

# Unsupervised Learning

We have thus far focused on *Supervised Learning* where we are given training set

$$\langle \mathbf{X}, \mathbf{y} \rangle = [\mathbf{x}^{(i)}, \mathbf{y}^{(i)} | 1 \leq i \leq m]$$

and we wish to construct a function to map from arbitrary feature vectors  $\mathbf{x}$  of length  $n$  to some target/label  $\mathbf{y}$ .

We now switch to *Unsupervised Learning* where we are given

$$\mathbf{X} = [\mathbf{x}^{(i)} | 1 \leq i \leq m]$$

That is: feature vectors *without* associated target/labels.

This may feel somewhat strange ? What can we do given only features ?

Quite a bit ! Learning relationships *between* features enables us to

- Make recommendations based on similarity to other examples
- Enables us to visualize the data and discover relationships among examples
- Filter out "noise" or low information features

For example: from the perspective of some companies **you** are a feature vector !

- Several thousand attributes that define your past behavior
  - Purchases
  - Books read, movies viewed (one feature per book/movie)
  - Music, food preferences
  - Activities, hobbies

- Sparse vector
  - You've seen only a small fraction of the thousands of possible movies (one feature per movie)
- "You may also like" recommendation
  - The relationship between features among the training examples
  - Supports an association between a subset of features that are defined for you
  - And other features (movies/products) that are not yet defined for you

# Dimensionality reduction

Discovering the relationship among features can facilitate a more compact representation of feature vectors.

Let

$$\mathbf{X}_j = [\mathbf{x}_j^{(i)} \mid 1 \leq i \leq m] \text{ for } 1 \leq j \leq n$$

(i.e.,  $\mathbf{X}_j$  denotes a column of  $\mathbf{X}$ ; feature  $j$  across all examples)

denote the values of feature  $j$  among the  $m$  examples in the training set.

- So  $\mathbf{X}_j$  is a vector of length  $m$ .

Is it possible that many  $(\mathbf{X}_j, \mathbf{X}_{j'})$  pairs are highly correlated ( $j' \neq j$ )?

*Dimensionality reduction* is the process of representing a dataset

- That has  $n$  raw features
- With  $n' \ll n$  synthetic features
- While retaining *most* of the information

## Examples

### Color 3D object to Black/white 2D still image

- Lose Depth, Color dimensions
  - Spatial dimensions ( $1000 \times 1000 \times 1000$ )
  - Color dimension: 3
  - $n = 1000 * 1000 * 1000 * 3$
  - $n' = 1000 * 1000$   
 $= \frac{n}{1000*3}$

For the purpose of recognizing the object, little information is lost



## Equity time series

Consider an equity index (e.g., the S&P 500) of  $n = 500$  stocks

- An example  $\mathbf{x}^{(i)}$  is a vector of daily returns of length  $n = 500$ 
  - Row  $i$  of  $\mathbf{X}$  corresponds to one day of returns, across all  $n$  stocks
- $\mathbf{x}_j^{(i)}$  is the daily return of stock  $j$  on day  $i$
- $\mathbf{X}_j$  is the *timeseries* of daily returns of stock  $j$
- $\mathbf{X}_j, \mathbf{X}_{j'}, j \neq j'$  can be correlated
  - All stocks in the "market" tend to move up/down together
  - Daily returns of stocks with similar characteristics tend to be more similar than for stocks with differing characteristics
    - Industry, Size

One way to interpret the high mutual correlation among equity returns

- There are *common influences (factors)* affecting many individual equities
- Pair-wise correlation of equity returns (i.e., features) arises through influence of the shared factors

We can write this mathematically:

Let  $\tilde{\mathbf{X}}_{\text{index}}$  be the time series of daily returns of a factor ("the market") that affects *all* equities

$$\begin{aligned}\mathbf{X}_1 &= \beta_1 * \tilde{\mathbf{X}}_{\text{index}} + \epsilon_1 \\ \mathbf{X}_2 &= \beta_2 * \tilde{\mathbf{X}}_{\text{index}} + \epsilon_2 \\ &\vdots \\ \mathbf{X}_{500} &= \beta_{500} * \tilde{\mathbf{X}}_{\text{index}} + \epsilon_{500}\end{aligned}$$

The return timeseries  $\mathbf{X}_j$  of each stock  $j$  in the index is decomposed into

- The return timeseries associated with factor  $\tilde{\mathbf{X}}_{\text{index}}$  :  $\beta_j * \tilde{\mathbf{X}}_{\text{index}}$
- A return timeseries  $\epsilon_j$  that is stock-specific
- the return of  $j$  at time  $i$  is

$$\mathbf{x}_j^{(i)} = \beta_j * \mathbf{x}_{\text{index}}^{(i)} + \epsilon_j^{(i)}$$

Note that we have actually *increased* the number of features

- From  $n$ 
  - $\mathbf{x}_j$  for  $1 \leq j \leq n$
- To  $(n + 1)$ 
  - $\tilde{\mathbf{x}}_{\text{index}}$
  - $\epsilon_j$  for  $1 \leq j \leq n$

But, by adding another factor

- e.g., a "size" factor
- similar to  $\tilde{\mathbf{X}}_{\text{index}}$  in that it influences all equities

$$\mathbf{X}_1 = \beta_{1,\text{idx}} * \tilde{\mathbf{X}}_{\text{index}} + \beta_{1,\text{size}} * \tilde{\mathbf{X}}_{\text{size}} + \epsilon'_1$$

$$\mathbf{X}_2 = \beta_{2,\text{idx}} * \tilde{\mathbf{X}}_{\text{index}} + \beta_{2,\text{size}} * \tilde{\mathbf{X}}_{\text{size}} + \epsilon'_2$$

$\vdots$

$$\mathbf{X}_{500} = \beta_{500,\text{idx}} * \tilde{\mathbf{X}}_{\text{index}} + \beta_{500,\text{size}} * \tilde{\mathbf{X}}_{\text{size}} + \epsilon'_{500}$$

the magnitude of the stock-specific  $\epsilon'_j$  decreases compared to the original  $\epsilon_j$

- some of the return previously attributed to  $\epsilon_j$
- has been explained by  $\beta_{j,\text{size}} * \mathbf{X}_{\text{size}}$

As we add even more factors

- some may be specific to *sub-sets* of the universe
  - e.g., industry factors
- the stock-specific  $\epsilon$  series approaches 0

Once this occurs

- we can drop the  $\epsilon$
- and have a model with many fewer factors than  $n$

We thus obtain an approximation of

- the effect of  $n = 500$  features
- with  $n \gg n'$

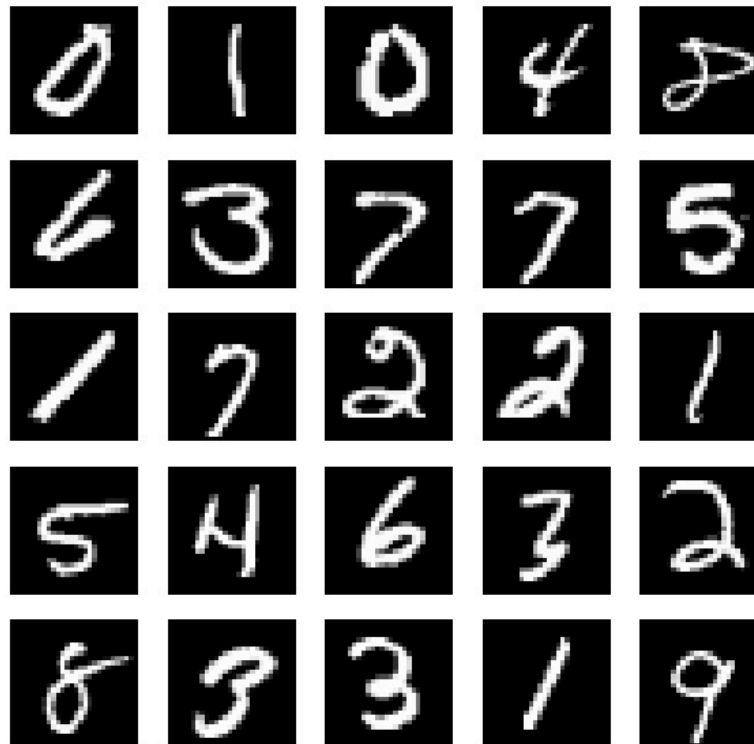
## Representing MNIST digits with 20% of the information

We will subsequently show how to represent the MNIST digits ( $n = 784$ ) with vectors of length  $n' \approx 150$

Here's what happens when

- We encode the digits with vectors of length  $n'$
- Perform the inverse mapping back to vectors of length  $n$  so we can display as a  $(28 \times 28)$  image

PCA: reconstructed MNIST digits (95% variance)







The reconstructed images are a little blurry (compared to the originals) but still very recognizable.

So it is *possible* to represent the information of the raw 784 features with only 20% ( $\approx 150$ ) as many synthetic features.

In other words: 80% of the pixels may be somewhat redundant.

Where are the correlated features in images of digits?

Consider the examples consisting of the  $(28 \times 28)$  pixel grids representing the MNIST digit dataset.

Here are some cases to consider:

Let  $j, j'$  be indices of two pixels in one of the 4 corners of the  $(28 \times 28)$  grid

- Most pixels in the corners are the same (background) colors so the correlation of  $\mathbf{x}_j$  and  $\mathbf{x}_{j'}$  is high

Let  $j, j'$  be indices of two pixels that lie in a vertical line in the center of the  $(28 \times 28)$  grid

- They will be somewhat correlated because they have the same value in 10% of the images
  - Corresponding to images of digit "1"

# Uses of dimensionality reduction

## Feature Engineering

If  $n$  is large and many features are mutually correlated

- A reduced number  $n' \ll n$  of synthetic features
- Obtained by dimensionality reduction techniques
- May result in better models
  - Some models, like Linear Regression, may be sensitive to correlated features (collinearity)

## Clustering

Dimensionality reduction can facilitate an understanding of the structure of examples.

Consider: Are the  $m$  examples in the training set

- Uniformly distributed across the  $n$  dimensional space ?
- Do they form *clusters* of examples with similar feature vectors ?

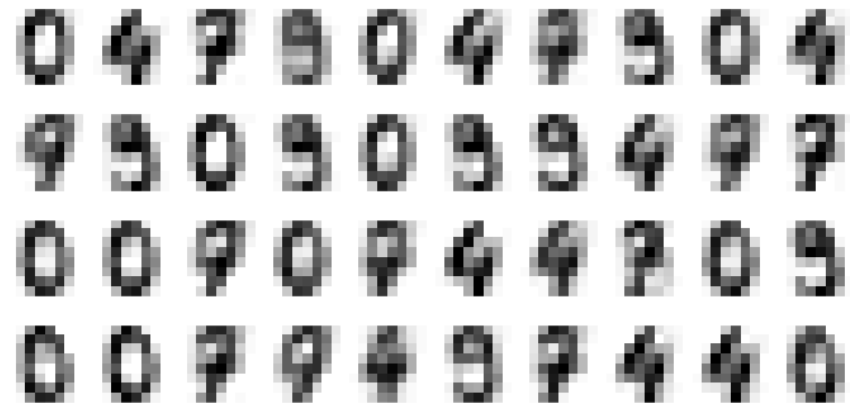
Unfortunately: it's hard to visualize  $n$  dimensions when  $n$  is large.

- By reducing the number of dimensions
- We may be able to visualize related examples
- In such a way that the reduced dimension examples don't lose too much information

Let's illustrate with a limited subset of the smaller ( $8 \times 8$ ) digits.

8 x 8 digits, subset

Digits subset: [0, 4, 7, 9]





It would be difficult to visualize an example in  $n = (8 * 8) = 64$  dimensional space.

By transforming each example to a smaller number ( $n' = 2$ ) of synthetic features we *can* visualize

- Each example as a point in two dimensional space

8 x 8 digits, subset clusters



You can see that our  $m \approx 700$  examples form 4 distinct clusters.

- The clusters were formed
  - Based solely on features

It turns out that the clusters correspond to examples mostly representing a single digit.

- The clusters organized themselves based on similarity of features
- This is unsupervised ! No targets were used in forming the clusters!
- We use the hidden target merely to color the point, not to form the clusters

This hints that dimensionality reduction may be useful for *supervised* learning as well

- Use commonality of features to reduce dimension
- Reduced dimensions more independent
  - Better mathematical properties (reduced collinearity)
  - More interpretable
- Under assumption that
  - Examples with similar features (i.e., in same cluster) have similar targets

## Noise reduction

Consider

- The MNIST example, where we reduced  $n$  by 80% without losing visual information.
- The equity return example, where the stock specific return  $\epsilon_j$  became increasingly small

Both examples suggest that there are many features

- With small significance
- Or that represent "noise" In the latter case, dropping features actually improves data quality by eliminating irrelevant features.

In [5]: `print("Done")`

Done

