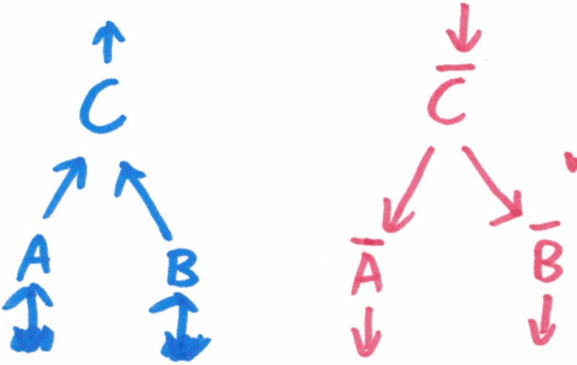
Reverse mode differentiation

Piece of computation:



Backpropagate derivatives wrt final scalar output

For any Z,
$$\bar{Z}_{ij} = \frac{\partial \text{output}}{\partial Z_{ij}}$$

Use standard rules:

$$C = \cos A \Rightarrow \bar{A} = \bar{C} \odot \sin A$$

$$C = AB \Rightarrow \bar{A} = \bar{C}B^\top, \quad \bar{B} = A^\top \bar{C}$$

$$C = A + B \Rightarrow \bar{A} = \bar{C}, \quad \bar{B} = \bar{C}$$

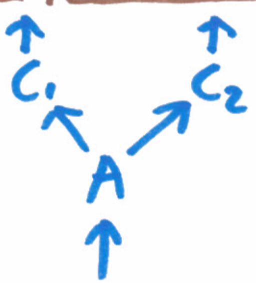
$$C = A^\top \Rightarrow \bar{A} = \bar{C}^\top$$

...

Stored in forward pass

Passed in by backprop

Multiple children



$$\bar{A} = \bar{A}_1 + \bar{A}_2$$

Apply rules separately for children and add

Matrix multiplication

$$\underset{L\times N}{C} = \underset{L\times M}{A}\;\underset{M\times N}{B} \qquad O(LMN)$$

$$\underbrace{C_{\ell n}} = \sum_m A_{\ell m} B_{mn}$$

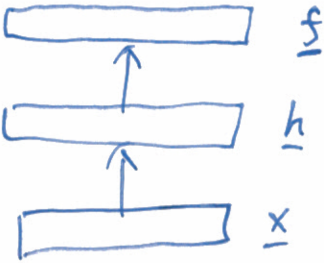
LN terms ↖ cost O(M) each

Square matrix-matrix multiply $O(N^3)$

There are $O(N^2)$ numbers in the matrices

[More in tutorial 5 (but not much)]

Autoencoder



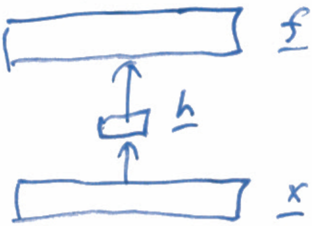
Learning task

$$f(\underline{x}) \approx \underline{x}$$

not useful

def autoencode(x):
  ~~return x~~
  h = np.dot(I, x)
  f = np.dot(I, h)
  return f

Dimensionality Reduction



K units $\quad K \ll D$

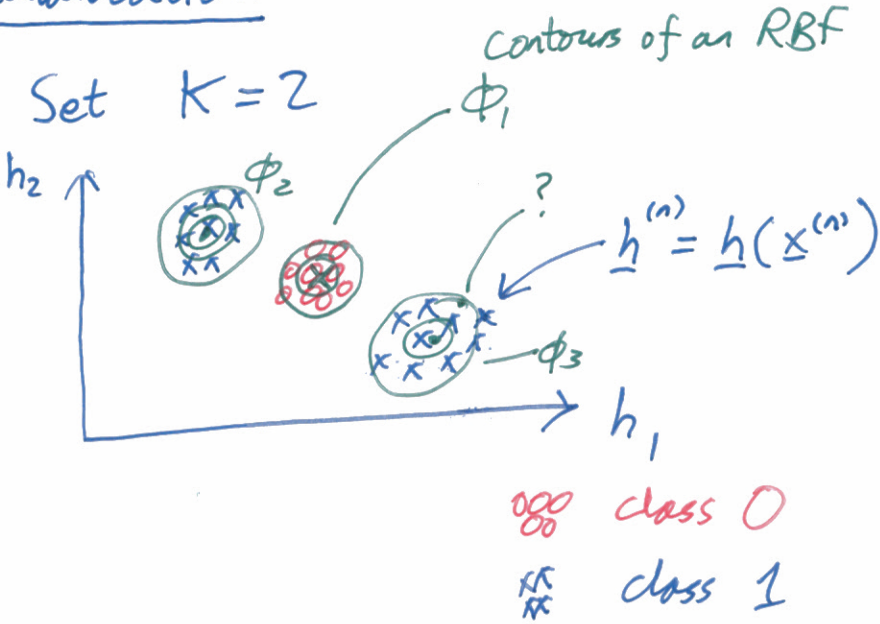
D inputs

$$\underline{h} = g^{(1)}(W^{(1)}\underline{x} + \underline{b}^{(1)})$$
$$\underline{f} = g^{(2)}(W^{(2)}\underline{h} + \underline{b}^{(2)})$$

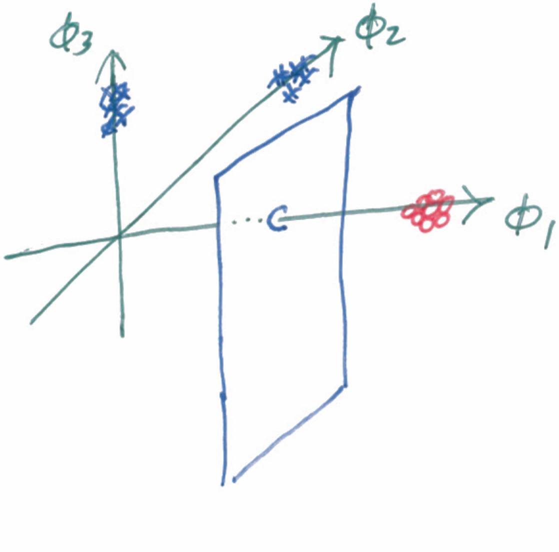
$\Rightarrow$ Use $\underline{h}$ as inputs to other ML methods

Visualization

Set $K=2$



We might want to increase dim. of data...



Sparse Autoencoder

Encourage most elements of $\underline{h}$ to be close to zero — sparse

Denoising Autoencoder

While training we mask out some of the inputs — set some x_d to zero

$\underline{m}$ mask vector, of random 0's & 1's

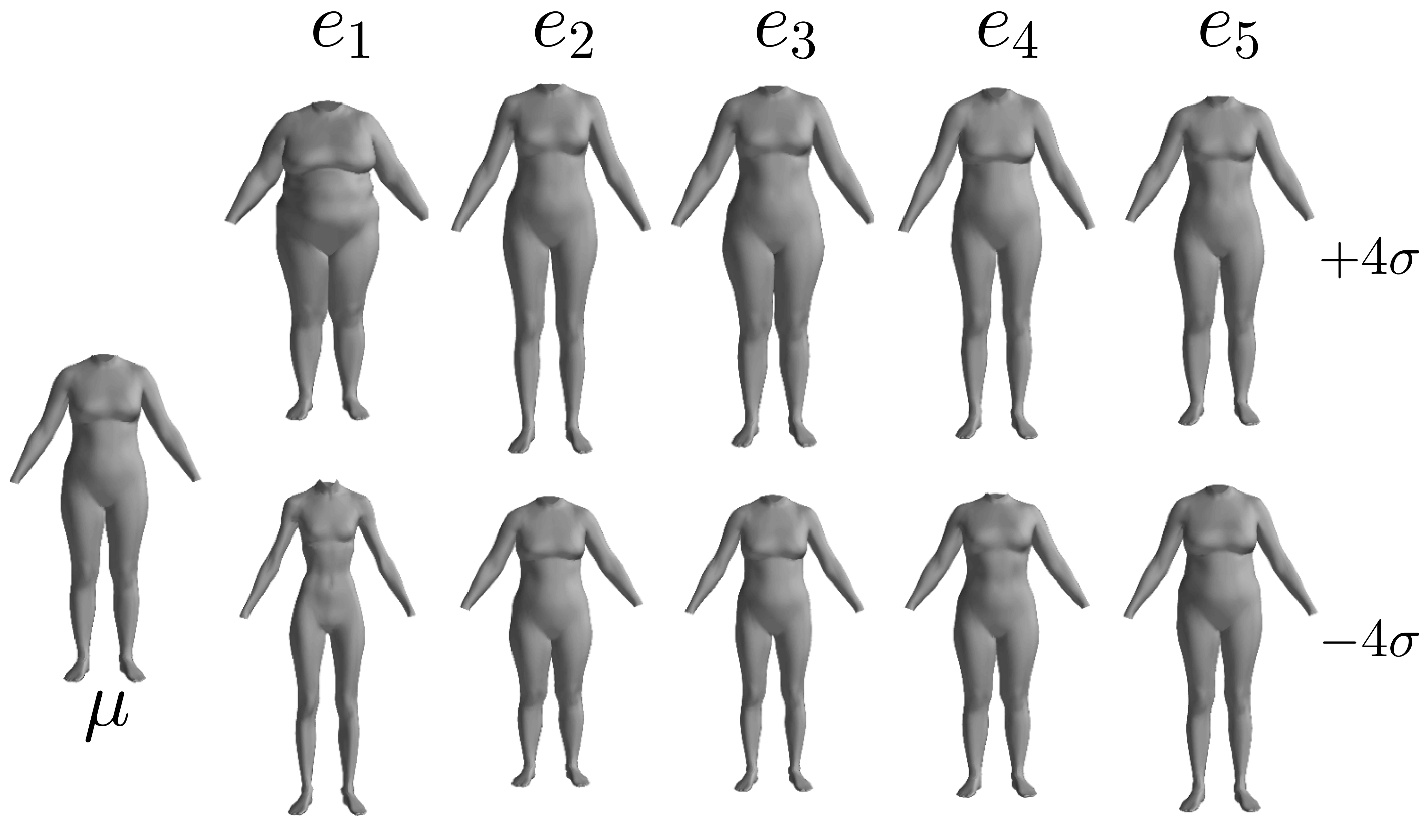
Cost on one example $\quad \| \underline{f}(\underline{x}^{(n)} \odot \underline{m}) - \underline{x}^{(n)} \|^2$

Cost function $\displaystyle \sum_{\underline{m}} p(\underline{m}) \frac{1}{N} \sum_{n=1}^{N} \| f(\underline{x}^{(n)} \odot \underline{m}) - \underline{x}^{(n)} \|^2$

Monte Carlo $\approx$

For random n, $\underline{m} \sim p(\underline{m})$

PCA applied to bodies



Freifeld and Black, ECCV 2012

PCA applied to DNA

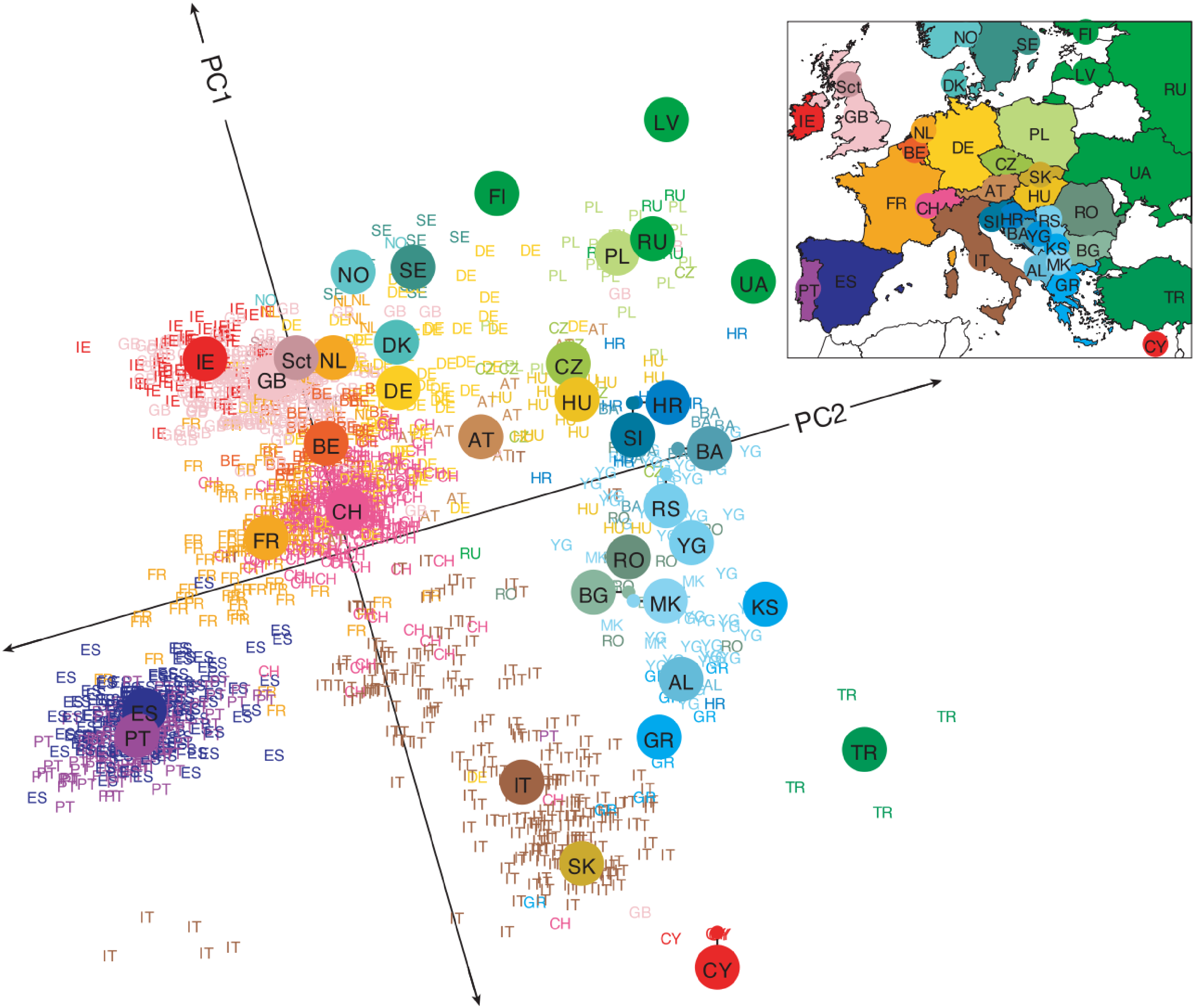
Novembre et al. (2008) — doi:10.1038/nature07331

Carefully selected both individuals and features

1,387 individuals

197,146 single nucleotide polymorphisms (SNPs)

Each person reduced to two(!) numbers with PCA


PC1
PC2
LV
FI
RU
PL
UA
NO
SE
DK
IE
Sct
GB
NL
DE
CZ
HU
HR
SI
BA
AT
BE
CH
FR
RS
YG
RO
BG
MK
KS
AL
GR
TR
ES
PT
IT
SK
CY
LV
FI
RU
UA
TR

MSc course enrollment data

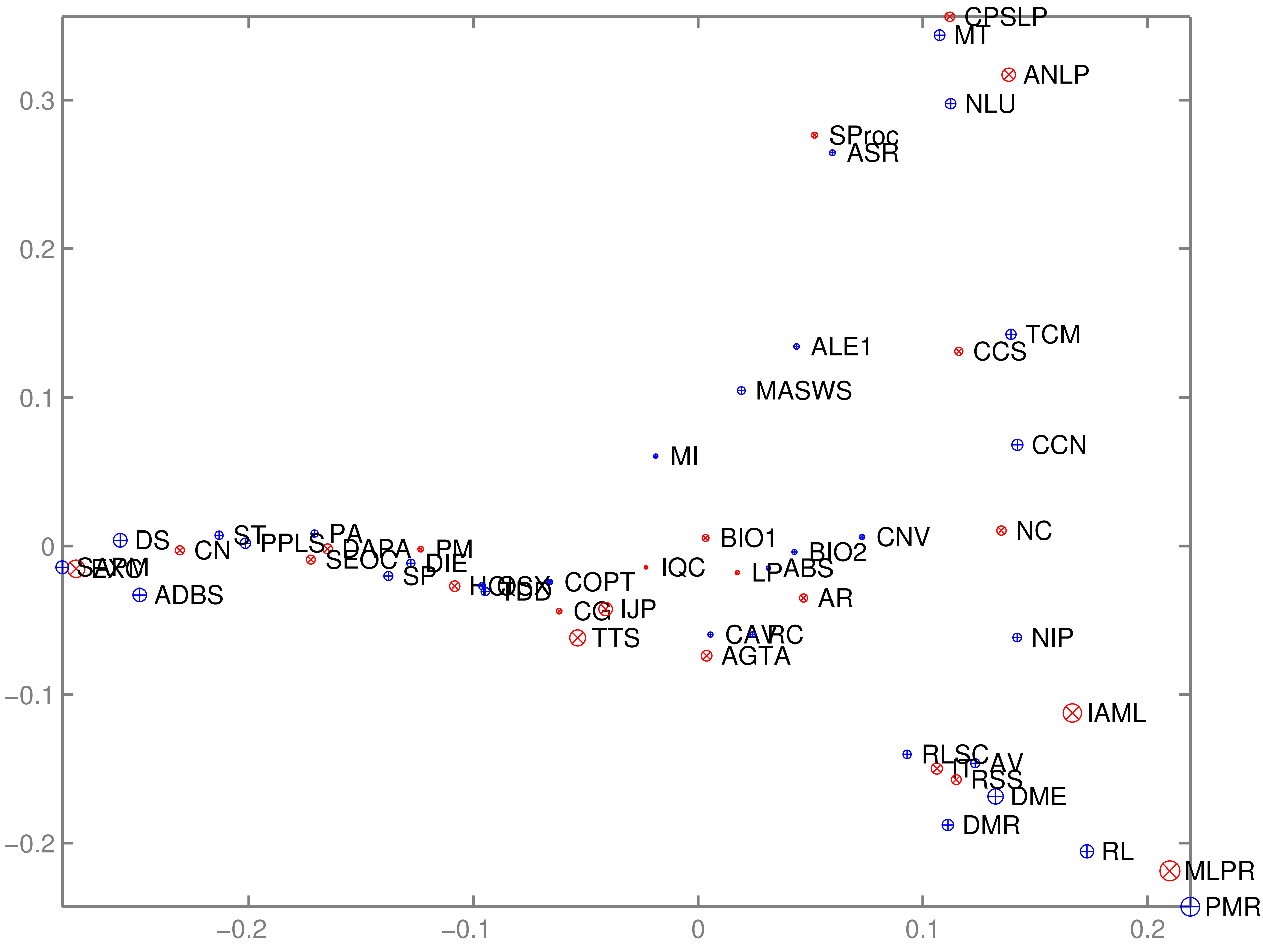
Binary $S \times C$ matrix M

$M_{sc} = 1,$ if student s taking course c

Each course is a length S vector

. . . OR each student is a length C vector

PCA applied to MSc courses



PCA applied to MSc students