



香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP 5212

Machine Learning

Lecture 10

Unsupervised Learning: Clustering, Expectation Maximization

Junxian He
Mar 17, 2026

Review: MLE vs. MAP

Maximum Likelihood estimation (MLE)

Choose value that maximizes the probability of observed data

$$\hat{\theta}_{MLE} = \arg \max_{\theta} P(D|\theta)$$

Maximum *a posteriori* (MAP) estimation

Choose value that is most probable given observed data and prior belief

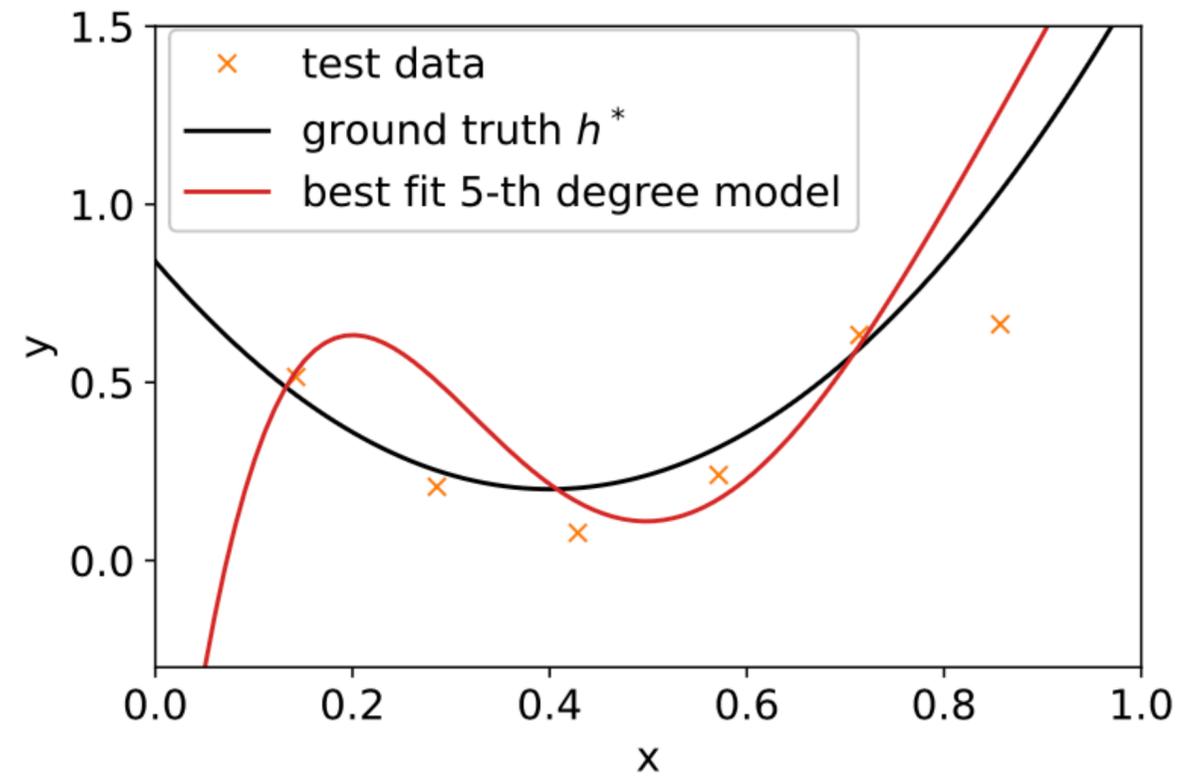
$$\begin{aligned}\hat{\theta}_{MAP} &= \arg \max_{\theta} P(\theta|D) \\ &= \arg \max_{\theta} P(D|\theta)P(\theta)\end{aligned}$$

When are they the same?

Recap: Generalization



Zero training error



Large test error

Train, Validation, Test

Validation dataset is another set of pairs $\{(\hat{x}^{(1)}, \hat{y}^{(1)}), \dots, (\hat{x}^{(m)}, \hat{y}^{(m)})\}$

Does not overlap with training dataset

Train, Validation, Test

Validation dataset is another set of pairs $\{(\hat{x}^{(1)}, \hat{y}^{(1)}), \dots, (\hat{x}^{(m)}, \hat{y}^{(m)})\}$

Does not overlap with training dataset

Test dataset is another set of pairs $\{(\tilde{x}^{(1)}, \tilde{y}^{(1)}), \dots, (\tilde{x}^{(L)}, \tilde{y}^{(L)})\}$

Does not overlap with training and validation dataset

Train, Validation, Test

pseudo-test

Validation dataset is another set of pairs $\{(\hat{x}^{(1)}, \hat{y}^{(1)}), \dots, (\hat{x}^{(m)}, \hat{y}^{(m)})\}$

Does not overlap with training dataset

Test dataset is another set of pairs $\{(\tilde{x}^{(1)}, \tilde{y}^{(1)}), \dots, (\tilde{x}^{(L)}, \tilde{y}^{(L)})\}$

Does not overlap with training and validation dataset

Completely unseen before deployment

Realistic setting

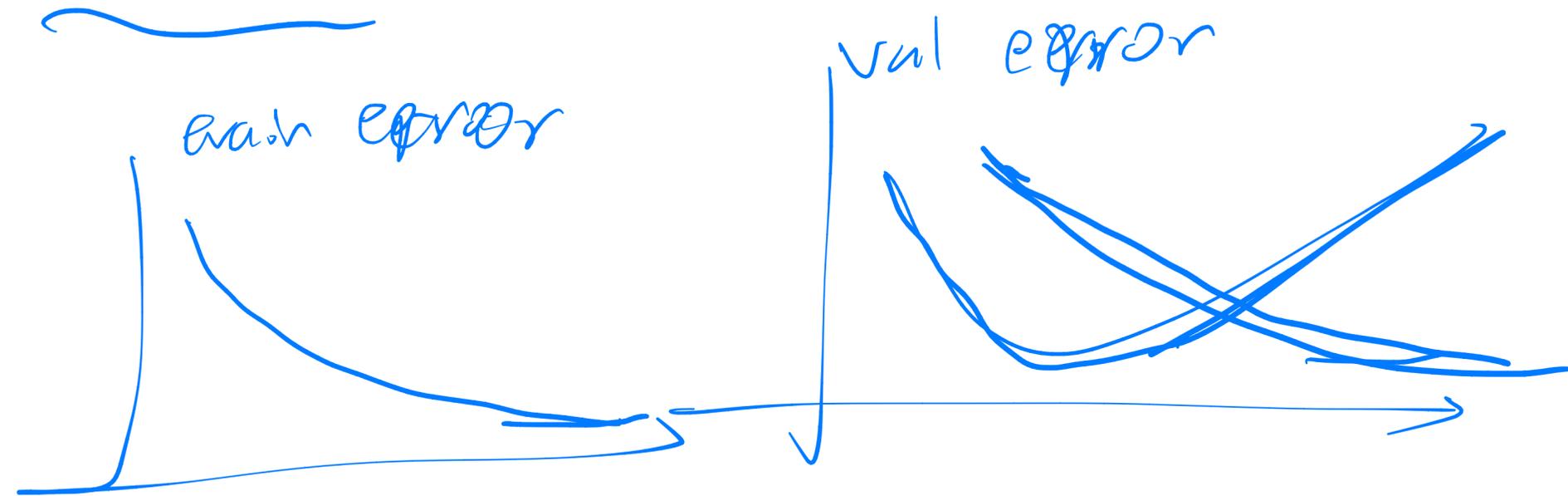
Validation is Very Important

Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)

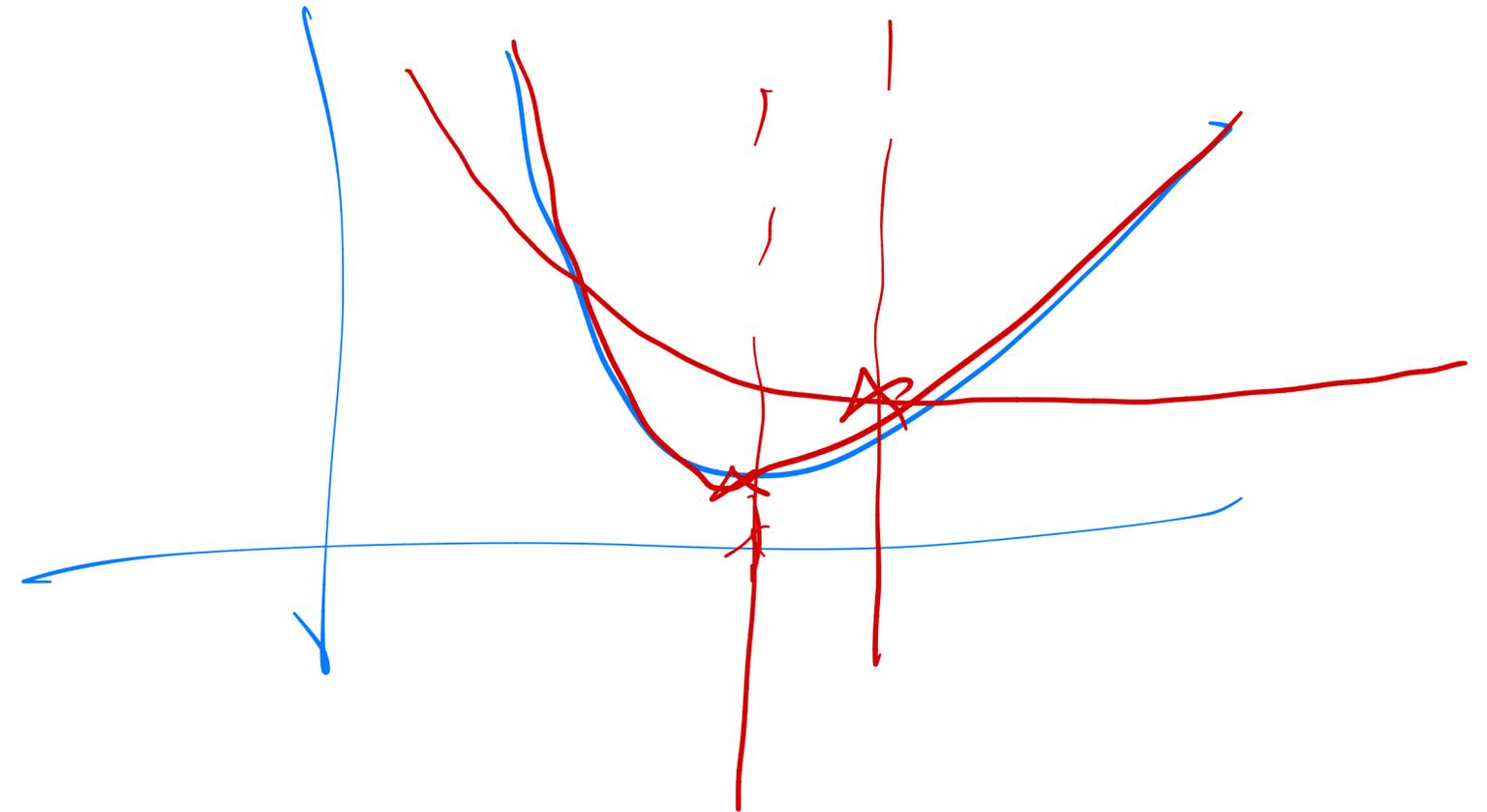
Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)



Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)
- Decide when to stop training



Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)
- Decide when to stop training

Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)
- Decide when to stop training
- Select hyperparameters

Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)
- Decide when to stop training
- Select hyperparameters

Hyperparameter tuning

Validation is Very Important

● Track underfitting/overfitting (in case of iterative training)

● Decide when to stop training

● Select hyperparameters

Hyperparameter tuning

When you tune hyperparameters harder, it is more likely the validation error would mismatch the test error, because you are overfitting on the validation

{ 1, 2, 3, 4, 5 }

{ 1, 1.1, 1.2, ..., 1.9, 1.95, ..., 5 }

manually

Validation is Very Important

- Track underfitting/overfitting (in case of iterative training)
- Decide when to stop training
- Select hyperparameters

Hyperparameter tuning

When you tune hyperparameters harder, it is more likely the validation error would mismatch the test error, because you are overfitting on the validation

Hyperparameter tuning is a form of training

Good ML Practice

Good ML Practice

- Do not look at or evaluate on the test dataset

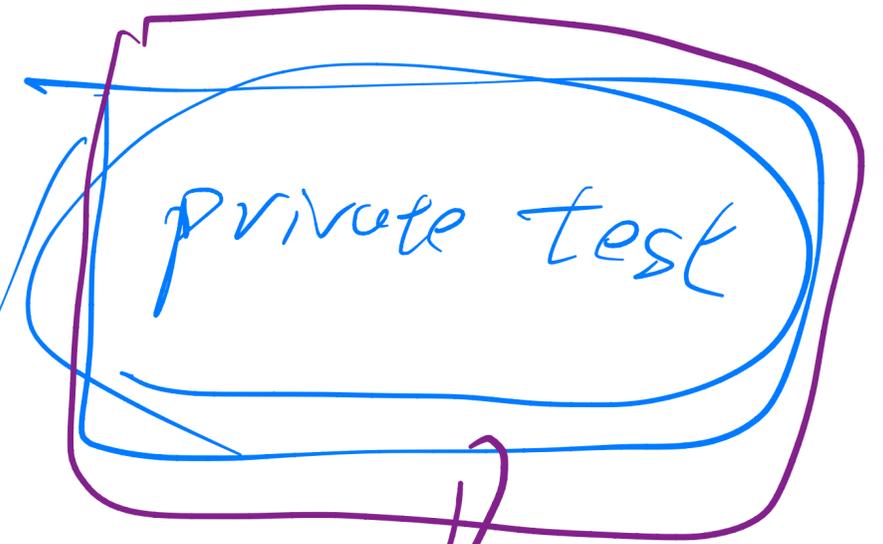
Good ML Practice

- Do not look at or evaluate on the test dataset
- Always track the training and validation metrics/errors/losses

Good ML Practice

Kaggle

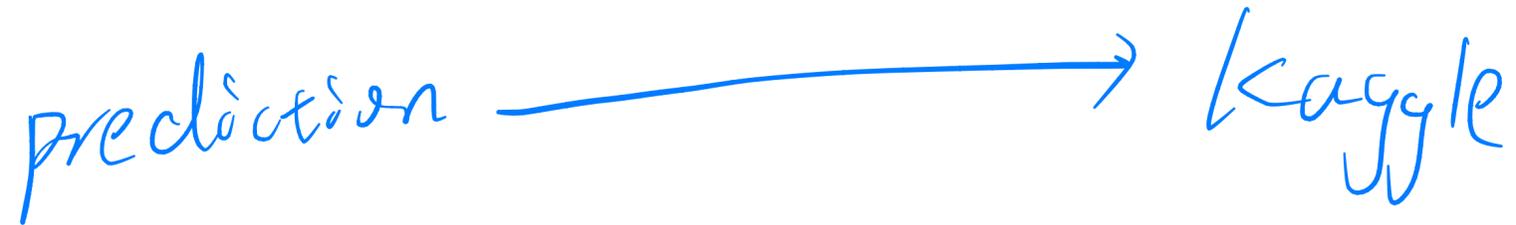
public test



- Do not look at or evaluate on the test dataset

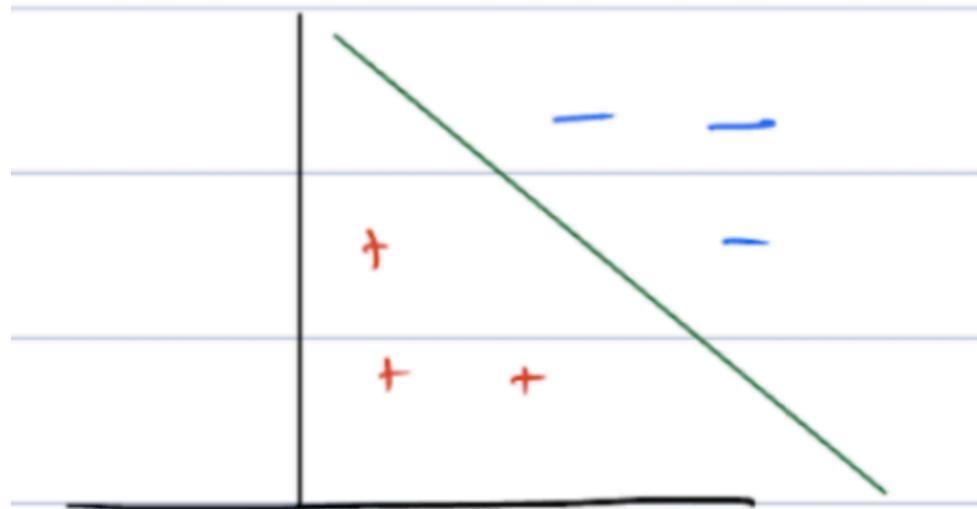
Many people are implicitly using test dataset as validation

- Always track the training and validation metrics/errors/losses

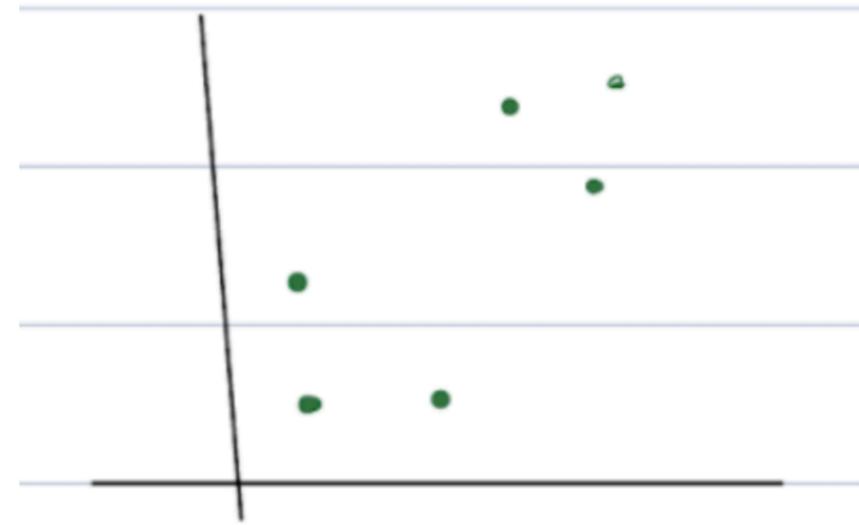
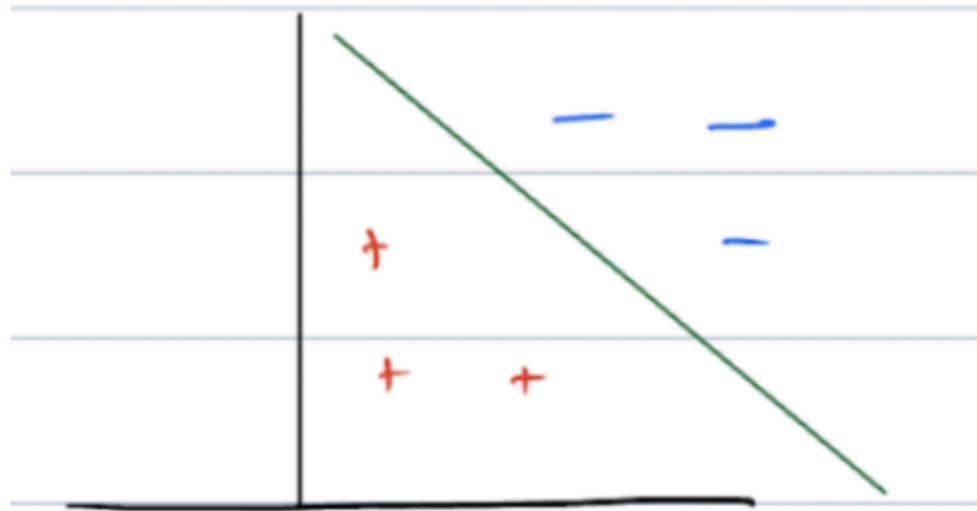


Unsupervised Learning

Unsupervised Learning

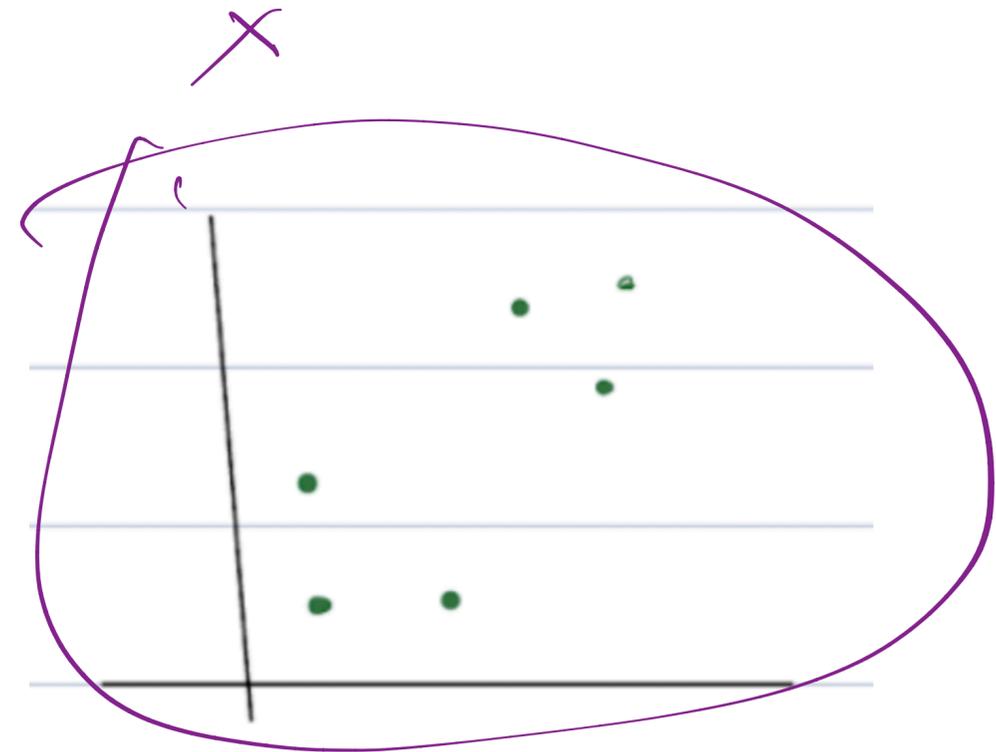
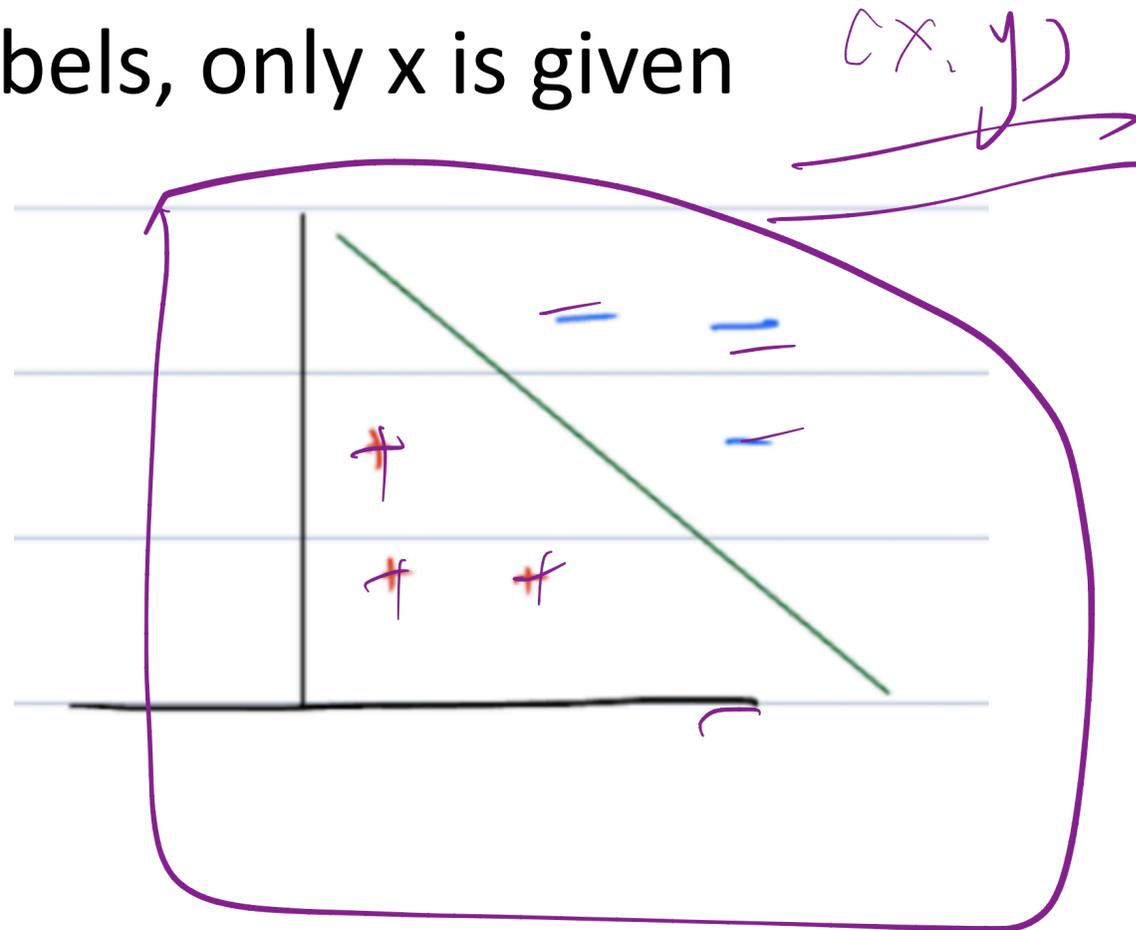


Unsupervised Learning



Unsupervised Learning

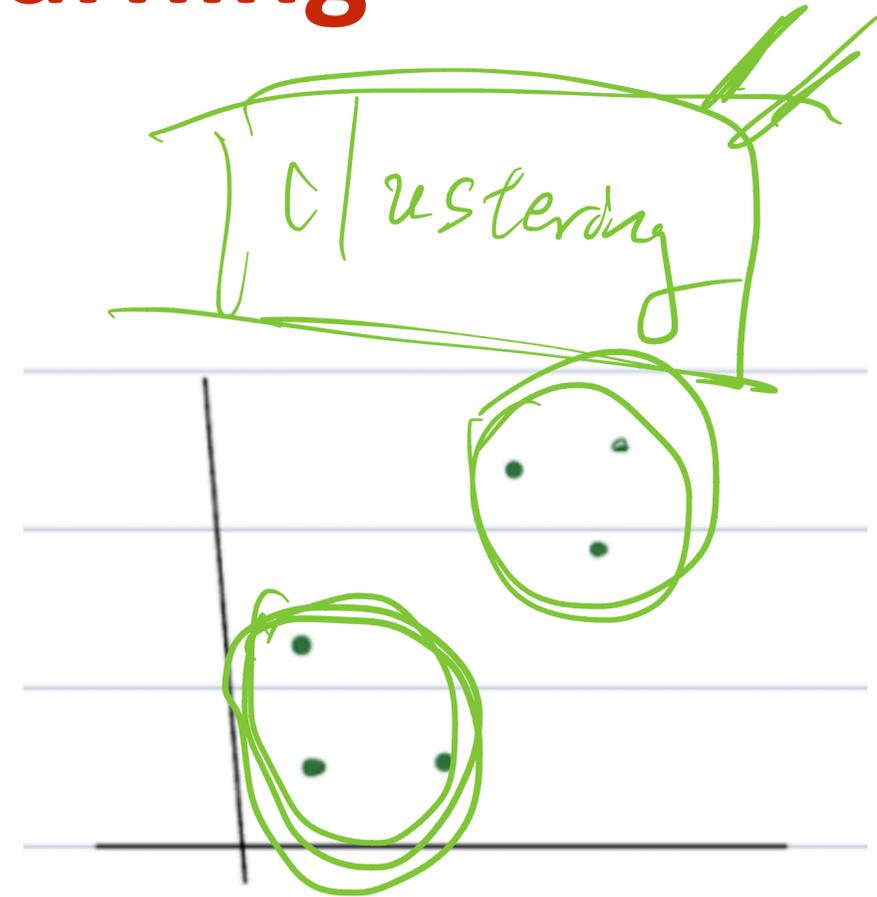
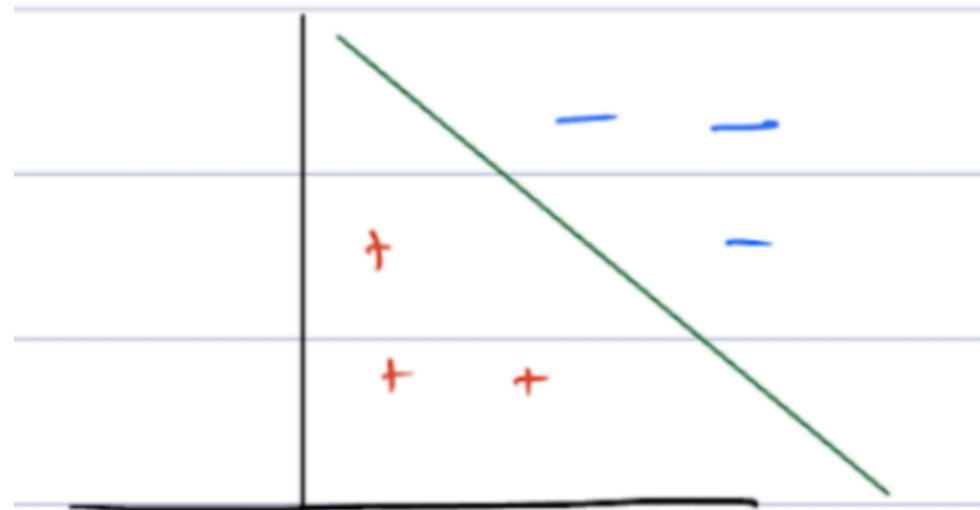
No labels, only x is given



Unsupervised Learning

representation learning

No labels, only x is given



Unsupervised learning is typically “harder” than supervised learning

What is Clustering

What is Clustering

Clustering: the process of grouping a set of objects into classes of similar objects

- high intra-class similarity
- low inter-class similarity
- It is the most common form of **unsupervised learning**

What is Clustering

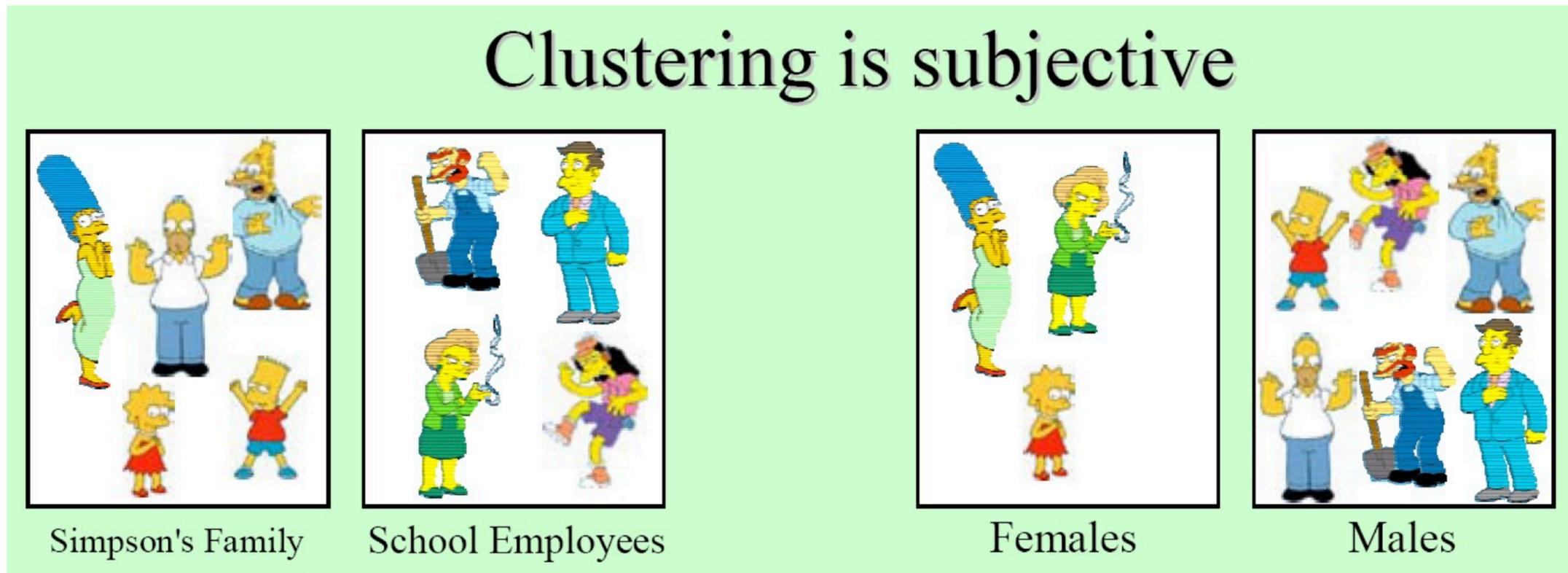
Clustering: the process of grouping a set of objects into classes of similar objects

- high intra-class similarity
 - low inter-class similarity
 - It is the most common form of **unsupervised learning**
- Similarity is subjective**

What is Clustering

Clustering: the process of grouping a set of objects into classes of similar objects

- high intra-class similarity
- low inter-class similarity
- It is the most common form of **unsupervised learning** **Similarity is subjective**



Distance Metrics

$$x = (x_1, x_2, \dots, x_p)$$

$$y = (y_1, y_2, \dots, y_p)$$

Distance Metrics

$$x = (x_1, x_2, \dots, x_p)$$

$$y = (y_1, y_2, \dots, y_p)$$

Euclidean distance

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

Manhattan distance

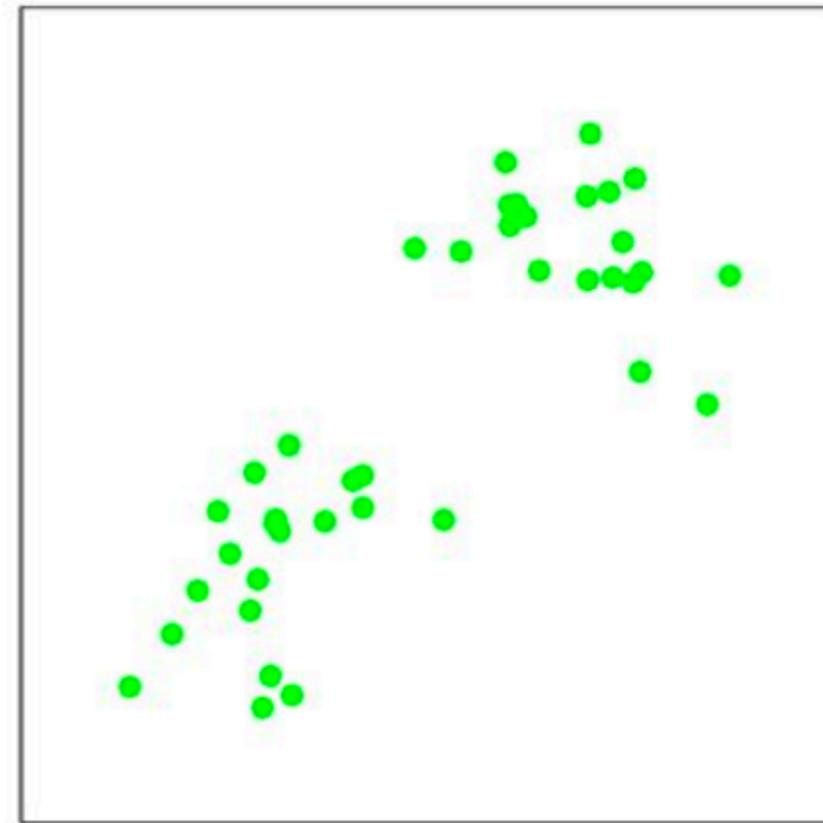
$$d(x, y) = \sum_{i=1}^p |x_i - y_i|$$

$$|x_i - y_i|$$

Sup-distance

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

K-Means Clustering



K-Means

K-Means

Algorithm

Input – Desired number of clusters, k

Initialize – the k cluster centers (randomly if necessary)

Iterate –

K-Means

Algorithm

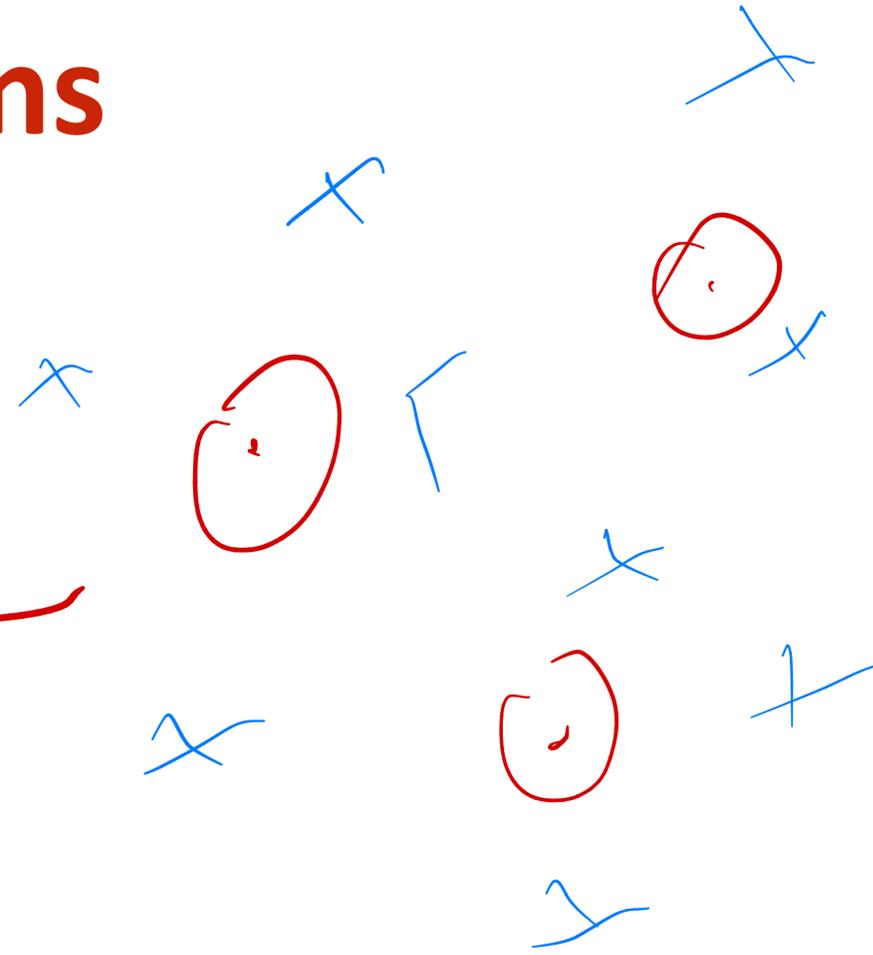
Input – Desired number of clusters, k

Initialize – the k cluster centers (randomly if necessary)

Iterate –

1. Assign points to the nearest cluster centers

distance metric



K-Means

Algorithm

Input – Desired number of clusters, k

Initialize – the k cluster centers (randomly if necessary)

Iterate –

1. Assign points to the nearest cluster centers
2. Re-estimate the k cluster centers (aka the **centroid** or **mean**), by assuming the memberships found above are correct.

$$\vec{\mu}_k = \frac{1}{C_k} \sum_{i \in C_k} \vec{x}_i$$

K-Means

Algorithm

Input – Desired number of clusters, k

Initialize – the k cluster centers (randomly if necessary)

Iterate –

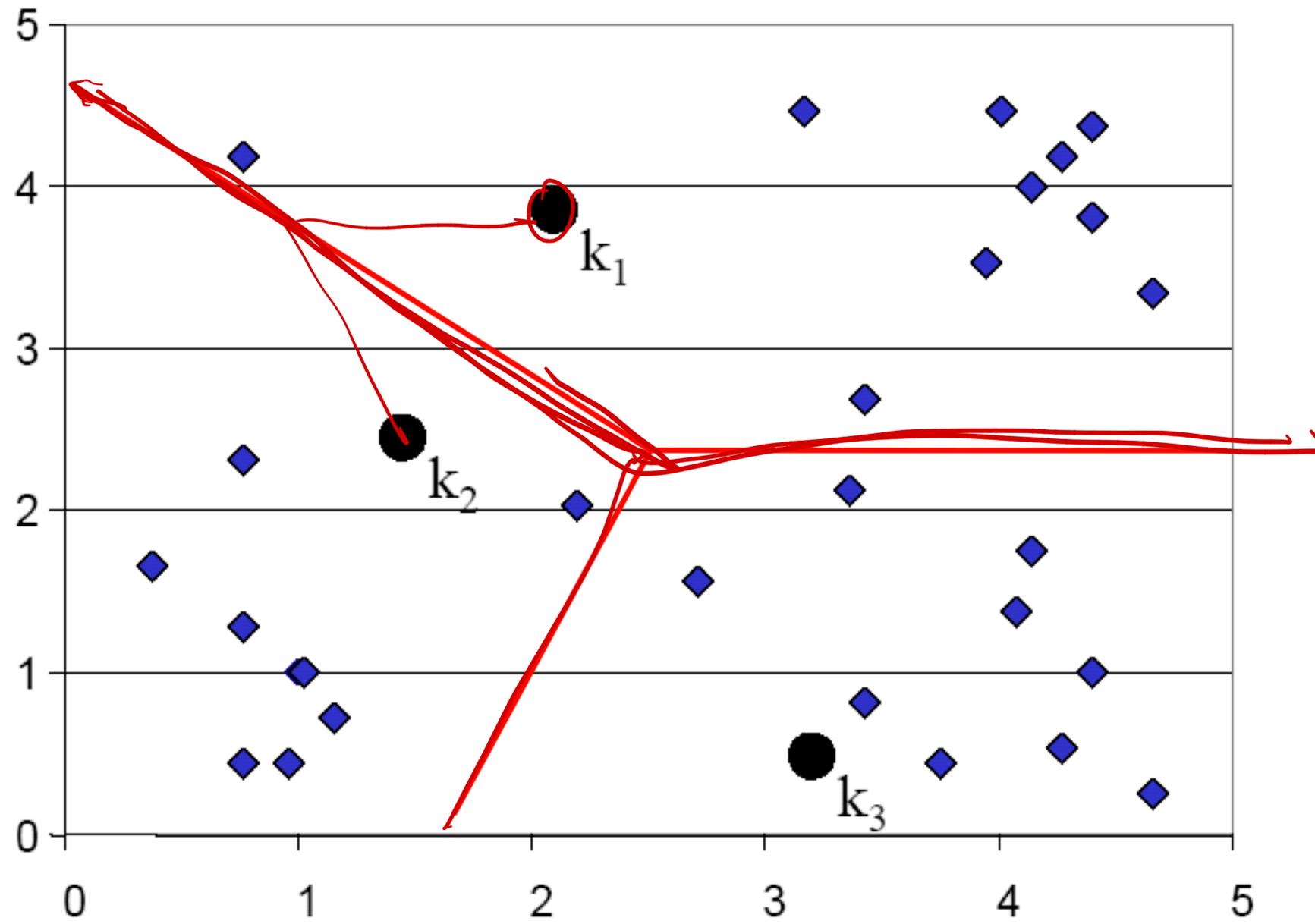
1. Assign points to the nearest cluster centers
2. Re-estimate the k cluster centers (aka the **centroid** or **mean**), by assuming the memberships found above are correct.

$$\vec{\mu}_k = \frac{1}{C_k} \sum_{i \in C_k} \vec{x}_i$$

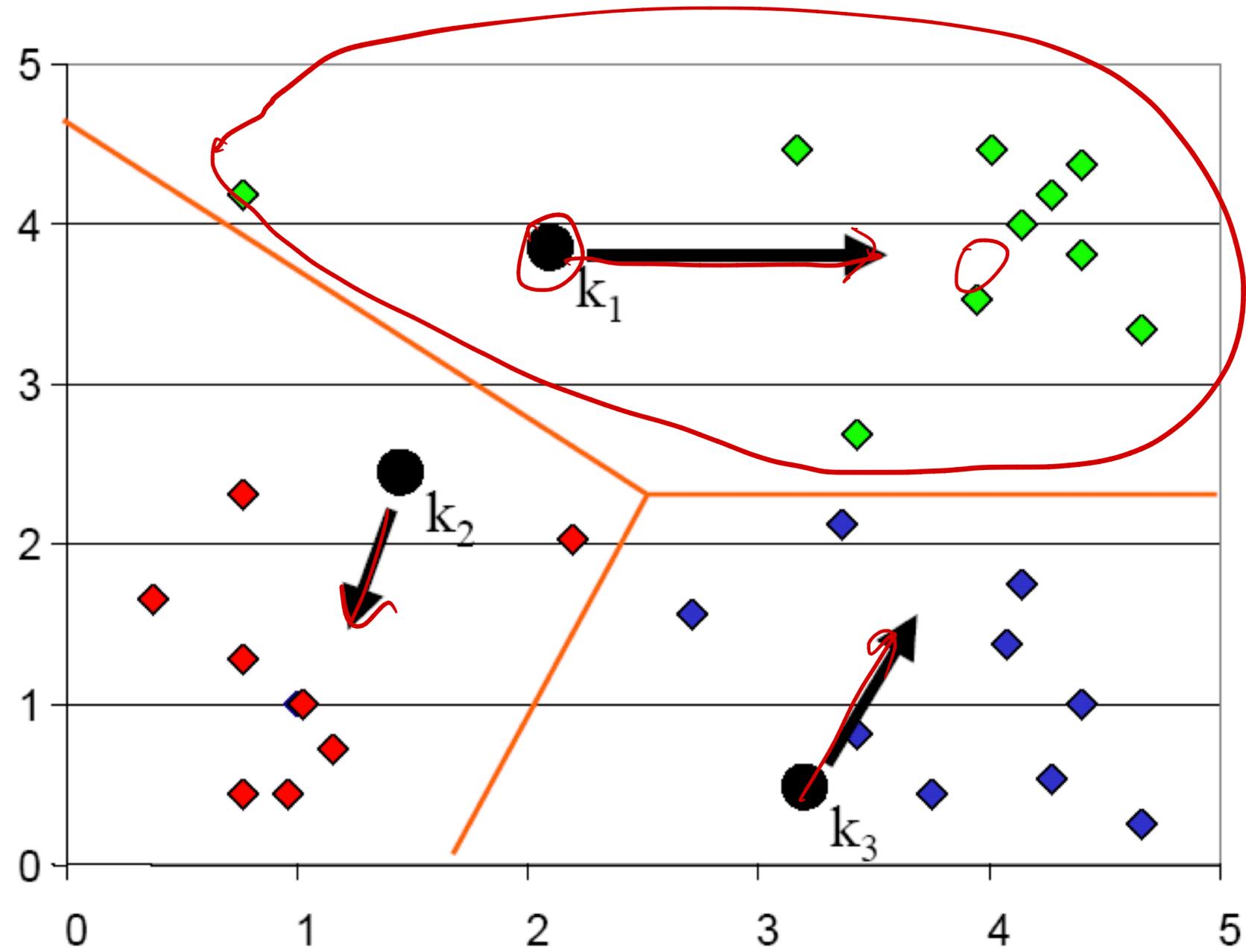
Termination –

If none of the objects changed membership in the last iteration, exit.
Otherwise go to 1.

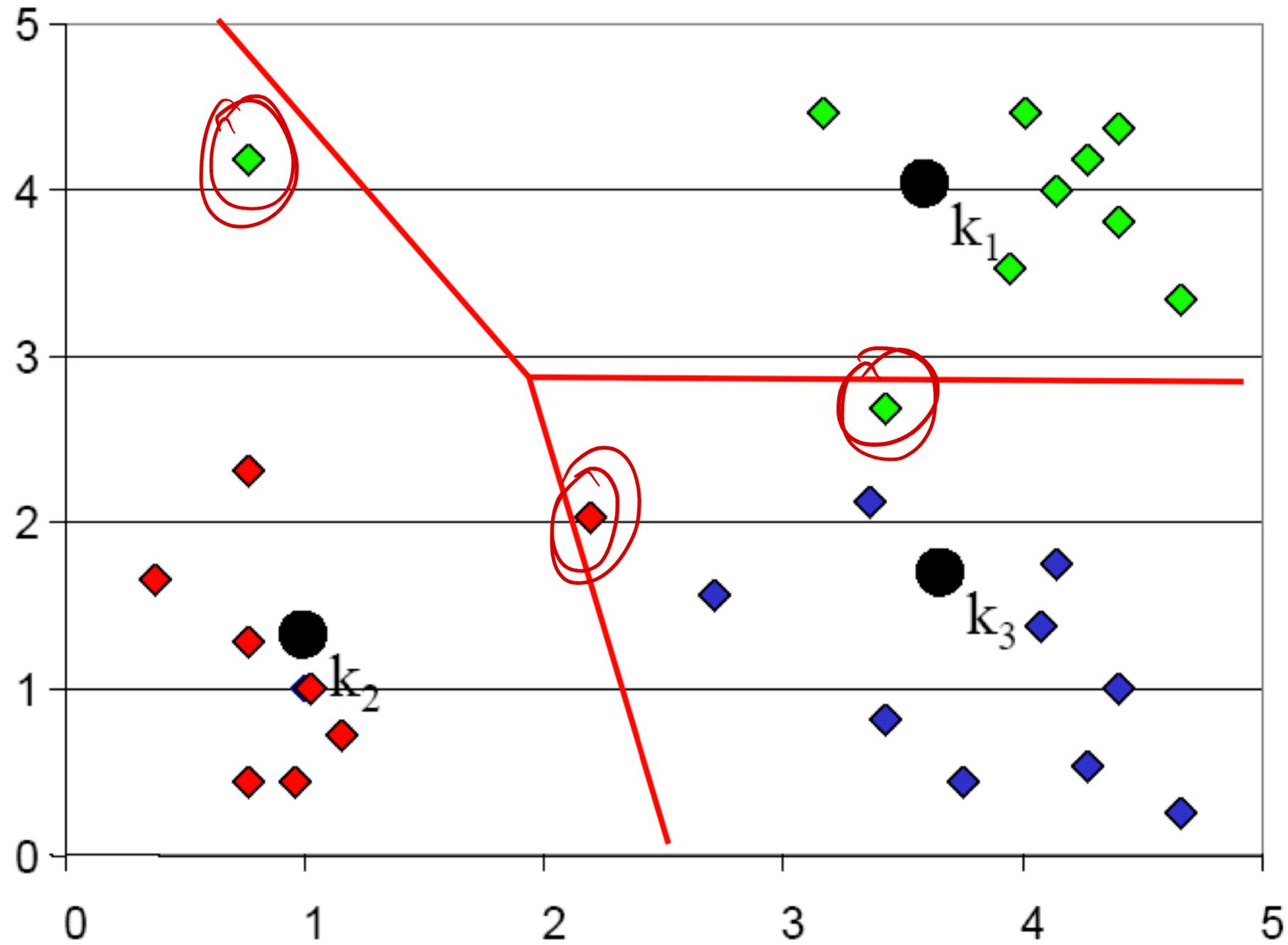
K-Means: Step 1



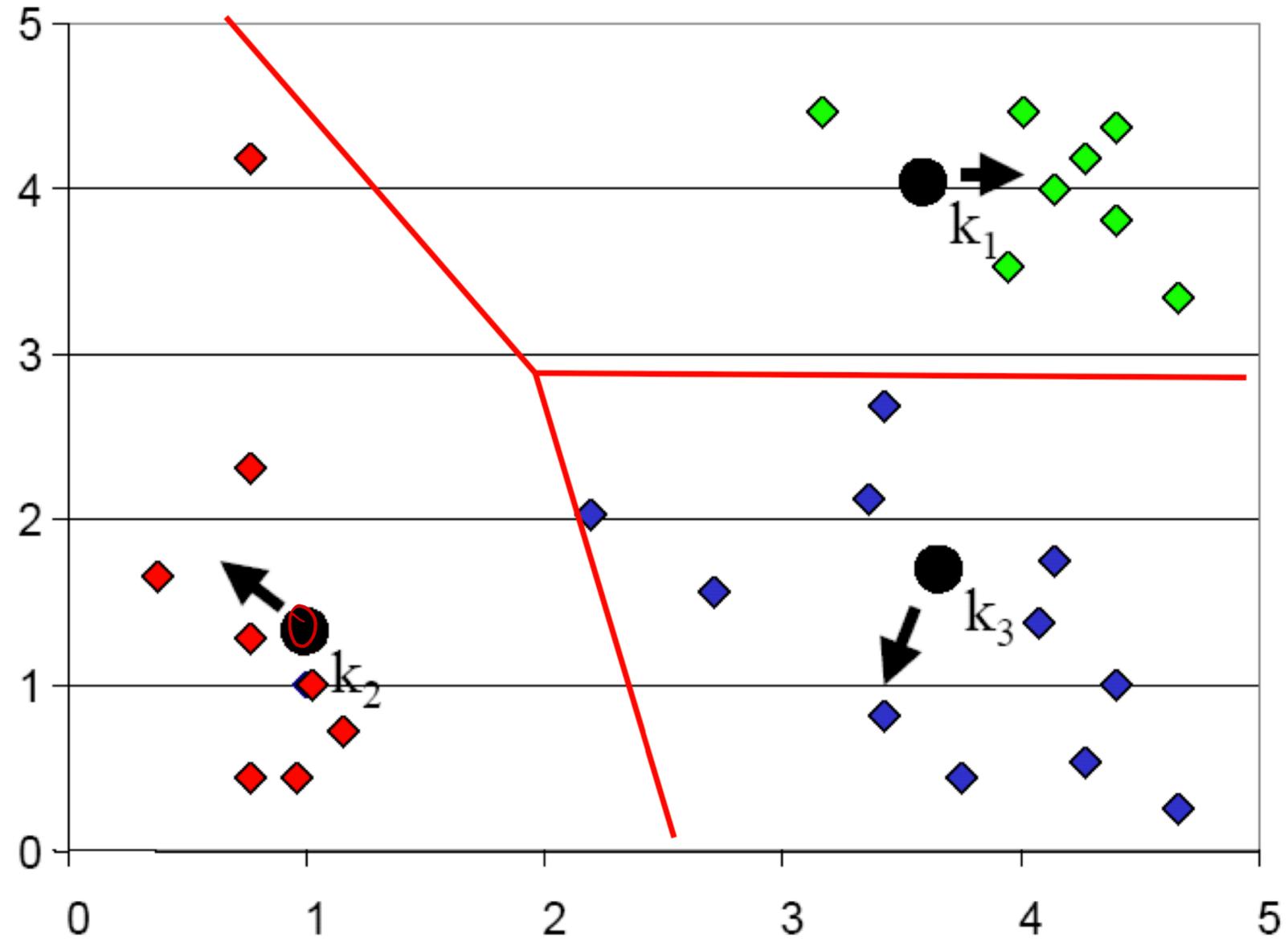
K-Means: Step 2



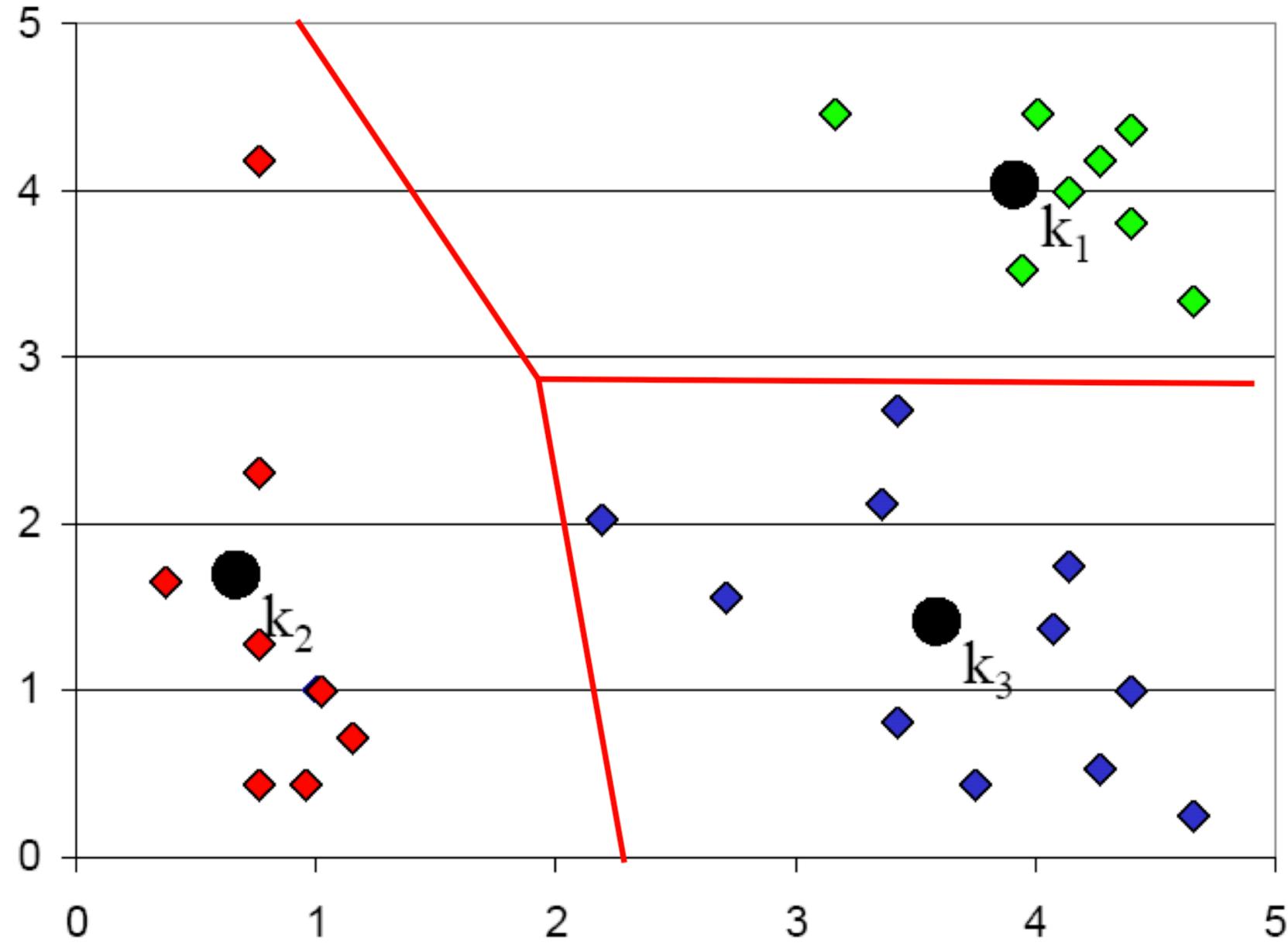
K-Means: Step 3



K-Means: Step 4



K-Means: Step 5



Objective of K-Means

Objective of K-Means

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C(i)}\|^2 \text{ decreases monotonically.}$$

Handwritten annotations:

- A red oval encircles the entire equation.
- A red oval encircles the summation symbol $\sum_{i=1}^n$.
- A red arrow points from the term $\mu^{C(i)}$ to the handwritten text "cluster center".

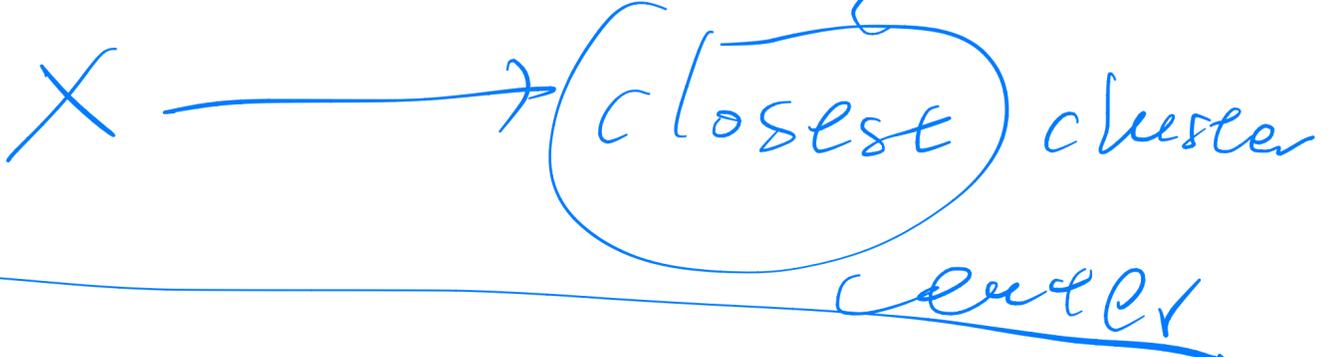
Objective of K-Means

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

decreases monotonically.

Proof?

1. assignment.

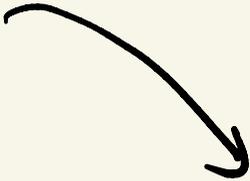


2. $C^{(i)}$ no change. $\mu = \frac{\sum x_i}{n}$

0

V \geq Bound

$$V_t \rightarrow V_{t+1} \rightarrow V_{t+2}$$

V 

always converge

Objective of K-Means

cluster ID of data i

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

decreases monotonically.

$\mu, C^{(i)}$

Proof?

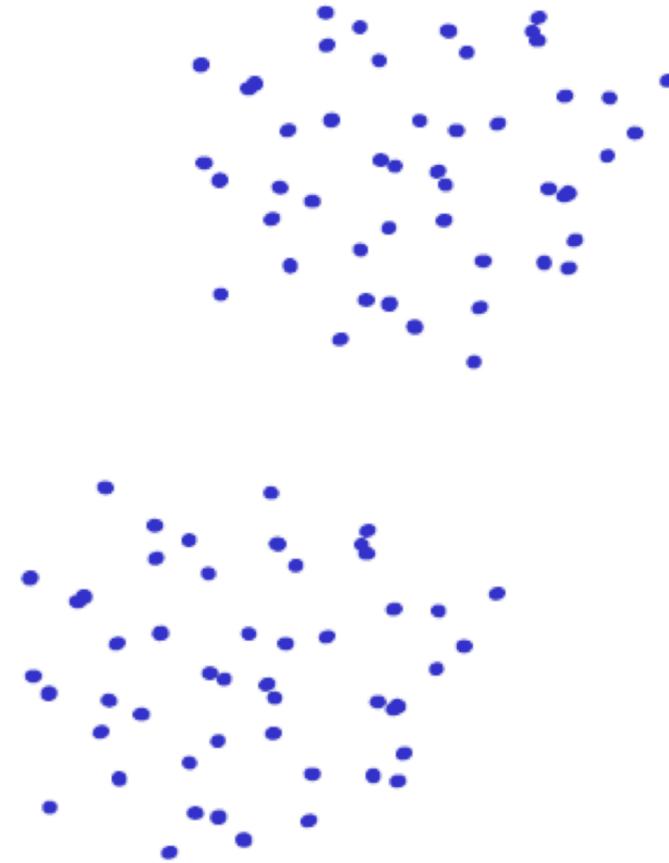
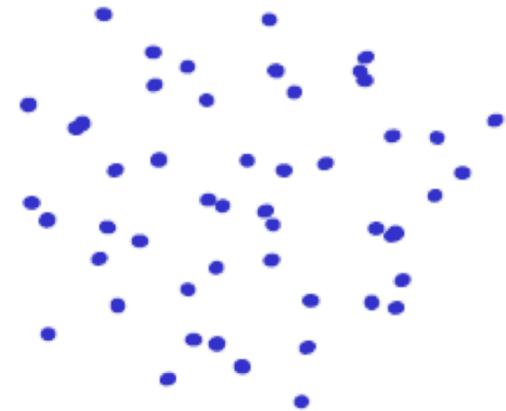
K-means does not find a global minimum in this objective (it is NP-Hard)

Initialization of Centers

Results are sensitive to the initialization

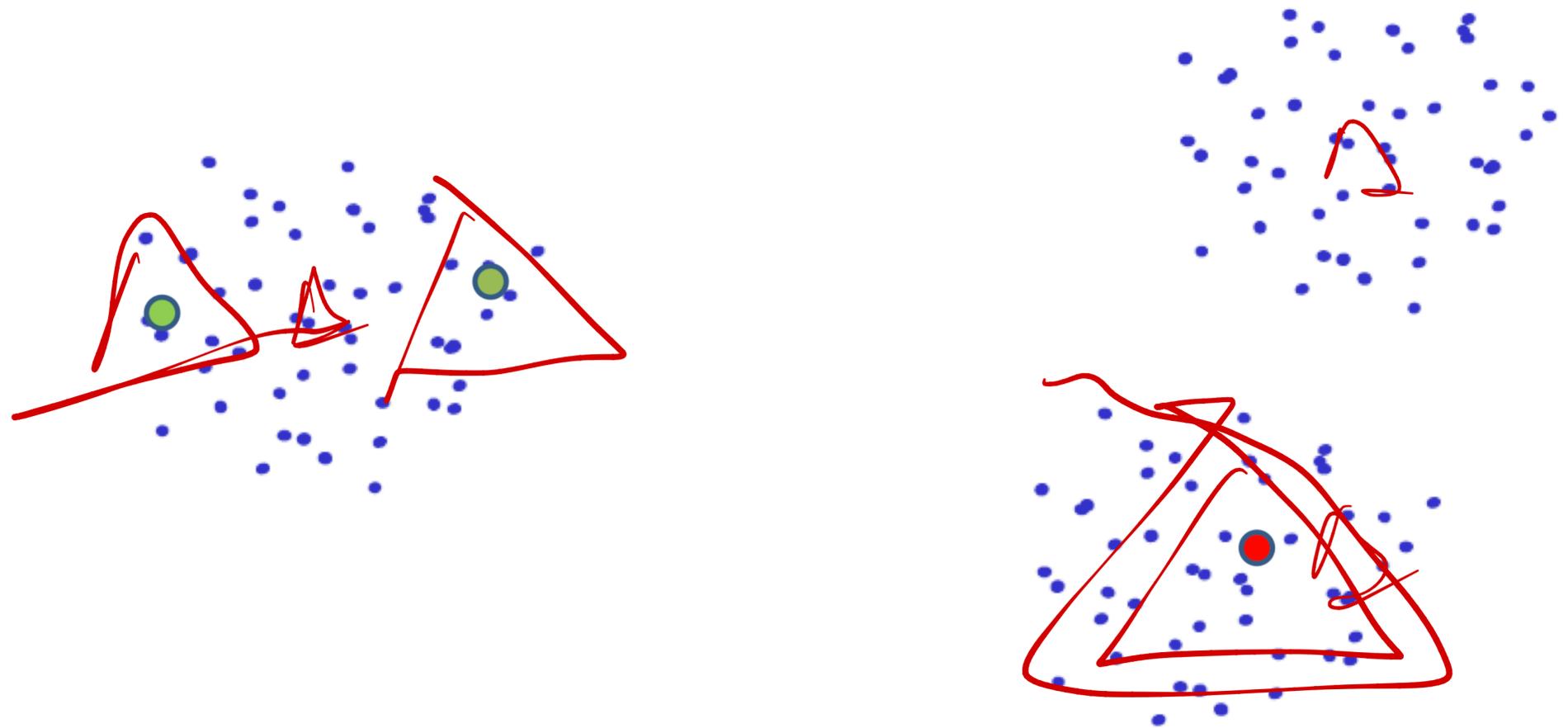
Initialization of Centers

Results are sensitive to the initialization



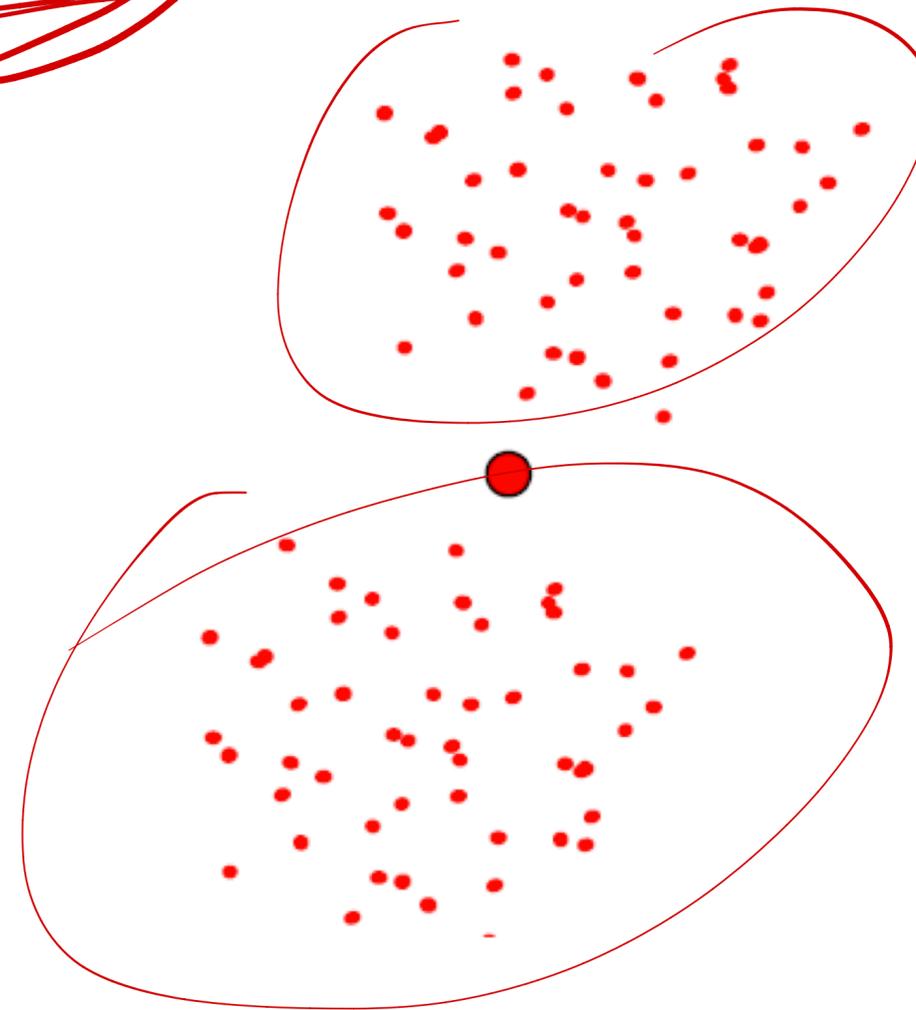
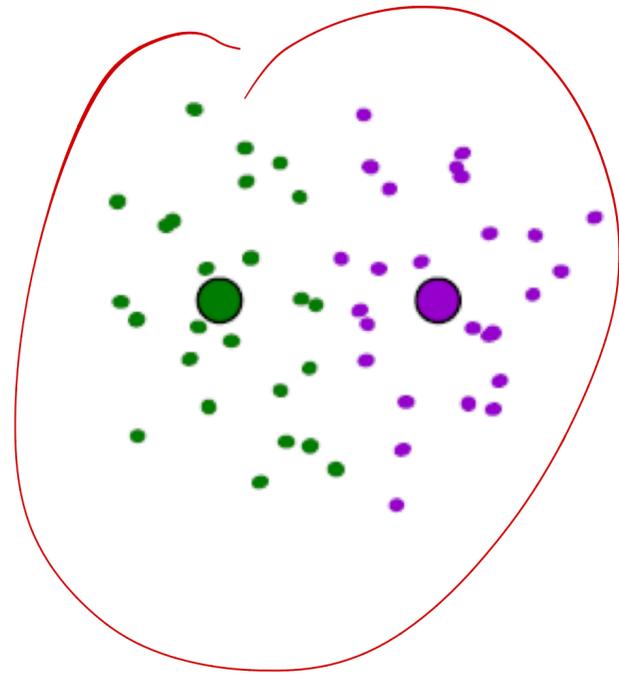
Initialization of Centers

Results are sensitive to the initialization



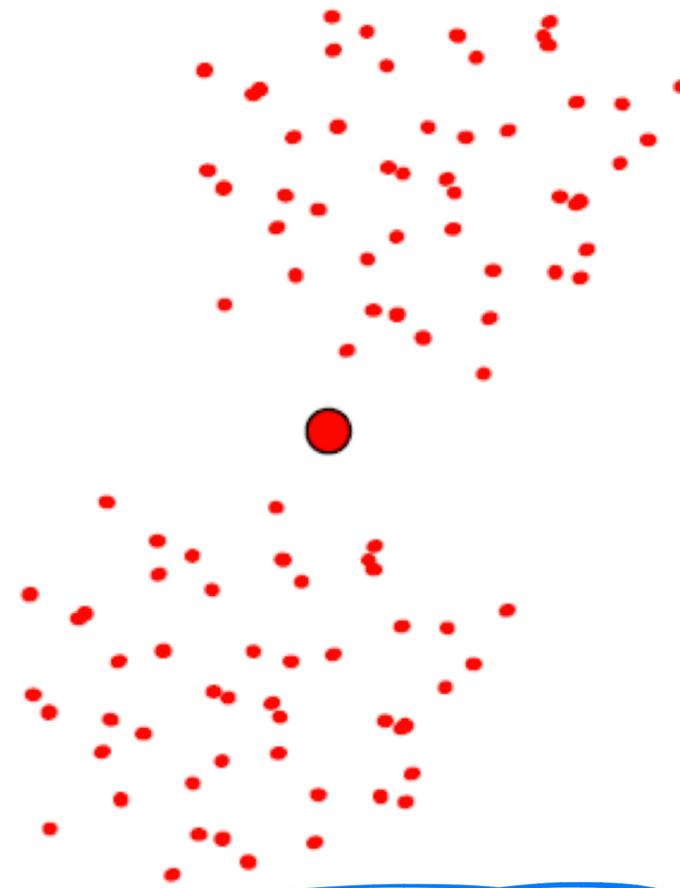
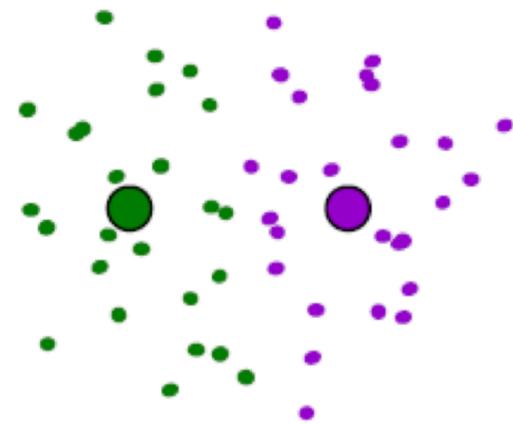
Initialization of Centers

Results are sensitive to the initialization



Initialization of Centers

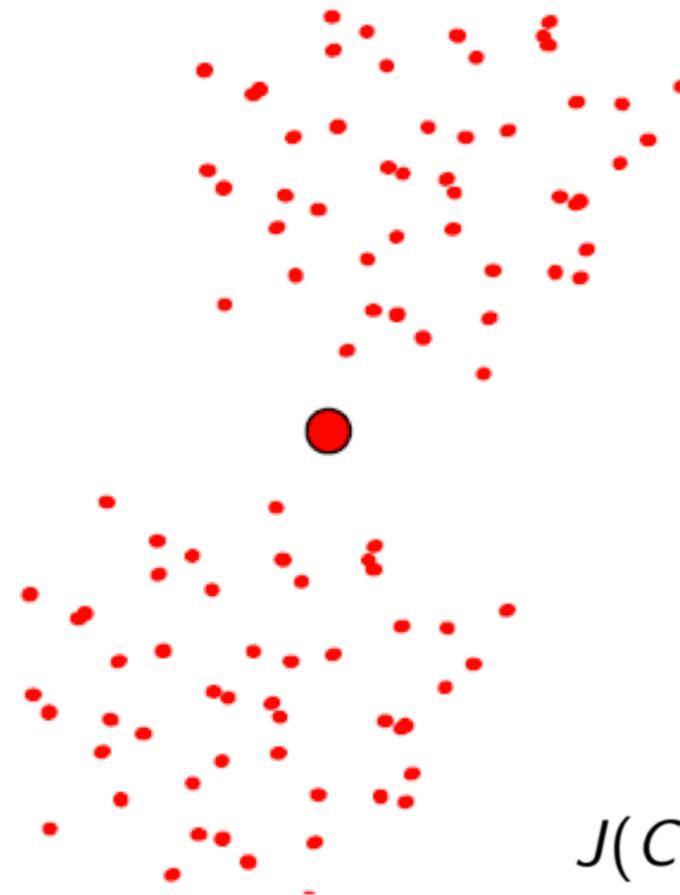
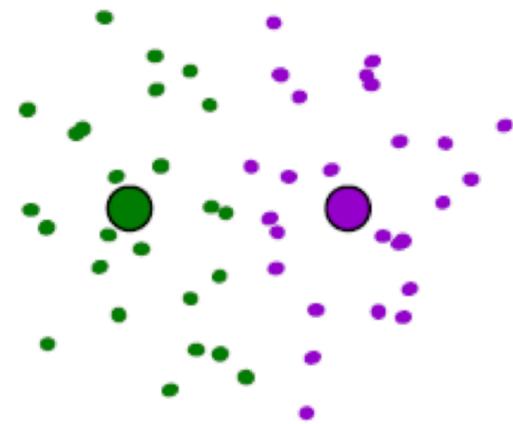
Results are sensitive to the initialization



1. Try out multiple starting points and compare the objective

Initialization of Centers

Results are sensitive to the initialization

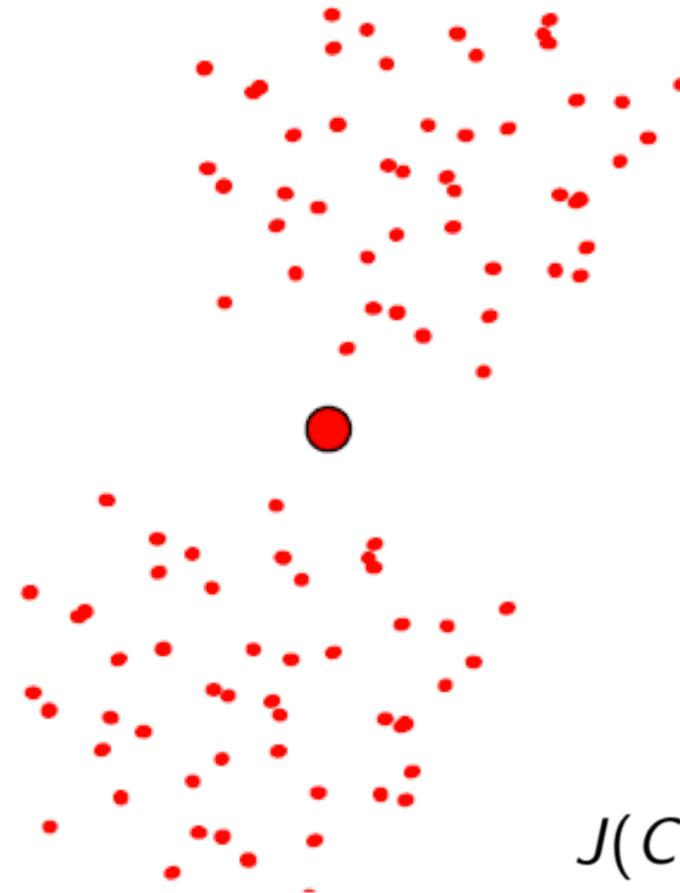
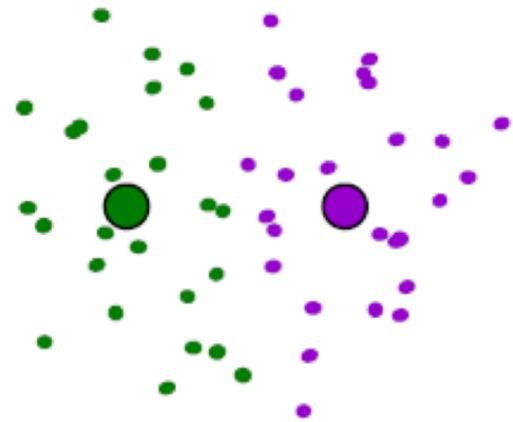


$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

1. Try out multiple starting points and compare the objective

Initialization of Centers

Results are sensitive to the initialization



$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

1. Try out multiple starting points and compare the objective
2. K-means++ algorithm improves the initialization

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

This is unsupervised metric

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

→ difference in other cases

This is unsupervised metric

Sometimes people use supervised metrics for validation, which is not strictly unsupervised learning

*0 0 f f
0 0 f f*

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

This is unsupervised metric

Sometimes people use supervised metrics for validation, which is not strictly unsupervised learning

1. Compute the metric on training set or test set?

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

This is unsupervised metric

Sometimes people use supervised metrics for validation, which is not strictly unsupervised learning

1. Compute the metric on training set or test set?
2. For unsupervised learning, what is the difference of train and test?

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

This is unsupervised metric

Sometimes people use supervised metrics for validation, which is not strictly unsupervised learning

1. Compute the metric on training set or test set?
2. For unsupervised learning, what is the difference of train and test?
3. Is it reasonable to assume the test input (x) is given?

Supervised learning



unlabeled B

unsupervised learning

unsupervised learning



unlabeled A

unlabeled B

train/test

unlabeled B

Model Selection of K-Means (or Unsupervised Learning in General)

Try out multiple starting points and compare the objective

$$J(C, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu^{C^{(i)}}\|^2$$

This is unsupervised metric

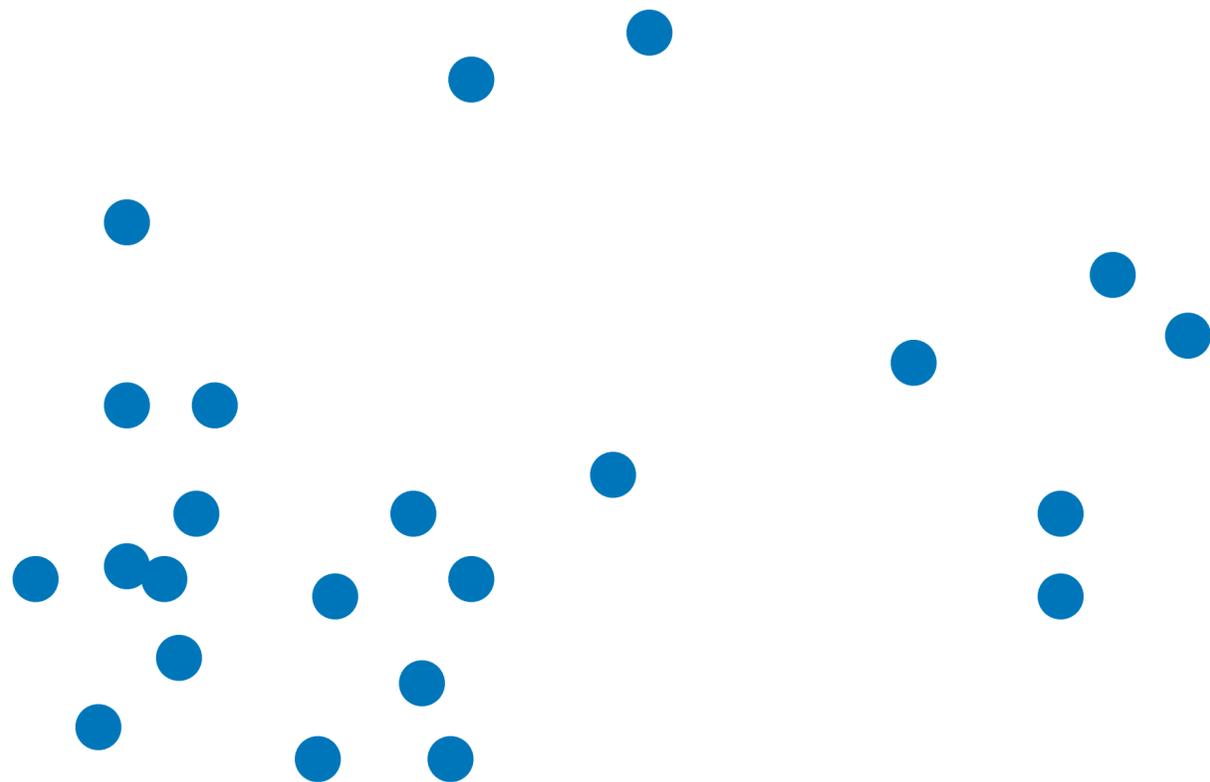
Sometimes people use supervised metrics for validation, which is not strictly unsupervised learning

1. Compute the metric on training set or test set?
2. For unsupervised learning, what is the difference of train and test?
3. Is it reasonable to assume the test input (x) is given?
4. If now I give you some data examples, ask you to cluster them. Are these data training or test?

Expectation Maximization (EM)

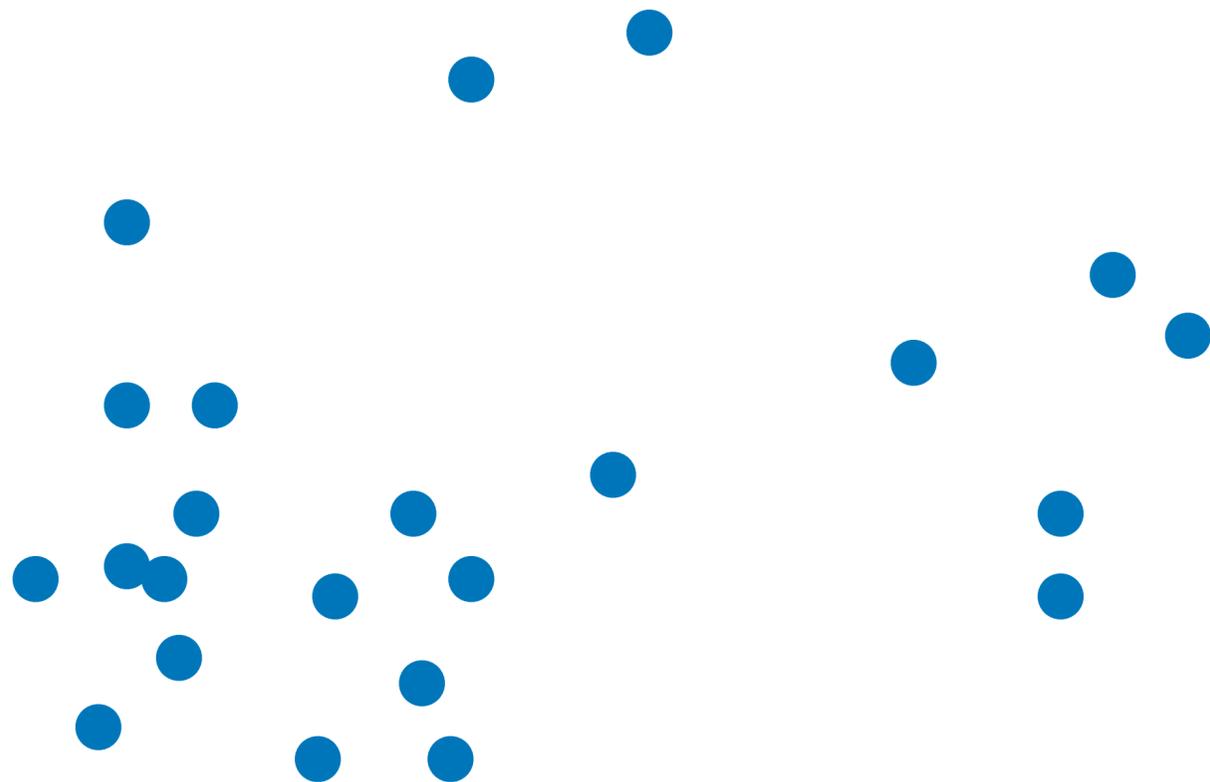
EM for Gaussian Mixture Model

Given a training set $\{x^{(1)}, \dots, x^{(n)}\}$



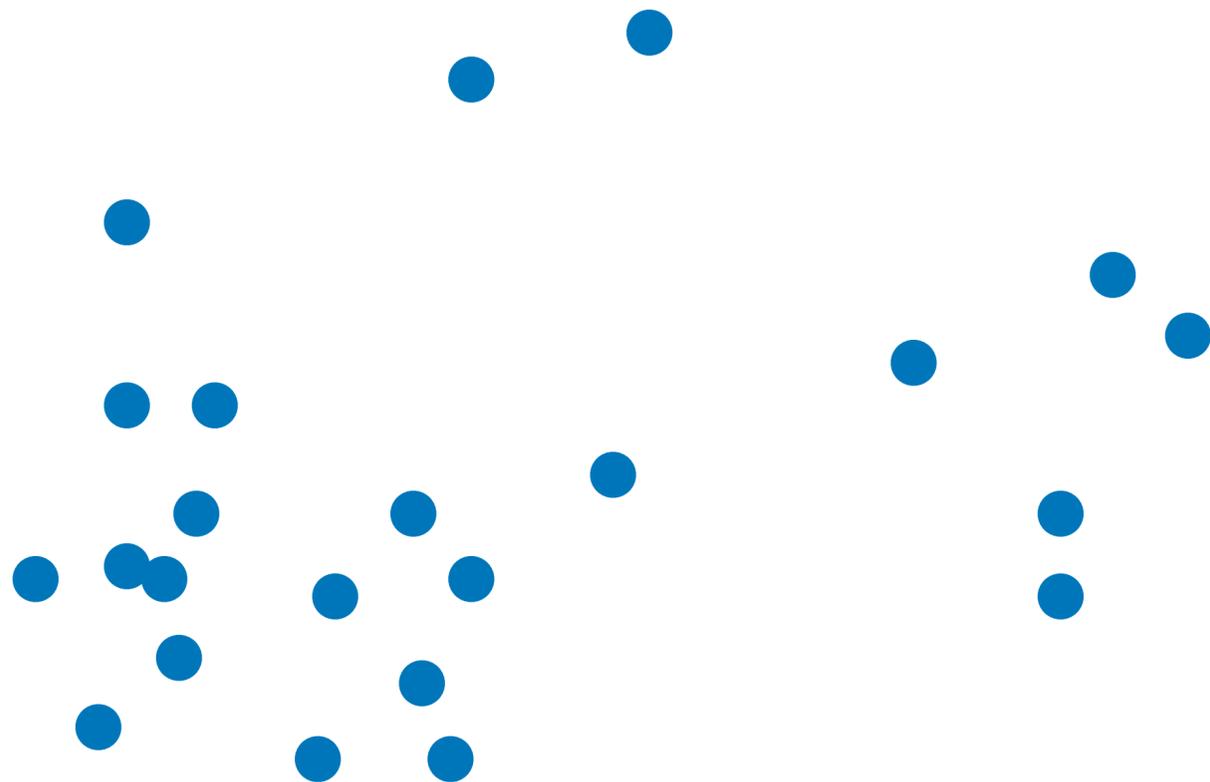
EM for Gaussian Mixture Model

Given a training set $\{x^{(1)}, \dots, x^{(n)}\}$ **No Labels**



EM for Gaussian Mixture Model

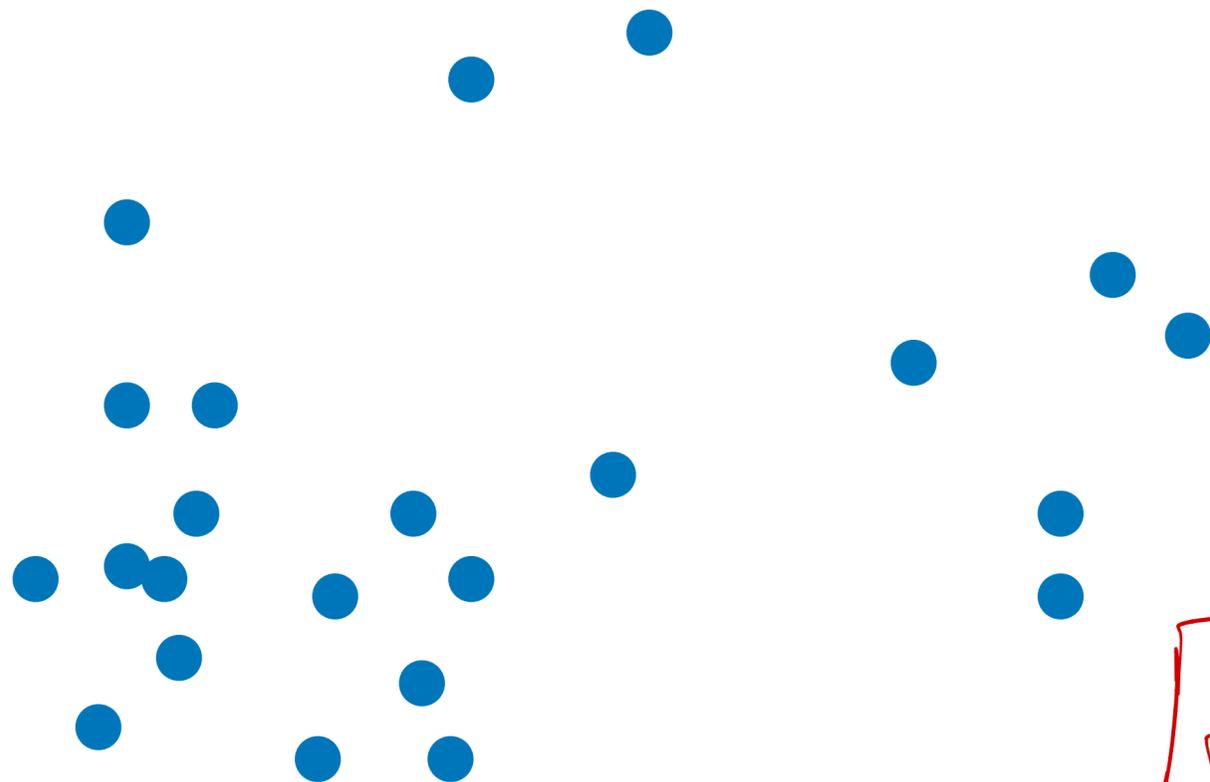
Given a training set $\{x^{(1)}, \dots, x^{(n)}\}$ **No Labels**



We have discussed the supervised case in Gaussian Discriminative Model

EM for Gaussian Mixture Model

Given a training set $\{x^{(1)}, \dots, x^{(n)}\}$ **No Labels**



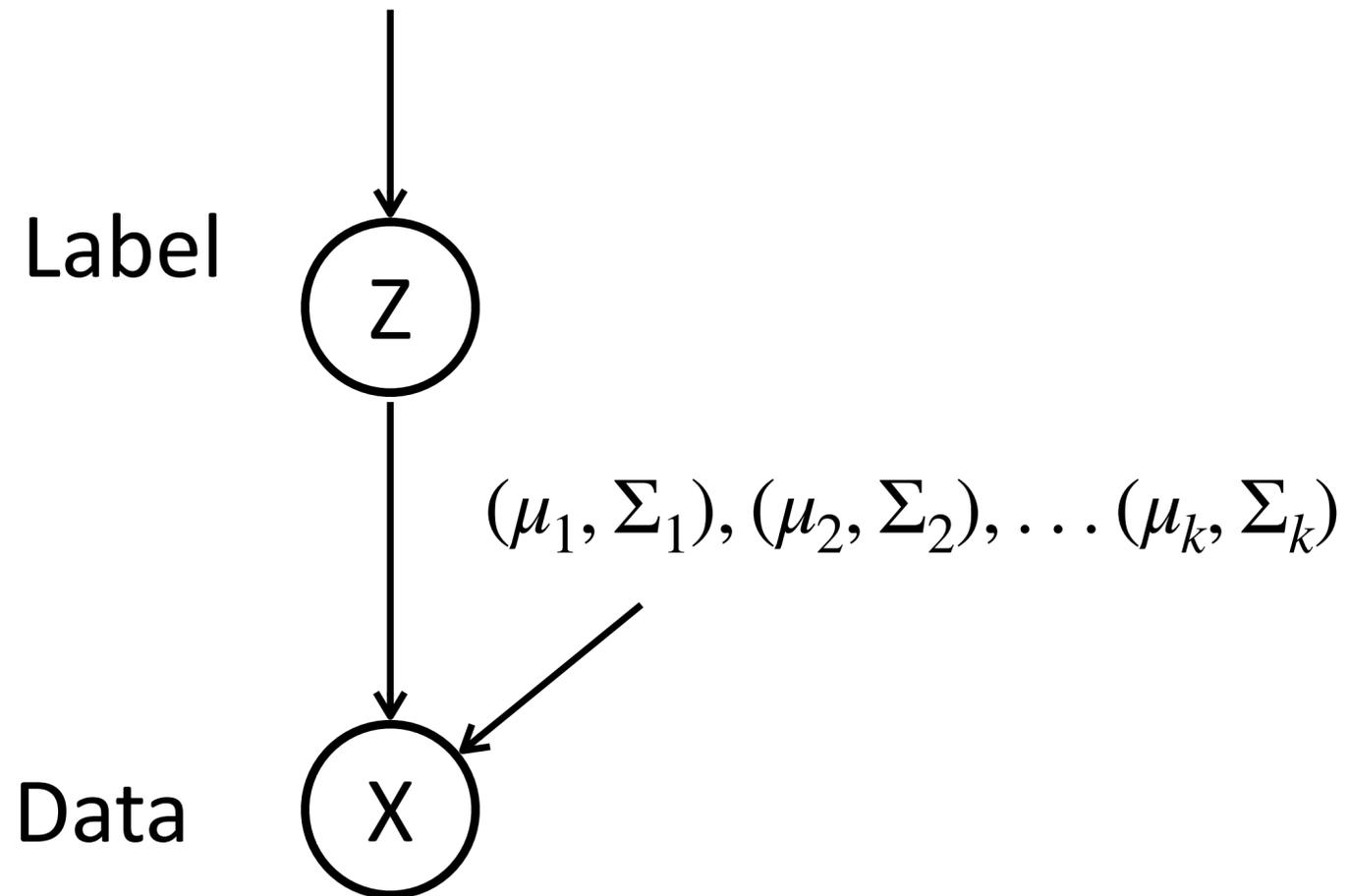
We have discussed the supervised case in Gaussian Discriminative Model

$P(x)$

Modeling data distribution is a fundamental goal in ML, not necessarily for classification

The Generative Model

$p(z)$: multinomial, k
classes (e.g. uniform)

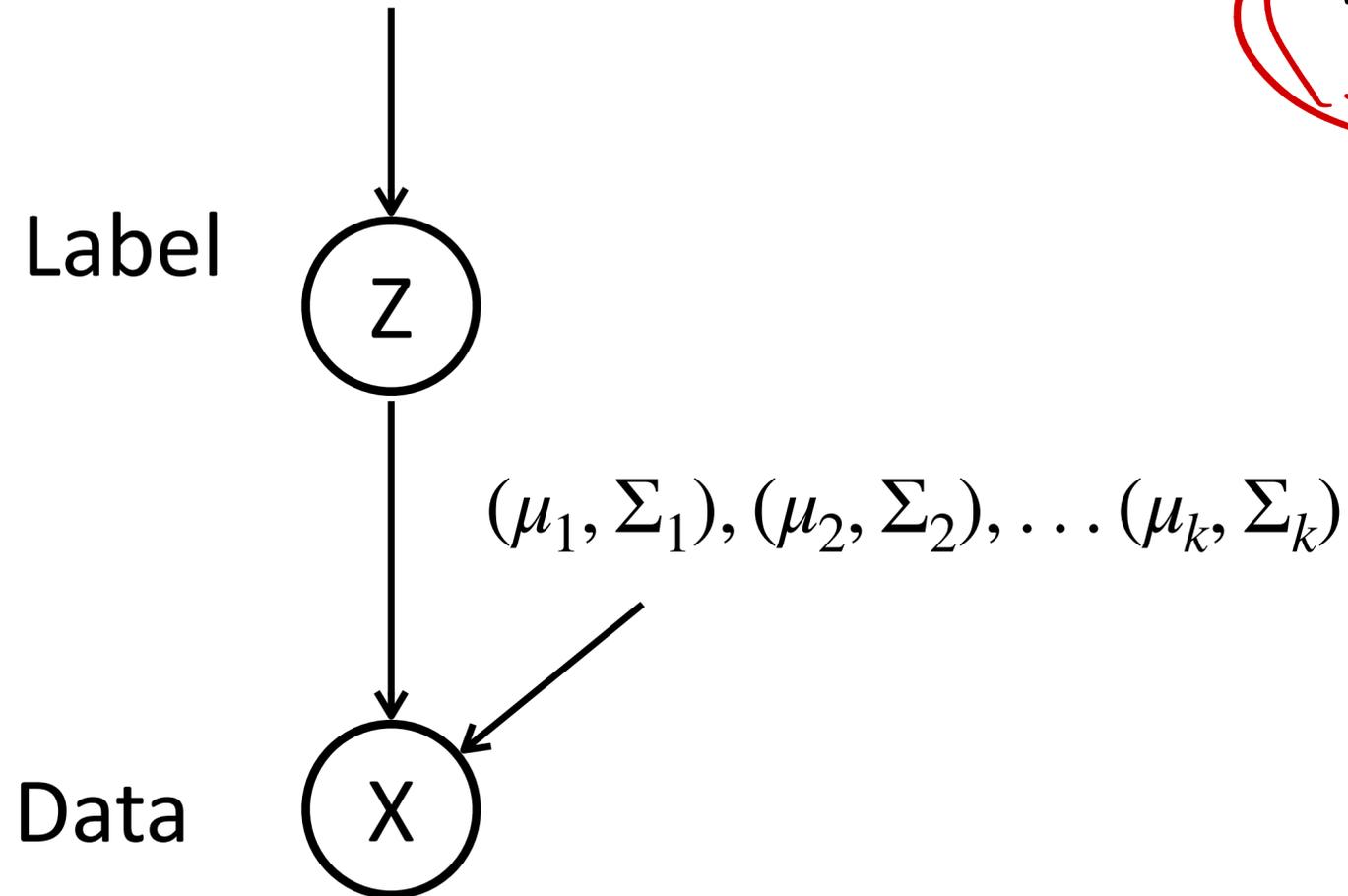


The Generative Model

$P(x)$

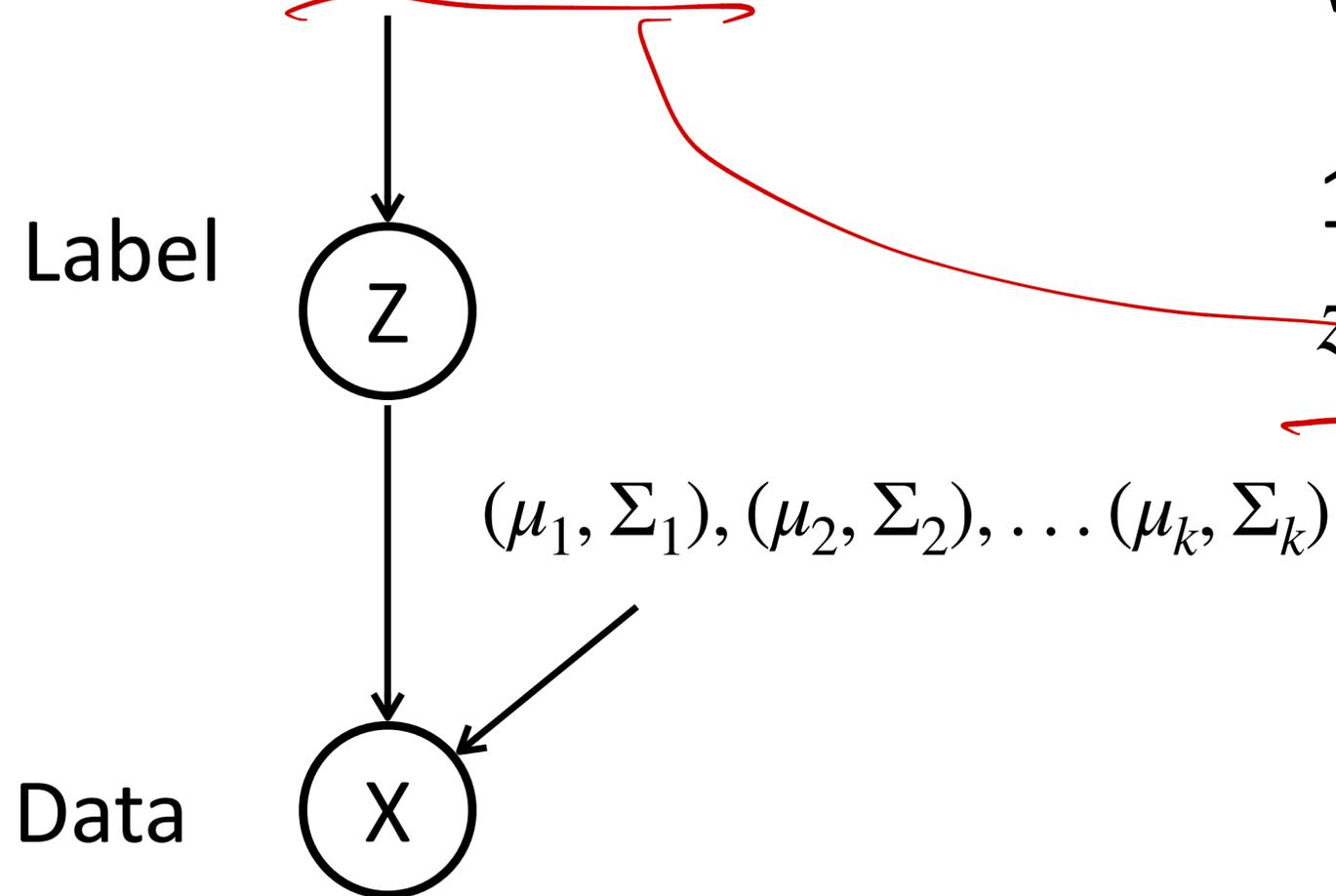
$p(z)$: multinomial, k classes (e.g. uniform)

We assume the generative process as:



The Generative Model

$p(z)$: multinomial, k classes (e.g. uniform)



We assume the generative process as:

1. For each data point, sample its label

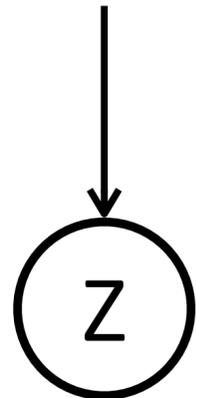
z_i from $p(z)$

$\{1, 2, \dots, k\}$

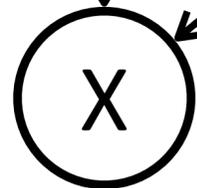
The Generative Model

$p(z)$: multinomial, k classes (e.g. uniform)

Label



Data



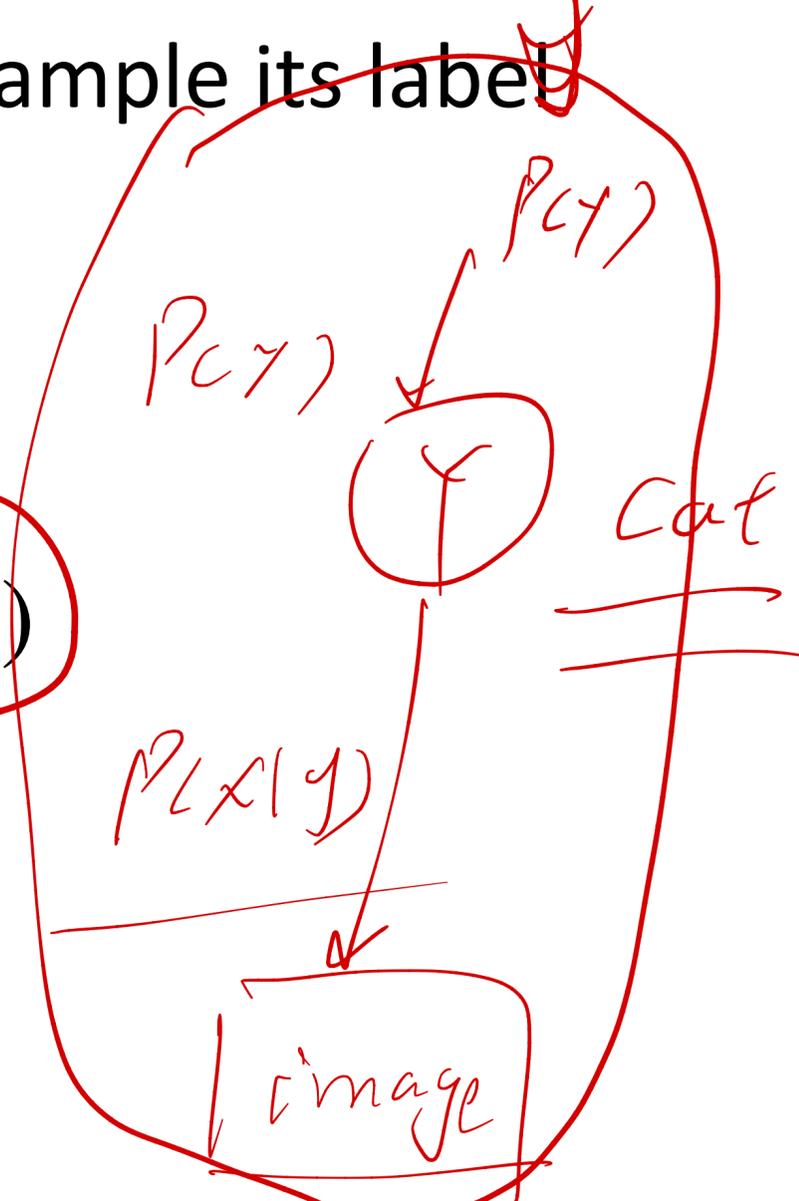
$(\mu_1, \Sigma_1), (\mu_2, \Sigma_2), \dots, (\mu_k, \Sigma_k)$

We assume the generative process as:

1. For each data point, sample its label z_i from $p(z)$

2. Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$

$y \sim p(y)$
 $p(x|y) \sim \mathcal{N}$



The Generative Model

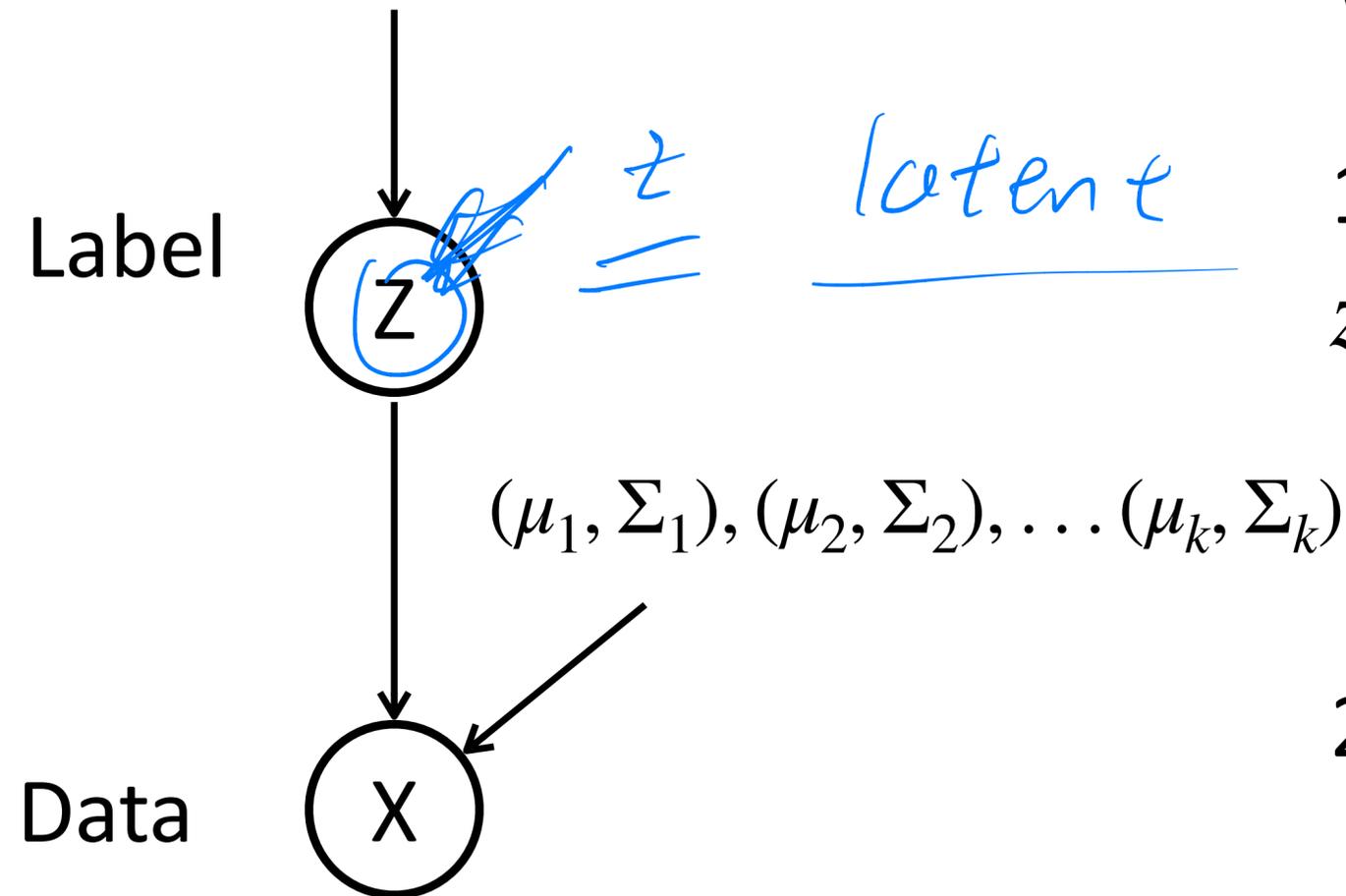
$p(z)$: multinomial, k classes (e.g. uniform)

K is a hyperparameter based on our assumption

We assume the generative process as:

1. For each data point, sample its label z_i from $p(z)$

2. Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$



The Generative Model

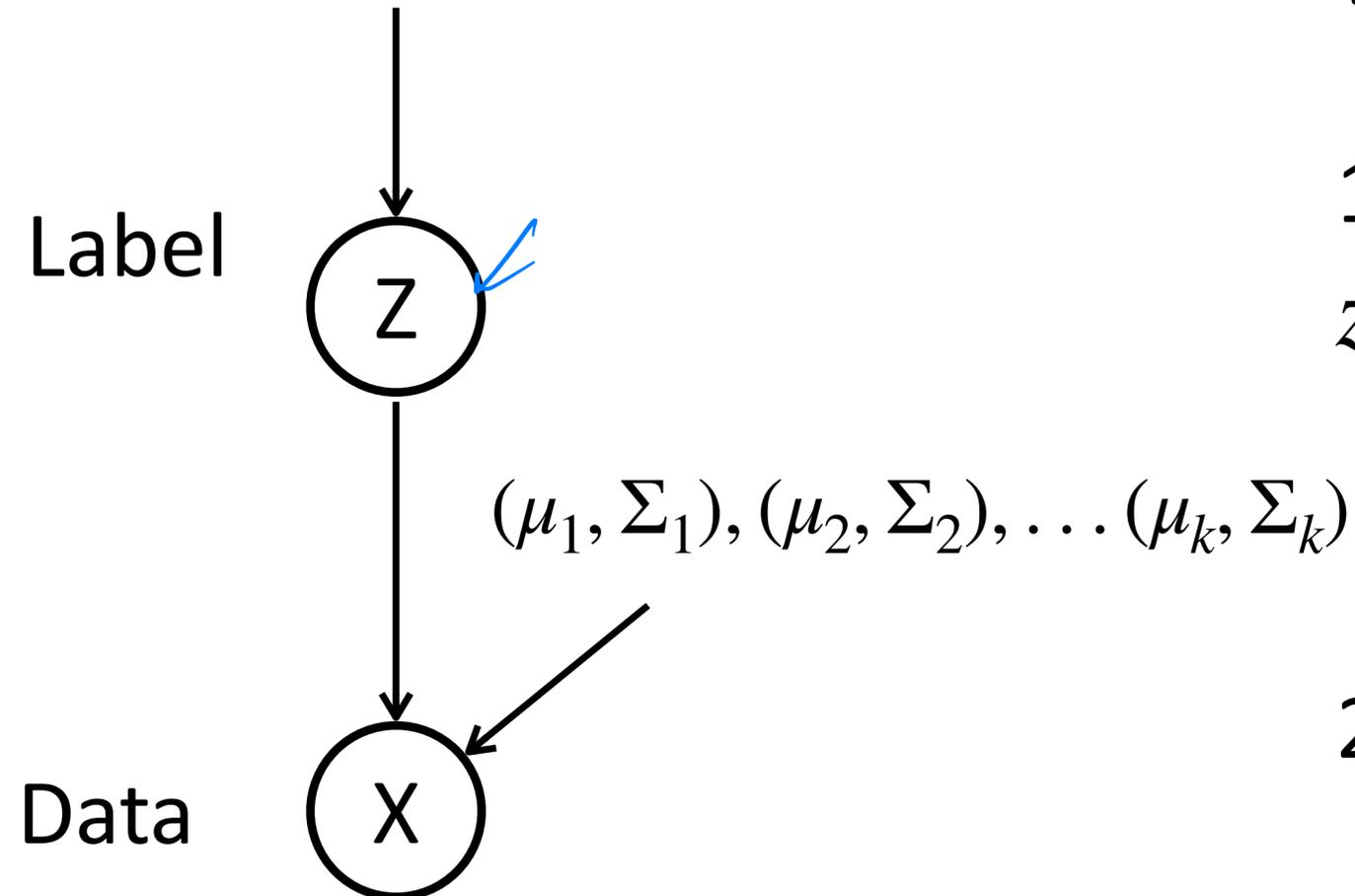
$p(z)$: multinomial, k classes (e.g. uniform)

K is a hyperparameter based on our assumption

We assume the generative process as:

1. For each data point, sample its label z_i from $p(z)$

2. Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$



Same as Gaussian Discriminative Analysis, but Z is observed in GDA

The Generative Model

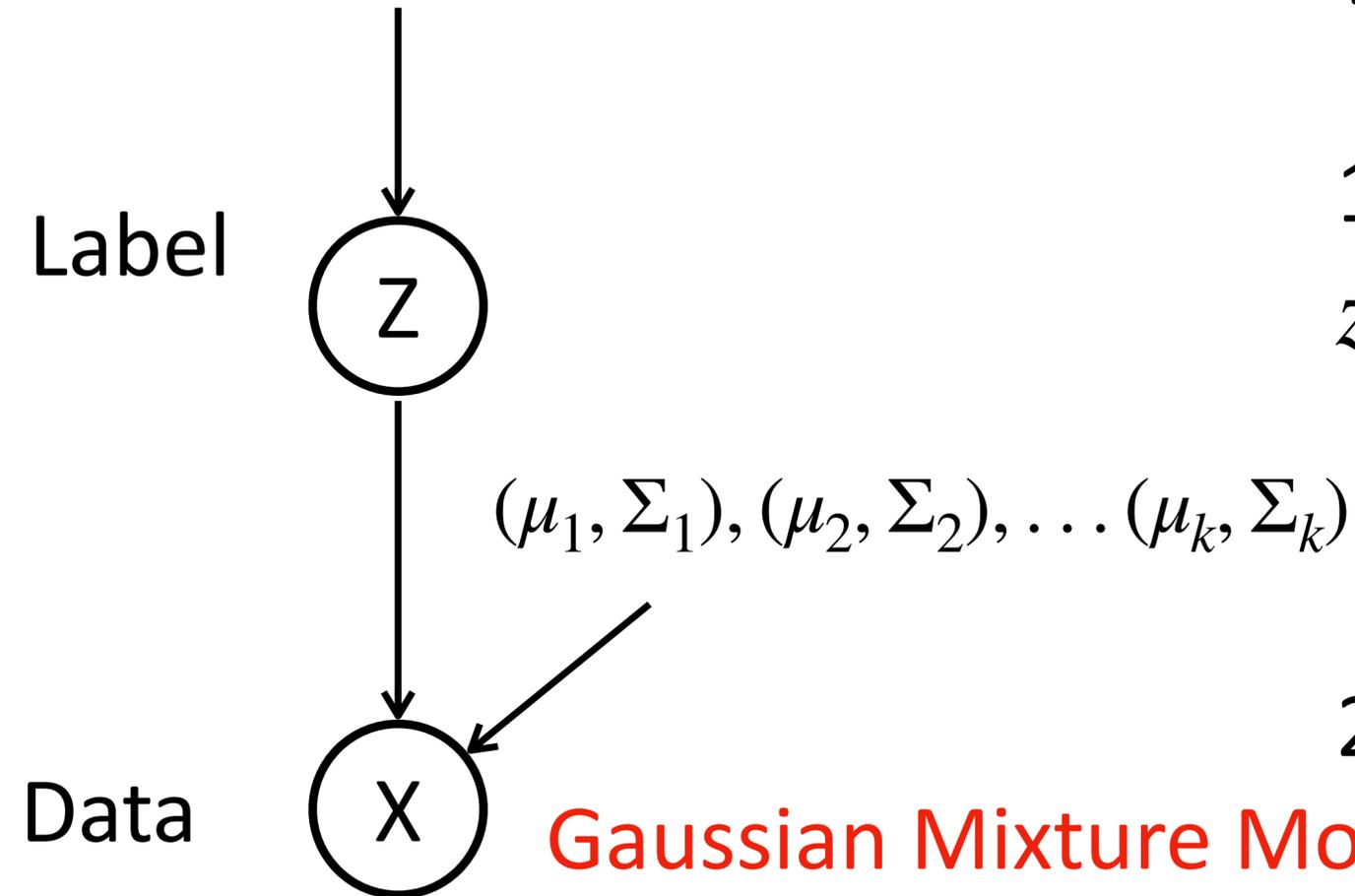
$p(z)$: multinomial, k classes (e.g. uniform)

K is a hyperparameter based on our assumption

We assume the generative process as:

1. For each data point, sample its label z_i from $p(z)$

2. Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$



Gaussian Mixture Model (GMM)

Same as Gaussian Discriminative Analysis, but Z is observed in GDA

Recap: How did we do in GDA?

Binary classification: $y \in \{0,1\}, x \in R^d$

Recap: How did we do in GDA?

Binary classification: $y \in \{0,1\}, x \in R^d$

Assumption

$$\begin{aligned} y &\sim \text{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Recap: How did we do in GDA?

Binary classification: $y \in \{0,1\}, x \in R^d$

$x \in R^d$
 (x, y)

Assumption

$$y \sim \text{Bernoulli}(\phi)$$

$$x|y=0 \sim \mathcal{N}(\mu_0, \Sigma)$$

$$x|y=1 \sim \mathcal{N}(\mu_1, \Sigma)$$

$$p(y) = \phi^y (1 - \phi)^{1-y}$$

$$p(x|y=0) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_0)^T \Sigma^{-1} (x - \mu_0)\right)$$

$$p(x|y=1) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_1)^T \Sigma^{-1} (x - \mu_1)\right)$$

Recap: How did we do in GDA?

Recap: How did we do in GDA?

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^n p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^n p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi). \end{aligned}$$

y is unknown

log p(x, y)

Recap: How did we do in GDA?

$$\begin{aligned}\ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^n p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^n p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi).\end{aligned}$$

$$\phi = \frac{1}{n} \sum_{i=1}^n 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^n 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^n 1\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)} = 1\}}$$

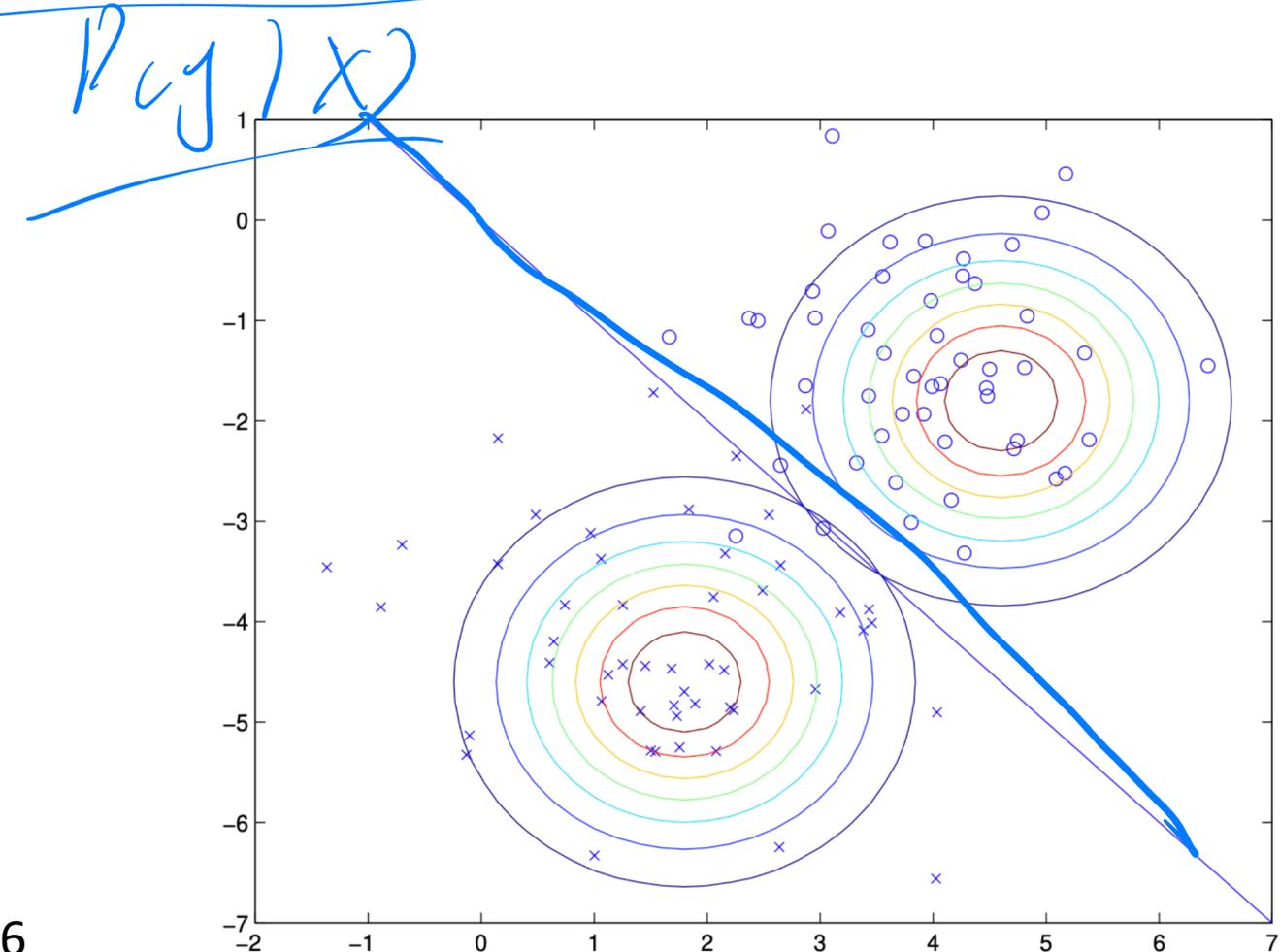
$$\Sigma = \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$$

counting

Recap: How did we do in GDA?

$$\begin{aligned}\ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^n p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^n p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi).\end{aligned}$$

$$\begin{aligned}\phi &= \frac{1}{n} \sum_{i=1}^n 1\{y^{(i)} = 1\} \\ \mu_0 &= \frac{\sum_{i=1}^n 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)} = 0\}} \\ \mu_1 &= \frac{\sum_{i=1}^n 1\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)} = 1\}} \\ \Sigma &= \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T\end{aligned}$$



The Generative Model

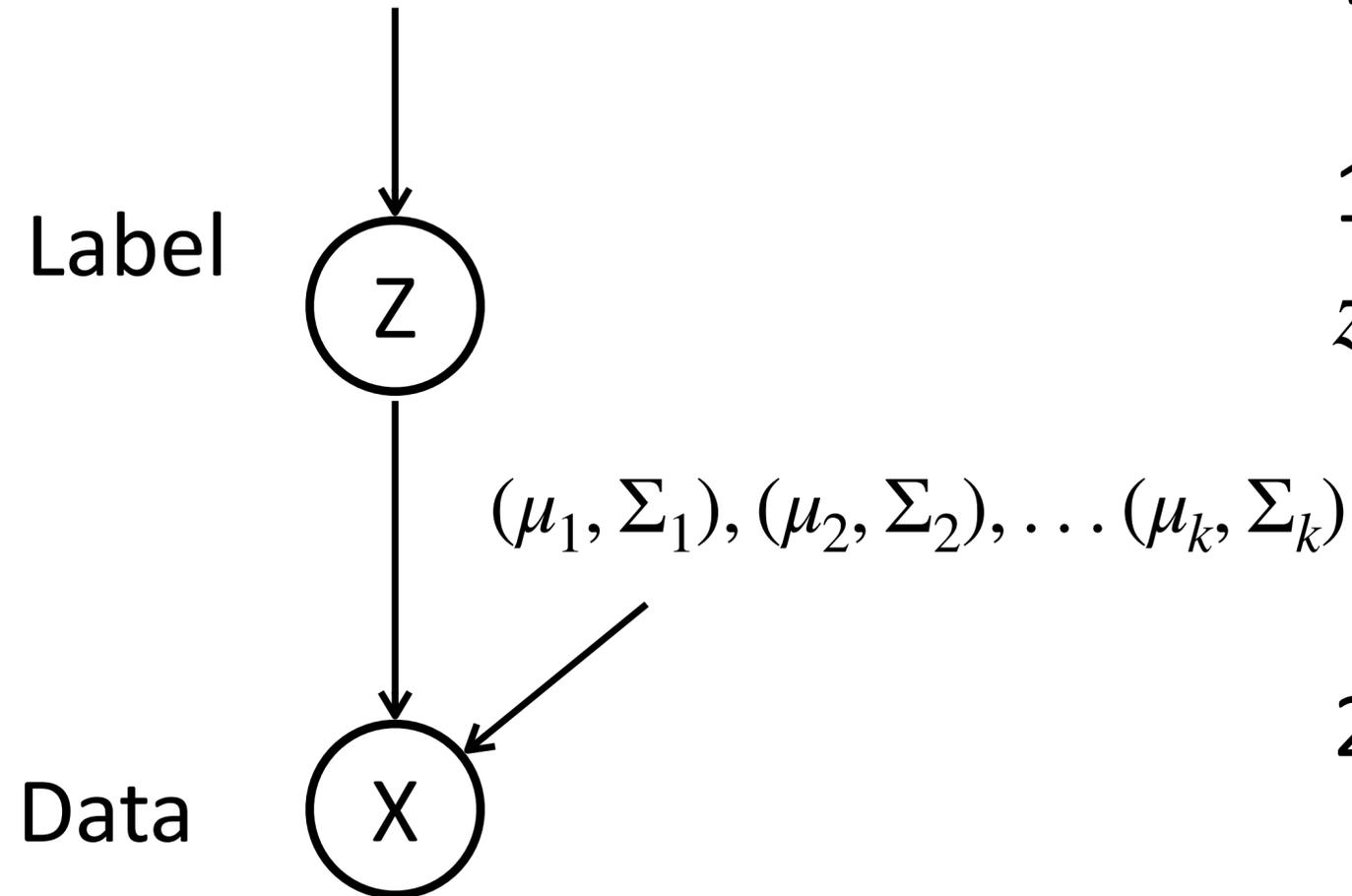
$p(z)$: multinomial, k classes (e.g. uniform)

K is a hyperparameter based on our assumption

We assume the generative process as:

1. For each data point, sample its label z_i from $p(z)$

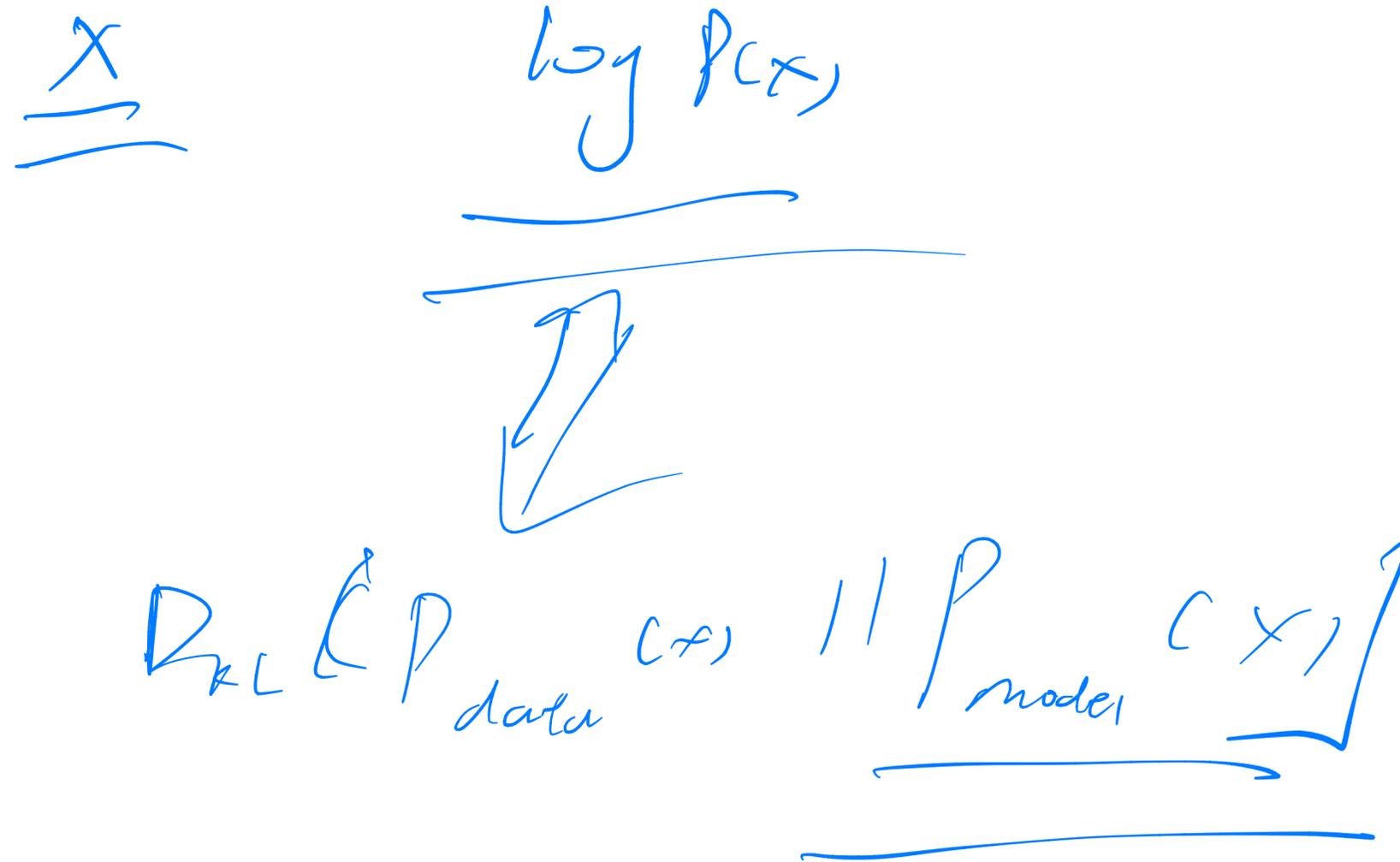
2. Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$



Same as Gaussian Discriminative Analysis, but Z is observed in GDA

Maximum Likelihood Estimation for GMM

Modeling data distribution is a fundamental goal in ML



Maximum Likelihood Estimation for GMM

Modeling data distribution is a fundamental goal in ML

Supervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x, z)$$

Maximum Likelihood Estimation for GMM

Modeling data distribution is a fundamental goal in ML

Supervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x, z)$$

Unsupervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x)$$


Maximum Likelihood Estimation for GMM

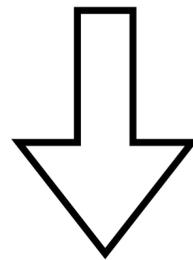
Modeling data distribution is a fundamental goal in ML

Supervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x, z)$$

Unsupervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x)$$



Prediction:

$$p(z | x) \propto p(z)p(x | z)$$

Maximum Likelihood Estimation for GMM

Modeling data distribution is a fundamental goal in ML

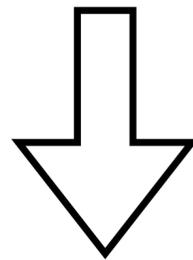
Supervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x, z)$$

Unsupervised:

$$\operatorname{argmax}_{\phi, \mu, \Sigma} \log p(x)$$

How to compute this?



Prediction:

$$p(z | x) \propto p(z)p(x | z)$$

Maximum Likelihood Estimation for GMM

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^n \log p(x^{(i)}; \phi, \mu, \Sigma)$$

$$= \sum_{i=1}^n \log \sum_{z^{(i)}=1}^k p(x^{(i)} | z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi).$$

$$p(x|z) p(z) = p(x, z)$$

$$p(x) = \sum_z p(x, z)$$

marginalization

Maximum Likelihood Estimation for GMM

$$\begin{aligned}\ell(\phi, \mu, \Sigma) &= \sum_{i=1}^n \log p(x^{(i)}; \phi, \mu, \Sigma) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}=1}^k p(x^{(i)} | z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi).\end{aligned}$$

1. Intractable (no closed-form for the solution)

Maximum Likelihood Estimation for GMM

$$\begin{aligned}\ell(\phi, \mu, \Sigma) &= \sum_{i=1}^n \log p(x^{(i)}; \phi, \mu, \Sigma) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}=1}^k p(x^{(i)} | z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi).\end{aligned}$$

z is continuous

1. Intractable (no closed-form for the solution)
2. Expensive when k is large (if you want to do gradient descent)

k = 10000



Things are easy when we know z ..

In case we know z

Things are easy when we know z ..

In case we know z

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^n \log p(x^{(i)} | z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi).$$

Things are easy when we know z .

In case we know z

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^n \log p(x^{(i)} | z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi).$$

$$\phi_j = \frac{1}{n} \sum_{i=1}^n 1\{z^{(i)} = j\},$$

$$\mu_j = \frac{\sum_{i=1}^n 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^n 1\{z^{(i)} = j\}},$$

$$\Sigma_j = \frac{\sum_{i=1}^n 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^n 1\{z^{(i)} = j\}}.$$

Things are easy when we know z .

In case we know z

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^n \log p(x^{(i)} | z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi).$$

$$\phi_j = \frac{1}{n} \sum_{i=1}^n 1\{z^{(i)} = j\},$$

$$\mu_j = \frac{\sum_{i=1}^n 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^n 1\{z^{(i)} = j\}},$$

$$\Sigma_j = \frac{\sum_{i=1}^n 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^n 1\{z^{(i)} = j\}}.$$

Expectation maximization is to infer the latent variables first (z here), and maximize the likelihood given the inferred z

Expectation Maximization for GMM

Repeat until convergence:

{

}

Expectation Maximization for GMM

Repeat until convergence:

{

(E-step) For each i, j , set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

}

Expectation Maximization for GMM

Repeat until convergence:

{

(E-step) For each i, j , set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

Compute the posterior distribution,
given current parameters

}

Expectation Maximization for GMM

Repeat until convergence:

{

No parameter change in E-step

(E-step) For each i, j , set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

Compute the posterior distribution,
given current parameters

}

Expectation Maximization for GMM

Repeat until convergence:

{

No parameter change in E-step

(E-step) For each i, j , set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

Compute the posterior distribution,
given current parameters

(M-step) Update the parameters:

$$\phi_j := \frac{1}{n} \sum_{i=1}^n w_j^{(i)},$$

$$\mu_j := \frac{\sum_{i=1}^n w_j^{(i)} x^{(i)}}{\sum_{i=1}^n w_j^{(i)}},$$

$$\Sigma_j := \frac{\sum_{i=1}^n w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^n w_j^{(i)}}$$

}

Expectation Maximization for GMM

Repeat until convergence:

{

No parameter change in E-step

(E-step) For each i, j , set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

Compute the posterior distribution,
given current parameters

(M-step) Update the parameters:

$$\phi_j := \frac{1}{n} \sum_{i=1}^n w_j^{(i)},$$

$$\mu_j := \frac{\sum_{i=1}^n w_j^{(i)} x^{(i)}}{\sum_{i=1}^n w_j^{(i)}},$$

$$\Sigma_j := \frac{\sum_{i=1}^n w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^n w_j^{(i)}}$$

update parameters using current $p(z | x)$

}

Expectation Maximization

- Why does it work?
- What is its relation to MLE estimation?
- How is convergence guaranteed?
- When we perform EM, what is the real objective that we are optimizing?

General EM Algorithm

General EM Algorithm

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

General EM Algorithm

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p(x^{(i)}; \theta) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta). \end{aligned}$$

General EM Algorithm

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p(x^{(i)}; \theta) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta). \end{aligned}$$

Let Q to be a distribution over z

General EM Algorithm

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p(x^{(i)}; \theta) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta). \end{aligned}$$

Let Q to be a distribution over z

$$\begin{aligned} \log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)} \end{aligned}$$

General EM Algorithm

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p(x^{(i)}; \theta) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta). \end{aligned}$$

Let Q to be a distribution over z

This lower bound holds for any $Q(z)$

$$\begin{aligned} \log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)} \end{aligned}$$

General EM Algorithm

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p(x^{(i)}; \theta) \\ &= \sum_{i=1}^n \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta). \end{aligned}$$

Let Q to be a distribution over z

This lower bound holds for any $Q(z)$

$$\begin{aligned} \log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)} \end{aligned}$$

Jensen inequality

Jensen Inequality

For a convex function f , and $t \in [0,1]$

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

Jensen Inequality

For a convex function f , and $t \in [0,1]$

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

In probability:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$

Jensen Inequality

For a convex function f , and $t \in [0,1]$

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

In probability:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$

If f is strictly convex, then equality holds only when X is a constant

Evidence Lower Bound (ELBO)

Evidence Lower Bound (ELBO)

$$\begin{aligned}\log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}\end{aligned}$$

Evidence Lower Bound (ELBO)

$$\begin{aligned}\log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} && \text{ELBO} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}\end{aligned}$$

Evidence Lower Bound (ELBO)

$$\begin{aligned}\log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} && \text{ELBO} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}\end{aligned}$$

Because the log likelihood is intractable, people often optimize its lower bound instead

Evidence Lower Bound (ELBO)

$$\begin{aligned}\log p(x; \theta) &= \log \sum_z p(x, z; \theta) \\ &= \log \sum_z Q(z) \frac{p(x, z; \theta)}{Q(z)} && \text{ELBO} \\ &\geq \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}\end{aligned}$$

Because the log likelihood is intractable, people often optimize its lower bound instead

Why optimizing lower bound works? How to choose $Q(z)$, why we computed posterior in the E step, what is the benefit?

Thank You!
Q & A