中国科学院大学人工智能学院硕士课《模式识别》

数据聚类
Data Clustering

张煦尧 (xyz@nlpr.ia.ac.cn)

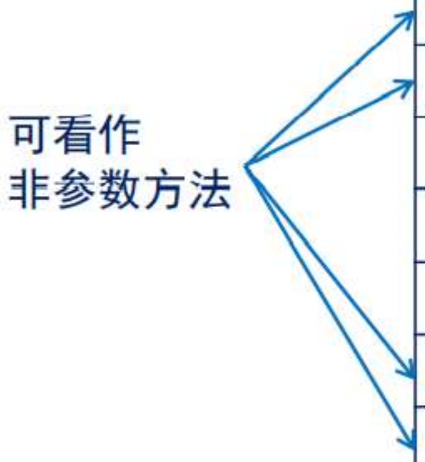
2022年11月30日

中国科学院大学
University of Chinese Academy of Sciences

本课程内容体系

12章，18次授课

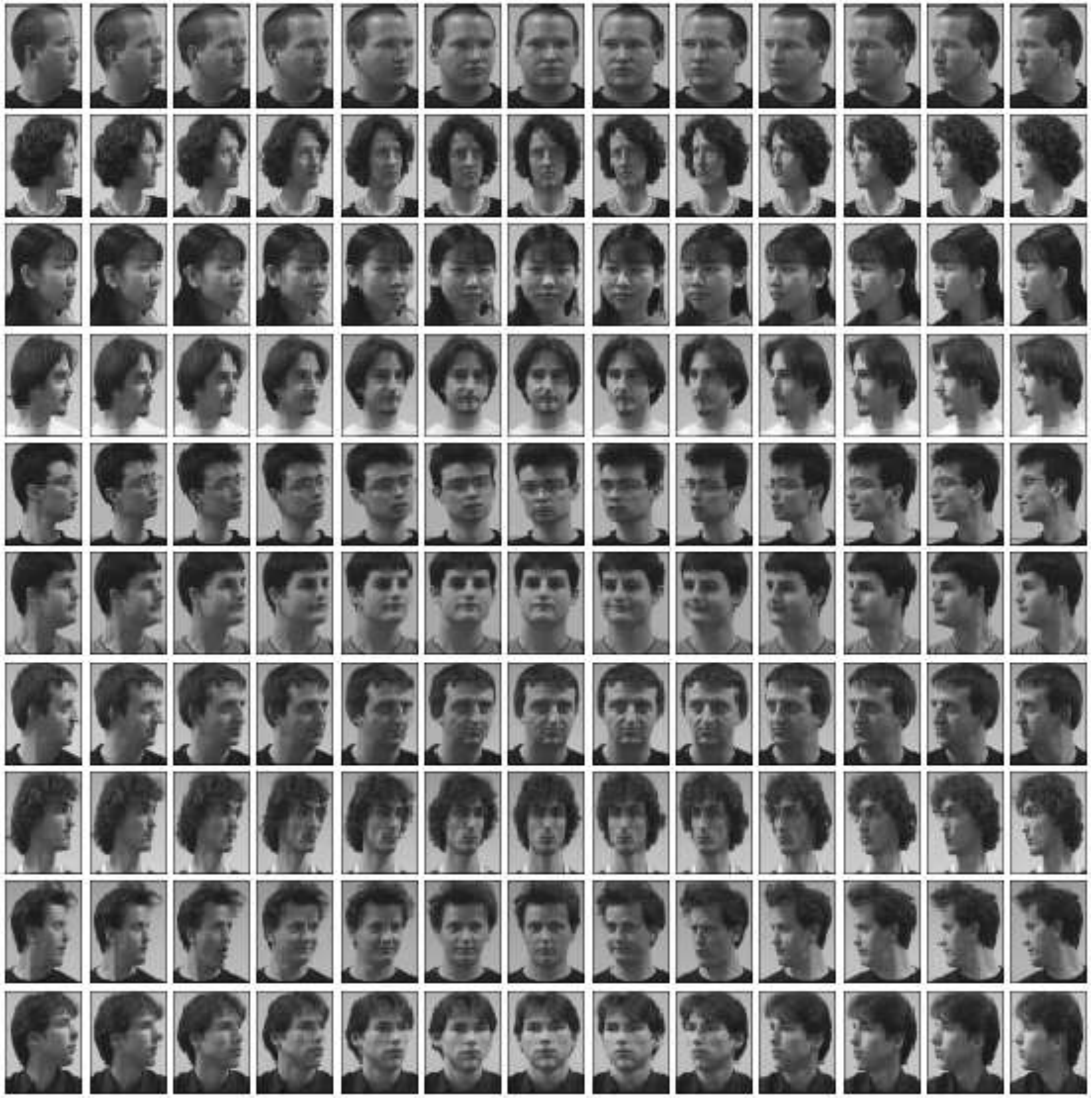
内容	课时	授课老师
绪论	5*3h	刘成林
贝叶斯决策理论		
概率密度参数估计		
非参数法		
线性分类器设计	7*3h	向世明
神经网络和深度学习		
特征提取与选择		
模型选择(DHS第9章)		
聚类分析(非监督学习)	5*3h	张煦尧
支撑向量机与核方法		
决策树方法(DHS第8章)		
模式识别前沿趋势	1*3h	刘成林



可看作
非参数方法

引言

- **聚类**
  - **物以类聚，人以群分。**
  - **将数据分成多个类别，在同一个类内，对象（实体）之间具有较高的相似性，不同类对象间差异性较大。**
  - 对一批没有类别标签的样本集，按照样本之间的相似程度分类，相似的归为一类，不相似的归为其它类。这种分类称为**聚类分析**，也称为无监督分类。
  - 聚类的质量(或结果)取决于对度量标准的选择。
  - 聚类结果**因不同任务而不同**。



身份识别 vs 姿态估计

引言

- **聚类任务**
  - 给定一个样本集合 X，给定一种度量样本间相似度或者相异度（距离）的标准。聚类系统的输出是关于样本集 X 的一个划分，即 $D = \{D_1 \cup D_2 \cup _{\dots} \cup D_k\}$。其中，$D_i$ (i=1,2,…,k)是 X 的一个子集，且满足：
    - $D_1 \cup D_2 \cup _{\dots} \cup D_k = X$
    - $D_i \cap D_j = \varnothing,\quad i \neq j$
  - D 中成员 D_1 , D_2 ,…, D_k 叫做类或者簇(cluster)，每个类均通过一些特征来描述：
    - 通过**类中心**或者**类的边界点**来表示；
    - 使用**聚类树**采用图形化方式来表示。

引言

- **聚类方法分类**
  - **按照聚类标准**
    - **统计聚类方法**：基于全局数据的聚类，即从全体样本中通过距离比较，获得聚类中心。主要采用欧氏距离度量、马氏距离度量等。
    - **概念聚类方法**：将数据按按一定的方式和准则进行分组，得到的分组代表着不同的概念。
  - **按聚类所处理的数据类型**
    - **数值型**数据聚类、**离散型**数据聚类、**混合型**数据聚类。

引言

- **聚类方法分类**
  - **按照度量准则**
    - 基于**距离**的聚类方法：基于各种不同的距离或者相似性来度量点对之间的关系，如**K-means**等。
    - 基于**密度**的聚类方法：采用密度函数对样本进行描述，并得到聚类结果。
    - 基于**连通性**的聚类方法：主要包含**基于图**的方法。高度连通的数据通常被聚为一簇，如**谱聚类**。

引言

- **聚类方法分类**
  - **按照不同的技术路线**
    - **模型法(原型聚类）：** 为每一个簇引入一个模型，然后对数据进行划分，使其满足各自分派的模型，如K-Means。
    - **层次法：** 对给定样本进行层次划分，如**层级聚类**。
    - **密度法：** 对数据的密度进行评价，如**混合高斯模型、Mean-Shift方法**。
    - **网格法：** 将数据空间划分为有限个单元网络结构，然后基于网络结构进行聚类，如**矢量量化**。

引言

- **挑战性问题**
  - **可伸缩性**
    - 可伸缩性是指聚类算法无论对于小数据集还是大数据集，都应有效；无论对小类别数据还是大别类数据，都应有效。
  - **具有不同类型的数据处理能力**
    - 既可处理数值型数据，也可处理非数植型数据；既可处理离散数据，也可处理连续域内的数据。比如布尔型、时序型、枚举型、以及这些类型的混合。
  - **能够发现任意形状的聚类**
    - 能够发现任意形状的簇，球状的、位于同一流形上的数据。因此，选择合适的距离度量很关键。

引言

- **挑战性问题**
  - **能够处理高维数据**
    - 既可处理属性较少的数据，也可处理属性较多的数据。
    - 在高维空间聚类更具挑战性，对于高维稀疏数据，这一点更突出。
  - **对噪声鲁棒**
    - 在实际中，绝大多数样本集都包含噪声、空缺、部分未知属性、孤立点、甚至错误数据。

引言

- **挑战性问题**
  - **具有约束的聚类**
    - 在实际应用中，通常需要在某种约束条件下进行聚类，既满足约束条件，以希望有高聚类精度，是一个挑战性问题。
  - **对初始输入参数鲁棒**
    - 具有自适应的簇数判定能力（一直没有解决好）。
    - 对初始聚类中心鲁棒。
  - **能够解决用户的问题**
    - 聚类结果能被用户所理解，并能带来经济效益，特别是在数据挖掘领域。

距离与相似性度量

- **距离**
  - 设有 d 维空间的三个样本**x**，**y** 和 **z**，记 $d(.\ ,\ .)$为一个$R^d \times R^d \rightarrow R$的映射，如满足如下几个条件则称$d(.\ ,\ .)$为一个距离:
    - $d(\mathbf{x}, \mathbf{y}) \geq 0$ 非负性
    - $d(\mathbf{x}, \mathbf{x}) = 0$ 自相似性
    - $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ 对称性
    - $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ 三角不等式
  - 距离可以描述对点间的相异程度，距离越大，两个点越不相似；距离越小，两个点越相似。

距离与相似性度量

- 设**x**, **y** $\in \mathrm{R}^d$，**Minkowski 距离度量**定义如下：

$$d(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^q \right)^{\frac{1}{q}}$$

$$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$$

城区距离

曼哈顿距离

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$$

欧氏距离

$$d(\mathbf{x},\mathbf{y}) = \max_{1 \le i \le d} |x_i - y_i|$$

切比雪夫距离

距离与相似性度量

- 设$\mathbf{x}, \mathbf{y} \in R^d$， **Mahalanobis (马氏)距离**定义如下：

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \mathbf{M}(\mathbf{x}-\mathbf{y})}$$

其中，**M**是半正定矩阵。

  - **M**为单位矩阵时，退化为欧氏距离度量。
  - **M**为对角矩阵时，退化为特征加权欧氏距离

距离与相似性度量

- **相似性**
  - 设**x**, **y** $\in \mathrm{R}^d$，**余弦相似度**定义如下：

$$s(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$$

（两个模为1的向量之内积）

距离与相似性度量

- **相似性**
  - 设**x**，**y** $\in R^d$，其每维特征只取{0,1}中的一个值。为了定义数据点之间的距离，通常先计算出如下几个值：

    - f_{00}：样本 **x** 和 **y** 中满足 $x_i=y_i=0$ 的属性的个数
    - f_{10}：样本 **x** 和 **y** 中满足 $x_i=1\&y_i=0$ 的属性的个数
    - f_{01}：样本 **x** 和 **y** 中满足 $x_i=0\&y_i=1$ 的属性的个数
    - f_{11}：样本 **x** 和 **y** 中满足 $x_i=y_i=1$ 的属性的个数

  - 进一步，可定义如下几种类型的相似性度量：

距离与相似性度量

- **相似性**
  - **简单匹配系数**(simple matching coefficient, SMC):

$$s_{SMC}(\mathbf{x},\mathbf{y}) = \frac{f_{00} + f_{11}}{f_{00} + f_{10} + f_{01} + f_{11}}$$

  - **Jaccard 相似系数：**

$$s_J(\mathbf{x},\mathbf{y}) = \frac{f_{11}}{f_{10} + f_{01} + f_{11}}$$

  - **Tanimoto系数：**

$$s_T(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T\mathbf{y}}{\mathbf{x}^T\mathbf{x}+\mathbf{y}^T\mathbf{y}-\mathbf{x}^T\mathbf{y}} = \frac{f_{11}}{\mathbf{x}^T\mathbf{x}+\mathbf{y}^T\mathbf{y}-f_{11}}$$

x取1的个数　　**y**取1的个数

距离与相似性度量

- **举例：**计算如下**两位顾客x和y**的相似度：

商品	面包	啤酒	牛奶	咖啡	茶叶	鸡蛋	猪肉	牛肉	洋葱	土豆	大米	白糖
x	1	1	1	1	0	0	0	1	1	1	0	1
y	1	0	1	0	1	1	0	0	0	1	1	1
商品	莲藕	花生	可乐	豆腐	菠菜	黄瓜	面粉	酱油	辣椒	白酒	黄鱼	茄子
x	0	0	1	0	1	1	1	0	1	1	1	0
y	1	0	1	1	1	1	1	1	0	0	1	0

距离与相似性度量

- **类间距离：**
  - **最短距离法**：定义两个类中最近的两个样本的距离为类间距离。

$$d(D_a, D_b) = \min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in D_a, \mathbf{y} \in D_b\}$$

  - **最长距离法**：定义两个类中最远的两个样本的距离为类间距离。

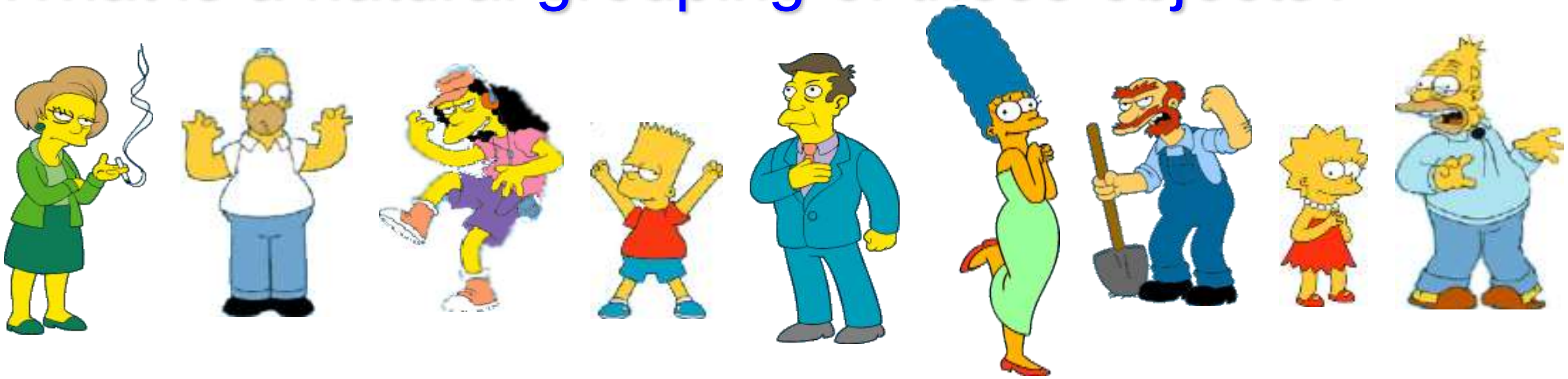
$$d(D_a, D_b) = \max\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in D_a, \mathbf{y} \in D_b\}$$

- **类直径：** 类直径反映类中样本之间的差异，可定义为类中各样本至**类中心点**的欧氏距离平方和：

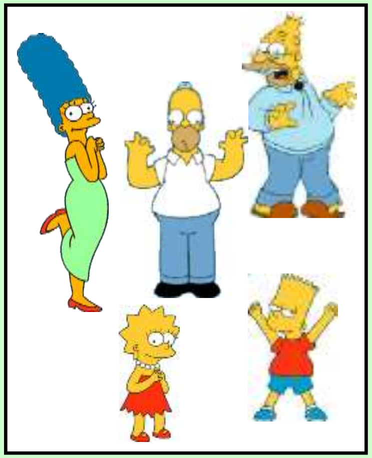
$$r(D_a) = \sum_{\mathbf{x}\in D_a} (\mathbf{x} - \bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}))$$

距离对聚类的影响

What is a natural grouping of these objects?

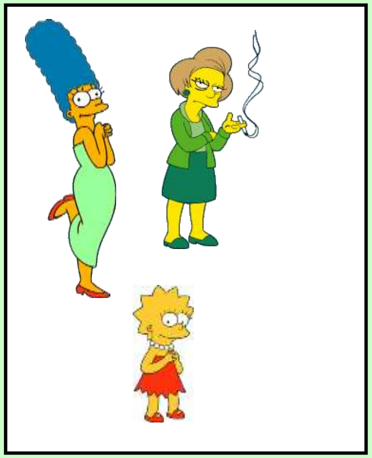


Clustering is subjective



Simpson's Family

School Employees



Females

Males

K-means Clustering

K-means algorithm (Lloyd, 1957)

- **Input:** N examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$; $\mathbf{x}_n \in \mathbb{R}^D$; the number of partitions K

- **Initialize:** K cluster means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, each $\boldsymbol{\mu}_k \in \mathbb{R}^D$
    - Usually initialized randomly, but good initialization is crucial; many smarter initialization heuristics exist (e.g., K-means++, Arthur & Vassilvitskii, 2007)

- **Iterate:**
    - (Re)-Assign each example $\mathbf{x}_n$ to its closest cluster center (based on the smallest Euclidean distance)

      $$\boxed{\mathcal{C}_k = \{n : \quad k = \arg\min_k ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2\}}$$

      ($\mathcal{C}_k$ is the set of examples assigned to cluster k with center $\boldsymbol{\mu}_k$)
    - Update the cluster means

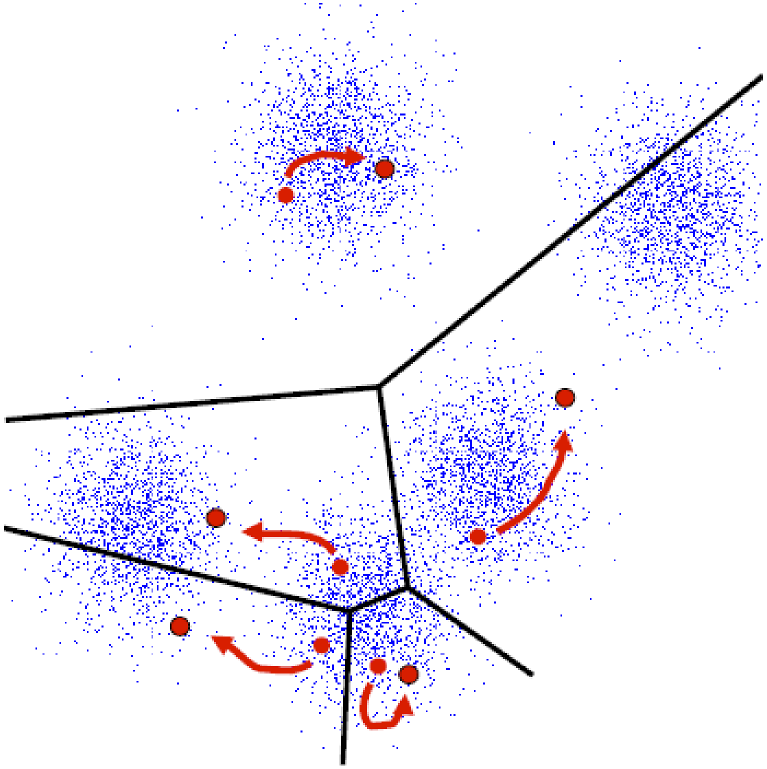
      $$\boxed{\mu_k = \text{mean}(\mathcal{C}_k) = \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n}$$
    - Repeat while not converged

- Stop when cluster means or the "loss" does not change by much

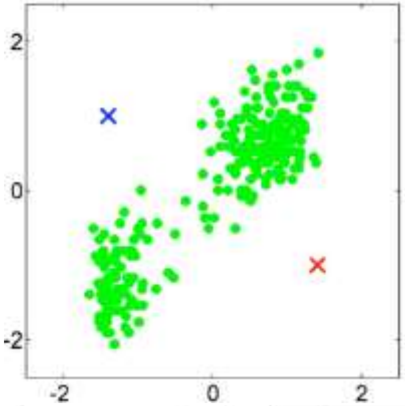
K-means algorithm (Lloyd, 1957)

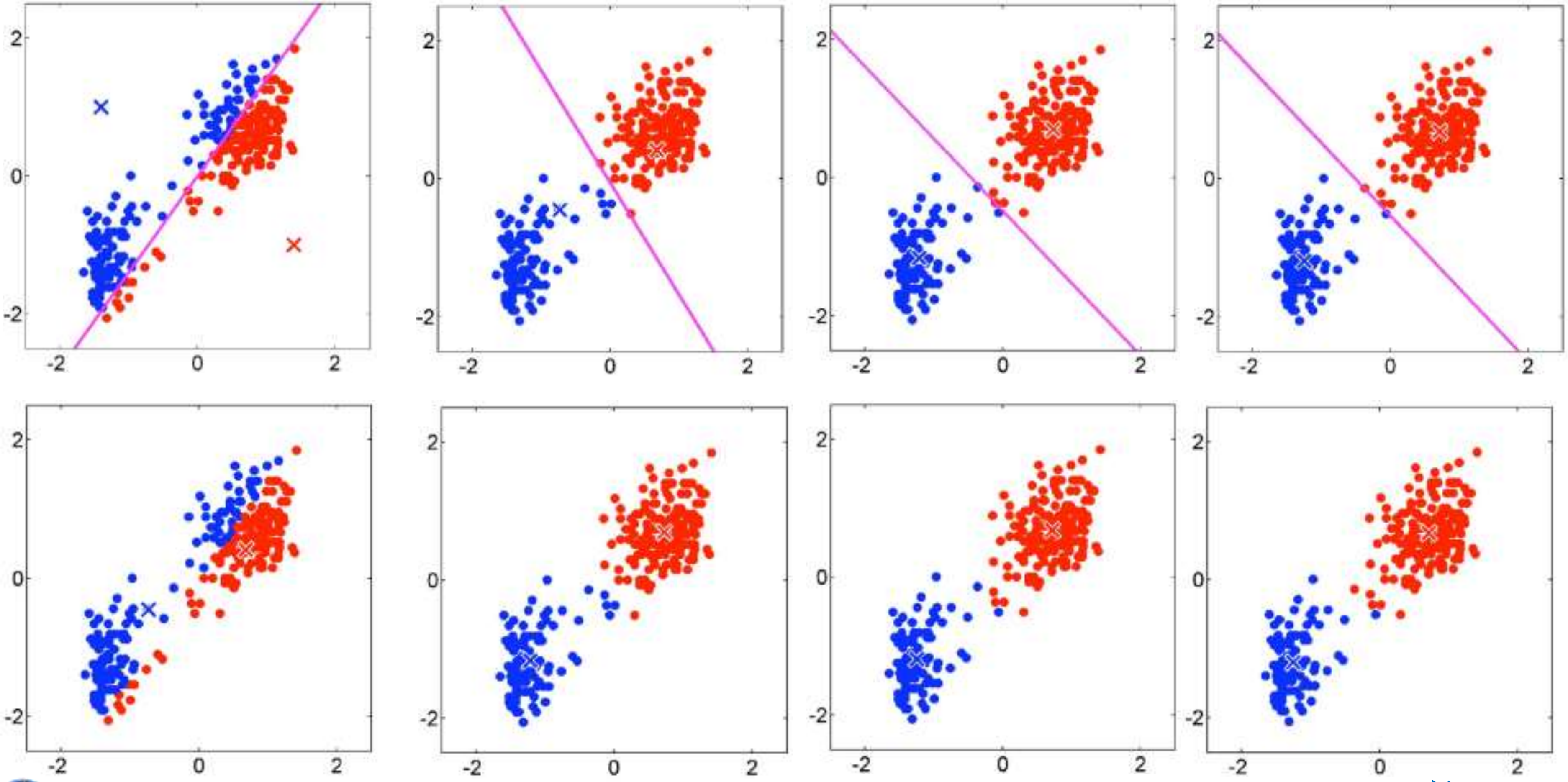
An iterative clustering algorithm

- Initialize: Pick *K* random points as cluster centers
- Alternate:
  1. Assign data points to closest cluster center
  2. Change the cluster center to the average of its assigned points
- Stop when no points' assignments change

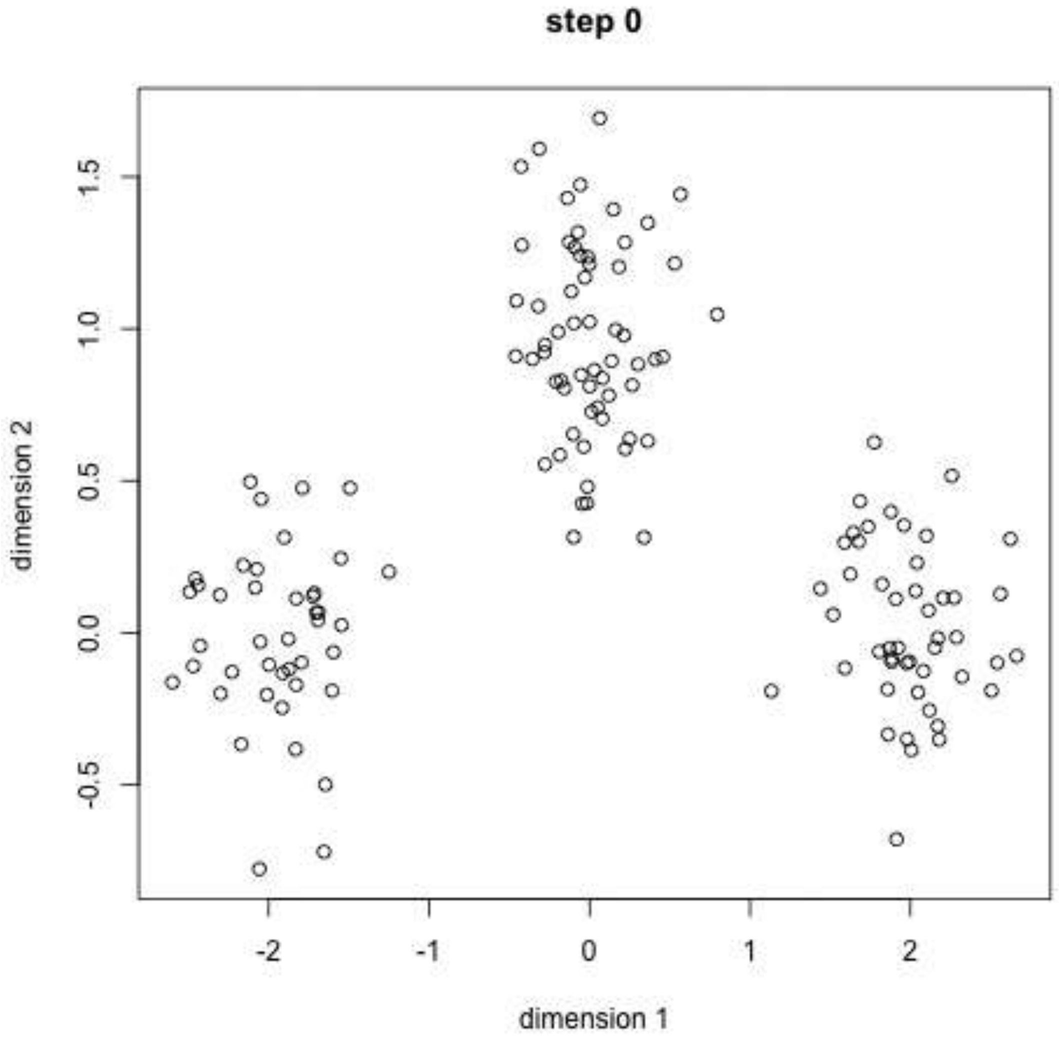


K-means algorithm





K-means illustration



Loss function of K-means?

- Let $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ be the K cluster centroids (means)
- Let $z_{nk} \in \{0, 1\}$ be s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster k, and 0 otherwise
  - Note: $\boldsymbol{z}_n = [z_{n1}\ z_{n2}\ \ldots\ z_{nK}]$ represents a length K one-hot encoding of $\boldsymbol{x}_n$
- Define the distortion or "loss" for the cluster assignment of $\boldsymbol{x}_n$

$$\ell(\boldsymbol{\mu}, \boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2$$

- Total distortion over all points defines the K-means "loss function"

$$L(\boldsymbol{\mu}, \mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2 = ||\mathbf{X} - \mathbf{Z}\boldsymbol{\mu}||^2$$

where $\mathbf{Z}$ is $N \times K$ (row n is $\boldsymbol{z}_n$) and $\boldsymbol{\mu}$ is $K \times D$ (row k is $\boldsymbol{\mu}_k$)

- The K-means **problem** is to minimize this objective w.r.t. $\boldsymbol{\mu}$ and $\mathbf{Z}$
  - Note that the objective only minimizes within-cluster distortions

K-means objective

- Consider the K-means objective function

$$\boxed{L(\boldsymbol{\mu}, \mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}||\mathbf{x}_n - \mu_k||^2}$$

- It is a non-convex objective function
  - Many local minima possible
- Also NP-hard to minimize in general (note that $\mathbf{Z}$ is discrete)
- The K-means algorithm we saw is a heuristic to optimize this function
- K-means algorithm alternated between the following two steps
  - Fix $\boldsymbol{\mu}$, minimize w.r.t. $\mathbf{Z}$ (assign points to closest centers)
  - Fix $\mathbf{Z}$, minimize w.r.t. $\boldsymbol{\mu}$ (recompute the center means)

Convergence of *K*-means

- Each step (updating $\mathbf{Z}$ or $\boldsymbol{\mu}$) can never increase the objective
- When we update $\mathbf{Z}$ from $\mathbf{Z}^{(t-1)}$ to $\mathbf{Z}^{(t)}$

$$L(\boldsymbol{\mu}^{(t-1)}, \mathbf{X}, \mathbf{Z}^{(t)}) \leq L(\boldsymbol{\mu}^{(t-1)}, \mathbf{X}, \mathbf{Z}^{(t-1)})$$

because the new $\mathbf{Z}^{(t)} = \arg\min_{\mathbf{Z}} L(\boldsymbol{\mu}^{(t-1)}, \mathbf{X}, \mathbf{Z})$

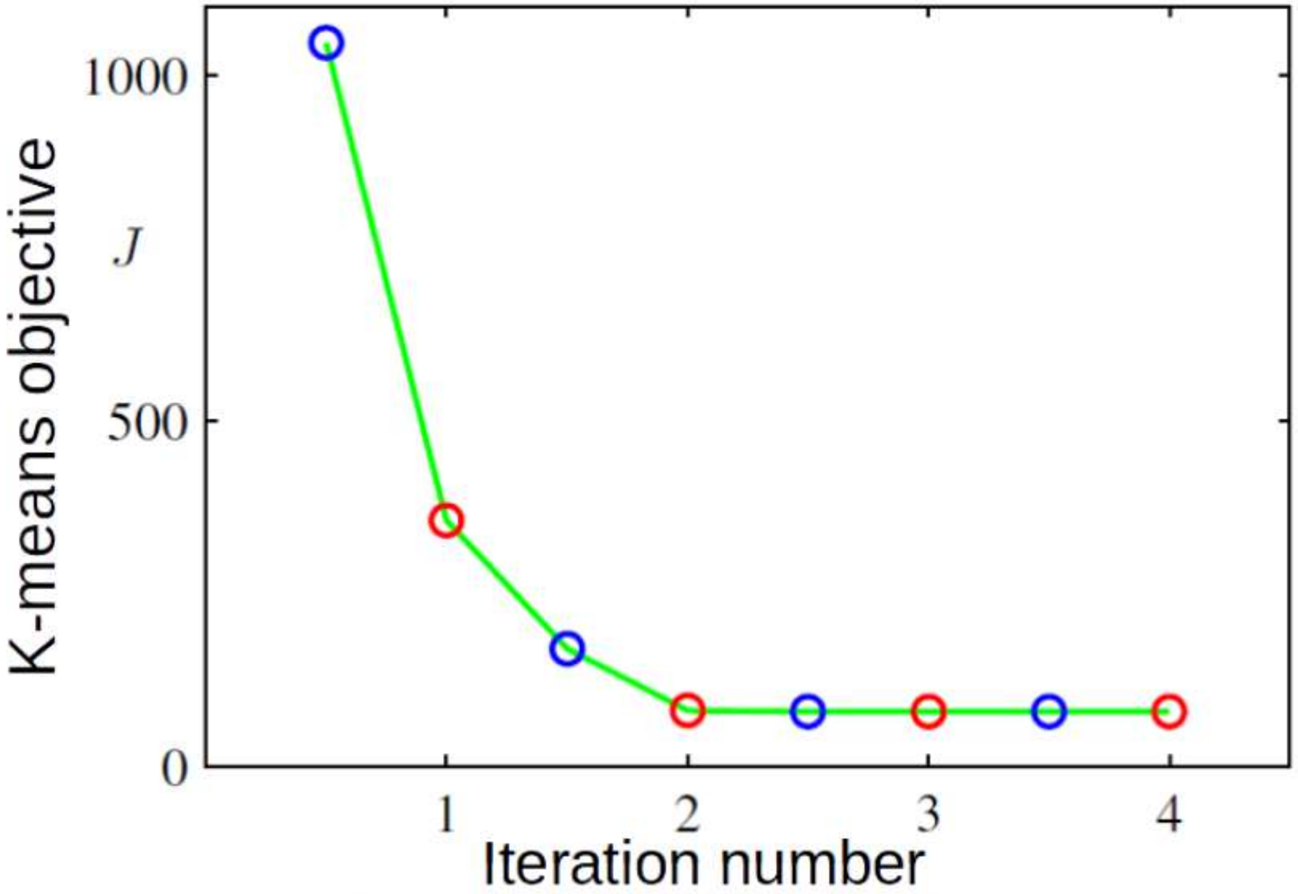
- When we update $\boldsymbol{\mu}$ from $\boldsymbol{\mu}^{(t-1)}$ to $\boldsymbol{\mu}^{(t)}$

$$L(\boldsymbol{\mu}^{(t)}, \mathbf{X}, \mathbf{Z}^{(t)}) \leq L(\boldsymbol{\mu}^{(t-1)}, \mathbf{X}, \mathbf{Z}^{(t)})$$

because the new $\boldsymbol{\mu}^{(t)} = \arg\min_{\boldsymbol{\mu}} L(\boldsymbol{\mu}, \mathbf{X}, \mathbf{Z}^{(t)})$

- Thus the K-means algorithm monotonically decreases the objective

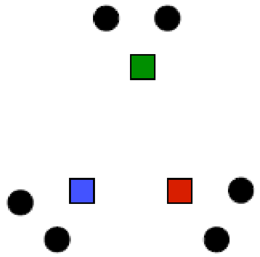
Convergence of *K*-means



(blue: after Z updated, red: after μ updated)

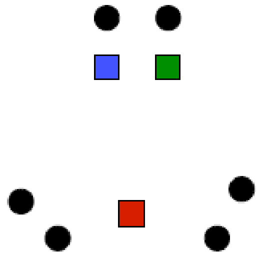
Local optimum: initialization matters

- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!



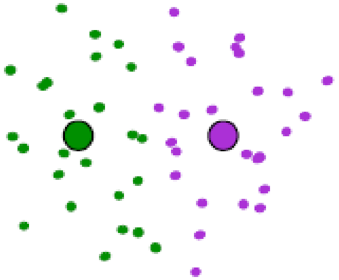
  - What can go wrong?

  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics

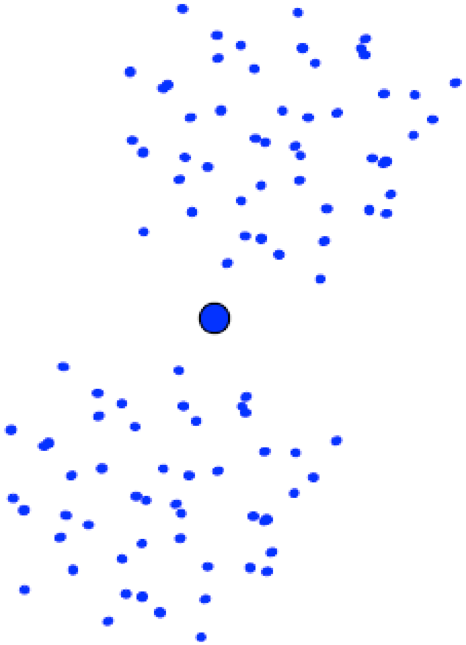


Local optimum: *K* matters

A local optimum:



Would be better to have one cluster here



… and two clusters here

Runing time of *K*-means

- Guaranteed to converge in a finite number of iterations

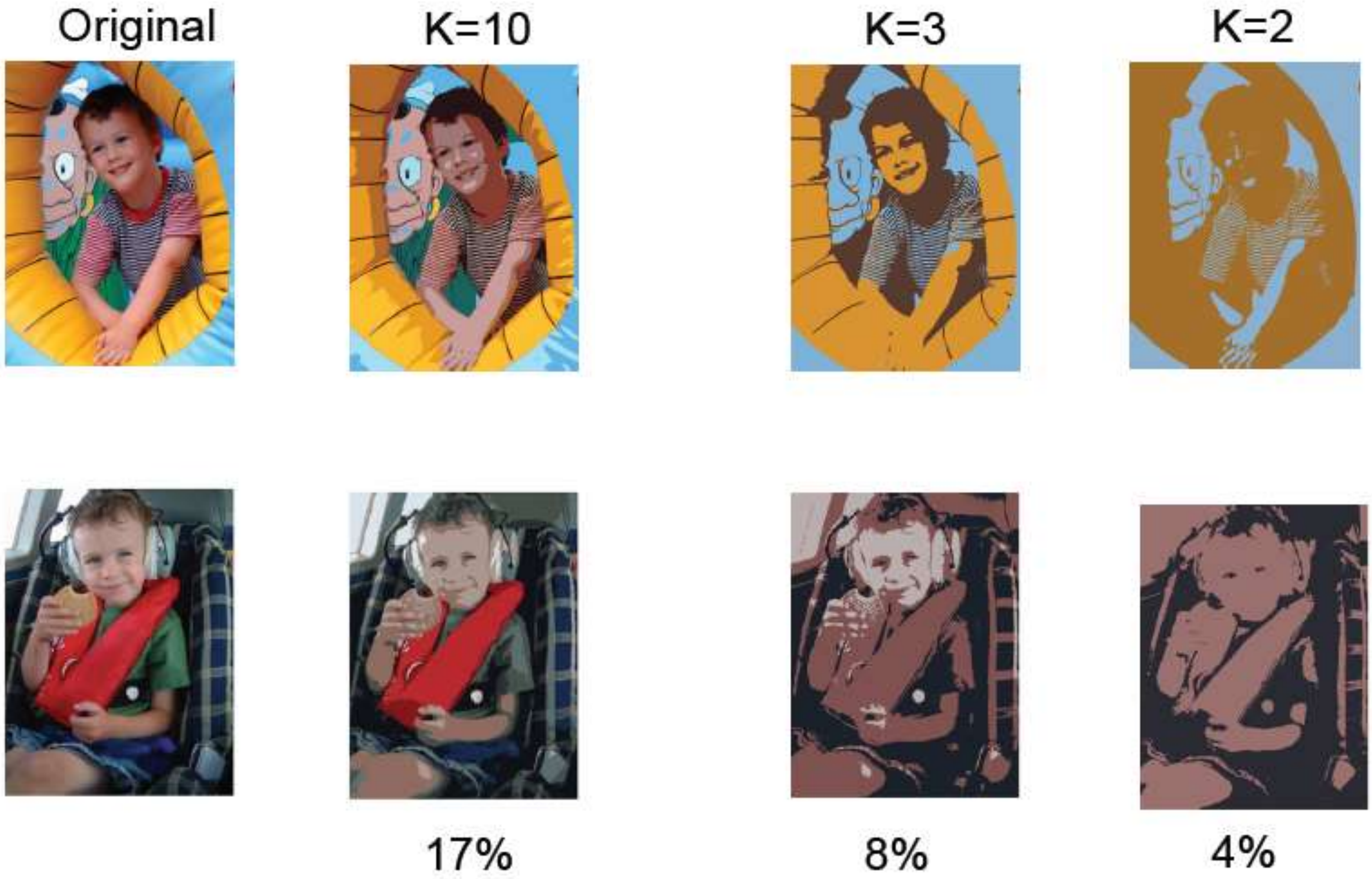
- Running time per iteration:
    1. Assign data points to closest cluster center

       O(KN) time

    2. Change the cluster center to the average of its assigned points

       O(N)

Example: *K*-means for segmentation



Example: vector quantization

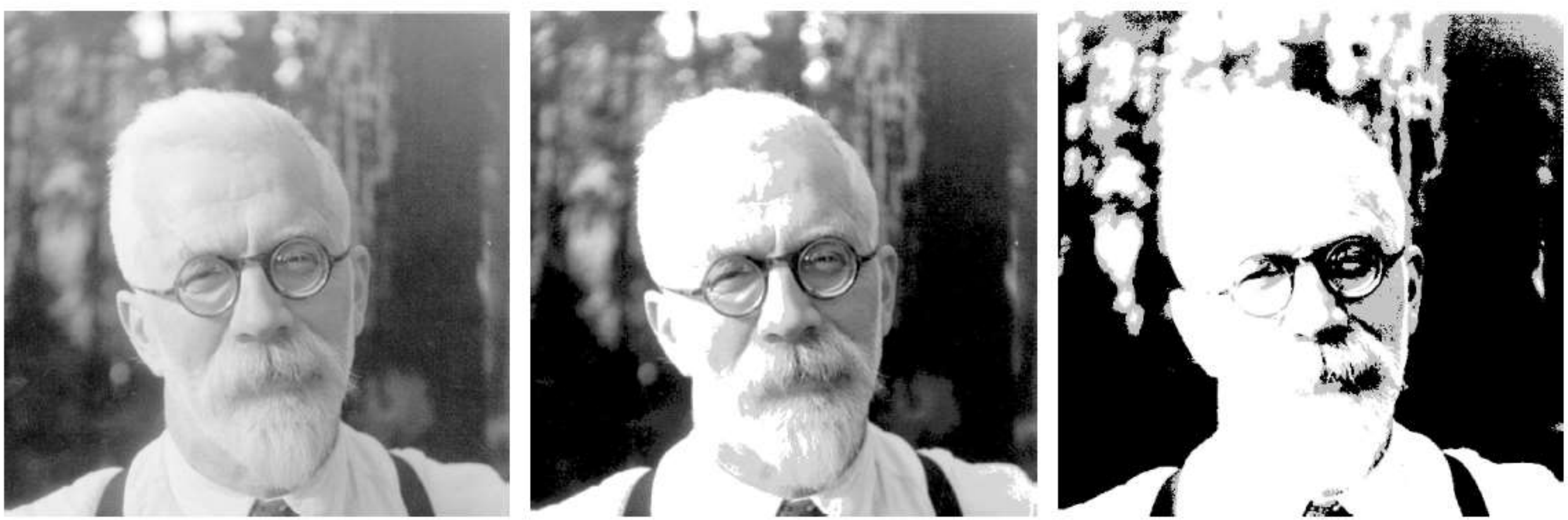
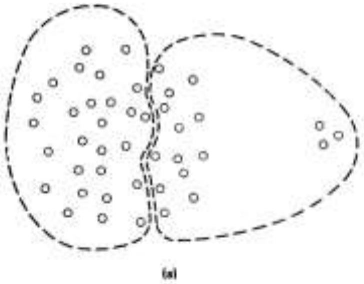


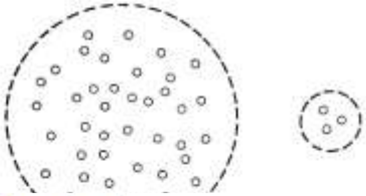
FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

Limitations of K-means



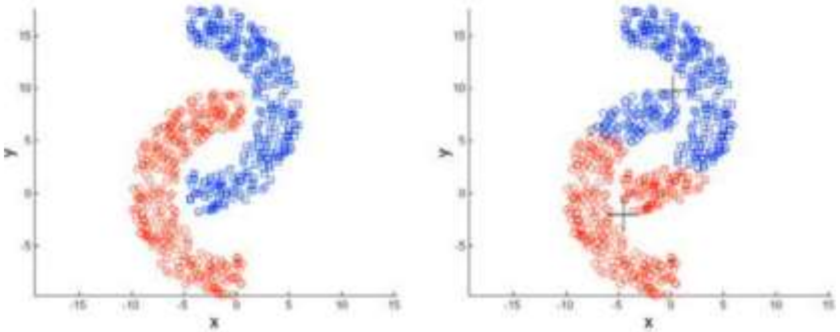
(a)

- Makes hard assignments of points to clusters
    - A point either completely belongs to a cluster or doesn't belong at all
    - No notion of a soft assignment (i.e., probability of being assigned to each clus some point $\mathbf{x}_n$, $p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$)



- Works well only is the clusters are roughly of equal sizes
- Probabilistic clustering methods such as Gaussian mixture models can handle both these issues (model each cluster using a Gaussian distribution)

- K-means also works well only when the clusters are round-shaped and does badly if the clusters have non-convex shapes



- Kernel K-means or Spectral clustering can handle non-convex

Selection of *K*

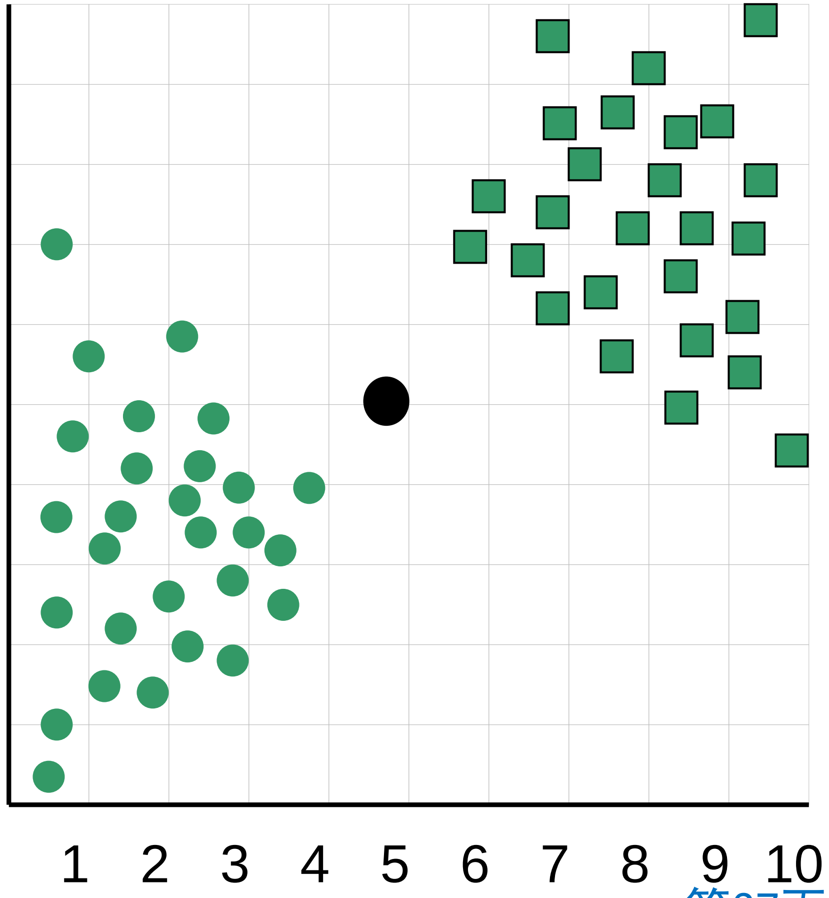
- Idea 1: Use trick of cross validation to select k
  - What should we optimize?
  - Problem?

- Idea 2: Let domain expert look at the clustering and decide if they like it
  - How should we show this to them?
  - Problem?

- Idea 3: The “knee” solution

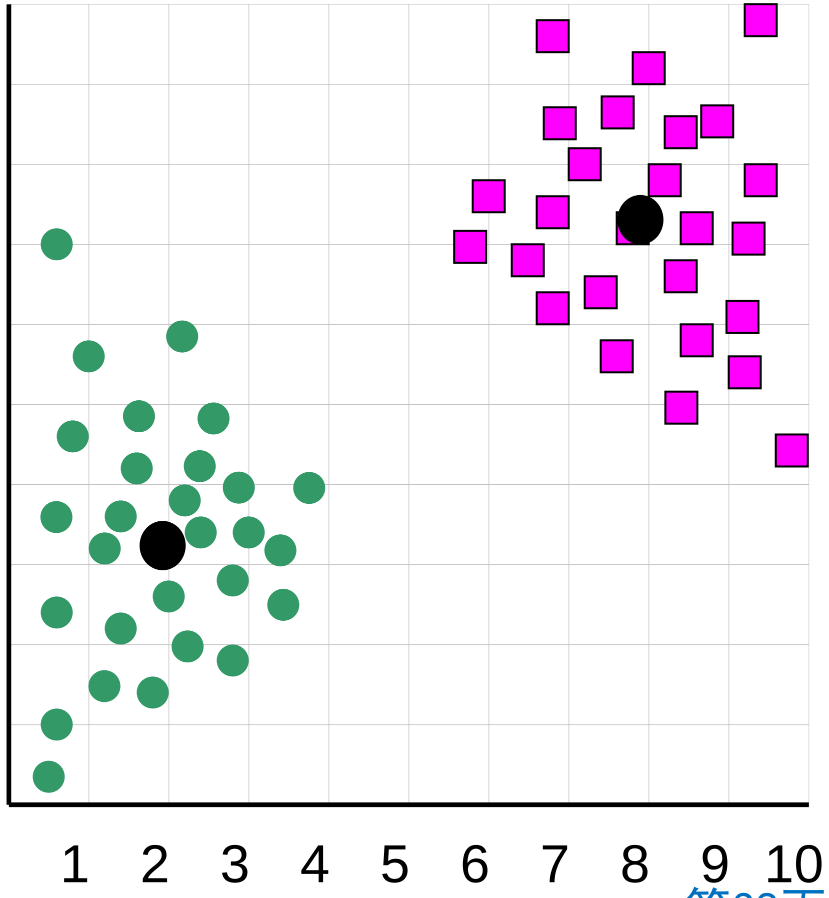
Selection of *K*

When k = 1, the objective function is 873.0



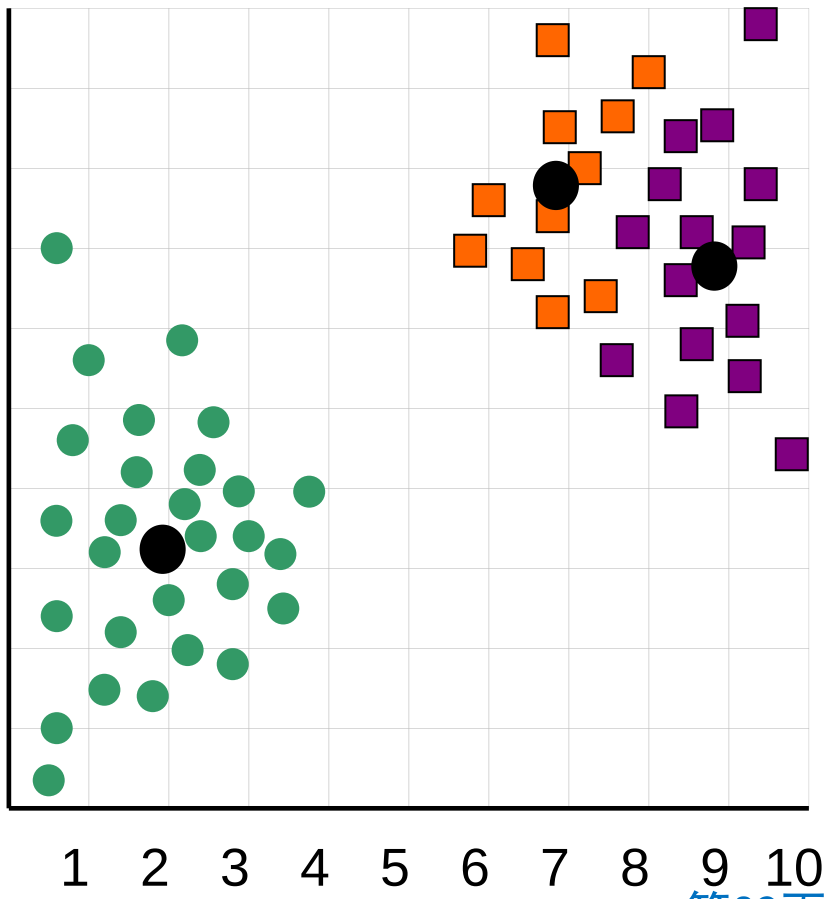
Selection of *K*

When k = 2, the objective function is 173.1



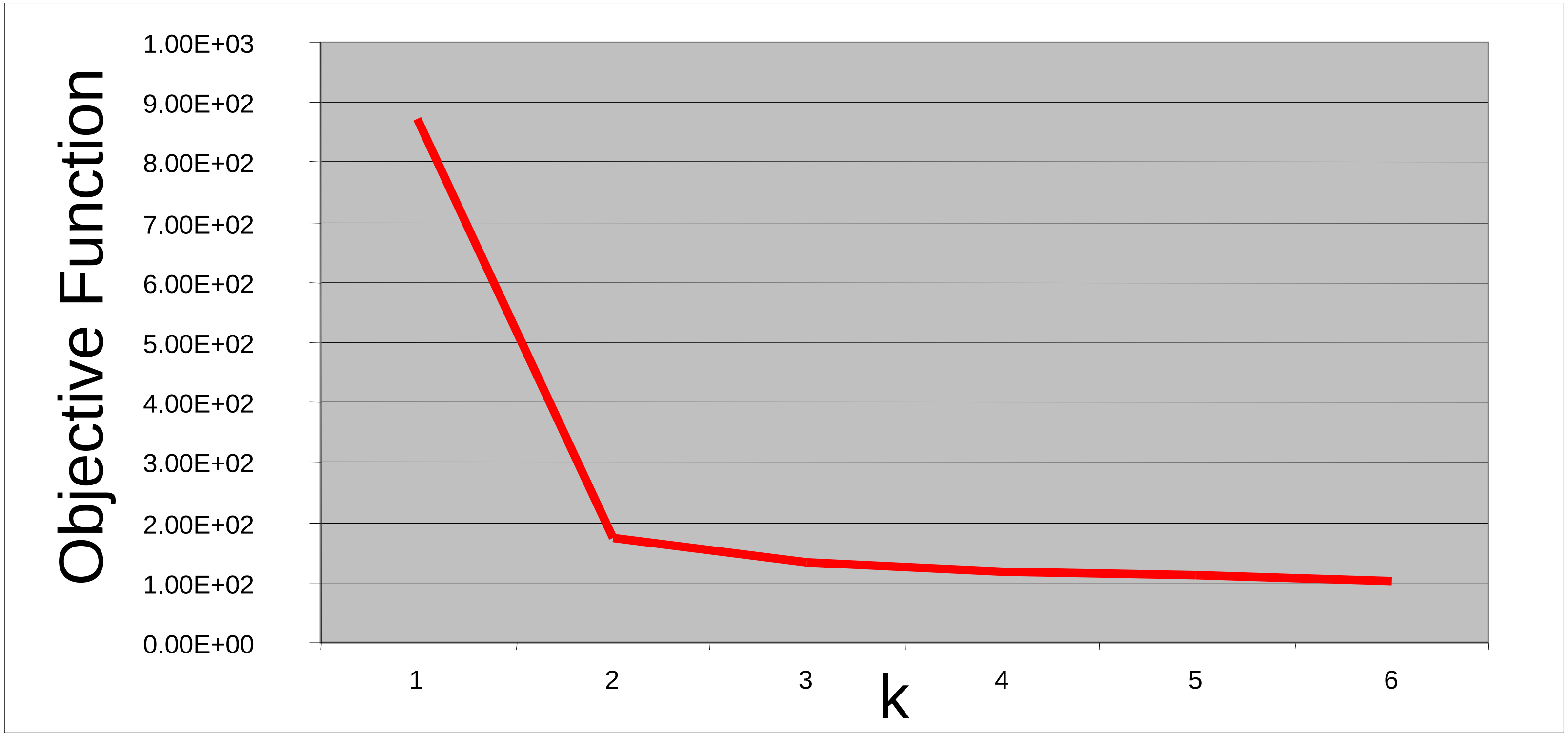
Selection of *K*

When k = 3, the objective function is 133.6



We can plot the objective function values for k equals 1 to 6…

The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as "knee finding" or "elbow finding".



High-dimensional *K*-means

- Difficult to find true clusters
  - Irrelevant and redundant features
  - All points are equally close
- Solutions: Dimension Reduction
  - PCA, LDA, CCA, ICA ……
  - Feature selection
  - Cluster ensembles using random projection

K-Medians

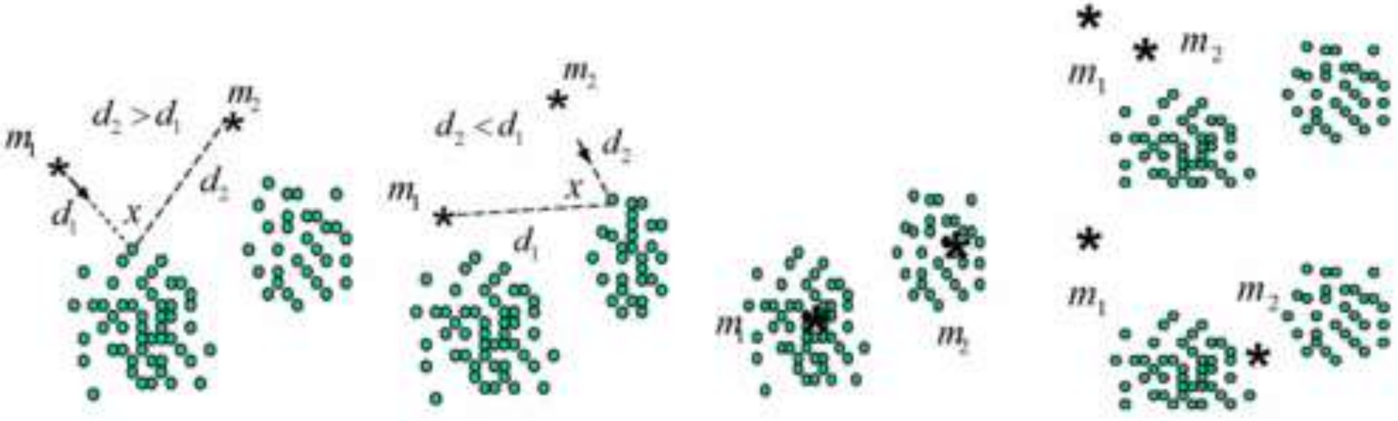
- ***K*-Medians** is another clustering algorithm related to ***K*-Means**.
- Instead of recomputing the group center points using the mean, we use the **median vector** of the group.
- This method is less sensitive to outliers (because of using the Median)
- But is much slower for larger datasets as sorting is required on each iteration when computing the median vector.

K-Medians

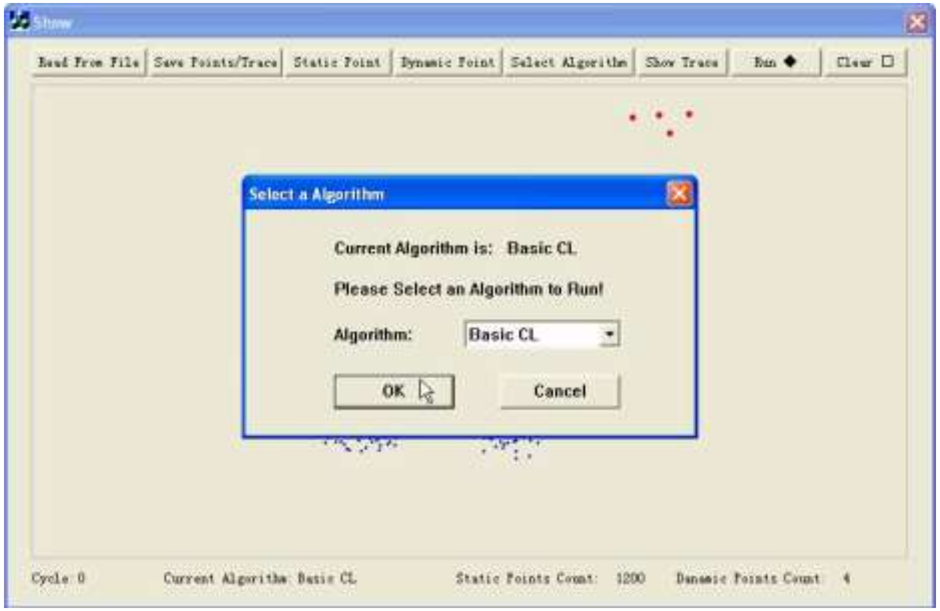
- Assignment step: assigns each observation to the cluster with the closest center.

- Update step: calculates the new median of each feature of each cluster to be the new center of that cluster.

Frequency Sensitive Competitive Learning



(*a*) m_1 is the winner (*b*) m_2 is the winner (*c*) converged (*d*) one unit dead

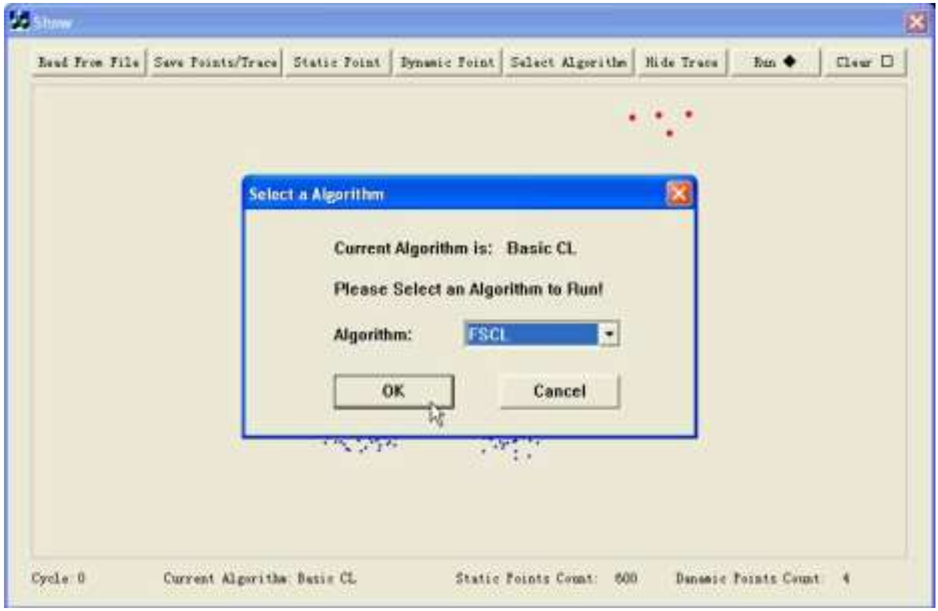


Current Algorithm is: Basic CL

Please Select an Algorithm to Run!

Algorithm: Basic CL

Cycle: 0 Current Algorithm: Basic CL Static Points Count: 1200 Danamic Points Count: 4



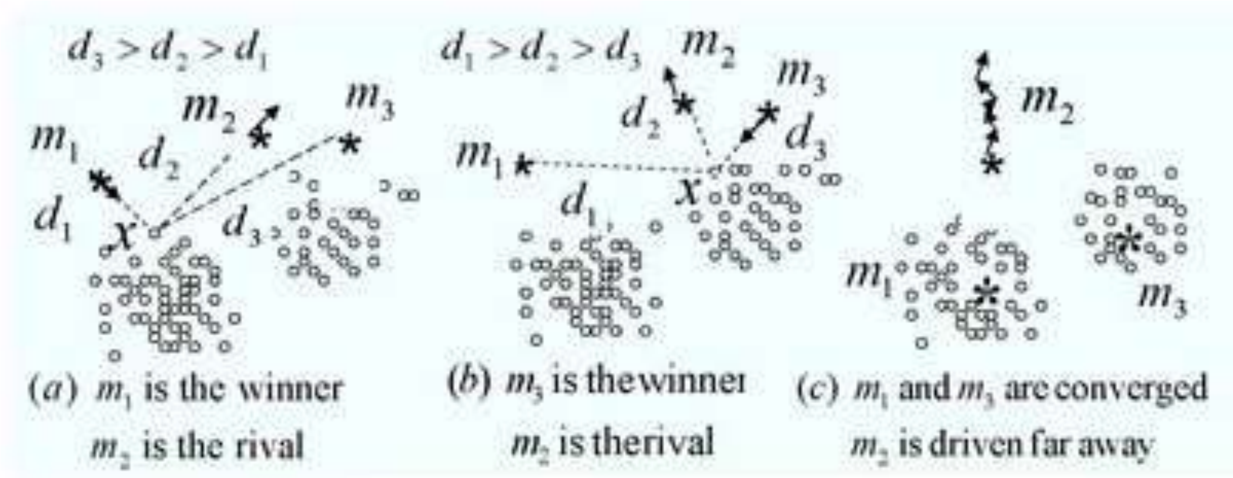
Current Algorithm is: Basic CL

Please Select an Algorithm to Run!

Algorithm: FSCL

Cycle: 0 Current Algorithm: Basic CL Static Points Count: 600 Danamic Points Count: 4

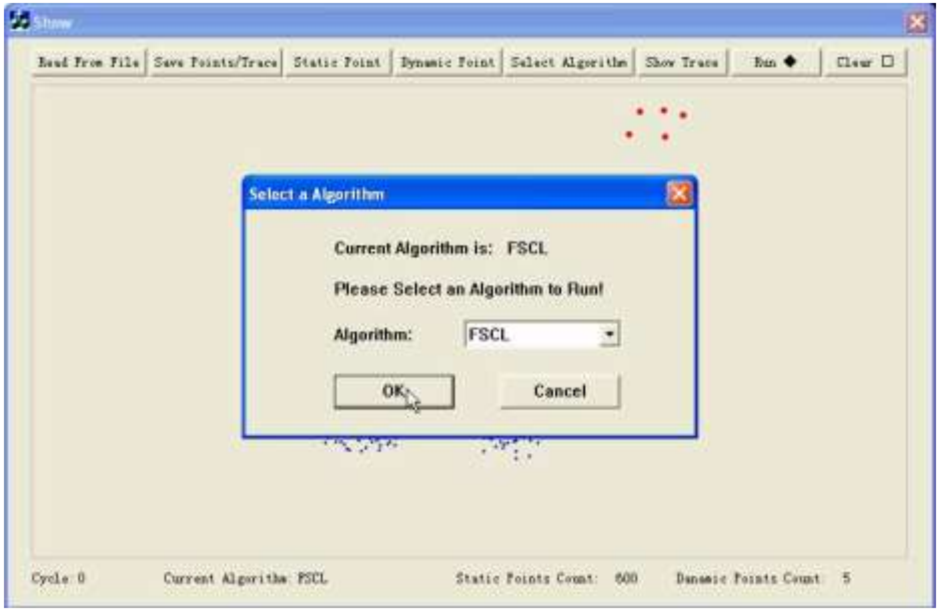
Rival Penalized Competitive Learning



(*a*) m_1 is the winner m_2 is the rival

(*b*) m_3 is thewinner m_2 is therival

(*c*) m_1 and m_3 are converged m_2 is driven far away

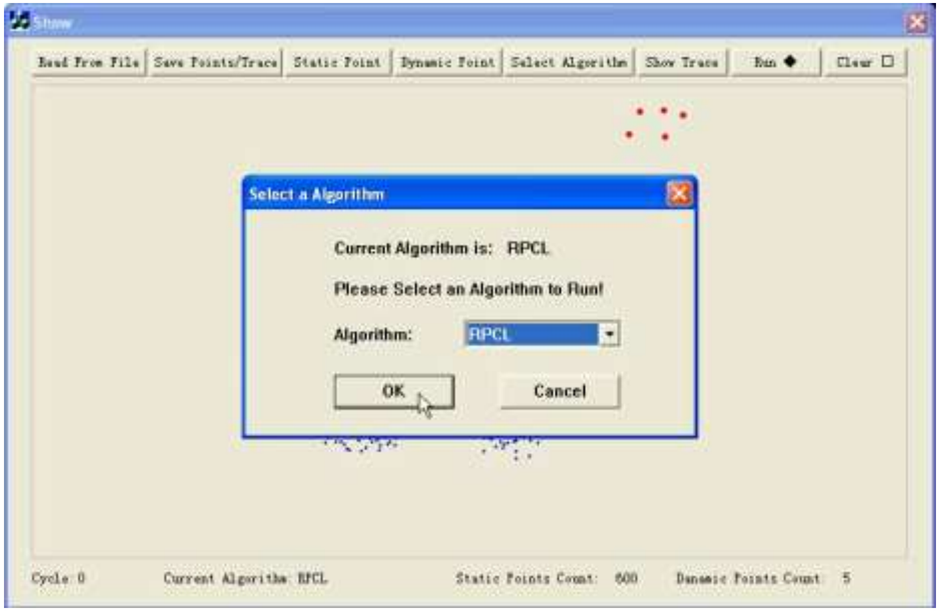


Current Algorithm is: FSCL

Please Select an Algorithm to Run!

Algorithm: FSCL

Cycle: 0 Current Algorithm: FSCL Static Points Count: 600 Dynamic Points Count: 5



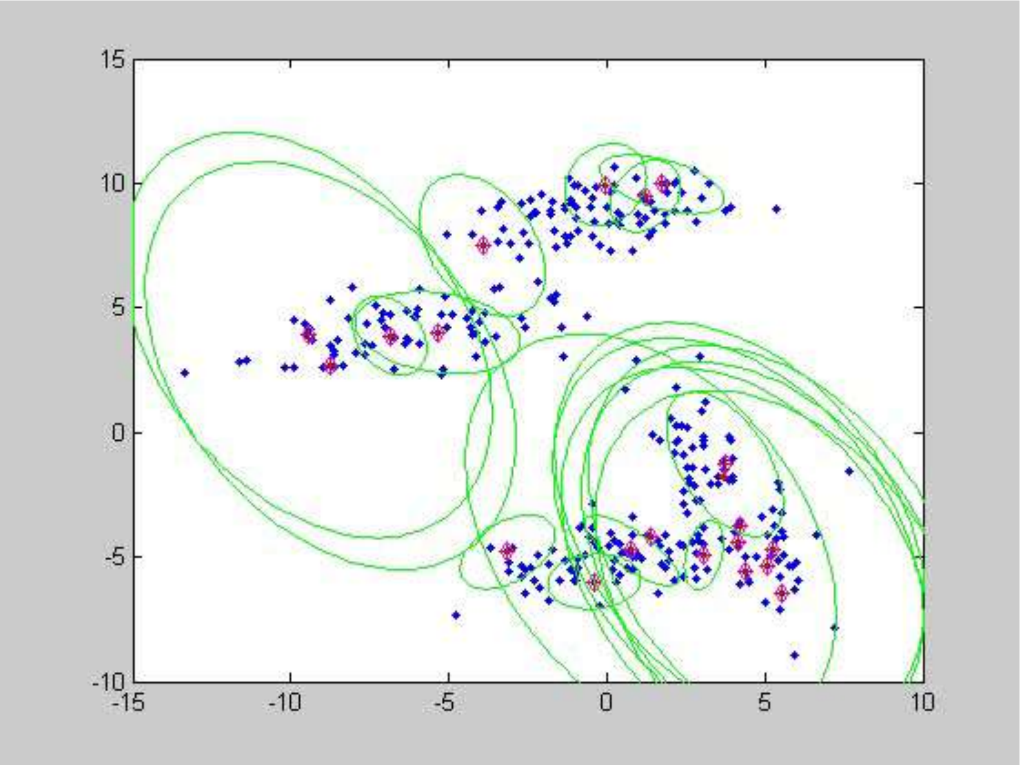
Current Algorithm is: RPCL

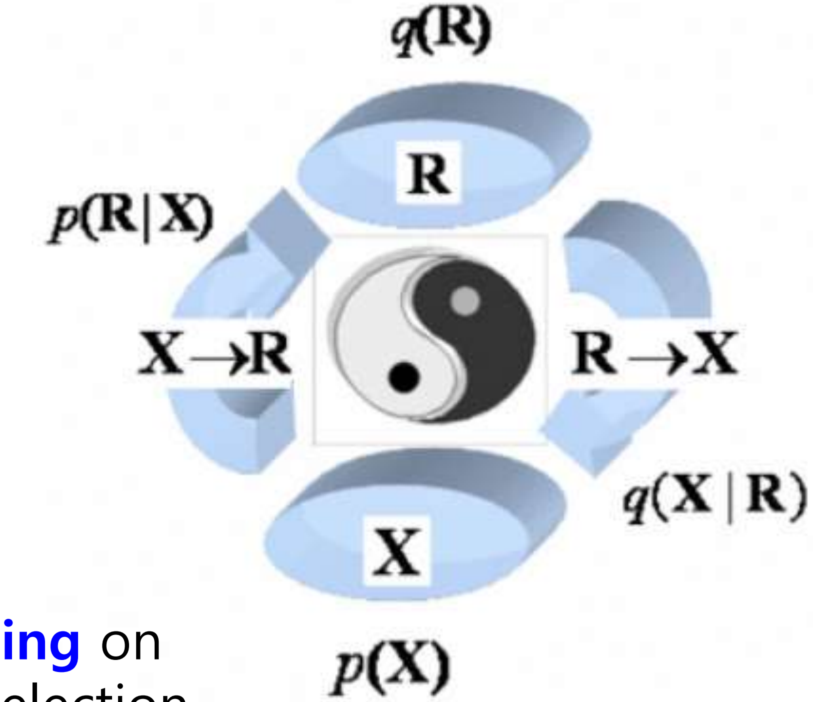
Please Select an Algorithm to Run!

Algorithm: RPCL

Cycle: 0 Current Algorithm: RPCL Static Points Count: 600 Dynamic Points Count: 5

Bayesian Ying Yang Learning



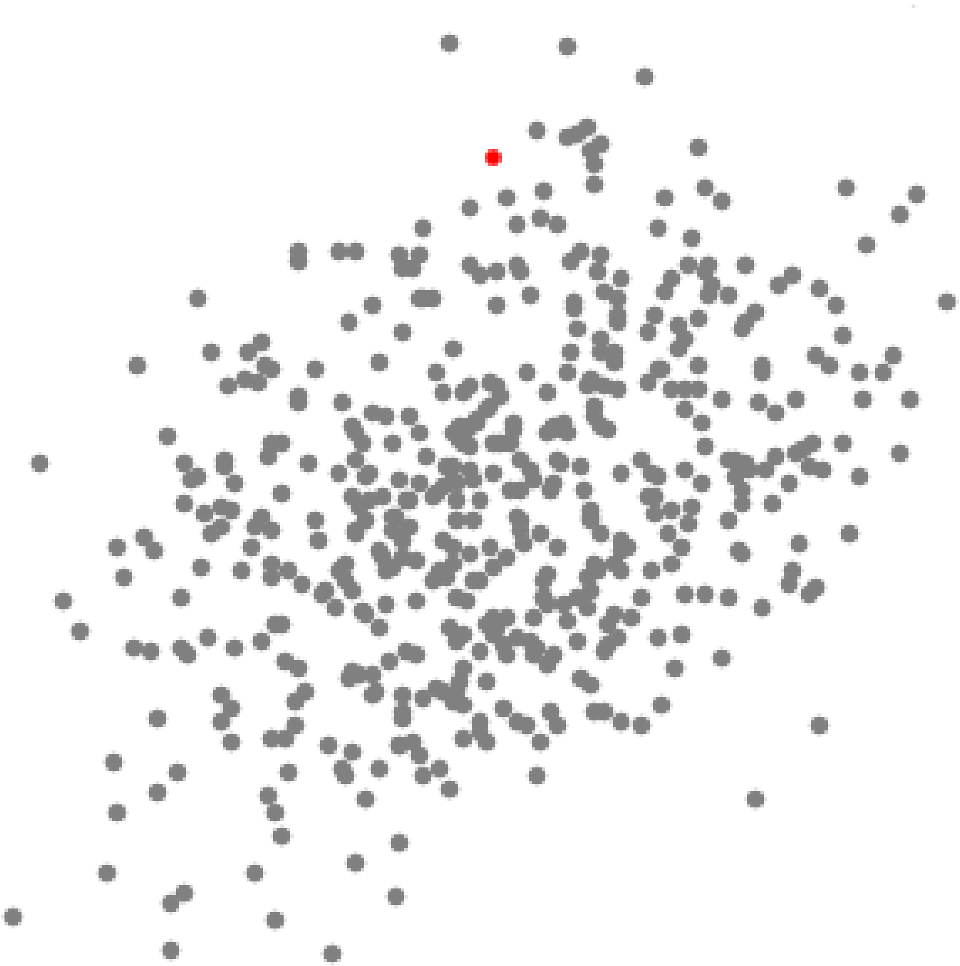


An illustration of the **BYY harmony learning** on Gaussian mixture with automatic model selection

Mean-Shift Clustering

- **mean shift** clustering
  - **sliding-window**-based algorithm
  - find dense areas of data points
- **centroid**-based algorithm
  - updating candidates for center points to be the **mean** of the points within the sliding-window
  - candidate windows are then filtered in a post-processing stage to eliminate near-duplicates, forming the final set of center points and their corresponding groups

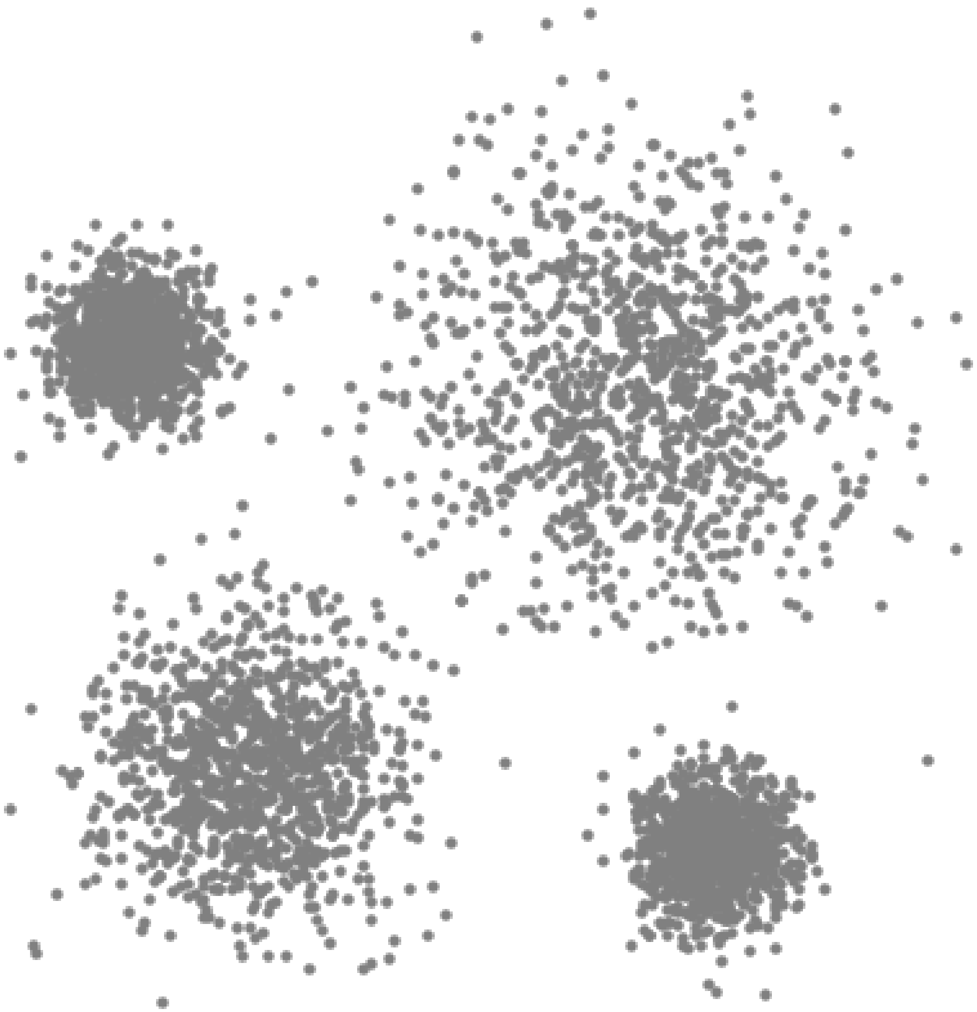
Mean-Shift with a Single Center



Mean-Shift Clustering

1. Begin with a **circular sliding window** centered at a point C (randomly selected) and having **radius r** as the kernel. Mean shift is a **hill-climbing algorithm** that involves shifting this kernel iteratively to a higher density region on each step until convergence.
2. At every iteration, the sliding window is shifted towards regions of higher density by **shifting the center point to the mean** of the points within the window (hence the name).
3. We continue shifting the sliding window according to the mean until there is no direction at which a shift can accommodate more points inside the kernel.
4. This process of steps 1 to 3 is done with **many sliding windows** until all points lie within a window.
5. When multiple sliding windows overlap the window containing the most points is preserved.
6. The data points are then clustered according to the sliding window in which they reside.

Mean-Shift with Multiple Centers



Mean-Shift Clustering

1. In contrast to K-means clustering, there is no need to select the number of clusters as mean-shift automatically discovers this. That’s a massive advantage.

2. The fact that the cluster centers converge towards the points of maximum density is also quite desirable as it is quite intuitive to understand and fits well in a naturally data-driven sense.

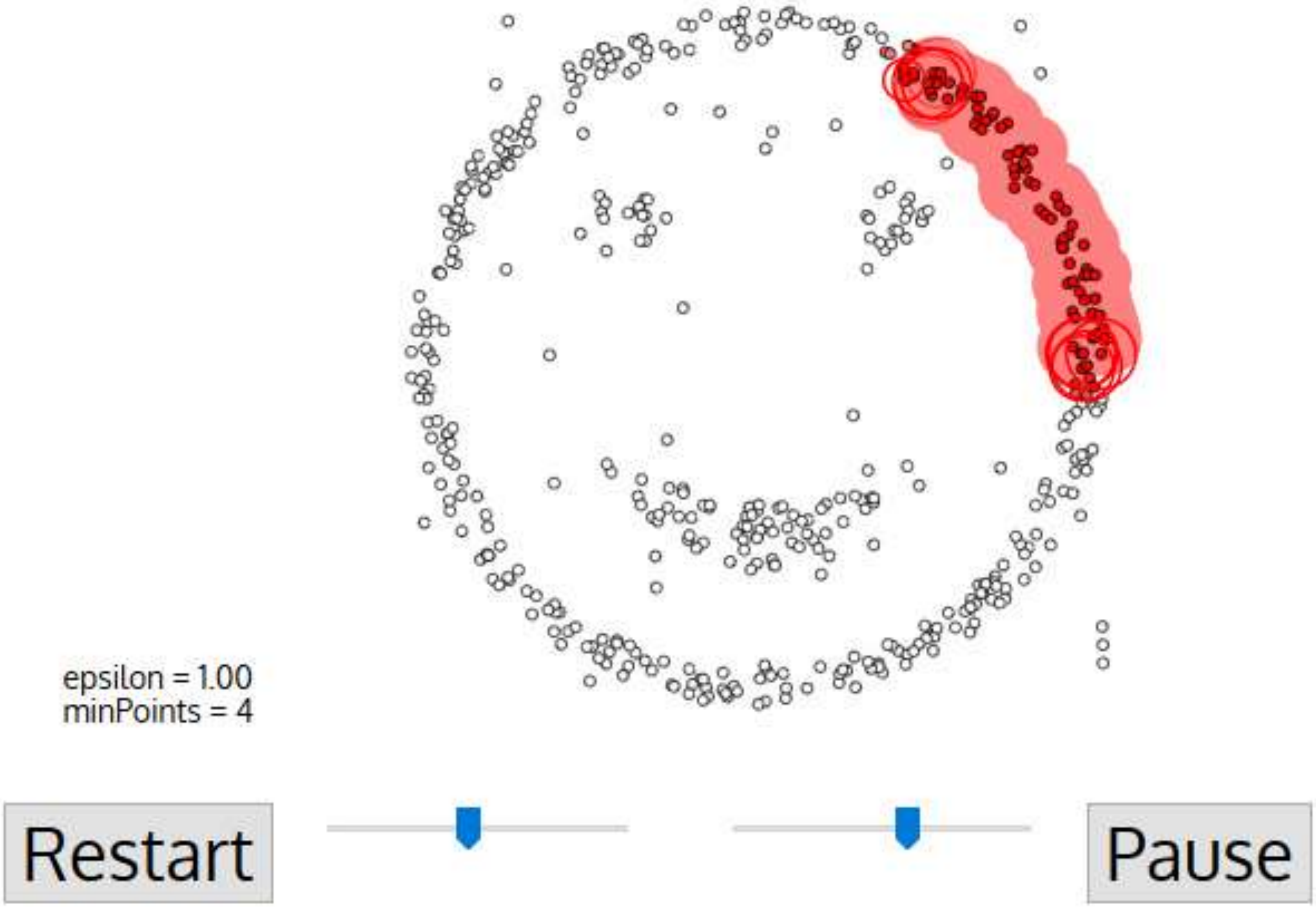
3. The drawback is that the selection of the **window size/radius “r”** can be non-trivial.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustered algorithm similar to mean-shift, but with a couple of notable advantages.

1. Begin with an arbitrary starting data point (**not visited**). The **neighborhood** of this point: all points within the **ε distance** to this point.
2. Sufficient number of points > **minPoints** within neighborhood: current data point becomes the first point in the new cluster.
3. Otherwise, the point will be labeled as **noise** (later this noisy point might become the part of another cluster).
4. In both cases that point is marked as **"visited".**
5. For this first point in the new cluster, the points within its ε distance neighborhood also become part of the same cluster. This procedure of **making all points in the ε neighborhood belong to the same cluster** is then repeated for all of the new points that have been just added to the cluster group.
6. This process repeats **until all points are marked as visited**.

DBSCAN



epsilon = 1.00
minPoints = 4

DBSCAN

1. DBSCAN does not require a pre-set number of clusters at all.
2. It also identifies outliers as noises, unlike mean-shift which simply throws them into a cluster even if the data point is very different.
3. Additionally, it can find arbitrarily sized and arbitrarily shaped clusters quite well.
4. The main drawback of DBSCAN is that it doesn’t perform as well as others when the clusters are of varying density. This is because the setting of the distance threshold ε and minPoints for identifying the neighborhood points will vary from cluster to cluster when the density varies.
5. This drawback also occurs with very high-dimensional data since again the distance threshold ε becomes challenging to estimate.

Gaussian Mixture Models

Gaussian Distribution

- Multivariate Gaussian

$$\mathcal{N}(\mathbf{x}|\underset{\text{mean}}{\boldsymbol{\mu}}, \underset{\text{covariance}}{\boldsymbol{\Sigma}}) = \frac{1}{(2\pi|\boldsymbol{\Sigma}|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

- Define precision to be the inverse of the covariance

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$$

- In 1-dimension

$$\tau = \frac{1}{\sigma^2}$$

Likelihood Function

- Data set $D = \{\mathbf{x}_n\} \quad n = 1, \ldots, N$

- Assume observed data points generated independently

$$p(D|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- Viewed as a function of the parameters, this is known as the *likelihood function*

Maximum Likelihood

- Set the parameters by maximizing the likelihood function
- Equivalently maximize the log likelihood

$$\begin{aligned}\ln p(D|\boldsymbol{\mu},\boldsymbol{\Sigma}) &= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2}\ln(2\pi) \\ &-\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n-\boldsymbol{\mu})\end{aligned}$$

Maximum Likelihood Solution

- Maximizing w.r.t. the mean gives the *sample mean*

$$\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$$

- Maximizing w.r.t covariance gives the *sample covariance*

$$\boldsymbol{\Sigma}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}})^{\mathrm{T}}$$

Gaussian Mixtures

- Linear super-position of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

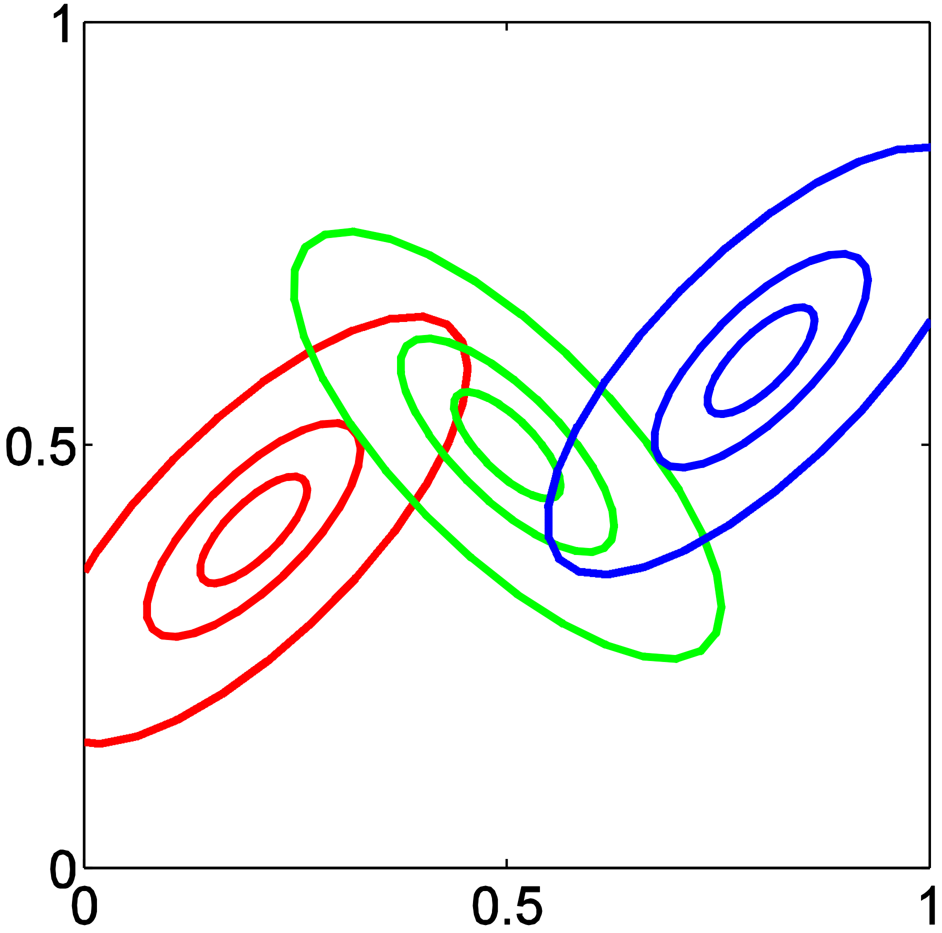
- Normalization and positivity require

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

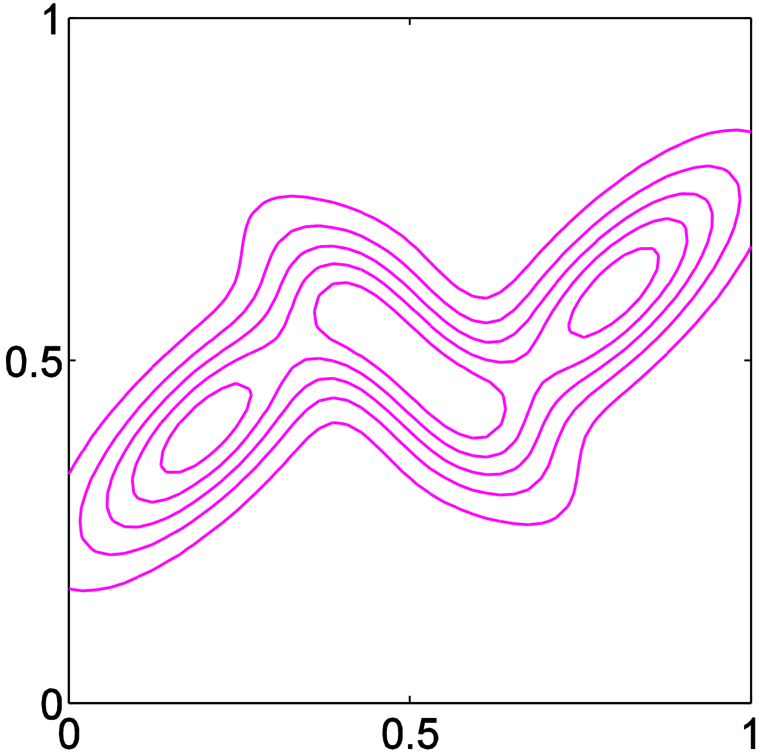
- Can interpret the mixing coefficients as prior probabilities

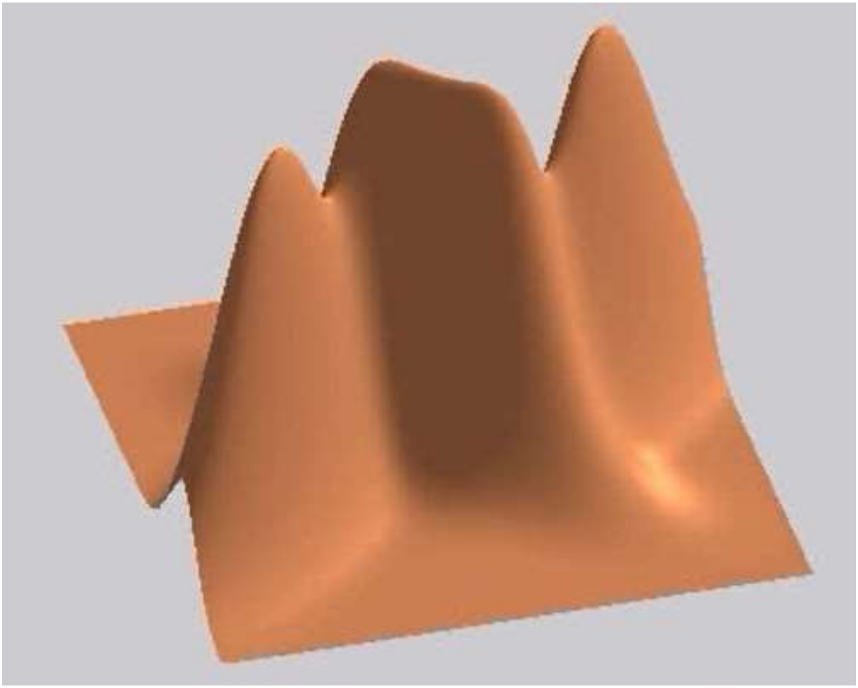
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k)$$

Example: Mixture of 3 Gaussians



Contours of Probability Distribution



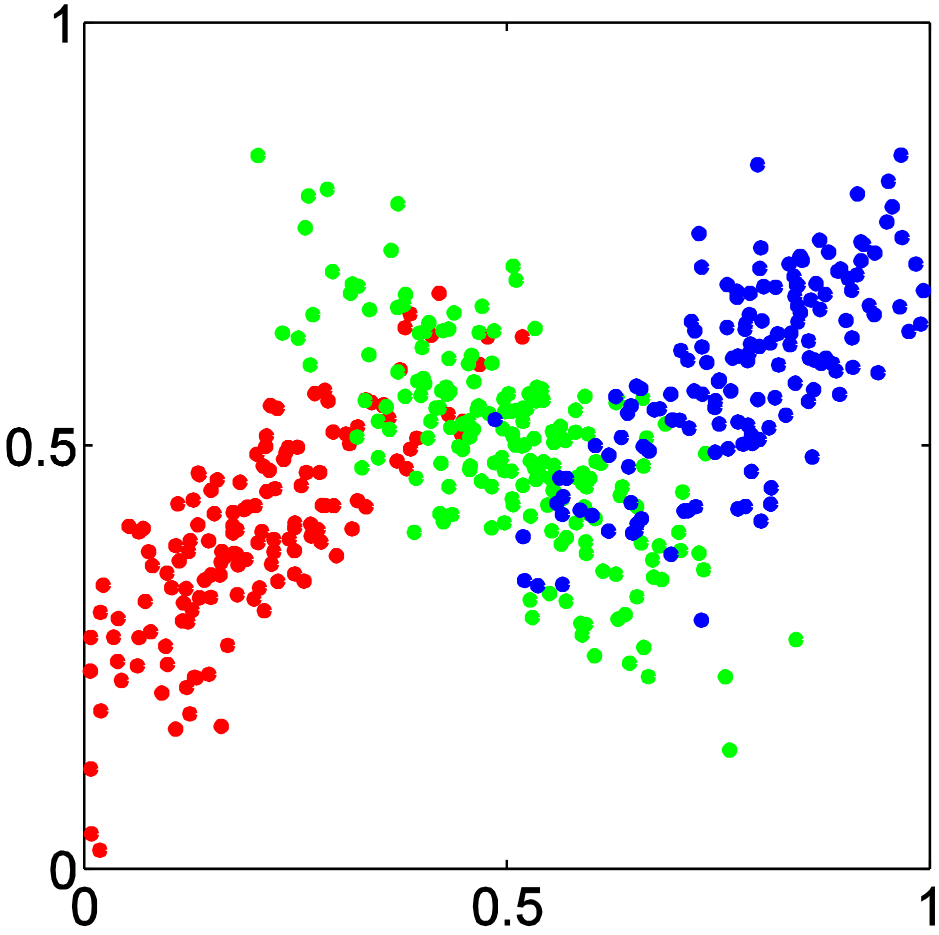


Sampling from the Gaussian

- To generate a data point:
  - first pick one of the components with probability π_k
  - then draw a sample $\mathbf{x}_n$ from that component

- Repeat these two steps for each new data point

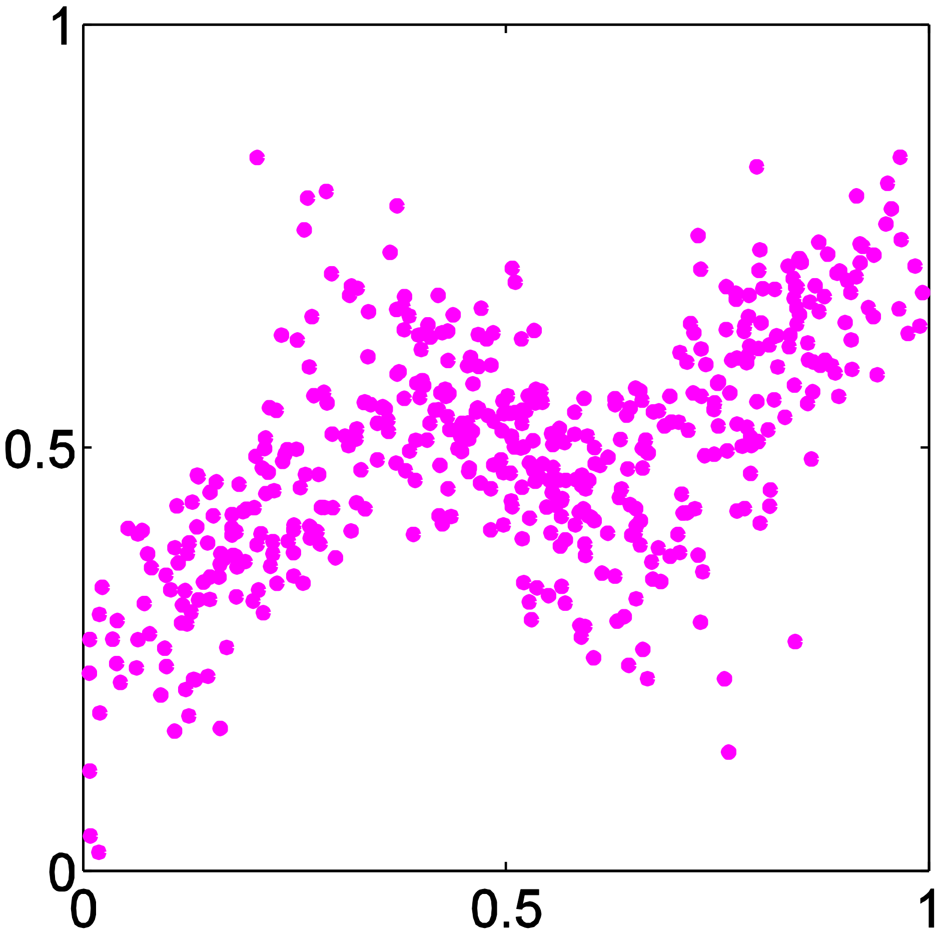
Synthetic Data Set



Fitting the Gaussian Mixture

- We wish to invert this process – given the data set, find the corresponding parameters:
  - mixing coefficients
  - means
  - covariances
- If we knew which component generated each data point, the maximum likelihood solution would involve fitting each component to the corresponding cluster
- Problem: the data set is unlabelled
- We shall refer to the labels as *latent* (= hidden) variables

Synthetic Data Set Without Labels

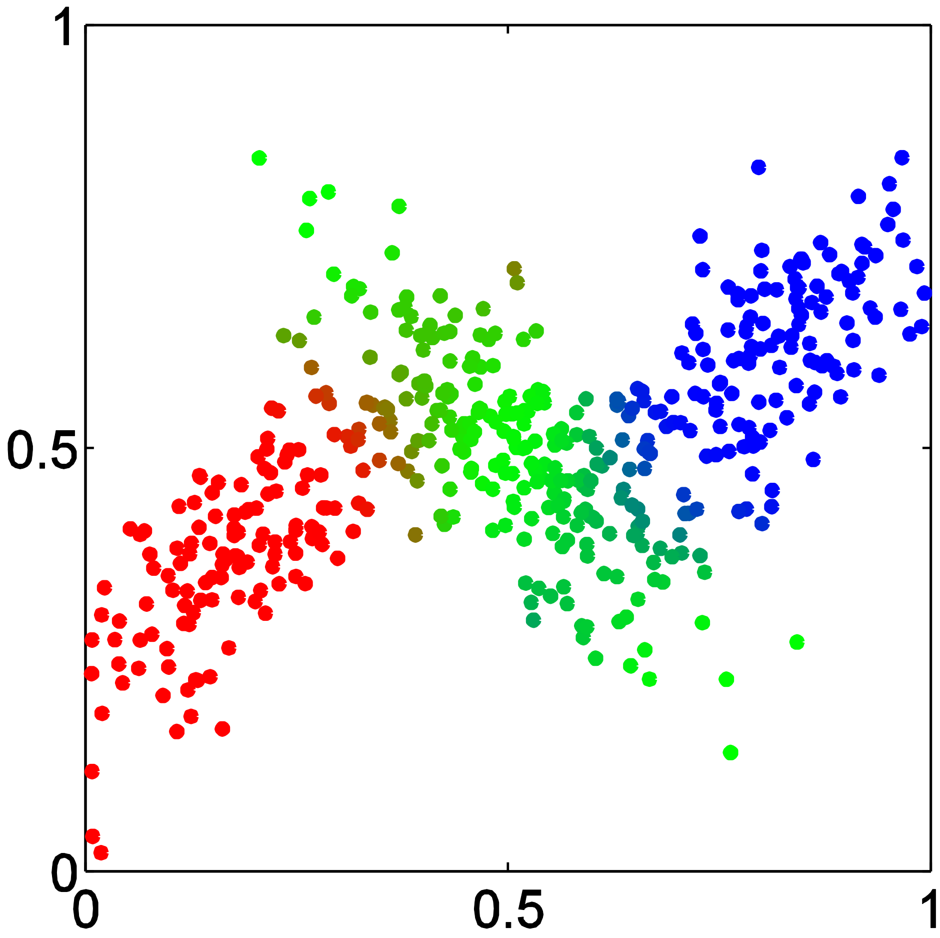


Posterior Probabilities

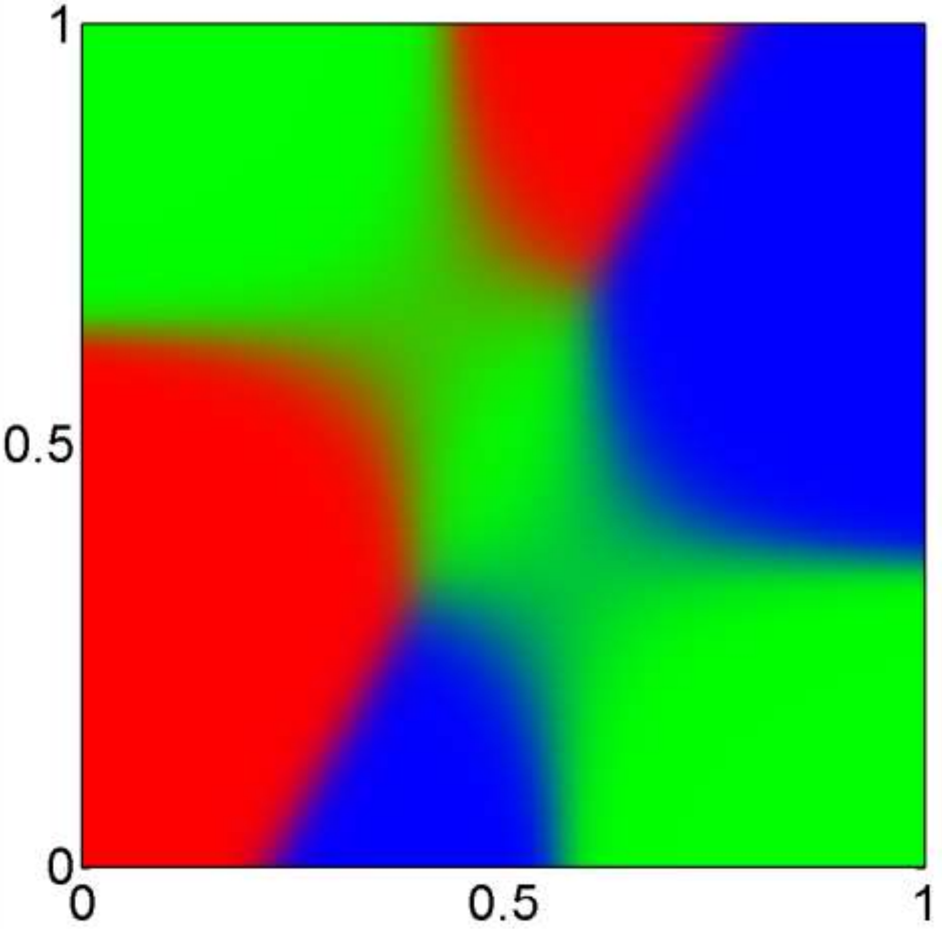
- We can think of the mixing coefficients as prior probabilities for the components
- For a given value of $\mathbf{x}$ we can evaluate the corresponding posterior probabilities, called *responsibilities*
- These are given from Bayes' theorem by

$$\begin{aligned}\gamma_k(\mathbf{x}) \equiv p(k|\mathbf{x}) &= \frac{p(k)p(\mathbf{x}|k)}{p(\mathbf{x})} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\end{aligned}$$

Posterior Probabilities (colour coded)



Posterior Probability Map



Maximum Likelihood for the GMM

- The log likelihood function takes the form

$$\ln p(D|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Note: sum over components appears *inside* the log
- There is no closed form solution for maximum likelihood

Problems and Solutions

- How to maximize the log likelihood
  - solved by expectation-maximization (EM) algorithm
- How to avoid singularities in the likelihood function
  - solved by a Bayesian treatment
- How to choose number K of components
  - also solved by a Bayesian treatment

EM Algorithm – Informal Derivation

- Let us proceed by simply differentiating the log likelihood
- Setting derivative with respect to $\boldsymbol{\mu}_j$ equal to zero gives

$$-\sum_{n=1}^{N} \underbrace{\frac{\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_k \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}}_{\gamma_j(\mathbf{x}_n)} \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j) = 0$$

giving

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

which is simply the weighted mean of the data

EM Algorithm – Informal Derivation

- Similarly for the covariances

$$\Sigma_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^\top}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

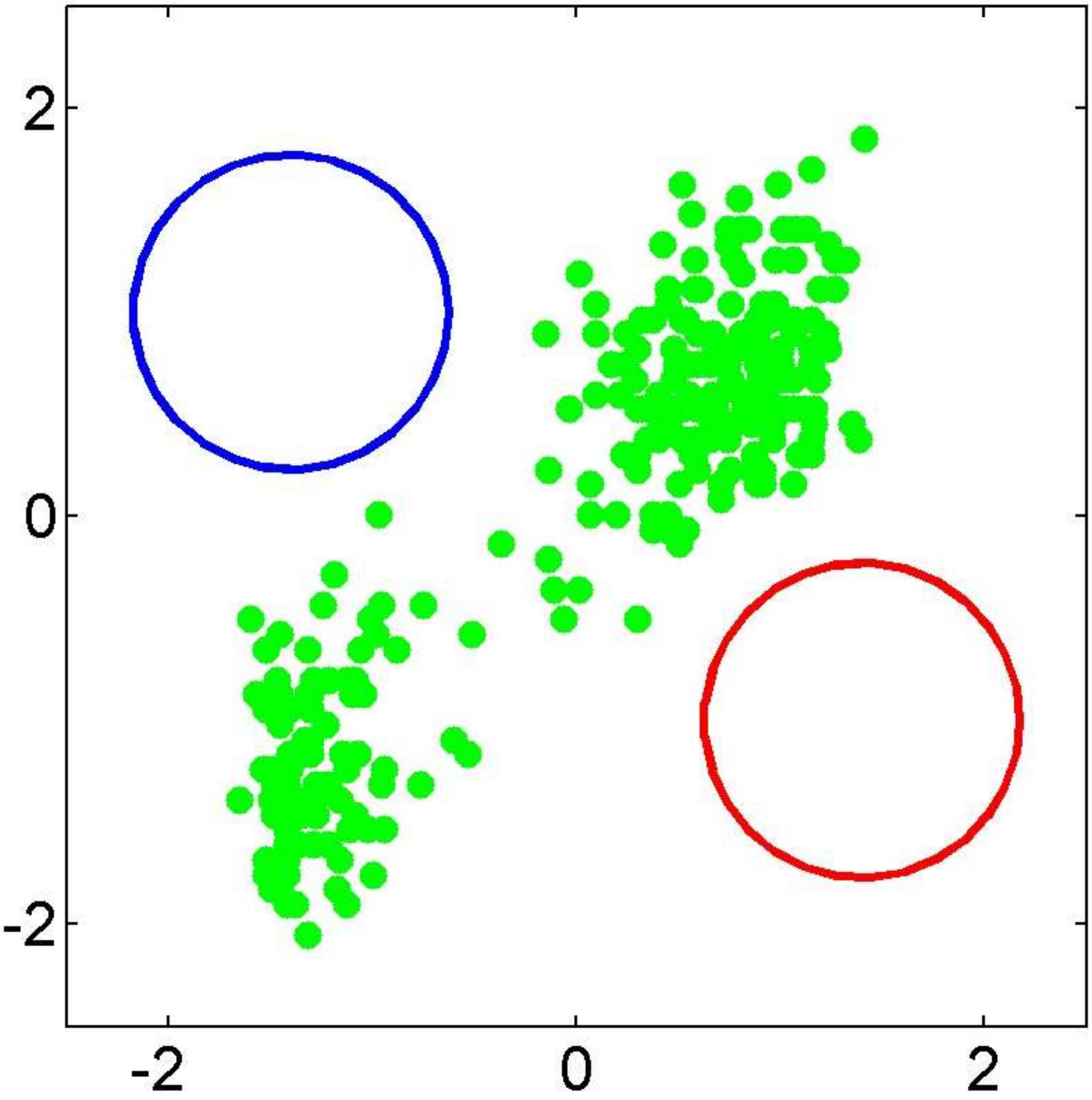
- For mixing coefficients use a Lagrange multiplier to give

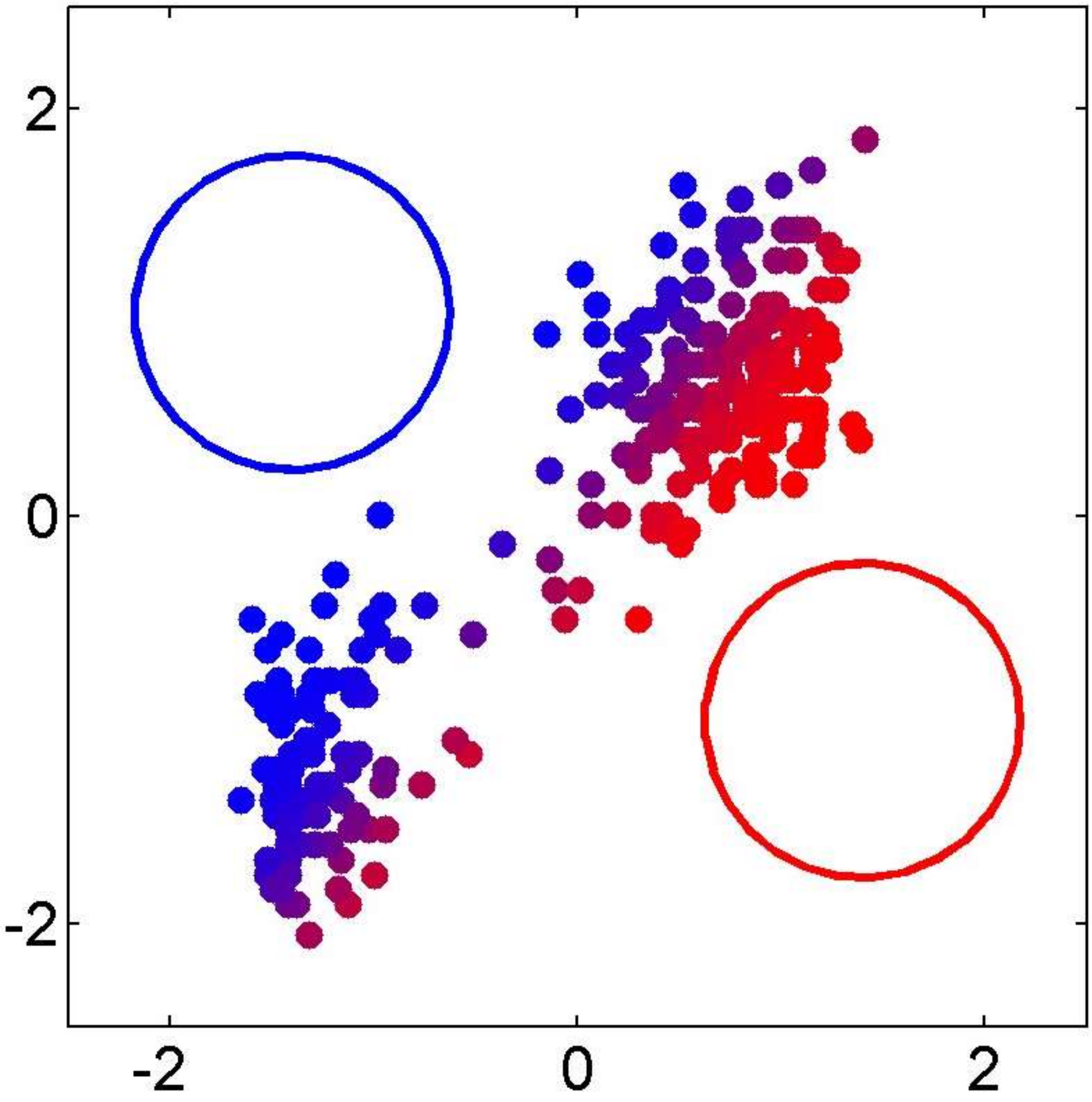
$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)$$

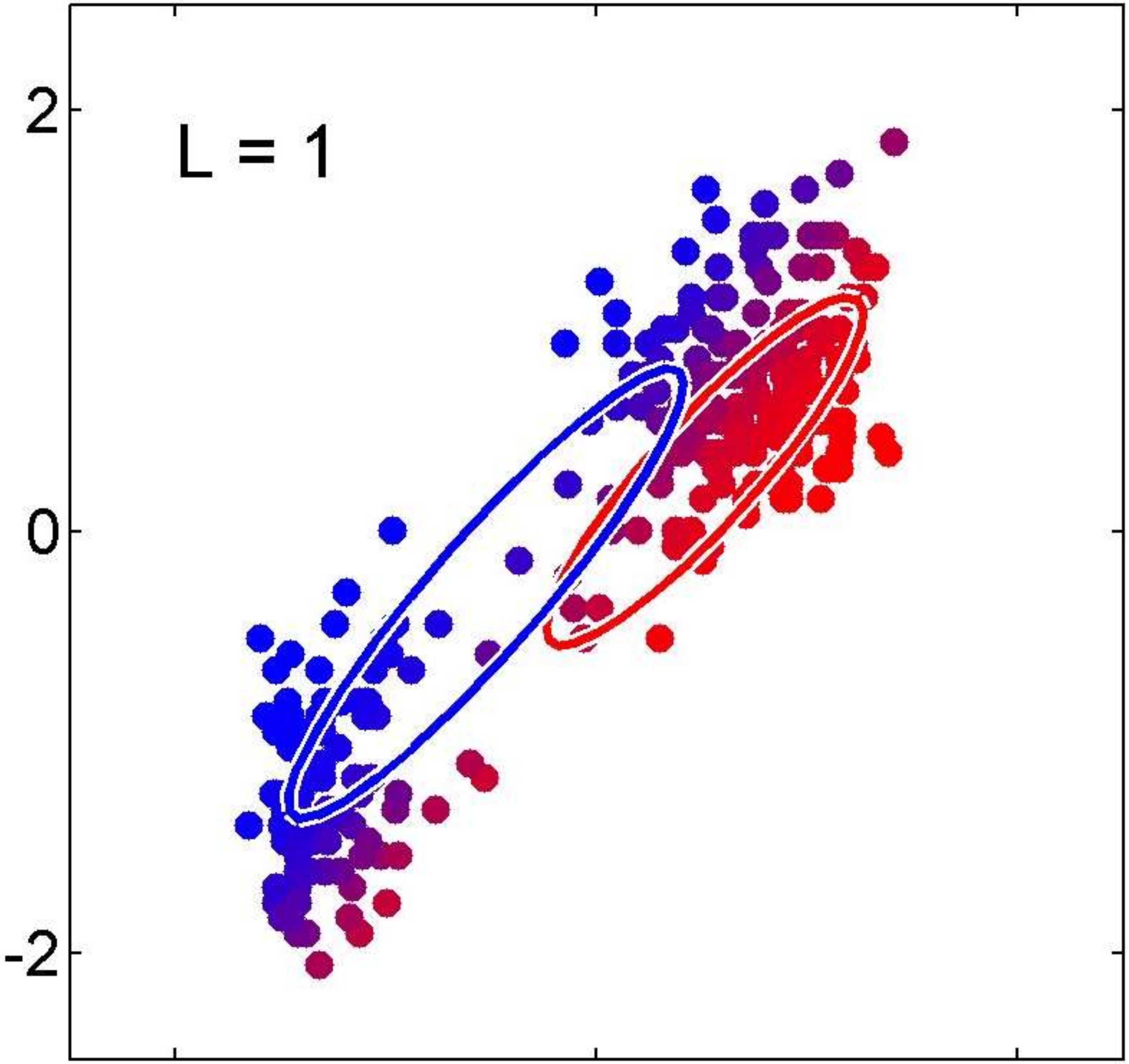
EM Algorithm – Informal Derivation

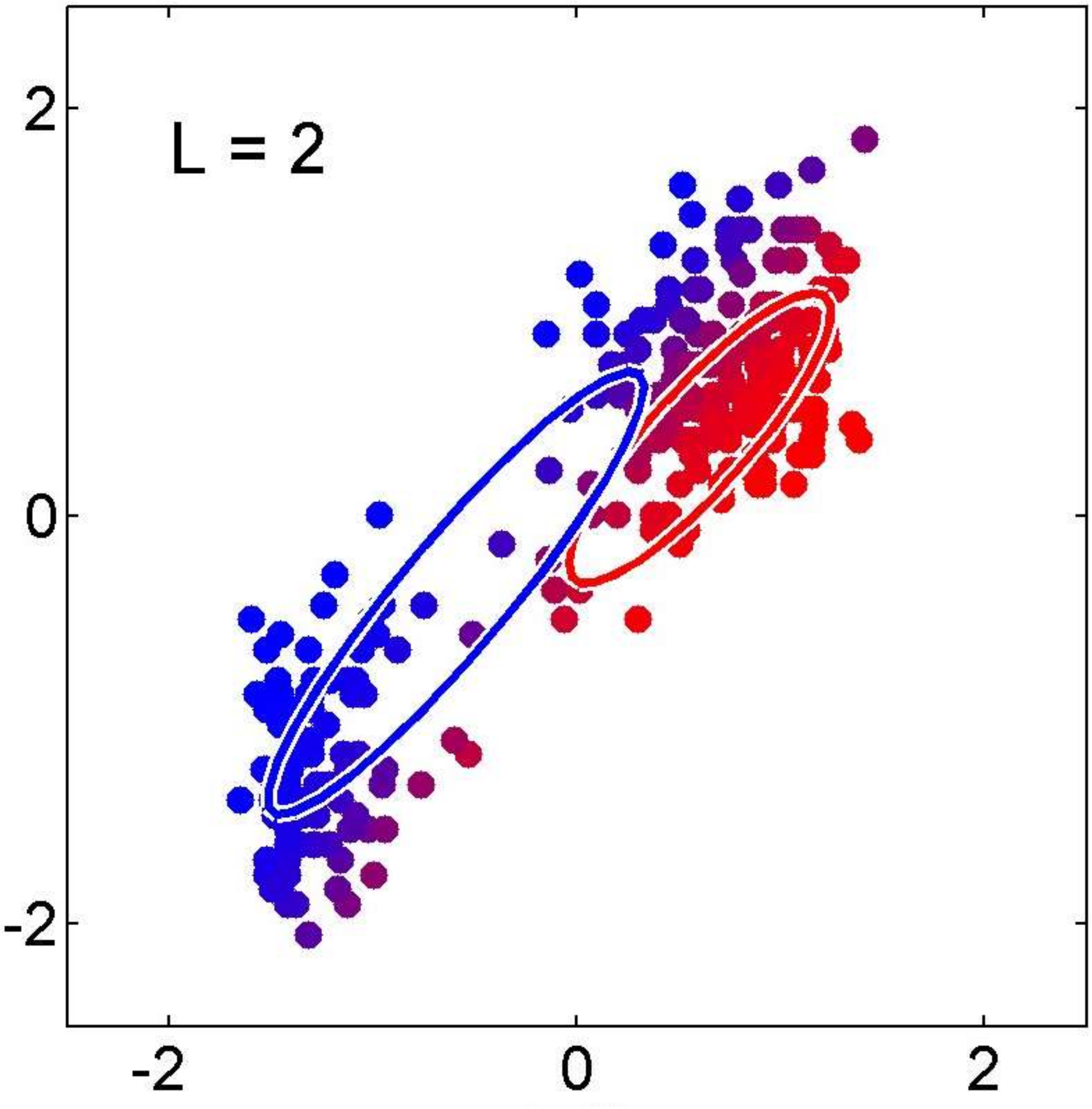
- The solutions are not closed form since they are coupled

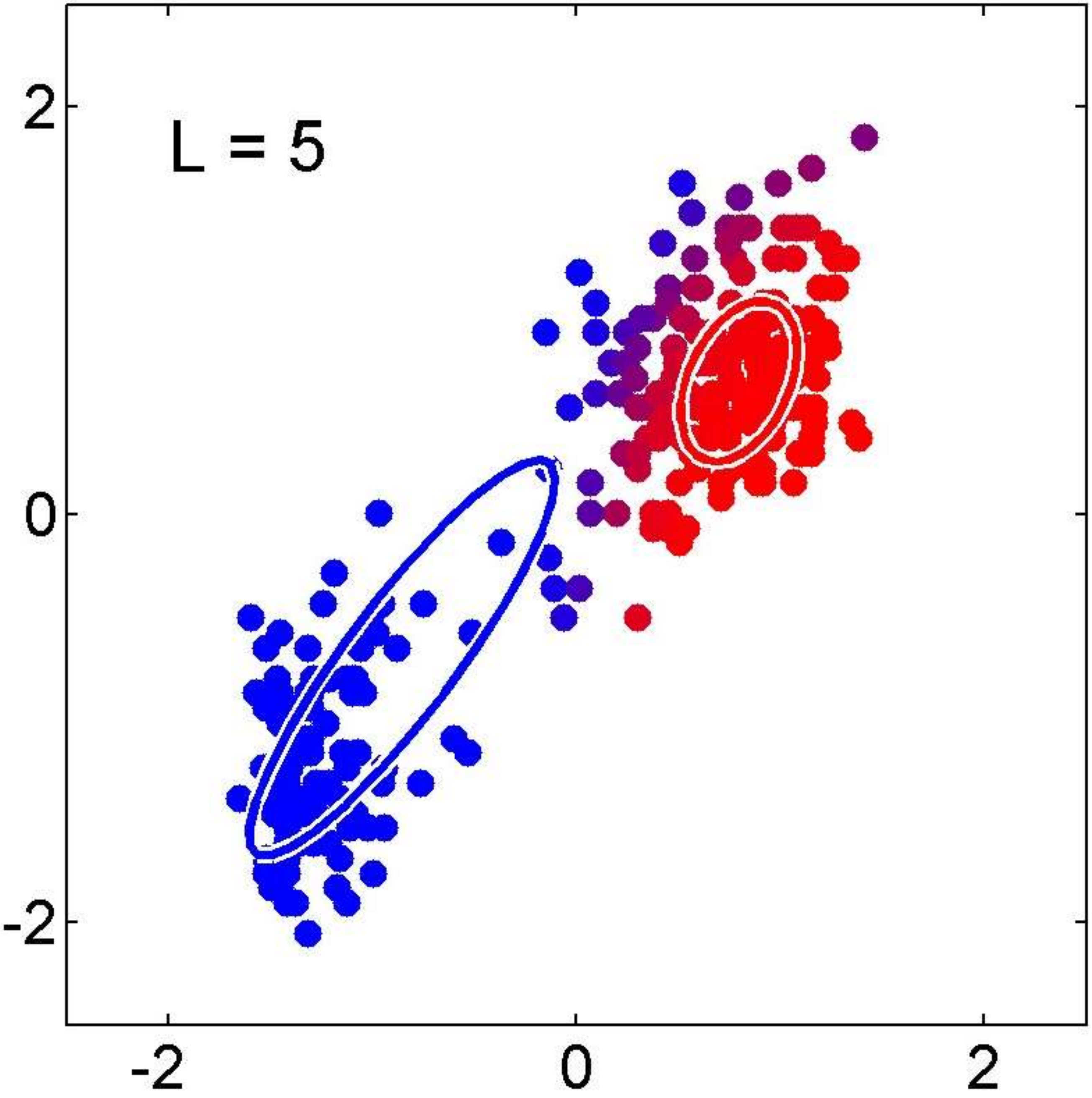
- Suggests an iterative scheme for solving them:
  - Make initial guesses for the parameters
  - Alternate between the following two stages:
    1. E-step: evaluate responsibilities
    2. M-step: update parameters using ML results

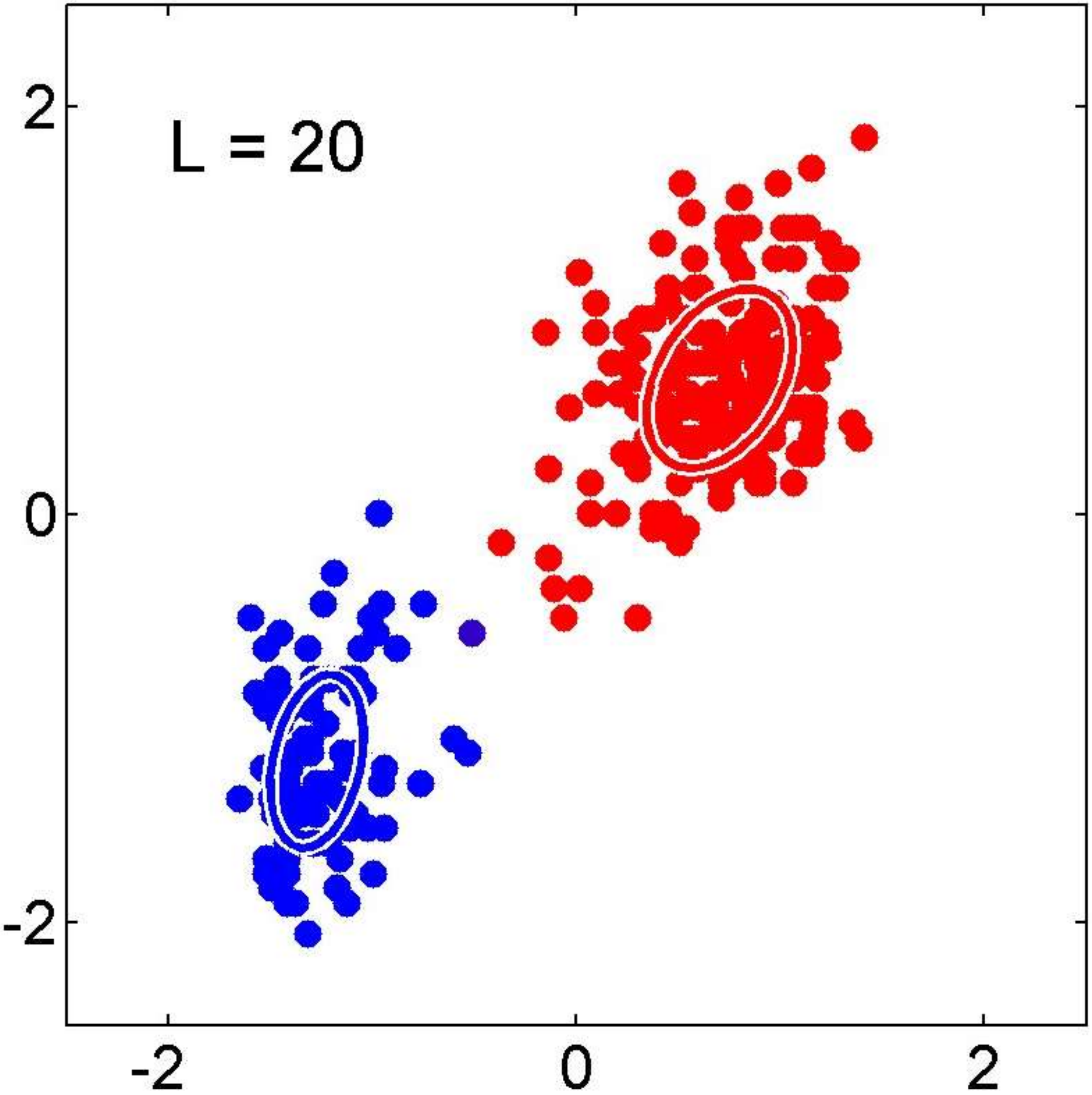

2
0
-2
-2
0
2


2
0
-2
-2
0
2


L = 1
2
0
-2
-2
0
2


L = 2
2
0
-2
-2
0
2


L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

GMM Algorithm Summary

K mixture components and *N* samples

E-step

$$\gamma_k(\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

M-step

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

$$\boldsymbol{\Sigma}_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^\top}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

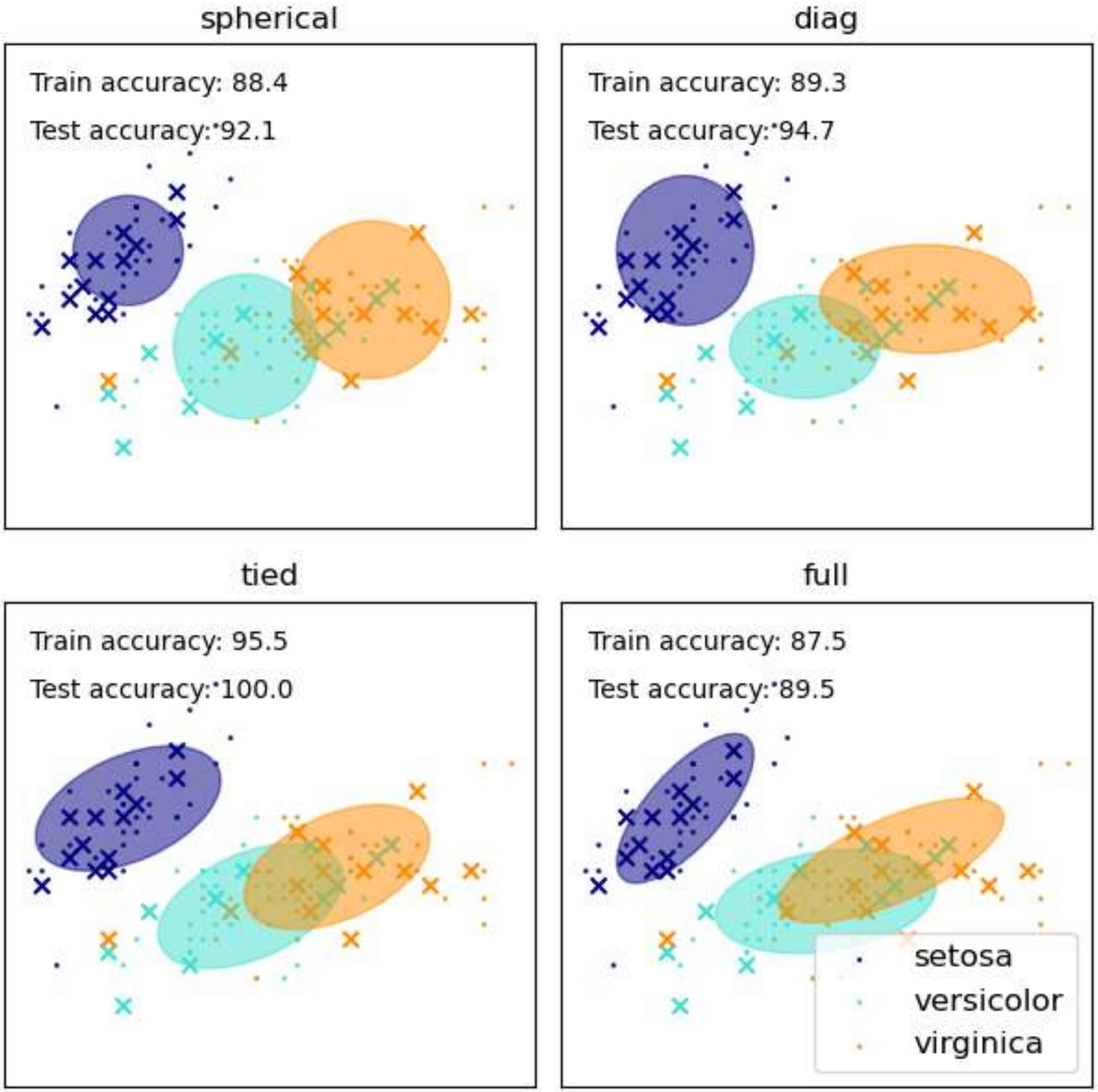
$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)$$

GMM Problems

- How to select K ?

- EM converge to local optimum
  - Initialization is important

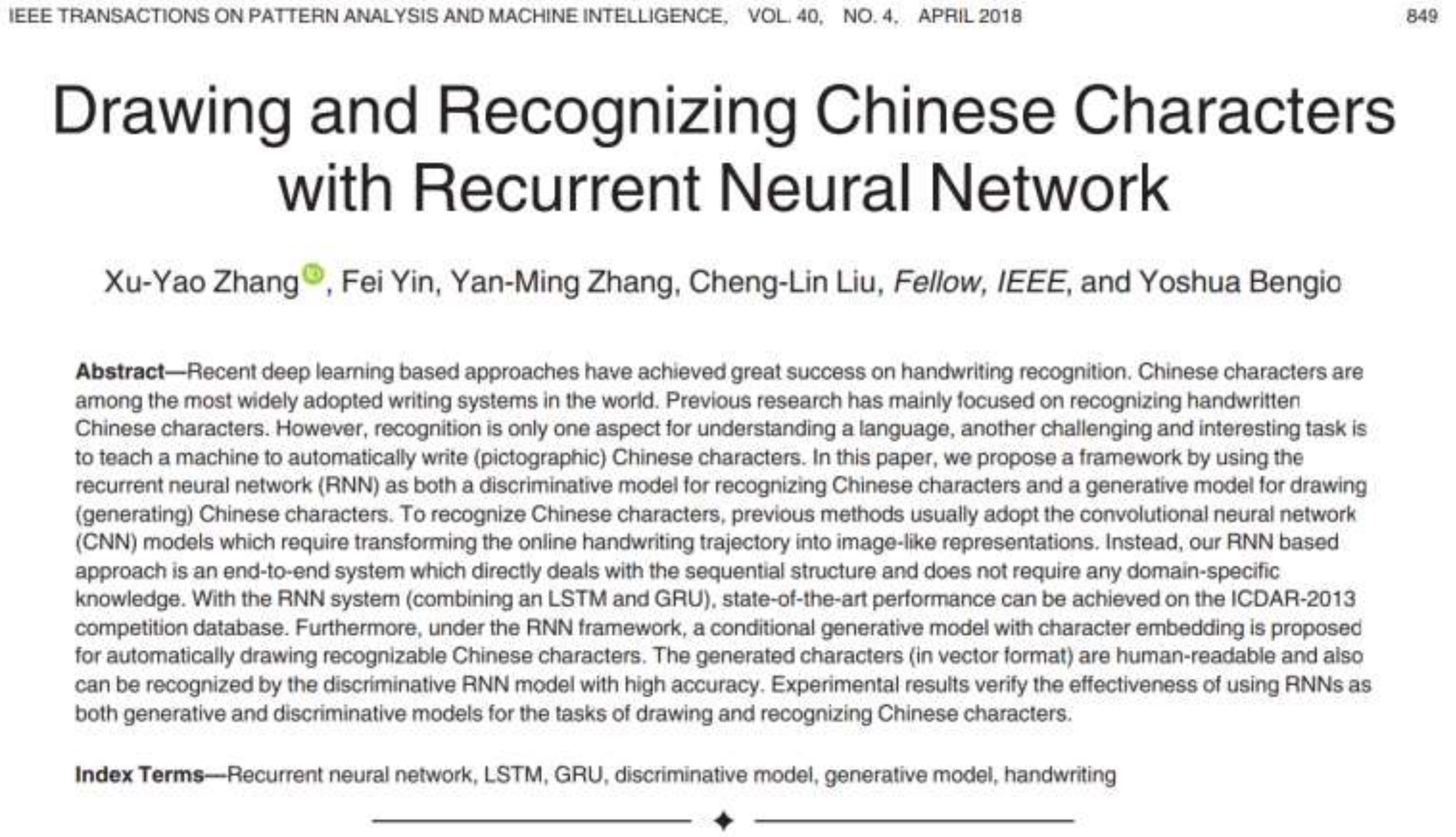
- Regularization of covariance matrix
  - identity matrix
  - diagonal matrix
  - shared among different components


spherical
Train accuracy: 88.4
Test accuracy: 92.1
diag
Train accuracy: 89.3
Test accuracy: 94.7
tied
Train accuracy: 95.5
Test accuracy: 100.0
full
Train accuracy: 87.5
Test accuracy: 89.5
setosa
versicolor
virginica

Applications of GMM

- As a soft extension of K-means
- Density estimation in Bayes classification
- GMM + HMM in speech recognition
- Data generalization using GMM
- As a general assumption for any complex distributions
- Many other applications in ML, CV, NLP ……

Using GMM for Generation



IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 40, NO. 4, APRIL 2018 849

Drawing and Recognizing Chinese Characters with Recurrent Neural Network

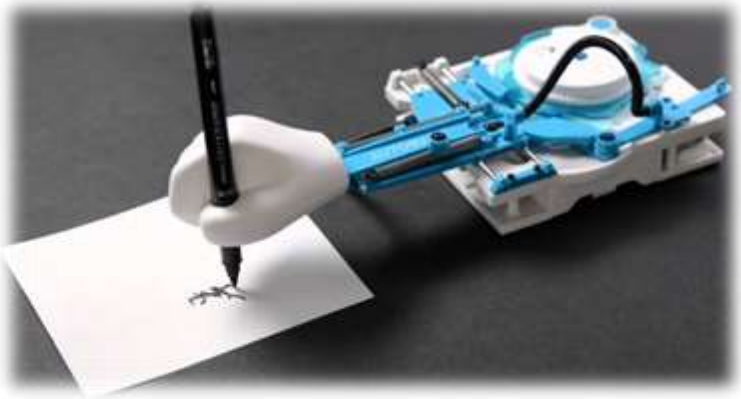
Xu-Yao Zhang, Fei Yin, Yan-Ming Zhang, Cheng-Lin Liu, *Fellow, IEEE*, and Yoshua Bengio

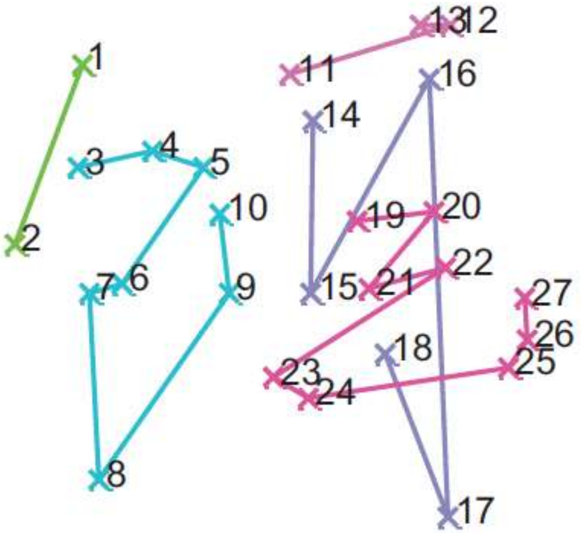
Abstract—Recent deep learning based approaches have achieved great success on handwriting recognition. Chinese characters are among the most widely adopted writing systems in the world. Previous research has mainly focused on recognizing handwritten Chinese characters. However, recognition is only one aspect for understanding a language, another challenging and interesting task is to teach a machine to automatically write (pictographic) Chinese characters. In this paper, we propose a framework by using the recurrent neural network (RNN) as both a discriminative model for recognizing Chinese characters and a generative model for drawing (generating) Chinese characters. To recognize Chinese characters, previous methods usually adopt the convolutional neural network (CNN) models which require transforming the online handwriting trajectory into image-like representations. Instead, our RNN based approach is an end-to-end system which directly deals with the sequential structure and does not require any domain-specific knowledge. With the RNN system (combining an LSTM and GRU), state-of-the-art performance can be achieved on the ICDAR-2013 competition database. Furthermore, under the RNN framework, a conditional generative model with character embedding is proposed for automatically drawing recognizable Chinese characters. The generated characters (in vector format) are human-readable and also can be recognized by the discriminative RNN model with high accuracy. Experimental results verify the effectiveness of using RNNs as both generative and discriminative models for the tasks of drawing and recognizing Chinese characters.

Index Terms—Recurrent neural network, LSTM, GRU, discriminative model, generative model, handwriting

Discriminative vs Generative

- Reading: discriminative model
- Writing: generative model
- Writing is more difficult than reading
- Use RNN as both:
    - discriminative model for online HCCR
    - generative model for drawing characters

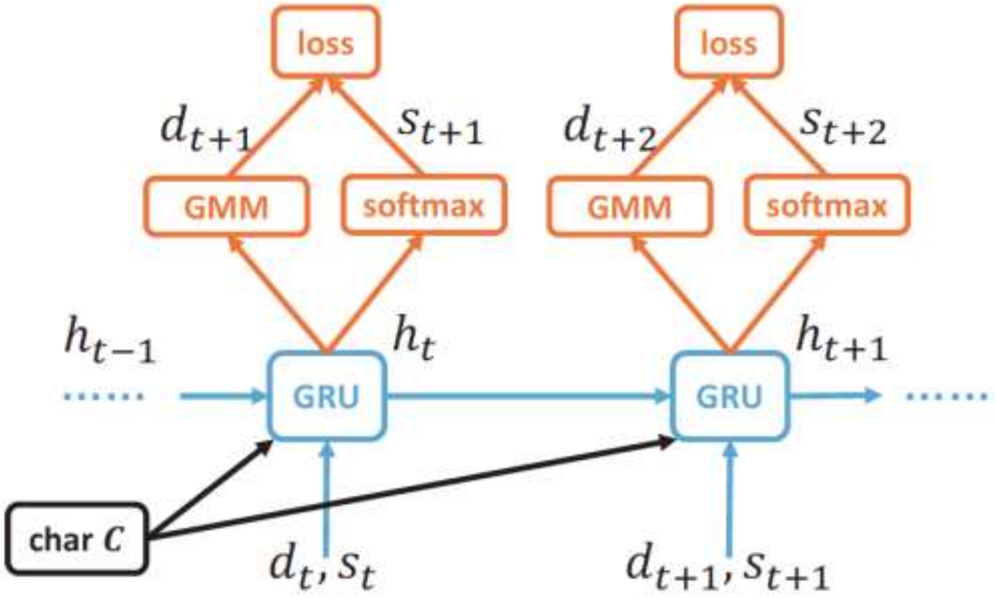




Xu-Yao Zhang et al, Drawing and Recognizing Chinese Characters with Recurrent Neural Network, IEEE Trans. PAMI, 2018.

Generative RNN model: training process

- Condition on character embedding ***c***
- Softmax for pen-state ***s***
    - pen-down
    - pen-up
    - end-of-char
- GMM for pen-direction ***d***
- Weighted loss function:
    - GMM: likelihood ***d***
    - Softmax: cross-entropy ***s***



$$\text{loss} = -\sum_t \left\{ \log\left(\mathrm{P}_d(d_{t+1})\right) + \sum_{i=1}^{3} w^i s^i_{t+1} \log(p^i_{t+1}) \right\},$$

GMM for Pen-direction

$$\left(\{\hat{\pi}^j, \hat{\mu}_x^j, \hat{\mu}_y^j, \hat{\delta}_x^j, \hat{\delta}_y^j\}_{j=1}^{M}\right) \in \mathbb{R}^{5M} = W_{\text{gmm}} \times o_t + b_{\text{gmm}},$$

$$\pi^j = \frac{\exp(\hat{\pi}^j)}{\sum_{j'} \exp(\hat{\pi}^{j'})} \Rightarrow \pi^j \in (0,1), \sum_j \pi^j = 1,$$

$$\mu_x^j = \hat{\mu}_x^j \Rightarrow \mu_x^j \in \mathbb{R},$$

$$\mu_y^j = \hat{\mu}_y^j \Rightarrow \mu_y^j \in \mathbb{R},$$

$$\delta_x^j = \exp\left(\hat{\delta}_x^j\right) \Rightarrow \delta_x^j > 0,$$

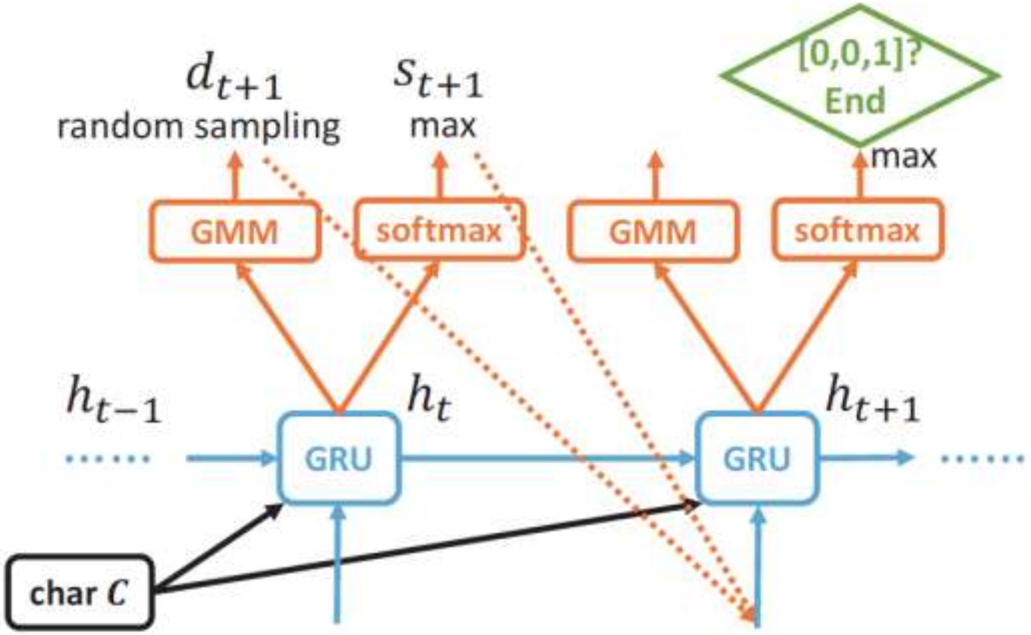
$$\delta_y^j = \exp\left(\hat{\delta}_y^j\right) \Rightarrow \delta_y^j > 0.$$

$$\mathcal{N}(x|\mu,\delta) = \frac{1}{\delta\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\delta^2}\right).$$

$$\begin{aligned} \mathrm{P}_d(d_{t+1}) &= \sum_{j=1}^{M} \pi^j \mathcal{N}\left(d_{t+1} | \mu_x^j, \mu_y^j, \delta_x^j, \delta_y^j\right) \\ &= \sum_{j=1}^{M} \pi^j \mathcal{N}(\triangle x_{t+1} | \mu_x^j, \delta_x^j) \mathcal{N}\left(\triangle y_{t+1} | \mu_y^j, \delta_y^j\right), \end{aligned}$$

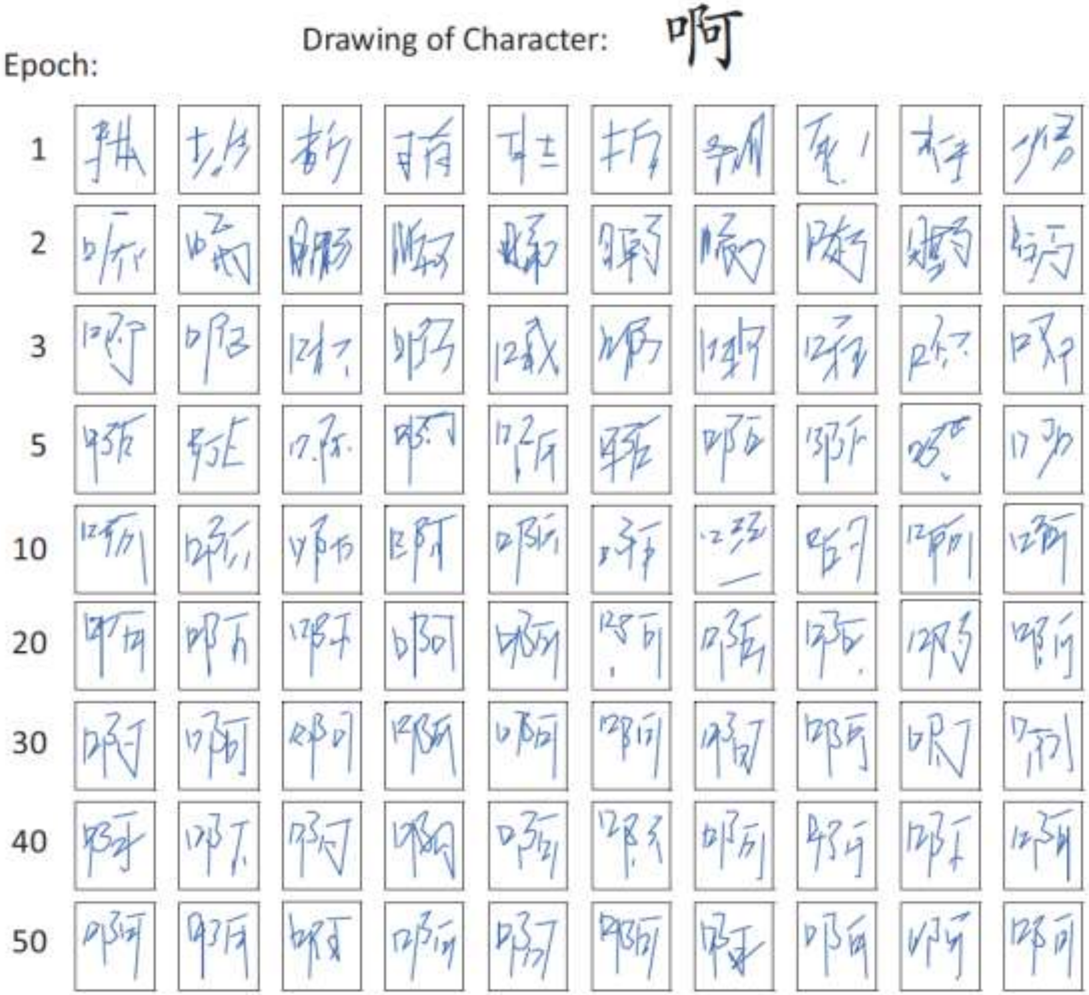
Generative RNN model: writing process

- Given a character index
- Choose its embedding vector ***c***
- Feed it into GRU
- Sampling a pen-movement from GMM ***d***
- Decide pen-state via softmax ***s***
- Update hidden state ***h***
- Feed ***cdsh*** to next step
- Continue writing until reach “end-of-char” state



Generative RNN model

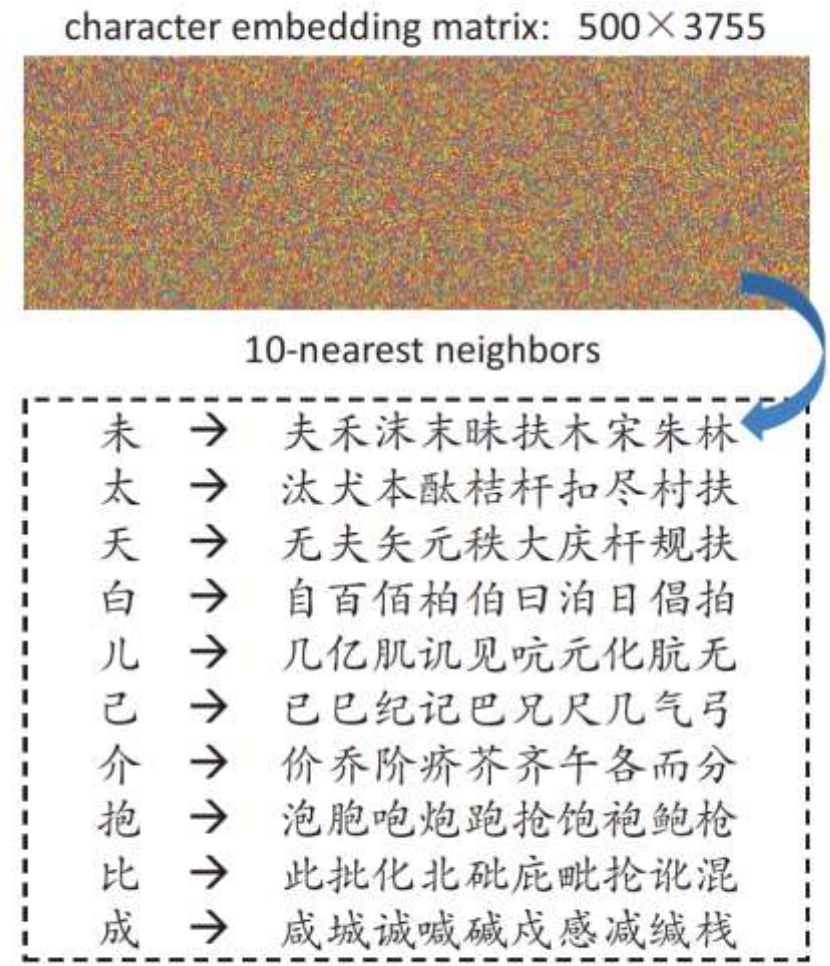
- First few epochs, it is just random drawing
- After 30 epochs, the character can be correctly written



Character embedding matrix

- 3755 character embed in 500D space
  - a 500x3755 matrix
- Nearest neighbor via Euclidean distance:
  - Similar characters are grouped
  - Automatically find structure during writing

Note we did not provide similar-character info
This is automatically discovered during generation

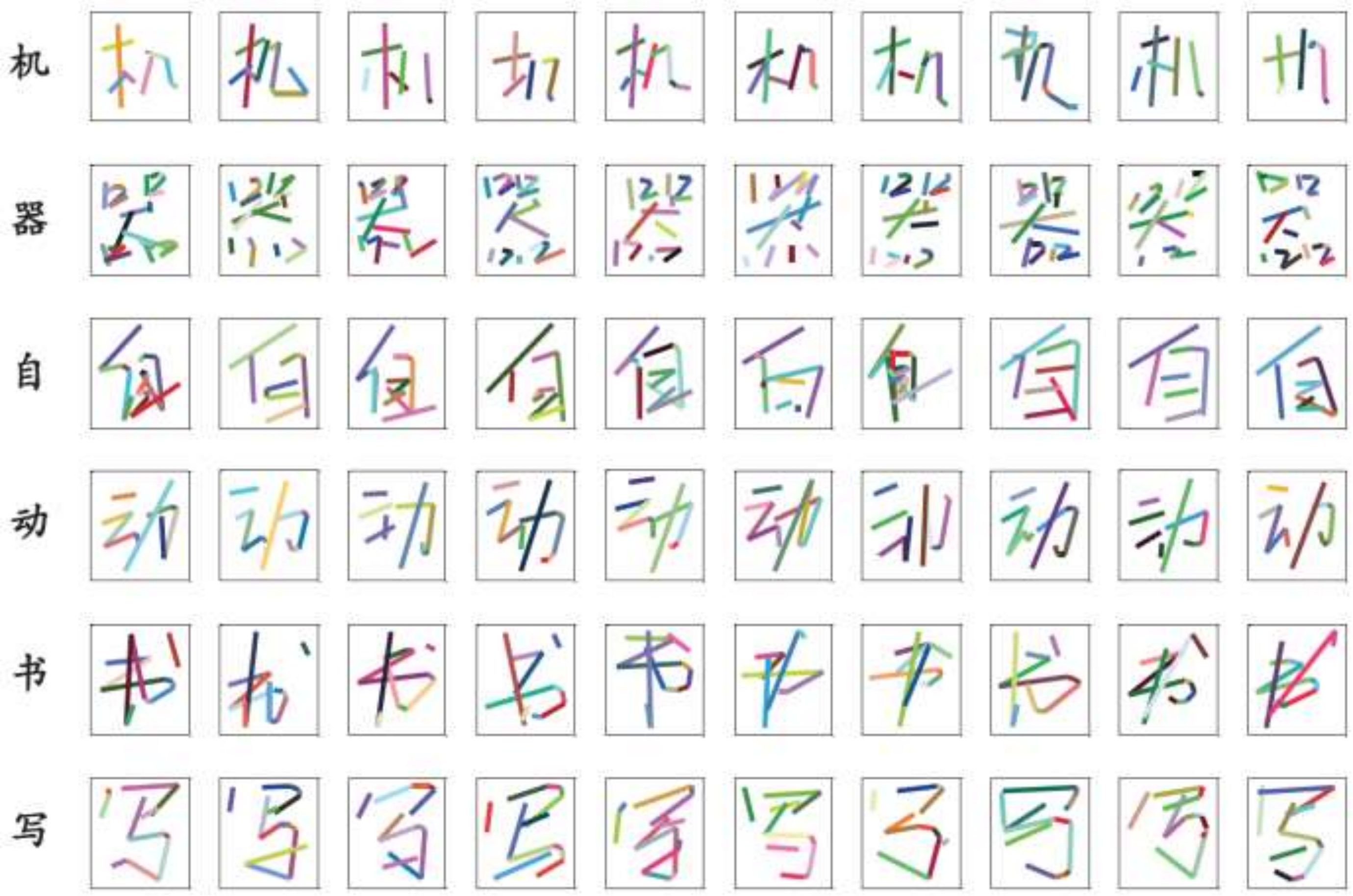


character embedding matrix: 500×3755

10-nearest neighbors

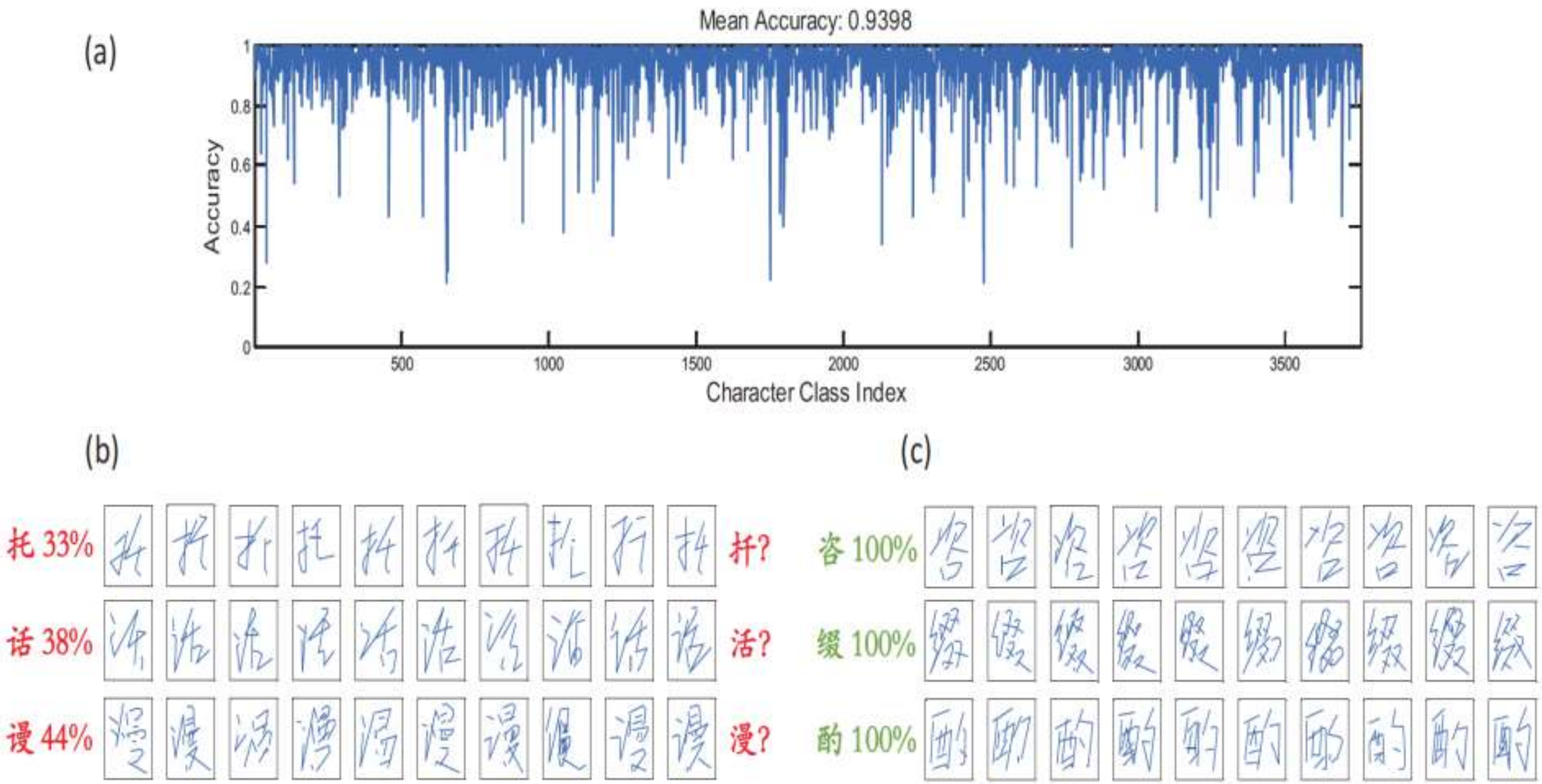
未	→	夫禾沫末昧扶木宋朱林
太	→	汰犬本酞桔杆扣尽村扶
天	→	无夫矢元秩大庆杆规扶
白	→	自百佰柏伯曰泊日倡拍
儿	→	几亿肌讥见吭元化肮无
己	→	已巳纪记巴兄尺几气弓
介	→	价乔阶疥芥齐午各而分
抱	→	泡胞咆炮跑抢饱袍鲍枪
比	→	此批化北砒庇毗抡讹混
成	→	咸城诚喊碱戌感减缄栈

Generative RNN model: examples



- Write 3755 characters: Character embedding
- Correctly writing: Softmax pen-state
- Diversity in style: GMM pen-direction

Generative RNN model: examples



YOUKU
Alibaba Cloud
Alibaba Cloud

Thank All of You!

(Questions?)