Dimensionality Reduction
Lecture 23

David Sontag
New York University

Slides adapted from Carlos Guestrin and Luke Zettlemoyer

Assignments

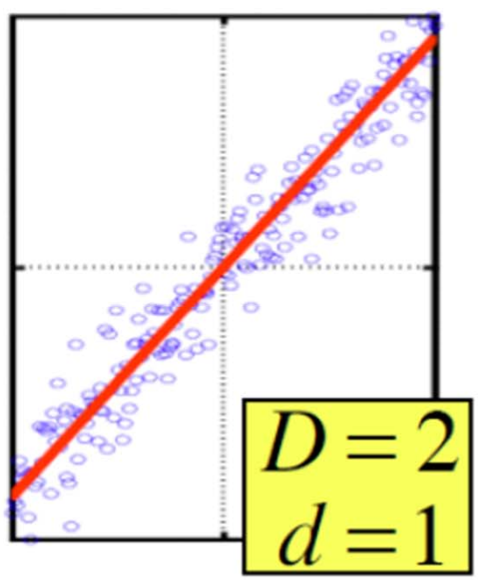
- Last homework assignment released tonight, due next Thursday (Dec. 5)
- Final project write-up due December 15
- 10 minute presentations (1 per group)
  - Part of your grade
  - During final exam period, Dec. 17, 10-11:50am
- I need 4 groups to volunteer to give their presentation on Dec. 12

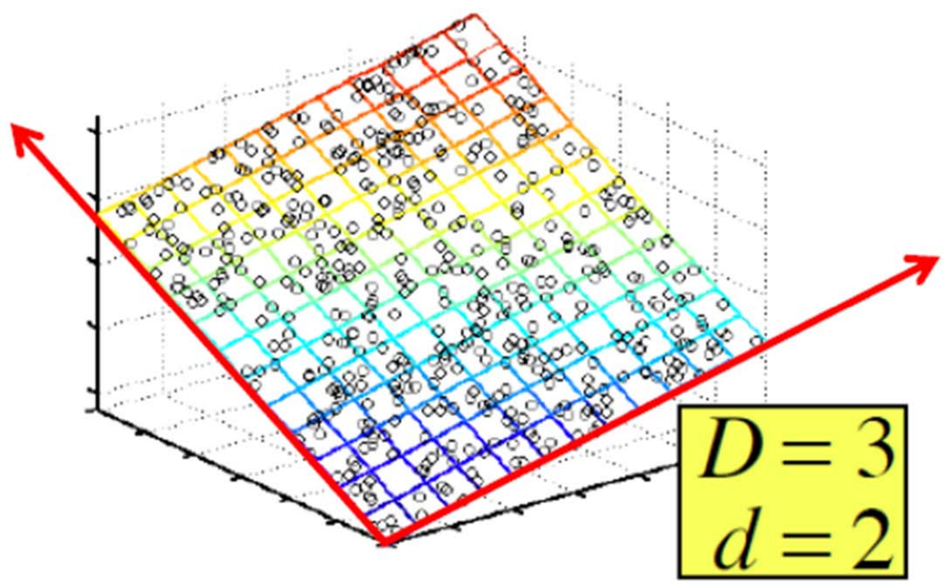
Dimensionality reduction

- Input data may have thousands or millions of dimensions!
  - e.g., text data has ???, images have ???
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – show high dimensional data in 2D
  - discover "intrinsic dimensionality" of data
    - high dimensional data that is truly lower dimensional
    - noise reduction

Dimension reduction

- Assumption: data (approximately) lies on a lower dimensional space
- Examples:





Slide from Yi Zhang

Lower dimensional projections

- Rather than picking a subset of the features, we can obtain new ones by combining existing features x_1 … x_n

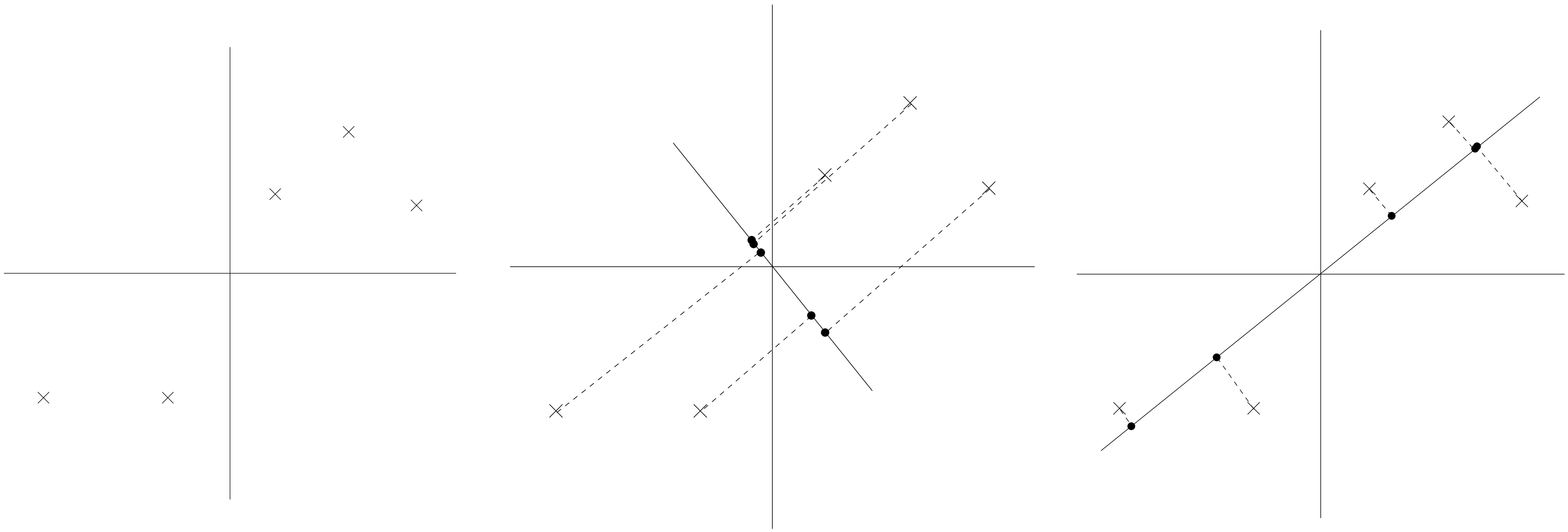
$$z_1 = w_0^{(1)} + \sum_i w_i^{(1)} x_i$$

$$\cdots$$

$$z_k = w_0^{(k)} + \sum_i w_i^{(k)} x_i$$

- New features are linear combinations of old ones
- Reduces dimension when k<n
- Let’s consider how to do this in the unsupervised setting
  - just **X**, but no Y

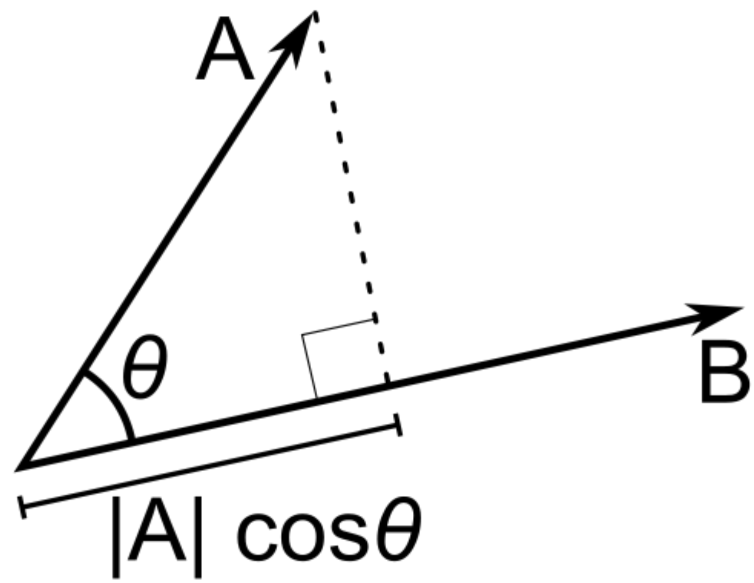
Which projection is better?



From notes by Andrew Ng

Reminder: Vector Projections

- Basic definitions:
  - $A.B = |A||B| \cos \theta$
  - $\cos \theta = |adj|/|hyp|$



- Assume $|B|=1$ (unit vector)
  - $A.B = |A| \cos \theta$
  - So, dot product is length of projection!!!

Using a new basis for the data

- Project a point into a (lower dimensional) space:
  - **point**: $\mathbf{x} = (x_1,\ldots,x_n)$
  - **select a basis** – set of unit (length 1) basis vectors $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$
    - we consider orthonormal basis:
      - $\mathbf{u}_i \bullet \mathbf{u}_i = 1$, and $\mathbf{u}_i \bullet \mathbf{u}_j = 0$ for $i \neq j$
  - **select a center** – $\bar{\mathbf{x}}$, defines offset of space
  - **best coordinates** in lower dimensional space defined by dot-products: $(z_1,\ldots,z_k)$, $z_j = (\mathbf{x}-\bar{\mathbf{x}}) \bullet \mathbf{u}_j$

$$\hat{\mathrm{x}}^i = \bar{\mathrm{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

Maximize variance of projection

Let $x^{(i)}$ be the i^{th} data point minus the mean.

Choose unit-length u to maximize:

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}({x^{(i)}}^T u)^2 &= \frac{1}{m}\sum_{i=1}^{m} u^T x^{(i)} {x^{(i)}}^T u \\ &= u^T \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T\right) u.\end{aligned}$$

Let ||u||=1 and maximize. Using the method of Lagrange multipliers, can show that the solution is given by the principal eigenvector of the covariance matrix! **(shown on board)**

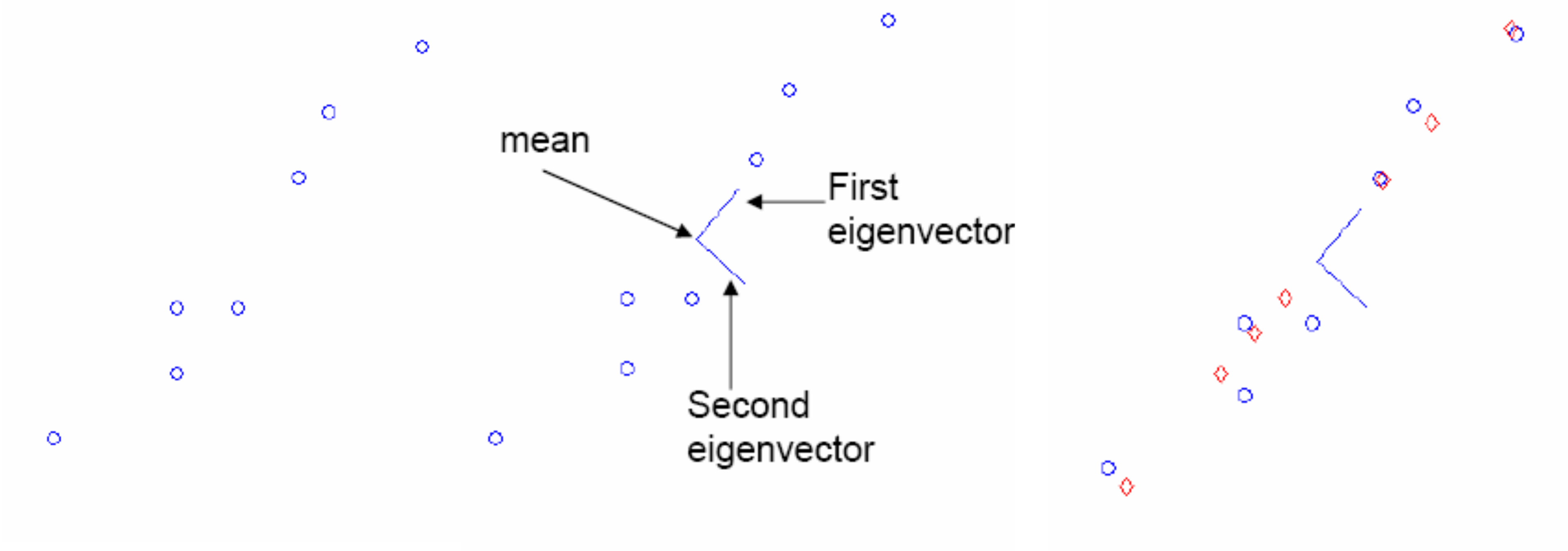
Basic PCA algorithm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- **Compute covariance** matrix:
  - $\Sigma \leftarrow 1/m\ \mathbf{X_c}^{\mathbf{T}}\ \mathbf{X_c}$
- Find **eigen vectors and values** of Σ
- **Principal components:** k eigen vectors with highest eigen values

PCA example

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

Data: Projection: Reconstruction:

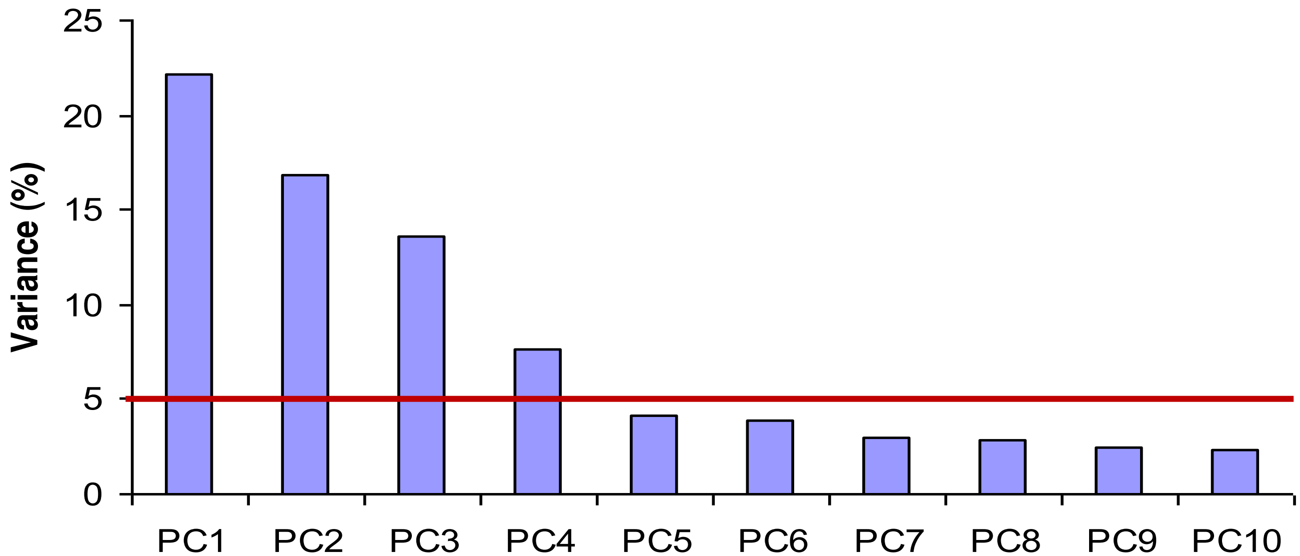


Dimensionality reduction with PCA

In high-dimensional problem, data usually lies near a linear subspace, as noise introduces small variability

Only keep data projections onto principal components with **large** eigenvalues

Can *ignore* the components of lesser significance.



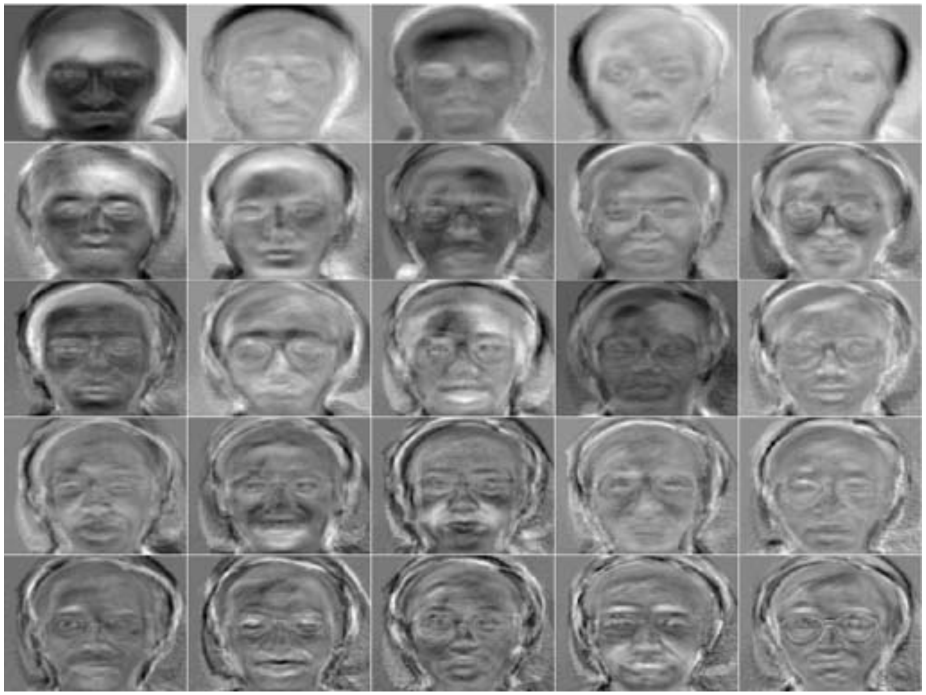
You might lose some information, but if the eigenvalues are small, you don't lose much

Slide from Aarti Singh

Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



Eigenfaces reconstruction

- Each image corresponds to adding together (weighted versions of) the principal components:

Scaling up

- Covariance matrix can be really big!
  - Σ is n by n
  - 10000 features can be common!
  - finding eigenvectors is very slow…

- Use singular value decomposition (SVD)
  - Finds k eigenvectors
  - great implementations available, e.g., Matlab svd

SVD

- Write $\mathbf{X} = \mathbf{W}\ \mathbf{S}\ \mathbf{V}^T$
  - $\mathbf{X} \leftarrow$ data matrix, one row per datapoint
  - $\mathbf{W} \leftarrow$ weight matrix, one row per datapoint – coordinate of $\mathbf{x}^i$ in eigenspace
  - $\mathbf{S} \leftarrow$ singular value matrix, diagonal matrix
    - in our setting each entry is eigenvalue λ_j
  - $\mathbf{V}^T \leftarrow$ singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algorithm

- Start from m by n data matrix $\mathbf{X}$
- **Recenter**: subtract mean from each row of $\mathbf{X}$
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- **Call SVD** algorithm on $\mathbf{X_c}$ – ask for k singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of $\mathbf{V}^T$)
  - **Coefficients:** project each point onto the new vectors