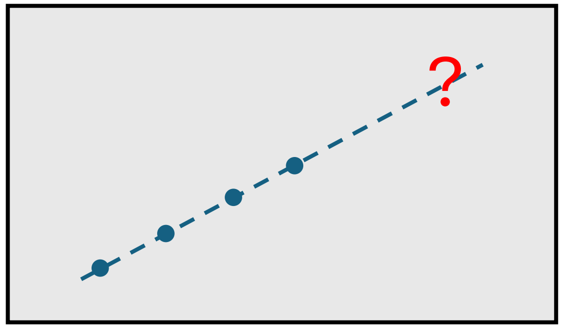
Lecture Three: How to Learn

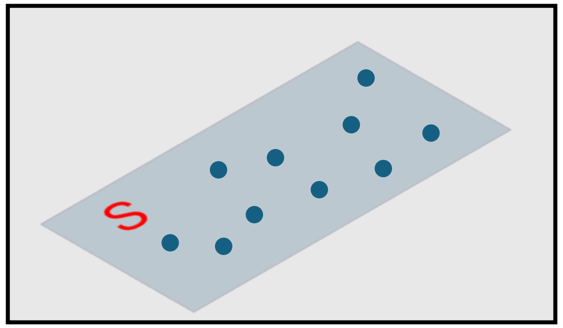
Yi Ma

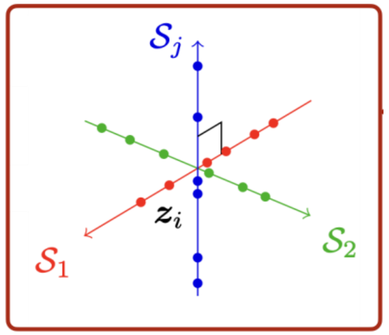
Director of the School of Computing and Data Science
Director of the Institute of Data Science

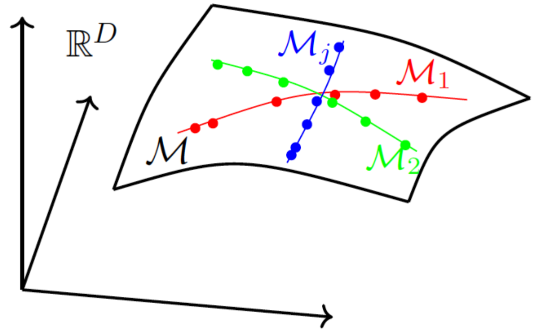
What to Learn? **Predictable information** from data sensed of the external world

Mathematically, all predictable information can be modeled as certain low-dimensional structures in the high-dimensional data







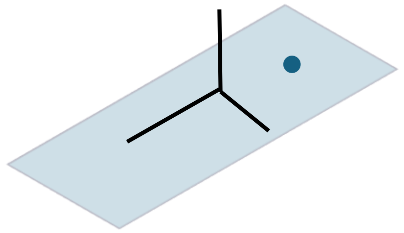


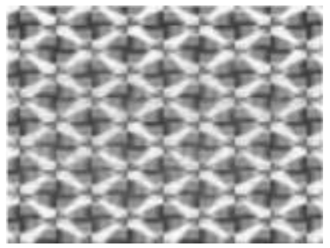
Computational complexity associated with realizing intelligence:
incomputable -> computable -> tractable -> scalable

Kolmogorov & Solomonoff	**Turing & Shannon**	**NP vs P**	**DNN and BP**

How to Learn? Pursuing parsimony

Data lie on a low-dim linear **subspace**





If we view the data (image) as a matrix

$$A = [\mathbf{a}_1 \mid \cdots \mid \mathbf{a}_n] \in \mathbb{R}^{m \times n}$$

then

$$r \doteq \operatorname{rank}(A) \ll m.$$

Principal Component Analysis (PCA) via singular value decomposition (SVD):

- Optimal estimate of A under iid Gaussian noise $D = A + Z$
- Efficient and scalable computation
- Fundamental statistical tool, with huge impact in practice...

How to Learn? Pursuing parsimony

Singular Value Decomposition (SVD)

Given $A \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(A) = r$, we like to decompose it into a special matrix form:

$$U_r = [\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r] \text{ orthogonal}$$

$$V_r = [\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_r] \text{ orthogonal}$$

$$\Sigma_r = \operatorname{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_r\} > 0$$

$$A = U_r \Sigma_r V_r^\top = [\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1^\top \\ \vec{v}_2^\top \\ \vdots \\ \vec{v}_r^\top \end{bmatrix}$$

How to Learn? Pursuing parsimony

Singular Value Decomposition (SVD)

Given $A \in \mathbb{R}^{m\times n}$ with $\mathrm{rank}(A) = r$, we like to decompose it into a special matrix form:

$$A = U_r \Sigma_r V_r^\top = [\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1^\top \\ \vec{v}_2^\top \\ \vdots \\ \vec{v}_r^\top \end{bmatrix}$$

Theorem: given $A \in \mathbb{R}^{m\times n}$ with $\mathrm{rank}(A) = r$, let $A^\top A = \sum_{i=1}^{r} \lambda_i \vec{v}_i \vec{v}_i^\top$ and $\sigma_i = \sqrt{\lambda_i}$, $\vec{u}_i = \frac{1}{\sigma_i} A\vec{v}_i \in \mathbb{R}^m,\ i = 1, \ldots, r$. Then we have $U_r = [\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r]$ orthogonal, and

$$A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top = U_r \Sigma_r V_r^\top \qquad \Sigma_r = \mathrm{diag}\{\sigma_1, \ldots, \sigma_r\} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_r \end{bmatrix}$$

How to Learn? Pursuing parsimony

Low-Rank Approximation: Eckart-Young Theorem

Approximate a matrix $A \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$ by a lower-rank matrix.

$$A = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n] = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top + \sum_{i=\ell+1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top \quad \text{with } \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$$

Theorem [Eckart-Young 1936]: The optimal solution to the low-rank approximation problem:

$$\min_{B \in \mathbb{R}^{m \times n}} \|A - B\|_F^2 \quad \text{subject to} \quad \text{rank}(B) = \ell$$

is given by: $B_\star = A_\ell = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top.$

How to Learn? Pursuing parsimony

Low-Rank Approximation: Rank Minimization

Approximate a matrix $A \in \mathbb{R}^{m\times n}$ with rank $r \leq \min\{m, n\}$ by a lower-rank matrix.

$$A = [\vec{a}_1, \vec{a}_2, \dots, \vec{a}_n] = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top + \sum_{i=\ell+1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top$$

Rank minimization problem:

$$\min_{B\in\mathbb{R}^{m\times n}} \ \operatorname{rank}(B) \quad \text{subject to} \quad \|A - B\|_F^2 \leq \epsilon^2\,?$$

How to Learn? Pursuing parsimony

Low-Rank Approximation: Model Selection

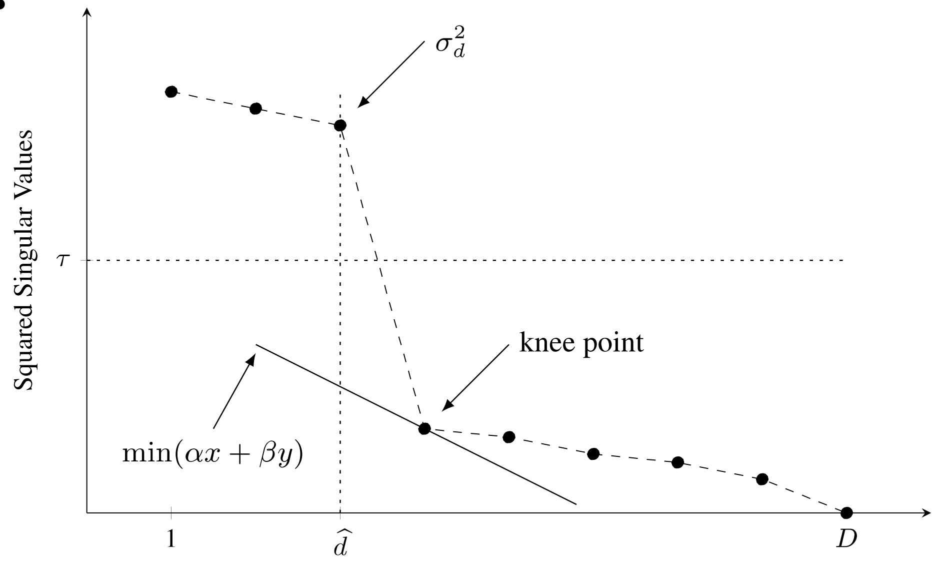
Approximate a matrix $A \in \mathbb{R}^{m\times n}$ with rank $r \leq \min\{m, n\}$ by a lower-rank matrix.

$$A = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n] = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^\top + \sum_{i=\ell+1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top$$

Selecting a good tradeoff between rank and residual:

1. $\min\limits_{B\in\mathbb{R}^{m\times n}} \ \text{rank}(B) = d \quad \text{subject to} \quad \sigma_{d+1}^2 \leq \tau?$

2. $\min\limits_{B\in\mathbb{R}^{m\times n}} \ \alpha \cdot \text{rank}(B) + \beta \cdot \sigma_{d+1}^2?$



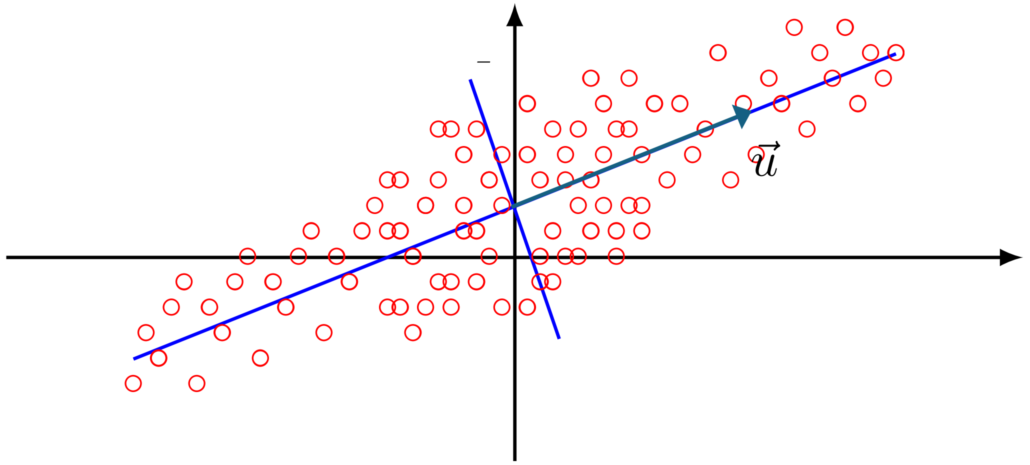
How to Learn? Pursuing parsimony

Principal Component Analysis (Statistics)

Problem [Pearson, 1901, Hotelling, 1933]**:** given

$$A = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n] \in \mathbb{R}^{m \times n} \qquad \vec{\mu} = \frac{1}{n}(\vec{a}_1 + \vec{a}_2 + \cdots + \vec{a}_n) = \mathbf{0}$$

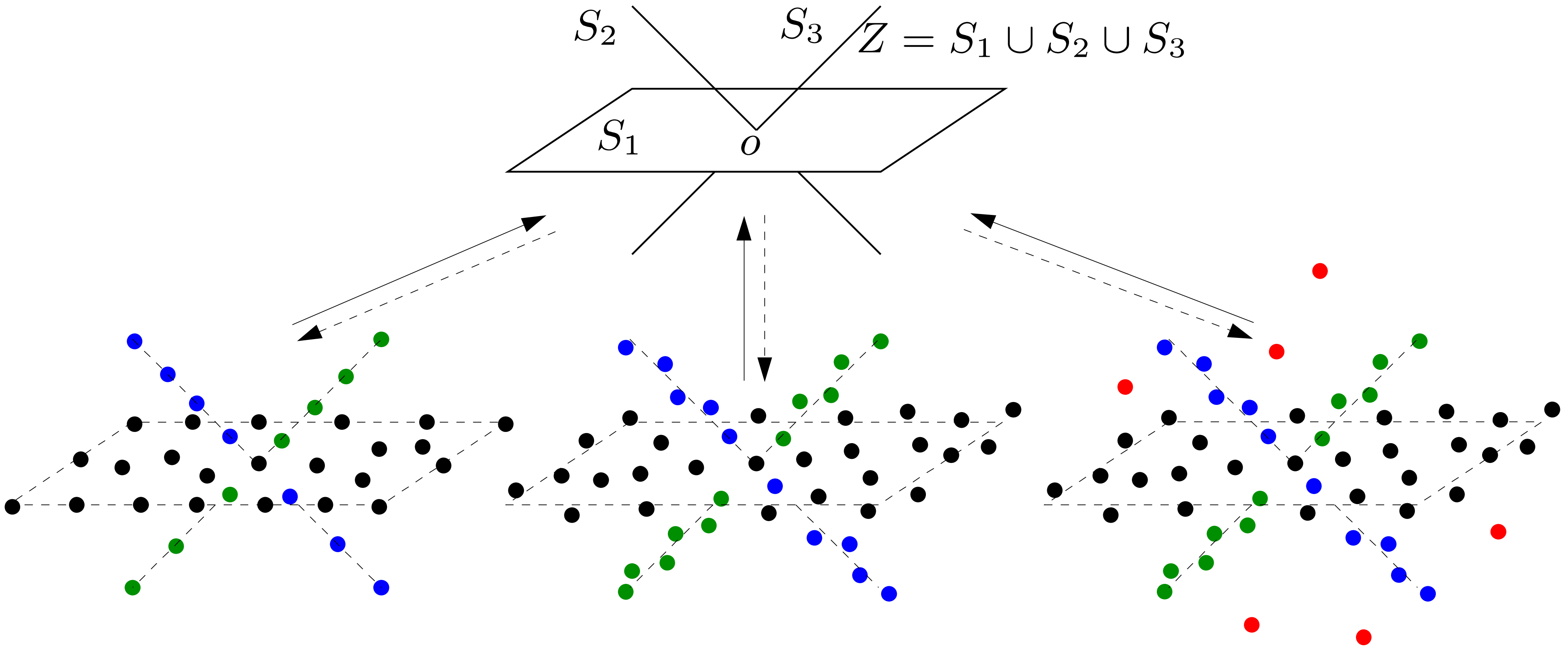
find a normal vector $\|\vec{u}\|_2 = 1$ such that $\max_{\vec{u}} \|\vec{u}^\top A\|_2^2 = \|\vec{u}\vec{u}^\top A\|_2^2.$



How to Learn? Pursuing low-dimensional models

Generalized Principal Component Analysis (GPCA):

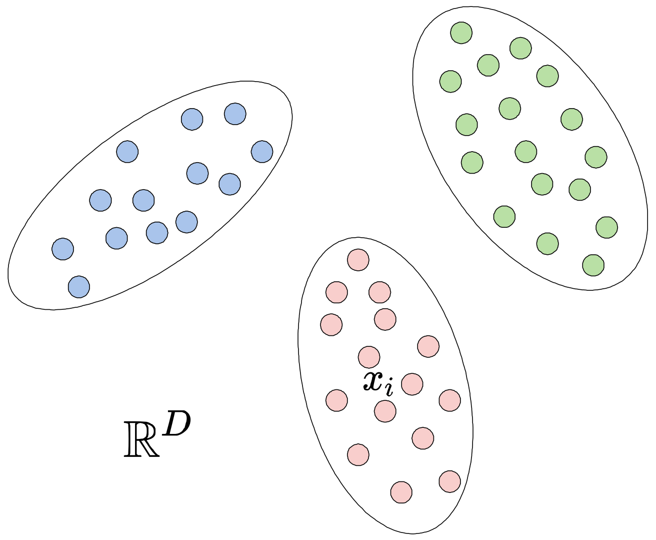
The data are on a mixture of subspaces



How to Learn? We compress to learn

1. Clustering Mixed Data (Interpolation)

Assume data $X = [x_1, x_2, \ldots, x_m]$ are i.i.d. samples from a mixture of distributions: $p(x, \theta) = \sum_{j=1}^{k} \pi_j p_j(x, \theta)$.



Classic approaches to cluster the data: the maximum-likelihood (ML) estimate via Expectation Maximization (EM):

$$\max_{\theta,\pi} \mathbb{E}\left[\log\left(\sum_{j=1}^{k} \pi_j p_j(x, \theta)\right)\right] \approx \max_{\theta,\pi} \frac{1}{m}\sum_{i=1}^{m} \log\left(\sum_{j=1}^{k} \pi_j p_j(x_i, \theta)\right).$$

Difficulties: ML is not well-defined when distributions are degenerate.

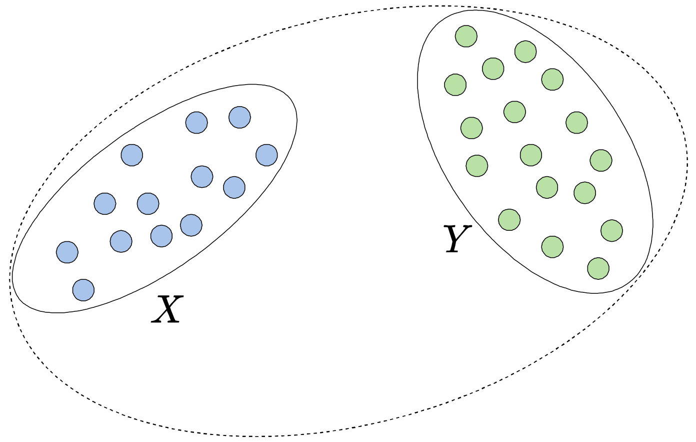
How to Learn? We compress to learn

Clustering via Compression

[Yi Ma, Harm Derksen, Wei Hong, and John Wright, TPAMI’07]

A Fundamental Idea:
Data belong to mixed low-dim structures should be compressible.

Cluster Criterion:
Whether the number of binary bits required to store the data is less (information gain):



$$\#\,\text{bits}(X\ [\ Y) \geq \#\,\text{bits}(X) + \#\,\text{bits}(Y)?$$

“The whole is greater than the sum of the parts.”
– Aristotle, 320 BC

How to Learn? We compress to learn

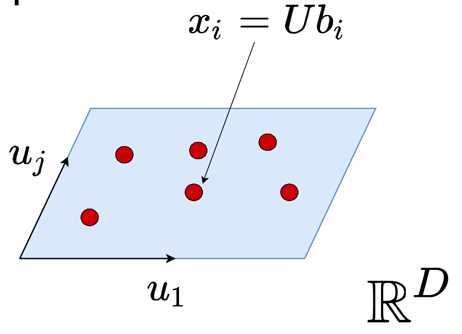
Theorem (Ma, TPAMI’07)

The number of bits needed to encode data $X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{D\times m}$ up to a precision $\|x - \hat{x}\|_2 \leq \epsilon$ is bounded by:

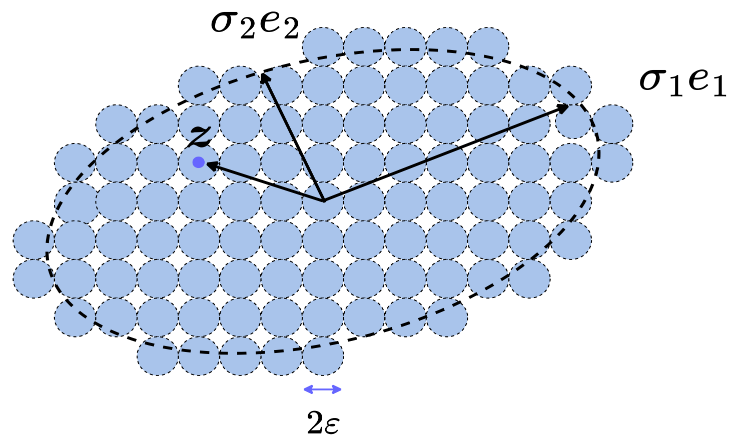
$$L(X, \epsilon) \doteq \left(\frac{m + D}{2}\right) \log\det\left(I + \frac{D}{m\epsilon^2} X X^\top\right).$$

This can be derived from constructively quantifying SVD of X or by sphere packing vol(X) as samples of a noisy Gaussian source.

Linear subspace



Gaussian

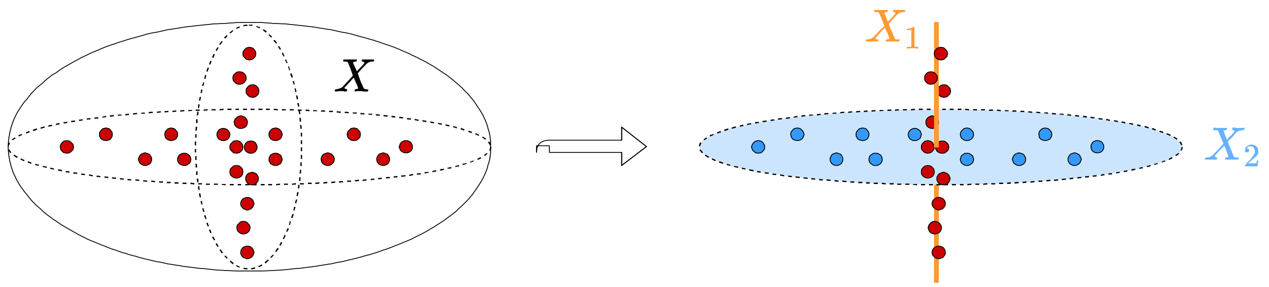


How to Learn? We compress to learn

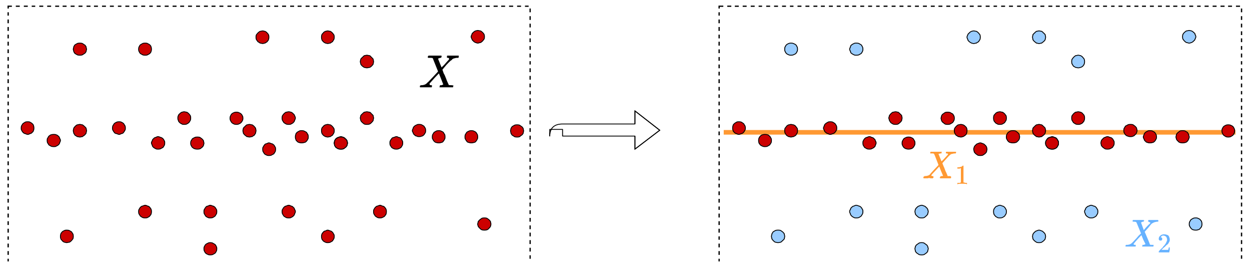
Cluster to Compress

$$L(X) \geq L^c(X) \doteq L(X_1) + L(X_2) + H(|X_1|, |X_2|)?$$

partitioning:



sifting:



How to Learn? We compress to learn

A Greedy Algorithm

Seek a partition of the data $\boldsymbol{X} \to [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_k]$ such that

$$\min L^c(\boldsymbol{X}) \doteq L(\boldsymbol{X}_1) + \cdots + L(\boldsymbol{X}_k) + H(|\boldsymbol{X}_1|, \ldots, |\boldsymbol{X}_k|).$$

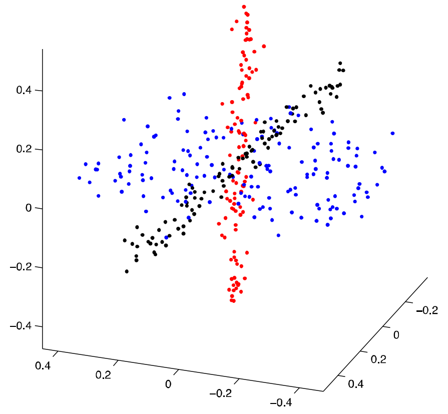
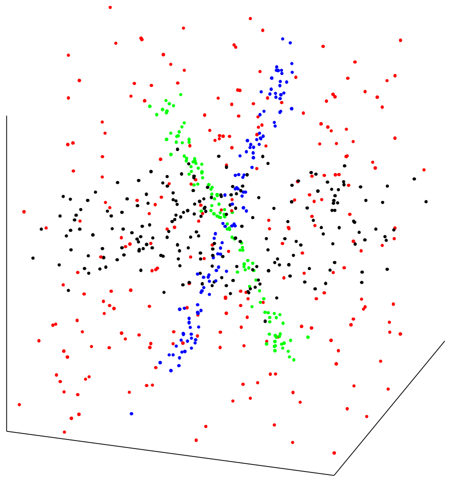
Optimize with a *bottom-up pair-wise* merging algorithm [Ma, TPAMI'07]:

```
1: input: the data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{D \times m}$ and a distortion $\epsilon^2 > 0$.
2: initialize $\mathcal{S}$ as a set of sets with a single datum $\{S = \{\boldsymbol{x}\} \mid \boldsymbol{x} \in \boldsymbol{X}\}$.
3: while $|\mathcal{S}| > 1$ do
4: choose distinct sets $S_1, S_2 \in \mathcal{S}$ such that
           $L^c(S_1 \cup S_2) - L^c(S_1, S_2)$ is minimal.
5: if $L^c(S_1 \cup S_2) - L^c(S_1, S_2) \geq 0$ then break;
6: else $\mathcal{S} := \big(\mathcal{S} \setminus \{S_1, S_2\}\big) \cup \{S_1 \cup S_2\}$.
7: end
8: output: $\mathcal{S}$
```

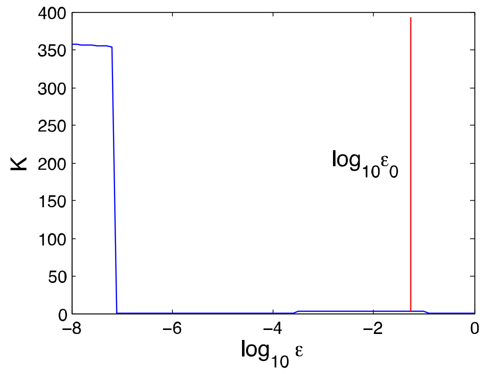
How to Learn? We compress to learn

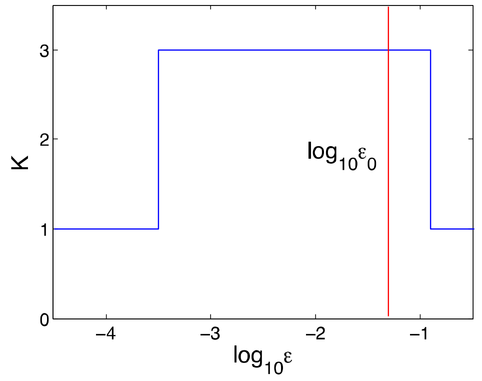
Surprisingly Good Performance

Empirically, **find global optimum and extremely robust to outliers**

A strikingly sharp **phase transition** w.r.t. quantization ϵ

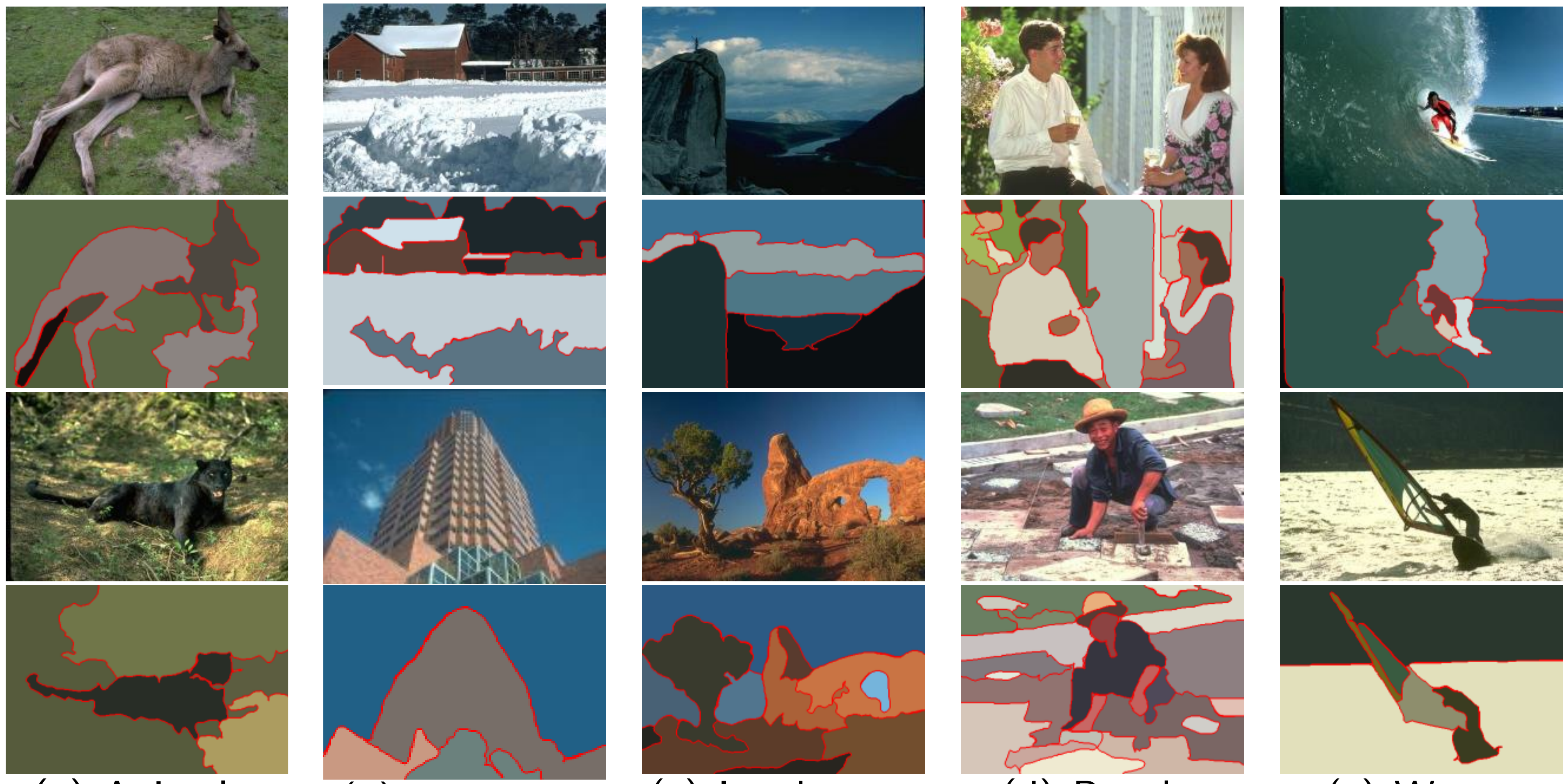




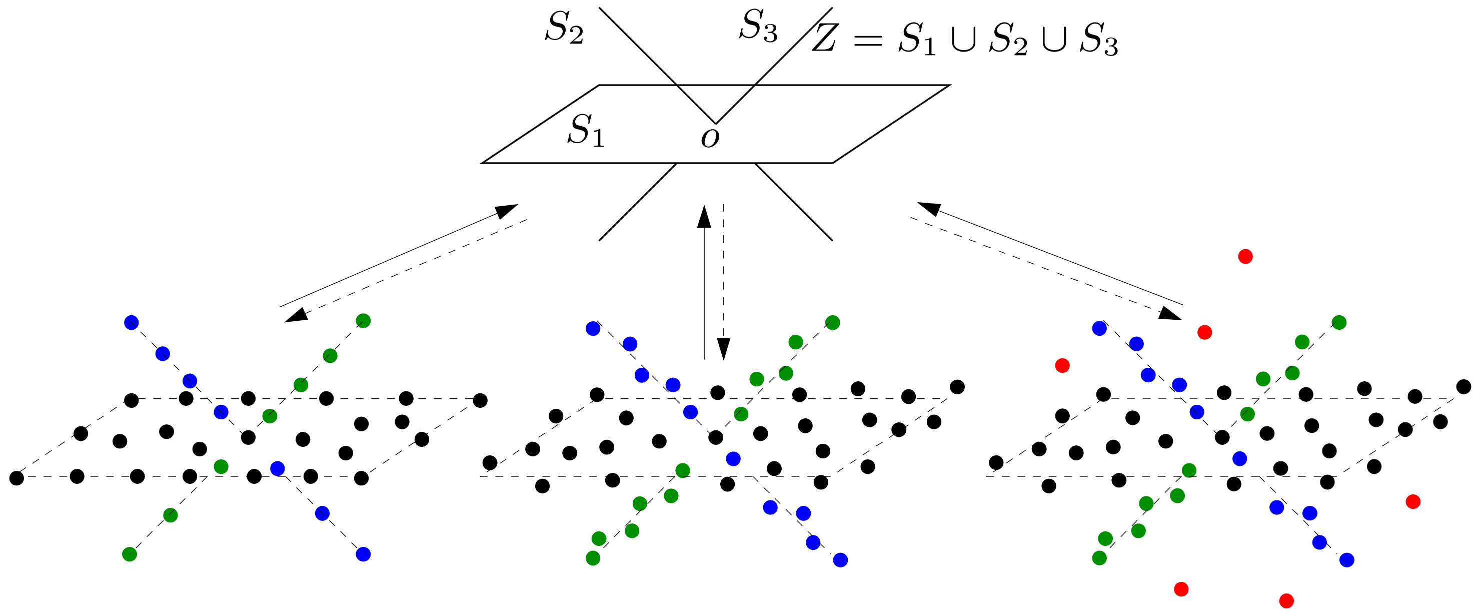
How to Learn? We compress to learn

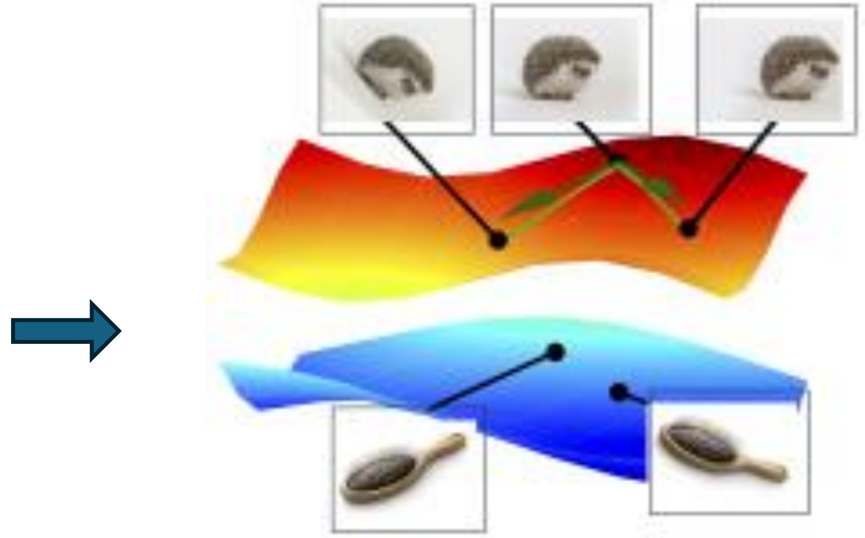
Natural Image Segmentation [Mobahi et.al., IJCV'09]

Compression alone, without any supervision, leads to **state of the art** segmentation on natural images (and many other types of data).


(a) Animals (b) Buildings (c) Landscape (d) People (e) Water

How to Learn a more general low-dim distribution?





Nonlinear Manifolds