Correlated features

Our goal is to find ways to reduce the dimensionality of feature vectors.

Let's explore correlated features in the notebook on <u>Correlated features</u> (<u>Unsupervised Correlated Features.ipynb</u>)

Principal Components: An alternate basis for our examples

Given that the features may be correlated

- We saw how changing the basis
- Can express the same examples
- In an alternate basis that is perhaps smaller

Let's formalize the notion of <u>alternate basis (Unsupervised.ipynb#Alternate-basis)</u>

Principal components: introduction

We have seen how we can express the examples in ${f X}$ in two coordinate systems

- The one with "original" features
- An alternate basis with "synthetic features"

Principal Components Analysis is the mechanism that we use

- To discover the new, alternate basis
- To find the feature values of examples, as measured in the alternate basis

Let's visit the notebook section introducing PCA (Unsupervised.ipynb#What-is-PCA)

PCA: The math

The goal of PCA is to find a way of expressing examples f X

- $\bullet \ \ \text{In a new basis} \ V^T$
- ullet With feature values $ilde{\mathbf{X}}$

$$\mathbf{X} = \mathbf{\tilde{X}}V^T$$

That is, we decompose ${f X}$ into a product

 \bullet factorization of ${f X}$

Let's go to the <u>notebook section on Matrix factorization (Unsupervised.ipynb#PCA-via-Matrix-factorization)</u> to explore how to factor \mathbf{X} .

PCA: reducing the number of dimensions

Thus far

- ullet Both the original basis and the new basis V have consisted of n basis vectors
- No information has been lost by the basis transformation

$$\mathbf{X} = \tilde{\mathbf{X}} V^T$$

If we are willing to lose some information

$$\mathbf{X}' pprox \mathbf{X}$$

we can achieve dimensionality reduction

- By an alternate basis $(V^T)'$ of $r \leq n$ basis vectors
- With synthetic feature vectors \mathbf{X}' of length r

That is: $\tilde{\textbf{X}}'$ is a reduced dimension representation.

Questions to consider

- Which synthetic features to drop
- How many synthetic features to drop/keep

Let's go to the notebook section on <u>dimensionality reduction</u> (<u>Unsupervised.ipynb#Dimensionality-reduction</u>)

Transforming between original and synthetic features

We have thus far been concerned with the transformation

- ullet From original features ${f X}$
- ullet To synthetic features $ilde{\mathbf{X}}$

We can also go in the opposite direction: from $ilde{\mathbf{X}}$ back to original features \mathbf{X}

Let's go to the <u>notebook section on inverse transformation (Unsupervised.ipynb#The-inverse-transformation)</u>

PCA in action

An example will hopefully tie together all the concepts.

Let's visit the <u>notebook section on PCA of small digits (Unsupervised.ipynb#Example:-Reconstructing-\$\x\$-from-\$\tilde\x\$-and-the-principal-components)</u>

Choosing the number of reduced dimensions

Let's visit the <u>notebook section on PCA of MNIST (Unsupervised.ipynb#MNIST-example)</u> in order to see how the quality of approximation varies with the number of features in $\tilde{\mathbf{X}}$

PCA in Finance

Long before Machine Learning became popular, PCA was used to "explain" the yield curve.

A Yield Curve is a vector of features

- ullet Whose length n corresponds to the number of bond maturities
- $\mathbf{x}_{j}^{(\mathbf{i})}$ is the yield, on day i of the j^{th} bond
 - ullet j increases with maturity

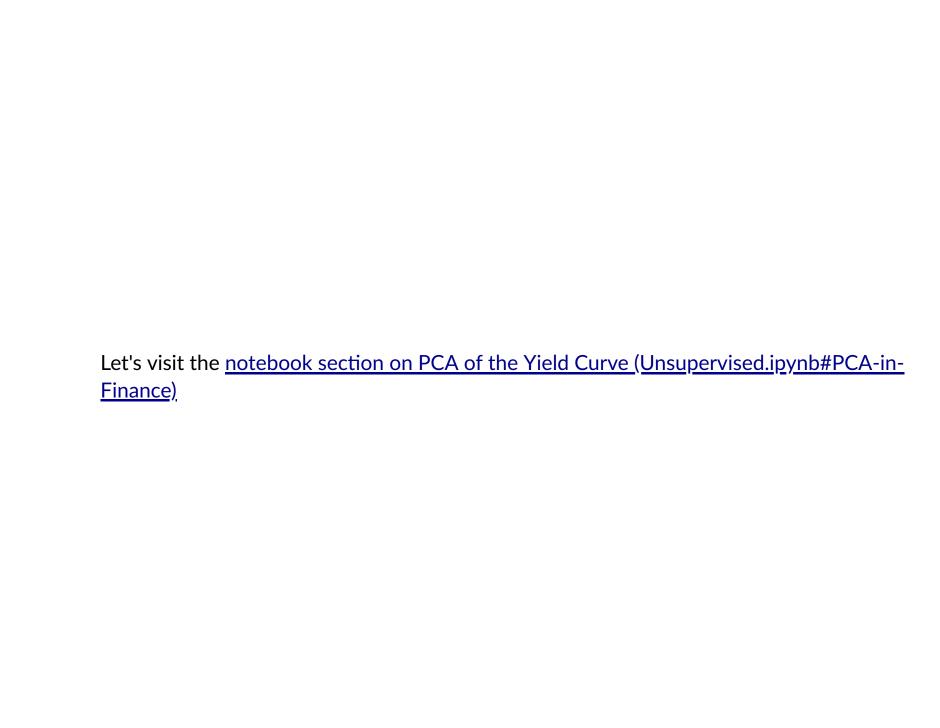
Does the yield of each maturity change (from day to day)

- Independently of other maturities?
- Or are there a small number of "common factors"/"concepts" that drive daily yield changes ?

PCA can help us answer the question.

In the process, we are also able to *interpret* the common factors

Which helps our intuition



Pseudo Matrix Factorization and Recommender Systems

You have no doubt been to a website that

- Has made a recommendation to you
- Based on your personal "features"

You might also like ...

How does this work?

- ullet You are an example $\mathbf{x^{(i)}}$, expressed as a vector of features
 - $\mathbf{x}_j^{(\mathbf{i})}$ is your "rating" for product j
- ullet The number n of products is large
 - ullet You have rated only a small fraction $n_i < n$
- ullet You have not rated product j^\prime
 - How can the system recommend j' to you?

PCA to the rescue!

- ullet Perform PCA on ${f X}$ (m is number of users; n is number of products)
- The Principal Components are
 - "Concepts" that identify groups of products

We will

- ullet Re-express your ratings of concrete product $\mathbf{x^{(i)}}$
- Into strength of concepts $\tilde{\mathbf{x}}^{(i)}$
- ullet Find other users i' with similar strength of concepts

$$ilde{\mathbf{x}}^{(i')} pprox ilde{\mathbf{x}}^{(\mathbf{i})}$$

- ullet Deduce that you (user i) have similar tastes to user i'
- ullet Recommend to you (user i) any product j
 - where $\mathbf{x}_{j}^{i')}$ is high

This is roughly how Netflix recommendations work.

- Products are Movies
- Principal Components ("concepts") turn out to be Movie genres
 - Action, Comedy, Romance, Gender-specific

So if your movie preferences lean to Comedy, Netflix will recommend to you Comedytype movies Although this seems like a simple application of PCA

- There is a giant catch!
- $oldsymbol{\cdot}$ $oldsymbol{X}$ is sparse (lots of empty entries)
 - How many of the thousands of movies in Netflix have you rated?

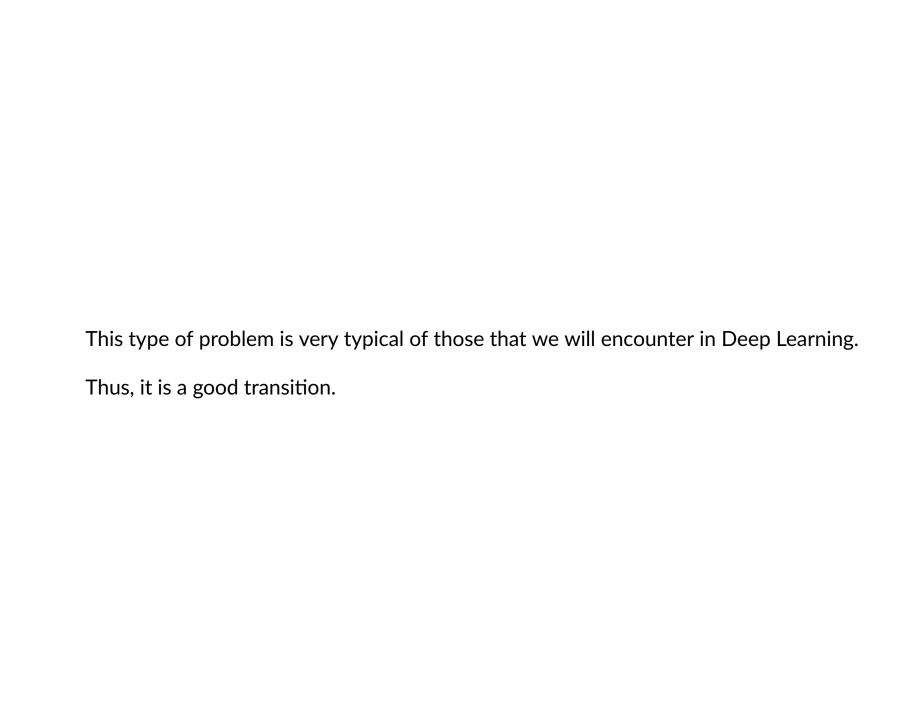
How do we factor a matrix with undefined entries?

Let's go the the <u>notebook section on Pseudo Matrix Factorization</u> (<u>Unsupervised.ipynb#Recommender-Systems:-Pseudo-SVD</u>)

Pseudo Matrix factorization wrap-up

The techniques in Pseudo factorization are a nice bridge between Classical ML and Deep Learning

- An interesting Loss function
- Not amenable to closed form solution (because of missing entries)
- But approximated using our generic optimization tool
 - Gradient Descent



```
In [4]: print("Done")
```

Done