Matrix Factorization

Brooks Paige

Week 8

Principal Components in practice

PCA recap

We covered a bit the motivation and derivation, but let's recap. PCA:

- Learns a linear projection from data $\mathbf{x}_i \in \mathbb{R}^D$ to a low-dimensional space $\mathbf{z}_i \in \mathbb{R}^K$, $K \ll D$
- This projection maximizes the amount of explained variance, or (equivalently) minimizes the reconstruction error

PCA recap

In practice, compute it from a data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ by

1. subtracting the mean $\mu = \frac{1}{N} \sum \mathbf{x}_i$, with $\tilde{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}$
2. decomposing using SVD:

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

PCA recap

In practice, compute it from a data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ by

1. subtracting the mean $\mu = \frac{1}{N} \sum \mathbf{x}_i$, with $\tilde{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}$
2. decomposing using SVD:

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

...where $\mathbf{D}$ is a diagonal matrix of singular values. The $\mathbf{V}$ matrix corresponds to the PCA projection; take only a subset of columns of $\mathbf{V}$ for dimensionality reduction. The transformation is given by

$$\mathbf{z}_i = \mathbf{V}^\top(\mathbf{x}_i - \boldsymbol{\mu}) \qquad \hat{\mathbf{x}}_i = \mathbf{V}\mathbf{z}_i + \boldsymbol{\mu}$$

PCA recap

In practice, compute it from a data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ by

1. subtracting the mean $\mu = \frac{1}{N} \sum \mathbf{x}_i$, with $\tilde{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}$
2. decomposing using SVD:

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

…where $\mathbf{D}$ is a diagonal matrix of singular values. The $\mathbf{V}$ matrix corresponds to the PCA projection; take only a subset of columns of $\mathbf{V}$ for dimensionality reduction. The transformation is given by

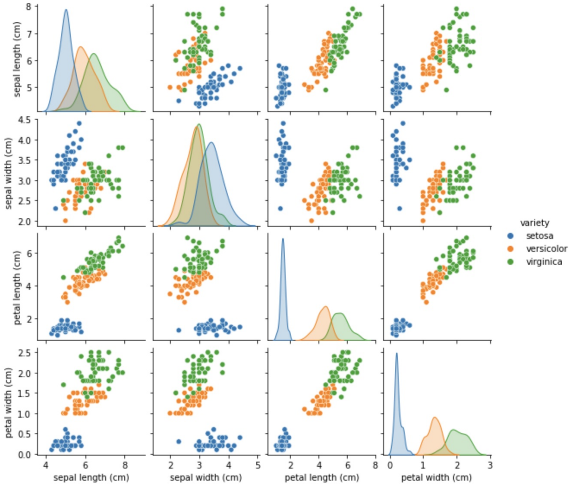
$$\mathbf{z}_i = \mathbf{V}^\top(\mathbf{x}_i - \boldsymbol{\mu}) \qquad \hat{\mathbf{x}}_i = \mathbf{V}\mathbf{z}_i + \boldsymbol{\mu}$$

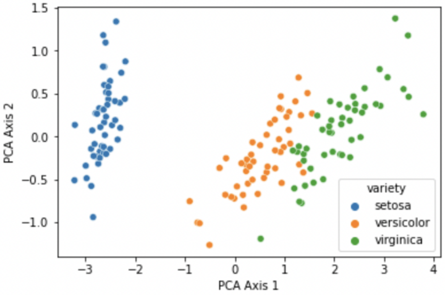
This is equivalent to finding the eigenvectors of the covariance matrix:

$$\mathbf{S} = \mathrm{Cov}(\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top$$

$$\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$$

Dimensionality reduction





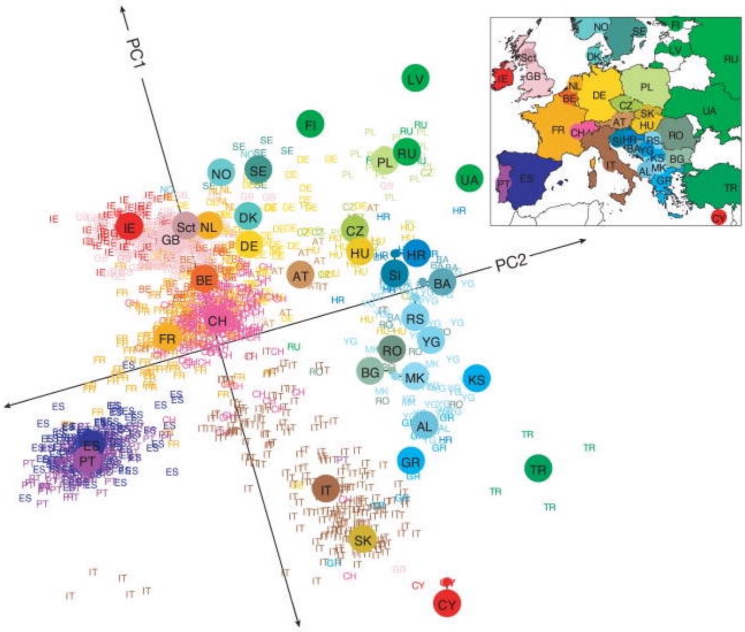
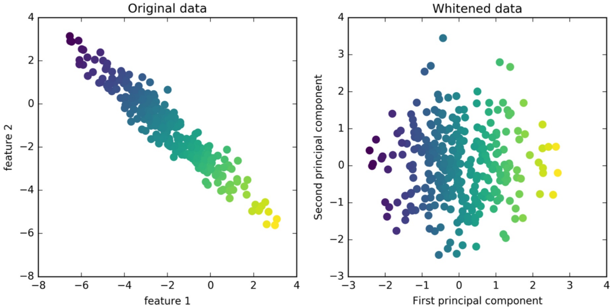


Figure: Novembre et al., Nature (2008)

Preprocessing / whitening transform



Applying PCA using ALL the projections gives $\mathbf{Z} = \mathbf{V}^\top \mathbf{X} \in \mathbb{R}^D$.

This "whitens" the data to have $\mathrm{Cov}(\mathbf{Z}) = \mathbf{I}$, which can be useful.

Figure: Andreas Müller

Learning latent features

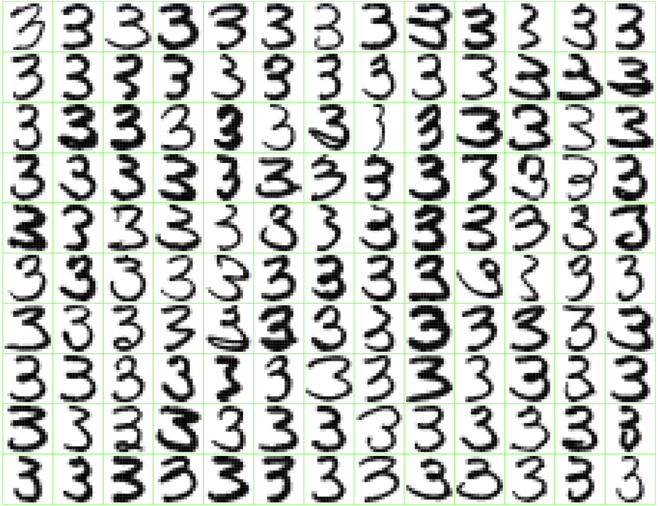
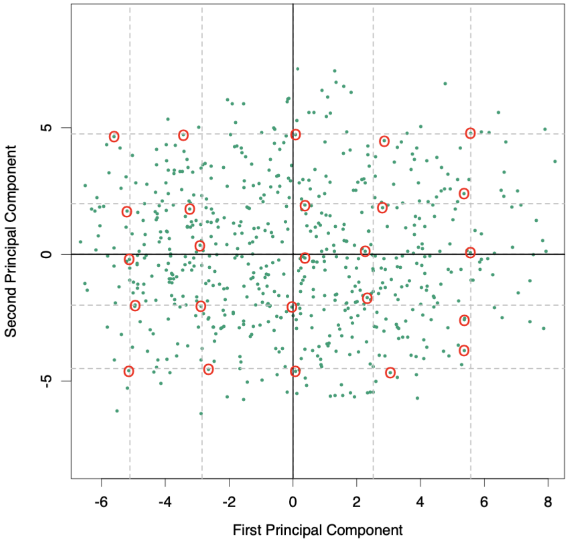


Figure: ESL

Learning latent features



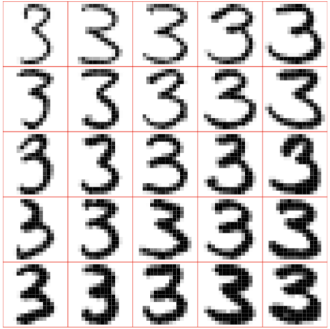
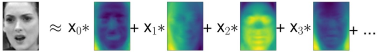
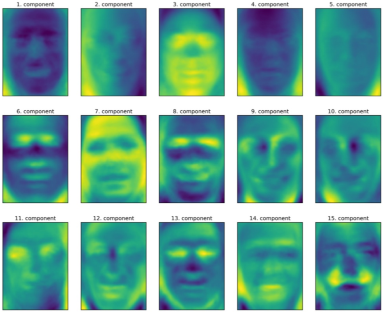


Figure: ESL

Learning latent features



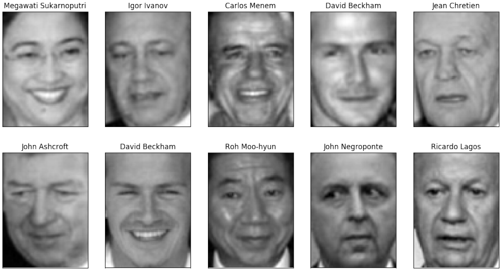


Remember:
Signs don't mean anything!

Slide: Andreas Müller

Reconstruction

Outlier detection

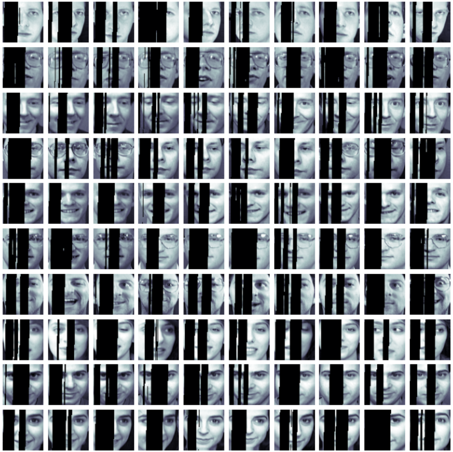


Best reconstructions

Worst reconstructions

Figure: Andreas Müller

Missing data



- Training dataset (400 faces) is missing large black bands
- Third image shows reconstruction of just the missing region

Figure: David Barber

Last word on PCA

- Projections are fairly interpretable
- Projections only defined up to a sign
  - (or jointly up to a rotation!)
- Sensitive to scaling of input dimensions (if you change the units of one column of $\mathbf{X}$, then its relative contribution to the variance changes!)
  - Can be addressed by rescaling all input dimensions to have variance of 1, but this has its own issues
- Preprocessing can be useful for nearest-neighbor methods (filters out "noisy" dimensions)
- Doesn't make sense for data that isn't real-valued
- Linear projections may or may not be adequate for complex data

Non-negative matrix factorization

Positive combination of positive bases

In PCA, we supposed each data point $\mathbf{x}_i$ could be written as

$$\mathbf{x}_i \approx \boldsymbol{\mu} + \sum_{k=1}^{K} z_{ik}\mathbf{v}_k$$

where $\mathbf{v}_k$ are the "basis" vectors, and each z_{ik} corresponds to the individual additive contribution to $\mathbf{x}_i$.

Positive combination of positive bases

In PCA, we supposed each data point $\mathbf{x}_i$ could be written as

$$\mathbf{x}_i \approx \boldsymbol{\mu} + \sum_{k=1}^{K} z_{ik} \mathbf{v}_k$$

where $\mathbf{v}_k$ are the "basis" vectors, and each z_{ik} corresponds to the individual additive contribution to $\mathbf{x}_i$.

For **data which is positive-valued**, then we might instead want a model which constrains both the weights and the bases to be nonnegative. To avoid confusion, we will use different notation here:

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}, \text{ or } \qquad \mathbf{x}_i \approx \sum_{k=1}^{K} w_{ik} \mathbf{h}_k,$$

where $\mathbf{W}$ is $N \times K$, $\mathbf{H}$ is $K \times D$, and we have $w_{ik} \geq 0, h_{kd} \geq 0$.

Non-negative matrix factorization (NNMF)

In PCA, our reconstruction used a squared-error loss, corresponding to a Gaussian log-likelihood.

Non-negative matrix factorization (NNMF)

In PCA, our reconstruction used a squared-error loss, corresponding to a Gaussian log-likelihood.

In NNMF, since we have positive-valued data, we instead consider a loss which corresponds to a Poisson log-likelihood, with mean $\mathbf{WH}$:

$$L(\mathbf{W}, \mathbf{H}) = \sum_{i=1}^{N} \sum_{j=1}^{D} x_{ij} \log(\mathbf{WH})_{ij} - (\mathbf{WH})_{ij}.$$

Non-negative matrix factorization (NNMF)

In PCA, our reconstruction used a squared-error loss, corresponding to a Gaussian log-likelihood.

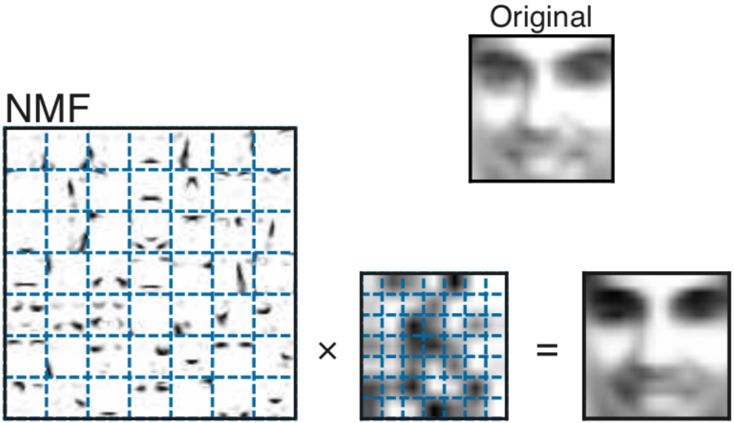
In NNMF, since we have positive-valued data, we instead consider a loss which corresponds to a Poisson log-likelihood, with mean $\mathbf{WH}$:

$$L(\mathbf{W}, \mathbf{H}) = \sum_{i=1}^{N} \sum_{j=1}^{D} x_{ij} \log(\mathbf{WH})_{ij} - (\mathbf{WH})_{ij}.$$

A local optimum can be found via an alternating maximization

$$w_{ik} \leftarrow \frac{\sum_{j=1}^{D} h_{kj} x_{ij} / (\mathbf{WH})_{ij}}{\sum_{j=1}^{D} h_{kj}}, \qquad h_{kj} \leftarrow \frac{\sum_{j=1}^{D} w_{ik} x_{ij} / (\mathbf{WH})_{ij}}{\sum_{j=1}^{D} w_{ik}}.$$

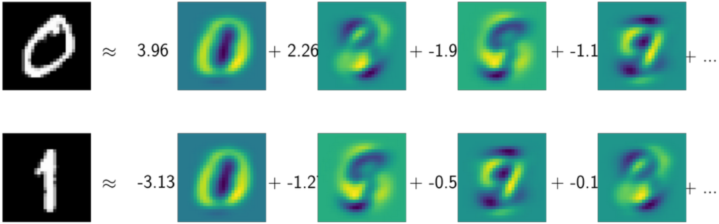
Non-negative decomposition of images



Images have positive-valued pixels; more interpretable decomposition than PCA

Figure: Lee & Seung (1999)

Learned features



PCA (above) vs NNMF (below)

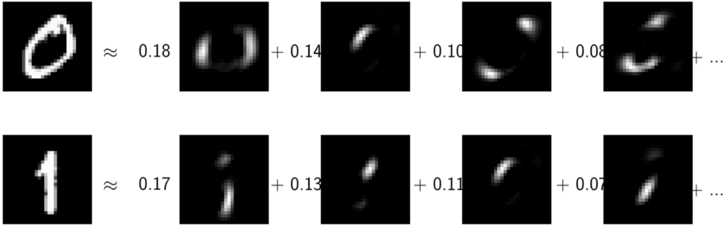
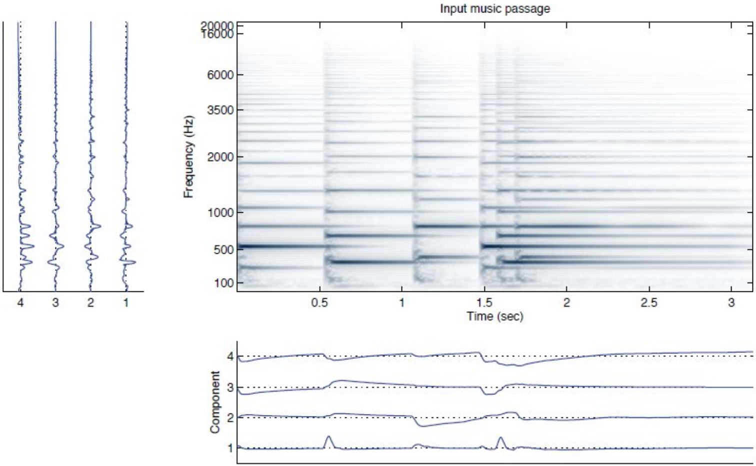


Figure: Andreas Müller

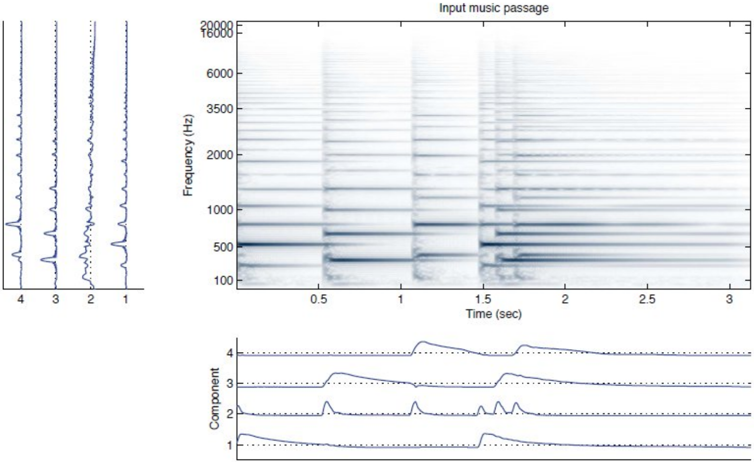
Decomposing music signals



Principal components on matrix spectrogram: no clear interpretation

Figure: Paris Smaragdis

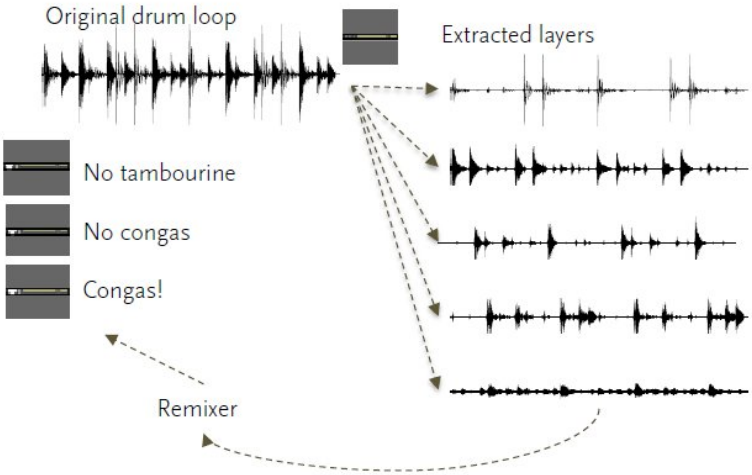
Decomposing music signals



NNMF on matrix spectrogram: recovers individual notes

Figure: Paris Smaragdis

Decomposing music signals



Separate recording into individual instruments

Figure: Paris Smaragdis

Modeling count or frequency data

NNMF is a good choice for matrices of count data.

Modeling count or frequency data

NNMF is a good choice for matrices of count data.

An example is in unsupervised representation learning for text, where NNMF functions as a topic model (one also known as "PLSA", or probabilistic latent semantic analysis):

Modeling count or frequency data

NNMF is a good choice for matrices of count data.

An example is in unsupervised representation learning for text, where NNMF functions as a topic model (one also known as "PLSA", or probabilistic latent semantic analysis):

- Documents correspond to multiple "topics"
- With a fixed dictionary of words, construct a data matrix $\mathbf{X}$ where each row contains per-document word counts
- Consider dropping "stop words" (common words like "and", "the", ...)
- Consider rescaling frequencies using **TF-IDF**: "term frequency"–"inverse document frequency"
  - Intuition: words that many times in one document, but are otherwise uncommon across documents, are more "important"

Modeling count or frequency data

NNMF is a good choice for matrices of count data.

An example is in unsupervised representation learning for text, where NNMF functions as a topic model (one also known as "PLSA", or probabilistic latent semantic analysis):

- Documents correspond to multiple "topics"
- With a fixed dictionary of words, construct a data matrix $\mathbf{X}$ where each row contains per-document word counts
- Consider dropping "stop words" (common words like "and", "the", ...)
- Consider rescaling frequencies using **TF-IDF**: "term frequency"–"inverse document frequency"
  - Intuition: words that many times in one document, but are otherwise uncommon across documents, are more "important"

(N.B. **Latent Dirichlet Allocation** *is often preferred for topic modelling)*

Topic modeling example

16,333 documents are taken from Associated Press corpus with a dictionary of 23,075 unique terms. Fit a topic model (NNMF) containing 4 topics.

Arts	Budgets	Children	Education
new	million	children	school
film	tax	women	students
show	program	people	schools
music	budget	child	education
movie	billion	years	teachers
play	federal	families	high
musical	year	work	public
best	spending	parents	teacher
actor	new	says	bennett
first	state	family	manigat
york	plan	welfare	namphy
opera	money	men	state
theater	programs	percent	president
actress	government	care	elementary
love	congress	life	haiti

(a)

The William **Randolph** **Hearst** **Foundation** will give **$ 1.25 million** to **Lincoln** **Center**, Metropolitan **Opera** Co., **New** **York** **Philharmonic** and Juilliard School. Our **board** felt that we had a **real** **opportunity** to **make** a **mark** on the **future** of the **performing** arts with these **grants** an **act** every **bit** as **important** as our **traditional** areas of **support** in health, medical **research**, education and the **social** **services**, **Hearst** **Foundation** **President** **Randolph** A. **Hearst** said Monday in **announcing** the **grants**. **Lincoln** Center's **share** will be **$200,000** for its **new** **building**, which will **house** **young** artists and **provide** **new** **public** **facilities**. The Metropolitan **Opera** Co. and **New** **York** **Philharmonic** will **receive** **$400,000** each. The Juilliard School, where **music** and the **performing** arts are taught, will get **$250,000**. The **Hearst** **Foundation**, a **leading** **supporter** of the **Lincoln** **Center** Consolidated Corporate **Fund**, will **make** its usual **annual** **$100,000** donation, too.

(b)

Slide: David Barber

Non-negative matrix factorization discussion

Pros

- Natural fit for positive-valued data; can be interpretable
- Meaningful signs; no “cancellation”

Non-negative matrix factorization discussion

Pros

- Natural fit for positive-valued data; can be interpretable
- Meaningful signs; no "cancellation"

Cons

- Only applicable to non-negative data
- Optimization procedure is non-convex; requires initialization
- "Interpretability" is unreliable
- Learned components aren't orthogonal, or naturally ordered
- Reducing K can completely change the basis functions (rather than simply selecting a subset)

Extensions

- **Recommender systems**: define the matrix $\mathbf{X}$ as the "user–item" matrix, where the entries are ratings.
  - The goal of the recommender system is to impute the values in the "missing" entries of the matrix, i.e. to fill in the gaps and estimate unknown ratings per-user

Extensions

- **Recommender systems**: define the matrix $\mathbf{X}$ as the "user–item" matrix, where the entries are ratings.
  - The goal of the recommender system is to impute the values in the "missing" entries of the matrix, i.e. to fill in the gaps and estimate unknown ratings per-user
- **Probabilistic variants of NNMF**: the algorithm in the slides here is the "classic" one, but it is prone to over-fitting. It's possible to define a probabilistic latent variable model (a Gamma-Poisson model) by putting a Gamma distribution over $\mathbf{w}_i$ and estimate the parameters using EM.

Extensions

- **Recommender systems**: define the matrix $\mathbf{X}$ as the "user–item" matrix, where the entries are ratings.
  - The goal of the recommender system is to impute the values in the "missing" entries of the matrix, i.e. to fill in the gaps and estimate unknown ratings per-user
- **Probabilistic variants of NNMF**: the algorithm in the slides here is the "classic" one, but it is prone to over-fitting. It's possible to define a probabilistic latent variable model (a Gamma-Poisson model) by putting a Gamma distribution over $\mathbf{w}_i$ and estimate the parameters using EM.
- **Latent Dirichlet Allocation**: a probabilistic mixed-membership model, in which each *document* is a probability distribution over *topics*, and each *topic* is a probability distribution over *words*.
  - Individual words are sampled from topics, topics sampled from documents

Not covered, but FYI

Independent Components Analysis

Goal: Find statistically *independent* components, not necessarily orthogonal

Many algorithms work by

1. whiten the data with PCA
2. find linear projections of the data which are as non-Gaussian as possible

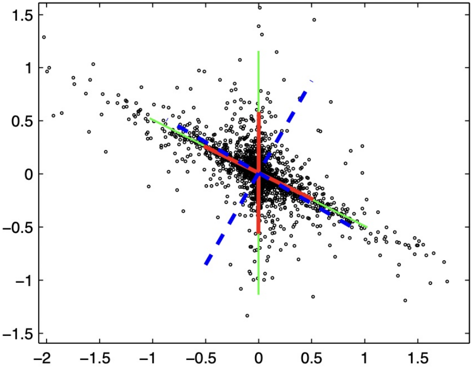
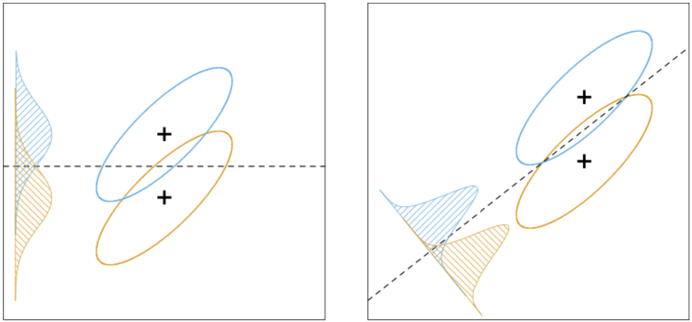


Figure: David Barber

Linear discriminant analysis

So far, everything was **unsupervised**. But, we can use labels, too.



e.g.: projection that maximizes variance *vs.* maximizes class separation

Figure: ESL

Non-linear extensions

There are a number of ways to introduce non-linear extensions to these models, representing the data as a **non-linear** combination of learned latent factors.

Non-linear extensions

There are a number of ways to introduce non-linear extensions to these models, representing the data as a **non-linear** combination of learned latent factors.

The most intuitive approach here is to use deep learning, with the same objective as "reconstruction" for PCA. Define

$$\mathbf{z}_i = f_\phi(\mathbf{x}_i) \qquad\qquad \hat{\mathbf{x}}_i = g_\theta(\mathbf{z}_i)$$

and minimize the loss

$$L(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - g_\theta(f_\phi(\mathbf{x}_i))\|_2^2.$$

This is called an **autoencoder**, with encoder f_ϕ and decoder g_θ.

Non-linear extensions

There are a number of ways to introduce non-linear extensions to these models, representing the data as a **non-linear** combination of learned latent factors.

The most intuitive approach here is to use deep learning, with the same objective as "reconstruction" for PCA. Define

$$\mathbf{z}_i = f_\phi(\mathbf{x}_i) \qquad \hat{\mathbf{x}}_i = g_\theta(\mathbf{z}_i)$$

and minimize the loss

$$L(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - g_\theta(f_\phi(\mathbf{x}_i))\|_2^2.$$

This is called an **autoencoder**, with encoder f_ϕ and decoder g_θ.

There are also kernel methods, including "kernelized" PCA, and Gaussian process latent variable models.