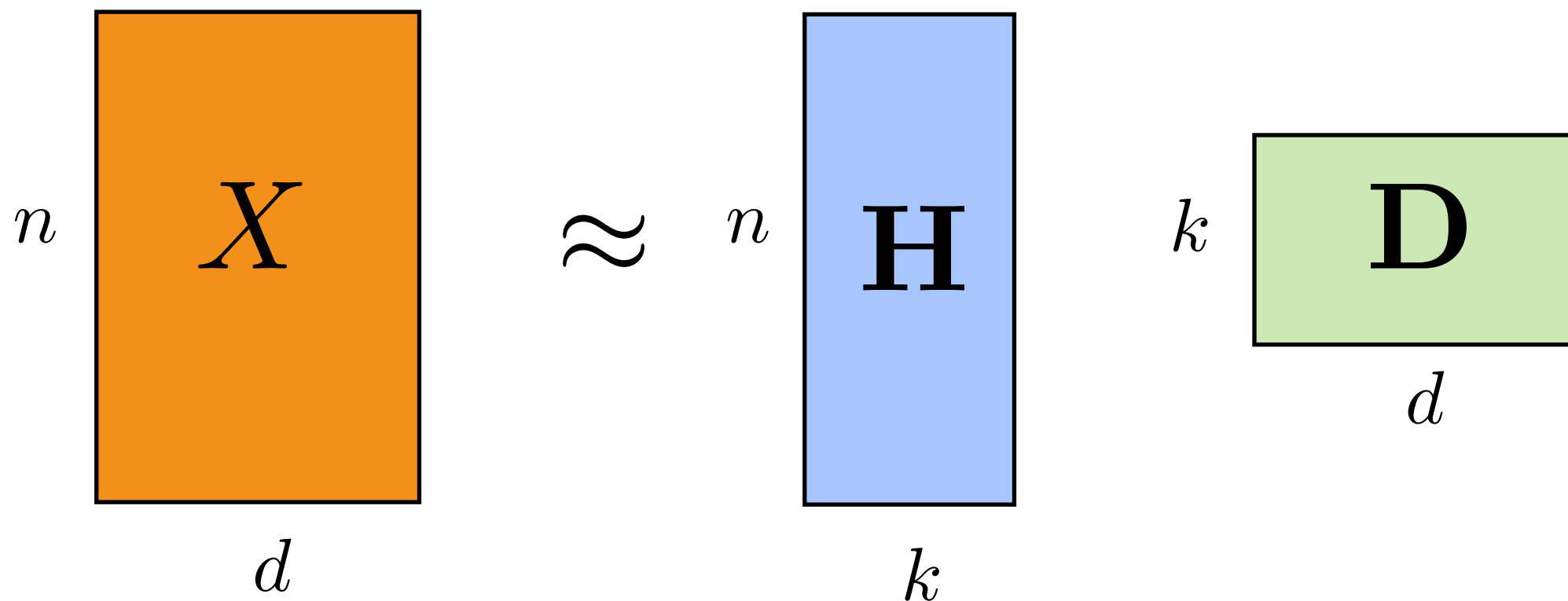


Matrix factorization and embeddings



Reminders/Comments

- Initial draft of Mini-project due today

Today

- Back to representation learning
- How do we transform an input x into a new vector $\phi(x)$ that is
 - composed of real values
 - enables nonlinear functions in terms of x , using only a generalized linear model with $\phi(x)$
 - and has other potentially desirable properties, like compactness or...

Neural networks summary

- Discussed basics, including
- Basic architectures (fully connected layers with activations like sigmoid, tanh, and relu)
- How to choose the output loss
 - i.e., still using the GLM formulation
- Learning strategy: gradient descent (called back-propagation)
- Basic regularization strategies

How else can we learn the representation?

- Discussed how learning can be done in simple ways even for “fixed representations”
 - e.g., learn the centres for radial basis function networks
 - e.g., learn the bandwidths for Gaussian kernel
- In general, this problem has been tackled for a long time in the field of unsupervised learning
 - where the goal is to analyze the underlying structure in the data

Using factorizations

- Many unsupervised learning and semi-supervised learning problems can be formulated as factorizations
 - PCA, kernel PCA, sparse coding, clustering, etc.
- Also provides an way to embed more complex items into a shared space using co-occurrence
 - e.g., matrix completion for Netflix challenge
 - e.g., word2vec

Intuition (factor analysis)

- Imagine you have test scores from 10 subjects (topics), for 1000 students
- As a psychologist, you hypothesize there are two kinds of intelligence: verbal and mathematical
- You cannot observe these 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: $x = d_1 h_1 + d_2 h_2$, where d_1 and d_2 are vectors h_1 = verbal intelligence and h_2 = mathematical intelligence
- Having features h_1 and h_2 would give a compact, intuitive rep

Example continued

- Imagine you have test scores from 10 subjects (topics), for 1000 students
- Learned basis vectors \mathbf{d}_1 and \mathbf{d}_2 that reflect scores for a student with high verbal or math intelligence, respectively

Obtain $\mathbf{x}_5 = \mathbf{d}_1 h_{5,1} + \mathbf{d}_2 h_{5,2}$

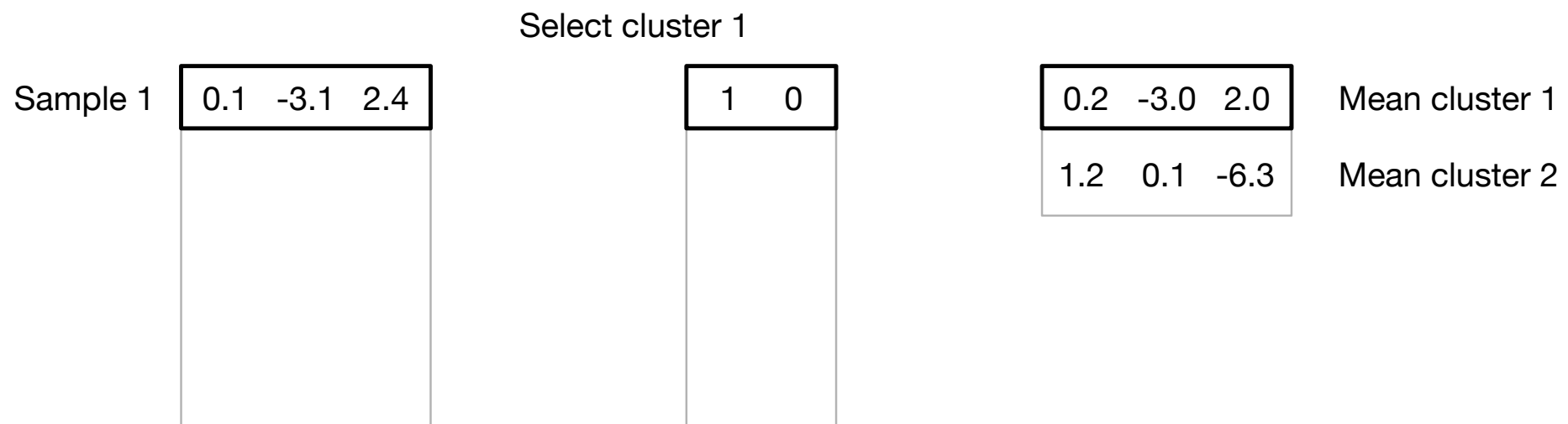
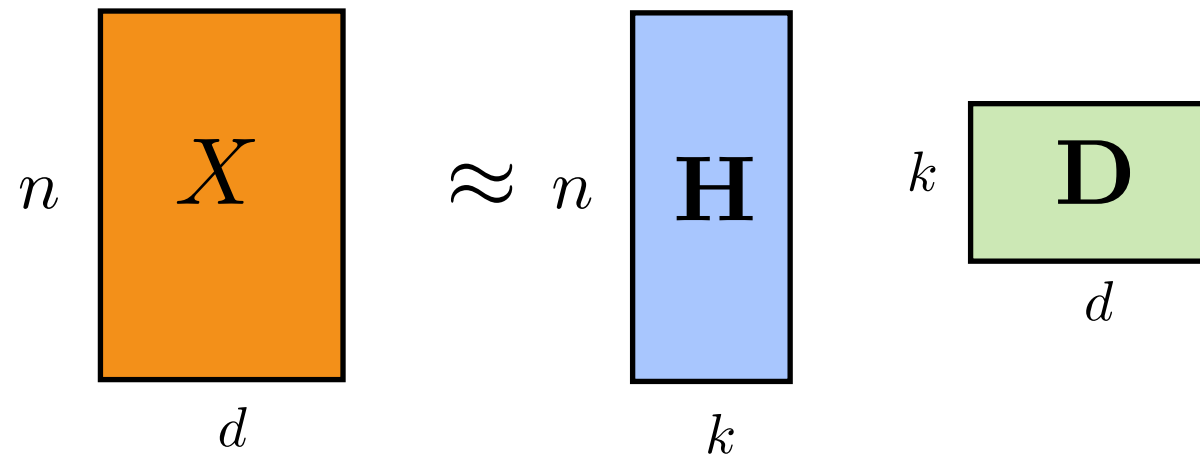
where $h_{5,1} =$ verbal intelligence and $h_{5,2} =$ math intelligence

- Features $[h_{5,1}, h_{5,2}]$ provide useful attributes about student 5

Whiteboard

- Linear neural network
- Auto-encoders and Matrix factorization
- Learning (latent) attributes of inputs

Example: K-means



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

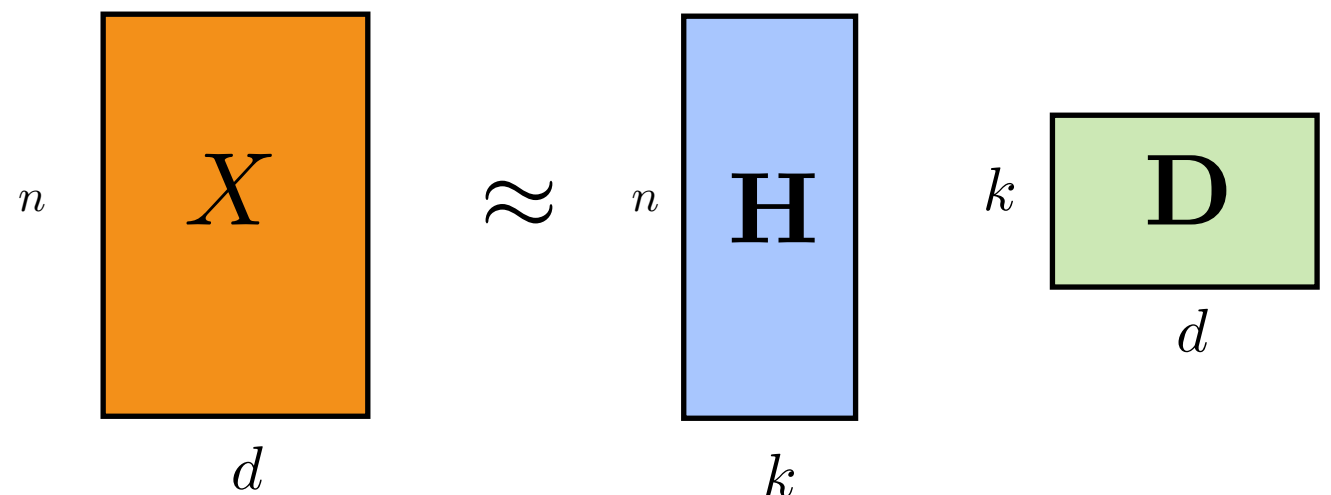
where $\mathbf{h} = [1 \ 0]$ or $\mathbf{h} = [0 \ 1]$ and $\mathbf{D} = [\mathbf{d}_1 ; \mathbf{d}_2]$.

Dimensionality reduction

- If set inner dimension $k < d$, obtain dimensionality reduction
- Recall that the product of two matrices \mathbf{H} and \mathbf{D} has rank at most the minimum rank of \mathbf{H} and \mathbf{D}

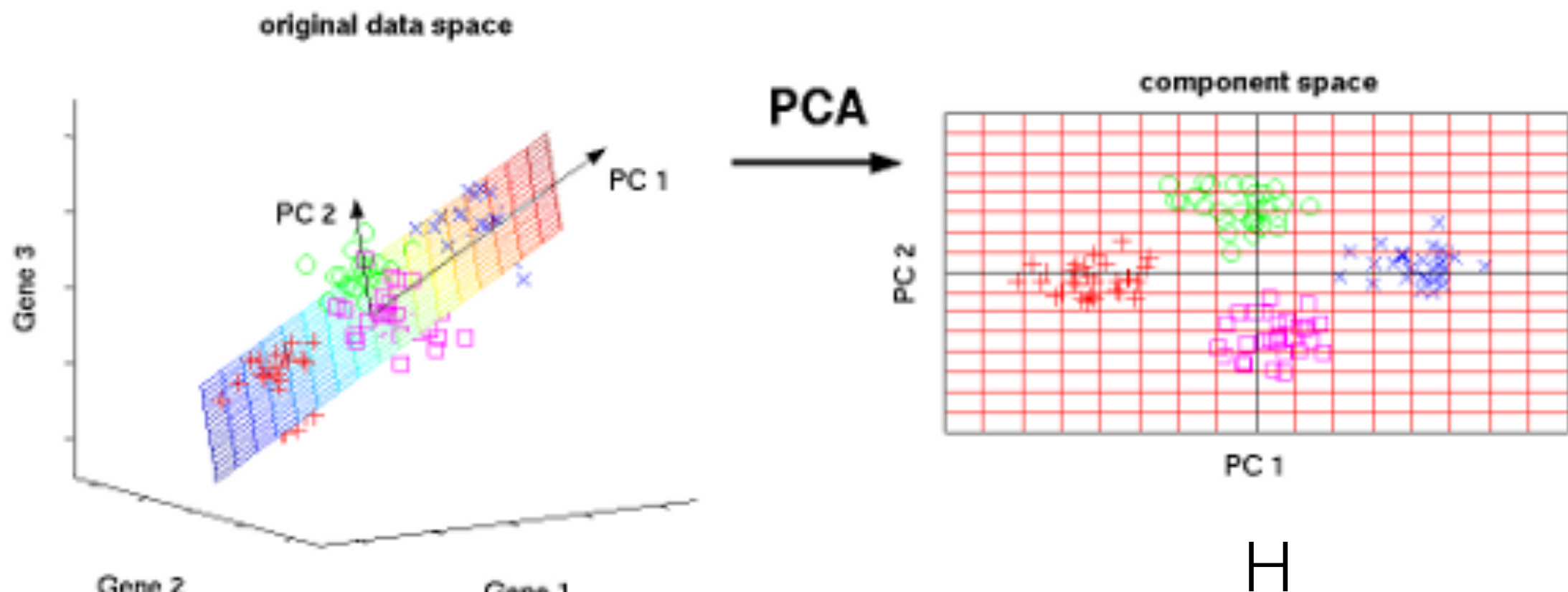
$$\text{rank}(\mathbf{HD}) \leq \min(\text{rank}(\mathbf{H}), \text{rank}(\mathbf{D}))$$

- Even if $d = 1000$, if we set $k = 2$, then we get a reconstruction of X that is only two-dimensional
 - we could even visualize the data! How?



Principal components analysis

- New representation is k left singular vectors that correspond to k largest singular values
 - i.e., for each sample x , the corresponding k -dimensional h is the rep
- Not the same as selecting k features, but rather projecting features into lower-dimensional space



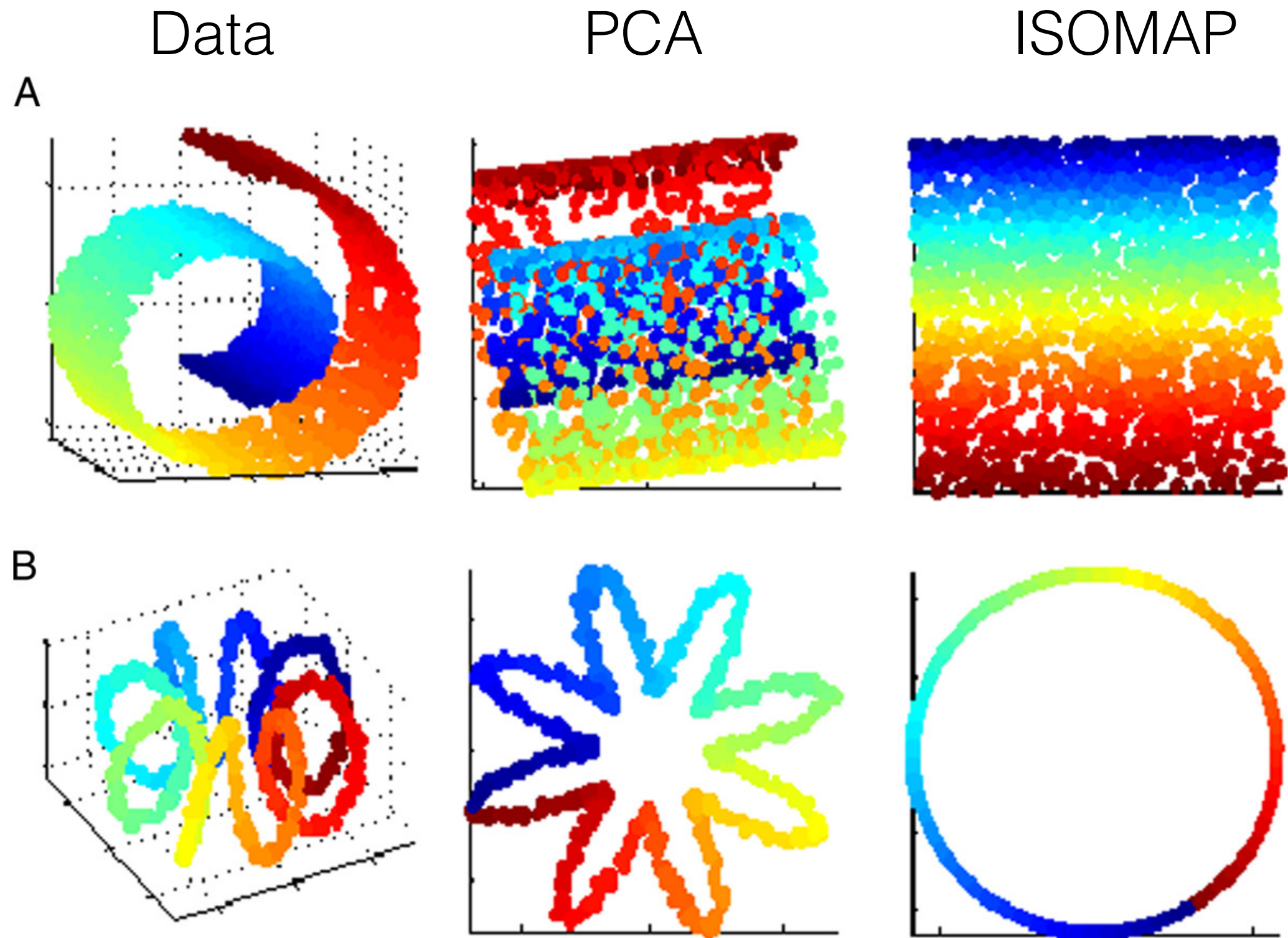
Do these make useful features?

- Before we were doing (huge) nonlinear expansions
- PCA takes input features and reduces the dimension
- This constrains the model, cannot be more powerful
- Why could this be helpful?
 - Constraining the model is a form of regularization: could promote generalization
 - Sometimes have way too many features (e.g., someone overdid their nonlinear expansion, redundant features), want to extract key dimensions and remove redundancy and noise
 - Can be helpful for simply analyzing the data, to choose better models

What if the data does not lie on a plane?

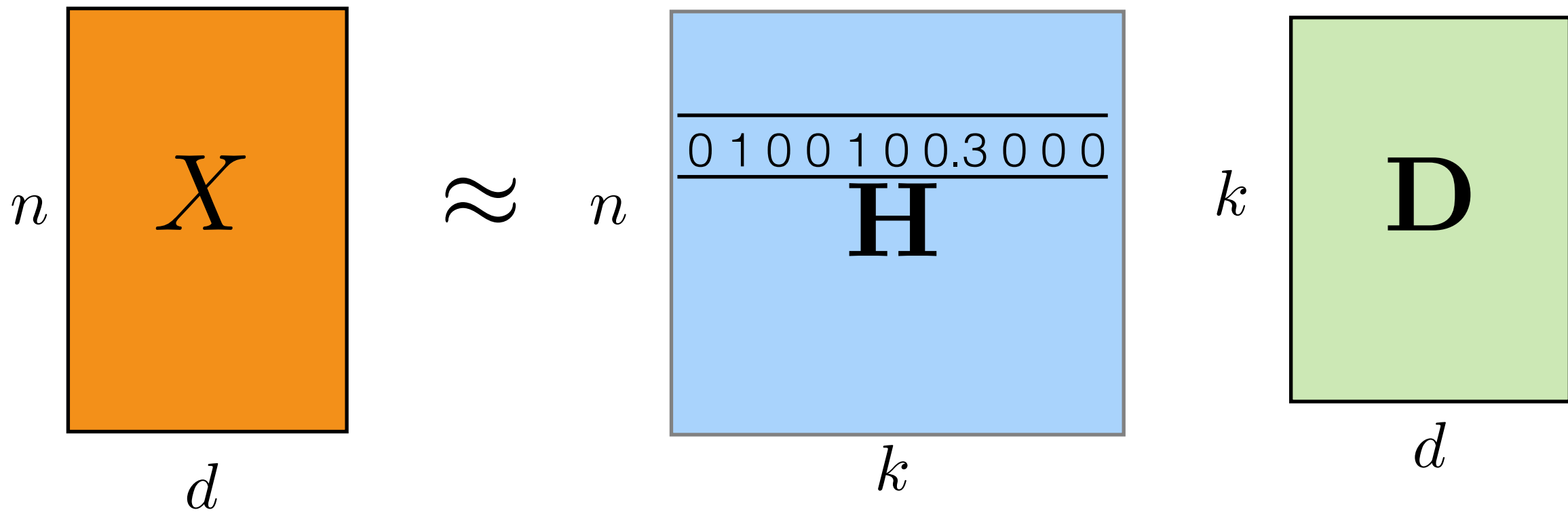
- Can do non-linear dimensionality reduction
- Interestingly enough, many non-linear dimensionality reduction techniques correspond to PCA, but first by taking a nonlinear transformation of the data with a (specialized) kernel
 - Isomap, Laplacian eigenmaps, LLE, etc.
- Can therefore extract a lower-dimensional representation on a curved manifold, can better approximate input data in a low-dimensional space
 - which would be hard to capture on a linear surface

Isomap vs PCA



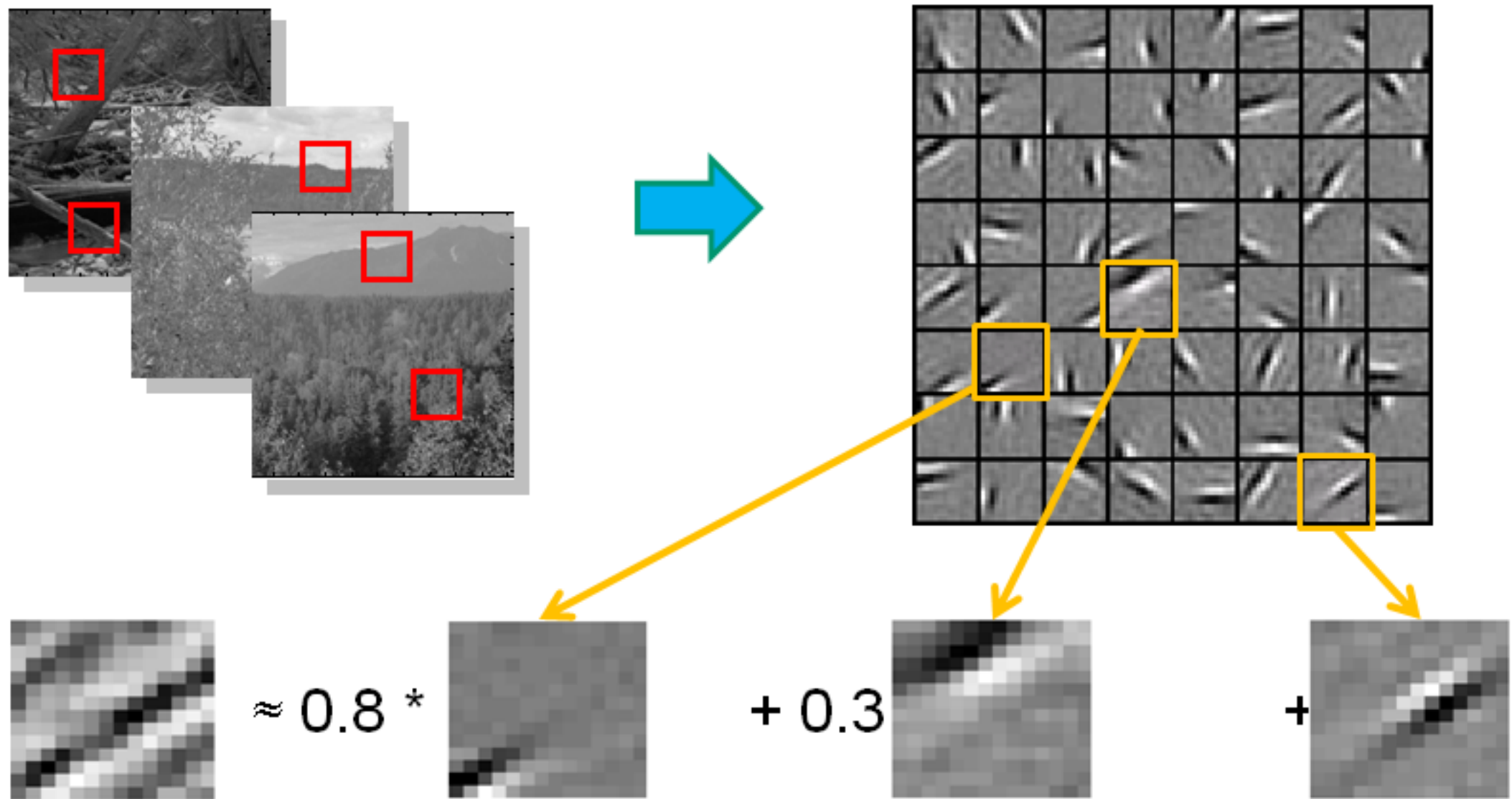
*Note: you don't need to know Isomap, just using it as an example

Sparse coding



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

Sparse coding illustration



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

Comments: Nov. 28, 2019

- TAs will send you two initial drafts to review
- Andy Patterson will do a review lecture next Thursday

Summary so far

- Factorization another way to obtain re-representations of data
- One advantage over NNs: lets us easily impose structure on the re-representation, since we optimize for it explicitly
- One disadvantage over NNs: usually restricted to pretty simple re-representation (e.g., linear weighting of dictionary items)
- **Disclaimer:** I won't actually ask you any questions on the final about Matrix Factorization

Embeddings with co-occurrence

- Embed complex items into a shared (Euclidean) space based on their relationships to other complex items
- Examples:
 - word2vec
 - users and movies
 - gene sequences

Consider word features

- Imagine want to predict whether a sentence is positive or negative (say with logistic regression)
- How do we encode words?
- One basic option: a one-hot encoding. If there are 10000 words, the i th word has a 1 in the i th location of a 10000 length vector, and zero everywhere else.
- This is a common way to deal with categorical variables, but with 10000 words this can get big!
- Can we get a more compact representation of a word?

Co-occurrence matrix example

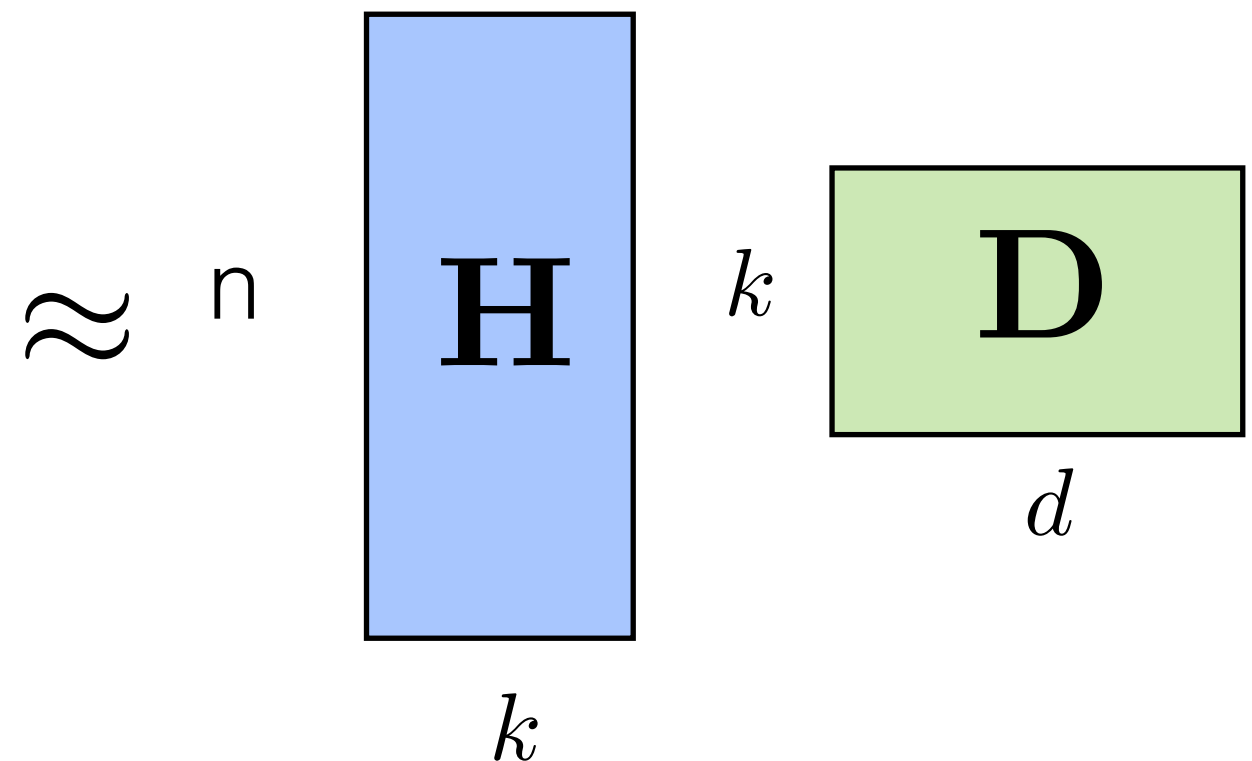
- X is count of words (rows) and context (columns), where for word i the count is the number of times a context word j is seen within 2 words (say) of word i
- Each word is a one-hot encoding; if there are 10000 words, each row corresponds to 1 word, and X is 10000x10000
 - I like deep learning.
 - I like NLP.
 - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

How obtain embeddings?

- Words i and s that have similar context counts should be embedded similarly
- Factorize co-occurrence matrix

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0



\mathbf{H}_i : = representation for word i

Algorithm

$$\min_{H, D} \sum_{(i, j)} (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2$$

$$\nabla_D \sum_{(i, j)} (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2 = \sum_{(i, j)} \nabla_D (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2$$

$$\nabla_{D_{:j}} (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2 = -2(X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j}) \mathbf{H}_{i:}$$

Gradient descent on H and D until convergence

Pros/cons of rep learning approaches

- Neural networks
 - ✓ demonstrably useful in practice
 - ✓ theoretical representability results
 - can be difficult to optimize, due to non-convexity
 - properties of solutions not well understood
 - not natural for missing data
- Matrix factorization models
 - ✓ widely used for unsupervised learning
 - ✓ simple to optimize, with well understood solutions in many situations
 - ✓ amenable to missing data
 - much fewer demonstrations of utility

Neural Network Embedding

- We saw that the solution for a linear Auto-encoder (an NN) corresponds to PCA (which is a factorization approach)
- Similarly, we can input one-hot encoding of words into an auto-encoder and predict context vectors as outputs
- The hidden layer can be used as an embedding
 - again, we have a bit less control over if it exactly has the properties we want it to have
- Why is it called an embedding? Isn't the hidden layer a representation?

Missing data

- Can easily perform factorization even with missing data
- Important in an area called matrix completion or collaborative filtering
- This contrasts NNs, where it is less clear how to handle missing data (why?)

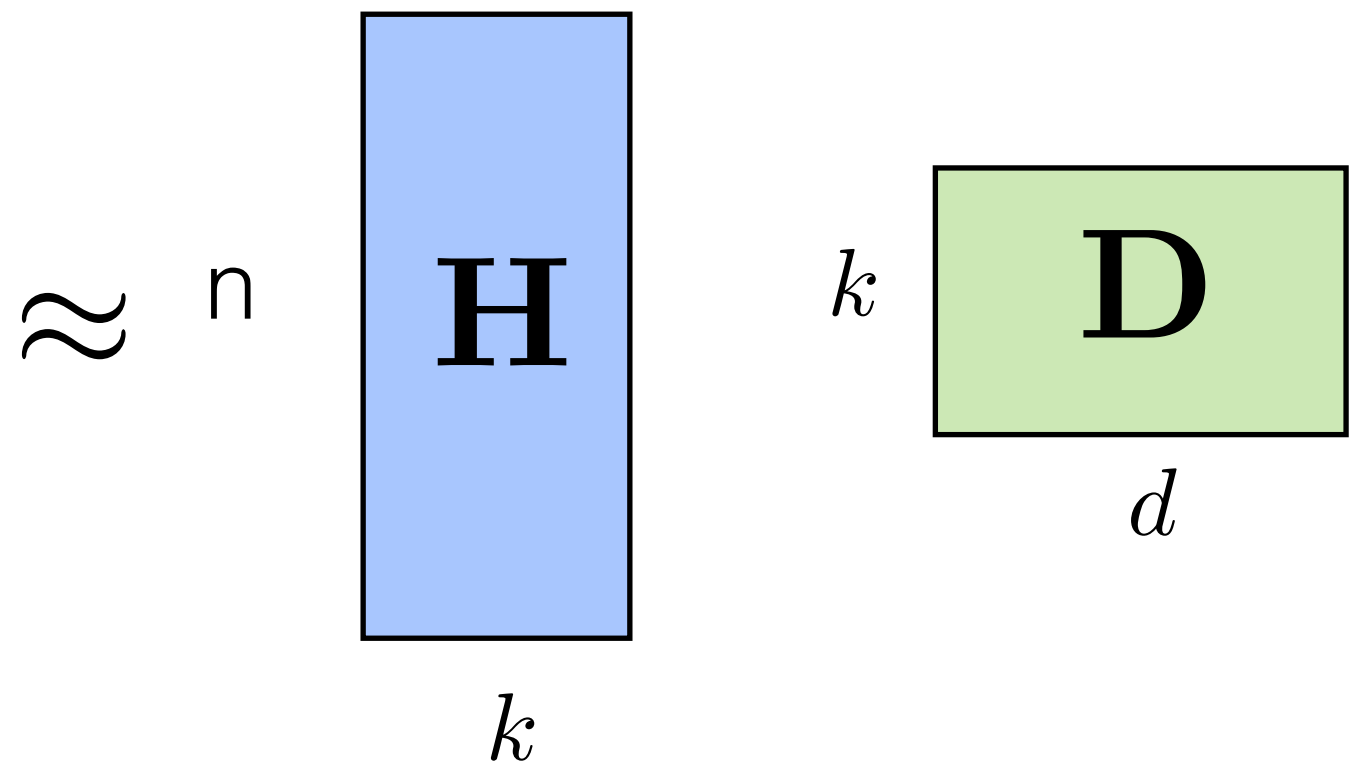
Embedding Movies and Users

Example $\mathbf{H}_{i:}$ = [like comedies, ...] = [1, ...]

$\mathbf{D}_{:j}$ = [is a comedy, ...] = [1, ...]

$\mathbf{H}_{i:} \mathbf{D}_{:j} = 1 + \dots$
 movies

		2		1			4				5	
		5		4				?		1		3
			3		5			2				
	4			?			5		3		?	
			4		1	3				5		
				2				1	?			4
		1					5		5		4	
			2		?	5		?		4		
		3		3		1		5		2		1
		3				1			2		3	
		4			5	1			3			
			3				3	?				5
	2	?		1		1						
			5			2	?		4		4	
		1		3		1	5		4		5	
	1		2			4				5	?	



$\mathbf{H}_{i:}$ is the representation of user i

$\mathbf{D}_{:j}$ is the representation of movie j

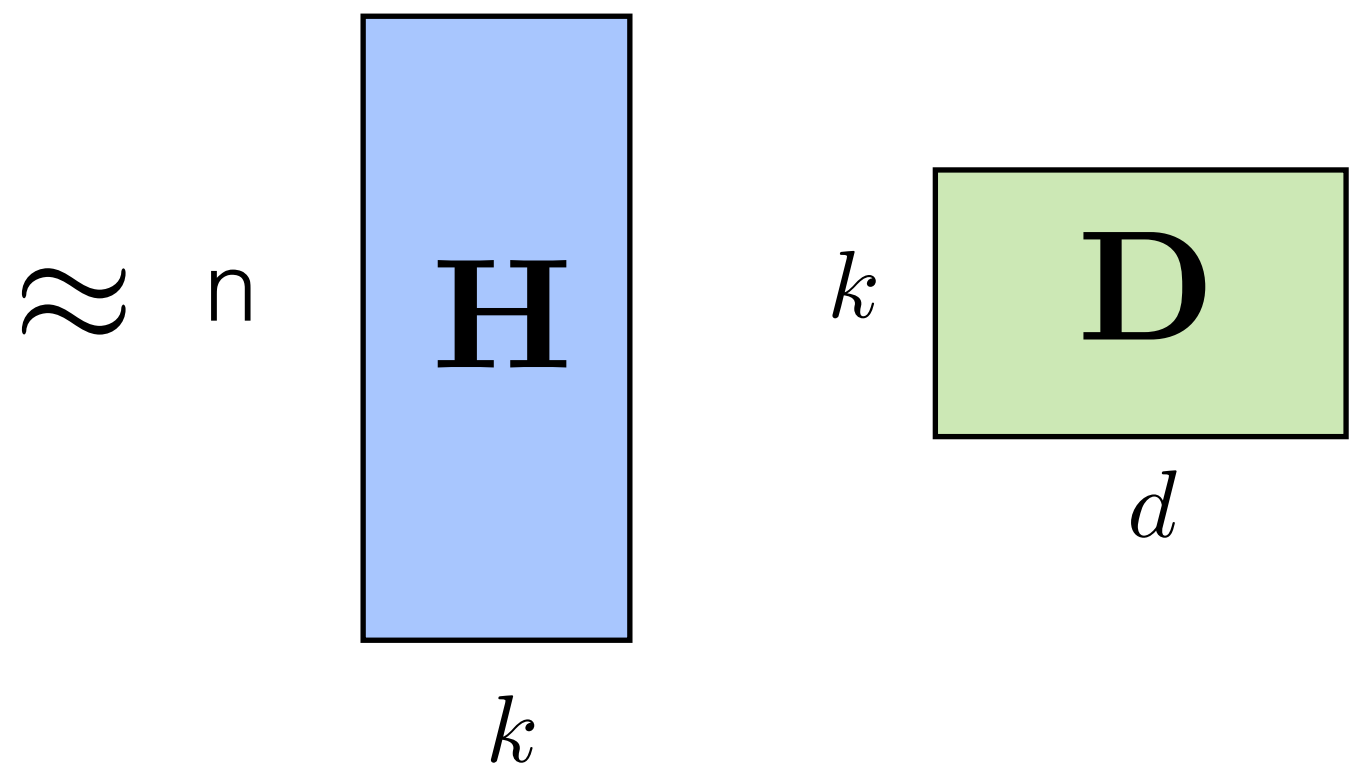
Matrix completion

movies

	2	1			4				5	
	5	4				?		1		3
		3		5		2				
4			?		5		3		?	
		4		1	3			5		
			2			1	?			4
	1				5		5		4	
		2		?	5		?		4	
	3		3		1		5		2	1
	3				1			2		3
	4			5	1			3		
		3				3	?			5
2	?		1		1					
		5			2	?		4		4
	1		3		1	5		4		5
1		2			4				5	?

users

Subspace (low-rank) form



$\mathbf{H}_{i:}$ is the representation of user i

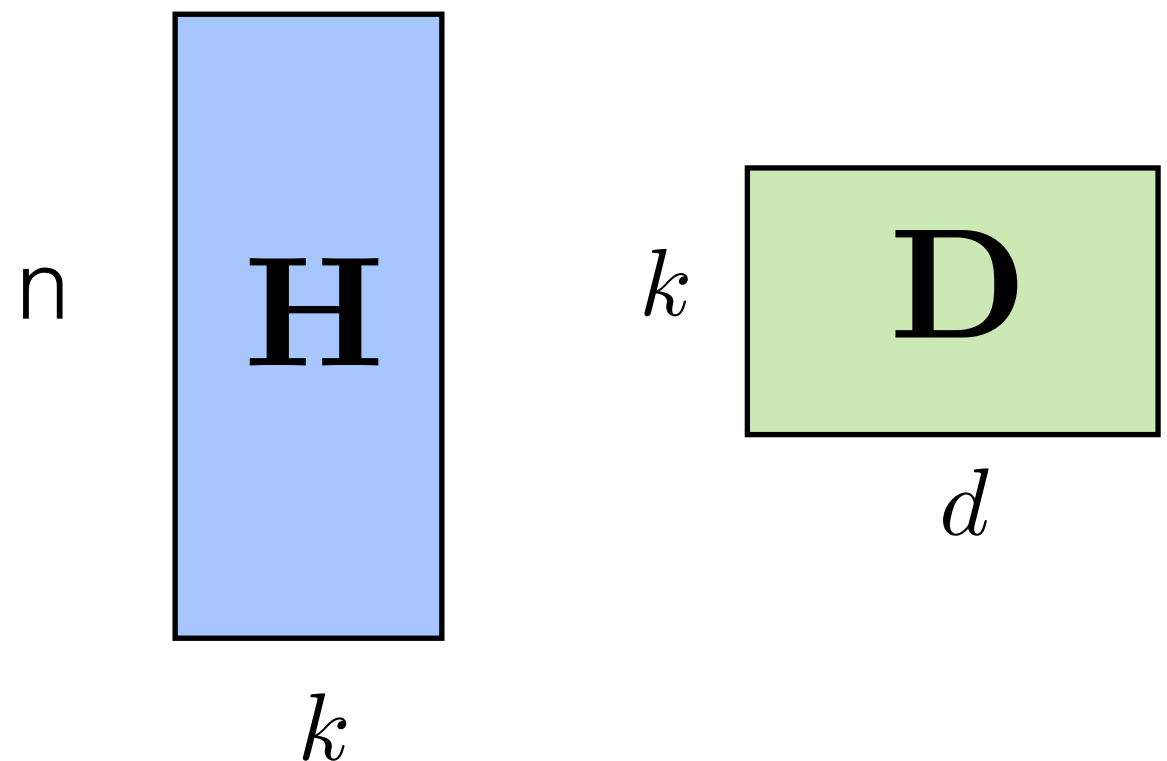
$\mathbf{D}_{:j}$ is the representation of movie j

Matrix completion

Example $\mathbf{H}_{i:} = [\text{like comedies}, \dots] = [1, \dots]$

$\mathbf{D}_{:j} = [\text{is a comedy}, \dots] = [1, \dots]$

$$\mathbf{H}_{i:} \mathbf{D}_{:j} = 1 + \dots$$



$\mathbf{H}_{i:}$ is the representation of user i

$\mathbf{D}_{:j}$ is the representation of movie j

How fill in missing data?

- The goal in factorization is to find $X = H D$
- This corresponds to finding $X_{ij} = H_{\{i,: \}} D_{\{:,j \}}$ for all i, j
- The $H_{\{i,: \}}$ is shared for i , across all j
- The $D_{\{:,j \}}$ is shared for j , across all i
- We can learn something about $H_{\{i,: \}}$, as long as X_{ij} available for some j
- We can learn something about $D_{\{:,j \}}$, as long as X_{ij} available for some i

Algorithm

$$\min_{H, D} \sum_{\text{available } (i,j)} (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2$$

$$\nabla_D \sum_{\text{available } (i,j)} (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2 = \sum_{\text{available } (i,j)} \nabla_D (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2$$

$$\nabla_{D_{:j}} (X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j})^2 = -2(X_{ij} - \mathbf{H}_{i:} \mathbf{D}_{:j}) \mathbf{H}_{i:}$$

Gradient descent on H and D until convergence

Matrix completion solution

- Once learn H and D , can complete the matrix
- Take entry (i,j) that was missing, compute $X_{ij} = H_{\{i:\}} D_{\{:j}}$
- $H_{\{i:\}}$ is the representation of user i
 - also called an embedding, where similar users in terms of movie preferences should have similar $H_{\{i:\}}$
- $D_{\{:j}}$ is the representation of movie j

Conclusion: factorization enables missing data to be inferred, and provides a new metric between items

$$\|\mathbf{H}_{i:} - \mathbf{H}_{s:}\|_2$$