COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

# Matrix factorizations and hidden variables

## Comments

- Only tested on readings from Chapters 1-7 and on basics of evaluation
  - More advanced topics I talk about now will help you use your basic knowledge, but you will not be tested on them
  - Not tested on SVMs, hidden variable models, matrix factorization, ensembles
- Hopefully you've started on Assignment 4, which is like a mini-project

## What is L(yhat, y)?

- Our goal is still to obtain discriminative model p(y | x)
- In generalized linear models, we learned a function of x to produce expected value of y:
  - func(x) = E[y | x], where func(x) = f( < x, w >)
- Want to use the same probabilistic assumption on p(y |x), e.g., Gaussian, Bernoulli, Poisson
- Now parameterize mean value with a more complicated function
  - func(x) = E[y | x], e.g. func(x) = f_1( f_2( < x, W2 >) w)

# Selecting loss functions

- If we pick an identity transfer, what is the loss?
  - squared-error (linear regression loss)
- If we pick a sigmoid transfer for the last layer, what is the loss?
  - cross-entropy (logistic regression loss)
- If we pick a tanh transfer for the last layer, what is the loss?
  - entropic loss (similar to cross-entropy)
  - this is the matching loss for tanh; see http://papers.nips.cc/paper/1610-linear-hinge-loss-and-average-margin.pdf
- If we pick rectified linear for the last layer, what is the loss?
  - no longer has a matching loss
  - usually pick least-squares loss, or do not use it as last layer

# Thought Questions

- What is the purpose of the hidden layer?
- Neural nets seem to come up in the news with grandiose titles pretty regularly these days. One recent one being Google's neural net invented its own secret language. It's difficult to parse what actually happens behind the scenes in these articles while being new to this. In this secret language article, is this just an exaggeration of a neural net doing a form of dictionary learning? Or is it simply referencing the increase of dimensionality in hidden layers?
  - Referenced article: https://techcrunch.com/2016/11/22/googles-ai-translation-tool-seems-to-have-invented-its-own-secret-internal-language/

# Handling missing features

- Straightforward for naive Bayes
  - why?

- For regression models (e.g., logistic regression), more difficult. Some options are:
  - replacing the missing values with the mean value for the attribute
  - replacing the missing value with the weighted mean of k nearest neighbors (most similar k other instances)
  - many more suggestions…

- Unsupervised learning (with matrix factorization) will be another option we will discuss
  - e.g. matrix completion

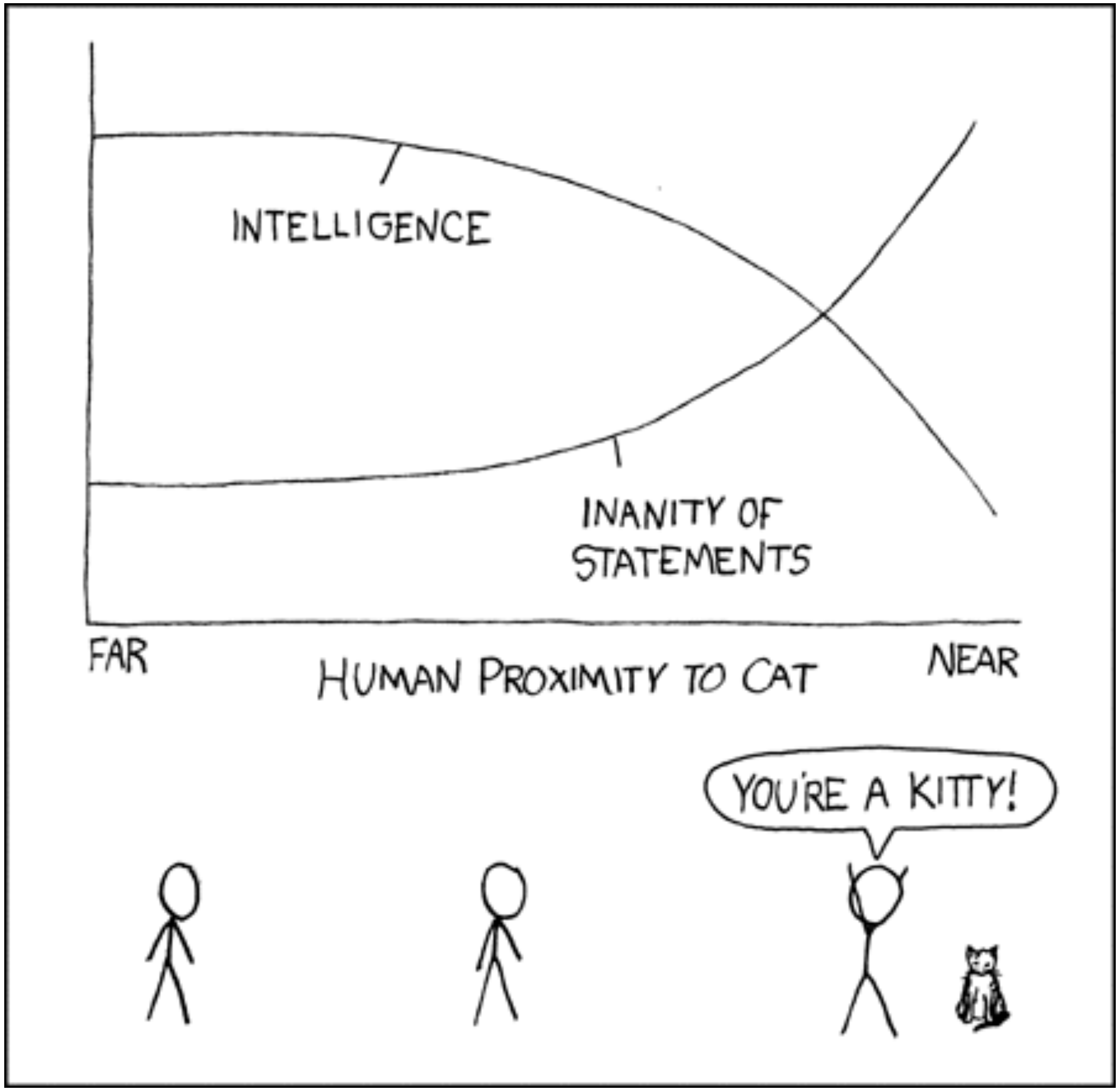
## Prediction in naive Bayes with missing features

- Training is straightforward for naive Bayes, with missing features
- How about testing, i.e., for a  new point x, how do we pick a class with naive Bayes, if we are missing a feature in x?

# Discussed hidden variables

- Underlying "state" influencing what we observe; partial observability makes what we observe difficult to interpret

- How is "proximity to a kitty" a hidden variable?



# Why hidden variables?

- Making up underlying or hidden variables can make it easier to specify a model
  - maybe you really think there are two factors causing the behavior
- Example: pdf over a variable is complex
  - simplifies the problem to represent the pdf as a weighted sum over Gaussians, where $i^{th}$ Gaussian corresponds to $h = i$
- Hidden variables are useful for interpretation
- Many approaches focus on extracting underlying factors
- Let's look at more examples!

# Gaussian mixture model

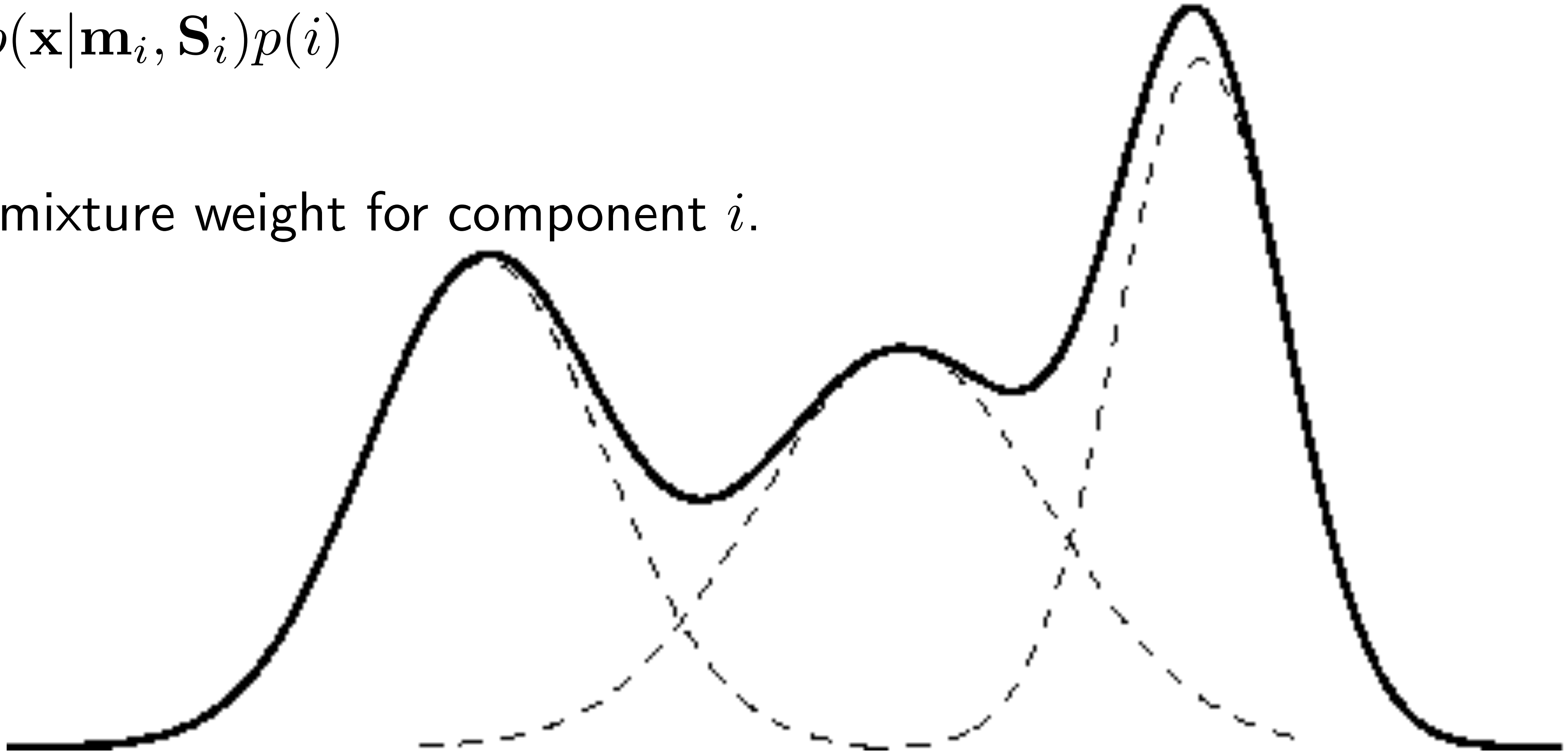
A $D$ dimensional Gaussian distribution for a continuous variable $\mathbf{x}$ is

$$p(\mathbf{x}|\mathbf{m},\mathbf{S}) = \frac{1}{\sqrt{\det(2\pi\mathbf{S})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\mathbf{S}^{-1}(\mathbf{x}-\mathbf{m})\right\}$$

where $\mathbf{m}$ is the mean and $\mathbf{S}$ is the covariance matrix. A mixture of Gaussians is then

$$p(\mathbf{x}) = \sum_{i=1}^{H} p(\mathbf{x}|\mathbf{m}_i,\mathbf{S}_i)p(i)$$

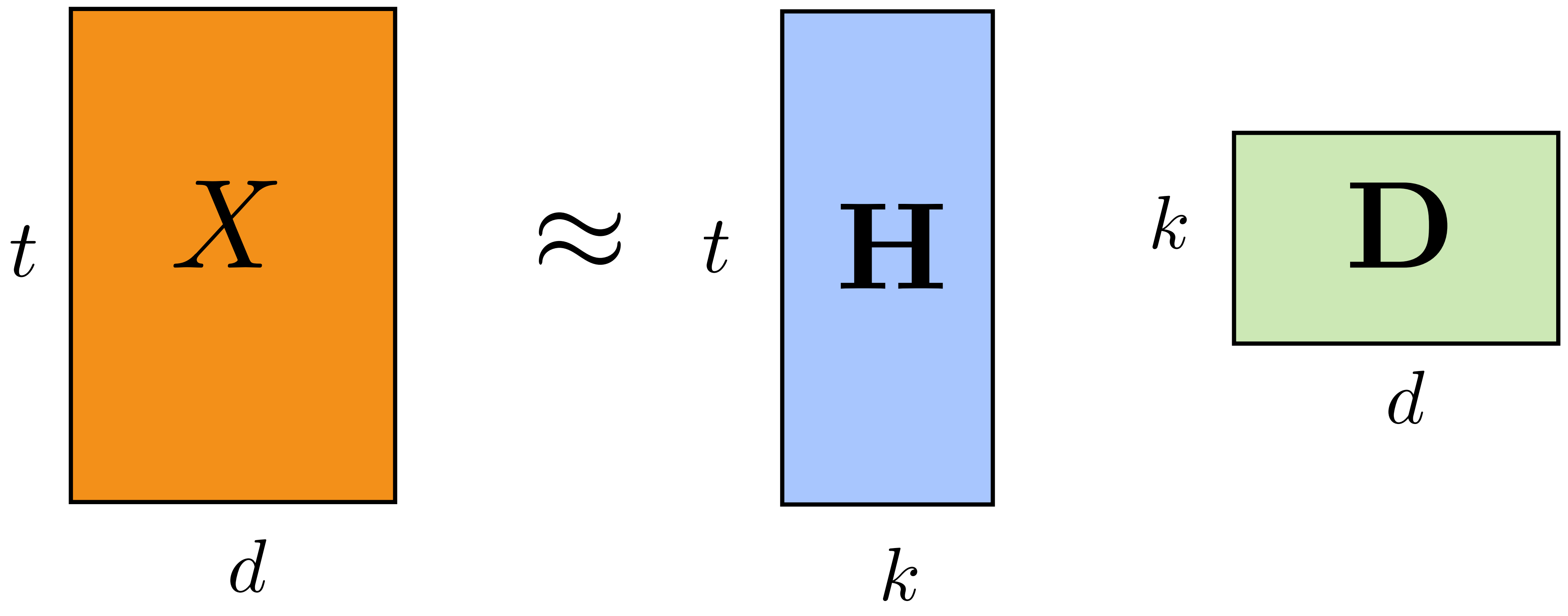
where $p(i)$ is the mixture weight for component $i$.



# Factor analysis

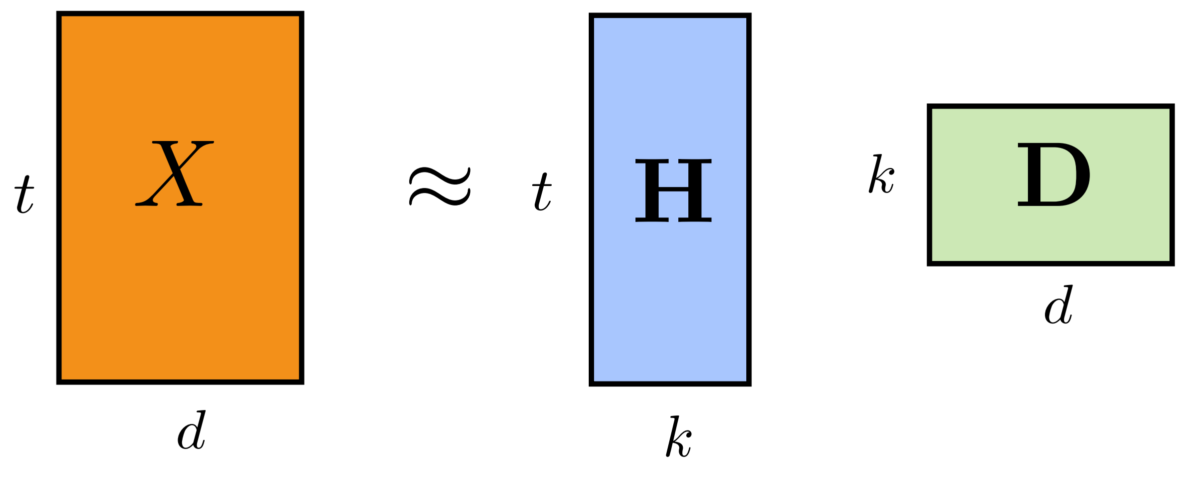
- Imagine you have test scores from 10 subjects, for 1000 students
- As a psychologist, you hypothesize there are two kinds of intelligence: verbal and mathematical
- You cannot observe the factors (hidden variables)
- Instead, you would like to see if these two factor explain the data, where x is the vector of test scores of a student
- Want to find: x = d1 h1 + d2 h2, where d1 and d2 are vectors h1 = verbal intelligence and h2 = mathematical intelligence

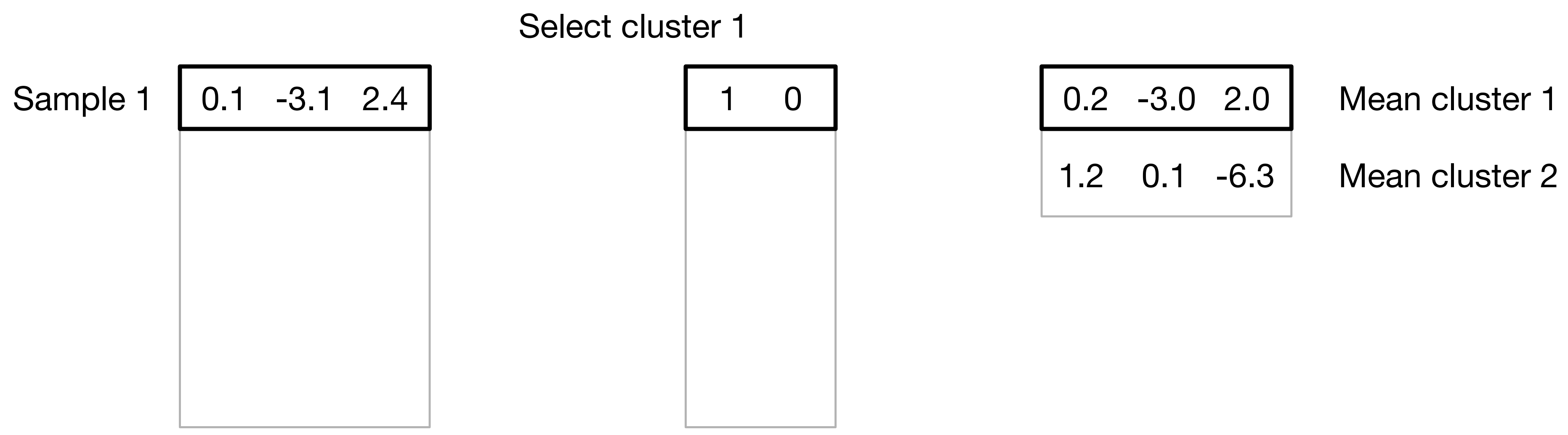
# Matrix factorization



If k < d, then we obtain dimensionality reduction (PCA)

# Example: K-means



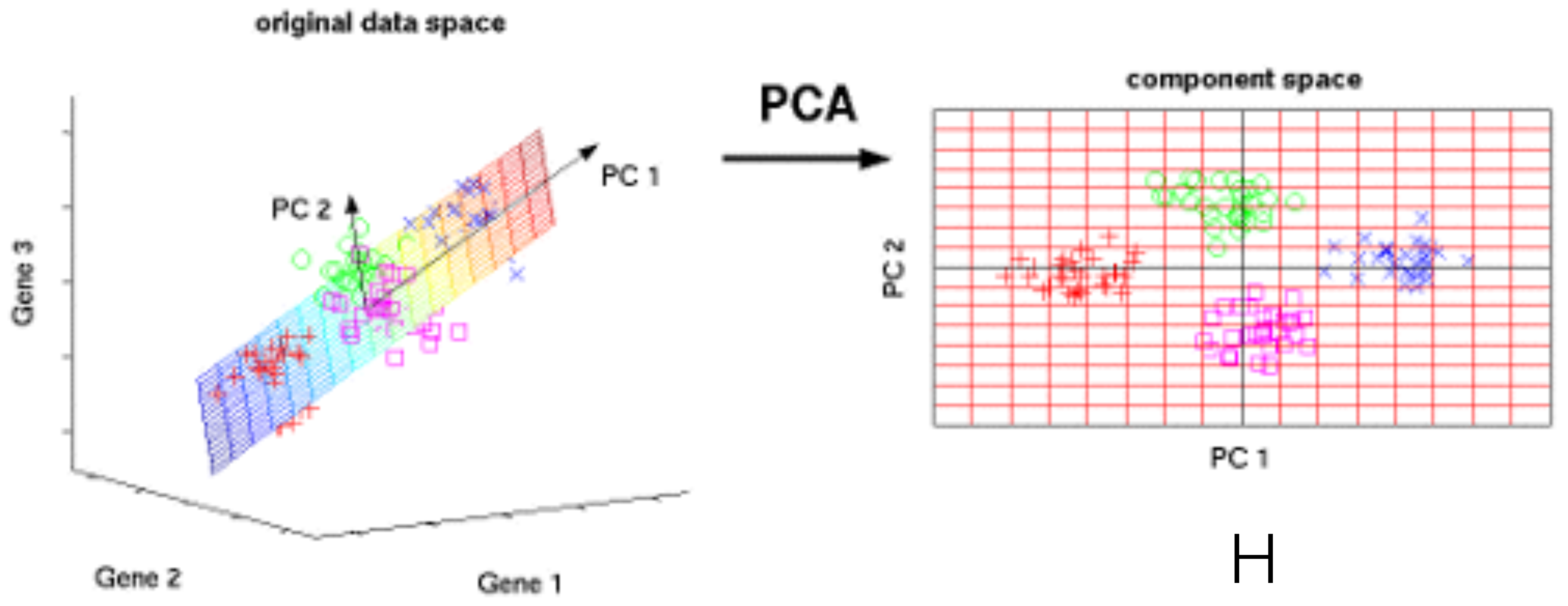


$$\left\|\mathbf{x} - \sum_{i=1}^{2} 1\left(\mathbf{x} \text{ in cluster } i\right) \mathbf{d}_i\right\|_2^2 = \|\mathbf{x} - \mathbf{h}\mathbf{D}\|_2^2$$

where $\mathbf{h} = [1 \quad 0]$ or $\mathbf{h} = [0 \quad 1]$ and $\mathbf{D} = [\mathbf{d}_1 \ ; \ \mathbf{d}_2]$.

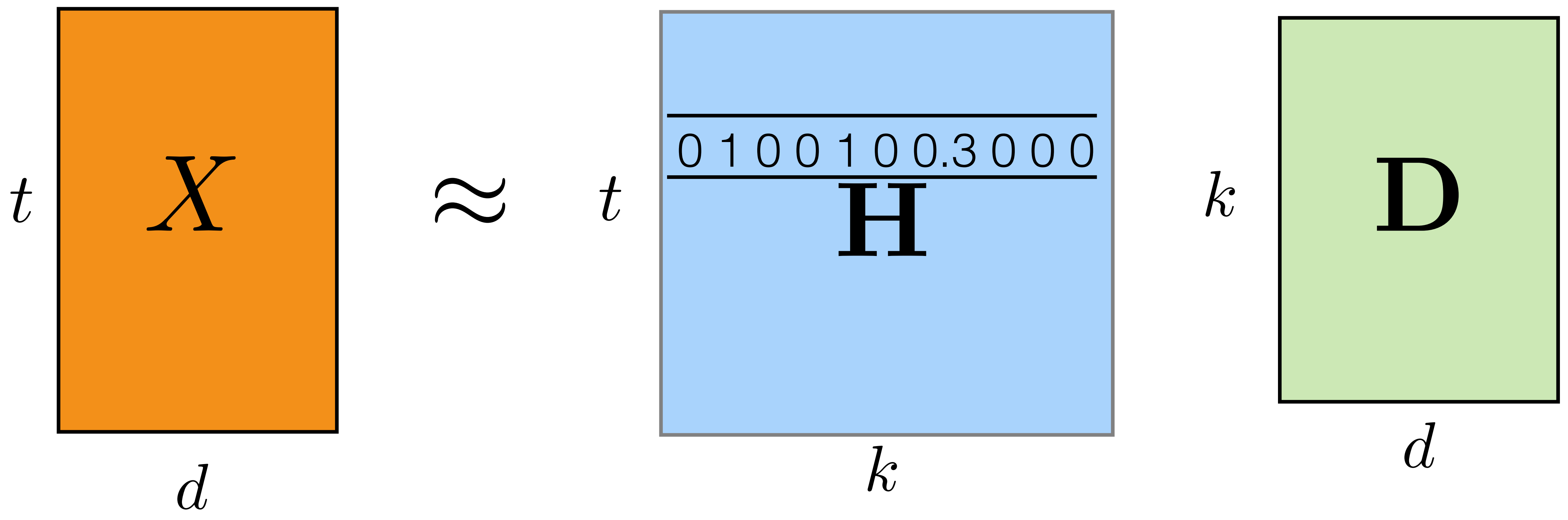
# Principal components analysis

- New representation is k left singular vectors that correspond to k largest singular values
  - can equivalently extract k eigenvectors of covariance matrix X^T X, that correspond to k largest eigenvalues
- Not the same as selecting k features, but rather projecting features into lower-dimensional space



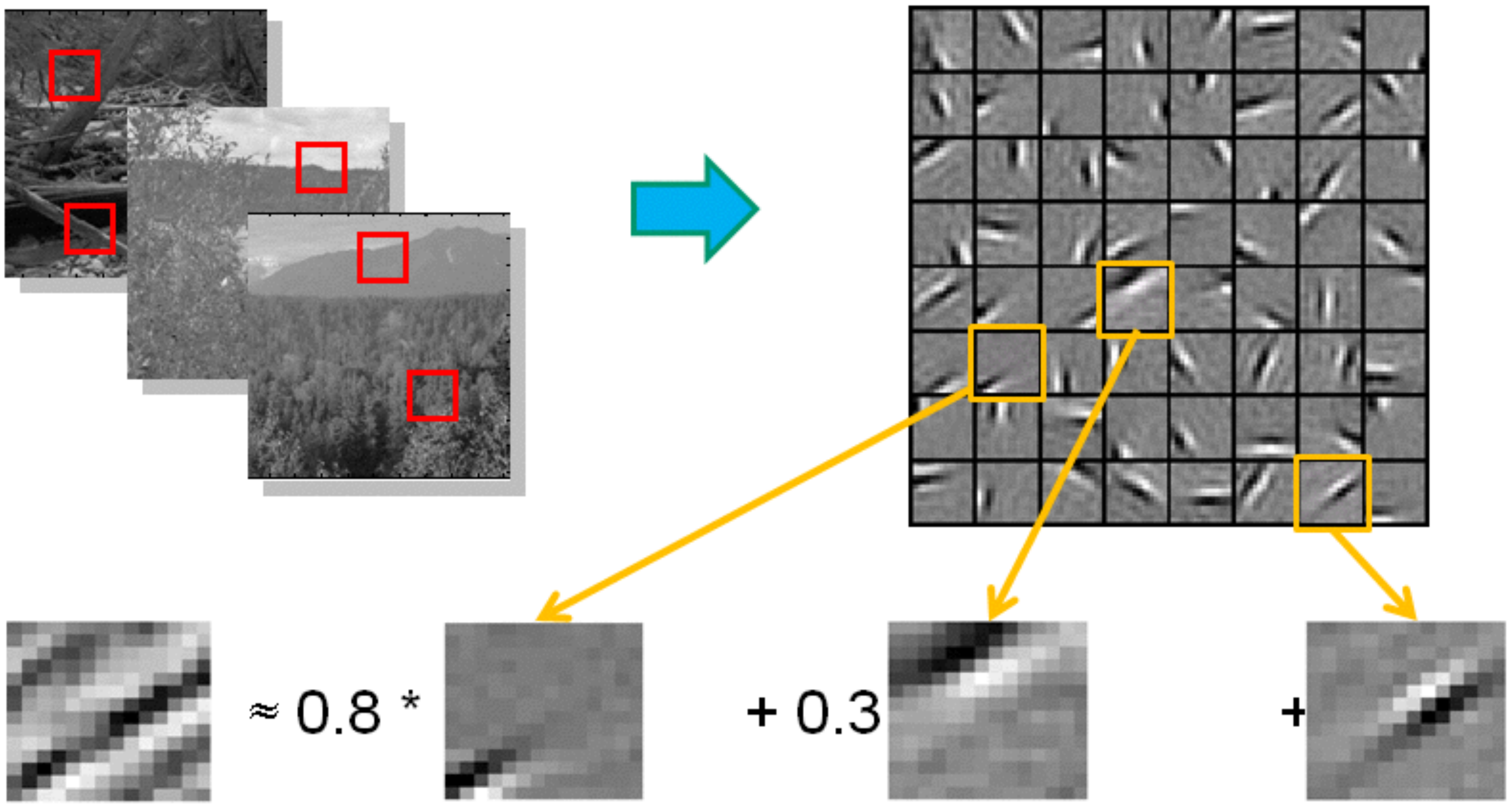
H

# Sparse coding



- For sparse representation, usually k > d
- Many entries in new representation are zero

## Sparse coding illustration



$[a_1, \ldots, a_{64}] = [0, 0, \ldots, 0, \mathbf{0.8}, 0, \ldots, 0, \mathbf{0.3}, 0, \ldots, 0, \mathbf{0.5}, 0]$
(feature representation)

Slide credit: Andrew Ng

Compact & easily interpretable

## Whiteboard

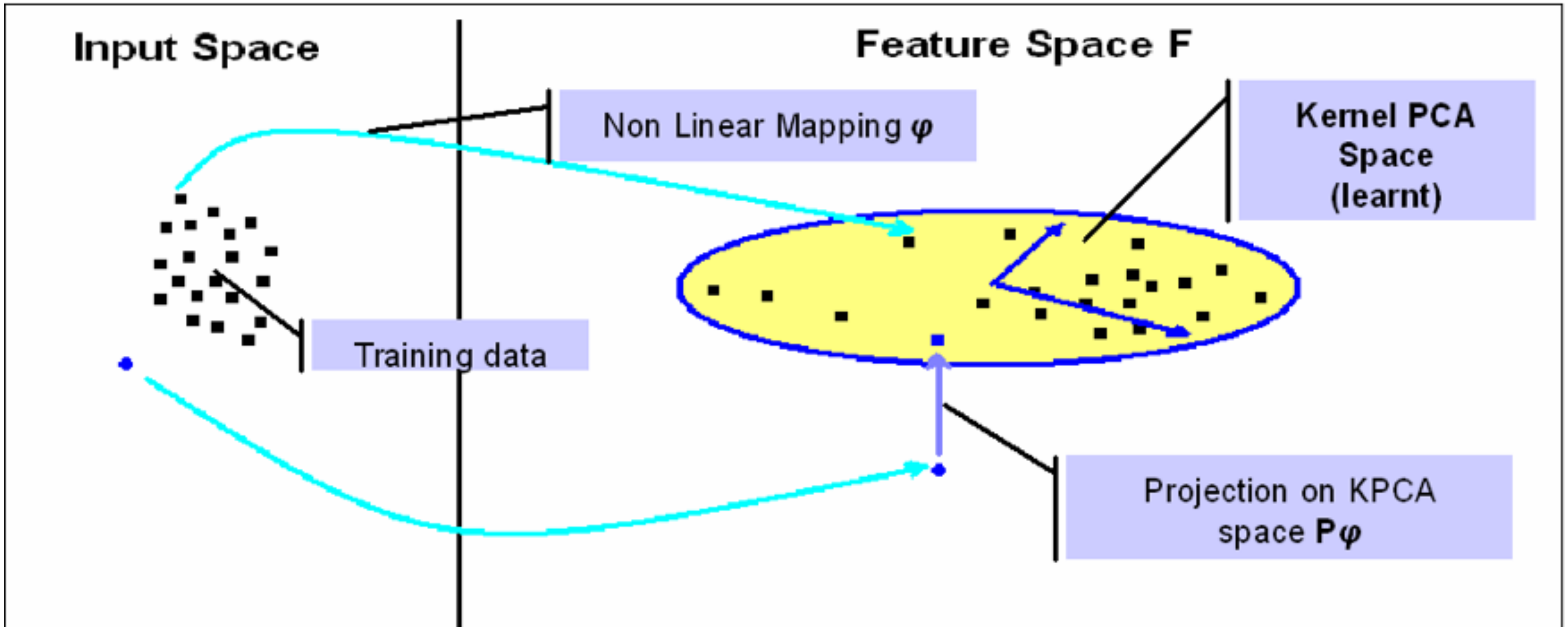
- Formulation of PCA as dictionary learning
- Sparse coding formulation
- General algorithmic approach

# Kernel PCA

- Non-linear dimensionality reduction: PCA, but on kernel matrix K
- **Step 1:** expand/transform original features into kernel features

$$\mathbf{X} \in \mathbb{R}^{T \times d} \rightarrow \mathbf{K} \in \mathbb{R}^{T \times T} \text{ where } \mathbf{K}_{ij} = k(\mathbf{X}_{i:}, \mathbf{X}_{j:})$$

- **Step 2:** perform PCA on kernel matrix (dimensionality reduction), to get new non-linear features (rather than just linear projection)



# Linear kernel PCA

- Linear kernel PCA is equivalent to PCA

- Linear kernel:

$$\mathbf{X}\mathbf{X}^\top \text{ where } k(\mathbf{X}_{i:}, \mathbf{X}_{j:}) =< \mathbf{X}_{i:}, \mathbf{X}_{j:} >= \mathbf{X}_{i:}\mathbf{X}_{j:}^\top$$

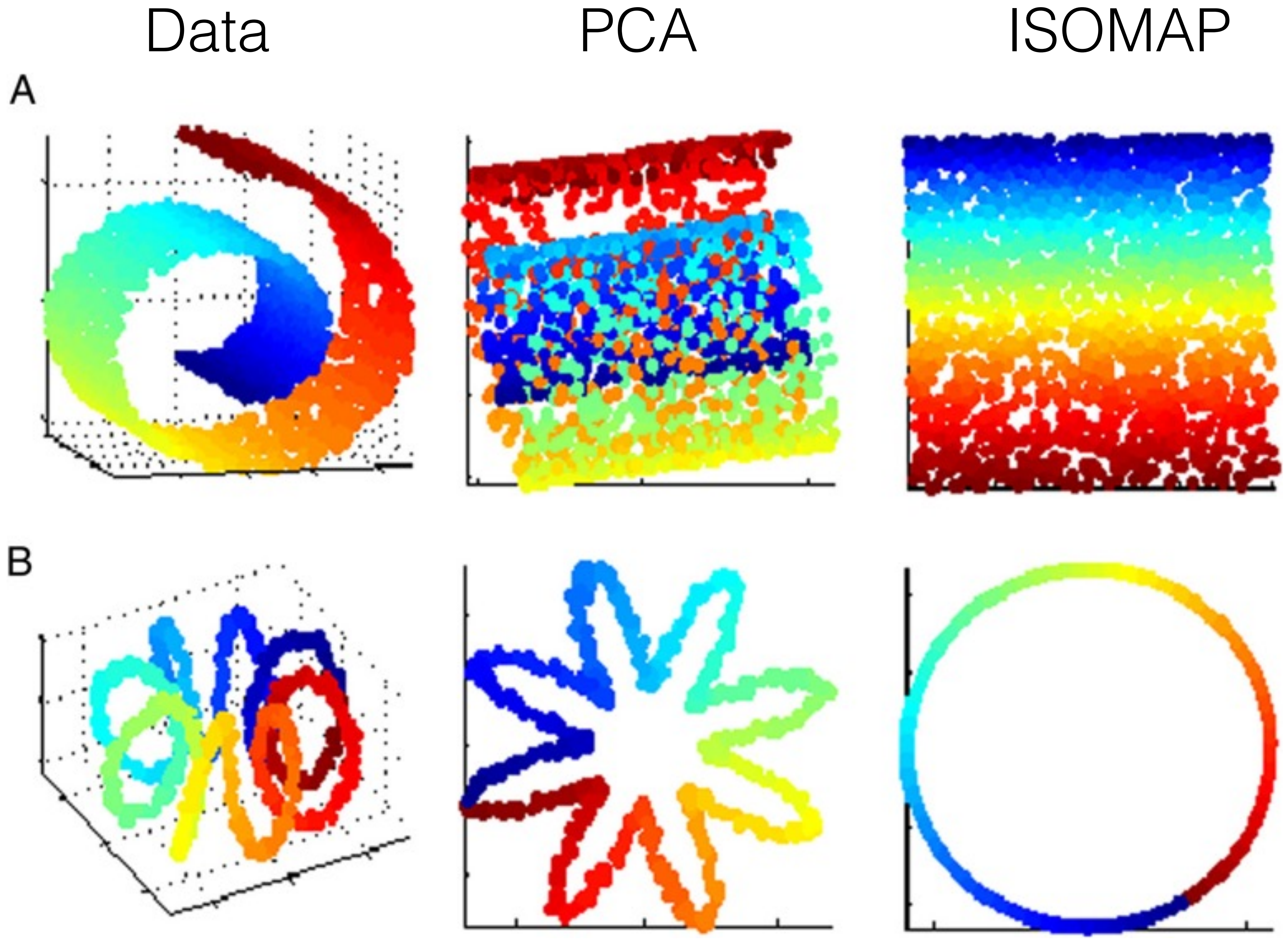
- How could you show that they give the same representation H (i.e., same principal components)?

- Hint: recall that the principal component are the left singular vectors of the matrix

## Isomap

- Non-linear dimensionality reduction using a specific kernel
    - graph kernel: similarity is shortest path between two nodes, after creating a neighborhood graph
- New kernel “unfolds” the data (better preserves distances than the distance used for PCA)

# Isomap vs PCA


Data
PCA
ISOMAP
A
B