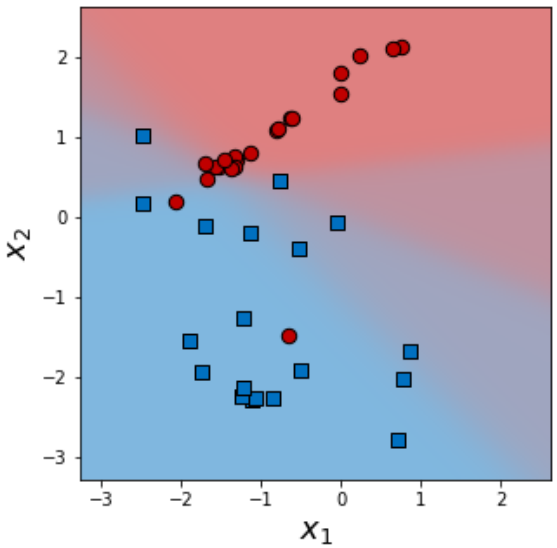
Photogrammetry & Robotics Lab

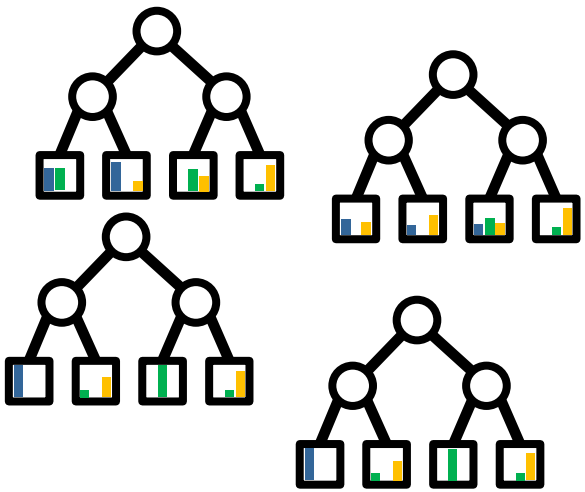
Machine Learning for Robotics and Computer Vision

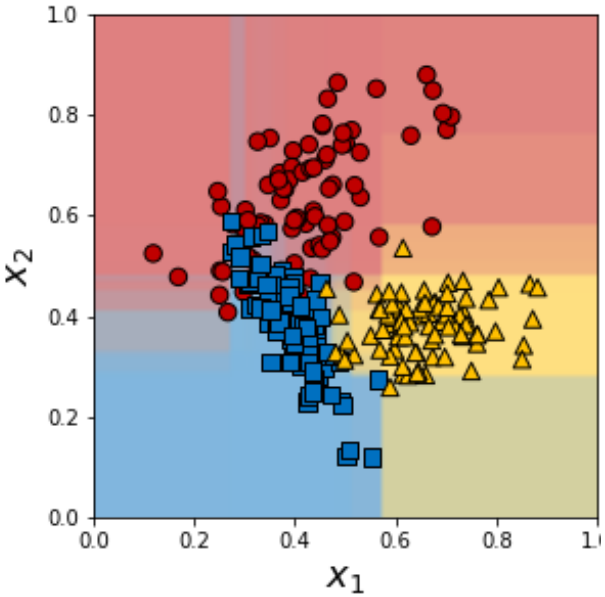
Unsupervised Learning

Jens Behley

Last Lecture

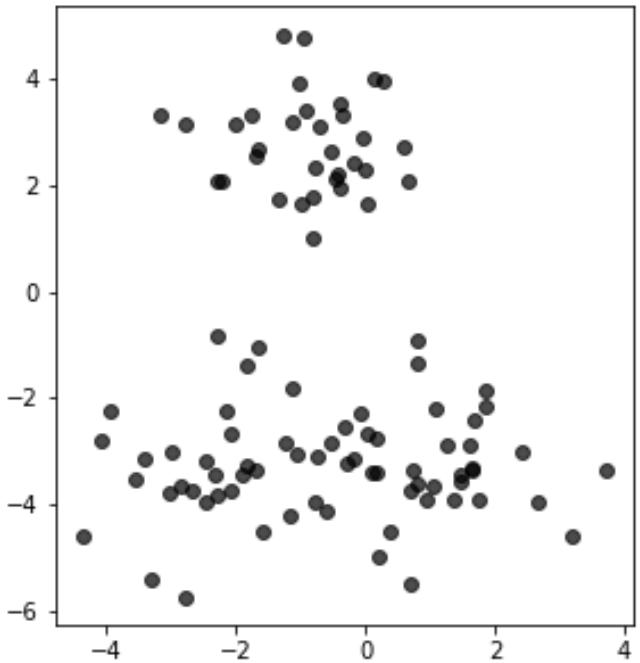






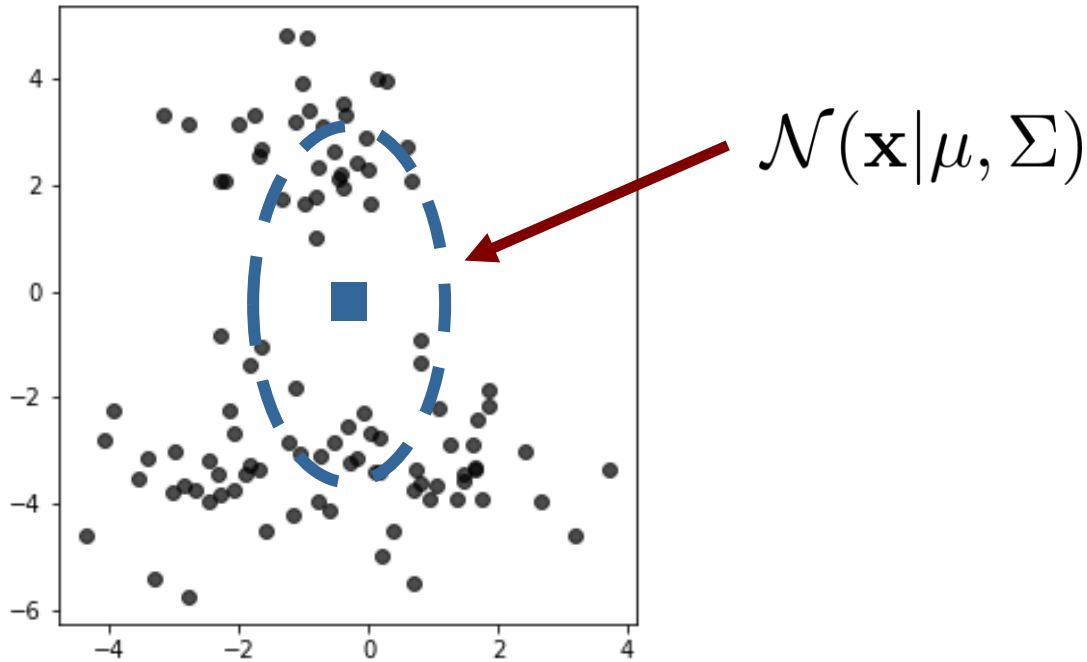
- Discussed ensemble methods:
  - Bagging
  - Random Forests
  - Boosting: AdaBoost and Gradient Tree Boosting
- Powerful off-the-shelf methods

Unsupervised Learning



- Unsupervised setting, we have data points without specific labels
- **Goal:** Learn a representation of the data that can be used to extract information or gain insights

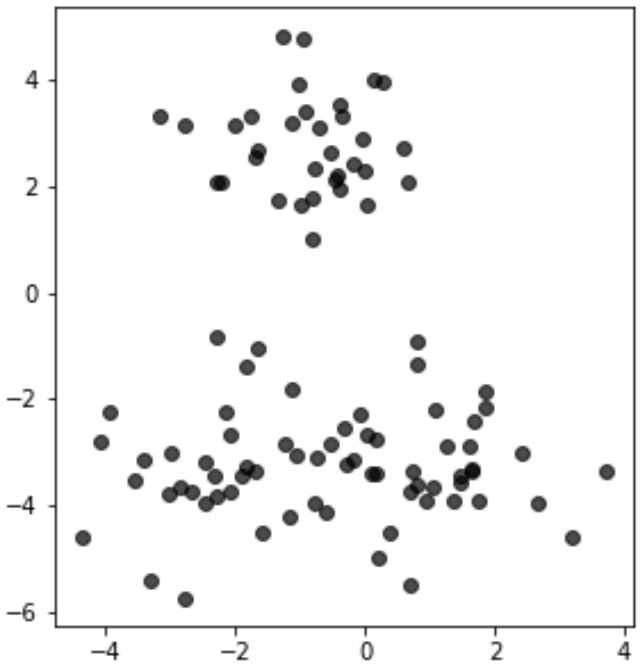
Density Estimation

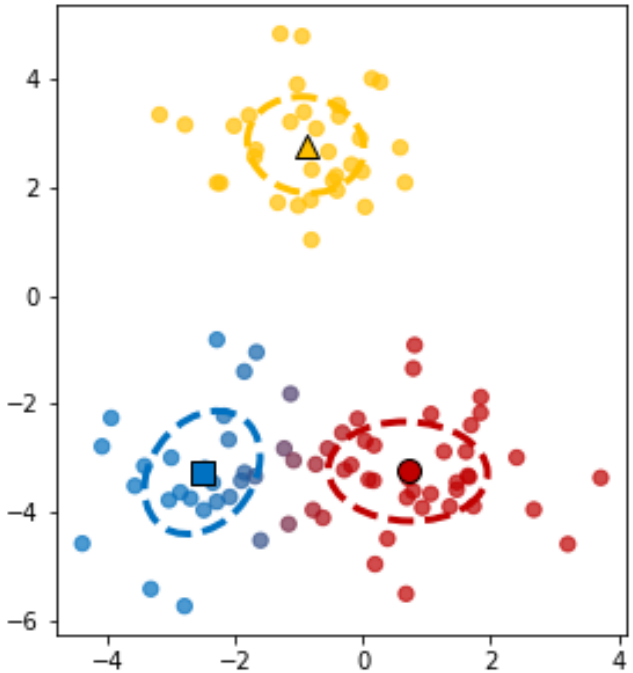


- In **density estimation**, we want determine a model for $P(\mathbf{x})$ of the given data

- Simple model with a single Gaussian cannot capture the structure of the data

Multiple Gaussians





- Now, we assume that clusters correspond to K Gaussians given by $\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$
- Each Gaussian models part of the data

[CVMLI Chap. 7] [MML Chap. 11]

Latent variables

- Assume that data can be decomposed in multiple parts by a **latent variable** h:

$$P(\mathbf{x}) = \sum_{i=0}^{K} P(h = i, \mathbf{x})$$

$$= \sum_{k=1}^{K} P(h = k) P(\mathbf{x} | h = k)$$

- For the **Gaussian Mixture Model (GMM)**:

$$P(h = k) = \lambda_k \qquad \text{with} \quad \sum_k \lambda_k = 1$$

$$P(\mathbf{x} | h = k) = \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

Gradient of Gaussian Mixture

- The log-likelihood takes the following form:

$$\theta^* = \arg\max_\theta \prod_{i=1}^N \sum_k P(h=k, \mathbf{x}_i)$$

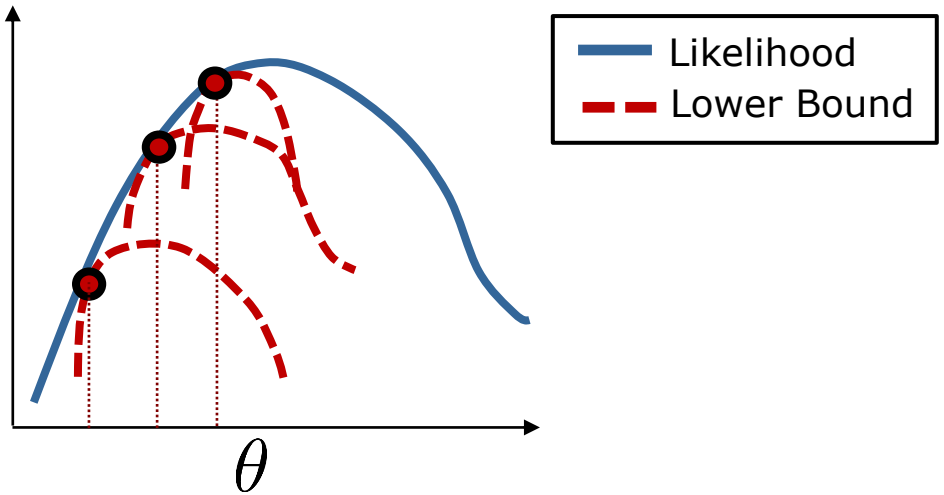
$$= \arg\max_\theta \log \prod_{i=1}^N \sum_k P(h=k, \mathbf{x}_i)$$

$$= \arg\max_\theta \sum_{i=1}^N \boxed{\log \sum_k P(h=k, \mathbf{x}_i)}$$

- Problem: To get parameters of $P(h)$ we need $P(\mathbf{x}|h)$ and vice versa.

[CVMLI Chap. 7] [MML Chap. 11]

Expectation Maximization



- Instead of maximizing directly the log-likelihood, we maximize a lower bound with the same parameters
- Alternate between getting a new lower bound (E-Step) and maximizing the lower bound (M-Step)

[CVMLI Chap. 7] [MML Chap. 11]

Finding the lower bound

- Adding the distribution $q_i(h)$, we can use Jensen's Inequality to get a lower bound:

$$\sum_{i=1}^{N} \log \sum_k \frac{q_i(h=k)}{q_i(h=k)} P(\mathbf{x}_i, h=k)$$

$$= \sum_{i=1}^{N} \log \sum_k q_i(h=k) \frac{P(\mathbf{x}_i, h=k)}{q_i(h=k)}$$

$$\geq \underbrace{\sum_{i=1}^{N} \sum_k q_i(h=k) \log \frac{P(\mathbf{x}_i, h=k)}{q_i(h=k)}}_{\mathcal{B}(q_i, \theta)}$$

Jensen's Inequality*

$$f\left(\sum_i \pi_i x_i\right) \geq \sum_i \pi_i f(x_i)$$

* f is concave

What is q?

- For finding $q_i(h)$, we rewrite one of the N terms in $\mathcal{B}(q_i, \theta)$:

$$\sum_k q_i(h=k) \log \frac{P(\mathbf{x}_i, h=k)}{q_i(h=k)}$$

$$= \sum_k q_i(h=k) \log \frac{P(h=k|\mathbf{x}_i) P(\mathbf{x}_i)}{q_i(h=k)}$$

$$= \underbrace{\sum_k q_i(h=k) \log \frac{P(h=k|\mathbf{x}_i)}{q_i(h=k)}}_{-\mathbb{KL}(q_i(h=k)||P(h=k|\mathbf{x}_i))} + \sum_k q_i(h=k) \log P(\mathbf{x}_i)$$

Kullback-Leibler (KL) Divergence

$$\mathbb{KL}(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

[CVMLI Chap. 7.8]

Lower Bound

- Thus, to get a tight bound $\mathcal{B}(q_i, \theta)$, we use

$$q_i(h = k) = P(h = k|\mathbf{x}_i)$$

- We compute $P(h = k|\mathbf{x}_i)$ as follows:

$$\begin{aligned} P(h = k|\mathbf{x}_i) &= \frac{P(h{=}k, \mathbf{x}_i)}{P(\mathbf{x}_i)} \\ &= \frac{P(h{=}k, \mathbf{x}_i)}{\sum_j P(h{=}j, \mathbf{x}_i)} \\ &= \frac{P(h{=}k)P(\mathbf{x}_i|h{=}k)}{\sum_k P(h{=}j)P(h{=}j, \mathbf{x}_i)} \end{aligned}$$

[CVMLI Chap. 7.8]

Expectation Maximization (EM)

- In EM algorithm, we optimize parameters iteratively as follows:
  1. E-Step: Update lower bound with respect to ***old*** parameters by determining $P(h = k|\mathbf{x}_i) =: r_{ki}$
  2. M-Step: Maximize lower bound with given r_{ki} to find updated ***new*** parameters.

- The responsibilities r_{ki} determine the influence of $\mathbf{x}_i$ on the parameter update.

[CVMLI Chap. 7] [MML Chap. 11]

M-Step

- In the M-Step, we maximize the bound:

$$\theta^{[t]} = \arg\max_\theta \sum_{i=1}^N \sum_k q_i(h=k) \log \frac{P(\mathbf{x}_i, h=k)}{q_i(h=k)}$$

$$= \arg\max_\theta \sum_{i=1}^N \sum_k r_{ik}^{[t-1]} \log P(h=k) P(\mathbf{x}_i | h=k) - \cancel{\log r_{ik}^{[t-1]}}$$

$$= \arg\max_\theta \sum_{i=1}^N \sum_k r_{ik}^{[t-1]} \left(\log \lambda_k^{[t]} + \log \mathcal{N}(\mathbf{x} | \mu_k^{[t]}, \Sigma_k^{[t]}) \right)$$

- For the Gaussian $\mathcal{N}(\mathbf{x} | \mu_k^{[t]}, \Sigma_k^{[t]})$, we already derived the maximum likelihood update; account for weights now:

$$\mu_k^{[t]} = \frac{\sum_{i=1}^N r_{ik}^{[t-1]} \mathbf{x}_i}{\sum_{i=1}^N r_{ik}^{[t-1]}} \qquad \Sigma_k^{[t]} = \frac{\sum_{i=1}^N r_{ik}^{[t-1]} \left(\mathbf{x}_i - \mu_k^{[t]}\right) \left(\mathbf{x}_i - \mu_k^{[t]}\right)^T}{\sum_{i=1}^N r_{ik}^{[t-1]}}$$

[CVMLI Chap. 7] [MML Chap. 11]

Updating the mixture weights

- For the maximum likelihood update of $\lambda_k^{[t]}$, we use a Lagrange multiplier:

$$\theta_h^{[t]} = \arg\max_\theta \sum_{i=1}^N \sum_k r_{ik}^{[t-1]} \log \lambda_k + \rho \left(\sum_k \lambda_k - 1 \right)$$

- Determining $\frac{\partial L}{\partial \lambda_j}$ and $\frac{\partial \mathcal{L}}{\partial \rho}$, setting this to zero, leads to the parameter update (see notes):

$$\lambda_j^{[t]} = \frac{1}{N} \sum_{i=1}^N r_{ij}^{[t-1]}$$

EM for Gaussian Mixture Model

- Algorithm for learning a GMM:

1. Initialize $\mu_k^{[0]}$ by selecting K random examples $\mathbf{x} \in \mathcal{X}$, $\Sigma_k^{[0]} = \mathbf{I}$, and $\lambda_k^{[0]} = K^{-1}$
2. E-Step: Determine $r_{ik}^{[t-1]}$ by computing:

$$r_{ki}^{[t-1]} = P(h = k|\mathbf{x}_i) = \frac{\lambda_k^{[t-1]} \mathcal{N}\left(\mathbf{x}_i | \mu_k^{[t-1]}, \Sigma_k^{[t-1]}\right)}{\sum_k \lambda_k^{[t-1]} \mathcal{N}\left(\mathbf{x}_i | \mu_k^{[t-1]}, \Sigma_k^{[t-1]}\right)}$$

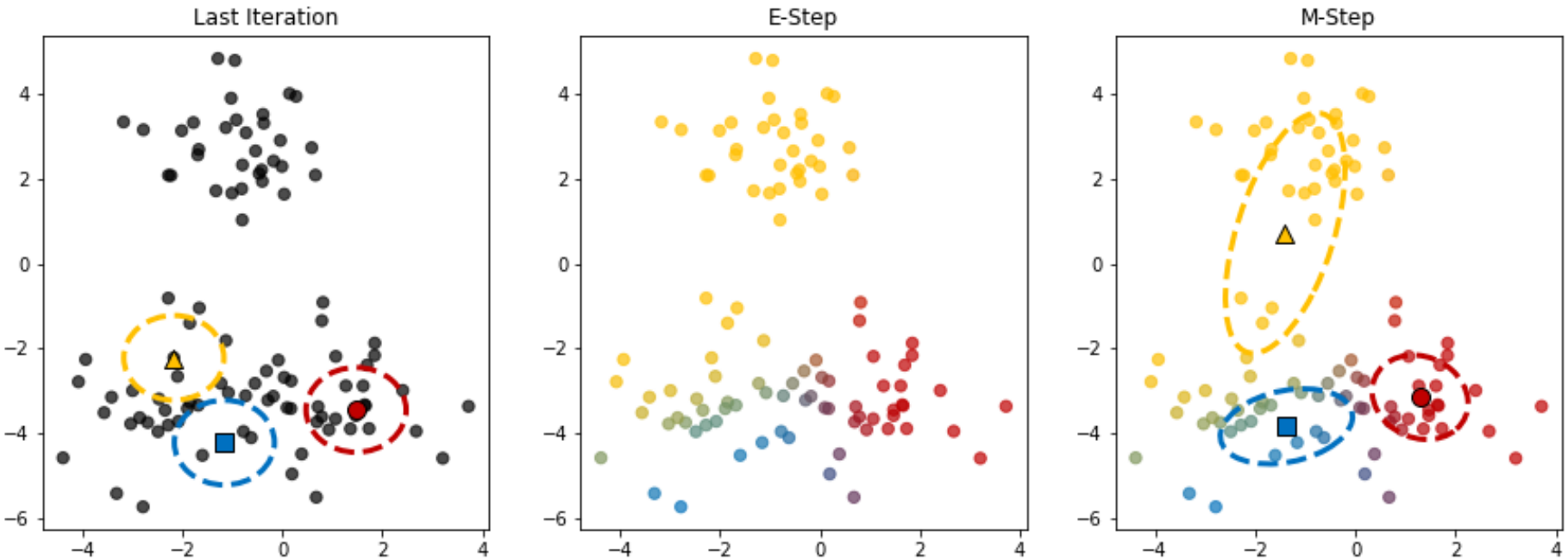
3. M-Step: Update parameters:

$$\lambda_k^{[t]} = \frac{1}{N} \sum_{i=1}^{N} r_{ik}^{[t-1]} \qquad \mu_k^{[t]} = \frac{\sum_{i=1}^{N} r_{ik}^{[t-1]} \mathbf{x}_i}{\sum_{i=1}^{N} r_{ik}^{[t-1]}}$$

$$\Sigma_k^{[t]} = \frac{\sum_{i=1}^{N} r_{ik}^{[t-1]} \left(\mathbf{x}_i - \mu_k^{[t]}\right)\left(\mathbf{x}_i - \mu_k^{[t]}\right)^T}{\sum_{i=1}^{N} r_{ik}^{[t-1]}}$$

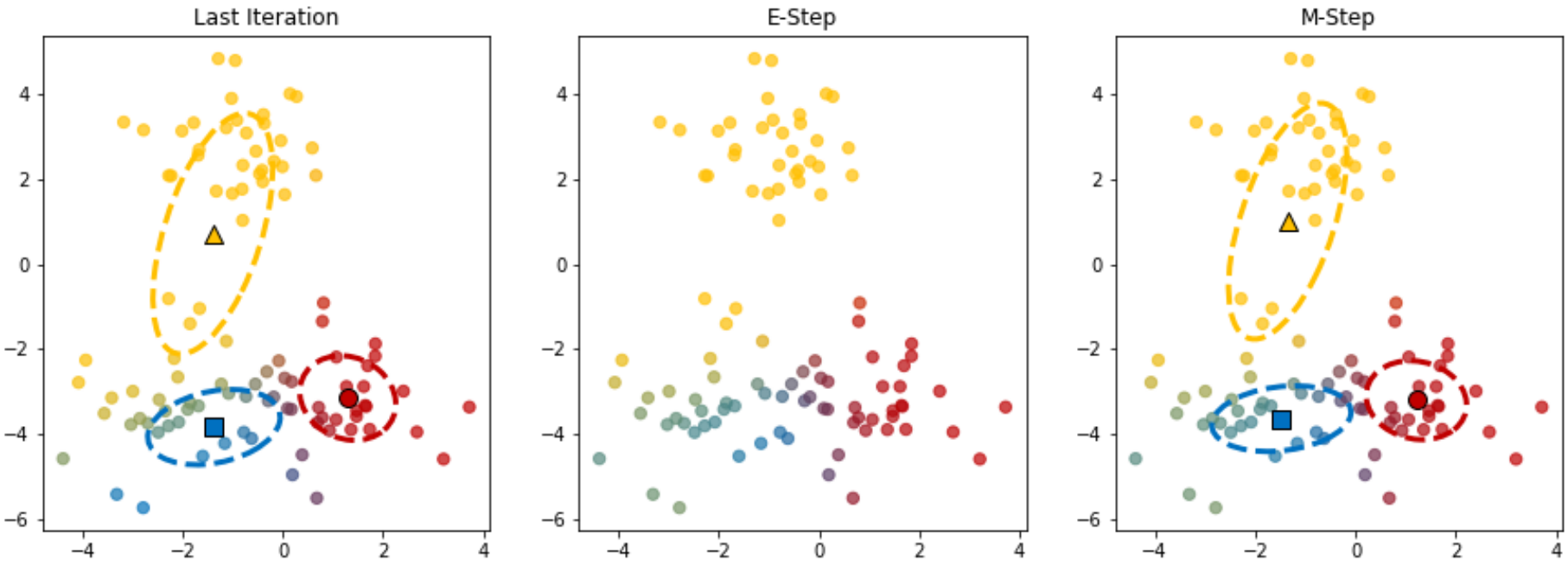
4. Repeat E-Step & M-Step until convergance.

Ex: Gaussian Mixture Model (1)



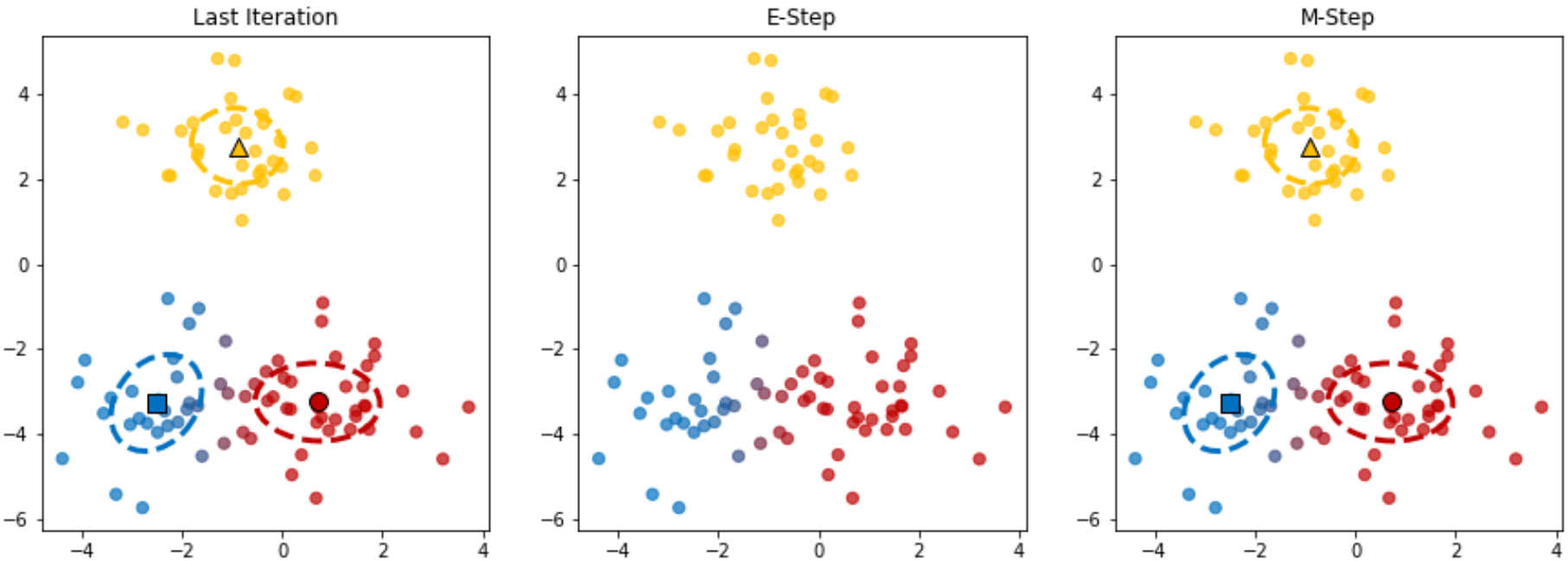
- Set K = 3 and initial means to random points.

Ex: Gaussian Mixture Model (2)



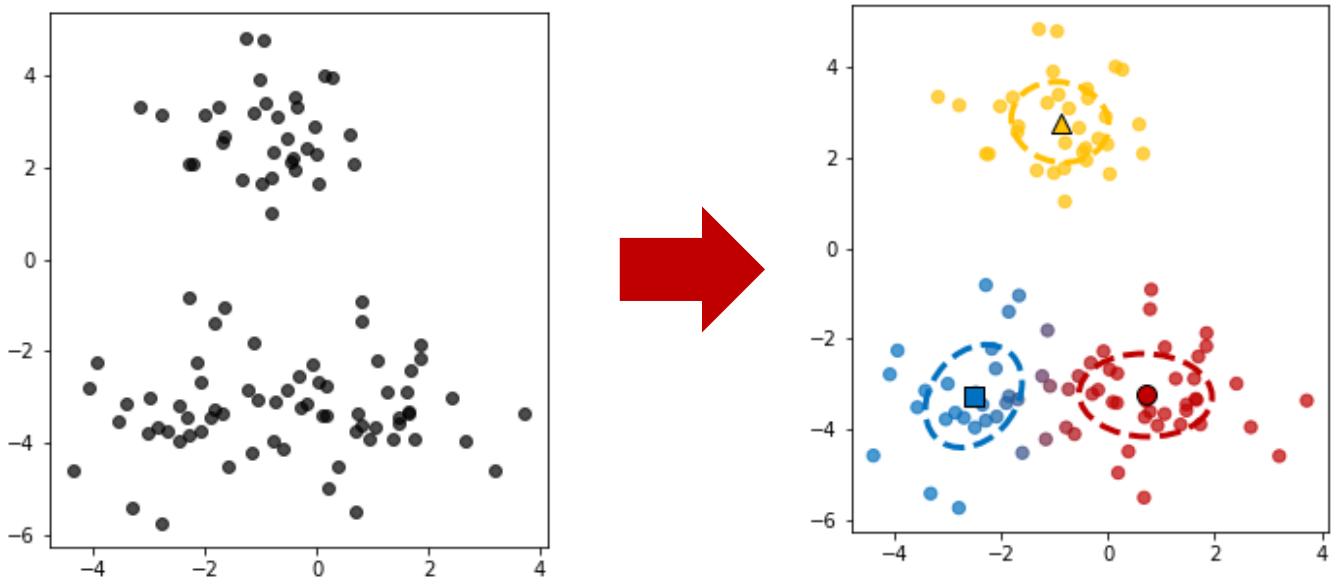
- Iterate until convergence.

Ex: Gaussian Mixture Model (3)



- Iterate until convergence.
- Here, 50 iterations until no changes.

Relation to Clustering



- Clustering aims at finding groups (clusters), where points are more similar inside the cluster than other clusters, e.g., distance, …
- Means μ_k correspond to cluster centers.
- Responsibilities r_{ki} are soft cluster assignments

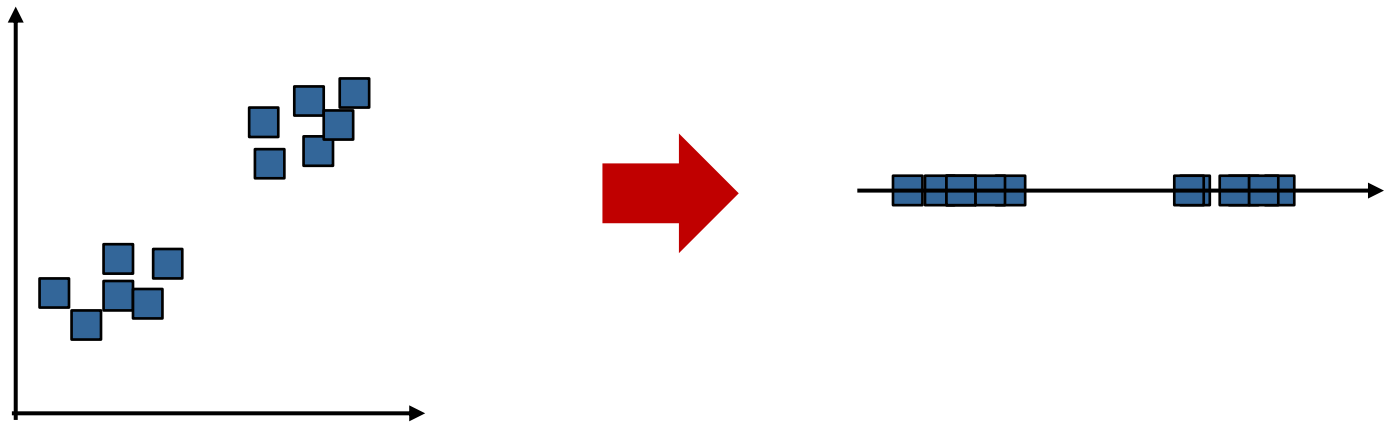
Relation to k-Means Clustering

- K-Means is special case with hard responsibilities, i.e., 0 or 1, and fixed Σ

- K-Means algorithm:
  1. Initialize means/cluster center randomly.
  2. Assign each point to nearest cluster center.
  3. Update cluster center by mean of assigned points

- Choice of initial cluster centers (more) important.

Dimensionality Reduction



- High-dimensional data hard to analyze, visualize, …
- **Goal:** Find low-dimensional representation that retains as much information as possible

[MML Chap. 10]

Projection

- We are looking for a matrix $\mathbf{B} \in \mathbb{R}^{D \times M}$ that projects points $\mathbf{x} \in \mathbb{R}^D$ to lower dimensional vectors $\mathbf{z} \in \mathbb{R}^M$, where $M < D$:

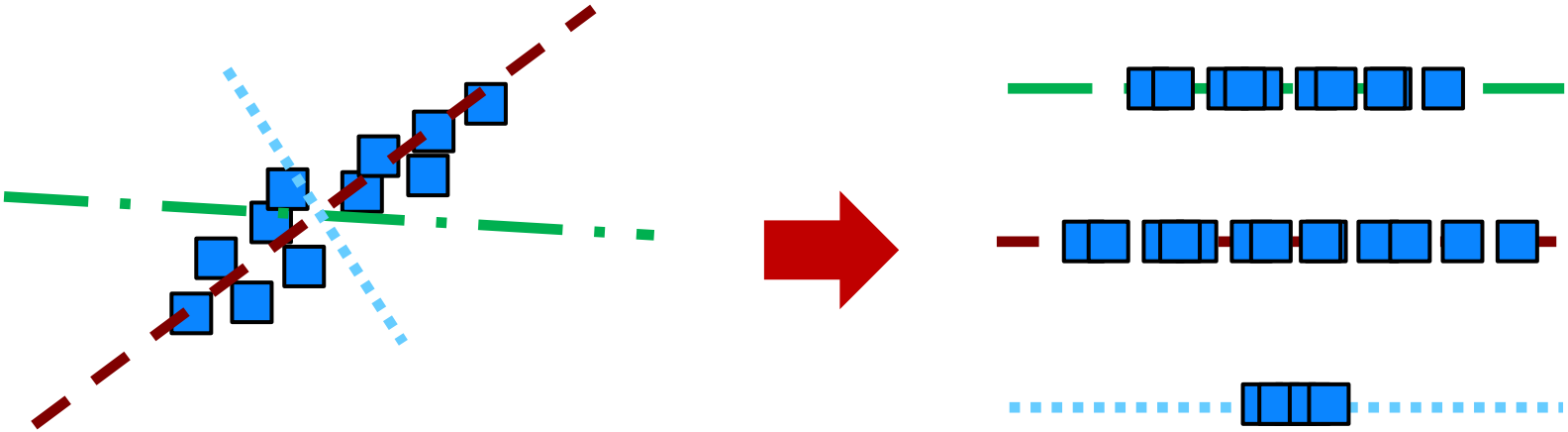
$$\mathbf{z} = \mathbf{B}^T \mathbf{x}$$

- We assume that $\mathbf{B}$ is orthonormal, i.e., columns $\mathbf{b}_i \in \mathbb{R}^D$ fulfill the properties:

$$\mathbf{b}_i^T \mathbf{b}_j = 0, i \neq j \quad \text{and} \quad \mathbf{b}_i^T \mathbf{b}_i = 1$$

[MML Chap. 10]

Maximum Variance Perspective



- Want to find projection that retains variance in each dimension as much as possible
- **Goal:** Encoded vectors $\mathbf{z} \in \mathbb{R}^M$ should maximize the variance in each dimension

[MML Chap. 10]

Finding Maximal Variance Direction of Projection

- For the projection, we now assume zero centered data, i.e., $\mu = N^{-1} \sum_{i=1}^{N} \mathbf{x} = \mathbf{0}$.
- Projection of first dimension with $\mathbf{b}_1 \in \mathbb{R}^D$

$$z_{1i} = \mathbf{b}_1^T \mathbf{x}_i$$

- Variance of projection given by:

$$V_1 = \frac{1}{N} \sum_{i=1}^{N} z_{1i}^2 = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{b}_1^T \mathbf{x}_i)^2$$

[MML Chap. 10]

Deriving the Objective

- Rearranging terms leads to

$$V_1 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{b}_1^T\mathbf{x}_i)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathbf{b}_1^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{b}_1$$

$$= \mathbf{b}_1^T\underbrace{\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T\right)}_{=:\,\mathbf{S}}\mathbf{b}_1$$

data covariance

- Therefore, we want to optimize:

$$\max_{\mathbf{b}_1}\ \mathbf{b}_1^T\mathbf{S}\mathbf{b}_1$$

$$\text{s.t. } \mathbf{b}_1^T\mathbf{b}_1 = 1$$

Principal Component

- Finding the maximum, we can use a Lagrange multiplier:

$$\mathcal{L} := \mathbf{b}_1^T \mathbf{S} \mathbf{b}_1 + \lambda_1 (1 - \mathbf{b}_1^T \mathbf{b}_1)$$

- With the partial derivatives:

$$\frac{\mathcal{L}}{\partial \mathbf{b}_1} = 2\mathbf{b}_1^T \mathbf{S} - 2\lambda_1 \mathbf{b}_1^T \qquad \frac{\mathcal{L}}{\partial \lambda_1} = 1 - \mathbf{b}_1^T \mathbf{b}_1$$

- Setting these to zero, we get the following:

$$\begin{aligned} \mathbf{S}\mathbf{b}_1 &= \lambda_1 \mathbf{b}_1 \\ \mathbf{b}_1^T \mathbf{b}_1 &= 1 \end{aligned}$$

Eigenvalue problem

- Thus, basis vector is the Eigenvector to the largest Eigenvalue (**principle component**)

Finding remaining basis vectors

- Remaining columns $\mathbf{b}_2, \ldots, \mathbf{b}_m$ can be derived similarly
- $\mathbf{b}_2, \ldots, \mathbf{b}_m$ correspond to eigenvectors of data covariance $\mathbf{S}$

- Therefore, finding projection is finding M eigenvectors of largest eigenvalues

- Maximum variance captured by M principle components:

$$V_M = \sum_{k=1}^{M} \lambda_k$$

[MML Chap. 10]

Principal Components Analysis

- Algorithm for PCA
  1. **Standardize** the data, i.e., subtract mean and divide each dimension by it's variance
  2. Determine Eigenvectors and Eigenvalues to get basis $\mathbf{B} \in \mathbb{R}^{D \times M}$

  - Projected points are then given by

    $$\tilde{\mathbf{x}}_* = \mathbf{B}\mathbf{B}^T \mathbf{x}_*$$

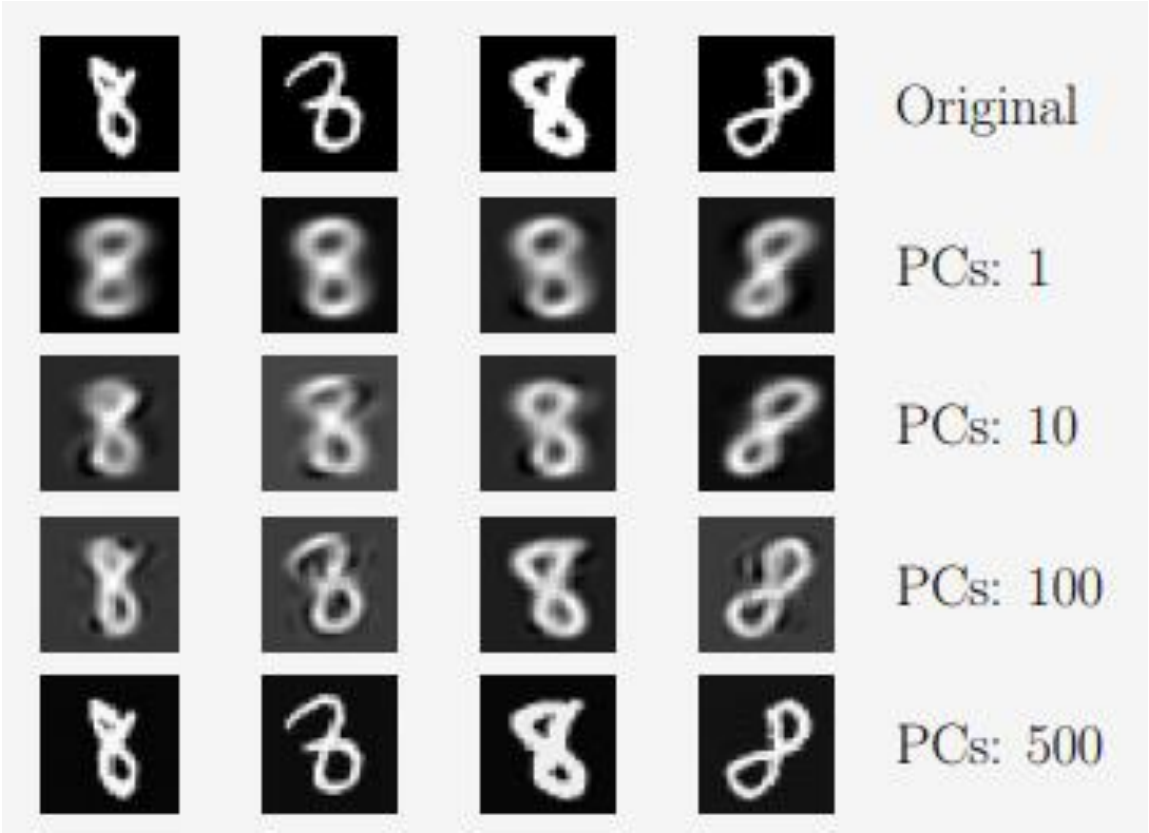
    where

    $$x_*^{(d)} = \frac{x^{(d)} - \mu^{(d)}}{\sigma_d}$$

  - Reconstruction in original space by de-standardization of $\tilde{\mathbf{x}}_*$

[MML Chap. 10]

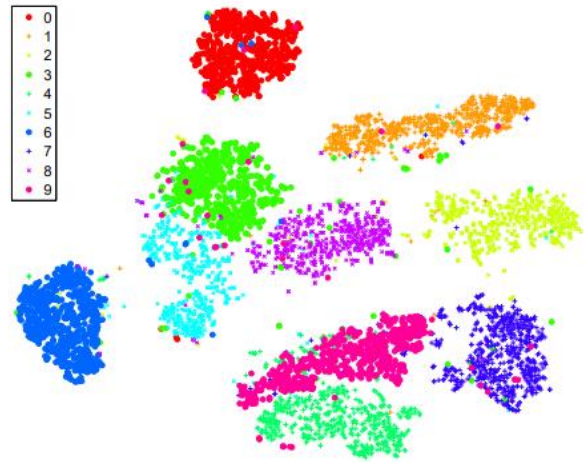
Example



- Reconstructions with increasing number of principal components (PCs)

Visualizing high-dimensional data

MNIST dataset
28x28 Images

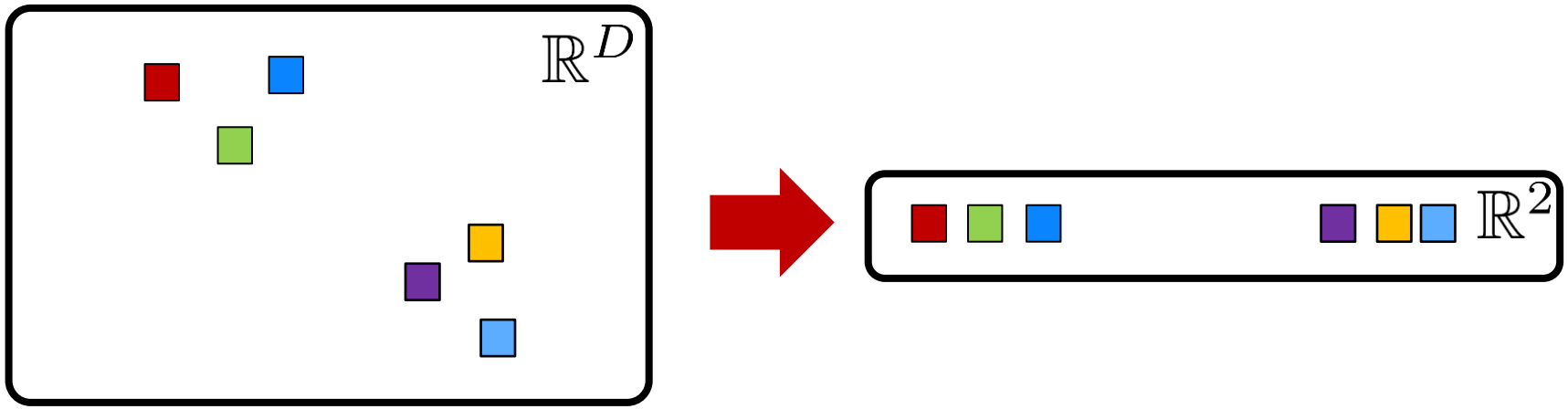


Low-dimensional Embedding
via t-SNE

- Here, want to find low-dimensional embedding that retains distances of high-dimensional data

[van der Maaten, 2008]

Distance Preserving Embedding



- **Task:** Find for each $\mathbf{x}_i \in \mathbb{R}^D$, an embedding vector $\mathbf{y}_i \in \mathbb{R}^2$ that preserves the structure of high dimensional data.

Stochastic Neighbor Embedding

- Convert Euclidean distances in original space into conditional prob. (= affinities):

$$P_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/(2\sigma_i^2))}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/(2\sigma_i^2))} \qquad (P_{i|i} = 0)$$

- For low-dimensional embeddings, we have:

$$Q_{j|i} = \frac{\exp(-||\mathbf{y}_i - \mathbf{y}_j||^2)}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||^2)} \qquad (Q_{i|i} = 0)$$

- SNE minimizes KL-divergence for all points:

$$\mathbf{y}_1, \ldots, \mathbf{y}_N = \arg\min \sum_{i=0}^{N} \underbrace{\sum_j P_{j|i} \log \frac{P_{j|i}}{Q_{j|i}}}_{\mathbb{KL}(P_{j|i}||Q_{j|i})}$$

[van der Maaten, 2008]

Determining variances

- For each data point, we have to determine a variance σ_i
- Want to have smaller σ_i in dense regions than in sparse regions

- In t-SNE, σ_i is determined via binary search to match desired perplexity π :

$$\sigma_i = \arg\min_\sigma |\mathrm{Perp}(P_i) - \pi|$$

Entropy

with $\mathrm{Perp}(P_i) = 2^{-\sum_j P_{j|i} \log_2 P_{j|i}}$

[van der Maaten, 2008]

From SNE to t-SNE

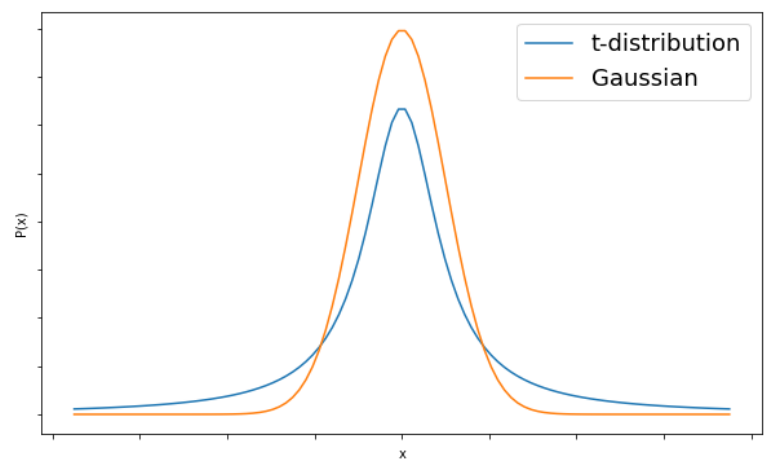
t-SNE improves upon SNE by following modifications:

1. Compute symmetric affinities:

$$P_{ij} = \frac{P_{i|j} + P_{j|i}}{2N}$$

2. Use Student's t-distribution for low-dimensional affinities:

$$Q_{ij} = \frac{(1+||y_i - y||^2)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||^2)^{-1}}$$



[van der Maaten, 2008]

Optimization in t-SNE

- As before, we minimize the KL-divergence:

$$C := \mathbb{KL}(P||Q) = \sum_{i=1}^{N} \sum_j P_{ij} \log_2 \frac{P_{ij}}{Q_{ij}}$$

- Using the gradient $\frac{\partial C}{\partial \mathbf{y}_i}$, we can optimize this via gradient descent:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (P_{ij} - Q_{ij})(\mathbf{y}_i - \mathbf{y}_j) \left(1 + ||y_i - y_j||^2\right)^{-1}$$

[van der Maaten, 2008]

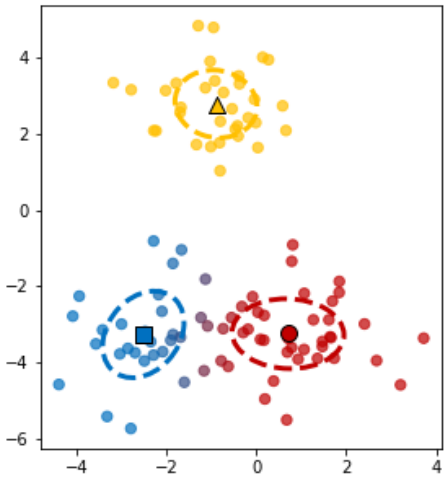
t-SNE Algorithm

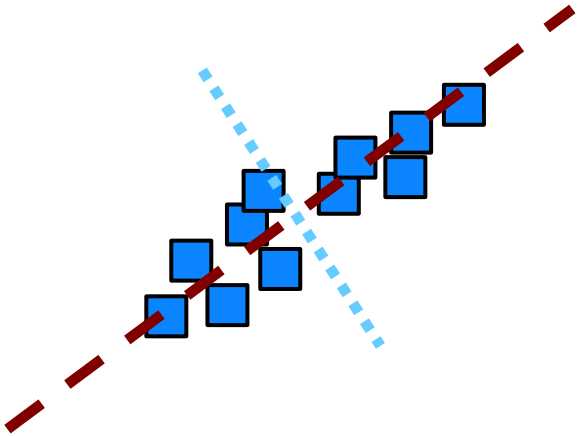
- Hyperparameters: perplexity π, number of iterations T, learning rate η, momentum $\alpha(t)$

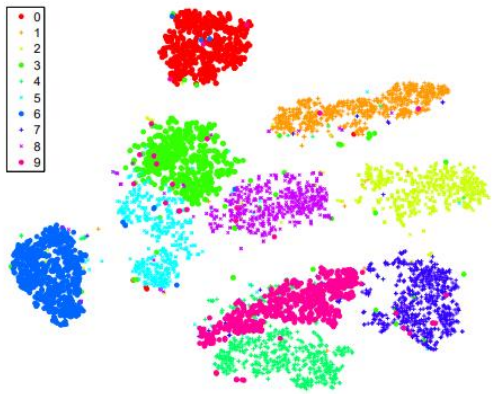
1. Compute pairwise affinities $P_{j|i}$ with desired perplexity π
2. Sample initial solution $\mathbf{y}_i \sim \mathcal{N}(x|0, 10^{-4}\mathbf{I})$
3. For $t = 1, \ldots, T$ do:
   1. Compute low-dimensional affinities Q_{ij}
   2. Compute gradient $\frac{\partial C}{\partial \mathbf{y}_i}$
   3. Update $\mathbf{y}_i^{[t]} = \mathbf{y}_i^{[t-1]} + \eta \frac{\partial C}{\partial \mathbf{y}_i} + \alpha(t)\left(\mathbf{y}_i^{[t-1]} - \mathbf{y}_i^{[t-2]}\right)$

[van der Maaten, 2008]

Summary







- Discussed several unsupervised learning approaches solving different tasks:
  - Density Estimation (Gaussian Mixture Models)
  - Dimensionality Reduction (PCA)
  - Visualization (t-SNE)

References

- L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. Journal of Machine Learning Research (JMLR), vol. 9, pp. 2579-2605, 2008.
- Wattenberg et al. How to Use t-SNE Effectively, Distill, 2016.

See you next week!