Dimensionality reduction

CISC 5800

Professor Daniel Leeds

Opening note on dimensional differences

Each dimension corresponds to a feature/measurement

Magnitude differences for each measurement (e.g., animals):

- x_1 – speed (mph) 0-100
- x_2 – weight (pounds) 10-1000
- x_3 – size (feet) 2-20

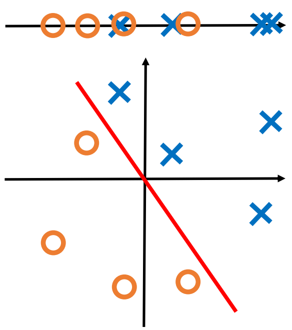
Problem for learning:

$$w_j \leftarrow w_j + \varepsilon x_j^i\left(y^i - g(w^T x^i)\right) - \frac{w_j}{\lambda}$$

Normalize: $r_1 = \frac{x_1-\mu_1}{\sigma_1}$ or $r_1 = \frac{x_1-\mu_1}{max_1-min_1}$

2

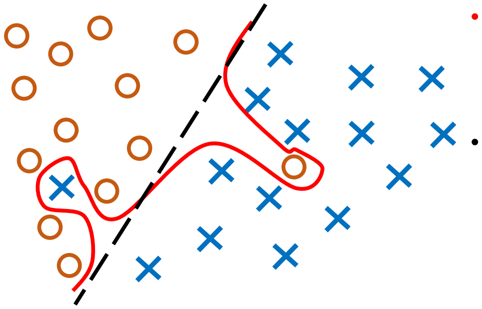
The benefits of extra dimensions



- **Finds existing complex separations between classes**

3

The risks of too-many dimensions



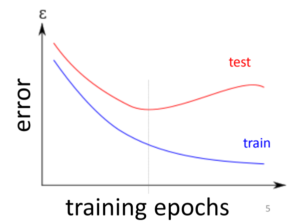
- **High dimensions with kernels over-fit the outlier data**
- Two dimensions ignore the outlier data

4

Training vs. testing

- **Training**: learn parameters from set of data in each class
- **Testing**: measure how often classifier correctly identifies new data

- More training reduces classifier error ε
  - More gradient ascent steps
  - More learned feature
- Too much training causes worse testing error – overfitting



5

Goal: High Performance, Few Parameters

- "Information criterion": performance/parameter trade-off
- Variables to consider:
  - **L** likelihood of train data after learning
  - **k** number of parameters (e.g., number of features)
  - **m** number of points of training data
- Popular information criteria:
  - Akaike information criterion **AIC**: log(L) - k
  - Bayesian information criterion **BIC**: log(L) - 0.5 k log(m)

6

Decreasing parameters

- Force parameter values to 0
  - L1 regularization
  - Support Vector selection
  - Feature selection/removal
- Consolidate feature space
  - Component analysis

8

Feature removal

- Start with feature set: F={x_1, …, x_k}
- Find classifier performance with set F: perform(F)
- Loop
  - Find classifier performance for removing feature x_1, x_2, …, x_k: argmax_i perform(F-x_i)
  - Remove feature that causes least decrease in performance: F=F-x_i

AIC: log(L) - k

BIC: log(L) - 0.5 k log(m)

Repeat, using AIC or BIC as termination criterion

9

AIC testing: log(L)-k

Features	k (num features)	L (likelihood)	AIC
F	40	0.1	-42.3
F-$\{x_3\}$	39	0.03	-41.5
F-$\{x_3,x_{24}\}$	38	0.005	-41.3
F-$\{x_3,x_{24},x_{32}\}$	37	0.001	-40.9
F-$\{x_3,x_{24},x_{32},x_{15}\}$	36	0.0001	-41.2

10

Feature selection

AIC: log(L) - k

BIC: log(L) - 0.5 k log(m)

- Find classifier performance for just set of 1 feature: argmax$_i$ perform($\{x_i\}$)
- Add feature with highest performance: F=$\{x_i\}$
- Loop
  - Find classifier performance for adding one new feature: argmax$_i$ perform(F+$\{x_i\}$)
  - Add to F feature with highest performance increase: F=F+$\{x_i\}$

Repeat, using AIC or BIC as termination criterion

11

Capturing links between features

	Document1	Document2	Document3
Wolf	12	4	1
Lion	16	3	2
Monkey	5	11	4
Sky	7	3	14
Tree	2	8	5
Cloud	6	2	12
⋮	⋮	⋮	⋮

With large number of features, some features x_j and x_k act similarly

x_{wolf} & x_{lion} -> $u_{predator}$

x_{sky} & x_{cloud} -> $u_{atmosphere}$

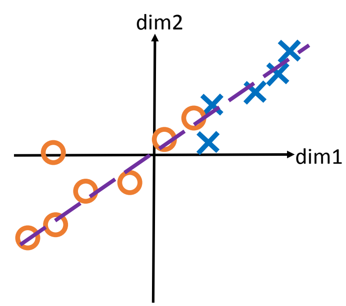
Approximate $\boldsymbol{x}^1 = \begin{bmatrix} x_1^1 \\ \vdots \\ x_N^1 \end{bmatrix}$

with $\boldsymbol{u}^1 = \begin{bmatrix} u_1^1 \\ \vdots \\ u_{N'}^1 \end{bmatrix}$

Automatically learn summary features

12

Defining new feature axes



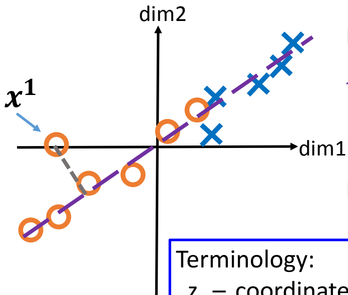
- Identify a common trend

$$\boldsymbol{u}^1 = \begin{bmatrix} 0.91 \\ 0.45 \end{bmatrix}$$

- Map data onto new dimension $\boldsymbol{u}^1$

13

Defining new feature axes



Project $\mathbf{x}^1$ onto $\mathbf{u}^1$

$$u^1 = \begin{bmatrix} 0.91 \\ 0.45 \end{bmatrix} \quad x^1 = \begin{bmatrix} -1.19 \\ 0.01 \end{bmatrix}$$

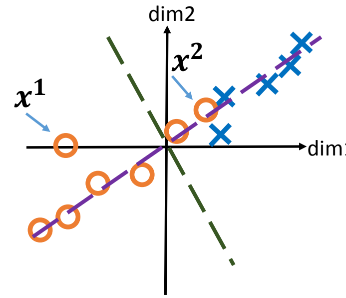
For x^1: dim1: **-1.2**

dim2: **0**

$dimU_1$: $\mathbf{u}^{1T}\mathbf{x}^1$ =
-1.19x.9+.01x.45 =
-1.06

Terminology:
z_q – coordinate on axis $\mathbf{u}_q$

Defining data points with new axes



$$x^1 = \overset{z_1}{-1} \times u^1 + \overset{z_2}{(-0.5)} \times u^2$$

$$x^2 = \mathbf{0.5} \times u^1 + (-0.02) \times u^2$$

Component analysis

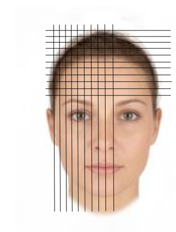
Each data point $\boldsymbol{x}^i$ in D can be reconstructed as sum of components $\boldsymbol{u}$:

- $\boldsymbol{x}^i = \sum_{q=1}^{T} z_q^i \boldsymbol{u}^q$
- z_q^i is weight on q[th] component to reconstruct data point $\mathbf{x}^i$

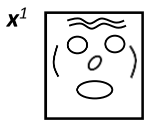
Image features

Image as grid of n x m pixels

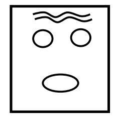
Find representative component features as pixel patterns

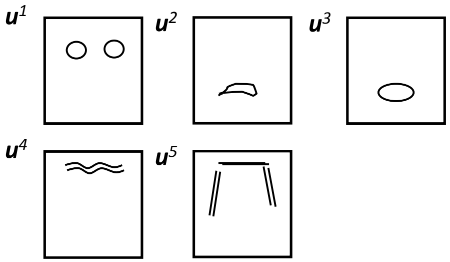


Cartoon face example:



$\approx 1 \times u^1 + 0 \times u^2 + 1 \times u^3 + 1 \times u^4 + 0 \times u^5$





Add relevant face components
1000-pixel image becomes
6 co-efficients

Estimate is fairly close to
actual image

Component analysis

Each data point $\boldsymbol{x}^i$ in D can be reconstructed as sum of components $\boldsymbol{u}$:

- $\boldsymbol{x^i} = \sum_{q=1}^{T} z_q^i \boldsymbol{u}^q$
- z_q^i is weight on qth component to reconstruct data point $\mathbf{x}^i$

Evaluating components

Components learned in order of descriptive power

Compute reconstruction error for all data by using first r components:

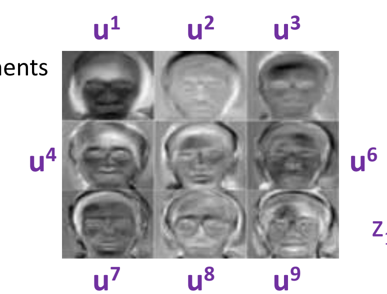
$$error = \sum_i \left(\sum_j \left(\boldsymbol{x}_j^i - \sum_{q=1}^{r} z_q^i \boldsymbol{u}_j^q \right)^2 \right)$$

Component analysis: examples

$$\boldsymbol{x^i} = \sum_{q=1}^{T} z_q^i \boldsymbol{u}^q$$

"Eigenfaces" – learned from set of face images

u: nine components



$\mathbf{x}^4$: data reconstructed

$z_1\mathbf{u}^1 + \ldots + z_9\mathbf{u}^9 \approx$

Types of component analysis

Capture links between features as “components”

- Principal component analysis (PCA)
- Independent component analysis (ICA)
- Non-negative matrix factorization (NMF)

Principal component analysis (PCA)

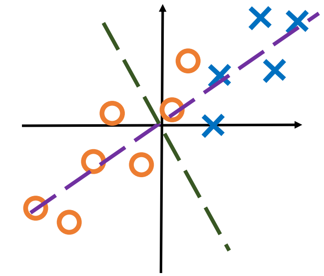
Describe every $\mathbf{x}^i$ with small set of components $\mathbf{u}^{1:Q}$

Use same $\mathbf{u}^1, \ldots \mathbf{u}^T$ for all $\mathbf{x}^i$

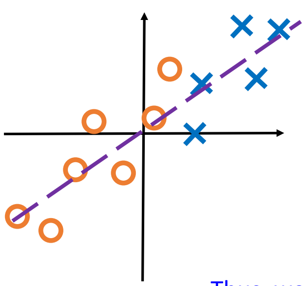
All components orthogonal:

$$(\boldsymbol{u}^i)^T \boldsymbol{u}^j = 0 \quad \forall i \neq j$$

$$\boldsymbol{x}^i = \sum_{q=1}^{T} z_q^i \boldsymbol{u}^q$$



Idea of learning in PCA



1. D = {$\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n$} , data 0-center
2. Component index: q=1
3. Loop

- Find direction of highest variance: $\mathbf{u}^q$
  - Ensure $|\boldsymbol{u}^q| = 1$
- Remove $\mathbf{u}_q$ from data:

  $$D = \{\boldsymbol{x}^1 - z_q^1 \boldsymbol{u}^q, \cdots, \boldsymbol{x}^n - z_q^n \boldsymbol{u}^q\}$$

$$(\boldsymbol{u}_i)^T \boldsymbol{u}_j = 0 \quad \forall i \neq j$$

Thus, we guarantee $z_j^i = \boldsymbol{u}_j^T \boldsymbol{x}^i$

Independent component analysis (ICA)

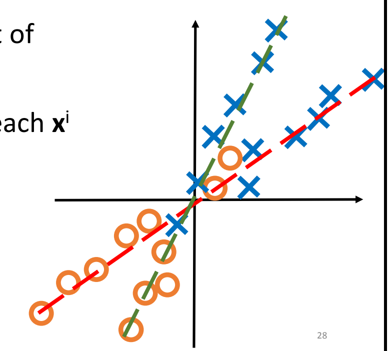
Describe every $\mathbf{x}^i$ with small set of components $\mathbf{u}^{1:T}$

Can use **different** $\mathbf{u}^1, \ldots \mathbf{u}^Q$ for each $\mathbf{x}^i$

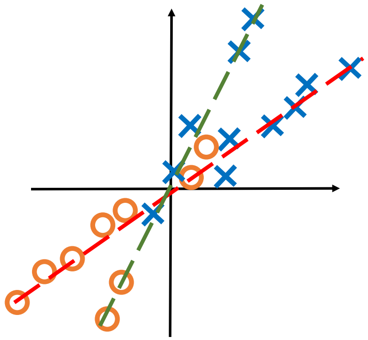
No orthogonality constraint:

$$(\boldsymbol{u}^i)^T \boldsymbol{u}^j \neq 0 \quad \forall i \neq j$$

$$\boldsymbol{x}^i = \sum_{q=1}^{T} z_q^i \boldsymbol{u}^q$$



Idea of learning ICA



1. D = {$\boldsymbol{x}^1$,…,$\boldsymbol{x}^n$} , data 0-center
2. Component index: q=1
3. Loop
- Find next most common group across data points
- Find component direct for group $\boldsymbol{u}^q$
  - Ensure $|\boldsymbol{u}^q| = 1$

We cannot guarantee $z_j^i = \boldsymbol{u}_j^T \boldsymbol{x}^i$

29

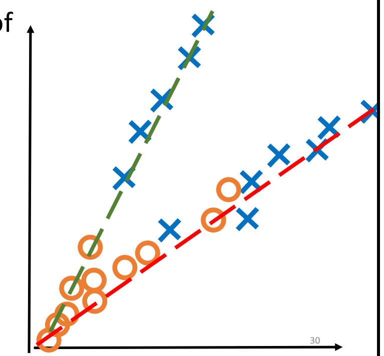
Non-negative matrix factorization (NMF)

Describe every **x**^{i} with small set of components **u**$^{1:T}$

All components and weights non-negative

$\boldsymbol{u}^i \geq 0,\ z_q^i \geq 0 \quad \forall i, q$

$$\boldsymbol{x}^i = \sum_{q=1}^{T} z_q^i \boldsymbol{u}^q$$



30

Comparing component analysis

PCA
- Represent each data point **x** with low number of components
- All components used to reconstruct each data point **x**

ICA / NMF
- Represent each data point **x** with low number of components
- Subset of components used to reconstruct each data point **x**

33