Introduction to Machine Learning (CSCI-UA 473): Fall 2021

Lecture 12: Unsupervised Learning - 1 (Dimensionality reduction, principle component analysis, and non-negative matrix factorization)

Sumit Chopra
Courant Institute of Mathematical Sciences
Department of Radiology - Grossman School of Medicine
NYU

Slides derived from materials from David Sontag, Percy Liang, Ben Peherstorfer, and Barani Raman

Unsupervised Learning

Problem setting: as part of your training set you are only given the inputs.

$$\mathscr{D} = \{x_1, x_2, \ldots, x_N\} \in \mathfrak{R}^d$$

There is no notion of labels here

Two of the major buckets of tasks that fall in this learning paradigm are

Task 1: Dimensionality reduction

Seeking a low dimensional representation of your data

Can also be seen as a way to perform feature selection

Task 2: Clustering

Detecting patterns in the dataset

Unsupervised Learning

Problem setting: as part of your training set you are only given the inputs.

$$\mathscr{D} = \{x_1, x_2, \ldots, x_N\} \in \mathfrak{R}^d$$

There is no notion of labels here

Two of the major buckets of tasks that fall in this learning paradigm are

This lecture

Task 1: Dimensionality reduction

Seeking a low dimensional representation of your data

Can also be seen as a way to perform feature selection

Next lecture

Task 2: Clustering

Detecting patterns in the dataset

Lecture Outline

Dimensionality reduction

Principle component analysis

Non-negative matrix factorization

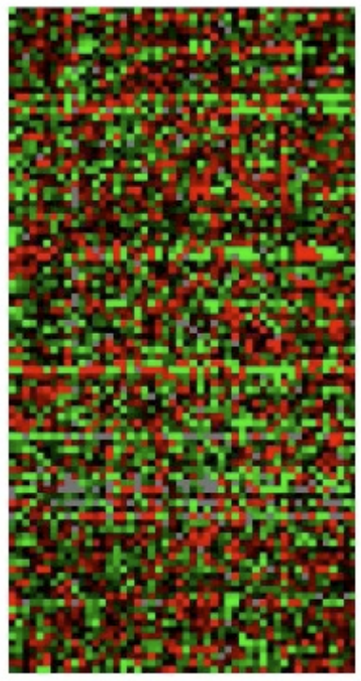
Most real-world data sets are high dimensional

face images

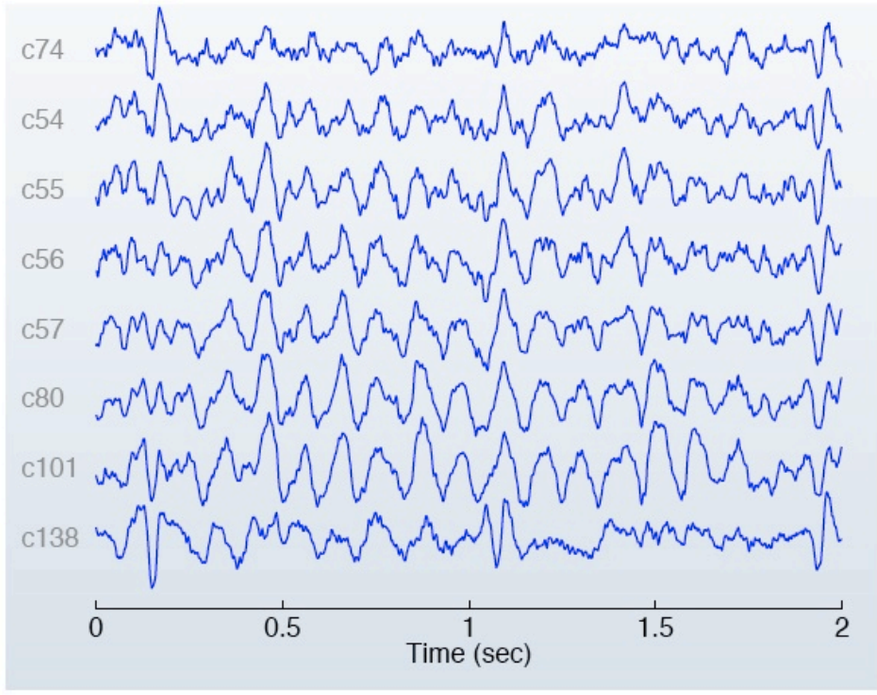
Zambian President Levy Mwanawasa has won a second term in office in an election his challenger Michael Sata accused him of rigging, official results showed on Monday.

According to media reports, a pair of hackers said on Saturday that the Firefox Web browser, commonly perceived as the safer and more customizable alternative to market leader Internet Explorer, is critically flawed. A presentation on the flaw was shown during the ToorCon hacker conference in San Diego.

documents



gene expression data



MEG readings

Figure taken from: Elements of Statistical Learning

Why do we want to reduce dimensionality?

Dimensionality reduction refers to representing the same high dimensional data with fewer dimensions

Why do we want to reduce dimensionality?

Computational reasons: compressing data would lead to fewer features and hence fewer parameters of the model to fit the data (space and time efficiency)

Statistical reasons: fewer dimensions would mean better generalization

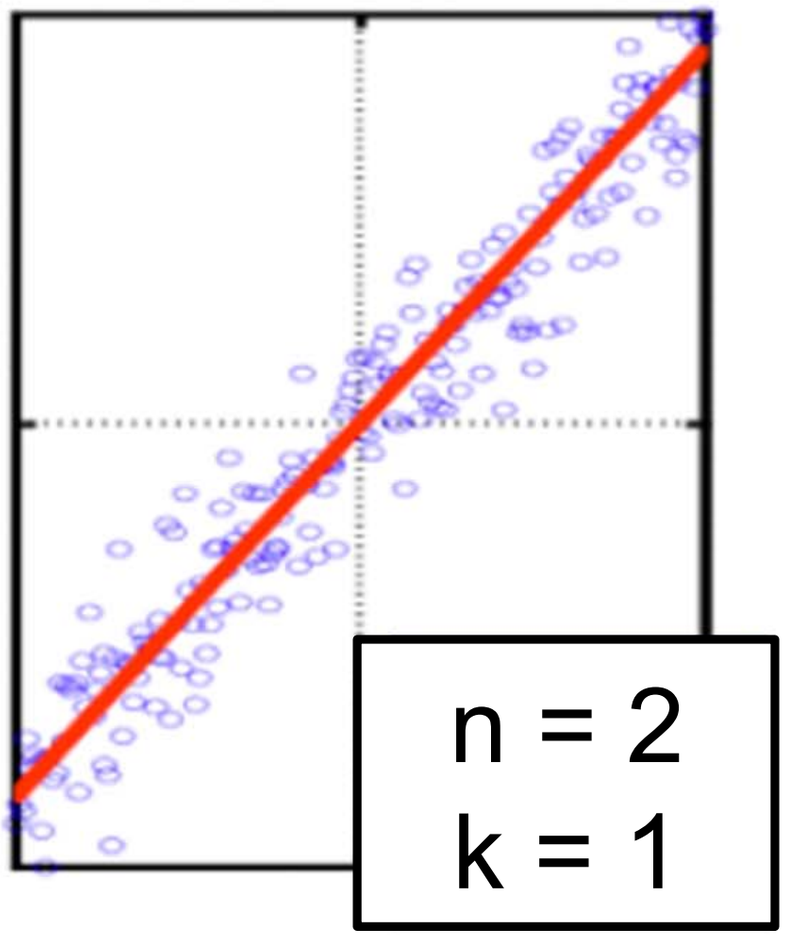
Visualization reasons: helps us better understand the structure of the data

Anomaly detection: describe normal data and detect outliers

Curse of dimensionality!

Dimensionality reduction assumption

Works under the assumption that the data inherently lies in the low dimensional space



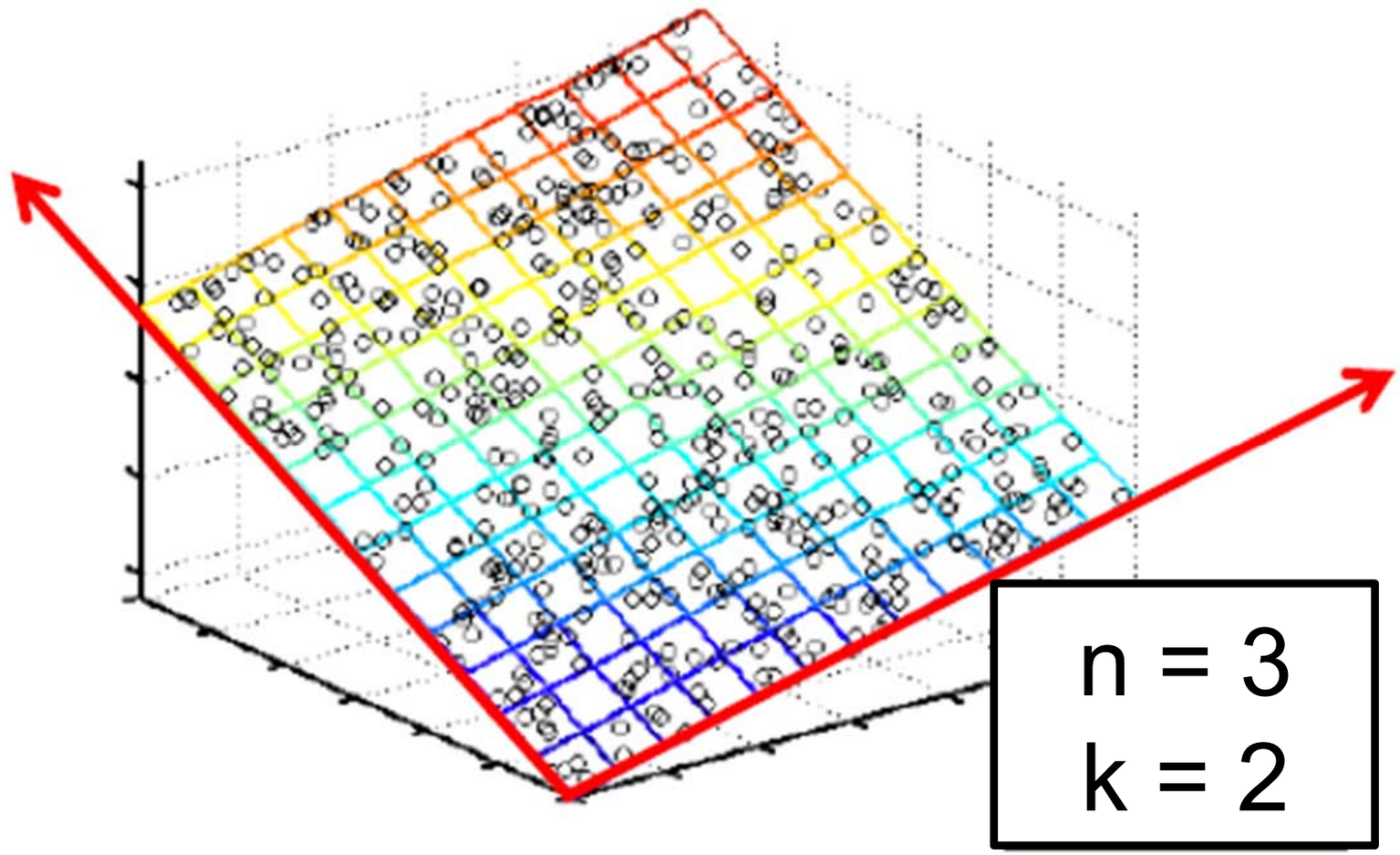
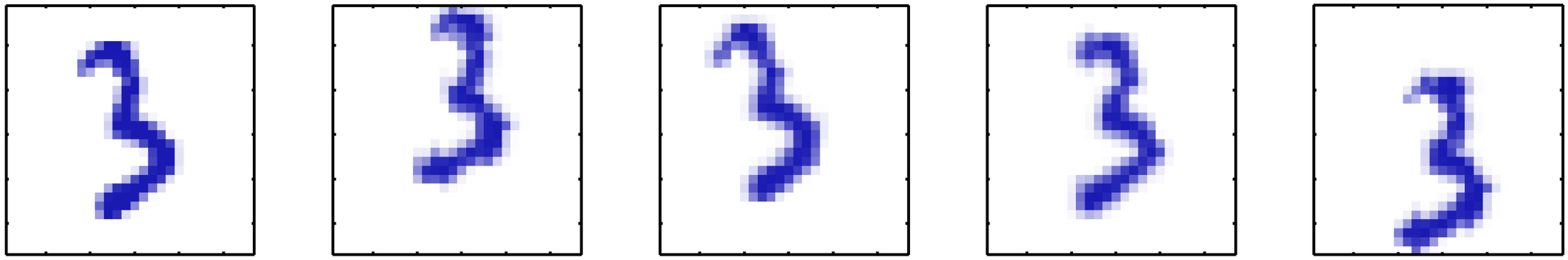


Figure taken from: David Sontag

Dimensionality reduction assumption



Assume we have a dataset composed of images of ‘3’

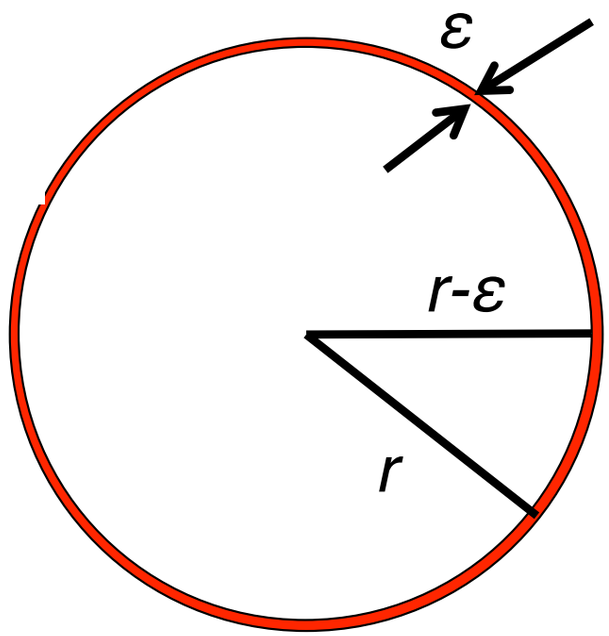
Images are of size 64x64

We know that the images were obtained by perturbing the digit in two ways: translation and rotation

What is the intrinsic dimensionality of this dataset?

Figure taken from: David Sontag

What is the curse of dimensionality?

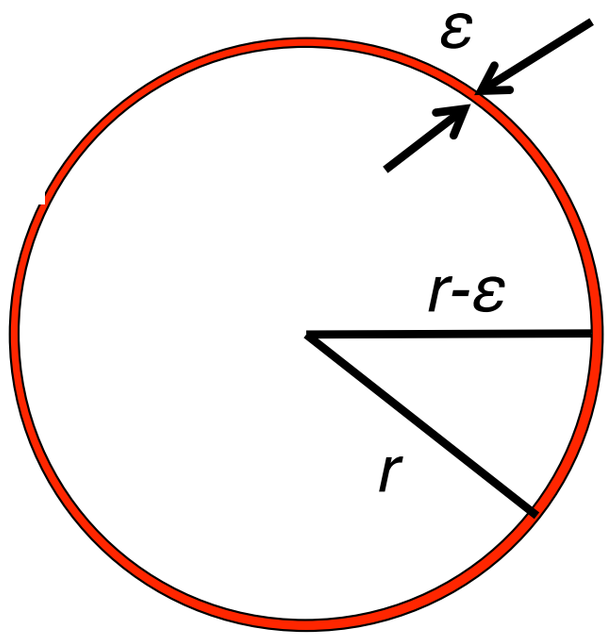


Consider the volume of a d dimensional sphere of radius r

$$V_d(r) = \frac{2\pi^{d/2} r^{d/2}}{d\Gamma(d/2)}$$

Where Γ is the gamma function given by $\Gamma(x) = (x-1)!$

What is the curse of dimensionality?



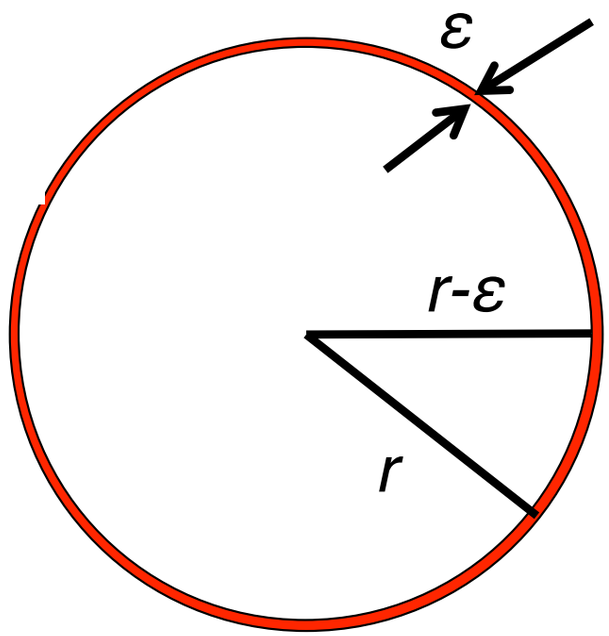
Consider the volume of a d dimensional sphere of radius r

$$V_d(r) = \frac{2\pi^{d/2} r^{d/2}}{d\Gamma(d/2)}$$

Where Γ is the gamma function given by $\Gamma(x) = (x-1)!$

Consider the volume of the outermost shell of thickness ϵ

What is the curse of dimensionality?



Consider the volume of a d dimensional sphere of radius r

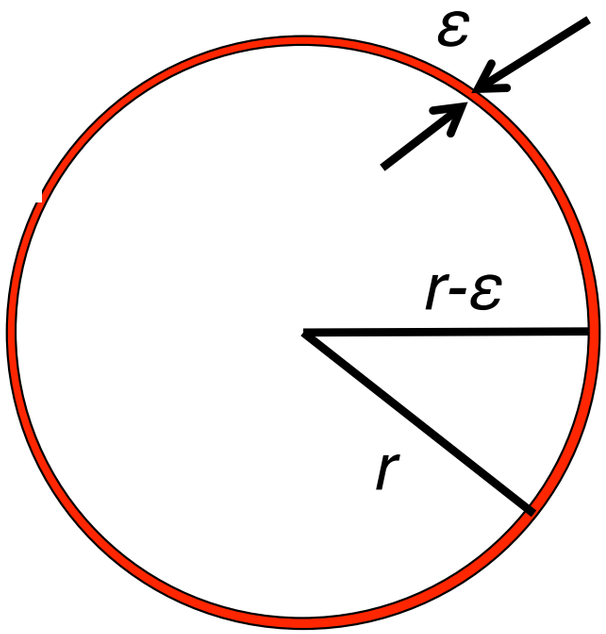
$$V_d(r) = \frac{2\pi^{d/2} r^{d/2}}{d\Gamma(d/2)}$$

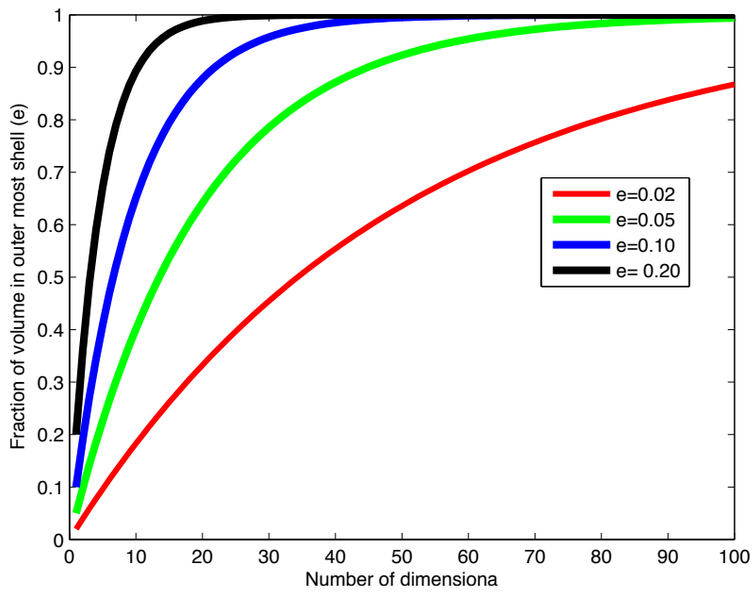
Where Γ is the gamma function given by $\Gamma(x) = (x-1)!$

Consider the volume of the outermost shell of thickness ϵ

$$\delta_d = \frac{V_d(r) - V_d(r-\epsilon)}{V_d(r)} = \frac{r^d - (r-\epsilon)^d}{r^d} = 1 - \left(1 - \frac{\epsilon}{r}\right)^d$$

What is the curse of dimensionality?





Consider the volume of a d dimensional sphere of radius r

$$V_d(r) = \frac{2\pi^{d/2} r^{d/2}}{d\Gamma(d/2)}$$

Where Γ is the gamma function given by $\Gamma(x) = (x-1)!$

Consider the volume of the outermost shell of thickness ϵ

$$\delta_d = \frac{V_d(r) - V_d(r-\epsilon)}{V_d(r)} = \frac{r^d - (r-\epsilon)^d}{r^d} = 1 - \left(1 - \frac{\epsilon}{r}\right)^d$$

What is the curse of dimensionality?

In high dimensional spaces, most of the volume is concentrated around the surface (of thickness ϵ) of the hypersphere. The center is essentially void

This is called the curse of dimensionality

The term was first coined by Bellman in 1961

It refers to the problems associated with high dimensional multi-variate data

So why is it a curse?

What is the curse of dimensionality?

Consider a three-class classification task

A simple solution

Divide the feature space uniformly into multiple bins

Compute the ratio of examples for each class in each bin

Assign the majority class to that bin

For a new example x_0, find the bin which it belongs to

Assign the class of that bin to the example x_0

What is the curse of dimensionality?

Consider a three-class classification task

A simple solution

Divide the feature space uniformly into multiple bins

Compute the ratio of examples for each class in each bin

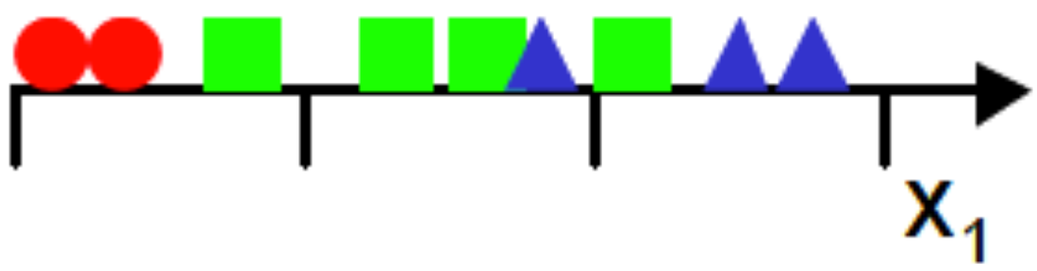
Assign the majority class to that bin

For a new example x_0, find the bin which it belongs to

Assign the class of that bin to the example x_0

We start with picking a single feature to compute bins: 1 Dimensional data

We observe that there's a lot of overlap among the examples of different class



What is the curse of dimensionality?

We select another feature to compute bins: 2 Dimensional data

We decide to preserve the granularity of each axis

Thus the number of bins now increase from 3 to $3^2 = 9$

At this point we need to make a decision

Keep the density of the bins constant

Keep the number of examples constant

What is the curse of dimensionality?

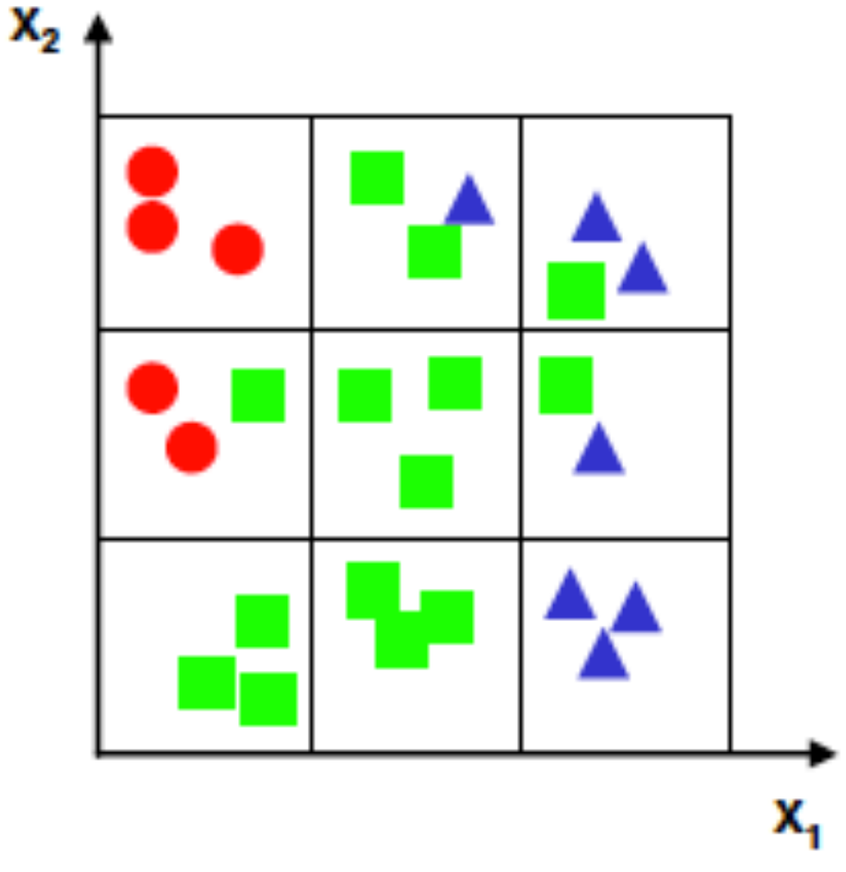
We select another feature to compute bins: 2 Dimensional data

We decide to preserve the granularity of each axis

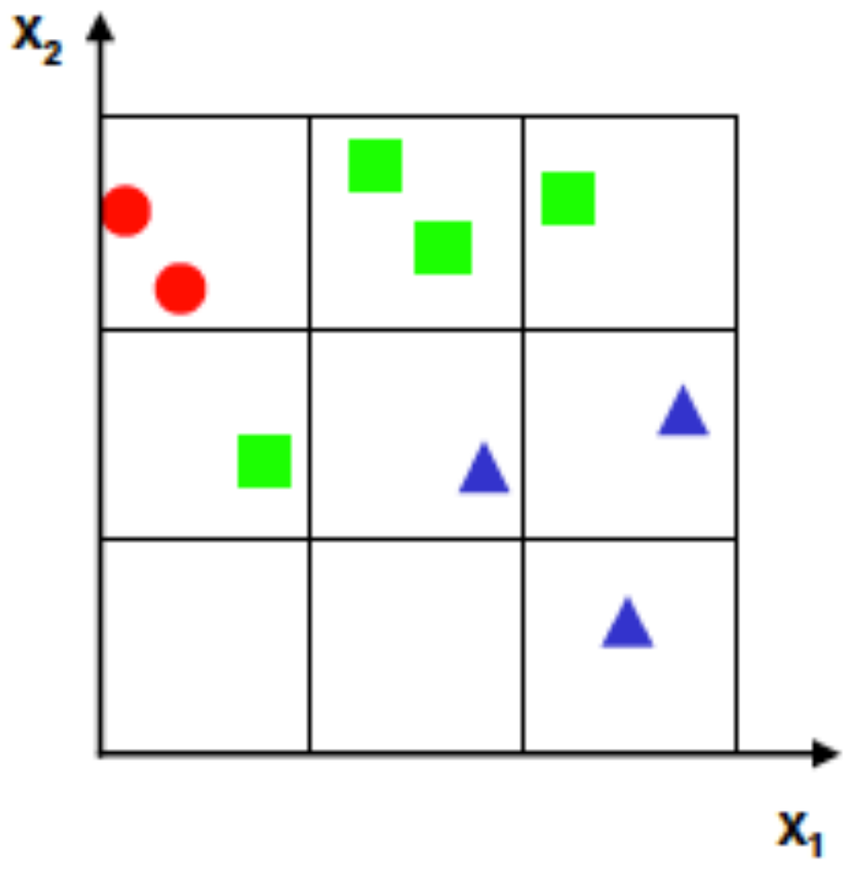
Thus the number of bins now increase from 3 to $3^2 = 9$

At this point we need to make a decision

Keep the density of the bins constant



Keep the number of examples constant



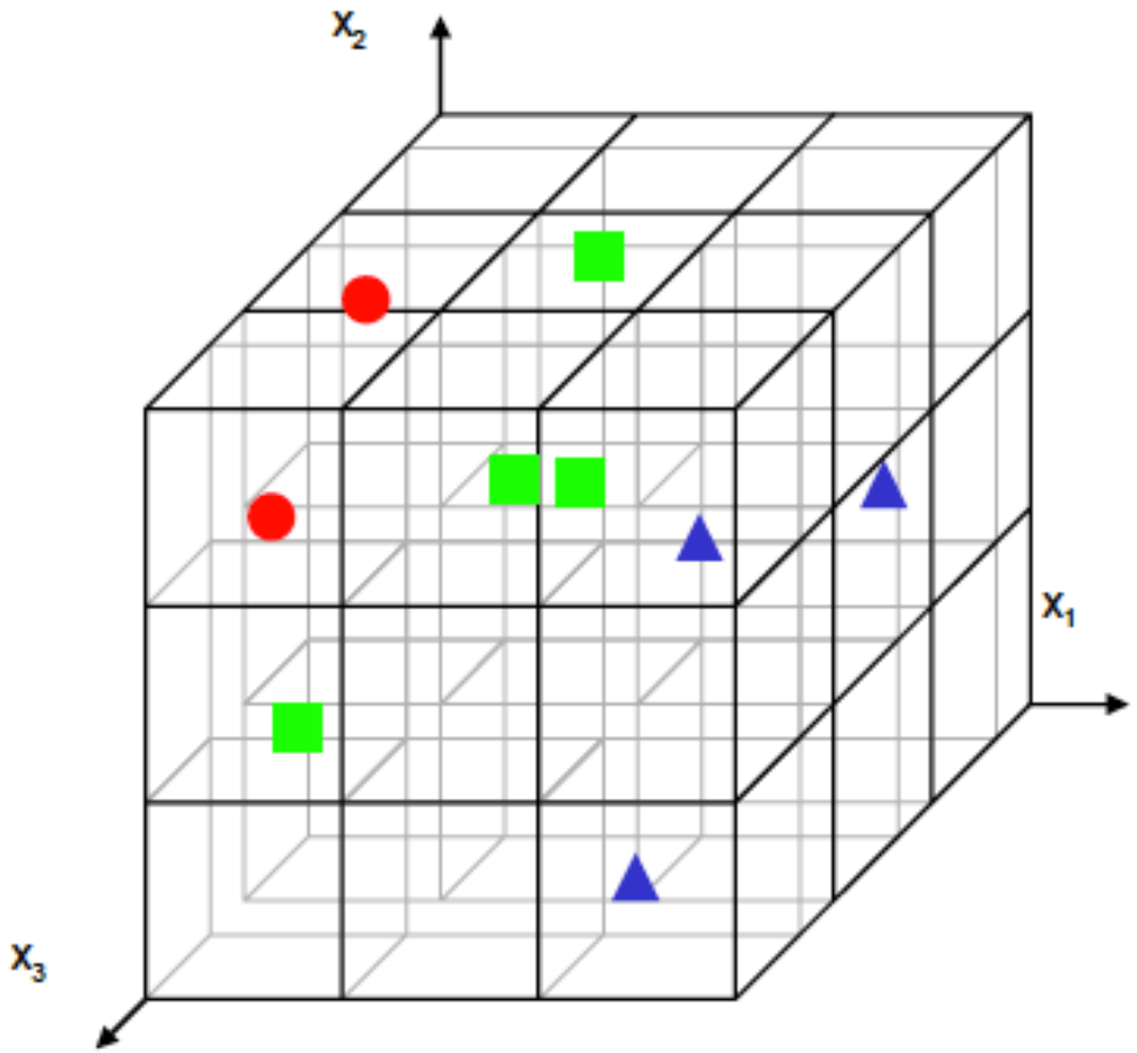
What is the curse of dimensionality?

We select another feature to compute bins: 3 Dimensional data

Now things are even worse

Number of bins grow to $3^3 = 27$

The number of examples needed to maintain the same density of bins are before is: 81



What is the curse of dimensionality?

But this was such a simple classifier?

While there are approaches less susceptible to it, the issue generally exists

What is the curse of dimensionality?

But this was such a simple classifier?

While there are approaches less susceptible to it, the issue generally exists

You can mitigate it by three ways

Incorporating prior knowledge

Increasing the smoothness of the target function

Dimensionality reduction

What is the curse of dimensionality?

But this was such a simple classifier?

While there are approaches less susceptible to it, the issue generally exists

You can mitigate it by three ways

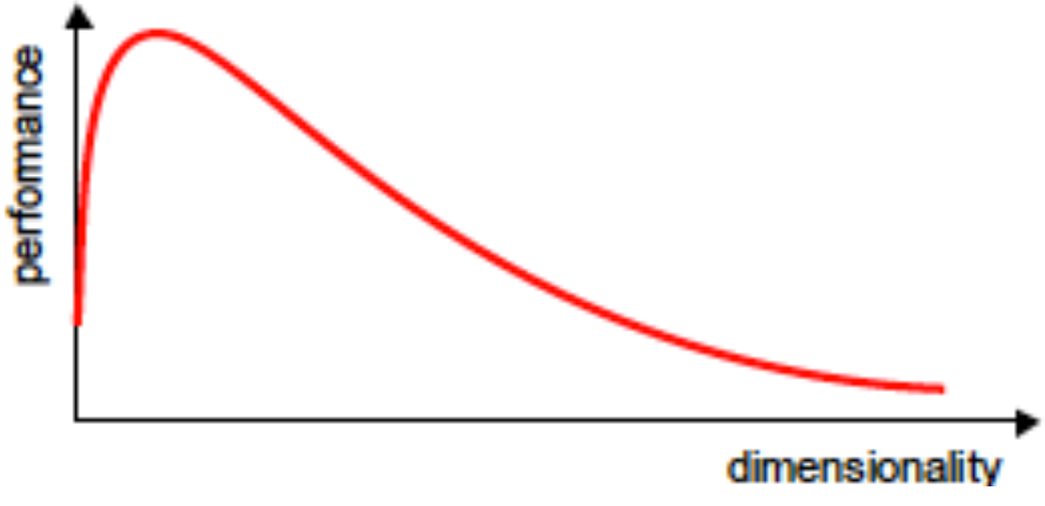
Incorporating prior knowledge

Increasing the smoothness of the target function

Dimensionality reduction

In practice the curse of dimensionality says that there is a maximum number of features above which the performance of classifiers will degrade rather than improve

The additional information that is lost by eliminating the dimensions is more than compensated by a more accurate mapping in a low dimensional space



So how do we reduce the dimensionality?

Feature selection

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \longrightarrow \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}, \text{ where } p < d$$

So how do we reduce the dimensionality?

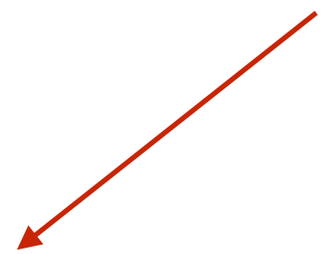
Feature selection

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \longrightarrow \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}, \text{ where } p < d$$

Transformed feature space should preserve most of the information

Feature extraction

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \longrightarrow f\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}\right) = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ x_p \end{pmatrix}$$



Types of feature extraction

There are two types of transforms (mappings) one could learn

Linear Feature Transform

$$\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1d} \\ w_{21} & w_{22} & \cdots & w_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ w_{p1} & w_{p2} & \cdots & w_{pd} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \longrightarrow \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix}$$

We will focus on these types of transforms in this lecture

Principal Component Analysis (PCA)
Non-Negative Matrix Factorization (NMF)
Linear Discriminant Analysis (LDA)
Local Linear Embedding (LLE)
ISOMAP, and many others

Non-Linear Feature Transform

Kernel PCA
Auto-encoders, and many others

Criteria for feature extraction

Selection of the mapping (feature extraction) function f is guided by the criteria we seek to optimize

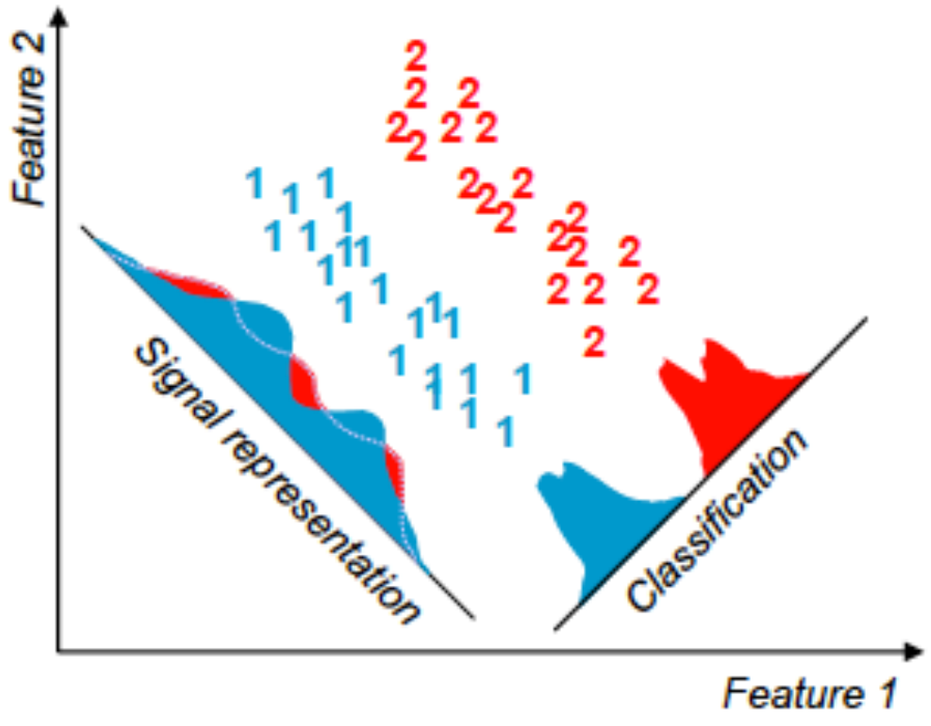
We can group the criterions into two major categories

Signal Representation

Preserve the original signal as much as possible: represent samples accurately in the lower dimensional space (PCA)

Classification Accuracy

Maximize the discriminatory information of the data in the lower dimensional space (LDA)



Dimensionality reduction problem formulation

Given n data points of dimensions d: $\mathscr{D} = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \mathfrak{R}^d, \forall i$

We want to reduce dimensionality from d to p such that $p < d$

Dimensionality reduction problem formulation

Given n data points of dimensions d: $\mathscr{D} = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \Re^d, \forall i$

We want to reduce dimensionality from d to p such that $p < d$

We choose p directions (unit vectors) $u_1, u_2, \ldots, u_p$

$u's$ are called basis vectors and they are orthogonal to each other (orthonormal basis): $u_i \cdot u_j = 0$ and $\|u_i\| = 1$

For each u_j, compute the similarity $z_j = u_j^T x$ (projection of x on u_j)

Project x down to $z = (z_1, z_2, \ldots, . z_p)^T$

Where z is given by $z = U^T x$ and

$$U = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_p \\ | & & | \end{pmatrix} \in \Re^{d \times p}$$

Slid

Dimensionality reduction problem formulation

Given n data points of dimensions d: $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \Re^d, \forall i$

We want to reduce dimensionality from d to p such that $p < d$

We choose p directions (unit vectors) $u_1, u_2, \ldots, u_p$

$u's$ are called basis vectors and they are orthogonal to each other (orthonormal basis): $u_i \cdot u_j = 0$ and $\|u_i\| = 1$

For each u_j, compute the similarity $z_j = u_j^T x$ (projection of x on u_j)

Project x down to $z = (z_1, z_2, \ldots, . z_p)^T$

Where z is given by $z = U^T x$ and

$$U = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_p \\ | & & | \end{pmatrix} \in \Re^{d \times p}$$

The question now is how to compute the matrix U ?

Slide adapted from Percy Liang

Principle Component Analysis (PCA): Reconstruction Error

Goal is to find the projections z so that we are able to reconstruct the original input x from it

Slide adapted from Percy Liang

Principle Component Analysis (PCA): Reconstruction Error

Goal is to find the projections z so that we are able to reconstruct the original input x from it

In PCA the projection matrix U serves are two purposes

1. Project (Encode): $z = U^T x, \quad z_j = u_j^T x$

2. Reconstruct (Decode): $\tilde{x} = Uz = \sum_{j=1}^{p} z_j u_j$

We want to find the matrix U such that the reconstruction error $||x - \tilde{x}||$ is small

$$\min_{U \in \Re^{dxp}} \sum_{i=1}^{n} ||x_i - Uz||^2 = \min_{U \in \Re^{dxp}} \sum_{i=1}^{n} ||x_i - UU^T x_i||^2$$

Slide adapted from Percy Liang

Principle Component Analysis (PCA): Reconstruction Error

Goal is to find the projections z so that we are able to reconstruct the original input x from it

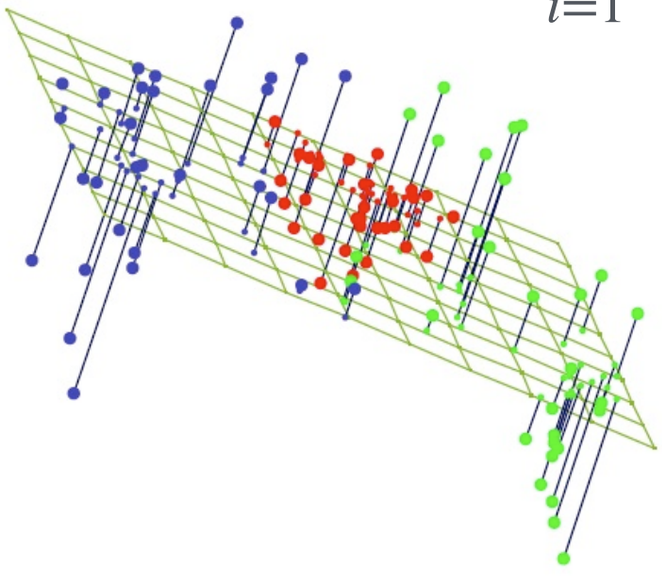
In PCA the projection matrix U serves are two purposes

1. Project (Encode): $z = U^T x, \quad z_j = u_j^T x$

2. Reconstruct (Decode): $\tilde{x} = Uz = \sum_{j=1}^{p} z_j u_j$

We want to find the matrix U such that the reconstruction error $||x - \tilde{x}||$ is small

$$\min_{U \in \Re^{dxp}} \sum_{i=1}^{n} ||x_i - Uz||^2 = \min_{U \in \Re^{dxp}} \sum_{i=1}^{n} ||x_i - UU^T x_i||^2$$



Slide adapted from Percy Liang

Principle Component Analysis (PCA): Project Variance

Goal is to find the projection that maximizes the variance of the data

Let $x_1, x_2, \ldots, x_n$ be some data points and f be any function (perhaps projection)

Expectation: $\hat{E}[f(x)] = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$

Variance: $\hat{V}[f(x)] = \hat{E}[f(x)^2] - \hat{E}[f(x)]^2$

If the mean $\hat{E}[f(x)] = 0$ (data is centered around the origin)

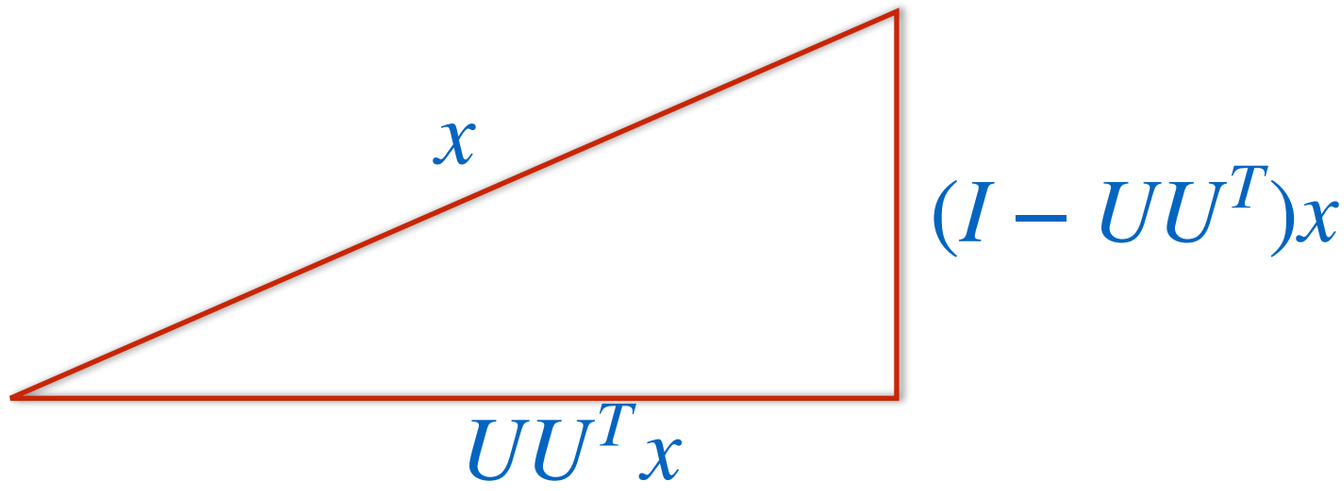
Then $\hat{V}[f(x)] = \hat{E}[f(x)^2]$

PCA Objective: maximize variance of projected data

$$\max_{U \in \Re^{d \times p}, U^T U = I} \hat{E}[||U^T x||^2]$$

Slide adapted from Percy Liang

How are the two formulations related?



Multiplying a vector with a rotation matrix U does not affect its length

$$x = UU^T x + (I - UU^T)x$$

By Pythogorean decomposition we have

$$\|x\|^2 = \|UU^T x\|^2 + \|(I - UU^T)x\|^2$$

Take expectation of this quantity

$$\begin{aligned}\hat{E}[\|x\|^2] &= \hat{E}[(\|UU^T x\|^2] + \hat{E}[\|(I - UU^T)x\|^2] \\ &= \hat{E}[(\|U^T x\|^2] + \hat{E}[\|x - UU^T x\|^2] \\ &= variance\_of\_projection + reconstruction\_error\end{aligned}$$

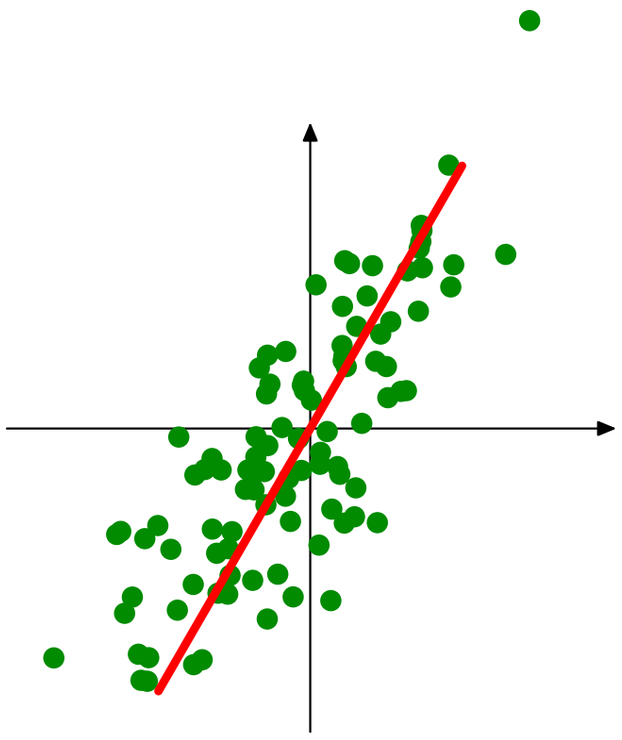
Thus minimizing reconstruction error is equivalent to maximizing the variance of the projection

Slide adapted from Percy Liang

Finding one principle component

Our goal is to maximize the variance of the projected data X

$$X = \begin{pmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{pmatrix}$$



Magnitude of Eigen value indicate the fraction of the variance captured

$$Obj = \max_{||u||=1} \hat{E}_x[(u^T x)^2]$$

$$= \max_{||u||=1} \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2$$

$$= \max_{||u||=1} \frac{1}{n} \|u^T X\|^2$$

$$= \max_{||u||=1} u^T \left(\frac{1}{n} XX^T\right) u\|^2$$

$$= largest\ eigenvalue\ of\ \frac{1}{n} XX^T$$

Covariance matrix of the data

Slide adapted from Percy Liang

The full PCA algorithm

Start with the data matrix of size $d \times n$

$$X = \begin{pmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{pmatrix}$$

Recenter the data: subtract the mean from each column of the matrix X

$$X_c \leftarrow X - \bar{X}$$

Compute the covariance matrix $C = \frac{1}{n} X_c X_c^T$

Find the Eigen vectors and Eigen values of C

The p eigen vectors with highest Eigen values give the principal components

How to choose the right number of principal components

Its a hyper-parameter

Which you typically estimate by visual inspection

Magnitude of Eigen values indicate the fraction of variance captured

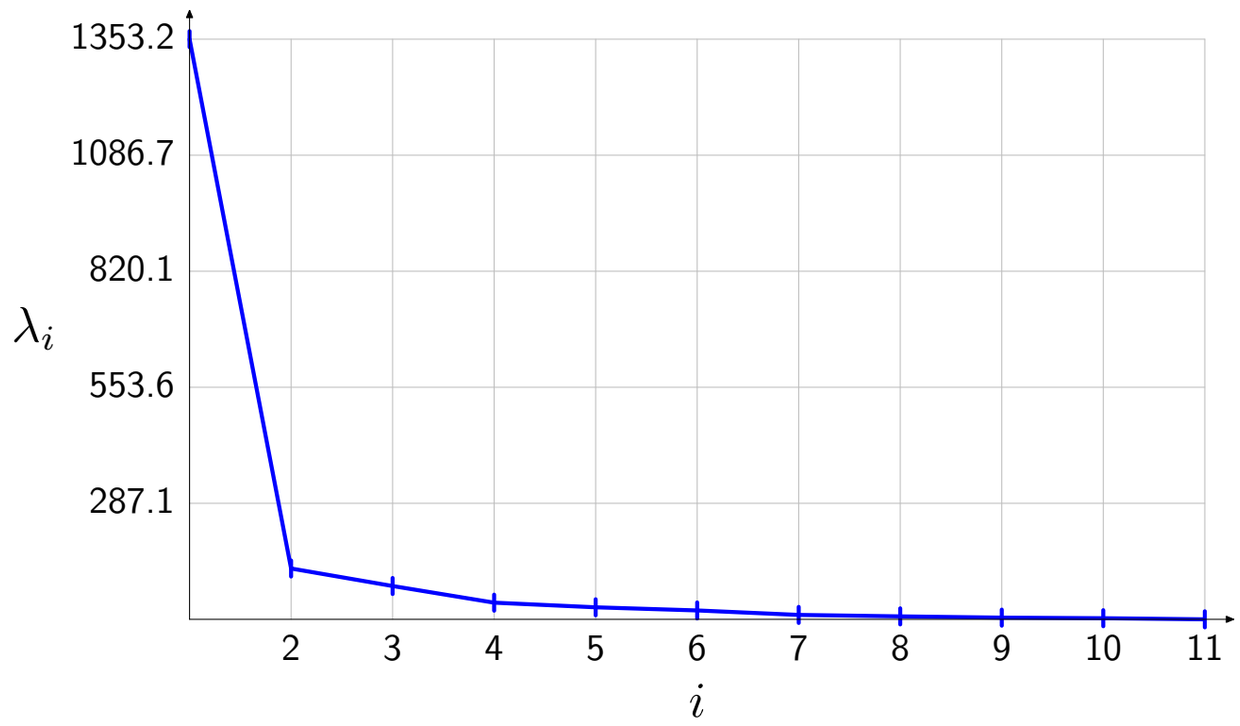
In high dimensions data usually lies near a linear subspace, as noise introduces small variability

Thus in many situations Eigen values drop off sharply after some value

Hence you don't need many of them

Ignoring the dimensions with small Eigen values will not hurt much

Eigen values on some dataset with faces



Slide adapted from Percy Liang

How to compute PCA in practice

PCA involves first estimating he covariance matrix $C = \frac{1}{n} XX^T$

This is a $\mathfrak{R}^{d \times d}$ matrix

Computing the matrix and its Eigen vectors/values can be expensive when d is large

For a data set of images of size 100×100 implied that $d = 10{,}000$

Computing C already takes $O(nd^2)$ time

How to compute PCA in practice

PCA involves first estimating he covariance matrix $C = \frac{1}{n} XX^T$

This is a $\mathfrak{R}^{d \times d}$ matrix

Computing the matrix and its Eigen vectors/values can be expensive when d is large

For a data set of images of size 100×100 implied that $d = 10{,}000$

Computing C already takes $O(nd^2)$ time

A possible solution is Singular Value Decomposition (SVD)

$$X = U_{d \times d} \Sigma_{d \times n} V^T_{n \times n}$$

Where $U^Y = I_{d \times d}$, $V^T V = I_{n \times n}$, and Σ is a diagonal matrix of singular values

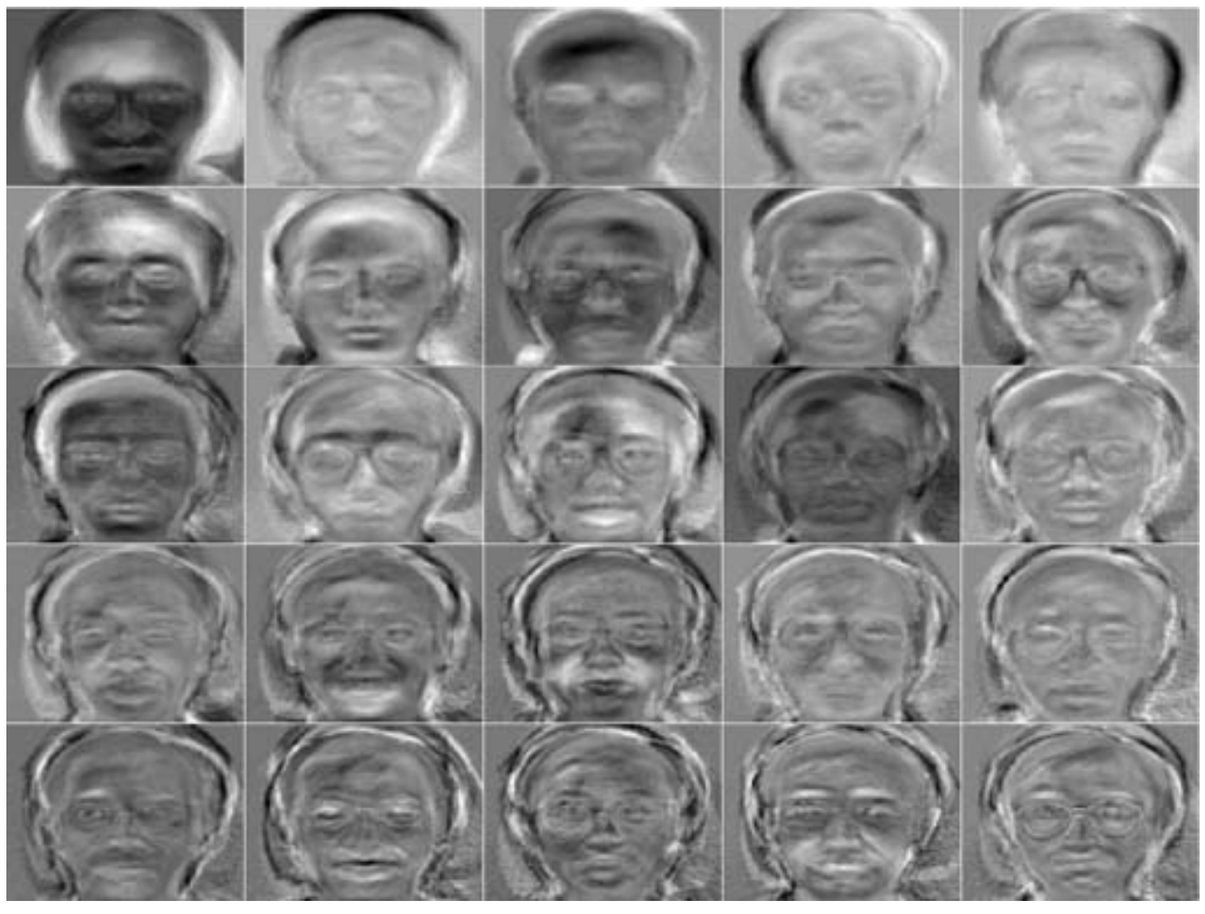
Left singular vectors are principal components of the matrix C

Computing p singular values takes around $O(ndp)$ time

Use cases of PCA

Eigen faces [Turk, Pentland 1991]

Input Images



Principal components

Use cases of PCA

Eigen faces [Turk, Pentland 1991]

Each image corresponds to adding together a weighted version of the principal components

The more components you use the more accurate the image becomes

PCA Summary

Capture the variance of data or minimize the reconstruction error

Can be achieved by finding the Eigen decomposition of covariance matrix or using an SVD of the data matrix

Reduces dimension, reduces storage, mitigates the curse of dimensionality

It is simple and fast

Unfortunately it is only a linear projection

Applications include Eigen faces, latent semantic analysis (Eigen documents), anomaly detection etc

Text Example for PCA

Let $v_1, v_2, \ldots, v_d$ denote the d words in the vocabulary of your corpus

Let $x_1, x_2, \ldots, x_N$ denote the N documents in your data set

Then each document x_i can be represented as a 1D vector of size d ($x_i \in \mathbb{R}^d$)

j-th element of this vector x_{ji} stores the number of times the word v_j appears in the document x_i

$$\underset{d \times n}{\mathbf{X}} \approx \underset{d \times k}{\mathbf{U}} \underset{k \times n}{\mathbf{Z}}$$

$$\begin{pmatrix} \text{stocks: } 2 & \cdots & 0 \\ \text{chairman: } 4 & \cdots & 1 \\ \text{the: } 8 & \cdots & 7 \\ \cdots & \vdots & \vdots \\ \text{wins: } 0 & \cdots & 2 \\ \text{game: } 1 & \cdots & 3 \end{pmatrix} \approx \begin{pmatrix} 0.4 & \cdots & -0.001 \\ 0.8 & \cdots & 0.03 \\ 0.01 & \cdots & 0.04 \\ \vdots & \cdots & \vdots \\ 0.002 & \cdots & 2.3 \\ 0.003 & \cdots & 1.9 \end{pmatrix} \begin{pmatrix} | & & | \\ \mathbf{z}_1 & \cdots & \mathbf{z}_n \\ | & & | \end{pmatrix}$$

Representations z are good for many down stream tasks

E.g., measuring similarity between two documents: $z_1^T z_2$ is probably better than $x_1^T x_2$

Slide adapted from Percy Liang

Types of feature extraction

There are two types of transforms (mappings) one could learn

Linear Feature Transform

$$\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1d} \\ w_{21} & w_{22} & \cdots & w_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ w_{p1} & w_{p2} & \cdots & w_{pd} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \longrightarrow \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix}$$

We will focus on these types of transforms in this lecture

Principal Component Analysis (PCA)

Non-Negative Matrix Factorization (NMF)

Linear Discriminant Analysis (LDA)

Local Linear Embedding (LLE)

ISOMAP, and many others

Non-Linear Feature Transform

Kernel PCA

Auto-encoders, and many others

Non-Negative Matrix Factorization (NMF)

Recall, one way to look at PCA is that it optimizes the reconstruction errors

$$\arg\min_{W,Z} \|X - WZ\|_F^2$$

Where W is the projection matrix and Z is the low dimensional representations (features/codes) of the data X

In PCA both W and Z are unconstrained. In particular they can take any value (positive or negative)

Non-Negative Matrix Factorization (NMF)

Recall, one way to look at PCA is that it optimizes the reconstruction errors

$$\arg\min_{W,Z} \|X - WZ\|_F^2$$

Where W is the projection matrix and Z is the low dimensional representations (features/codes) of the data X

In PCA both W and Z are unconstrained. In particular they can take any value (positive or negative)

The NMF puts a non-negativity constraints on W, Z

Essentially it says that decompose X into additive components

$$\arg\min_{W,Z} \|X - WZ\|_F^2$$

$$s.t.\ w_{ij} \geq 0 \quad i = 1,\ldots,n \quad j = 1,\ldots,p$$

$$z_{ij} \geq 0 \quad i = 1,\ldots,p \quad j = 1,\ldots,n$$

Motivation Behind NMF

Consider the columns of W as elements of a dictionary

Then every $x \in X$ can be interpreted as being composed of these elements by taking a linear combination of these elements whose weights are given by the vector z

$$\hat{x} = w_1 z_1 + w_2 z_2 + \cdots + w_p z_p$$

A positivity constraint on w, z ensures that x is created only by adding different elements of the dictionary

This helps in interpreting things

Example

If elements of a dictionary correspond to nose, lips, eyes, ears etc
Then a human face is composed by adding these elements with different weights
There is no notion of subtraction here

In PCA for example, since the coefficients $z_1, \ldots, z_p$ and dictionary elements $z_1, \ldots, z_p$ are allowed to have negative values, their meaning is harder to interpret

Faces with PCA

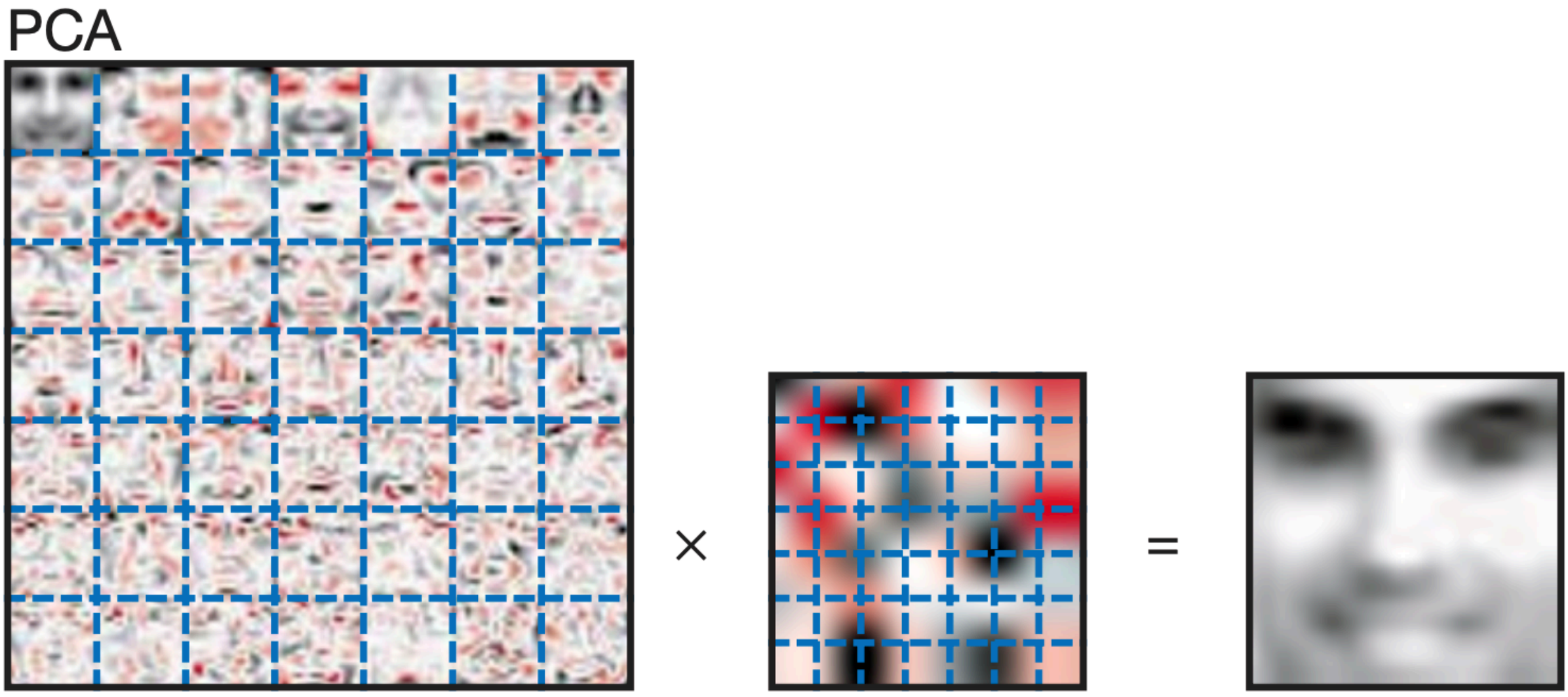


Figure from: Lee, Seung. Nature 1999

Faces with NMF

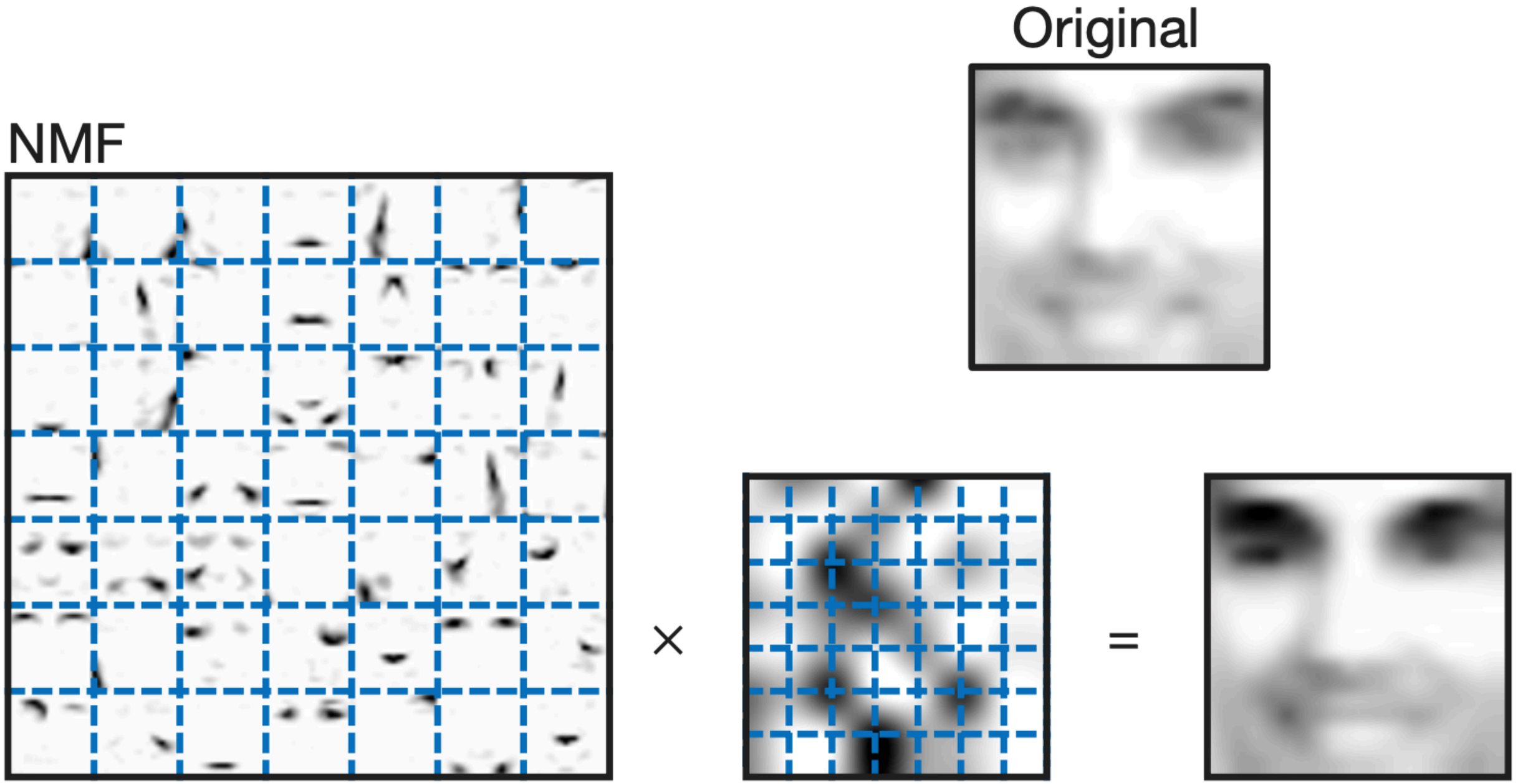


Figure from: Lee, Seung. Nature 1999

Solving for NMF is Hard

In comparison to PCA (which can be solved using an SVD solver), the optimization problem of NMF is harder to solve

$$\arg\min_{W,Z} \|X - WZ\|_F^2 \quad s.t.\ W, Z \geq 0$$

It requires a constrained optimization algorithm

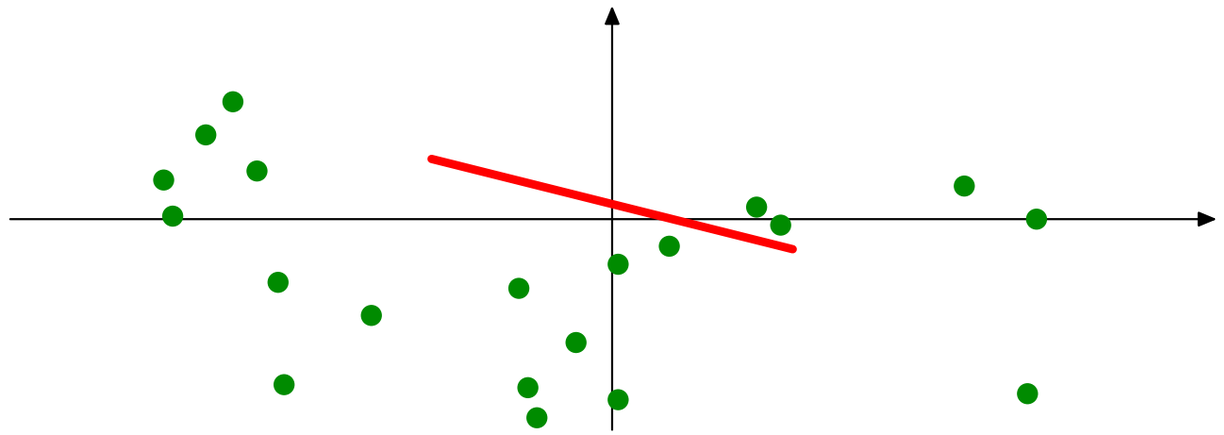
Scikit-learn implements NMF which we will play with in the lab and homework

Linear Discriminant Analysis

Up until now we have performed dimensionality reduction in an unsupervised manner

We used PCA to maximize the variance of the projection

It made sense because we want to preserve as much information of the original feature space as possible



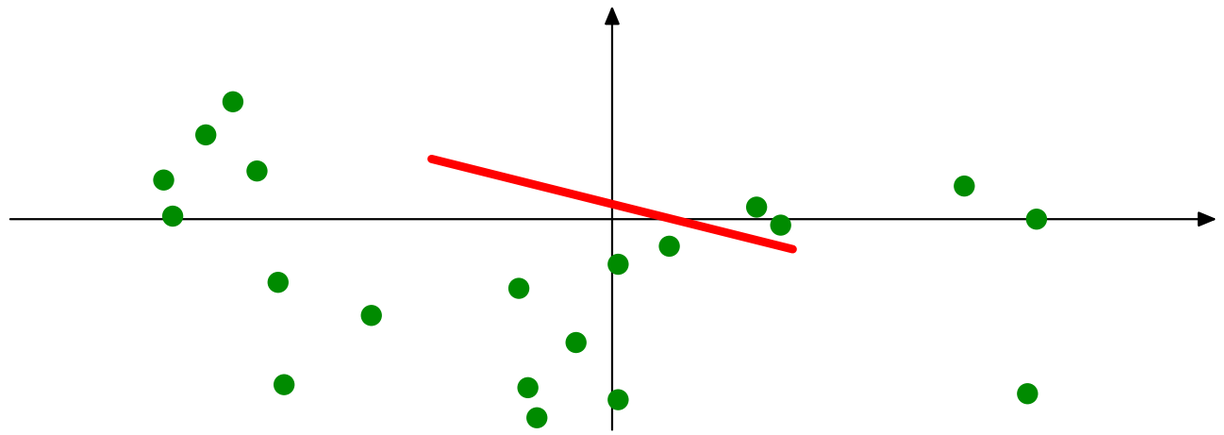
Slide adapted from Percy Liang

Linear Discriminant Analysis

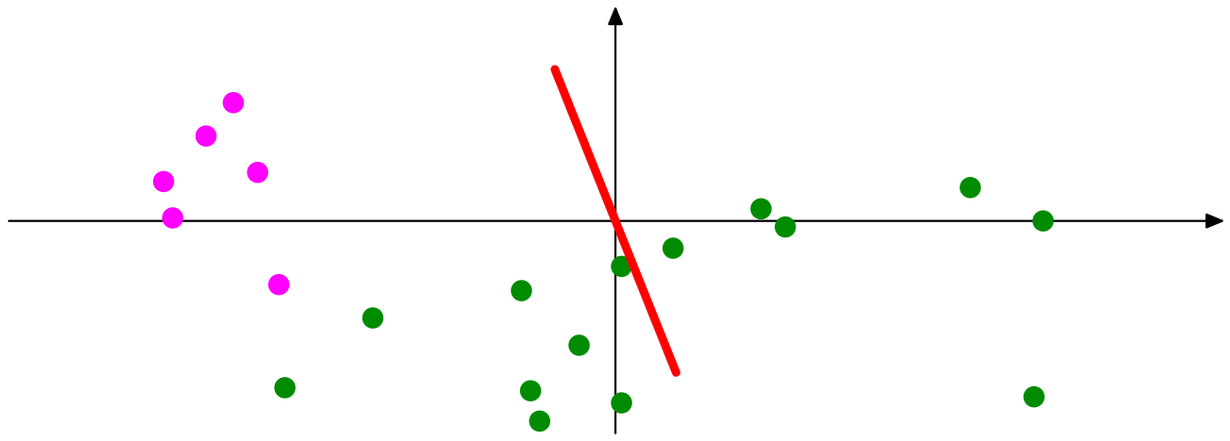
Up until now we have performed dimensionality reduction in an unsupervised manner

We used PCA to maximize the variance of the projection

It made sense because we want to preserve as much information of the original feature space as possible

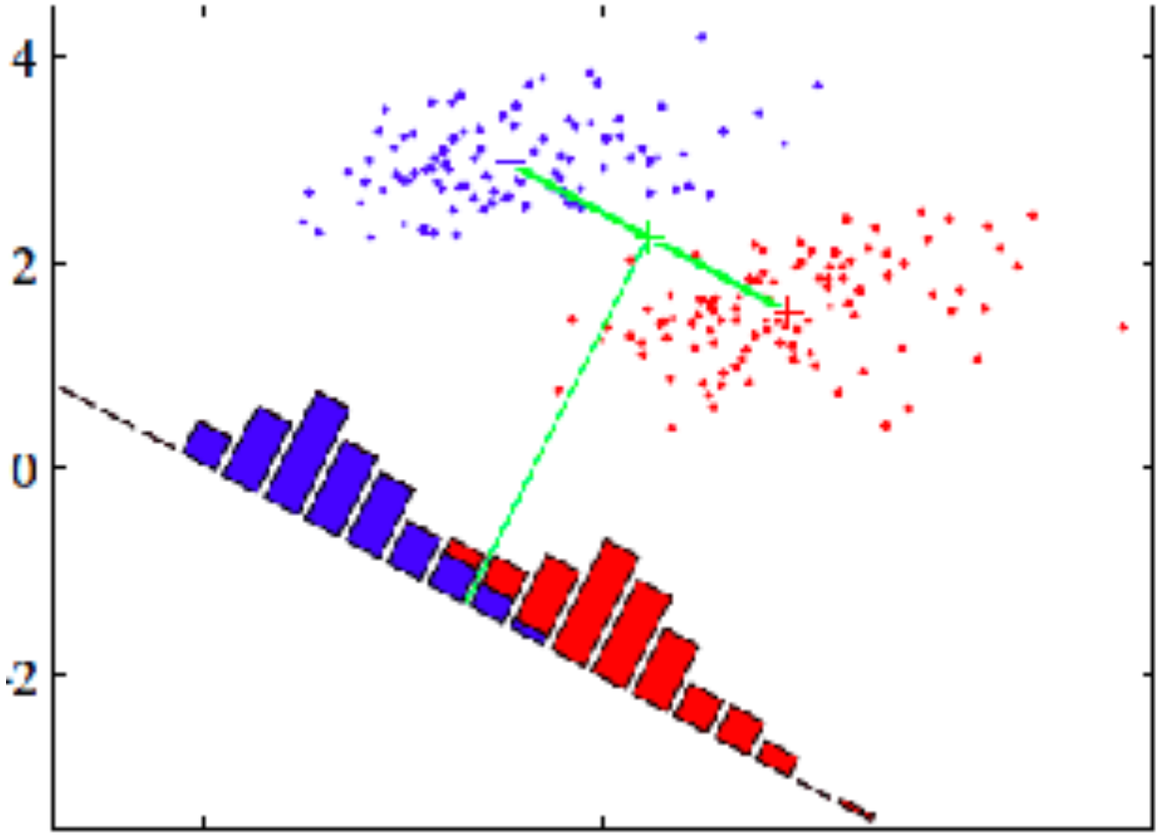


But what if we have labels for each data point?



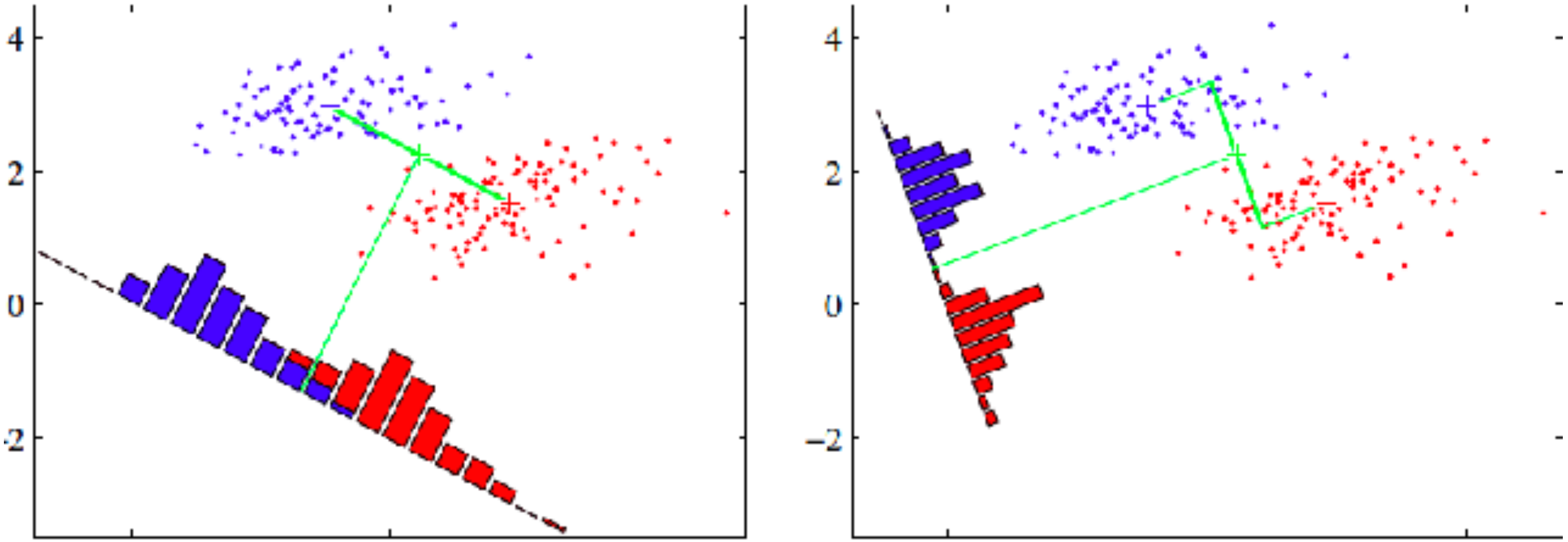
Slide adapted from Percy Liang

Linear Discriminant Analysis



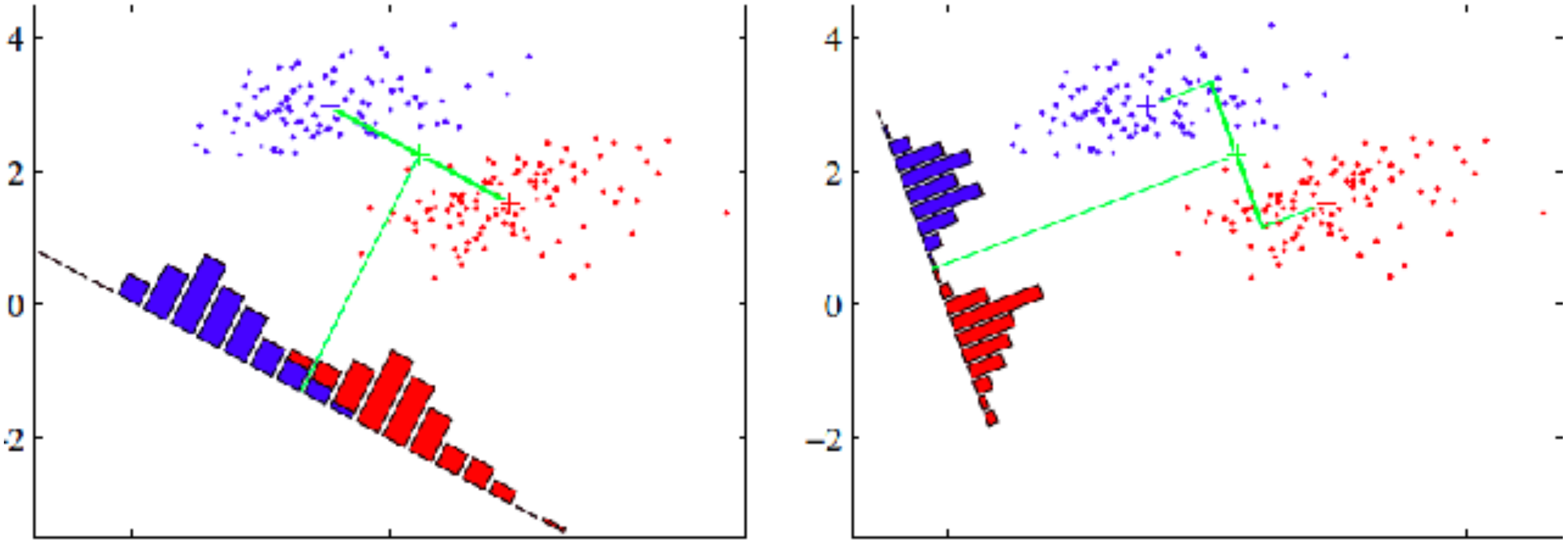
Slide adapted from Barani Raman

Linear Discriminant Analysis



Slide adapted from Barani Raman

Linear Discriminant Analysis



The objective of FDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible

In order to find a good projection vector we need to define a measure of separation between the projections

Slide adapted from Barani Raman

Linear Discriminant Analysis

Let $x_i \in \Re^d$ be the data in original space and $f(x_i) \in \Re^p$ be its mapping in the low dimensional space ($p < d$)

For simplicity and without loss of generality let the data belong to two classes C_1 and C_2

One possible measure of separation could be the distance between the projected means

$$\mu_i = \frac{1}{N} \sum_{x_j \in C_i} x_j$$: mean of the points in class C_i in original dimension

$$\tilde{\mu}_i = \frac{1}{N} \sum_{x_j \in C_i} f(x_j) = \frac{1}{N} \sum_{x_j \in C_i} w^T x_j = w^T \mu_i$$: mean of the points in projected dimension

We could then use distance between the two projected means as our objective

$$L(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T(\mu_1 - \mu_2)|$$

Linear Discriminant Analysis

Let $x_i \in \Re^d$ be the data in original space and $f(x_i) \in \Re^p$ be its mapping in the low dimensional space ($p < d$)

For simplicity and without loss of generality let the data belong to two classes C_1 and C_2

One possible measure of separation could be the distance between the projected means

$$\mu_i = \frac{1}{N} \sum_{x_j \in C_i} x_j$$: mean of the points in class C_i in original dimension

$$\tilde{\mu}_i = \frac{1}{N} \sum_{x_j \in C_i} f(x_j) = \frac{1}{N} \sum_{x_j \in C_i} w^T x_j = w^T \mu_i$$: mean of the points in projected dimension

We could then use distance between the two projected means as our objective

$$L(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T(\mu_1 - \mu_2)|$$

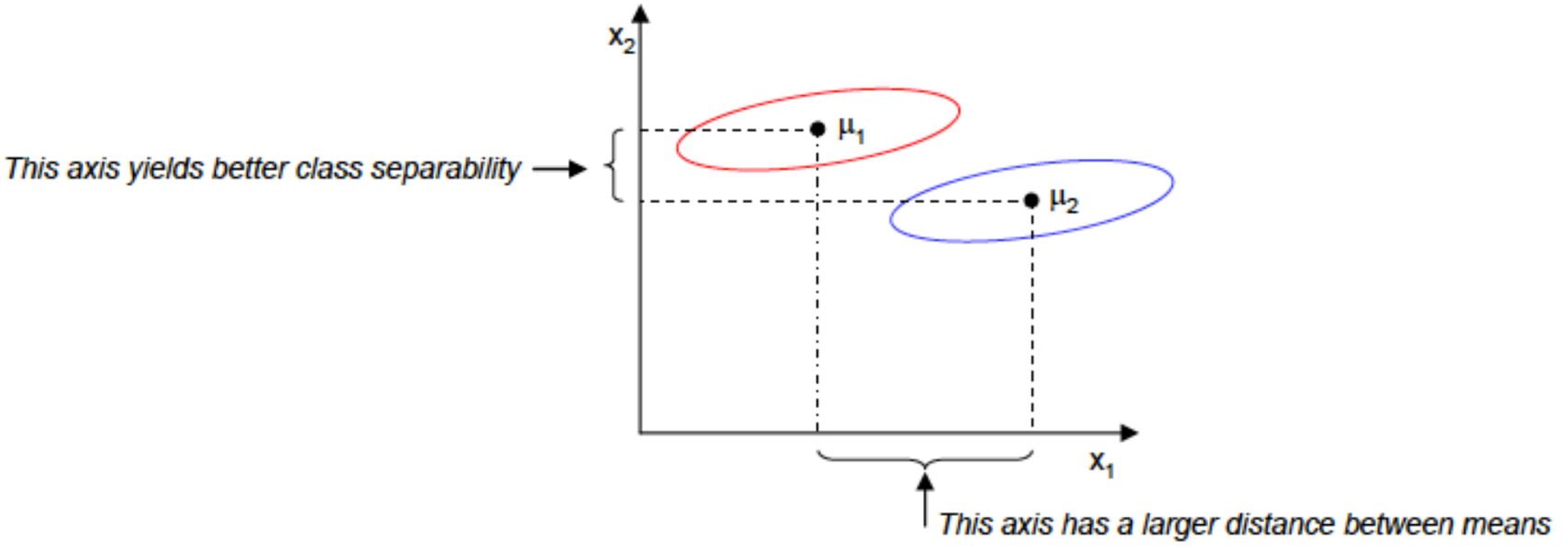


Figure from Barani Raman

Fischer Discriminant Analysis

Scatter for a class is defined as

$$\tilde{s}_i^2 = \sum_{x_j \in C_i} (f(x_j) - \tilde{\mu}_i)^2 = \sum_{x_j \in C_i} (w^T x_j - \tilde{\mu}_i)^2$$

Within-class scatter is defined as $\tilde{s_1}^2 + \tilde{s_2}^2$

Fischer Discriminant Analysis

Scatter for a class is defined as

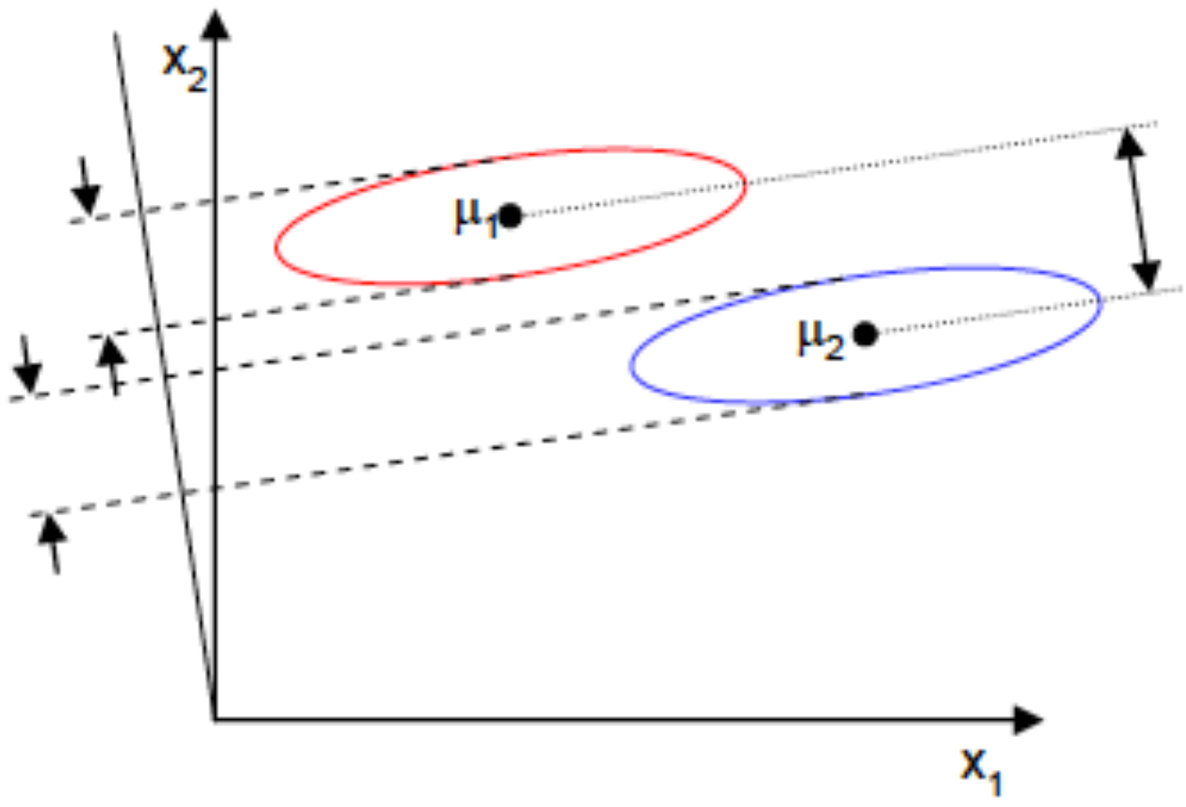
$$\tilde{s}_i^2 = \sum_{x_j \in C_i} (f(x_j) - \tilde{\mu}_i)^2 = \sum_{x_j \in C_i} (w^T x_j - \tilde{\mu}_i)^2$$

Within-class scatter is defined as $\tilde{s}_1^2 + \tilde{s}_2^2$

Fischer says, maximize a function that represents the difference between the class means, normalized by within-class scatter

$$L(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

In words: find w so as to increase the distance between class means and decrease the variance within classes



Slide adapted from Barani Raman

Non-Linear Dimensionality Reduction

The general objective of dimensionality reduction problem can be written as

$$L(W, Z) = \sum_{i=1}^{N} \|x_i - f_W(z_i)\|$$

In the linear setting, f_W is a linear function of the parameters W ($f_W(z) = Wz$)

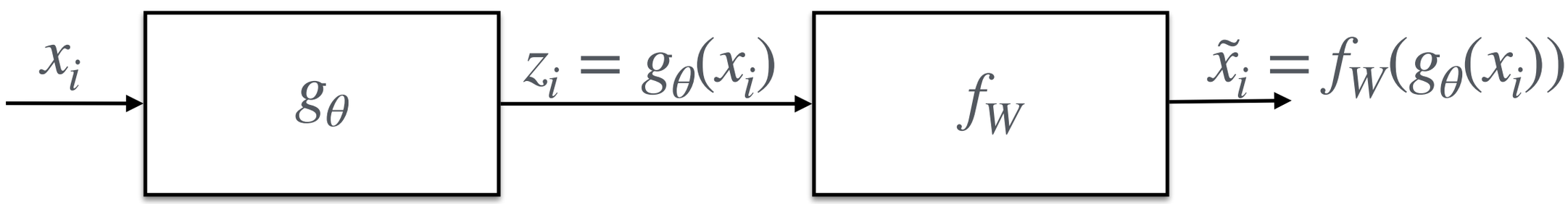
In the non-linear setting it could be any function

Another way to look at it is in the form of an encoder-decoder architecture

Encoder: g_θ and Decoder: f_W

Then the dimensionality reduction objective is

$$J(\theta, W) = \sum_{i=1}^{N} \|x_i - f_W(g_\theta(x_i))\|$$



End of Lecture 12