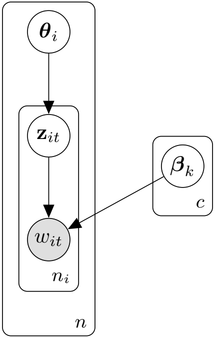
Machine Learning (CS 181):
16. Dimensionality Reduction

David C. Parkes and Sasha Rush

Spring 2017

Topic Model: Last Class



1. For each i pick a document-topic distribution $\boldsymbol{\theta}_i$.

2. For each word position t in the document,
   - Draw a topic indicator $\mathbf{z}_{it}$ from $\boldsymbol{\theta}_i$
   - Draw next word w_{it} from topic-word distribution $\boldsymbol{\beta}_{\mathbf{z}_{it}}$

Topic-Word Distributions

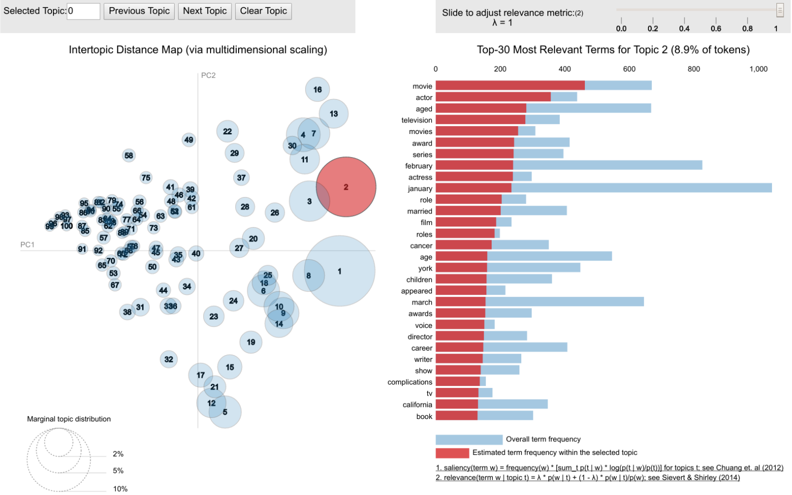
- $\boldsymbol{\beta}_1$; probability of any word in the "sports" topic
- $\boldsymbol{\beta}_2$; probability of any word in the "acting" topic
- $\boldsymbol{\beta}_3$; probability of any word in the "film" topic
- ...

- Each topic-word distribution $\boldsymbol{\beta}_1 \in \mathbb{R}^m$ where m is the vocabulary
- In real-life m is very big, up to 50,000 dimensions.
- How can we compare nearby topics?

Visualization from Demo

In [48]:
```
pyLDAvis.display(vis_data)
```

Out[48]:



- Shows topics ($\mathbb{R}^m$) in two-dimensions. Reduced dimensionality.

Contents

Review: Features in Supervised Learning

- In supervised learning wanted richer features for our input.
- Often means developing basis functions

$$\mathbf{x} \in \mathbb{R}^m \rightarrow \boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^d$$

- One strategy is to find higher-dimensional features, $d > m$
  - Periodic basis
  - Polynomial basis
  - Learned neural network features.

Dimensionality Reduction

Today's lecture:

- Assume high-dim $\mathbf{x}$ to start with.
- Try to find a *low-dim* vectors with $d < m$.
- Why is this helpful?
  - Interpretability
  - Reducing model size
  - Denoising of data

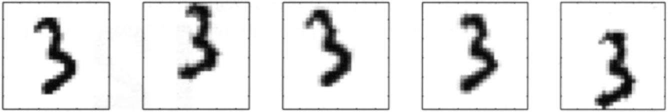
Why Reduce Dimensionality?

- Often times *signal* of data is in a low-dimension.
- But we only observe a *rendering* of the data in high-dimension with additional high-dimensional noise.
- This makes the data seem arbitrarily high-dimensional even though the true structure is more simple.
- Different from clustering, try to find latent lower dimensional representation.

Why Reduce Dimensionality?

- Often times *signal* of data is in a low-dimension.
- But we only observe a *rendering* of the data in high-dimension with additional high-dimensional noise.
- This makes the data seem arbitrarily high-dimensional even though the true structure is more simple.
- Different from clustering, try to find latent lower dimensional representation.

Example: Synthetic Digits [Bishop]



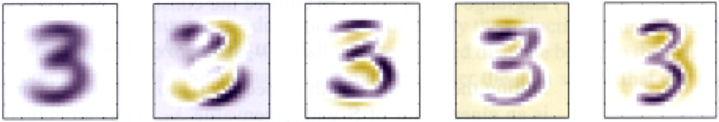
- Each $\mathbf{x}$ is an grey-scale image of a "3" in $\mathbb{R}^{100\times 100}$ $(m = 10000)$
- But each $\mathbf{x}$ generated from a much smaller source vector.
- Source vector transposition and rotation of same image + noise.
- Goal: Recover source vector and reverse transformation.

Lower Dimensional Basis

- Our aim will be to find a d-dimensional basis to represent these rendered images presented in m dimensions.
- Formally, this basis will be of the form,

$$\{\mathbf{u}_1 \in \mathbb{R}^m, \ldots, \mathbf{u}_d \in \mathbb{R}^m\} \quad \text{or} \quad \mathbf{U} \in \mathbb{R}^{d \times m}$$

- In the case of images, a sample $d = 4$ basis looks like this:



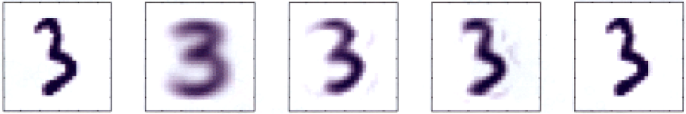
Four vectors forming an image basis (rows of $\mathbf{U}$) and their mean (left).

Reconstruction

- For each $\mathbf{x} \in \mathbb{R}^m$ in our data, our reduced dimensional vector will be called $\mathbf{z} \in \mathbb{R}^d$ (note: unlike clustering this is not a one-hot vector.)
- We try to reconstruct $\mathbf{x}$ using a linear combination of basis vectors,

$$z_1\mathbf{u}_1 + \ldots + z_d\mathbf{u}_d \quad \text{or} \quad \mathbf{U}^\top \mathbf{z}$$

- Here we reconstruct the original image with $d \in \{1, 10, 50, 250\}$. Note that it is lossy, as these are low-dimensional reconstructions.



An image $\mathbf{x}$ and reconstructions using different size d.

Review: Orthonomal Basis (Linear Algebra)

- Orthogonal vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{u}^\top \mathbf{v} = 0$$

- Normal vectors $\mathbf{u}$:

$$\mathbf{u}^\top \mathbf{u} = 1$$

- Orthonormal basis $\{\mathbf{u}_1, \ldots, \mathbf{u}_d\}$

$$\mathbf{u}_i^\top \mathbf{u}_j = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{o.w.} \end{cases}$$

Review: Change of Basis

- Assume we have any basis,

$$\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$$

- If vector $\mathbf{x} \in \mathbb{R}^m$ is represented coefficients in a different basis, we change to our new basis by *projecting* onto each basis vector.

$$\langle(\mathbf{x}^\top \mathbf{u}_1), \ldots, (\mathbf{x}^\top \mathbf{u}_m)\rangle$$

- This transformation m to m dim is lossless, can return to coordinate basis with linear combination of basis vectors.

$$(\mathbf{x}^\top \mathbf{u}_1)\mathbf{u}_1 + \ldots + (\mathbf{x}^\top \mathbf{u}_m)\mathbf{u}_m$$

Contents

Linear Dimensionality Reduction

Standard unsupervised learning problem:

- Given *normalized* $\mathbf{x}_1 \ldots \mathbf{x}_n$ in $\mathbb{R}^m$.
- Find:
  - Basis vectors $\mathbf{u}_1 \ldots \mathbf{u}_d$ in $\mathbb{R}^m$
  - Reconstruction coefficients $\mathbf{z}_1 \ldots \mathbf{z}_n$ in $\mathbb{R}^d$
- Example:
  - Combination of basis images to reconstruct digit.
  - Combination of basis topics to reconstruct true topics.
  - Combination of basis faces to reconstruct true faces.

Linear Dimensionality Reduction

Standard unsupervised learning problem:

- Given *normalized* $\mathbf{x}_1 \ldots \mathbf{x}_n$ in $\mathbb{R}^m$.
- Find:
  - Basis vectors $\mathbf{u}_1 \ldots \mathbf{u}_d$ in $\mathbb{R}^m$
  - Reconstruction coefficients $\mathbf{z}_1 \ldots \mathbf{z}_n$ in $\mathbb{R}^d$
- Example:
  - Combination of basis images to reconstruct digit.
  - Combination of basis topics to reconstruct true topics.
  - Combination of basis faces to reconstruct true faces.

Loss Function

As always we state our goal using a loss objective:

- Minimizes a least squares loss function:

$$
\begin{aligned}
\mathcal{L}(\mathbf{z}, \mathbf{U}) &= \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{U}^\top \mathbf{z}_i||_2^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{U}^\top \mathbf{z}_i)^\top (\mathbf{x}_i - \mathbf{U}^\top \mathbf{z}_i)
\end{aligned}
$$

- Where $\mathbf{U} \in \mathbb{R}^{d \times m}$ is a set of orthonormal vectors.
- Intuition: Find basis in d dimensions and coefficients $\mathbf{z}$ that reconstruct $\mathbf{x}$ vectors as close as possible.

Loss

Loss comes from this term, i.e. how well did we reconstruct.

$$||\mathbf{x}_i - \mathbf{U}^\top \mathbf{z}_i||$$

Now we assume there exists some a $\mathbf{v}_{d+1}, \ldots, \mathbf{v}_m$ that completes $\mathbf{u}_1, \ldots, \mathbf{u}_d$, giving a m dimensional orthonormal basis.
Using change of basis we can write $\mathbf{x}$ as:

$$\langle (\mathbf{x}^\top \mathbf{u}_1), \ldots, (\mathbf{x}^\top \mathbf{u}_d), (\mathbf{x}^\top \mathbf{v}_{d+1}), \ldots (\mathbf{x}^\top \mathbf{v}_m) \rangle$$

Claim: With fixed $\mathbf{u}_1, \ldots, \mathbf{u}_d$, best we can do is set $\mathbf{z}$ to match first d dimensions (projection onto $\mathbf{U}$):

$$\langle (\mathbf{x}^\top \mathbf{u}_1), \ldots, (\mathbf{x}^\top \mathbf{u}_d), 0, \ldots 0 \rangle = \mathbf{U}\mathbf{x} = \mathbf{z}$$

Loss

Loss comes from this term, i.e. how well did we reconstruct.

$$||\mathbf{x}_i - \mathbf{U}^\top \mathbf{z}_i||$$

Now we assume there exists some a $\mathbf{v}_{d+1}, \ldots, \mathbf{v}_m$ that completes $\mathbf{u}_1, \ldots, \mathbf{u}_d$, giving a m dimensional orthonormal basis.

Using change of basis we can write $\mathbf{x}$ as:

$$\langle (\mathbf{x}^\top \mathbf{u}_1), \ldots, (\mathbf{x}^\top \mathbf{u}_d), (\mathbf{x}^\top \mathbf{v}_{d+1}), \ldots (\mathbf{x}^\top \mathbf{v}_m) \rangle$$

Claim: With fixed $\mathbf{u}_1, \ldots, \mathbf{u}_d$, best we can do is set $\mathbf{z}$ to match first d dimensions (projection onto $\mathbf{U}$):

$$\langle (\mathbf{x}^\top \mathbf{u}_1), \ldots, (\mathbf{x}^\top \mathbf{u}_d), 0, \ldots 0 \rangle = \mathbf{U}\mathbf{x} = \mathbf{z}$$

Loss

Loss comes from this term, i.e. how well did we reconstruct.

$$||\mathbf{x}_i - \mathbf{U}^\top \mathbf{z}_i||$$

Now we assume there exists some a $\mathbf{v}_{d+1}, \dots, \mathbf{v}_m$ that completes $\mathbf{u}_1, \dots, \mathbf{u}_d$, giving a m dimensional orthonormal basis.
Using change of basis we can write $\mathbf{x}$ as:

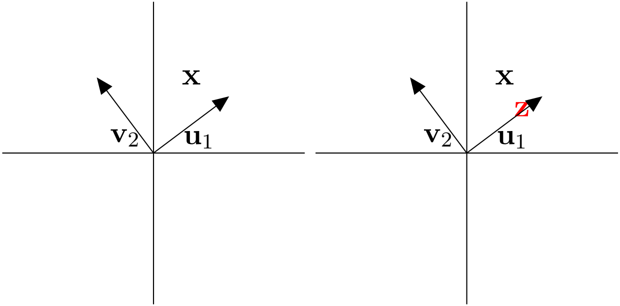
$$\langle (\mathbf{x}^\top \mathbf{u}_1), \dots, (\mathbf{x}^\top \mathbf{u}_d), (\mathbf{x}^\top \mathbf{v}_{d+1}), \dots (\mathbf{x}^\top \mathbf{v}_m) \rangle$$

Claim: With fixed $\mathbf{u}_1, \dots, \mathbf{u}_d$, best we can do is set $\mathbf{z}$ to match first d dimensions (projection onto $\mathbf{U}$):

$$\langle (\mathbf{x}^\top \mathbf{u}_1), \dots, (\mathbf{x}^\top \mathbf{u}_d), 0, \dots 0 \rangle = \mathbf{U}\mathbf{x} = \mathbf{z}$$

Simple Example

Assume $m = 2$, $d = 1$ and fixed $\mathbf{U}$ in this case $\mathbf{u}_1$, completed with $\mathbf{v}_2$.



In $\mathbb{R}^2$:

$$z = \mathbf{x}^\top \mathbf{u}_1$$

$$\mathcal{L} = (\mathbf{x}^\top \mathbf{v}_2)^\top (\mathbf{x}^\top \mathbf{v}_2)$$

- Goal: find the $\mathbf{v}$ and $\mathbf{u}$ vectors that minimize this projection loss.

Loss for one basis vector

For one basis dimension $\mathbf{v}_j$, loss is:

$$\begin{aligned} \min_{\mathbf{v}_j} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{v}_j)^\top (\mathbf{x}_i^\top \mathbf{v}_j) &= \min_{\mathbf{v}_j} \mathbf{v}_j^\top (\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top) \mathbf{v}_j \\ &= \min_{\mathbf{v}_j} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j \end{aligned}$$

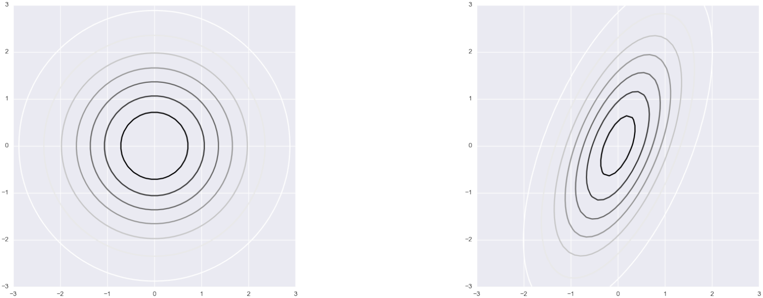
Let's name this middle term the normalized **feature covariance** matrix:

$$\mathbf{S} = (\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top) = \mathbf{X}^\top \mathbf{X}$$

Discussion: Normalized Feature Covariance Matrix

Emprical covariance between different features $\mathbb{R}^{m \times m}$.

$$\mathbf{S} = (\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top) = \mathbf{X}^\top \mathbf{X}$$



Plots of $\mathcal{N}(0, \mathbf{S})$ for independent and correlated features.

[We have seen this before, recall linear regression $(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$]

Loss for one basis vector: Minimization

$$\min_{\mathbf{v}_j,\lambda} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{v}_j)^\top (\mathbf{x}_i^\top \mathbf{v}_j) + \lambda(1 - \mathbf{v}_j^\top \mathbf{v}_j)$$

Where λ is Lagrange multiplier for normality ($\mathbf{v}_j^\top \mathbf{v}_j = 1$)

Take a partials and set to zero:

$$\frac{\partial}{\partial \mathbf{v}_j} = 2\sum_{i=1}^{n} \mathbf{x}_i(\mathbf{x}_i^\top \mathbf{v}_j) + 2\lambda \mathbf{v}_j$$

$$(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top)\mathbf{v}_j = \mathbf{S}\mathbf{v}_j = \lambda \mathbf{v}_j$$

And loss:

$$\mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j = \lambda$$

Loss for one basis vector: Minimization

$$\min_{\mathbf{v}_j,\lambda} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{v}_j)^\top (\mathbf{x}_i^\top \mathbf{v}_j) + \lambda(1 - \mathbf{v}_j^\top \mathbf{v}_j)$$

Where λ is Lagrange multiplier for normality ($\mathbf{v}_j^\top \mathbf{v}_j = 1$)

Take a partials and set to zero:

$$\frac{\partial}{\partial \mathbf{v}_j} = 2\sum_{i=1}^{n} \mathbf{x}_i(\mathbf{x}_i^\top \mathbf{v}_j) + 2\lambda \mathbf{v}_j$$

$$(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top)\mathbf{v}_j = \mathbf{S}\mathbf{v}_j = \lambda \mathbf{v}_j$$

And loss:

$$\mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j = \lambda$$

Single Dimension Loss Interpretation

Optimality condition:

$$\mathbf{S}\mathbf{v}_j = \lambda \mathbf{v}_j$$

Loss at optimum:

$$\mathbf{v}_j^\top \mathbf{S}\mathbf{v}_j = \lambda$$

1. To satisfy first condition, must be an *eigenvector* of $\mathbf{S}$.
2. To minimize second condition, want to pick *smallest* eigenvectors to make up the $\mathbf{v}$'s. Conversely, utilize the largest eigenvalues for the $\mathbf{u}_1 \dots \mathbf{u}_d$.

Exercise: Why is λ non-negative?

PCA Algorithm

1. Decide on desired dimension d
2. Compute d-largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
3. The corresponding eigenvectors $\mathbf{u}_1 \ldots \mathbf{u}_d$ make up the matrix $\mathbf{U}$
4. Perform dimensionality reduction on a new $\mathbf{x}$ by computing $\mathbf{Ux}$

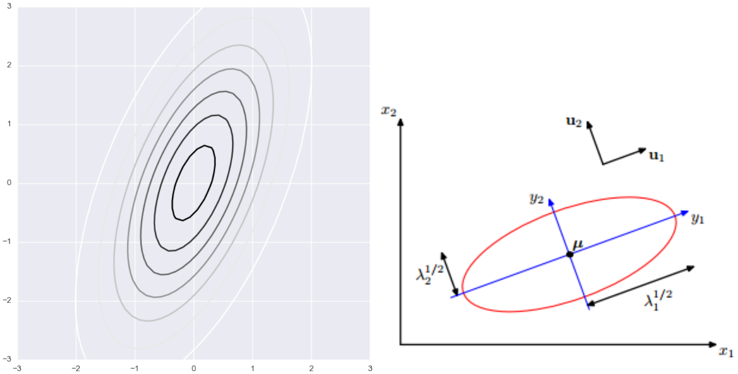
In practice there are fast randomized algorithms for computing eigenvectors.

Contents

Visual Interpretation

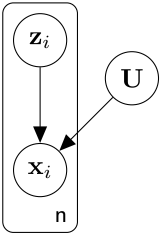
Final algorithm projects to eigenvectors with largest eigenvalues of covariance matrix.

Reconstruction penalty will be based on smaller eigenvalues



Sample covariance matrix and from Bishop relationship to eigenvalues.

Probabilistic Interpretation [Bishop]



Generative process for PCA

$$
\begin{aligned}
\mathbf{z}_i &\sim \mathcal{N}(0, \mathbf{I}) \\
\mathbf{x}_i &\sim \mathcal{N}(\mathbf{U}^\top \mathbf{z}_i, \sigma^2 \mathbf{I})
\end{aligned}
$$

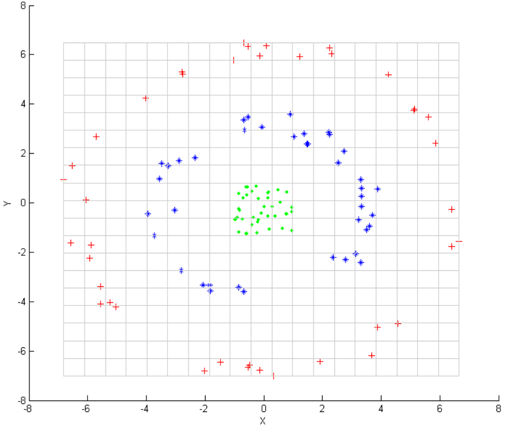
Here $\mathbf{x}_i = \mathbf{U}^\top \mathbf{z}_i + \epsilon$, interpret dimensions $d+1, \ldots, m$ as noise.

(Can even run EM on PCA, introduce priors etc.)

Example: PCA

Contents

PCA Failure Cases



Data from several sources with (roughly) spherical covariance.

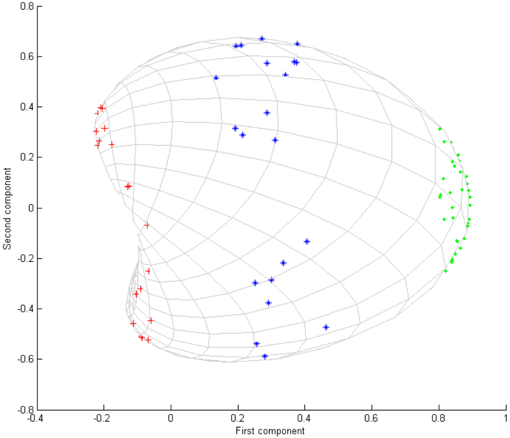
Kernel PCA

- As with supervised learning can utilize different transformation of the input.
- Similarly can apply Kernel trick to efficiently compute PCA with implicit basis ϕ

Sketch:

1. Construct kernel matrix of data $\mathbf{K}$
2. Compute eigenvalues/eigenvectors of this matrix.
3. To project new data, compute kernel function with each data point and take sum.

PCA Failure Cases



Output of Kernel PCA with Gaussian kernel on data

Encoding and Reconstruction Interpretation

Since the $\mathbf{z}$ variables are implied by projection, loss can be written as:

$$\mathcal{L}(\mathbf{U}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{U}^\top(\mathbf{U}\mathbf{x}_i)||_2^2$$

Consider the latter term:

$$\mathbf{U}^\top(\mathbf{U}\mathbf{x}_i)$$

- Use $\mathbf{U}$ to "encode" $\mathbf{x}$ at lower dimension and then "reconstruct".
- Many other, possibly non-linear, ways of doing this.

Autoencoders

We interpreted PCA as a linear encoder/reconstruct step.

$$\mathcal{L}(\mathbf{U}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{U}^\top(\mathbf{U}\mathbf{x}_i)||_2^2$$

Can also substitute with a two parameterized non-linear transformation

$$\mathcal{L}(\mathbf{U}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \boldsymbol{\phi}(\boldsymbol{\phi}(\mathbf{x}_i; \mathbf{U}^1); \mathbf{U}^2)||_2^2$$

Where $\mathbf{U}^1 \in \mathbb{R}^{m \times d}$ and $\mathbf{U}^2 \in \mathbb{R}^{d \times m}$ are neural network weights.

Similar idea as in supervised learning, adaptive dimensionality reduction.

Autoencoders in Practice

- Autoencoders are a very active area of deep learning research.
- Denoising autoencoders, variational autoencoders, autoencoders for embeddings, autoencoders for pretraining...
- Examples