

Machine Learning 10-601, 10-301

Tom M. Mitchell

Machine Learning Department
Carnegie Mellon University

May 3, 2021

Today:

Representation learning:

- PCA

Privacy in ML:

Fairness, bias in ML:

- How do we avoid learning biased classifiers?

Recommended reading:

- Wall et al., PCA tutorial,
<https://arxiv.org/pdf/physics/0208101.pdf>
- “Key ideas in ML,” Mitchell,
<http://www.cs.cmu.edu/~tom/mlbook/keyIdeas.pdf>
- Bias and fairness in AI, Borealis,
<https://www.borealisai.com/en/blog/tutorial1-bias-and-fairness-ai>

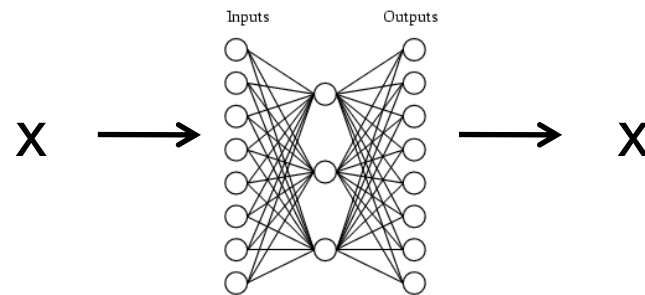
Principle Components Analysis

Principle Components Analysis

- Idea:
 - Given data points in d -dimensional space, project into lower dimensional space while preserving as much information as possible
 - E.g., find best planar approximation to 3D data
 - E.g., find best planar approximation to 10^4 D data
 - In particular, choose an orthogonal projection that minimizes the squared error in reconstructing original data

Principle Components Analysis

- Like auto-encoding neural networks, learn re-representation of input data that can best reconstruct it

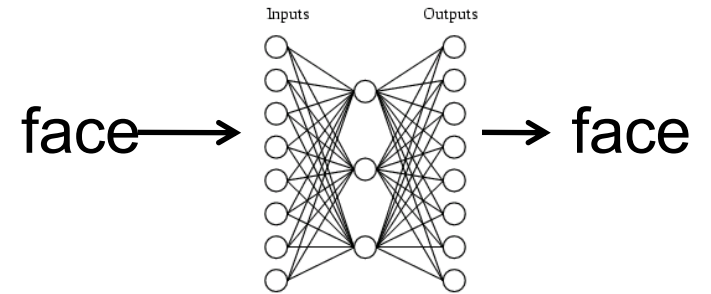


PCA:

- learned encoding is *linear* function of inputs
- No local minimum problems when training!
- Given d -dimensional input data X , learns a new d -dimensional representation, where
 - the dimensions are orthogonal
 - top k dimensions are the k -dimensional linear re-representation that minimizes reconstruction error (sum of squared errors)

PCA Example

$$\text{face}_i = \sum_k c_{ik} \text{eigenface}_k$$



faces



eigenfaces



Thanks to Christopher DeCoro
see <http://chrisdecoro.com/eigenfaces/index.html>

Reconstructing a face from the first N components (eigenfaces)

Adding 1 additional PCA component at each step

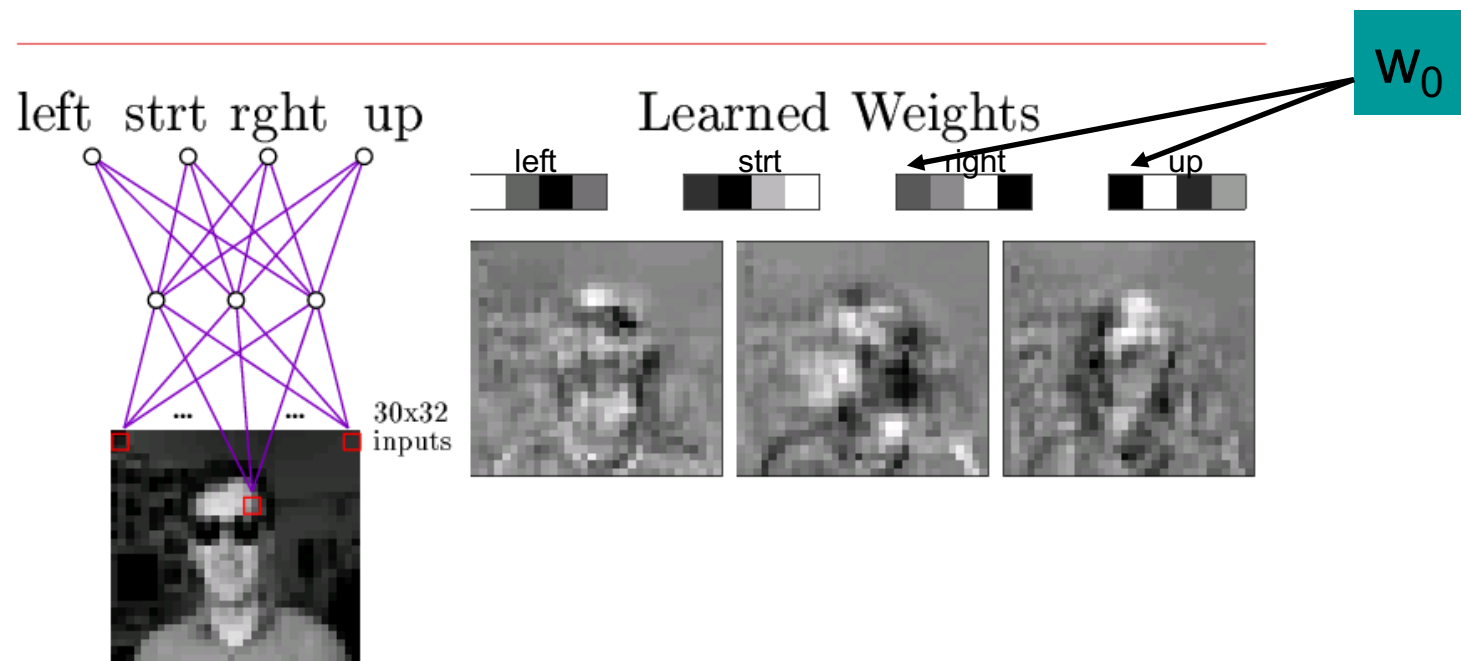


In this next image, we show a similar picture, but with each additional face representing an additional 8 principle components. You can see that it takes a rather large number of images before the picture looks totally correct.

Adding 8 additional PCA components at each step



Learned Hidden Unit Weights



Typical input images

<http://www.cs.cmu.edu/~tom/faces.html>

PCA: Find Projections to Minimize Reconstruction Error

Assume data is set of d -dimensional vectors, where n th example vector is

$$\mathbf{x}^n = \langle x_1^n \dots x_d^n \rangle$$

We can represent these in terms of any d orthogonal vectors $\mathbf{u}_1 \dots \mathbf{u}_d$

$$\mathbf{x}^n = \sum_{i=1}^d z_i^n \mathbf{u}_i; \quad \mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$$

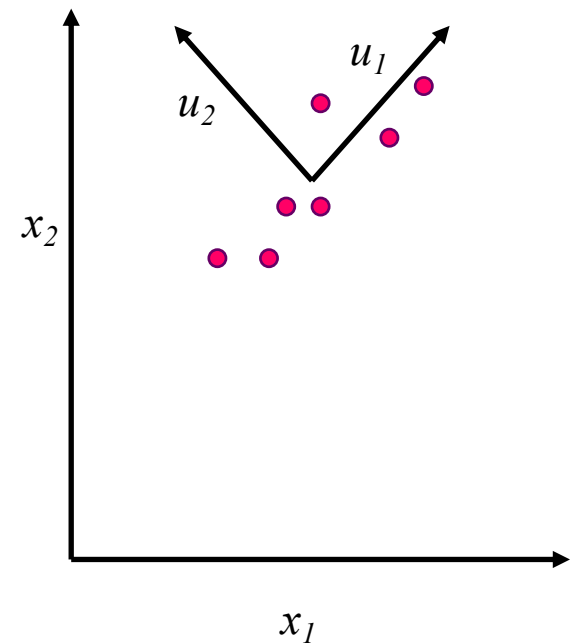
PCA: given $M < d$. Find $\langle \mathbf{u}_1 \dots \mathbf{u}_M \rangle$

that minimizes $E_M \equiv \sum_{n=1}^N \|\mathbf{x}^n - \hat{\mathbf{x}}^n\|^2$

where $\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^M z_i^n \mathbf{u}_i$

↑
Mean

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}^n$$

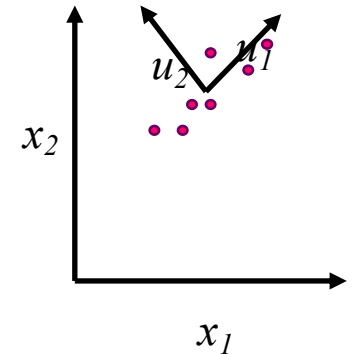


PCA

PCA: given $M < d$. Find $\langle \mathbf{u}_1 \dots \mathbf{u}_M \rangle$

that minimizes $E_M \equiv \sum_{n=1}^N \|\mathbf{x}^n - \hat{\mathbf{x}}^n\|^2$

where $\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^M z_i^n \mathbf{u}_i$



Note we get zero error if $M=d$, so all error is due to missing components.

Therefore,

$$E_M = \sum_{i=M+1}^d \sum_{n=1}^N [\mathbf{u}_i^T (\mathbf{x}^n - \bar{\mathbf{x}})]^2$$
$$= \sum_{i=M+1}^d \mathbf{u}_i^T \Sigma \mathbf{u}_i$$

Covariance matrix: $\Sigma = \sum (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T$

$$\Sigma_{ij} = \sum_{n=1}^N (x_i^n - \bar{x}_i)(x_j^n - \bar{x}_j)$$

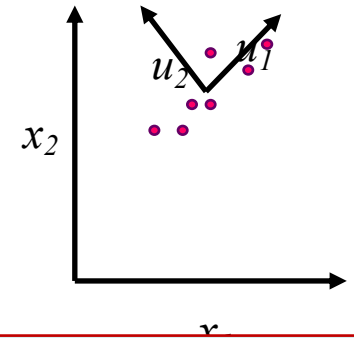
This minimized when \mathbf{u}_i is eigenvector of Σ , the covariance matrix of X . i.e., minimized when: $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$

PCA

PCA: given $M < d$. Find $\langle \mathbf{u}_1 \dots \mathbf{u}_M \rangle$

that minimizes $E_M \equiv \sum_{n=1}^N \|\mathbf{x}^n - \hat{\mathbf{x}}^n\|^2$

where $\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^M z_i^n \mathbf{u}_i$



Recall eigenvector \mathbf{v} of matrix \mathbf{A} is a vector \mathbf{v} such that $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ where λ is a scalar. (i.e., $\mathbf{A}\mathbf{v}$ is just some multiple of vector \mathbf{v}).

Note we get zero error if $M=d$, so all

Therefore,

$$E_M = \sum_{i=M+1}^d \sum_{n=1}^N [\mathbf{u}_i^T (\mathbf{x}^n - \bar{\mathbf{x}})]^2$$

$$= \sum_{i=M+1}^d \mathbf{u}_i^T \Sigma \mathbf{u}_i$$

This minimized when \mathbf{u}_i is eigenvector of Σ , the covariance matrix of \mathbf{X} . i.e., minimized when: $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$

Covariance matrix: $\Sigma = \sum (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T$

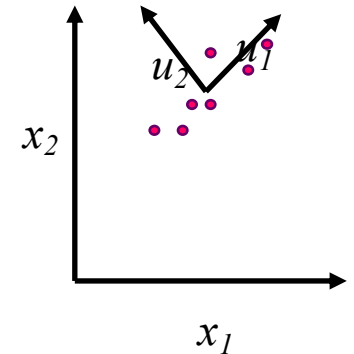
$$\Sigma_{ij} = \sum_{n=1}^N (x_i^n - \bar{x}_i)(x_j^n - \bar{x}_j)$$

PCA

PCA: given $M < d$. Find $\langle \mathbf{u}_1 \dots \mathbf{u}_M \rangle$

that minimizes $E_M \equiv \sum_{n=1}^N \|\mathbf{x}^n - \hat{\mathbf{x}}^n\|^2$

where $\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^M z_i^n \mathbf{u}_i$



Note we get zero error if $M=d$, so all error is due to missing components.

Therefore,

$$E_M = \sum_{i=M+1}^d \sum_{n=1}^N [\mathbf{u}_i^T (\mathbf{x}^n - \bar{\mathbf{x}})]^2$$
$$= \sum_{i=M+1}^d \mathbf{u}_i^T \Sigma \mathbf{u}_i$$

Covariance matrix: $\Sigma = \sum (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T$

$$\Sigma_{ij} = \sum_{n=1}^N (x_i^n - \bar{x}_i)(x_j^n - \bar{x}_j)$$

This minimized when \mathbf{u}_i is eigenvector of Σ , the covariance matrix of X . i.e., minimized when: $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$

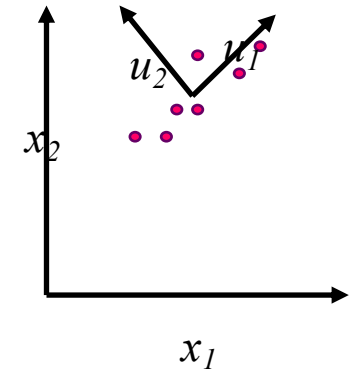
PCA

$$\text{Minimize } E_M = \sum_{i=M+1}^d \mathbf{u}_i^T \Sigma \mathbf{u}_i$$

$$\rightarrow \Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

↙ Unit length Eigenvector of Σ
↘ Eigenvalue (scalar)

$$\rightarrow E_M = \sum_{i=M+1}^d \lambda_i$$

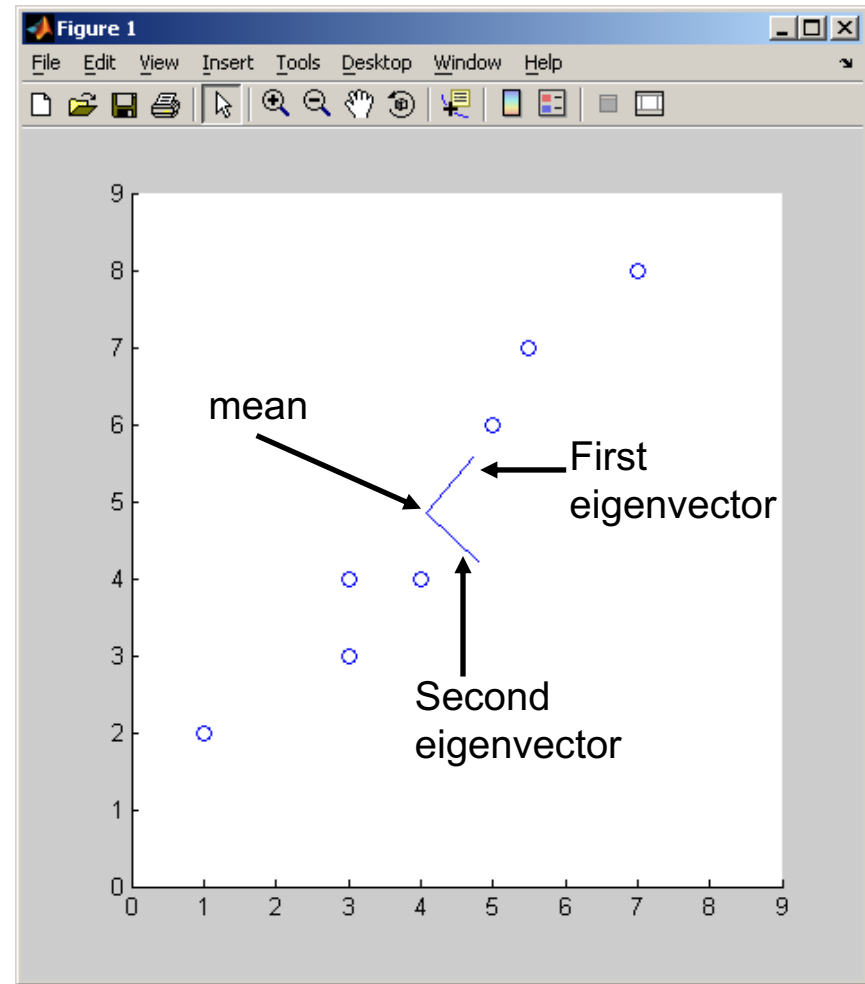
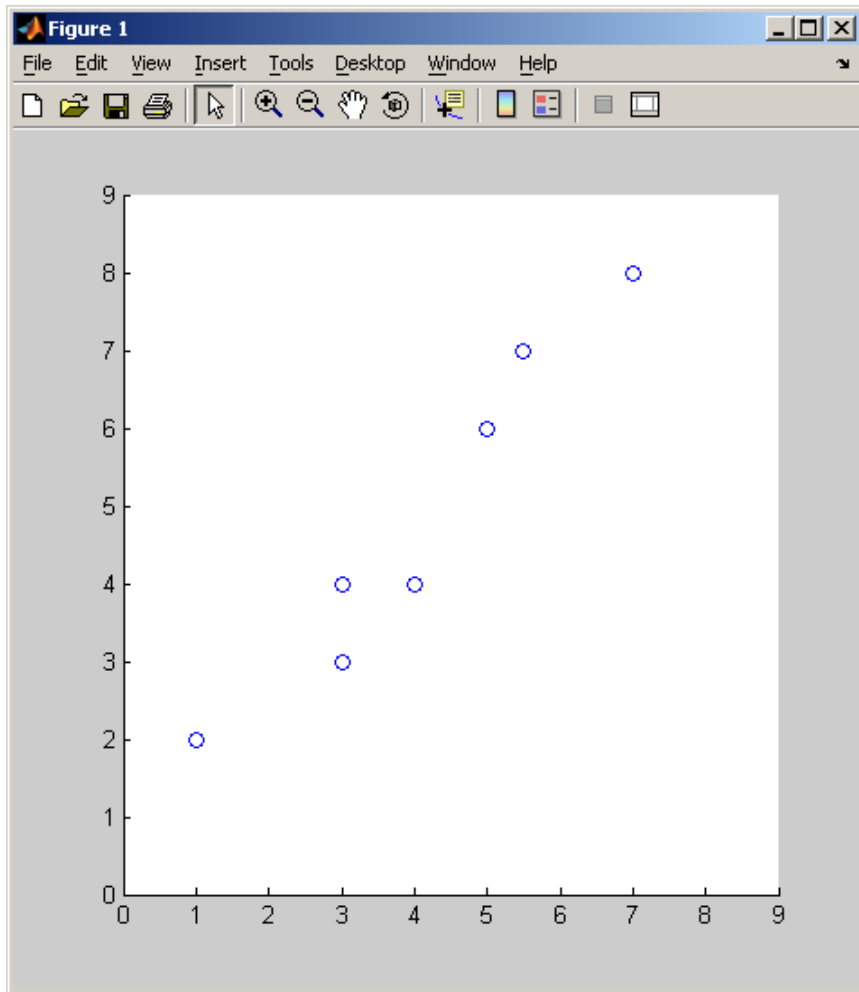


PCA algorithm 1:

1. $X \leftarrow$ Create $N \times d$ data matrix, with one row vector x^n per data point
2. $X \leftarrow$ subtract mean \bar{x} from each row vector x^n in X
3. $\Sigma \leftarrow$ covariance matrix of X
4. Find eigenvectors and eigenvalues of Σ
5. PC's \leftarrow the M eigenvectors with largest eigenvalues

PCA Example

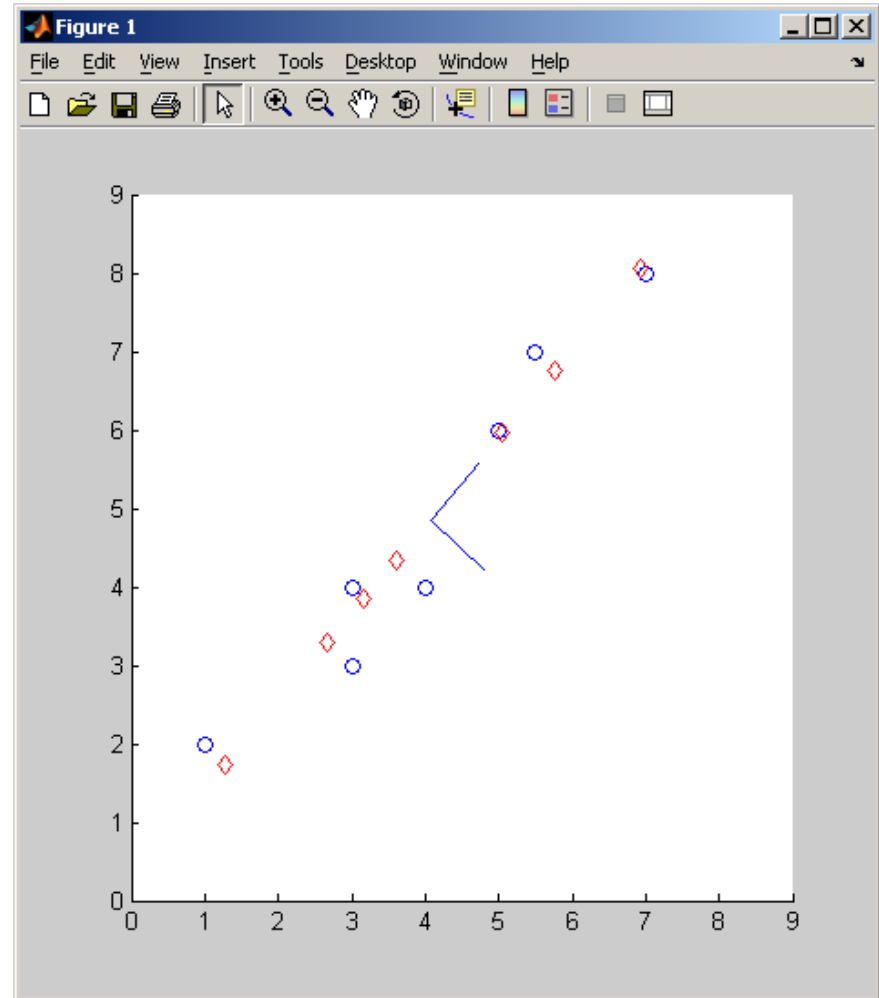
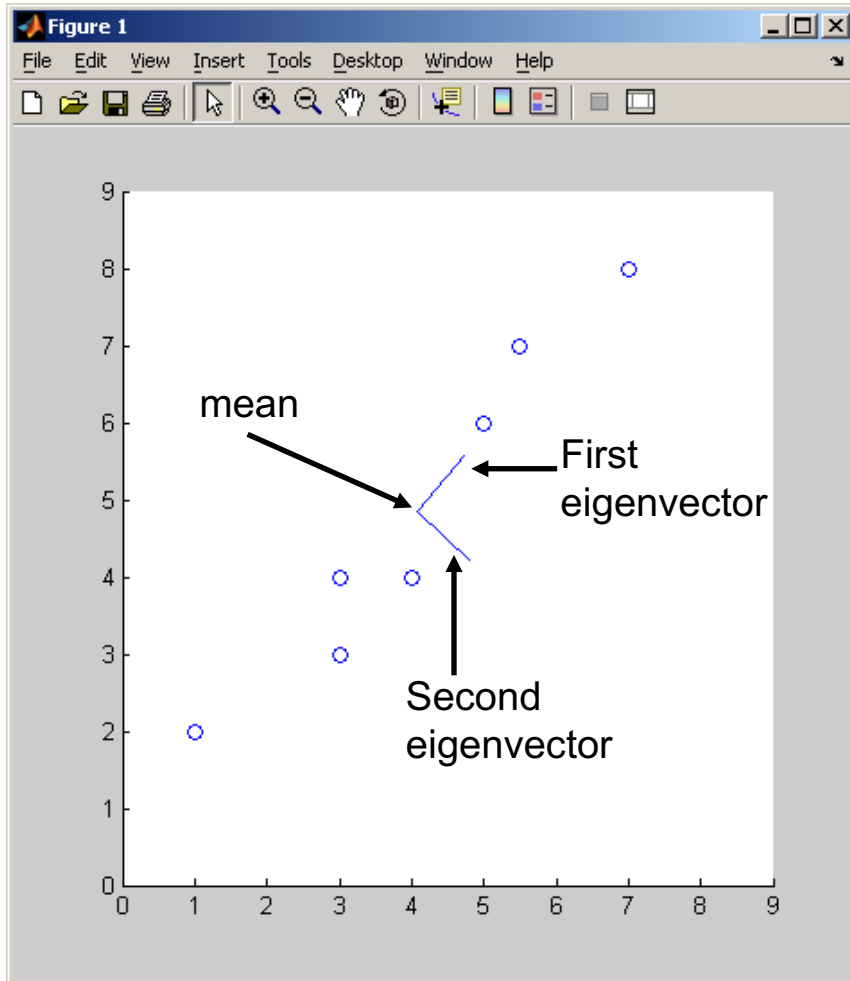
$$\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^M z_i^n \mathbf{u}_i$$



PCA Example

$$\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^M z_i^n \mathbf{u}_i$$

Reconstructed data
using only first
eigenvector (M=1)



Representation Learning Overview

- Principle Components Analysis
 - Singular Value Decomposition
 - application to face image compression
- Neural Nets
 - Hidden layers
- Matrix Factorization
 - relationship to linear neural networks
- Canonical Correlation Analysis
 - analysis of brain image data across human subjects
- Independent Components Analysis
- Latent Dirichlet Allocation
 - analysis of email interactions

Very Nice When Initial Dimension Not Too Big

What if very large dimensional data?

- e.g., Images ($d \sim 10^4$)

Problem:

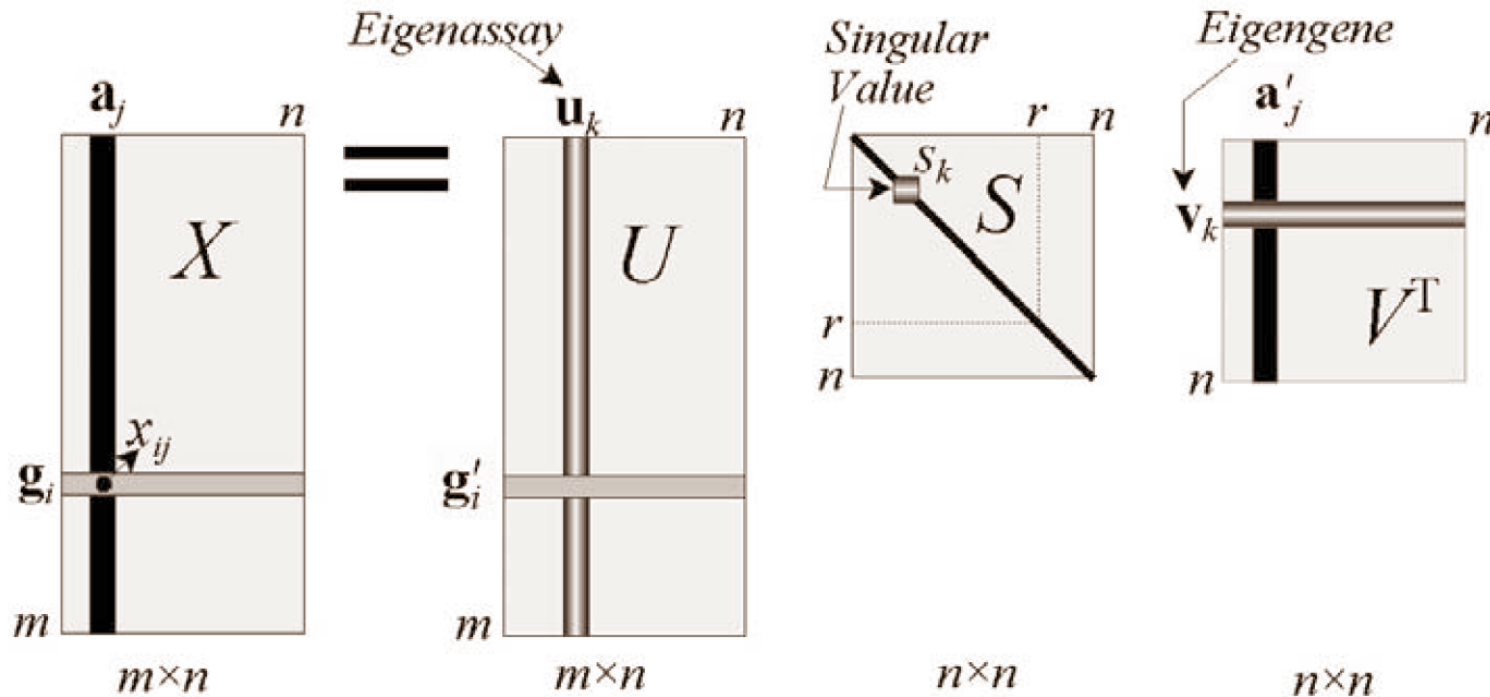
- Covariance matrix Σ is size ($d \times d$)
- $d=10^4 \rightarrow |\Sigma| = 10^8$

Singular Value Decomposition (SVD) to the rescue!

- pretty efficient algs available, including Matlab SVD
- some implementations find just top N eigenvectors

SVD

$$X = USV^T$$



Data X , one row per data point

US gives coordinates of rows of X in the space of principle components

S is diagonal, $S_k > S_{k+1}$, S_k^2 is kth largest eigenvalue

Rows of V^T are unit length eigenvectors of $X^T X$

If cols of X have zero mean, then $X^T X = c \Sigma$ and eigenvects are the Principle Components

Singular Value Decomposition

To generate principle components:

- Subtract mean $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}^n$ from each data point, to create zero-centered data
- Create matrix X with one row vector per (zero centered) data point
- Solve SVD: $X = USV^T$
- Output Principle components: columns of V (= rows of V^T)
 - Eigenvectors in V are sorted from largest to smallest eigenvalues
 - S is diagonal, with s_k^2 giving eigenvalue for k th eigenvector

Singular Value Decomposition

To project a point (column vector x) into PC coordinates:

$$V^T x$$

If x_i is i^{th} row of data matrix X , then

- (i^{th} row of US) = $V^T x_i^T$
- $(US)^T = V^T X^T$

To project a column vector x to M dim Principle Components subspace, take just the first M coordinates of $V^T x$

Privacy Enhanced Machine Learning

How Can We Minimize Privacy Impacts of ML?

Some approaches:

1. Trusted third party.
 - E.g. US Census Bureau collects/disseminates data
2. Homomorphic encryption.
3. Differential privacy.

Encryption-Based Methods

Problem setting:

- Have N sources of data (e.g., students)
- Person P wishes to know some summary statistic about their joint data (e.g., mean, variance of 2020 summer income)
- No source wants to reveal their individual data to others, including to person P

Encryption-Based Methods

Problem setting:

- Have N sources of data (e.g., students)
- Person P wishes to know some summary statistic about their joint data (e.g., mean, variance of 2020 summer income)
- No source wants to reveal their individual data to others, including to person P

Approach:

- Send encrypted message sequentially to the N sources
- Each adds in their data to the encrypted message, and passes it to the next data source
- Message finally returns to person P , who decodes it

Encryption-Based Methods

Approach:

- Send encrypted message sequentially to the N sources
- Each adds in their data to the encrypted message, and passes it to the next data source
- Message finally returns to person P, who decodes it

Example:

- Find average summer student income for 2020

Encryption-Based Methods

Approach:

- Send encrypted message sequentially to the N sources
- Each adds in their data to the encrypted message, and passes it to the next data source
- Message finally returns to person P, who decodes it

Example:

- Find average summer student income for 2020

Encryption-Based Methods

Example:

- Find average summer student income for 2020

Approach:

1. Person P makes up a huge random number, and sends it to data source 1
2. Data source 1 adds in their summer 2020 income, and passes result to data source 2
3. Data source 2 adds in their data, and the sum continues around to each data source who add in their data
4. Final data source N returns the final message to person P, who subtracts out their random number, divides by N

Nobody learns anything about any individual!

Encryption-Based Methods

We can use this to learn, e.g., decision trees, Naïve Bayes classifiers, which require only statistics of the form:

- How many of your data points that satisfy property p_1 also satisfy property p_2 ?

Example:

- Learn decision tree to predict treatment effectiveness for hospitalized Covid patients
- Thousands of hospitals as data sources, who don't/won't/can't share individual patient data
- → Can learn the decision tree by repeatedly applying this method to answer above query type

Encryption-Based Methods

A problem:

- Data source j and data source $j+2$ might collude to calculate the contribution of data source $j+1$
- → Use homomorphic encryption instead of a big random number
 - Can operate directly on the encrypted sum, to add in your contribution, without first decrypting it
 - Prevents this kind of collusion
 - But only certain arithmetic operations can be accomplished as operations on the encoded message...

Differential Privacy

Problem setting:

- Want to release some statistics about a dataset describing people
- But don't want to reveal any personal details

Key idea:

- Privacy of person p cannot be compromised if p is not in the dataset
- Therefore, release only statistical summaries that would be (almost) the same if p was not in the dataset


Differential Privacy

Problem setting:

- Want to release some statistics about a dataset describing people
- But don't want to reveal any personal details

Precise idea:

Set of all outputs
A may produce



Definition of ϵ -differential privacy [\[edit\]](#)

Let ϵ be a positive **real number** and \mathcal{A} be a **randomized algorithm** that takes a dataset as input (representing the actions of the trusted party holding the data). Let $\text{im } \mathcal{A}$ denote the **image** of \mathcal{A} . The algorithm \mathcal{A} is said to provide ϵ -differential privacy if, for all datasets D_1 and D_2 that differ on a single element (i.e., the data of one person), and all subsets S of $\text{im } \mathcal{A}$:

$$\Pr[\mathcal{A}(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(D_2) \in S],$$

where the probability is taken over the **randomness** used by the algorithm.^[9]

[Wikipedia]

Fairness and Bias in Machine Learning

The EU proposes a suite of new AI regulations, the FTC warns AI companies to fix biased algorithms,

[news from April 29, 2021]

FTC Warns Companies – Make AI Fair, Or Else: The Federal Trade Commission issued updated guidance on commercial AI last week, warning companies that using or selling biased AI systems, or exaggerating the capabilities of their algorithms, could constitute a violation of federal law. In the blog post, an FTC lawyer advises companies to take steps to ensure that their AI systems don't discriminate on the basis of race or other legally protected classes. Among those steps: limiting the use of datasets that don't include minority populations, testing algorithms for discriminatory outcomes, and opening up data and source code to outside inspection. The post warns companies to "hold yourself accountable – or be prepared for the FTC to do it for you." While the post is not the first time the FTC has focused on fairness in AI systems, observers noted the "unusually stark language" in the new guidance and called it a "shot across the bow" for U.S. AI development.

[news from April 29, 2021]

EU Proposes New AI Regulations: Last week, the European Commission proposed [a slate of new AI regulations](#) that would impose strict controls on certain “high risk” commercial AI applications and ban others entirely. Prohibited AI would include systems used for social scoring, “subliminal” manipulation, and real-time biometric surveillance by law enforcement, though certain exceptions (such as to identify specific missing children or prevent an imminent terrorist attack) would apply. [Systems deemed “high risk”](#) would be subjected to extensive inspections before deployment to ensure they are trained on well-organized and unbiased data, provide clear and transparent information to users, and are subject to human oversight. Fines for violations of the proposed regulations would be severe: up to 6 percent of the offending company’s global sales. While the regulations would be some of the strictest in the world, [they have attracted criticism from advocates due to their broad exceptions for law enforcement](#). They would also not affect military

[news from April 29, 2021]

Fairness and Bias in Machine Learning

Consider scenario:

- You have a historical data set of loan applications, labeled by whether they were approved.
- It involves two groups of people, the blue and yellow group
- Blue group members have historically received loan approvals at a higher rate than yellow
- A statistically unbiased ML algorithm, when run on this data, will learn to perpetuate the social bias inherent in the data
- You wish to learn a socially unbiased loan approval classifier

What can you do?

What Can You Do?

- Change the data?
- Change the training objective function?
- Train the same way, but change how you use the learned classifier?

What Does it Mean to be Fair?

See: <https://www.borealisai.com/en/blog/tutorial1-bias-and-fairness-ai/>

Definitions of fairness

A model is considered *fair* if errors are distributed similarly across protected groups, although there are many ways to define this. Consider taking data \mathbf{x} and using a machine learning model to compute a score $f[\mathbf{x}]$ that will be used to predict a binary outcome $\hat{y} \in \{0, 1\}$. Each data example \mathbf{x} is associated with a *protected attribute* p . In this tutorial, we consider it to be binary $p \in \{0, 1\}$. For example, it might encode sub-populations according to gender or ethnicity.

We will refer to $p = 0$ as the *deprived population* and $p = 1$ as the *favored population*. Similarly we will refer to $\hat{y} = 1$ as the *favored outcome*, assuming it represents the more desirable of the two possible results.

Assume that for some dataset, we know the ground truth outcomes $y \in \{0, 1\}$. Note that these outcomes may differ statistically between different populations, either because there are genuine differences between the groups or because the model is somehow biased. According to the situation, we may want our estimate \hat{y} to take account of these differences or to compensate for them.

Most definitions of fairness are based on *group fairness*, which deals with statistical fairness across the whole population. Complementary to this is *individual fairness* which mandates that similar individuals should be treated similarly regardless of group membership. In this blog, we'll mainly focus on group fairness, three definitions of which include: (i) demographic parity, (ii) equality of odds, and (iii) equality of opportunity. We now discuss each in turn.

Demographic Parity

Demographic parity or *statistical parity* suggests that a predictor is unbiased if the prediction \hat{y} is independent of the protected attribute p so that

$$Pr(\hat{y}|p) = Pr(\hat{y}). \quad (2.1)$$

Here, the same proportion of each population are classified as positive. However, this may result in different false positive and true positive rates if the true outcome y does actually vary with the protected attribute p .

Equality of odds

Equality of odds is satisfied if the prediction \hat{y} is conditionally independent to the protected attribute p , given the true value y :

$$Pr(\hat{y}|y, p) = Pr(\hat{y}|y). \quad (2.3)$$

This means that the true positive rate and false positive rate will be the same for each population; each error type is matched between each group.

Equality of opportunity

Equality of opportunity has the same mathematical formulation as equality of odds, but is focused on one particular label $y = 1$ of the true value so that:

$$Pr(\hat{y}|y = 1, p) = Pr(\hat{y}|y = 1). \quad (2.4)$$

In this case, we want the true positive rate $Pr(\hat{y} = 1|y = 1)$ to be the same for each population with no regard for the errors when $y = 0$. In effect it means that the same proportion of each population receive the "good" outcome $y = 1$.

Some Things are Impossible

Impossible to guarantee all three of these simultaneously:

- Equal false positive rates across groups
 - False positive rate = $P(\hat{y} = 1|y = 0)$
- Equal false negative rates across groups
 - False negative rate = $P(\hat{y} = 0|y = 1)$
- Calibrated probabilities produced by classifier
 - Among people classified $Y=1$ with probability p , a fraction p of them do have $Y=1$