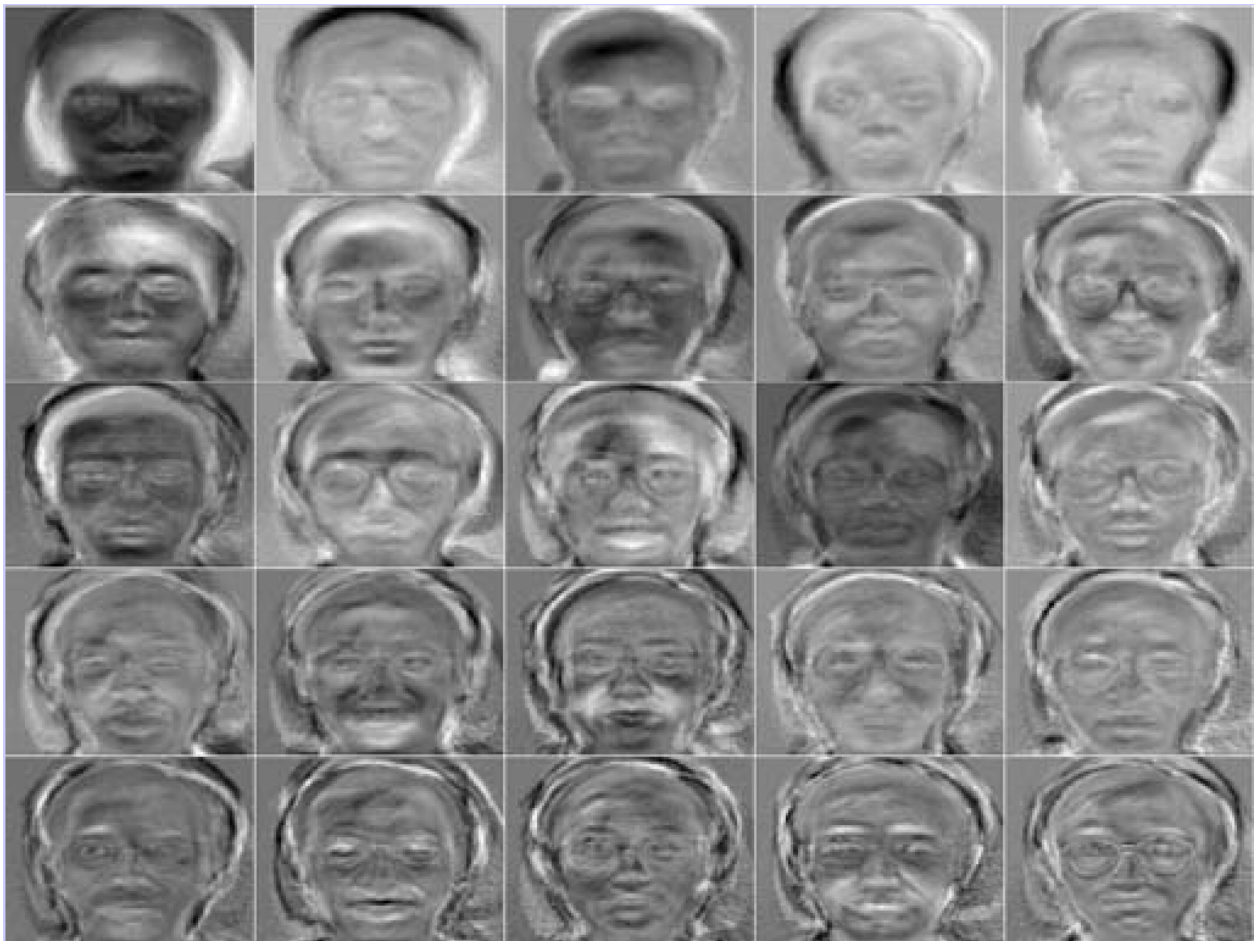
Any matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ can be decomposed **uniquely** as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ (up to permutations and signs):



Eckart-Young Theorem

Let $\mathbf{A}$ be a matrix of rank R, if we wish to approximate $\mathbf{A}$ using a matrix of a lower rank K then, $\tilde{\mathbf{A}} = \sum_{k=1}^{K} d_k \mathbf{u}_k \mathbf{v}_k^\top$ is the closest matrix in the Frobenius norm. (Assumes ordering of singluar values $d_k \geq d_{k+1}$)

Theory Part I: Dimension Reduction

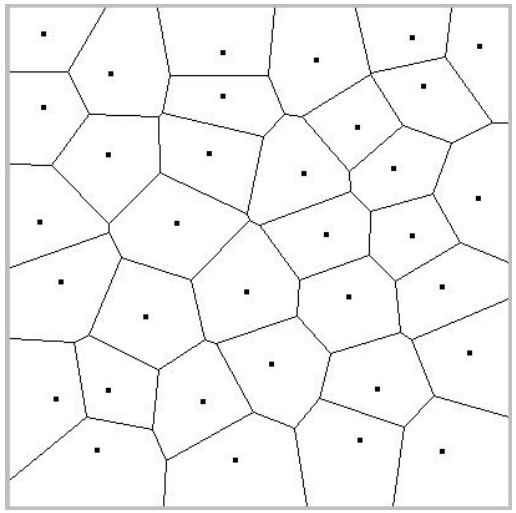


Christopher DeCoro `http://www.cs.princeton.edu/~cdecoro/eigenfaces`

"Eigenfaces": basis vectors produced by SVD on a set of face images

Theory Part II: Clustering

Quantization of the space $\mathbb{R}^D$



- data points $\mathbf{x} \in \mathbb{R}^D$ are represented by K **centroids** $\mathbf{u}_k$
- assign $\mathbf{x}$ to the centroid $\mathbf{u}_{k^*}$ that has minimal distance to $\mathbf{x}$.
- induces Voronoi tesselation of $\mathbb{R}^D$

Theory Part II: Clustering for Color Quantization

Figure : Top: original image, Bottom: same image where each pixel is represented by one of 10 colors.

Theory Part II: Clustering

K-means algorithm

Alternate between two steps until convergence:

1. Update assignments $z_{k,n}$ of data points to centroids:

$$z_{k,n} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|\mathbf{x}_n - \mathbf{u}_j\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

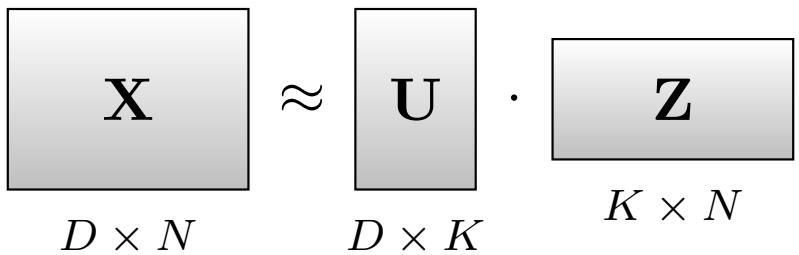
2. Update centroid positions:

$$\mathbf{u}_k = \frac{\sum_n z_{k,n} \mathbf{x}_n}{\sum_n z_{k,n}}.$$

Theory Part II: Clustering

K-means clustering formulated as matrix factorization

Assign each datapoint (column of $\mathbf{X}$) to one of K clusters:

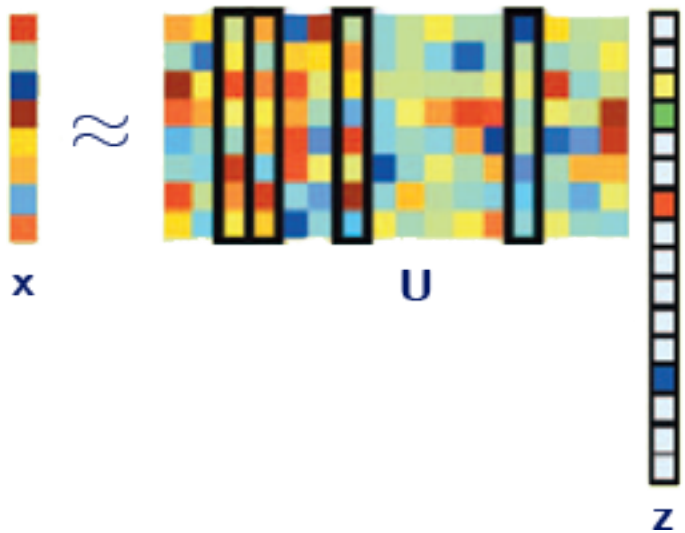


with $\mathbf{X} \in \mathbb{R}^{D \times N}$, $\mathbf{U} \in \mathbb{R}^{D \times K}$ and $\mathbf{Z} \in \{0, 1\}^{K \times N}$. In addition we enforce the unique assignment constraint $\sum_k z_{k,n} = 1 \ (\forall n)$.

The centroids define the columns of $\mathbf{U}$.

Theory Part III: Sparse Coding

Approximate representation of a signal $\mathbf{x} \in \mathbb{R}^D$ by a **sparse** linear combination $\mathbf{z} \in \mathbb{R}^L$ of K atoms of a **dictionary** $\mathbf{U} \in \mathbb{R}^{D \times L}$:



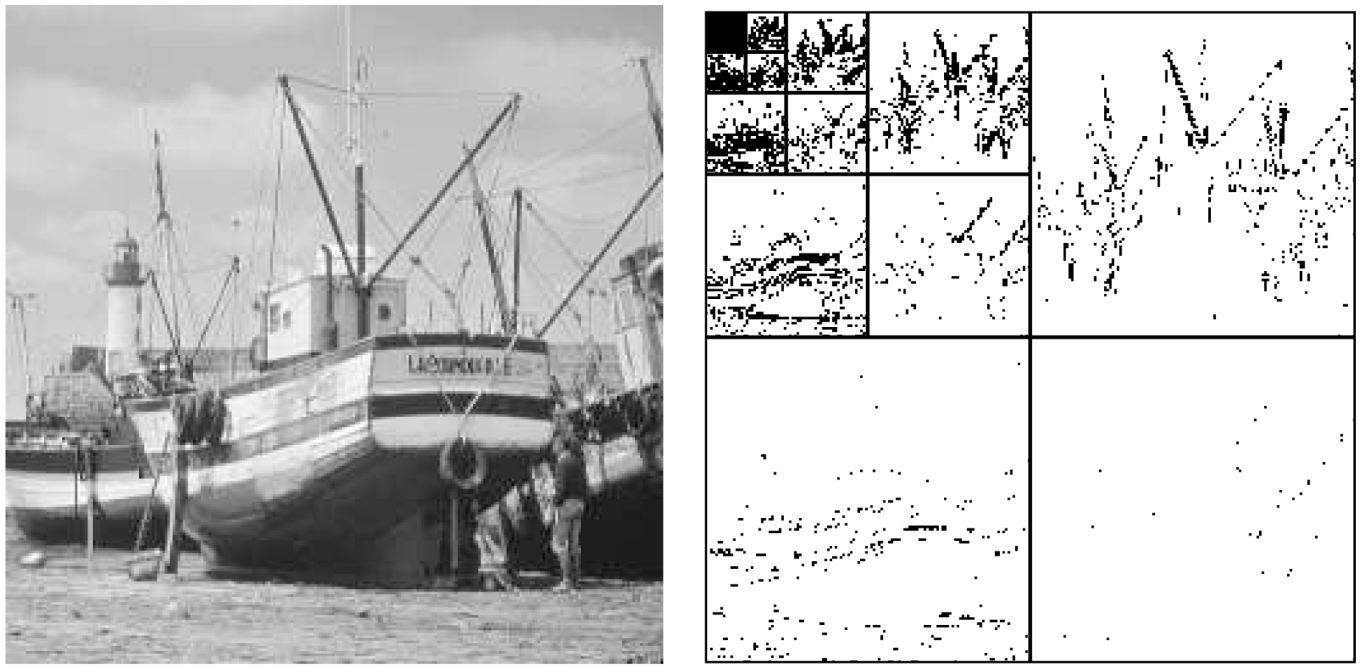
Formally, minimize the number of non-zero coefficients

$$\min_{\mathbf{z}} \|\mathbf{z}\|_0 , \quad \text{s.t.} \quad \|\mathbf{U}\mathbf{z} - \mathbf{x}\|_2 < \sigma,$$

while keeping the approximation error below σ.

Theory Part III: Sparse Coding

Representation of grayscale image in a **wavelet** dictionary:



Only few coefficients are above a certain magnitude threshold (black dots in right figure). Efficient **compression** approach: keep large coefficients, disregard small ones.

Theory Part III: Dictionary Learning as Matrix Factorization

Can we find even sparser representations? Yes, by adapting $\mathbf{U}$ to the image.

Dictionary learning: Find dictionary $\mathbf{U}$ and coding $\mathbf{Z}$ by solving

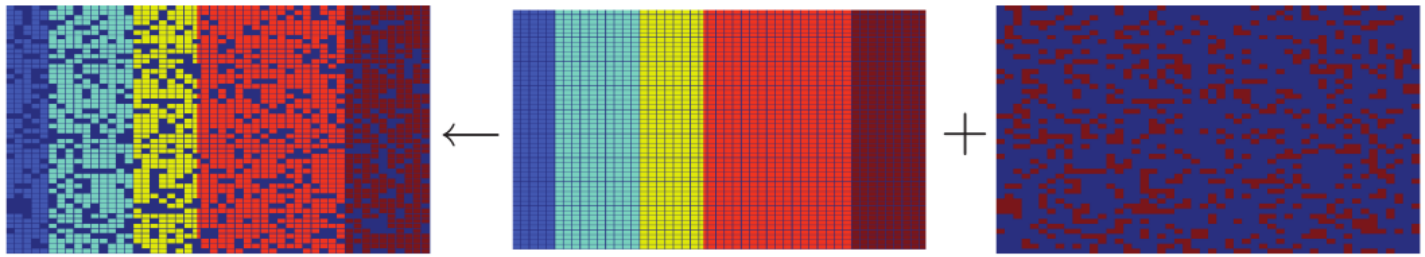
$$\min_{\mathbf{U},\mathbf{Z}} \|\mathbf{X} - \mathbf{U} \cdot \mathbf{Z}\|_F^2 ,$$

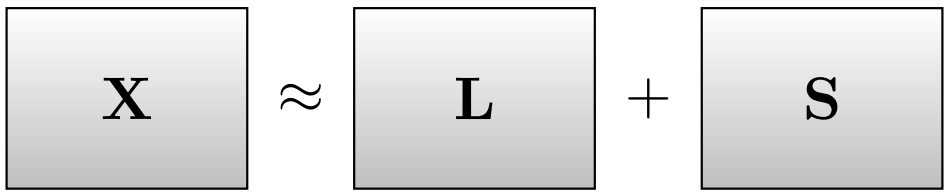
where $\mathbf{Z}$ must be sparse.

- Problem can be made convex in $\mathbf{U}$ or $\mathbf{Z}$, but not in both arguments.
- Simple algorithm: iterate between optimizing $\mathbf{U}$ and $\mathbf{Z}$.

Theory Part IV: Robust PCA

Separate a matrix $\mathbf{X}$ into *low-rank* $\mathbf{L}$ and *sparse* $\mathbf{S}$?





- Highly corrupted measurements ($\|\mathbf{S}\|_2$ can be big).
- Number of corrupted measurements ($\|\mathbf{S}\|_0$) should be small.

Theory Part IV: Robust PCA

Corrupted data in computer vision

Figure : Sunglasses or shadows occluding the face; old films

Figure : Decompose a video into static background and activity in the foreground

Theory Part IV: Robust PCA

Principal Component Pursuit

$$\begin{aligned} &\text{minimize} && \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \\ &\text{subject to} && \mathbf{L} + \mathbf{S} = \mathbf{X} \end{aligned}$$

Unobserved values

	Ben	Tom	John	Fred	Jack
Star Wars	?	?	1	?	4
WallE	5	?	3	4	?
Avatar	3	4	?	4	4
Trainspotting	?	1	5	?	?
Shrek	5	?	?	5	?
Ice Age	5	?	4	?	1

- Robust PCA: can also be applied to matrix $\mathbf{X}$ with unknown values
- $\Rightarrow$ collaborative filtering in recommender systems

Theory and Application

Group Project Example

	Collaborative Filtering	Inpainting	Compression
Dim. Reduction	**o**		✓
Clustering			✓
Robust PCA			
Sparse Coding			

Programming assignments: You implemented several solutions for each application, e.g. image compression by dimension reduction and by clustering (✓ in the matrix).

Group project: Develop a novel solution for one application, e.g. by transferring a technique to this new application domain (**o** in the matrix).

Lab Application I: Collaborative Filtering

Recommender system: present items that are of likely interest to the user.

- Products: Amazon
- Movies: MovieLens, IMDB
- Music: LastFM

In collaborative filtering (CF), we base our recommendations:

- On the (known) preference of the user towards other items,
- Take into account the preferences of other users.

Lab Application I: Collaborative Filtering

Viewers were asked to rate some movies (items):

	Ben	Tom	John	Fred	Jack
Star Wars	?	?	1	?	4
WallE	5	?	3	4	?
Avatar	3	4	?	4	4
Trainspotting	?	1	5	?	?
Shrek	5	?	?	5	?
Ice Age	5	?	4	?	1

- Not all viewers rated all movies.
- Unrated user-movie pairs are **missing values**: we want to predict them.
- Should we recommend Fred to watch "Ice Age"?

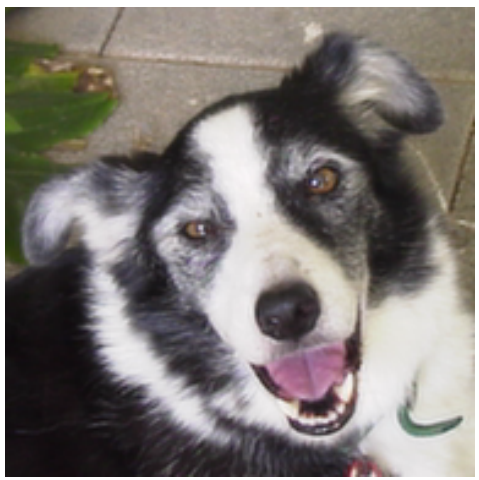
Lab Application II: Inpainting

Inpainting aims at the **restoration of missing pixels**, based on the intact surroundings:

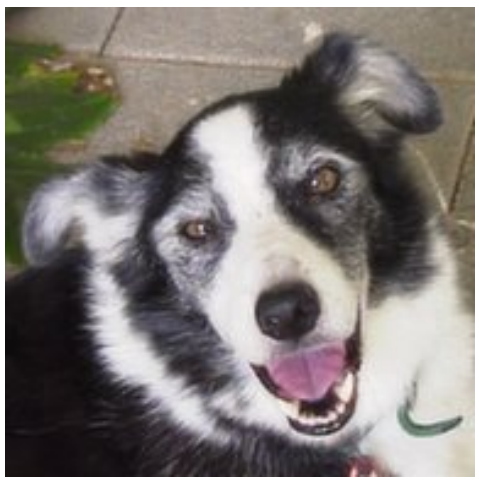
As with collaborative filtering, the missing values are predicted based on the statistics of known pixels of the target image, and possibly other images, too.

Lab Application III: Compression

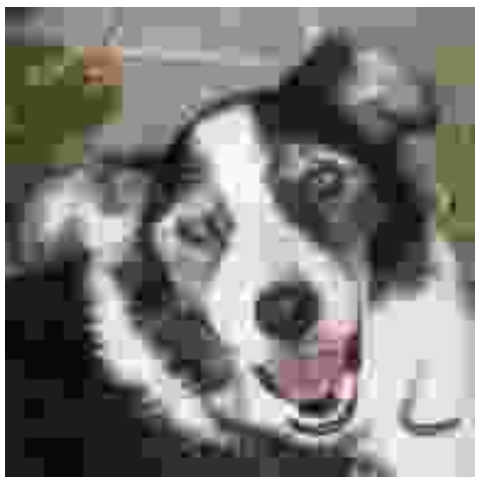
- Encode information using fewer bits than original representation
- **Lossy compression**: loss between original and compressed version is acceptable according to some loss function
- Achieve this by neglecting irrelevant parts of the data
  - noise vs. signal
- Here: Compression of images



Original (61 KB)



Low Compression (10 KB)



High Compression (2 KB)

Table of Contents

Organization

Overview

Course Information

Theory

Applications

Linear Algebra Primer

Vectors and Matrices

Vector Spaces

Range and Nullspace

Basis and Dimension

Inverse and Eigenvalues

Notation

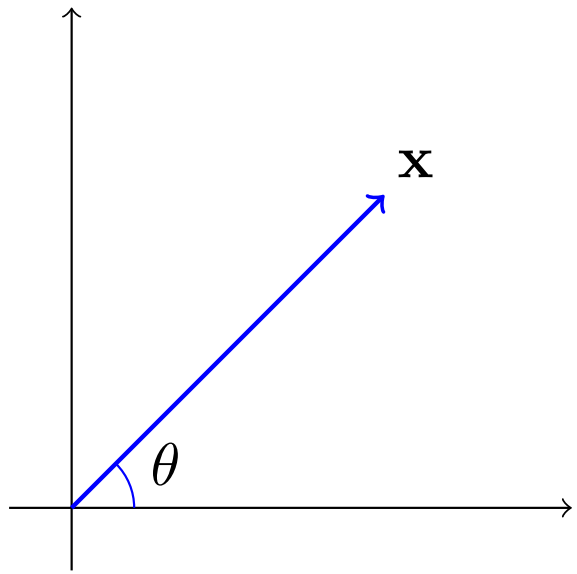
- $\mathbf{x}$ is a column vector
- $^\top$ denotes the **transpose** operator, so $\mathbf{x}^\top$ is a row vector
- The elements of a vector are denoted as $\mathbf{x} = (x_1, x_2, \ldots, x_D)^\top$
- $\mathbf{U}$ is a matrix:
  - $\mathbf{u}_k$ or $\mathbf{u}_{\cdot,k}$ is the k-th column of $\mathbf{U}$
  - $\mathbf{u}_d^\top$ or $\mathbf{u}_{d,\cdot}$ is the d-th row of $\mathbf{U}$
  - $u_{d,k}$ is the element in the d-th row and k-th column of $\mathbf{U}$

Vectors

- n-th data sample $\mathbf{x}_n$ is a D-dimensional vector containing D features:

$$\mathbf{x}_n \in \mathbb{R}^{D \times 1}$$

- Vectors are basic **geometric** entities that allow you to talk about **magnitude** (or length) and **direction**.



Matrices

- **Data matrix**: represent a dataset of N datapoints $\mathbf{x}_1, ..., \mathbf{x}_N$:

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) = \begin{pmatrix} x_{1,1} & \dots & x_{1,N} \\ \vdots & & \vdots \\ x_{D,1} & \dots & x_{D,N} \end{pmatrix}$$

- Matrices can also be used to apply geometric transformations (rotations, scaling) to vectors.
- In particular: express dataset in a different coordinate system (using projections).

Example

Viewers were asked to rate the some movies (items):

	Ben	Tom	John	Fred	Jack
Star Wars	0	0	1	0	4
WallE	5	0	3	4	0
Avatar	3	4	0	4	4
Trainspotting	0	1	5	0	0
Shrek	5	0	0	5	0
Ice Age	5	0	4	0	1

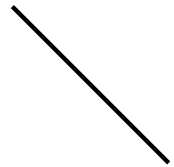
$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 1 & 0 & 4 \\ 5 & 0 & 3 & 4 & 0 \\ 3 & 4 & 0 & 4 & 4 \\ 0 & 1 & 5 & 0 & 0 \\ 5 & 0 & 0 & 5 & 0 \\ 5 & 0 & 4 & 0 & 1 \end{pmatrix}$$

$\mathbf{X}$: Users are the samples, ratings on movies are the features.

$\mathbf{X}^\top$: Movies are the samples, ratings by users are the features.

Matrix Transpose

Transpose: reflect vector/matrix on a diagonal line



Two examples:

$$\begin{pmatrix} a \\ b \end{pmatrix}^\top = (a\ b) \qquad \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}^\top = \begin{pmatrix} a & c & e \\ b & d & f \end{pmatrix}$$

Note that $(\mathbf{A}\mathbf{x})^\top = \mathbf{x}^\top \mathbf{A}^\top$.

Special Matrices

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ $\mathbf{I}$ - identity matrix

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$$ diagonal matrix

$$\begin{pmatrix} a & d & f \\ 0 & b & e \\ 0 & 0 & c \end{pmatrix}$$ upper triangular

$$\begin{pmatrix} a & 0 & 0 \\ d & b & 0 \\ f & e & c \end{pmatrix}$$ lower triangular

$$\begin{pmatrix} a & d & f \\ d & b & e \\ f & e & c \end{pmatrix}$$ symmetric: $\mathbf{A}^\top = \mathbf{A}$

Scalar Product

Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{D \times 1}$, then define:

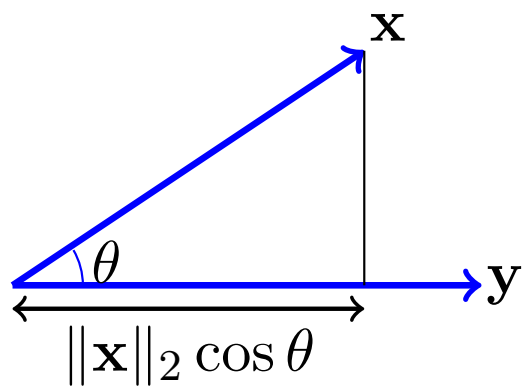
$$\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^\top \mathbf{y} = (x_1, x_2, \ldots, x_D) \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \end{pmatrix}$$

$$= \sum_{d=1}^{D} x_d y_d$$

- $\mathbf{x}^\top \mathbf{x} = \langle \mathbf{x}, \mathbf{x} \rangle$ gives the squared Euclidean length of $\mathbf{x}$.
- L_2 Norm: $\|\mathbf{x}\|_2 := \sqrt{\mathbf{x}^\top \mathbf{x}}$
- Generally: $\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos\theta$, where $\theta := \angle(\mathbf{x}, \mathbf{y})$

Scalar Product

- If $\|\mathbf{y}\|_2 = 1$ (unit vector), then $\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\|_2 \cos\theta$ gives the magnitude of the projection of $\mathbf{x}$ onto $\mathbf{y}$.
  - the scalar component of $\mathbf{x}$ in the direction of $\mathbf{y}$:



Question: When is $\mathbf{x}^\top \mathbf{y} = 0$?

Vector Matrix Multiplications

Let $\mathbf{x}$ be a D-dimensional column vector and $\mathbf{A}$ be a $M \times D$ matrix. Then $\mathbf{b} = \mathbf{A}\mathbf{x}$ is an M vector defined as

$$b_m = \sum_{d=1}^{D} a_{m,d} x_d, \quad m = 1, \ldots, M.$$

We interpret this by saying that "$\mathbf{A}$ acts on $\mathbf{x}$ to produce $\mathbf{b}$".

This product can be rewritten as $\mathbf{b} = \mathbf{A}\mathbf{x} = \sum_{d=1}^{D} x_d \mathbf{a}_d$ and visualized in the following manner

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \cdots & a_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}.$$

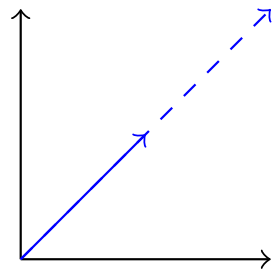
Matrices as Linear Transformations

We can transform $\mathbf{x}$ into $\mathbf{y} \in \mathbb{R}^{M \times 1}$ using a transformation matrix $\mathbf{A} \in \mathbb{R}^{M \times D}$

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

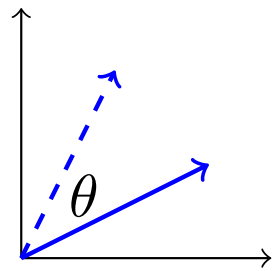
Some special cases (2-D):

$$\mathbf{A} = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}$$



scaling in all directions by α

$$\mathbf{A} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$



counterclockwise rotation by θ

Vector Spaces

- Formally, a vector space is a set of vectors which is closed under vector addition and (scalar) multiplication
  - typically: scalars = real numbers
- A subspace is a subset of a vector space which is itself a vector space
  - example: the plane $z = 0$ is a subspace of $\mathbb{R}^3$ (essentially $\mathbb{R}^2$).
- Note: subspaces must include the origin (zero vector).

Column Space

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$x_1 \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + x_2 \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

- Linear systems define certain subspaces
- $\mathbf{Ax} = \mathbf{b}$ is solvable iff $\mathbf{b}$ can be written as a linear combination of the columns of $\mathbf{A}$.
- The set of possible vectors $\mathbf{b}$ forms a subspace called the **column space** of $\mathbf{A}$.

Range and Nullspace

Every linear mapping from $\mathbb{R}^D$ to $\mathbb{R}^M$ can be represented by an appropriate $\mathbf{A} \in \mathbb{R}^{M \times D}$.

- range$(\mathbf{A})$ is the set of vectors $\{\mathbf{b} \,|\, \exists \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$.
  - this is just the column space of $\mathbf{A}$
- null$(\mathbf{A})$ is the set of vectors $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{0}\}$.
  - the entries of each $\mathbf{x} \in$ null$(\mathbf{A})$ form an expansion of zero as a linear combination of the columns of $\mathbf{A}$:

$$\mathbf{0} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_D\mathbf{a}_D.$$

Nullspace

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \Longrightarrow \text{ null space: } \{(0,0)\}$$

$$\begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & 4 \\ 1 & 3 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Longrightarrow \text{ null space: } \{(c, \frac{2}{3}c, -c)\}$$

Linear Independence

The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are called **linearly independent** if

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_n\mathbf{v}_n = 0 \quad \text{implies}$$
$$c_1 = c_2 = \ldots = c_n = 0$$

Note that in this case the nullspace of the matrix is the origin.

$$\begin{pmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

In the previous example, the nullspace contained only $(0, 0)$ therefore the columns of the matrix are linearly independent

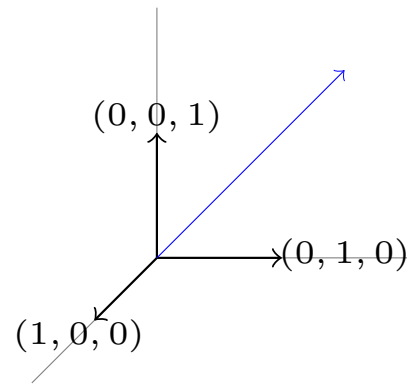
$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

null space: $\{(0, 0)\}$

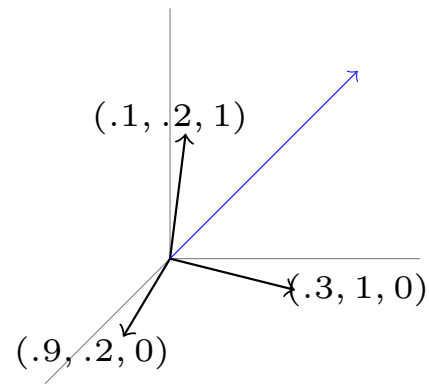
Spanning the Space

If all vectors in a vector space can be expressed as linear combinations of the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, then $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ **span** the space.

$$\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



$$\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} = 1.57 \begin{pmatrix} .9 \\ .2 \\ 0 \end{pmatrix} + 1.29 \begin{pmatrix} .3 \\ 1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} .1 \\ .2 \\ 1 \end{pmatrix}$$



Basis

A **basis** B of a vector space V is a **linearly independent** subset of V that **spans** V.

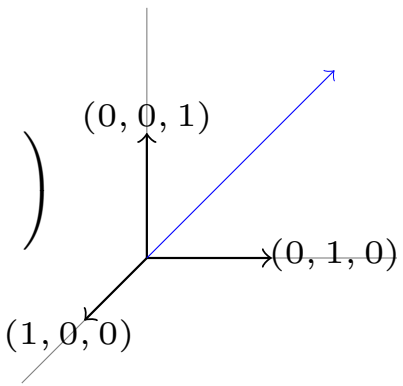
- There can be multiple different bases for a space.
- It can be shown: all bases have the same number of vectors
- Define: **dimension** of a space $=$ number of vectors in a/any basis for the space.
- A basis is a maximal set of linearly independent vectors and a minimal set of spanning vectors.

Basis Transformation

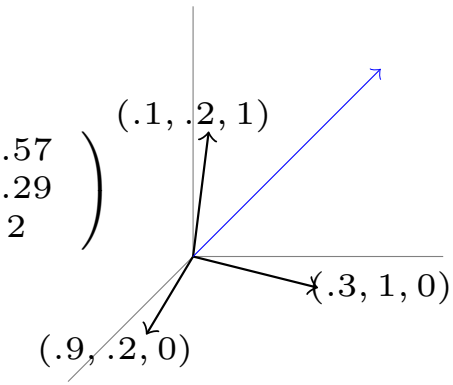
We may write $\mathbf{x} = (2, 2, 2)$ in terms of an alternate basis:

$$\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$$



$$\begin{pmatrix} .9 & .3 & .1 \\ .2 & 1 & .2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1.57 \\ 1.29 \\ 2 \end{pmatrix}$$



* The components of $(1.57, 1.29, 2)$ are projections of $\mathbf{x}$ onto the new vectors basis, normalized such that the new $\mathbf{x}$ still has the same length.

Rank

The column **rank** of $\mathbf{A}$ is the dimension of range($\mathbf{A}$).

- equivalently: maximal number of linearly independent column vectors
- one can also define the row rank of a matrix $\mathbf{A}$, however it can be shown that row and column rank are always equal (see below)

Exercise

Consider:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1. What is the range?
2. What is the rank?
3. What is the null space?

Theorem

Fundamental Theorem of Linear Algebra:

If $\mathbf{A}$ is $M \times N$ with rank R, then

- range($\mathbf{A}$) has dimension R
- null($\mathbf{A}$) has dimension $N - R$ (= nullity of $\mathbf{A}$)
- row space of $\mathbf{A}$ = range($\mathbf{A}^\top$), has dimension R

Rank-Nullity Theorem: rank + nullity = N

Inverse

A **nonsingular** or **invertible** matrix $\mathbf{A}$ is a square $M \times M$ matrix of full rank.

- its columns form a basis for the whole space $\mathbb{R}^M$.
- any vector can be uniquely expressed as a linear combination of the basis vectors
- in particular: canonical unit vectors $\mathbf{e}_j$ can be expanded as

$$\mathbf{e}_j = \sum_{m=1}^{M} z_{m,j} \mathbf{a}_m \quad (\forall j = 1, \ldots, M)$$

- this fact can be compactly written as

$$\left[\begin{array}{c|c|c} & & \\ \mathbf{e}_1 & \cdots & \mathbf{e}_M \\ & & \end{array}\right] = \mathbf{I} = \left[\begin{array}{c|c|c} & & \\ \mathbf{a}_1 & \cdots & \mathbf{a}_M \\ & & \end{array}\right] \mathbf{Z},$$

  where $\mathbf{Z}$ is the matrix with entries $z_{m,j}$.
- $\mathbf{Z}$ is the **inverse** $\mathbf{A}^{-1}$ of $\mathbf{A}$, satisfying $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

Eigenvalues and Eigenvectors

Given a square matrix $\mathbf{A}$,

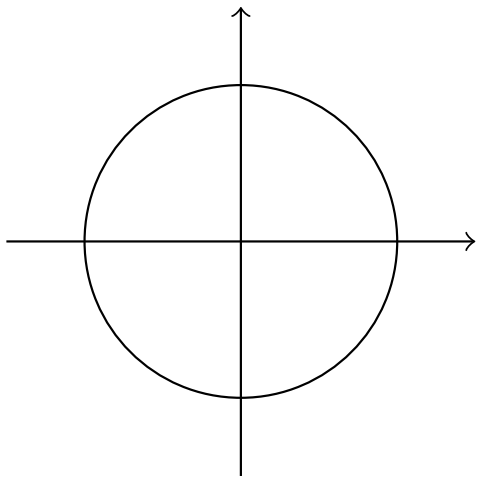
- The set of solutions to $\mathbf{Au} = \lambda\mathbf{u}$ is in the form of eigen-pairs $(\lambda, \mathbf{u})$ = (eigenvalue,eigenvector) where $\mathbf{u}$ is non-zero.

$$\mathbf{Au} = \overbrace{\lambda}^{\text{eigenvalue}} \underbrace{\mathbf{u}}_{\text{eigenvector}}$$

Eigenvector $\mathbf{u}$: Direction of $\mathbf{u}$ is not changed by transformation $\mathbf{A}$. It is only scaled by a factor λ (eigenvalue)

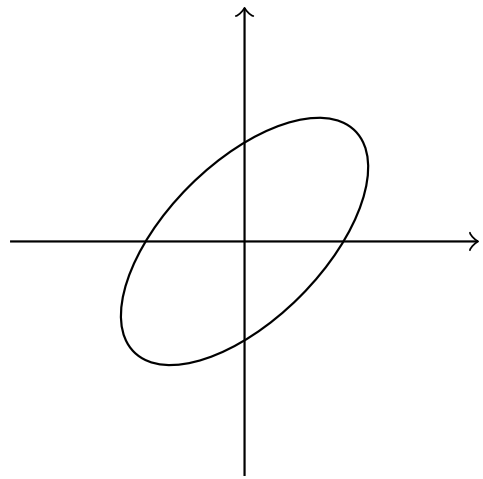
Eigendecomposition: $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues on the diagonal.

Eigenvalues and Eigenvectors



$$\Longrightarrow$$

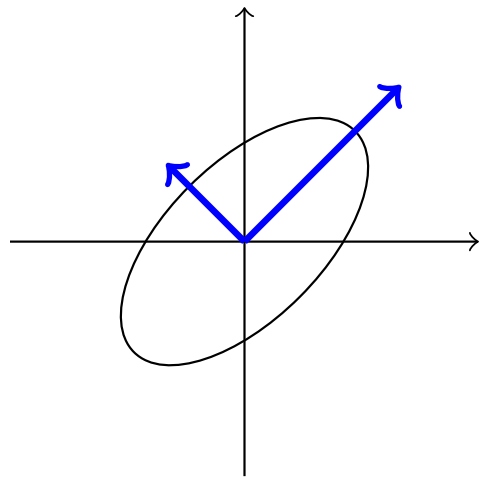
$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$



What are the (geometric) eigenvectors of this projection?

Solution

The axes of the ellipse do not change their directions:



Equivalent Conditions

For $\mathbf{A} \in \mathbb{R}^{M \times M}$, the following conditions are equivalent:

- $\mathbf{A}$ has an inverse $\mathbf{A}^{-1}$,
- $\text{rank}(\mathbf{A}) = M$,
- $\text{range}(\mathbf{A}) = \mathbb{R}^M$,
- $\text{null}(\mathbf{A}) = \{\mathbf{0}\}$,
- 0 is not an eigenvalue of $\mathbf{A}$,
- 0 is not a singular value of $\mathbf{A}$,