

CSC 311: Introduction to Machine Learning

Lecture 8 - Principal Component Analysis

Amir-massoud Farahmand & Emad A.M. Andrews

University of Toronto

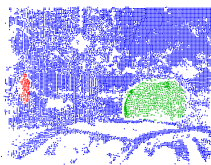
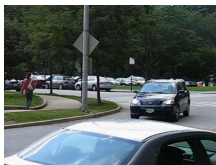
Unsupervised Learning: Motivating Examples

- Some examples of situations where you would use **unsupervised learning**:
 - ▶ You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
 - ▶ You are a biologist studying animal behaviour, so you want to infer a high-level description of their behaviour from video. You don't know the set of behaviour ahead of time.
 - ▶ You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (when refrigerator, washing machine, etc. were operating or not).
- Common theme: you have some data, and you want to infer the structure underlying the data.
- This structure is **latent**, which means it is not observed.

Motivating Examples



- Determine groups of people in image above
 - ▶ based on clothing styles
 - ▶ gender, age, etc



- Determine moving objects in videos

Overview

- Today we will cover the first unsupervised learning algorithm for this course: [Principal Component Analysis \(PCA\)](#)
- PCA is a [dimensionality reduction](#) method.
- Dimensionality reduction: Mapping the data to a lower dimensional space.
 - ▶ Saving computation and memory
 - ▶ Reducing overfitting and achieve better generalization
 - ▶ Visualizing
- PCA is a linear model. It is useful for understanding many other similar algorithms.
 - ▶ Autoencoders
 - ▶ Matrix factorizations
- We use a lot of linear algebra in today's lecture.
 - ▶ Especially orthogonal matrices and eigendecompositions.
 - ▶ Don't worry if you don't get it immediately.

Setup: Multivariate Inputs

- Setup: Given an i.i.d. dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\} \subset \mathbb{R}^D$.
- N instances/observations/examples

$$\mathbf{X} = \begin{bmatrix} [\mathbf{x}^{(1)}]^\top \\ [\mathbf{x}^{(2)}]^\top \\ \vdots \\ [\mathbf{x}^{(N)}]^\top \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_D^{(N)} \end{bmatrix}$$

- Mean

$$\mathbb{E}[\mathbf{x}^{(i)}] = \boldsymbol{\mu} = [\mu_1, \dots, \mu_D]^\top \in \mathbb{R}^D$$

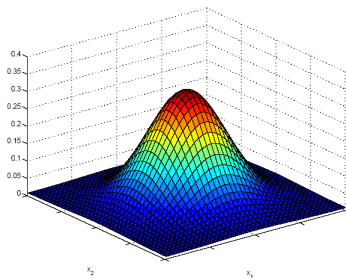
- Covariance

$$\boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}^{(i)}) = \mathbb{E}[(\mathbf{x}^{(i)} - \boldsymbol{\mu})(\mathbf{x}^{(i)} - \boldsymbol{\mu})^\top] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \sigma_D^2 \end{bmatrix}$$

Multivariate Gaussian Model

- $\mathbf{x}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, a Gaussian (or normal) distribution defined as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$



Mean and Covariance Estimators

- Observed data: $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$.
- Recall that the MLE estimators for the mean $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ under the multivariate Gaussian model is given by (previous lecture)

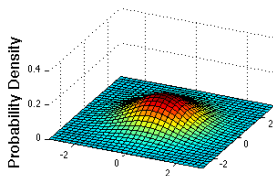
$$\text{Sample mean: } \hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)}$$

$$\text{Sample covariance: } \hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top$$

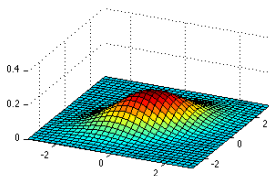
- $\hat{\boldsymbol{\mu}}$ quantifies (approximately) where your data is located in space.
- $\hat{\boldsymbol{\Sigma}}$ quantifies (approximately) how your data points are spread.

Bivariate Normal

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$

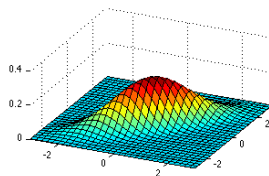


Figure: Probability density function

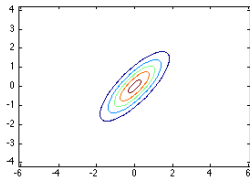
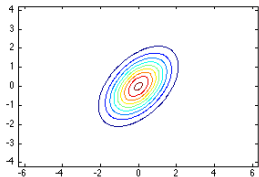
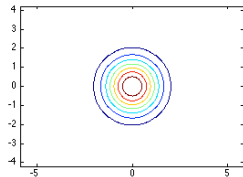
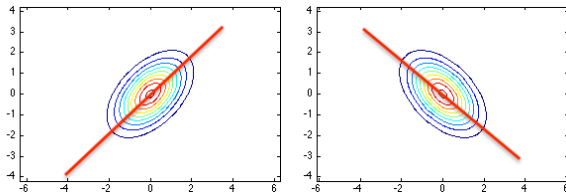


Figure: Contour plot of the pdf

Low dimensional representation

- Sometimes in practice, even though data is very high dimensional, its important features can be accurately captured in a low dimensional subspace.



- Find a low dimensional representation of data.
 - ▶ Computational benefits
 - ▶ Interpretability, visualization
 - ▶ Generalization

Low Dimensional Representation

- Sometimes in practice, even though data is very high dimensional, its important features can be accurately captured in a low dimensional subspace.

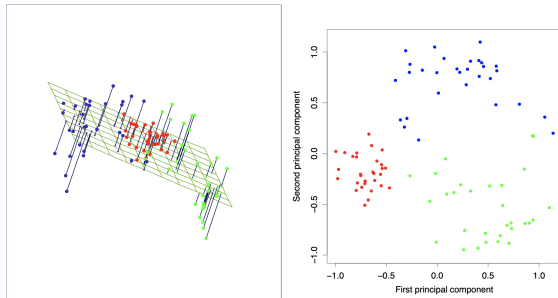
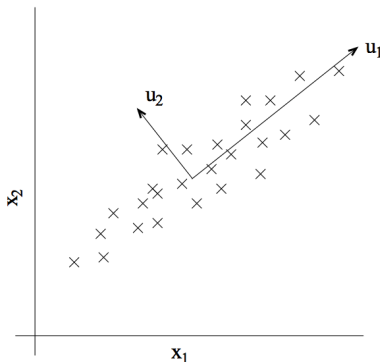


Image credit: Elements of Statistical Learning

Projection onto a Subspace

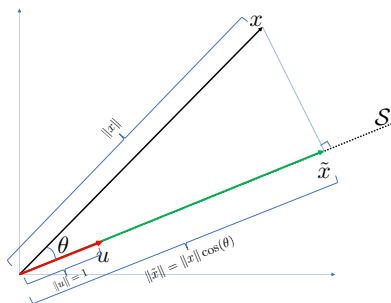
- Given: A dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\} \subset \mathbb{R}^D$.
- Set $\hat{\boldsymbol{\mu}}$ to the sample mean of the data, $\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)}$.
- Goal: Find a K -dimensional linear subspace $\mathcal{S} \subset \mathbb{R}^D$ such that $\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}$ is “well-represented” by its projection onto a K -dimensional $\mathcal{S}$.
- Recall: The [projection](#) of a point $\mathbf{x}$ onto $\mathcal{S}$ is the point in $\mathcal{S}$ closest to $\mathbf{x}$. More on this coming soon.

We are Looking for Directions



- For example, in a 2-dimensional problem, we are looking for the direction u_1 along which the data is **well represented**.
- Different interpretation of “well represented”:
 - ▶ (1) Direction of highest variance
 - ▶ (2) Direction of minimum difference after projection
- It turns out they are the same.

Euclidean Projection

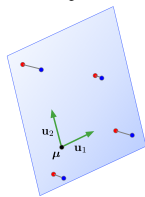


- Here, $\mathcal{S}$ is the line along the unit vector $\mathbf{u}$ (1-dimensional subspace)
 - ▶ $\mathbf{u}$ is a basis for $\mathcal{S}$: any point in $\mathcal{S}$ can be written as $z\mathbf{u}$ for some z .

- Projection of $\mathbf{x}$ on $\mathcal{S}$ is denoted by $\text{Proj}_{\mathcal{S}}(\mathbf{x})$
- Recall: $\mathbf{x}^{\top} \mathbf{u} = \|\mathbf{x}\| \|\mathbf{u}\| \cos(\theta) = \|\mathbf{x}\| \cos(\theta)$
- $\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \underbrace{\mathbf{x}^{\top} \mathbf{u}}_{\text{length of proj}} \cdot \underbrace{\mathbf{u}}_{\text{direction of proj}} = \|\tilde{\mathbf{x}}\| \mathbf{u}$

General Subspaces

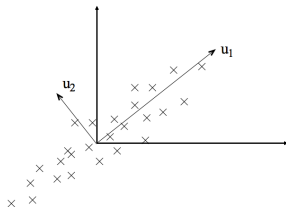
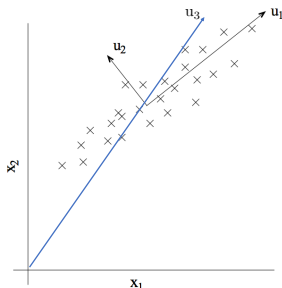
- In general, $\mathcal{S}$ is not one dimensional (i.e., line), but a (linear) subspace with a dimension K .
- In this case, we have K basis vectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K \in \mathbb{R}^D$: any vector $\mathbf{y}$ in $\mathcal{S}$ can be written as $\mathbf{y} = \sum_{i=1}^K z_i \mathbf{u}_i$ for some $z_1, \dots, z_K$.



- Projection of $\mathbf{x} \in \mathbb{R}^D$ on this subspace is given by

$$\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \sum_{i=1}^K z_i \mathbf{u}_i \quad \text{where} \quad z_i = \mathbf{x}^{\top} \mathbf{u}_i.$$

First Step: Center Data



- Directions we compute will pass through origin, and should represent the direction of highest variance.
- We need to center our data since we don't want location of data to influence our calculations. We are only interested in finding the direction of highest variance. This is independent from its mean.
- $\implies$ We are **not** interested in u_3 , we are interested in u_1 .

Projection onto a Subspace

- Let $\{\mathbf{u}_k\}_{k=1}^K$ be an **orthonormal** basis of the subspace $\mathcal{S}$ (a K -dimensional linear subspace of $\mathbb{R}^D$).
- Approximate each data point $\mathbf{x} \in \mathbb{R}^D$ as:
 1. Center (subtract the mean)
 2. Project onto $\mathcal{S}$
 3. Add the mean back

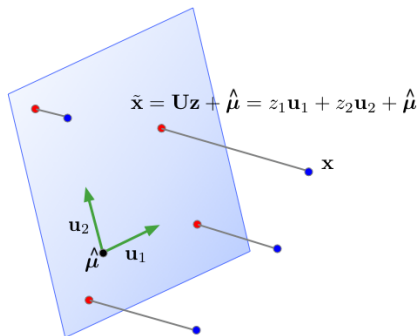
$$\begin{aligned}\tilde{\mathbf{x}} &= \hat{\boldsymbol{\mu}} + \text{Proj}_{\mathcal{S}}(\mathbf{x} - \hat{\boldsymbol{\mu}}) \\ &= \hat{\boldsymbol{\mu}} + \sum_{k=1}^K z_k \mathbf{u}_k\end{aligned}$$

- We also know: $z_k = \mathbf{u}_k^T(\mathbf{x} - \hat{\boldsymbol{\mu}})$
- Let $\mathbf{U} \in \mathbb{R}^{D \times K}$ be a matrix with columns $\{\mathbf{u}_k\}_{k=1}^K$.
- Then $\mathbf{z} = \mathbf{U}^T(\mathbf{x} - \hat{\boldsymbol{\mu}})$ (Note that $\mathbf{z} \in \mathbb{R}^K$).
- Also: $\tilde{\mathbf{x}} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{z} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{U}^T(\mathbf{x} - \hat{\boldsymbol{\mu}})$ (Note that $\tilde{\mathbf{x}} \in \mathbb{R}^D$).
- Here, $\mathbf{U}\mathbf{U}^T$ is the projector onto $\mathcal{S}$, and $\mathbf{U}^T\mathbf{U} = \mathbf{I}$.

Projection onto a Subspace

- Note that $\mathbf{x}$ and $\tilde{\mathbf{x}}$ have the same dimensionality. That is, they are both in $\mathbb{R}^D$.
- But $\tilde{\mathbf{x}}$ lives in a low dimensional subspace in $\mathbb{R}^D$.
- Its low dimensional representation is $\mathbf{z} \in \mathbb{R}^K$.

Projection onto a Subspace

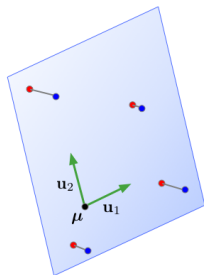


$$\mathbf{z} = \mathbf{U}^\top (\mathbf{x} - \hat{\boldsymbol{\mu}})$$

- In machine learning, $\tilde{\mathbf{x}}$ is also called the **reconstruction** of $\mathbf{x}$.
- $\mathbf{z}$ is its **representation** or **code**.

Projection onto a Subspace

- If we have a K -dimensional subspace in a D -dimensional input space, then $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{z} \in \mathbb{R}^K$.
- If the data points $\mathbf{x}$ all lie close to their reconstructions, then we can approximate distances, etc. in terms of the same operations on the code vectors $\mathbf{z}$.
- If $K \ll D$, then it is much cheaper to work with $\mathbf{z}$ than $\mathbf{x}$.
- A mapping to a space that is easier to manipulate or visualize is called a **representation**, and learning such a mapping is **representation learning**.
- Mapping data to a low-dimensional space is called **dimension reduction**.



Learning a Subspace

- How to choose a good subspace $\mathcal{S}$?
 - ▶ Need to choose $D \times K$ matrix $\mathbf{U}$ with orthonormal columns.
- Two criteria:
 - ▶ Minimize the **reconstruction error**: Find vectors in a subspace that are closest to data points.

$$\min_{\mathbf{U}} \frac{1}{N} \sum_{i=1}^N \left\| \mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)} \right\|^2$$

- ▶ Maximize the **variance of reconstructions**: Find a subspace where data has the most variability.

$$\max_{\mathbf{U}} \frac{1}{N} \sum_i \left\| \tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}} \right\|^2$$

- ▶ The data and its reconstruction has the same means (exercise)!

Learning a Subspace

- These two criteria are equivalent!

We show that

$$\frac{1}{N} \sum_{i=1}^N \left\| \mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)} \right\|^2 = \text{const} - \frac{1}{N} \sum_i \left\| \tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}} \right\|^2$$

- Recall that $\tilde{\mathbf{x}}^{(i)} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{z}^{(i)}$ and $\mathbf{z}^{(i)} = \mathbf{U}^\top (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})$.
- **Observation 1:** As $\|\mathbf{v}\|^2 = \mathbf{v}^\top \mathbf{v}$ and $\mathbf{U}^\top \mathbf{U} = I$, we have $\|\mathbf{U}\mathbf{v}\|^2 = \mathbf{v}^\top \mathbf{U}^\top \mathbf{U} \mathbf{v} = \|\mathbf{v}\|^2$. Therefore,
$$\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 = (\mathbf{U}\mathbf{z}^{(i)})^\top (\mathbf{U}\mathbf{z}^{(i)}) = [\mathbf{z}^{(i)}]^\top \mathbf{U}^\top \mathbf{U} \mathbf{z}^{(i)} = [\mathbf{z}^{(i)}]^\top \mathbf{z}^{(i)} = \|\mathbf{z}^{(i)}\|^2$$

Norm of centered reconstruction is equal to norm of representation.

Recall

For any two (compatible) matrices, $A + B$,

$$(AB)^T = B^T A^T$$

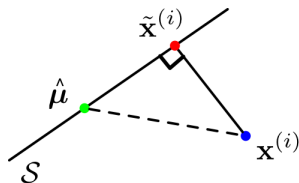
Pythagorean Theorem

- Observation 1: $\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 = \|\mathbf{z}^{(i)}\|^2$
 - ▶ Variance of reconstructions is equal to variance of code vectors:
 $\frac{1}{N} \sum_i \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 = \frac{1}{N} \sum_i \|\mathbf{z}^{(i)}\|^2$ (exercise $\frac{1}{N} \sum_i \mathbf{z}^{(i)} = 0$)

- **Observation 2:** Orthogonality of $\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}$ and $\tilde{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}$
(Two vectors $\mathbf{a}, \mathbf{b}$ are orthogonal $\iff \mathbf{a}^\top \mathbf{b} = 0$)
- Recall that $\tilde{\mathbf{x}}^{(i)} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{U}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})$.

To show the orthogonality of $\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}$ and $\tilde{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}$, observe that

$$\begin{aligned} & (\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}})^\top (\tilde{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}) \\ &= (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{U}\mathbf{U}^\top (\hat{\boldsymbol{\mu}} - \mathbf{x}^{(i)} + \mathbf{U}\mathbf{U}^\top (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})) \\ &= (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{U}\mathbf{U}^\top (\hat{\boldsymbol{\mu}} - \mathbf{x}^{(i)}) + (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{U}\mathbf{U}^\top (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}) \\ &= 0 \end{aligned}$$

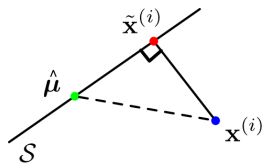


Pythagorean Theorem

Because of the orthogonality of $\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}$ and $\tilde{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}$, we can use the Pythagorean theorem to conclude that

$$\left\| \tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}} \right\|^2 + \left\| \mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)} \right\|^2 = \left\| \mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}} \right\|^2$$

By averaging over data, we obtain



$$\begin{aligned} & \underbrace{\frac{1}{N} \sum_{i=1}^N \left\| \tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}} \right\|^2}_{\text{projected variance}} + \underbrace{\frac{1}{N} \sum_{i=1}^N \left\| \mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)} \right\|^2}_{\text{reconstruction error}} \\ &= \underbrace{\frac{1}{N} \sum_{i=1}^N \left\| \mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}} \right\|^2}_{\text{constant}} \end{aligned}$$

Therefore, projected variance = constant – reconstruction error.

Maximizing the variance is equivalent to minimizing the reconstruction error!

Principal Component Analysis

Choosing a subspace to maximize the projected variance, or minimize the reconstruction error, is called **principal component analysis (PCA)**.

Recall:

- **Spectral Decomposition**: a symmetric matrix $\mathbf{A}$ has a full set of eigenvectors, which can be chosen to be orthogonal. This gives a decomposition

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top,$$

where $\mathbf{Q}$ is orthogonal and $\mathbf{\Lambda}$ is diagonal. The columns of $\mathbf{Q}$ are eigenvectors, and the diagonal entries λ_j of $\mathbf{\Lambda}$ are the corresponding eigenvalues.

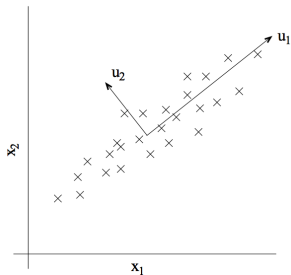
- That is, symmetric matrices are diagonal in some basis.
- A symmetric matrix $\mathbf{A}$ is positive semidefinite iff each $\lambda_j \geq 0$.
- The matrix $\mathbf{Q}$ is an orthogonal matrix, i.e., it satisfies $\mathbf{Q}^\top \mathbf{Q} = \mathbf{Q}\mathbf{Q}^\top = \mathbf{I}$.

Principal Component Analysis

- Consider the **empirical covariance matrix**:

$$\hat{\Sigma} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top$$

- Recall: Covariance matrices are symmetric and positive semidefinite.
- The optimal PCA subspace is spanned by the top K eigenvectors of $\hat{\Sigma}$.
 - More precisely, choose the first K of any orthonormal eigenbasis for $\hat{\Sigma}$.
 - The general case is tricky, but we will show this for $K = 1$.
- These eigenvectors are called **principal components**, analogous to the principal axes of an ellipse.



Deriving PCA

- For $K = 1$, we are fitting a unit vector $\mathbf{u}$, and the code is a scalar $z^{(i)} = \mathbf{u}^\top (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})$. Let's maximize the projected variance. From **Observation 1**, we have

$$\begin{aligned}\frac{1}{N} \sum_i \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 &= \frac{1}{N} \sum_i |z^{(i)}|^2 = \frac{1}{N} \sum_i (\mathbf{u}^\top (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}))^2 \\&= \frac{1}{N} \sum_{i=1}^N \mathbf{u}^\top (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}) (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{u} && (\mathbf{a}^\top \mathbf{b})^2 = \mathbf{a}^\top \mathbf{b} \mathbf{b}^\top \mathbf{a} \\&= \mathbf{u}^\top \left[\frac{1}{N} \sum_{i=1}^N (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}) (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \right] \mathbf{u} \\&= \mathbf{u}^\top \hat{\boldsymbol{\Sigma}} \mathbf{u} \\&= \mathbf{u}^\top \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top \mathbf{u} && \text{Spectral Decomposition } \hat{\boldsymbol{\Sigma}} = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top \\&= \mathbf{a}^\top \boldsymbol{\Lambda} \mathbf{a} && \text{for } \mathbf{a} = \mathbf{Q}^\top \mathbf{u} \\&= \sum_{j=1}^D \lambda_j a_j^2\end{aligned}$$

Deriving PCA

- So in order to maximize $\frac{1}{N} \sum_i \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2$, we can maximize $\mathbf{a}^\top \boldsymbol{\Lambda} \mathbf{a} = \sum_{j=1}^D \lambda_j a_j^2$ for $\mathbf{a} = \mathbf{Q}^\top \mathbf{u}$ instead.
 - ▶ This is a change-of-basis to the eigenbasis of $\boldsymbol{\Sigma}$.
- Assume the λ_i are in sorted order, $\lambda_1 \geq \lambda_2 \geq \dots$
- Observation: since $\mathbf{u}$ is a unit vector, then by unitarity, $\mathbf{a}$ is also a unit vector: $\mathbf{a}^\top \mathbf{a} = \mathbf{u}^\top \mathbf{Q} \mathbf{Q}^\top \mathbf{u} = \mathbf{u}^\top \mathbf{u}$, i.e., $\sum_j a_j^2 = 1$.
- To maximize $\sum_{j=1}^D \lambda_j a_j^2$, under the constraint that $\|\mathbf{a}\|_2 = 1$, we can set $a_1 = \pm 1$ and $a_j = 0$ for $j \neq 1$.
- Hence, $\mathbf{u} = \mathbf{Q} \mathbf{a} = \mathbf{q}_1$ (the top eigenvector).
- A similar argument shows that the k th principal component is the k th eigenvector of $\boldsymbol{\Sigma}$. If you're interested, look up the [Courant-Fischer Theorem](#).

Decorrelation

- Interesting fact: the dimensions of $\mathbf{z}$ are decorrelated. For now, let $\mathbf{Cov}(\cdot)$ denote the empirical covariance.

$$\begin{aligned}\mathbf{Cov}(\mathbf{z}) &= \mathbf{Cov}\left(\mathbf{U}^\top(\mathbf{x} - \boldsymbol{\mu})\right) \\ &= \mathbf{U}^\top \mathbf{Cov}(\mathbf{x}) \mathbf{U} \\ &= \mathbf{U}^\top \boldsymbol{\Sigma} \mathbf{U} \\ &= \mathbf{U}^\top \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top \mathbf{U} \\ &= \begin{pmatrix} \mathbf{I} & \mathbf{0} \end{pmatrix} \boldsymbol{\Lambda} \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} && \text{by orthogonality} \\ &= \text{top left } K \times K \text{ block of } \boldsymbol{\Lambda}\end{aligned}$$

- If the covariance matrix is diagonal, this means the features are uncorrelated.

Recap:

- Dimensionality reduction aims to find a low-dimensional representation of the data.
- PCA projects the data onto a subspace which maximizes the projected variance, or equivalently, minimizes the reconstruction error.
- The optimal subspace is given by the top eigenvectors of the empirical covariance matrix.
- PCA gives a set of decorrelated features.

Applying PCA to faces

- Consider running PCA on 2429 19x19 grayscale images (CBCL data)
- Can get good reconstructions with only 3 components



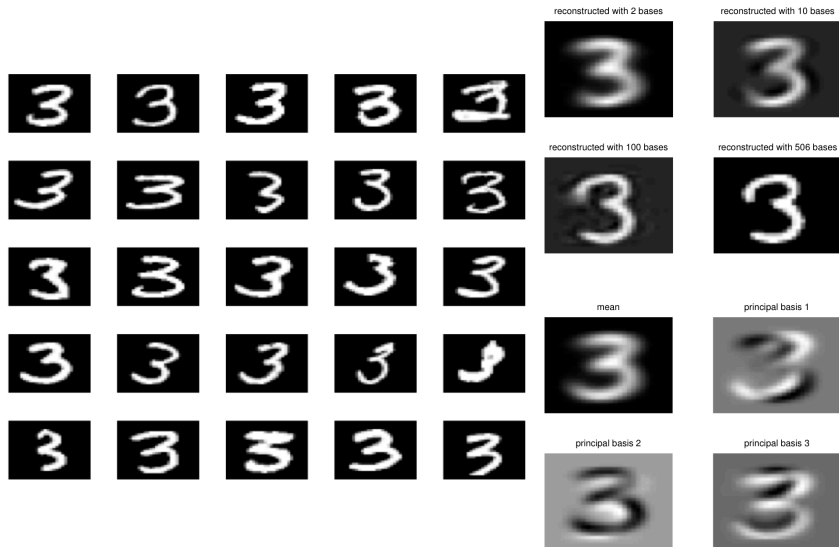
- PCA for pre-processing: can apply classifier to low-dimensional representation
 - ▶ Original data is 361 dimensional
 - ▶ For face recognition PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for a Gaussian mixture model (GMM) with 84 states.
- Can also be good for visualization

Applying PCA to faces: Learned basis

Principal components of face images (“eigenfaces”)



Applying PCA to digits



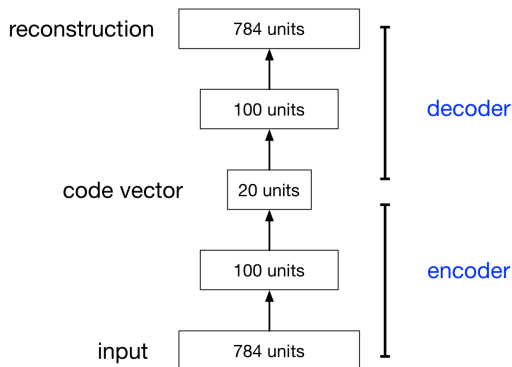
Next

Next: two more interpretations of PCA, which have interesting generalizations.

1. Autoencoders
2. ~~Matrix factorization (a tutorial)~~

Autoencoders

- An **autoencoder** is a feed-forward neural net whose job is to take an input $\mathbf{x}$ and predict $\mathbf{x}$.
- To make this non-trivial, we need to add a **bottleneck layer** whose dimension is much smaller than the input.



Why autoencoders?

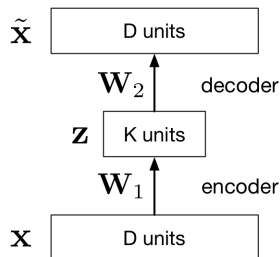
- Map high-dimensional data to two dimensions for visualization
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
 - ▶ Unlabeled data can be much more plentiful than labeled data

Linear Autoencoders

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

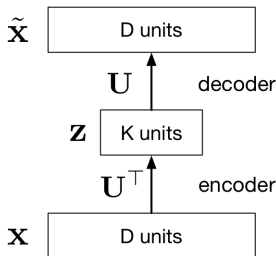
$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

- This network computes $\tilde{\mathbf{x}} = \mathbf{W}_2 \mathbf{W}_1 \mathbf{x}$, which is a linear function.
- If $K \geq D$, we can choose $\mathbf{W}_2$ and $\mathbf{W}_1$ such that $\mathbf{W}_2 \mathbf{W}_1$ is the identity matrix. This isn't very interesting.
- But suppose $K < D$:
 - ▶ $\mathbf{W}_1$ maps $\mathbf{x}$ to a K -dimensional space, so it is doing a dimensionality reduction.



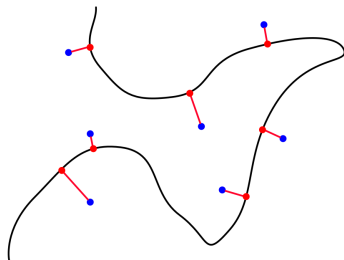
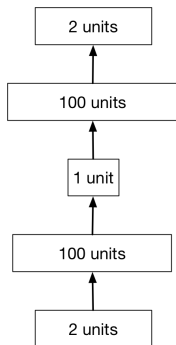
Linear Autoencoders

- Observe that the output of the autoencoder must lie in a K -dimensional subspace spanned by the columns of $\mathbf{W}_2$. This is because $\tilde{\mathbf{x}} = \mathbf{W}_2 \mathbf{z}$
- We saw that the best possible (min error) K -dimensional linear subspace in terms of reconstruction error is the PCA subspace.
- The autoencoder can achieve this by setting $\mathbf{W}_1 = \mathbf{U}^\top$ and $\mathbf{W}_2 = \mathbf{U}$.
- Therefore, the optimal weights for a linear autoencoder are just the principal components.



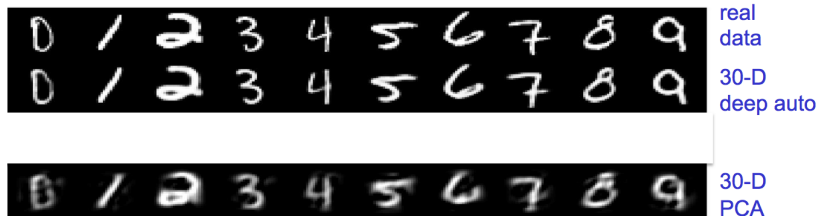
Nonlinear Autoencoders

- Deep nonlinear autoencoders learn to project the data, not onto a linear subspace, but onto a nonlinear **manifold**
- This is a **nonlinear dimensionality reduction**.



Nonlinear Autoencoders

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)



Nonlinear Autoencoders

Here's a 2-dimensional autoencoder representation of newsgroup articles. They're color-coded by topic, but the algorithm wasn't given the labels.

