Expectation Maximization
Mixture Models
Clustering

Class 14. 6 Oct 2010

Administrivia

- Many homeworks still due
- Is everyone on the “projects” page?
  - Where are your project proposals?

Covered

- Learning distributions from data
  - Given a collection of examples from some data, estimate its distribution
  - Solution: Assign a model to the distribution
  - Learn parameters of model from data
- Complex models: Learning must be done using Expectation Maximization
- Following slides: An intuitive explanation using a simple example of multinomials

A Thought Experiment

COVERED

6 3 1 5 4 1 2 4 …

- A person shoots a loaded dice repeatedly
- You observe the series of outcomes
- **You can form a good idea of how the dice is loaded**
  - Figure out what the probabilities of the various numbers are for dice
- P(number) = count(number)/sum(rolls)
- This is a *maximum likelihood* estimate
  - Estimate that makes the observed sequence of numbers most probable

The Multinomial Distribution

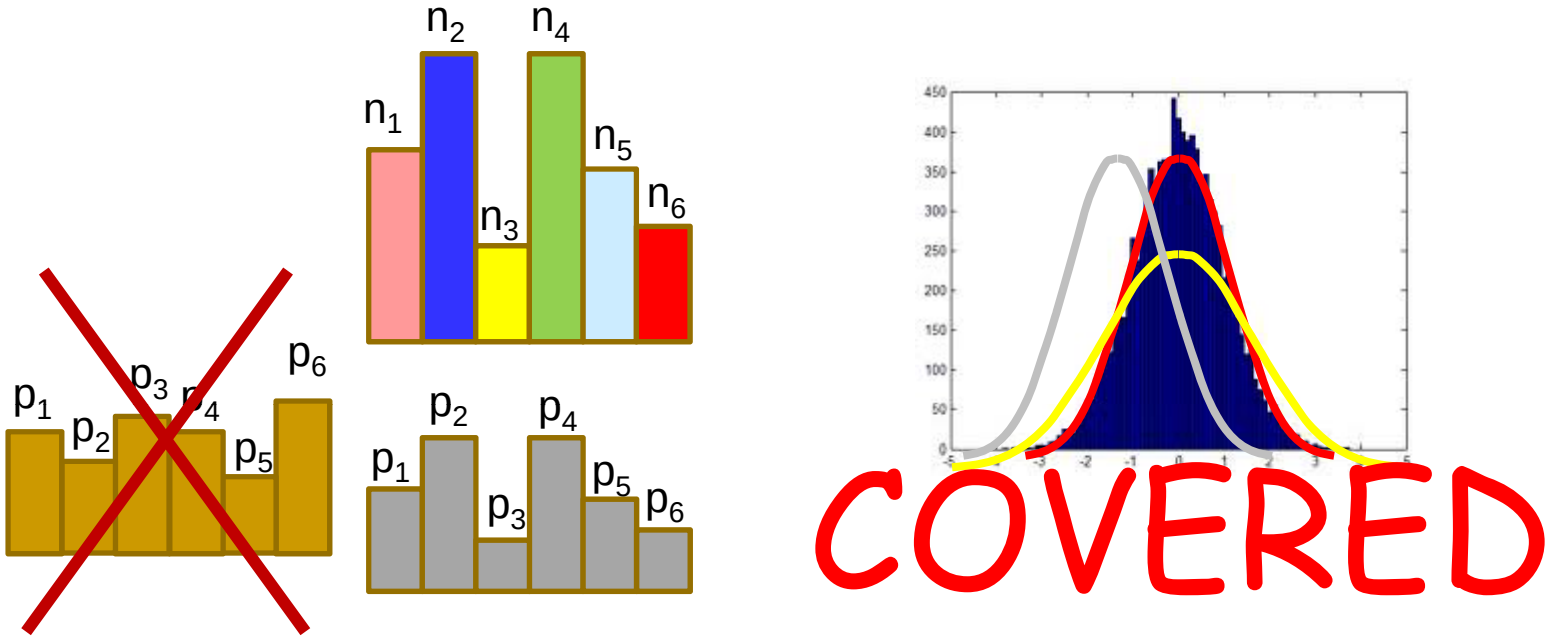
- A probability distribution over a discrete collection of items is a *Multinomial*

$$P(X : X \text{ belongs to a discrete set}) = P(X)$$

- E.g. the roll of dice
  - X : X in (1,2,3,4,5,6)
- Or the toss of a coin
  - X : X in (head, tails)

COVERED

Maximum Likelihood Estimation



- Basic principle: Assign a form to the distribution
  - E.g. a multinomial
  - Or a Gaussian
- Find the *distribution* that best fits the histogram of the data

Defining "Best Fit"

- The data are generated by draws from the distribution
  - I.e. the generating process draws from the distribution

COVERED

- Assumption: The distribution has a high probability of generating the observed data
  - Not necessarily true
- Select the distribution that has the *highest* probability of generating the data
  - Should assign lower probability to less frequent observations and vice versa

Maximum Likelihood Estimation: Multinomial

- Probability of generating (n_1, n_2, n_3, n_4, n_5, n_6)

$$P(n_1, n_2, n_3, n_4, n_5, n_6) = Const \prod_i p_i^{n_i}$$

- Find p_1,p_2,p_3,p_4,p_5,p_6 so that the above is maximized
- Alternately maximize

COVERED

$$\log\left(P(n_1, n_2, n_3, n_4, n_5, n_6)\right) = \log(Const) + \sum_i n_i \log\left(p_i\right)$$

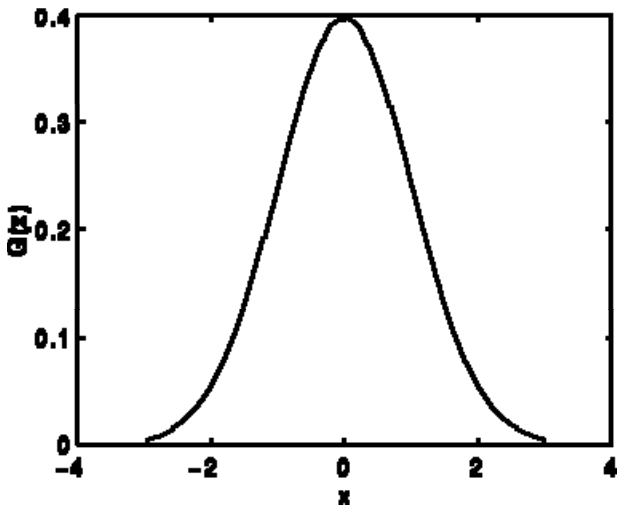
  - Log() is a monotonic function
    - argmax_x f(x) = argmax_x log(f(x))

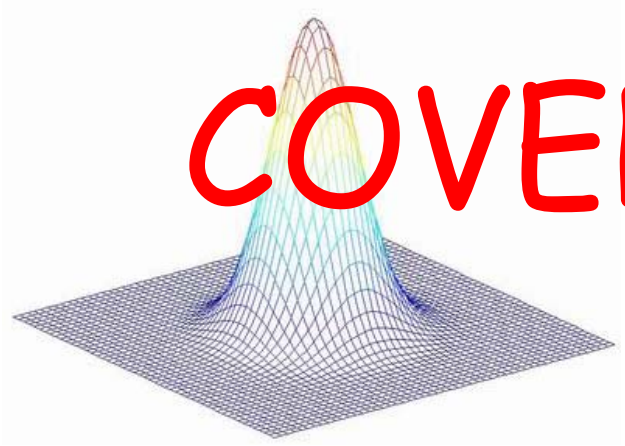
- Solving for the probabilities gives us
  - Requires constrained optimization to ensure probabilities sum to 1

$$p_i = \frac{n_i}{\sum_j n_j}$$

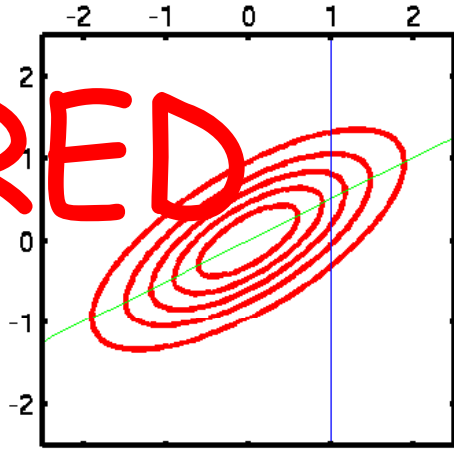
EVENTUALLY ITS JUST COUNTING!

Segue: Gaussians





COVERED



$$P(X) = N(X;\mu,\Theta) = \frac{1}{\sqrt{(2\pi)^d\,|\Theta|}}\exp\left(-0.5(X-\mu)^T\Theta^{-1}(X-\mu)\right)$$

- Parameters of a Gaussian:
  - Mean μ, Covariance Θ

Maximum Likelihood: Gaussian

- Given a collection of observations $(X_1, X_2, \ldots)$, estimate mean μ and covariance Θ

COVERED

$$P(X_1, X_2, \ldots) = \prod_i \frac{1}{\sqrt{(2\pi)^d \,|\Theta|}} \exp\left(-0.5(X_i - \mu)^T \Theta^{-1}(X_i - \mu)\right)$$

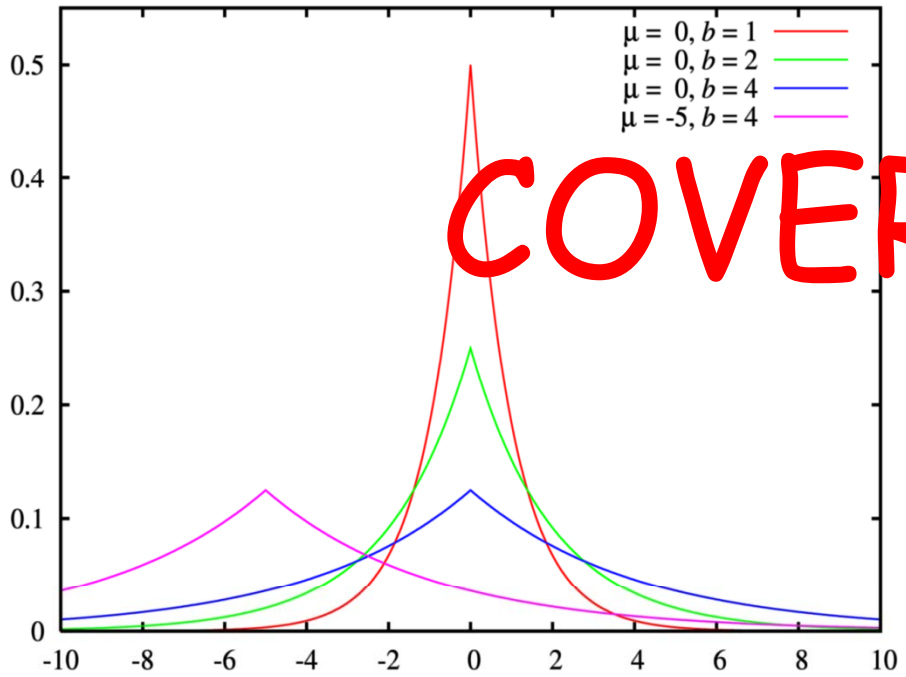
$$\log\left(P(X_1, X_2, \ldots)\right) = C - 0.5\sum_i \left(\log(|\Theta|) + (X_i - \mu)^T \Theta^{-1}(X_i - \mu)\right)$$

- Maximizing w.r.t μ and Θ gives us

$$\mu = \frac{1}{N}\sum_i X_i \qquad \Theta = \frac{1}{N}\sum_i (X_i - \mu)(X_i - \mu)^T$$

ITS STILL JUST COUNTING!

Laplacian



COVERED

$$P(x) = L(x;\mu,b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$$

- Parameters: Mean μ, scale b ($b > 0$)

Maximum Likelihood: Laplacian

- Given a collection of observations (x_1, x_2,…), estimate mean μ and scale b

COVERED

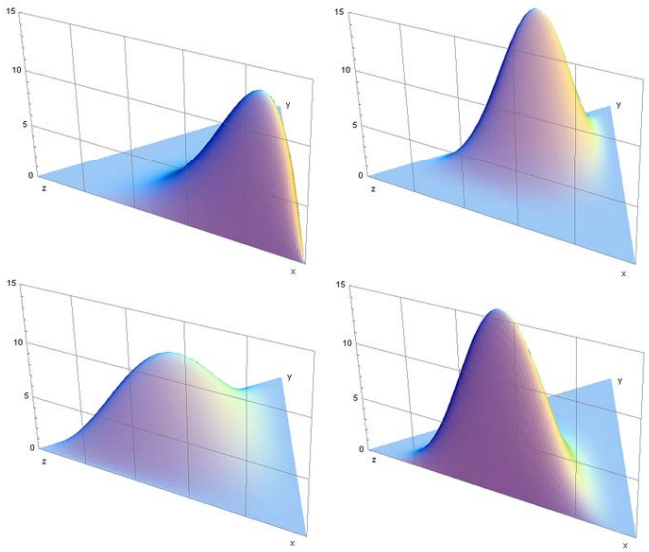
$$\log\big(P(x_1, x_2, \ldots)\big) = C - N\log(b) - \sum_i \frac{|x_i - \mu|}{b}$$

- Maximizing w.r.t μ and b gives us

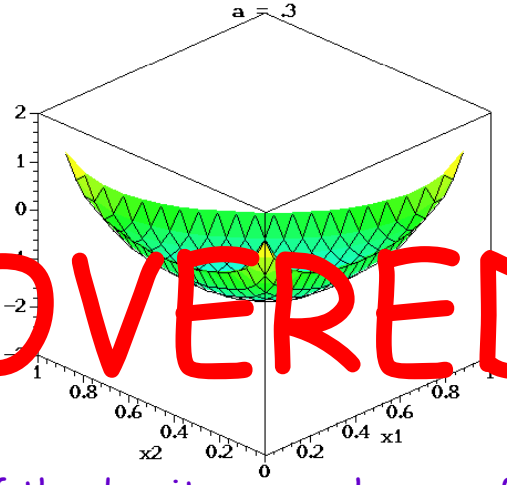
$$\mu = \frac{1}{N}\sum_i x_i \qquad\qquad b = \frac{1}{N}\sum_i |x_i - \mu|$$

Dirichlet

(from wikipedia)



K=3. Clockwise from top left:
α=(6, 2, 2), (3, 7, 5), (6, 2, 6), (2, 3, 4)



COVERED

log of the density as we change a from α=(0.3, 0.3, 0.3) to (2.0, 2.0, 2.0), keeping all the individual αi's equal to each other.

$$P(X) = D(X;\alpha) = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma\left(\sum_i \alpha_i\right)} \prod_i x_i^{\alpha_i - 1}$$

- Parameters are αs
  - Determine mode and curvature
- Defined only of probability vectors
  - $X = [x_1\ x_2\ ..\ x_K]$, $\Sigma_i\ x_i = 1$, $x_i >= 0$ for all i

Maximum Likelihood: Dirichlet

- Given a collection of observations (X_1, X_2,…), estimate α

COVERED

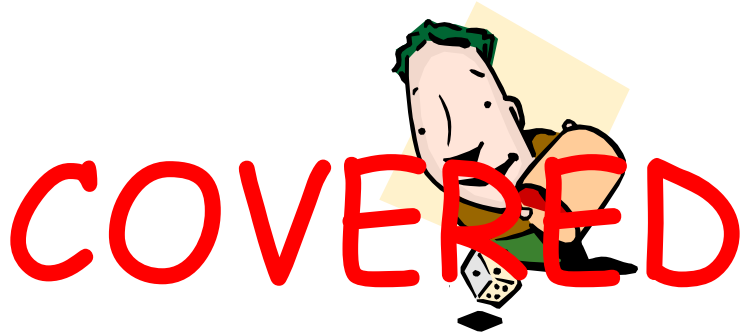
$$\log\left(P(X_1, X_2, \ldots)\right) = \sum_j \sum_i (\alpha_i - 1)\log(X_{j,i}) + N\sum_i \log\left(\Gamma(\alpha_i)\right) - N\log\left(\Gamma\left(\sum_i \alpha_i\right)\right)$$

- No closed form solution for αs.
  - Needs gradient ascent

- Several distributions have this property: the ML estimate of their parameters have no closed form solution

Continuing the Thought Experiment

6 3 1 5 4 1 2 4 …



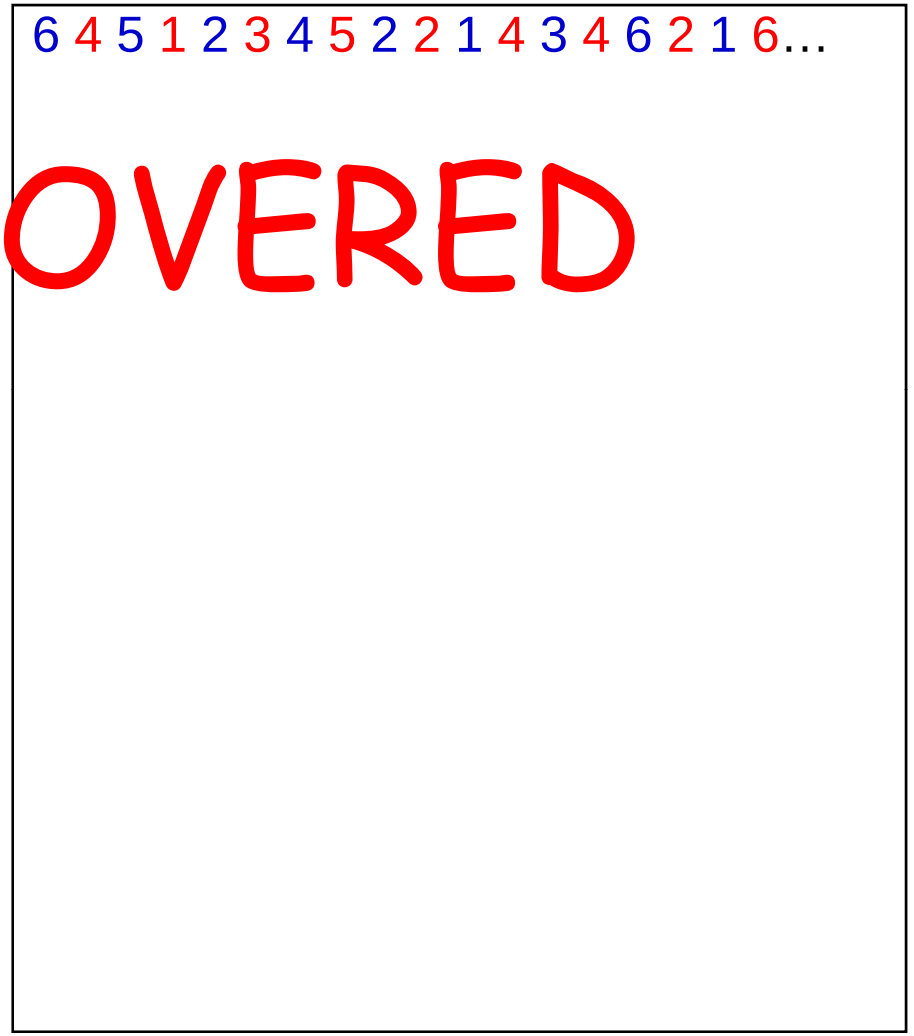
4 4 1 6 3 2 1 2 …

- Two persons shoot loaded dice repeatedly
  - The dice are differently loaded for the two of them
- We observe the series of outcomes for both persons

- **How to determine the probability distributions of the two dice?**

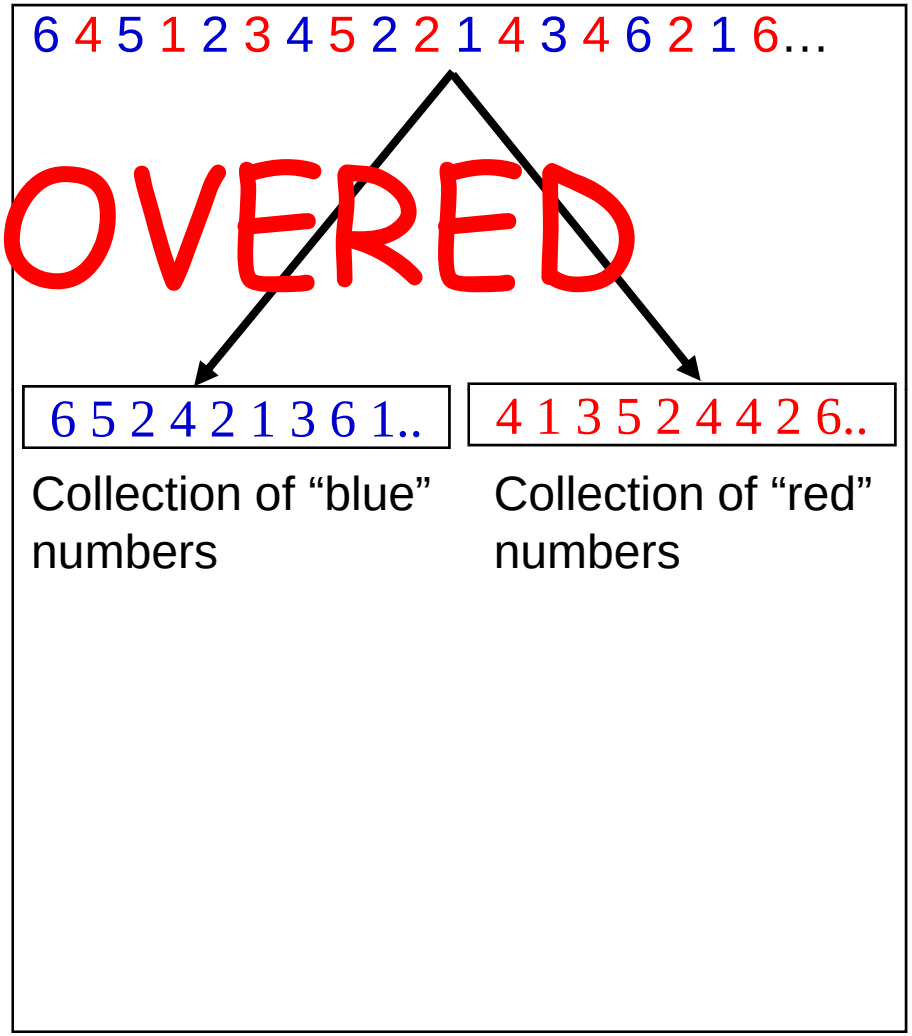
Estimating Probabilities

- Observation: The sequence of numbers from the two dice
  - As indicated by the colors, we know who rolled what number



Estimating Probabilities

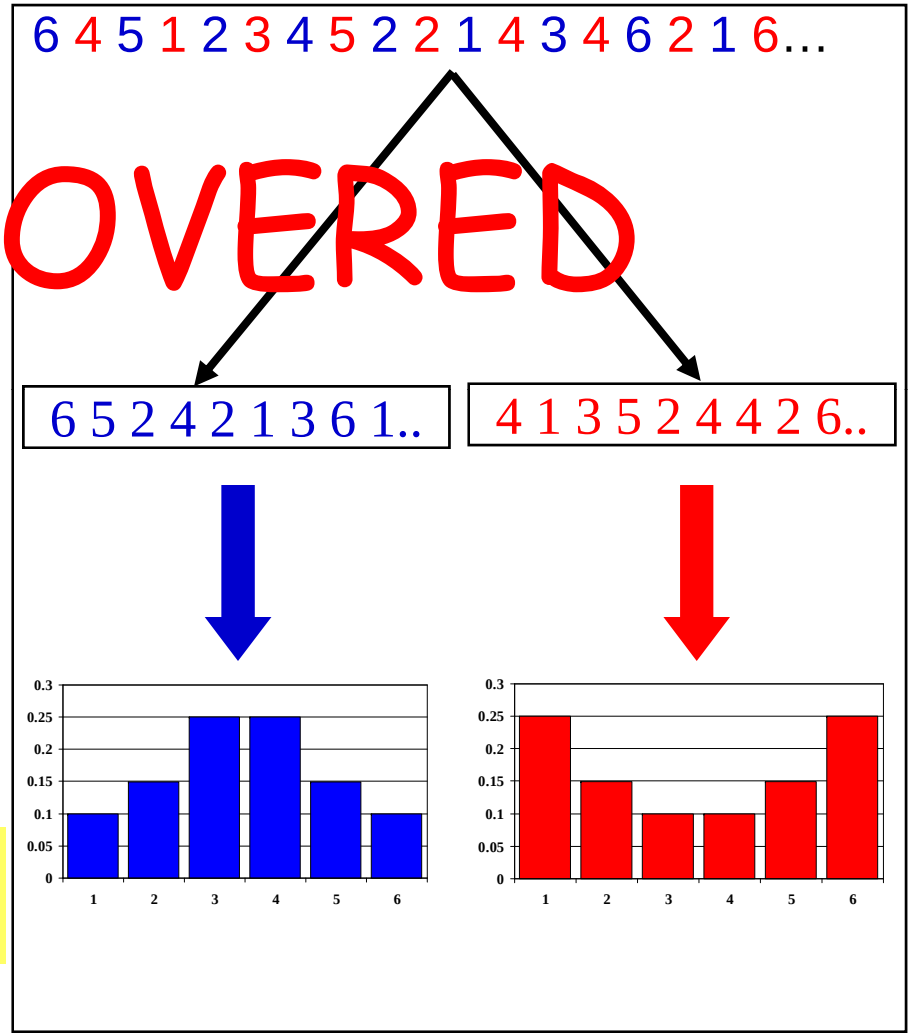
- Observation: The sequence of numbers from the two dice
  - As indicated by the colors, we know who rolled what number
- Segregation: Separate the blue observations from the red



Estimating Probabilities

- Observation: The sequence of numbers from the two dice
  - As indicated by the colors, we know who rolled what number
- Segregation: Separate the blue observations from the red
- From each set compute probabilities for each of the 6 possible outcomes

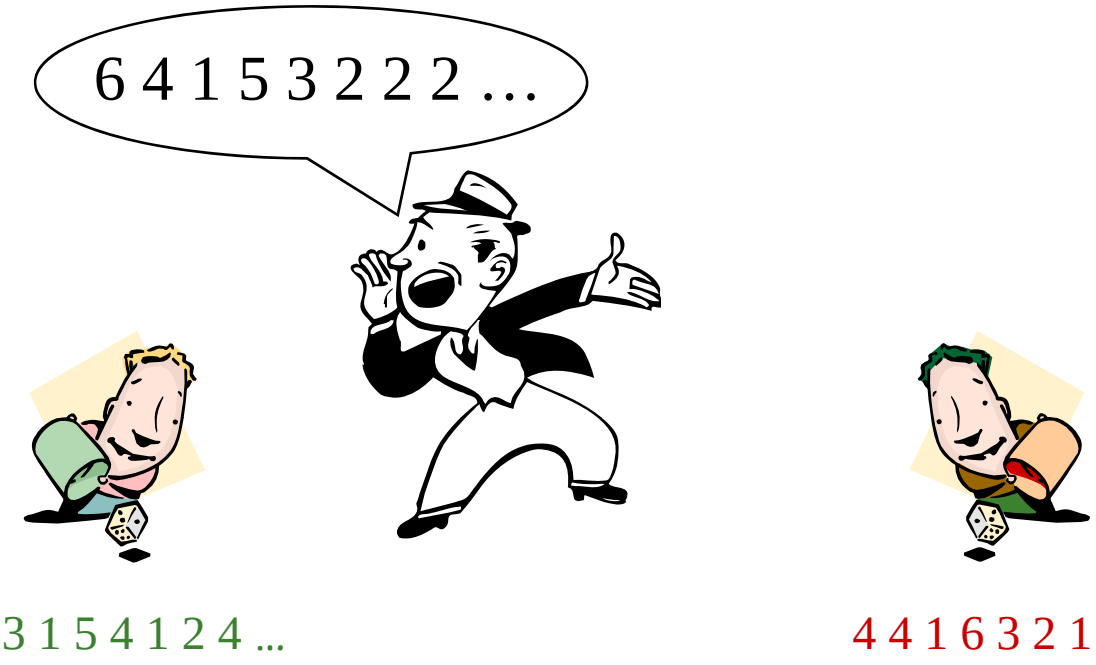
$$P(number) = \frac{\text{no. of times number was rolled}}{\text{total number of observed rolls}}$$



A Thought Experiment

- Now imagine that you cannot observe the dice yourself
- Instead there is a “caller” who randomly calls out the outcomes
  - 40% of the time he calls out the number from the left shooter, and 60% of the time, the one from the right (and you know this)
- At any time, you do not know which of the two he is calling out
- How do you determine the probability distributions for the two dice?

A Thought Experiment



- How do you now determine the probability distributions for the two sets of dice …
- .. If you do not even know what fraction of time the blue numbers are called, and what fraction are red?

A Mixture Multinomial

- The caller will call out a number X in any given callout IF
  - He selects “RED”, and the Red die rolls the number X
  - OR
  - He selects “BLUE” and the Blue die rolls the number X

- P(X) = P(Red)P(X|Red) + P(Blue)P(X|Blue)
  - E.g. P(6) = P(Red)P(6|Red) + P(Blue)P(6|Blue)

- A distribution that *combines* (or *mixes*) multiple multinomials is a *mixture* multinomial

$$P(X) = \sum_{Z} P(Z)P(X \mid Z)$$

Mixture weights Component multinomials

Mixture Distributions

$$P(X) = \sum_Z P(Z)P(X \mid Z)$$

Mixture weights

Component distributions

Mixture Gaussian

$$P(X) = \sum_Z P(Z)N(X;\mu_z,\Theta_z)$$

Mixture of Gaussians and Laplacians

$$P(X) = \sum_Z P(Z)N(X;\mu_z,\Theta_z) + \sum_Z P(Z)\prod_i L(X_i;\mu_z,b_{z,i})$$

- Mixture distributions mix several component distributions
  - Component distributions may be of varied type
- Mixing weights must sum to 1.0
- Component distributions integrate to 1.0
- Mixture distribution integrates to 1.0

Maximum Likelihood Estimation

- For our problem: $$P(X) = \sum_Z P(Z)P(X \mid Z)$$
  - Z = color of dice

$$P(n_1, n_2, n_3, n_4, n_5, n_6) = Const \prod_X P(X)^{n_X} = Const \prod_X \left(\sum_Z P(Z)P(X \mid Z) \right)^{n_X}$$

- Maximum likelihood solution: Maximize

$$\log(P(n_1, n_2, n_3, n_4, n_5, n_6)) = \log(Const) + \sum_X n_X \log \left(\sum_Z P(Z)P(X \mid Z) \right)$$

- No closed form solution (summation inside log)!
  - In general ML estimates for mixtures do not have a closed form
  - USE EM!

Expectation Maximization

- It is possible to estimate all parameters in this setup using the Expectation Maximization (or EM) algorithm
- First described in a landmark paper by Dempster, Laird and Rubin
  - Maximum Likelihood Estimation from incomplete data, via the EM Algorithm, Journal of the Royal Statistical Society, Series B, 1977
- Much work on the algorithm since then
- The principles behind the algorithm existed for several years prior to the landmark paper, however.

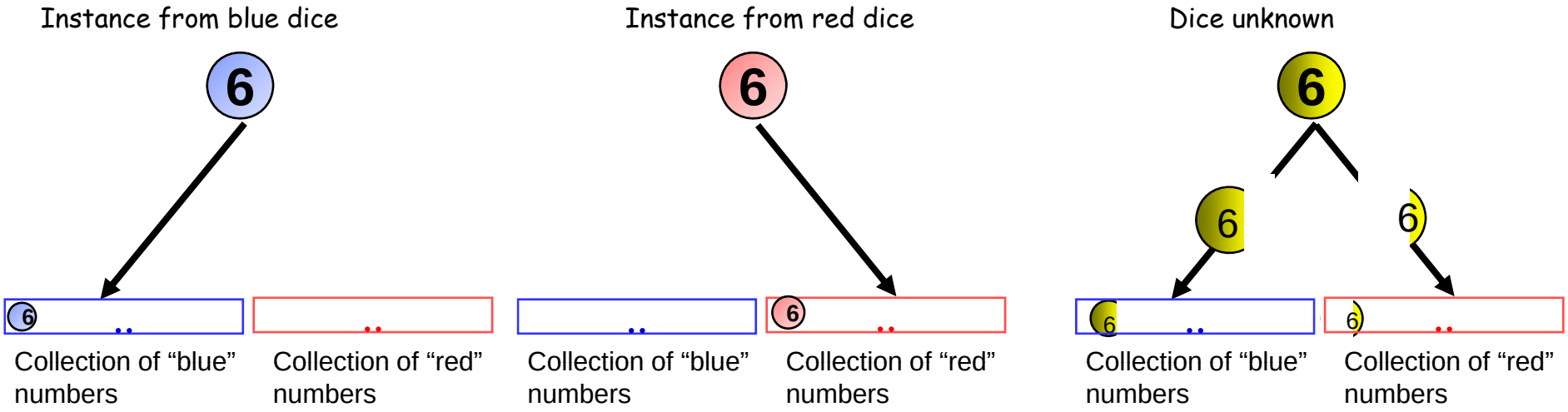
Expectation Maximization

- Iterative solution
- Get some initial estimates for all parameters
  - Dice shooter example: This includes probability distributions for dice AND the probability with which the caller selects the dice
- Two steps that are iterated:
  - ***Expectation Step:*** Estimate statistically, the values of ***unseen*** variables
  - ***Maximization Step:*** Using the estimated values of the unseen variables as truth, estimates of the model parameters

EM: The auxiliary function

- EM iteratively optimizes the following auxiliary function
- $Q(\theta, \theta') = \Sigma_Z P(Z|X,\theta') \log(P(Z,X \mid \theta))$
  - Z are the unseen variables
  - Assuming Z is discrete (may not be)
- θ' are the parameter estimates from the previous iteration
- θ are the estimates to be obtained in the current iteration

Expectation Maximization as counting



- Hidden variable: *Z*
  - Dice: The identity of the dice whose number has been called out
- If we knew *Z* for every observation, we could estimate all terms
  - By adding the observation to the correct bin
- Unfortunately, we do not know *Z* – it is hidden from us!
- Solution: FRAGMENT THE OBSERVATION

Fragmenting the Observation

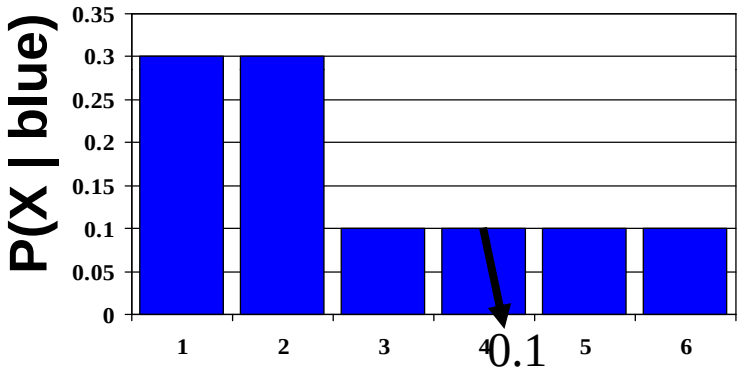
- EM is an iterative algorithm
  - At each time there is a *current* estimate of parameters
- The "size" of the fragments is proportional to the *a posteriori probability* of the component distributions
  - The *a posteriori* probabilities of the various values of *Z* are computed using Bayes' rule:

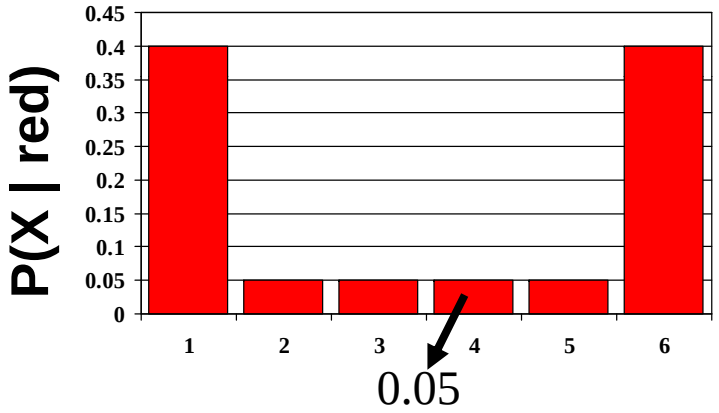
$$P(Z \mid X) = \frac{P(X \mid Z)P(Z)}{P(X)} = CP(X \mid Z)P(Z)$$

- Every dice gets a fragment of size P(dice | number)

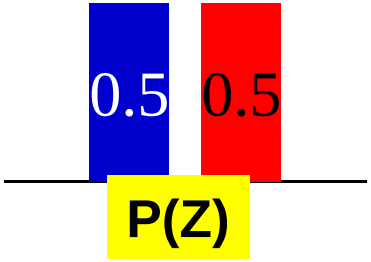
Expectation Maximization

- Hypothetical Dice Shooter Example:
- We obtain an initial estimate for the probability distribution of the two sets of dice (somehow):





- We obtain an initial estimate for the probability with which the caller calls out the two shooters (somehow)



Expectation Maximization

- Hypothetical Dice Shooter Example:
- Initial estimate:
  - P(blue) = P(red) = 0.5
  - P(4 | blue) = 0.1, for P(4 | red) = 0.05

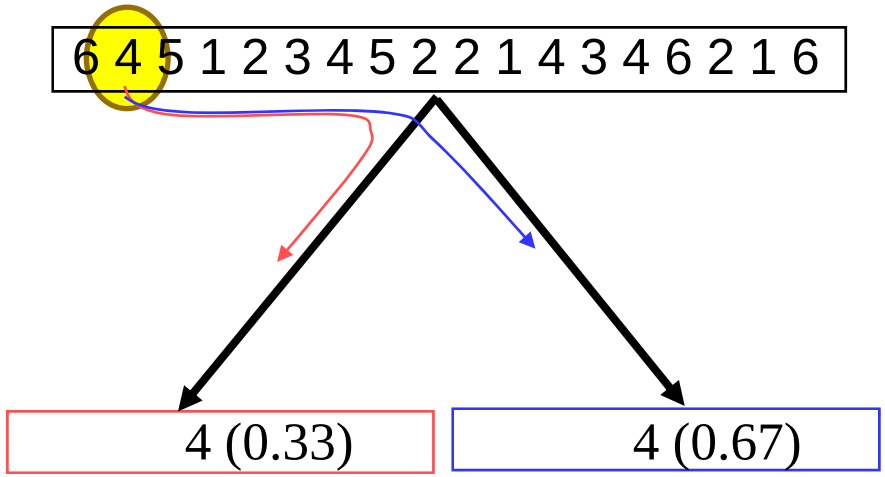
- Caller has just called out 4
- Posterior probability of colors:

$$P(red \mid X = 4) = CP(X = 4 \mid Z = red)P(Z = red) = C \times 0.05 \times 0.5 = C0.025$$

$$P(blue \mid X = 4) = CP(X = 4 \mid Z = blue)P(Z = blue) = C \times 0.1 \times 0.5 = C0.05$$

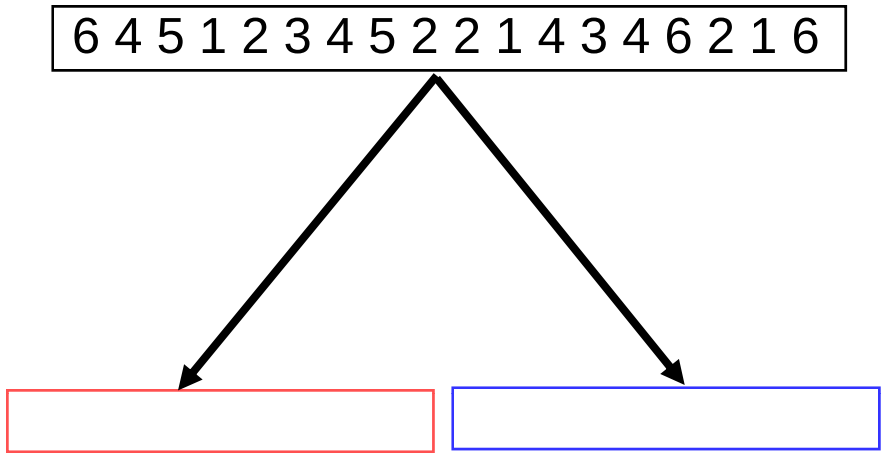
Normalizing : $P(red \mid X = 4) = 0.33$**;** $P(blue \mid X = 4) = 0.67$

Expectation **Maximization**



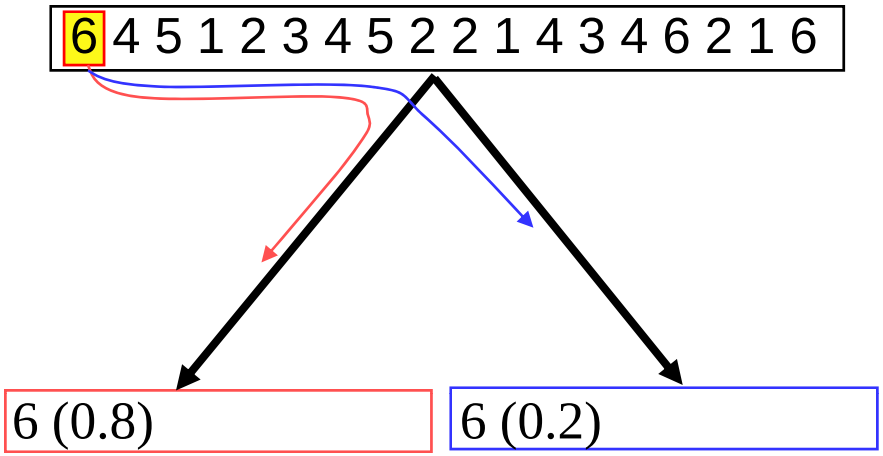
Expectation **Maximization**

- Every observed roll of the dice contributes to both “Red” and “Blue”



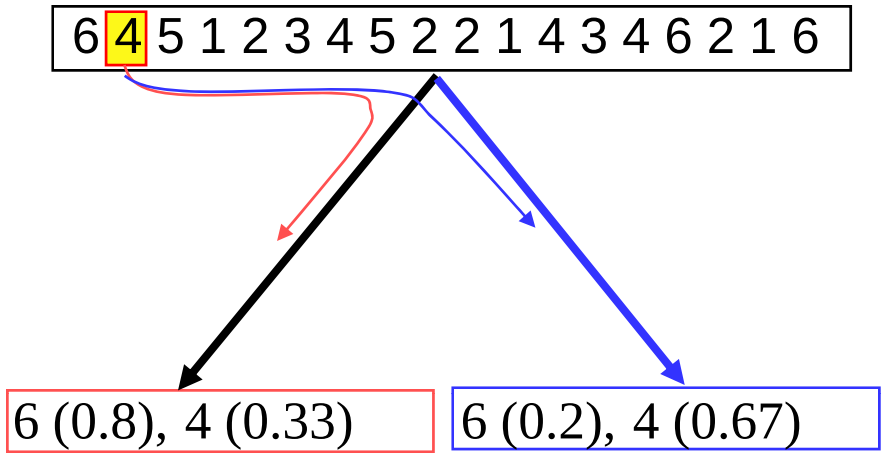
Expectation **Maximization**

- Every observed roll of the dice contributes to both “Red” and “Blue”



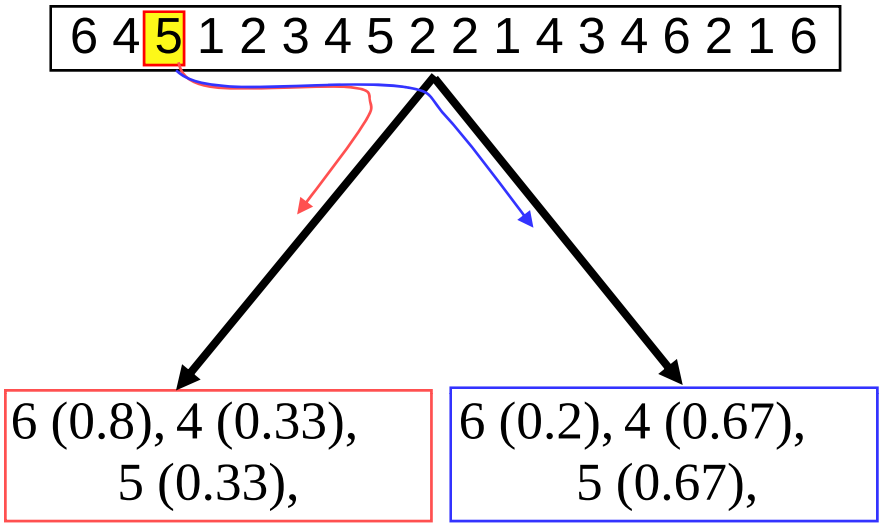
Expectation **Maximization**

- Every observed roll of the dice contributes to both “Red” and “Blue”



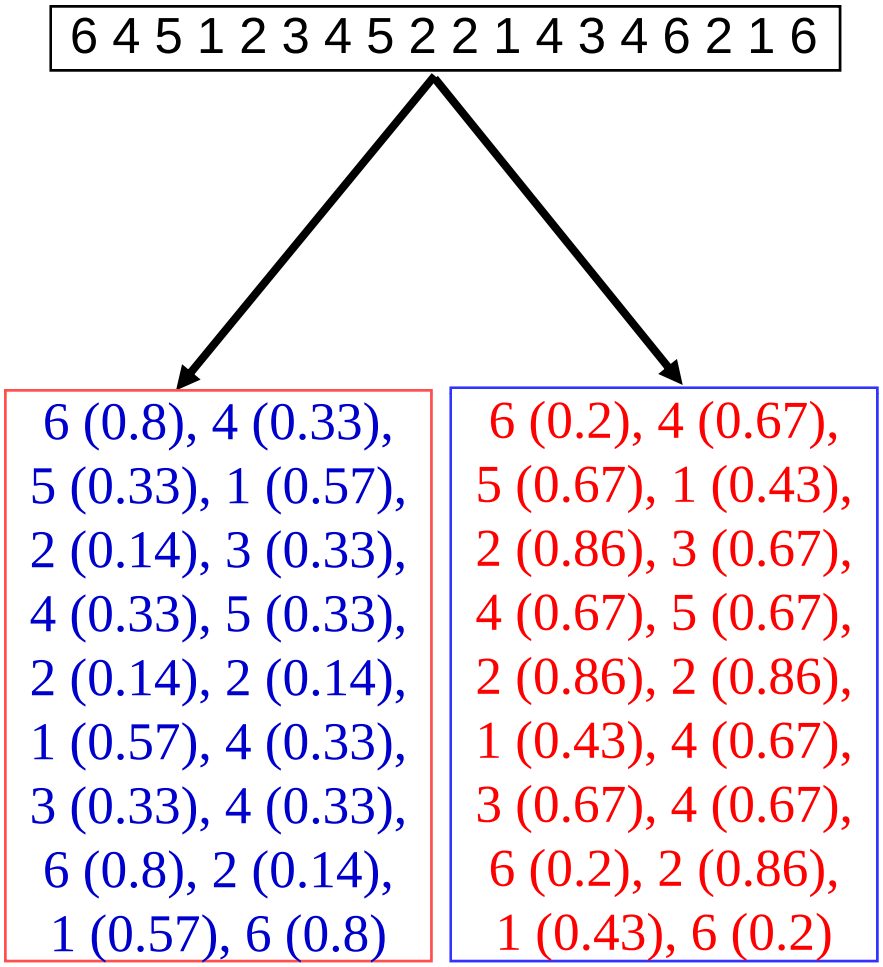
Expectation **Maximization**

- Every observed roll of the dice contributes to both “Red” and “Blue”



Expectation **Maximization**

- Every observed roll of the dice contributes to both “Red” and “Blue”



Expectation **Maximization**

- Every observed roll of the dice contributes to both “Red” and “Blue”
- Total count for “Red” is the sum of all the posterior probabilities in the red column
  - 7.31
- Total count for “Blue” is the sum of all the posterior probabilities in the blue column
  - 10.69
  - Note: 10.69 + 7.31 = 18 = the total number of instances

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Red:
  - Total count for 1: 1.71

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Red:
  - Total count for 1: 1.71
  - Total count for 2: 0.56

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Red:
  - Total count for 1: 1.71
  - Total count for 2: 0.56
  - Total count for 3: 0.66

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Red:
  - Total count for 1: 1.71
  - Total count for 2: 0.56
  - Total count for 3: 0.66
  - Total count for 4: 1.32

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Red:
  - Total count for 1: 1.71
  - Total count for 2: 0.56
  - Total count for 3: 0.66
  - Total count for 4: 1.32
  - Total count for 5: 0.66

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Red:
  - Total count for 1: 1.71
  - Total count for 2: 0.56
  - Total count for 3: 0.66
  - Total count for 4: 1.32
  - Total count for 5: 0.66
  - Total count for 6: 2.4

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation Maximization

- Total count for "Red" : 7.31
- Red:
  - Total count for 1: 1.71
  - Total count for 2: 0.56
  - Total count for 3: 0.66
  - Total count for 4: 1.32
  - Total count for 5: 0.66
  - Total count for 6: 2.4

- **Updated probability of Red dice:**
  - P(1 | Red) = 1.71/7.31 = 0.234
  - P(2 | Red) = 0.56/7.31 = 0.077
  - P(3 | Red) = 0.66/7.31 = 0.090
  - P(4 | Red) = 1.32/7.31 = 0.181
  - P(5 | Red) = 0.66/7.31 = 0.090
  - P(6 | Red) = 2.40/7.31 = 0.328

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Blue” : 10.69
- Blue:
  - Total count for 1: 1.29

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Blue” : 10.69
- Blue:
  - Total count for 1: 1.29
  - Total count for 2: 3.44

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Blue” : 10.69
- Blue:
  - Total count for 1: 1.29
  - Total count for 2: 3.44
  - Total count for 3: 1.34

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Blue” : 10.69
- Blue:
  - Total count for 1: 1.29
  - Total count for 2: 3.44
  - Total count for 3: 1.34
  - Total count for 4: 2.68

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Blue” : 10.69
- Blue:
  - Total count for 1: 1.29
  - Total count for 2: 3.44
  - Total count for 3: 1.34
  - Total count for 4: 2.68
  - Total count for 5: 1.34

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Blue” : 10.69
- Blue:
  - Total count for 1: 1.29
  - Total count for 2: 3.44
  - Total count for 3: 1.34
  - Total count for 4: 2.68
  - Total count for 5: 1.34
  - Total count for 6: 0.6

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation Maximization

- Total count for "Blue" : 10.69
- Blue:
  - Total count for 1: 1.29
  - Total count for 2: 3.44
  - Total count for 3: 1.34
  - Total count for 4: 2.68
  - Total count for 5: 1.34
  - Total count for 6: 0.6

- **Updated probability of Blue dice:**
  - P(1 | Blue) = 1.29/11.69 = 0.122
  - P(2 | Blue) = 0.56/11.69 = 0.322
  - P(3 | Blue) = 0.66/11.69 = 0.125
  - P(4 | Blue) = 1.32/11.69 = 0.250
  - P(5 | Blue) = 0.66/11.69 = 0.125
  - P(6 | Blue) = 2.40/11.69 = 0.056

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

Expectation **Maximization**

- Total count for “Red” : 7.31
- Total count for “Blue” : 10.69
- Total instances = 18
  - Note 7.31+10.69 = 18
- We also revise our estimate for the probability that the caller calls out Red or Blue
  - i.e the fraction of times that he calls Red and the fraction of times he calls Blue

- P(Z=Red) = 7.31/18 = 0.41
- P(Z=Blue) = 10.69/18 = 0.59

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

7.31 10.69

The updated values

- Probability of Red dice:
  - P(1 | Red) = 1.71/7.31 = 0.234
  - P(2 | Red) = 0.56/7.31 = 0.077
  - P(3 | Red) = 0.66/7.31 = 0.090
  - P(4 | Red) = 1.32/7.31 = 0.181
  - P(5 | Red) = 0.66/7.31 = 0.090
  - P(6 | Red) = 2.40/7.31 = 0.328
- Probability of Blue dice:
  - P(1 | Blue) = 1.29/11.69 = 0.122
  - P(2 | Blue) = 0.56/11.69 = 0.322
  - P(3 | Blue) = 0.66/11.69 = 0.125
  - P(4 | Blue) = 1.32/11.69 = 0.250
  - P(5 | Blue) = 0.66/11.69 = 0.125
  - P(6 | Blue) = 2.40/11.69 = 0.056
- P(Z=Red) = 7.31/18 = 0.41
- P(Z=Blue) = 10.69/18 = 0.59

Called	P(red|X)	P(blue|X)
6	.8	.2
4	.33	.67
5	.33	.67
1	.57	.43
2	.14	.86
3	.33	.67
4	.33	.67
5	.33	.67
2	.14	.86
2	.14	.86
1	.57	.43
4	.33	.67
3	.33	.67
4	.33	.67
6	.8	.2
2	.14	.86
1	.57	.43
6	.8	.2

THE UPDATED VALUES CAN BE USED TO REPEAT THE PROCESS. ESTIMATION IS AN ITERATIVE PROCESS

The Dice Shooter Example

6 3 1 5 4 1 2 4 …

4 4 1 6 3 2 1 2 …

1. Initialize $P(Z)$, $P(X \mid Z)$
2. Estimate $P(Z \mid X)$ for each Z, for each called out number
   - Assign X with to value of Z, with weight $P(Z \mid X)$
3. Re-estimate $P(X \mid Z)$ for every value of X and Z
4. Re-estimate $P(Z)$
5. If not converged, return to 2

In Squiggles

- Given a sequence of observations O_1, O_2, ..
  - N_X is the number of observations of number X
- Initialize P(Z), P(X|Z) for dice Z and numbers X
- Iterate:
  - For each number X:

$$P(Z \mid X) = \frac{P(X \mid Z)P(Z)}{\sum_{Z'} P(Z')P(X \mid Z')}$$

  - Update:

$$P(X \mid Z) = \frac{\sum_{O \text{ such that } O==X} P(Z \mid X)}{\sum_{O} P(Z \mid O)} = \frac{N_X P(Z \mid X)}{\sum_{X} N_X P(Z \mid X)}$$

$$P(Z) = \frac{\sum_{X} N_X P(Z \mid X)}{\sum_{Z'} \sum_{X} N_X P(Z' \mid X)}$$

Solutions may not be unique

- The EM algorithm will give us one of many solutions, all equally valid!
  - The probability of 6 being called out:

$$P(6) = \alpha P(6 \mid red) + \beta P(6 \mid blue) = \alpha P_r + \beta P_b$$

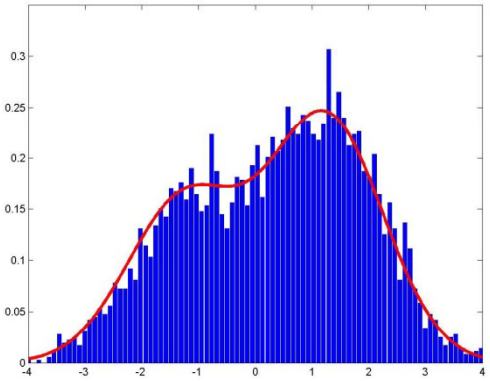
    - Assigns P_r as the probability of 6 for the red die
    - Assigns P_b as the probability of 6 for the blue die

  - The following too is a valid solution

$$P(6) = 1.0(\alpha P_r + \beta P_b) + 0.0 anything$$

    - Assigns 1.0 as the a priori probability of the red die
    - Assigns 0.0 as the probability of the blue die
- The solution is NOT unique

A More Complex Model



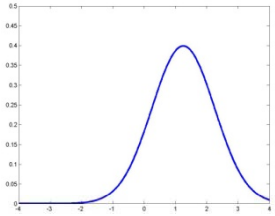
$$P(X) = \sum_k P(k) N(X; \mu_k, \Theta_k) = \sum_k \frac{P(k)}{\sqrt{(2\pi)^d \mid \Theta_k \mid}} \exp\left(-0.5(X-\mu_k)^T \Theta_k^{-1} (X-\mu_k)\right)$$

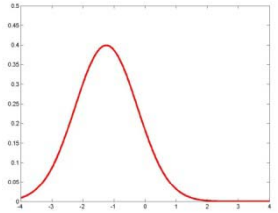
- Gaussian mixtures are often good models for the distribution of multivariate data
- Problem: Estimating the parameters, given a collection of data

Gaussian Mixtures: Generating model

$$P(X) = \sum_k P(k) N(X; \mu_k, \Theta_k)$$

6.1 1.4 5.3 1.9 4.2 2.2 4.9 0.5

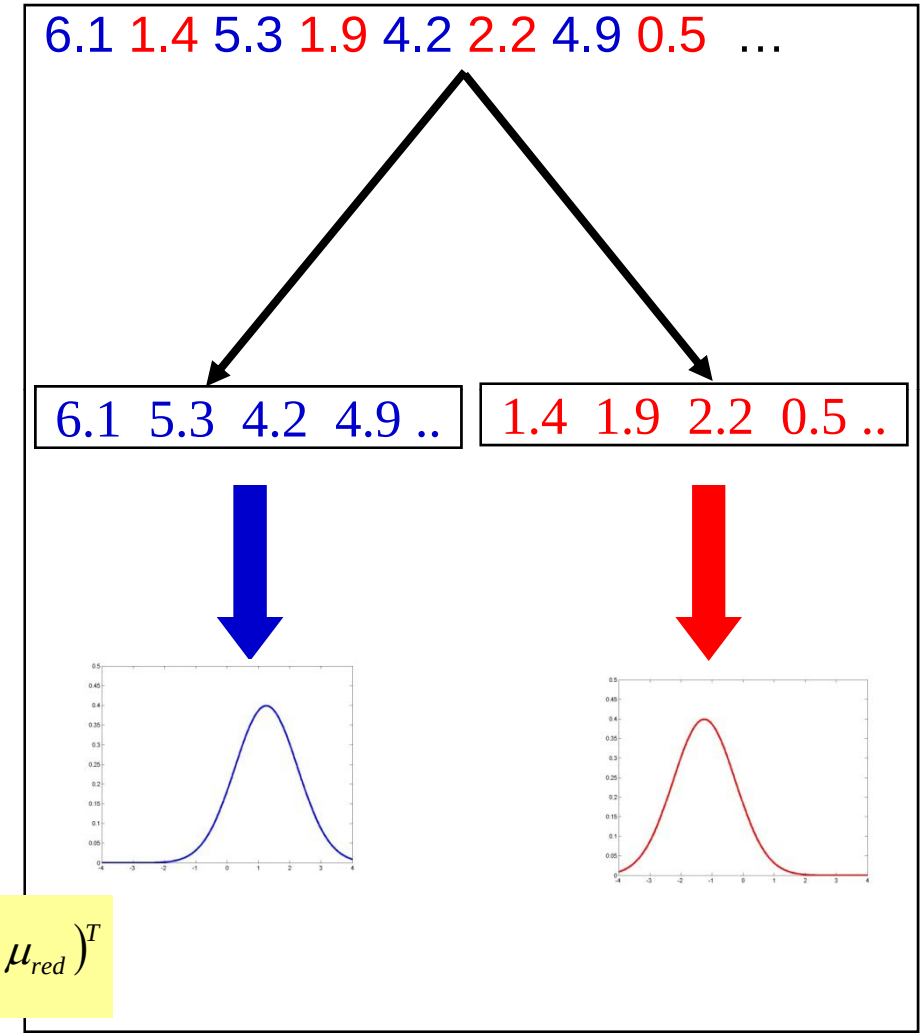




- The caller now has two Gaussians
  - At each draw he randomly selects a Gaussian, by the mixture weight distribution
  - He then draws an observation from that Gaussian
  - Much like the dice problem (only the outcomes are now real numbers and can be anything)

Estimating GMM with complete information

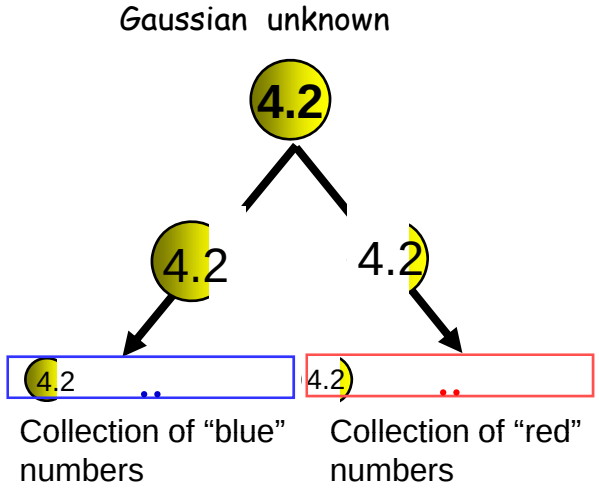
- Observation: A collection of numbers drawn from a mixture of 2 Gaussians
  - As indicated by the colors, we know which Gaussian generated what number
- Segregation: Separate the blue observations from the red
- From each set compute parameters for that Gaussian



$$\mu_{red} = \frac{1}{N_{red}} \sum_{i \in red} X_i \qquad \Theta_{red} = \frac{1}{N_{red}} \sum_{i \in red} (X_i - \mu_{red})(X_i - \mu_{red})^T$$

$$P(red) = \frac{N_{red}}{N}$$

Fragmenting the observation



- The identity of the Gaussian is not known!
- Solution: **Fragment the observation**
- Fragment size proportional to *a posteriori* probability

$$P(k \mid X) = \frac{P(X \mid k)P(k)}{\sum_{k'} P(k')P(X \mid k')} = \frac{P(k)N(X;\mu_k,\Theta_k)}{\sum_{k'} P(k')N(X;\mu_{k'},\Theta_{k'})}$$

Expectation **Maximization**

- Initialize P(k), μ_k and Θ_k for both Gaussians
  - Important how we do this
  - Typical solution: Initialize means randomly, Θ_k as the global covariance of the data and P(k) uniformly
- Compute fragment sizes for each Gaussian, for each observation

Number	P(red|X)	P(blue|X)
6.1	.81	.19
1.4	.33	.67
5.3	.75	.25
1.9	.41	.59
4.2	.64	.36
2.2	.43	.57
4.9	.66	.34
0.5	.05	.95

$$P(k \mid X) = \frac{P(k)N(X;\mu_k,\Theta_k)}{\sum_{k'} P(k')N(X;\mu_{k'},\Theta_{k'})}$$

Expectation **Maximization**

- ***Each observation contributes only as much as its fragment size to each statistic***
- Mean(red) =
  (6.1*0.81 + 1.4*0.33 + 5.3*0.75 + 1.9*0.41 + 4.2*0.64 + 2.2*0.43 + 4.9*0.66 + 0.5*0.05) /
  (0.81 + 0.33 + 0.75 + 0.41 + 0.64 + 0.43 + 0.66 + 0.05)
  = 17.05 / 4.08 = 4.18

Number	P(red|X)	P(blue|X)
6.1	.81	.19
1.4	.33	.67
5.3	.75	.25
1.9	.41	.59
4.2	.64	.36
2.2	.43	.57
4.9	.66	.34
0.5	.05	.95
	4.08	3.92

- Var(red) = ((6.1-4.18)^2*0.81 + (1.4-4.18)^2*0.33 +
  (5.3-4.18)^2*0.75 + (1.9-4.18)^2*0.41 +
  (4.2-4.18)^2*0.64 + (2.2-4.18)^2*0.43 +
  (4.9-4.18)^2*0.66 + (0.5-4.18)^2*0.05) /
  (0.81 + 0.33 + 0.75 + 0.41 + 0.64 + 0.43 + 0.66 + 0.05)

$$P(red) = \frac{4.08}{8}$$

EM for Gaussian Mixtures

1. Initialize P(k), μ_k and Θ_k for all Gaussians
2. For each observation X compute *a posteriori* probabilities for all Gaussian

$$P(k \mid X) = \frac{P(k)N(X;\mu_k,\Theta_k)}{\sum_{k'} P(k')N(X;\mu_{k'},\Theta_{k'})}$$

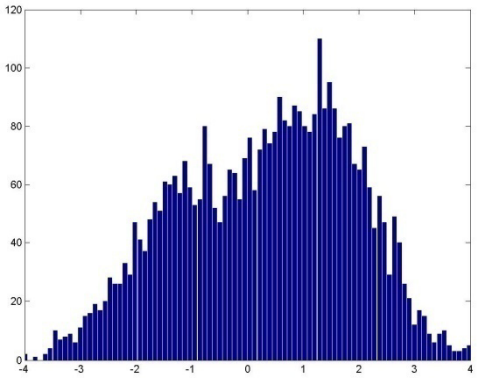
3. Update mixture weights, means and variances for all Gaussians

$$P(k) = \frac{\sum_X P(k|X)}{N} \qquad \mu_k = \frac{\sum_X P(k|X)\, X}{\sum_X P(k|X)} \qquad \Theta_k = \frac{\sum_X P(k|X)\,(X-\mu_k)^2}{\sum_X P(k|X)}$$

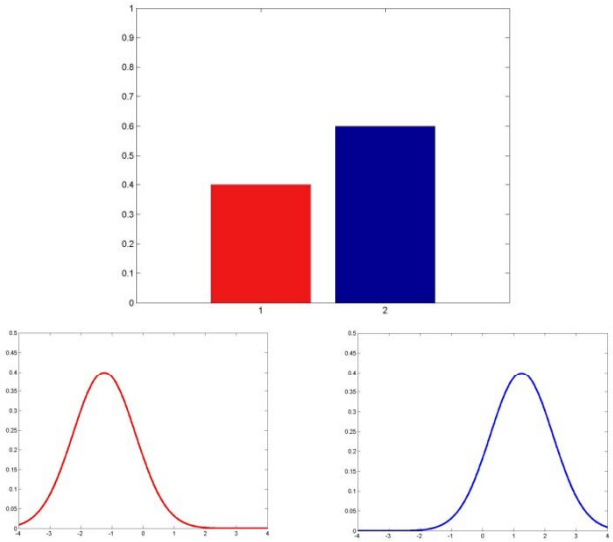
4. If not converged, return to 2

EM estimation of Gaussian Mixtures

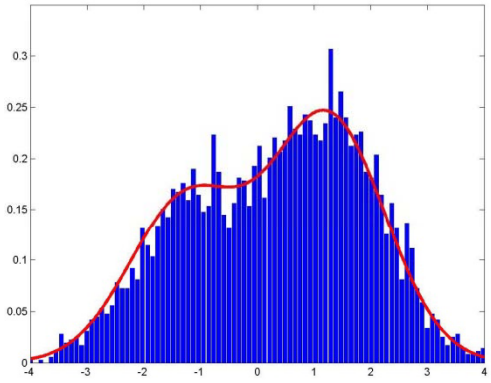
- An Example



Histogram of 4000 instances of a randomly generated data



Individual parameters of a two-Gaussian mixture estimated by EM



Two-Gaussian mixture estimated by EM

Expectation Maximization

- The same principle can be extended to mixtures of other distributions.

- E.g. Mixture of Laplacians: Laplacian parameters become

$$\mu_k = \frac{1}{\sum_x P(k \mid x)} \sum_x P(k \mid x) x \qquad b_k = \frac{1}{\sum_x P(k \mid x)} \sum_x P(k \mid x) \mid x - \mu_k \mid$$

- In a mixture of Gaussians and Laplacians, Gaussians use the Gaussian update rules, Laplacians use the Laplacian rule

Expectation Maximization

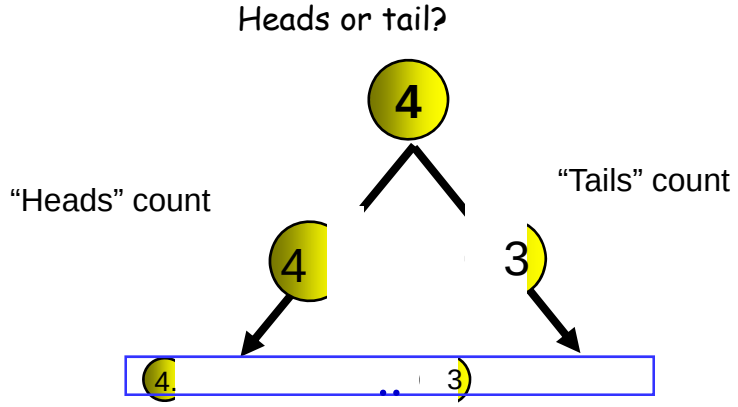
- The EM algorithm is used whenever proper statistical analysis of a phenomenon requires the knowledge of a hidden or missing variable (or a set of hidden/missing variables)
  - The hidden variable is often called a “latent” variable

- Some examples:
  - Estimating mixtures of distributions
    - Only data are observed. The individual distributions and mixing proportions must both be learnt.
  - Estimating the distribution of data, when some attributes are missing
  - Estimating the dynamics of a system, based only on observations that may be a complex function of system state

Solve this problem:

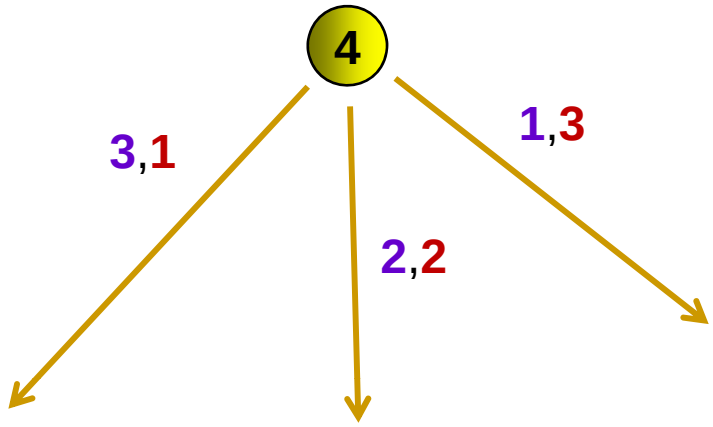
- Caller rolls a dice and flips a coin
  - He calls out the number rolled if the coin shows head
  - Otherwise he calls the number+1
  - Determine p(heads) and p(number) for the dice from a collection of ouputs
- Caller rolls two dice
  - He calls out the sum
  - Determine P(dice) from a collection of ouputs

The dice and the coin



- Unknown: Whether it was head or tails

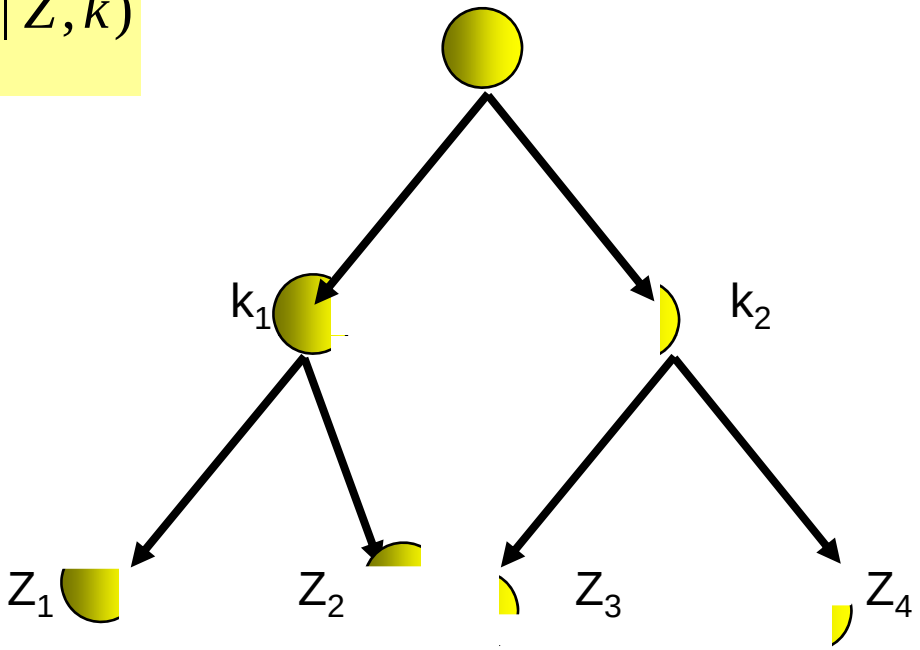
The two dice



- Unknown: How to partition the number
- $Count_{blue}(3)$ += P(3,1 | 4)
- $Count_{blue}(2)$ += P(2,2 | 4)
- $Count_{blue}(1)$ += P(1,3 | 4)

Fragmentation can be hierarchical

$$P(X) = \sum_k P(k) \sum_Z P(Z \mid k) P(X \mid Z, k)$$

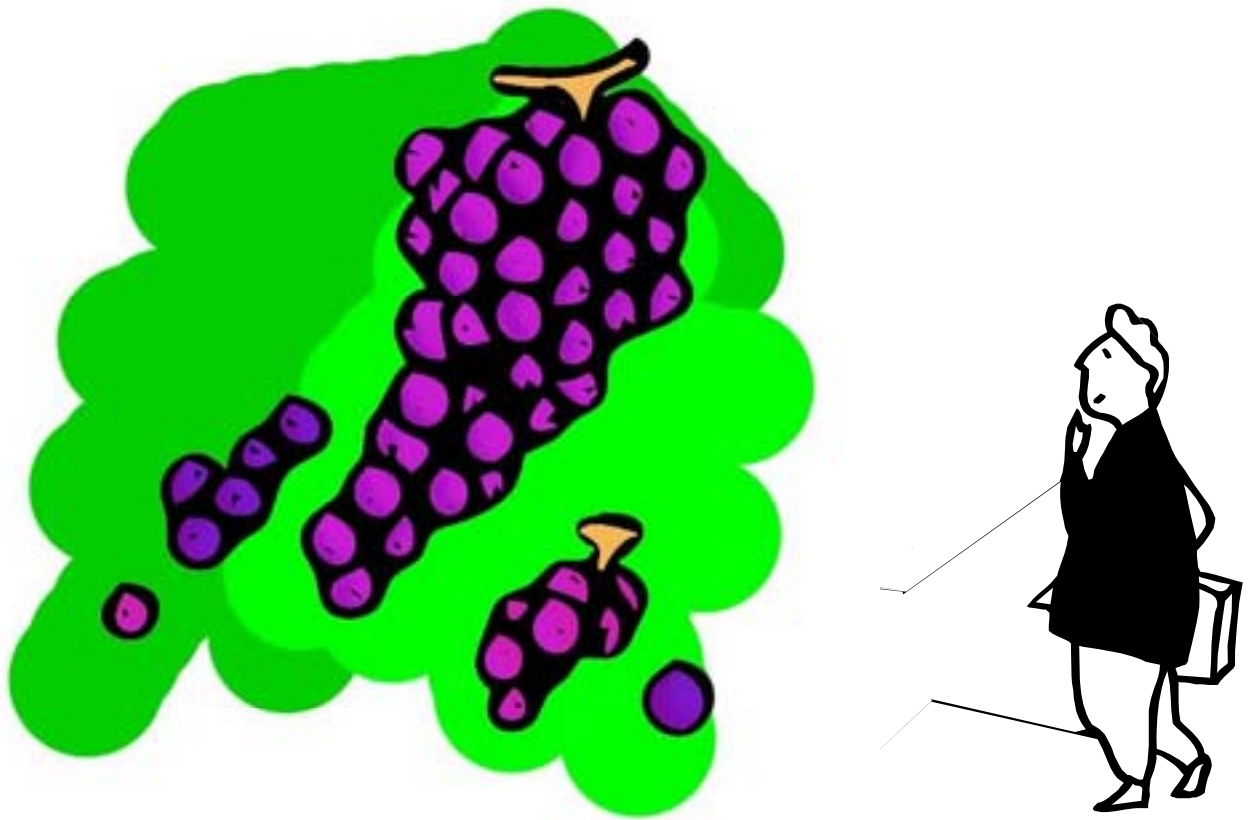


- E.g. mixture of mixtures
- Fragments are further fragmented..
  - Work this out

More later

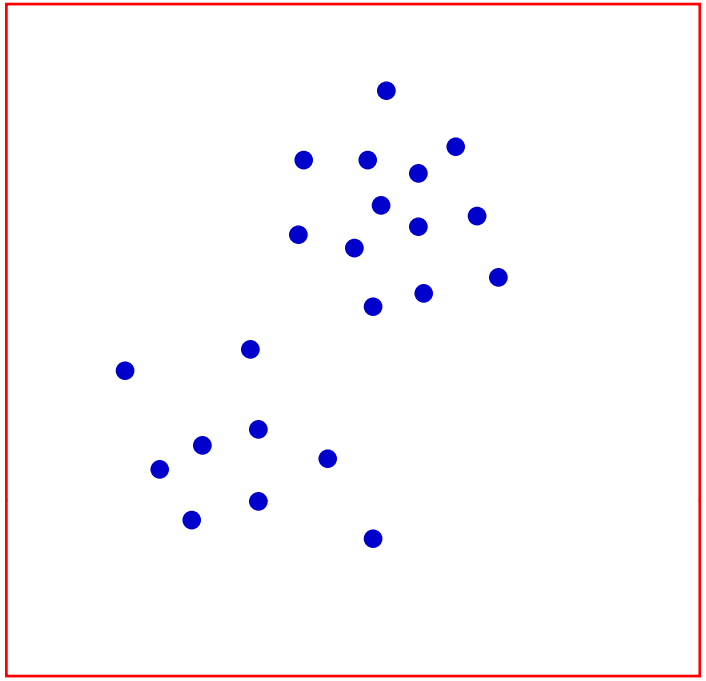
- Will see a couple of other instances of the use of EM
  - E.g. HMM training
  - Homework problems

Clustering



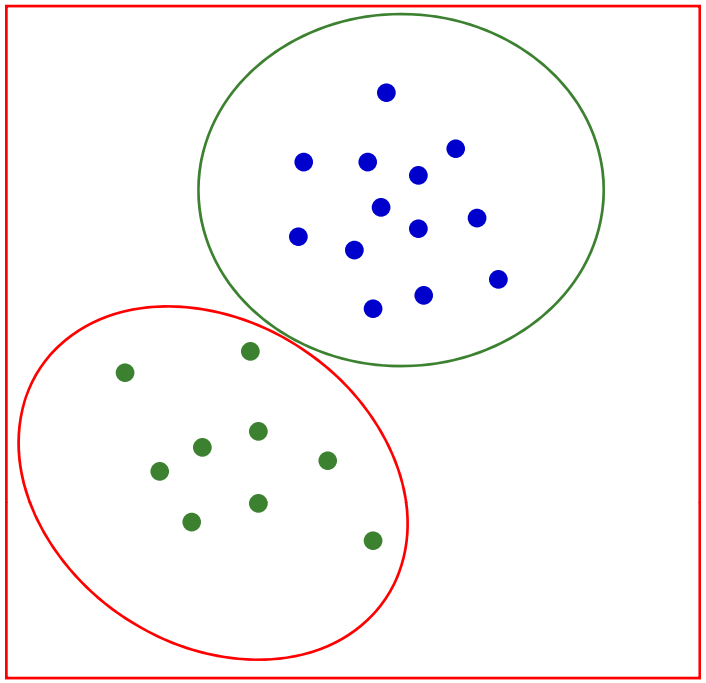
Clustering

- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)



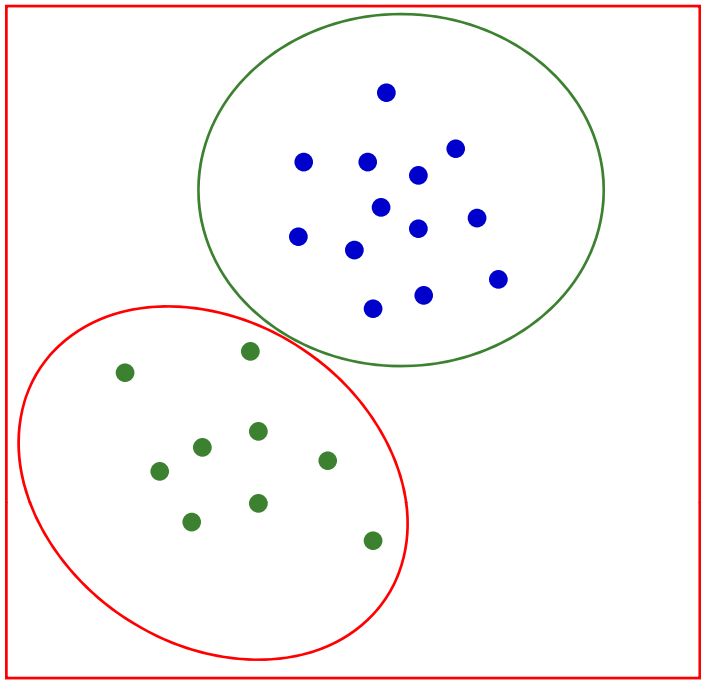
Clustering

- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)



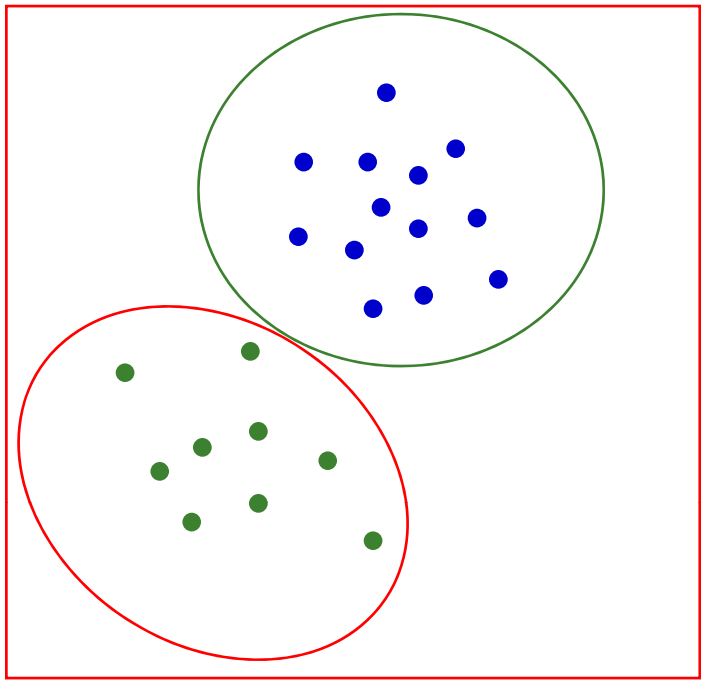
Clustering

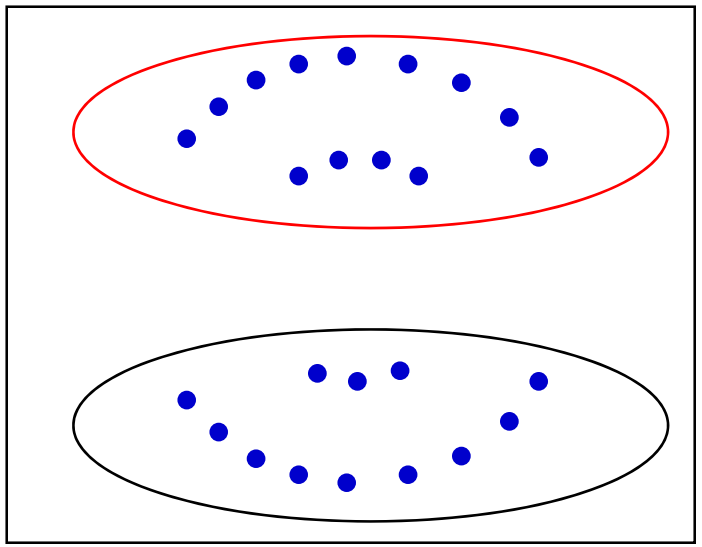
- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)
- How is it done
  - Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind



Clustering

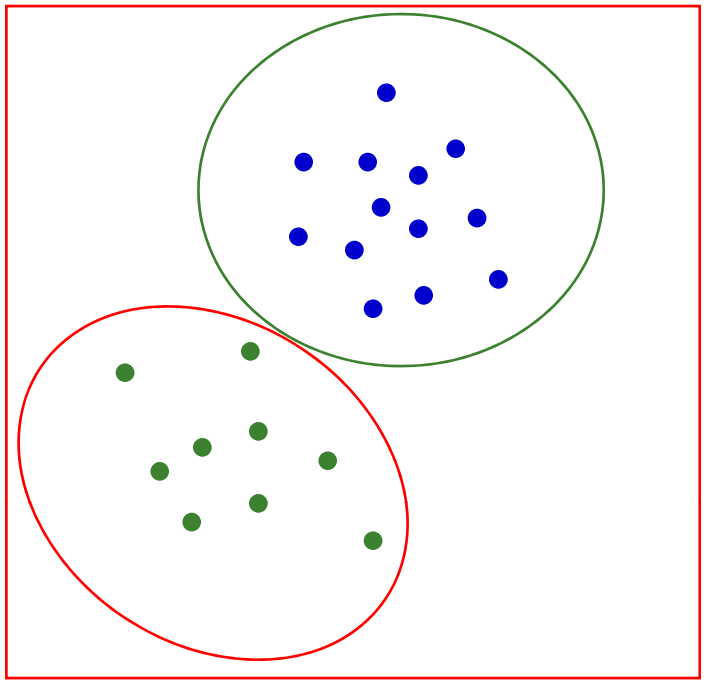
- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)
- How is it done
  - Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind
  - The objective function used affects the nature of the discovered clusters
    - E.g. Euclidean distance and distance from center result in different clusters in this example

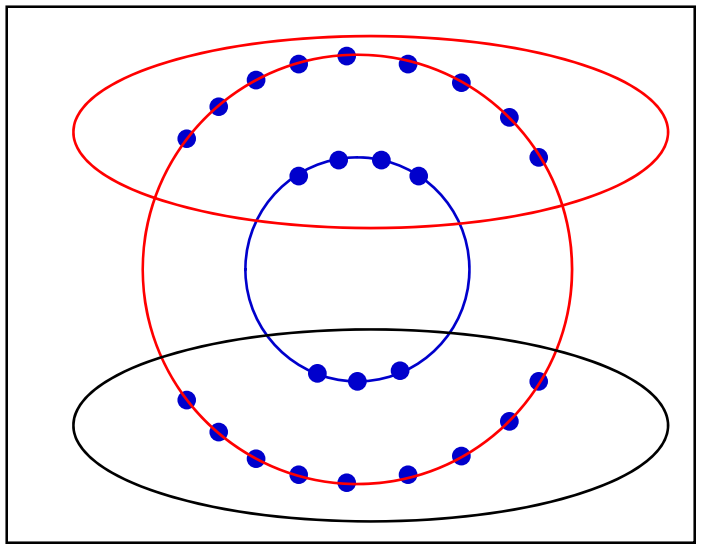




Clustering

- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)
- How is it done
  - Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind
  - The objective function used affects the nature of the discovered clusters
    - E.g. Euclidean distance and distance from center result in different clusters in this example





Why Clustering

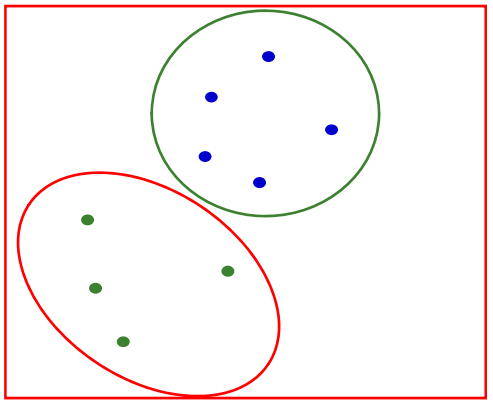
- Automatic grouping into “Classes”
  - Different clusters may show different behavior
- Quantization
  - All data within a cluster are represented by a single point
- Preprocessing step for other algorithms
  - Indexing, categorization, etc.

Clustering criteria

- Compactness criterion
  - Measure that shows how “good” clusters are
    - The objective function
- Distance of a point from a cluster
  - To determine the cluster a data vector belongs to

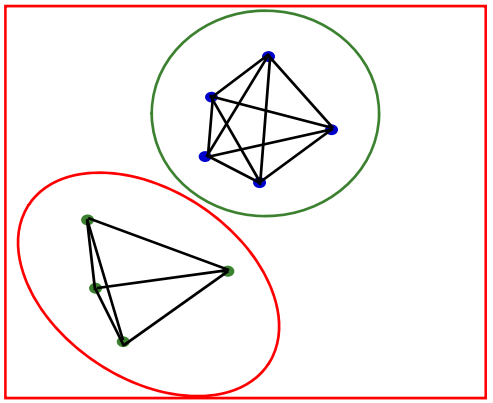
"Compactness" criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster



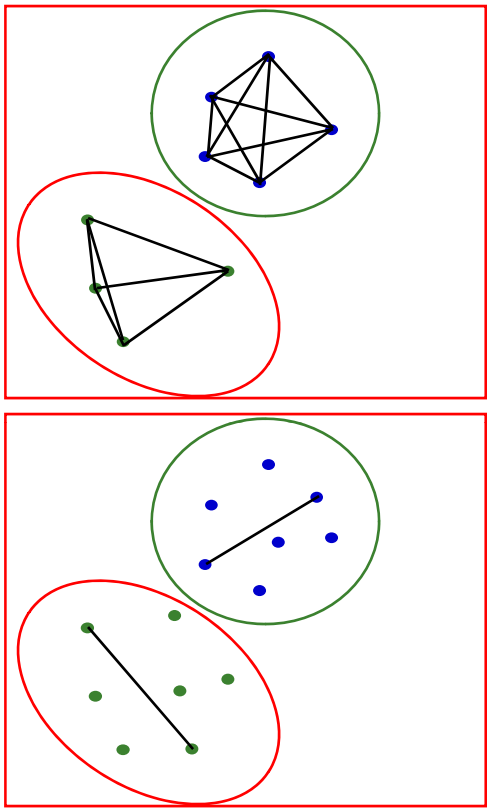
"Compactness" criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster



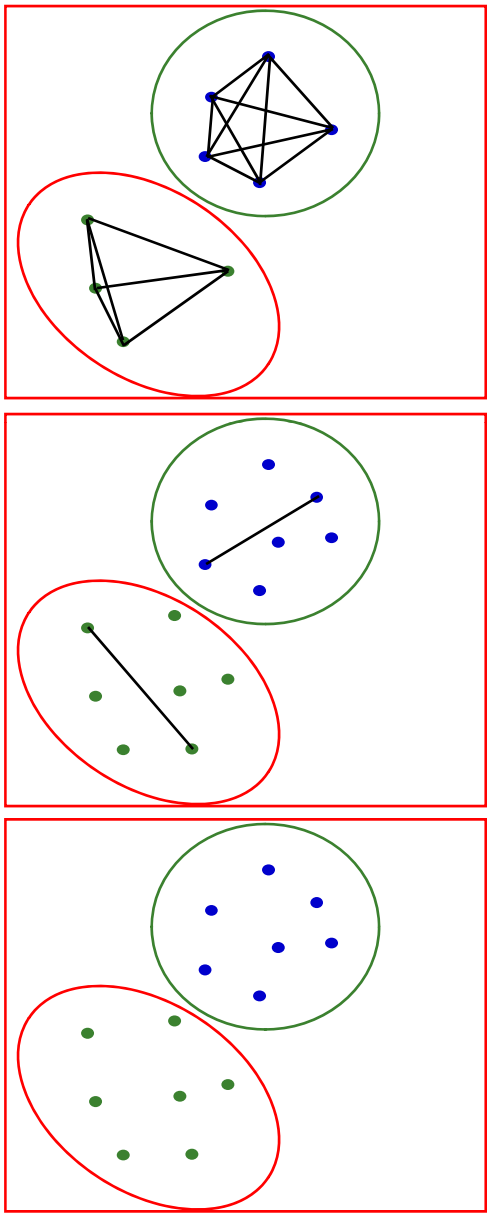
"Compactness" criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster



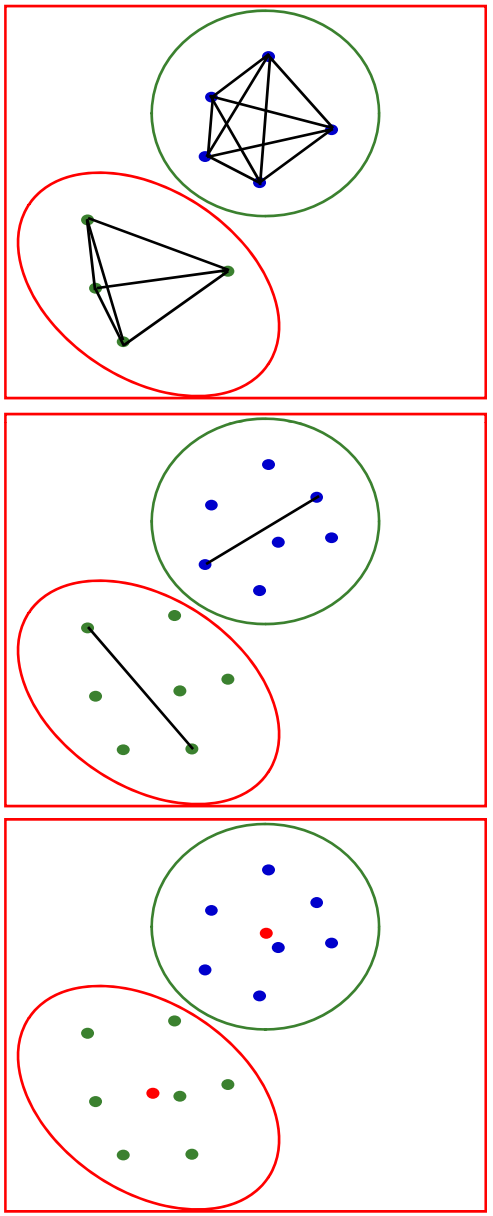
"Compactness" criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster
  - Total distance of every element in the cluster from the centroid of the cluster



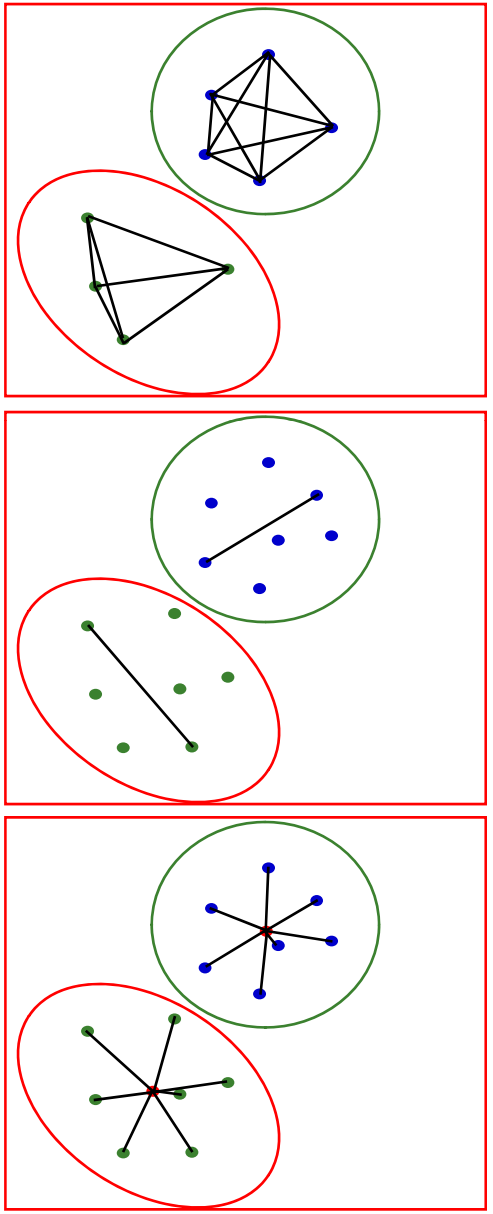
"Compactness" criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster
  - Total distance of every element in the cluster from the centroid of the cluster



"Compactness" criteria for clustering

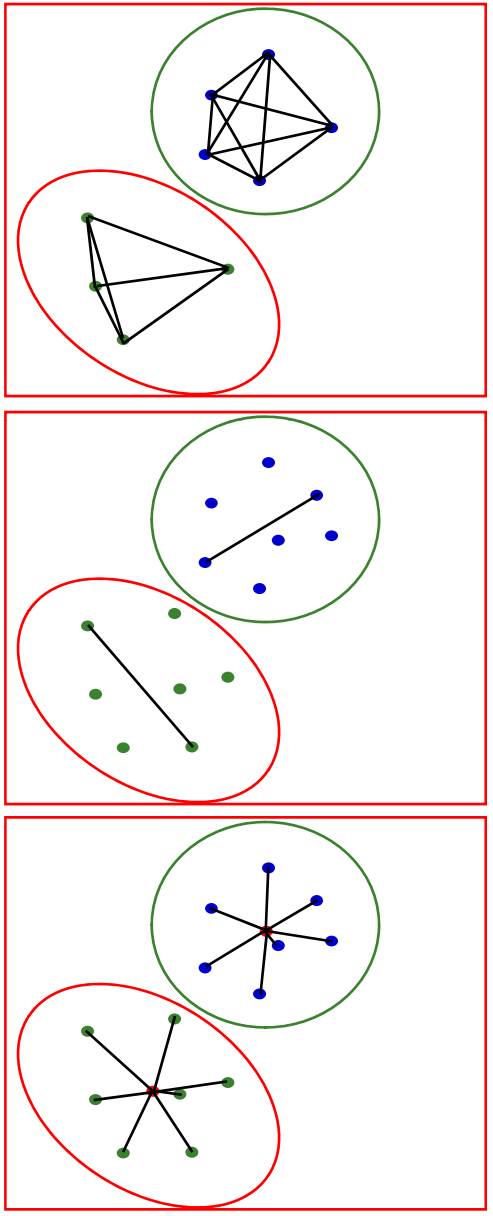
- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster
  - Total distance of every element in the cluster from the centroid of the cluster



"Compactness" criteria for clustering

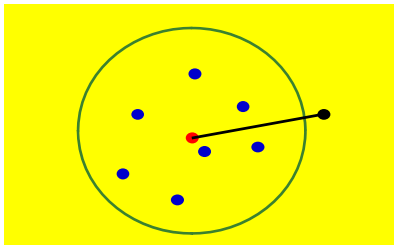
- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster
  - Total distance of every element in the cluster from the centroid of the cluster
  - Distance measures are often weighted Minkowski metrics

$$dist = \sqrt[n]{w_1|a_1 - b_1|^n + w_2|a_2 - b_2|^n + ... + w_M|a_M - b_M|^n}$$



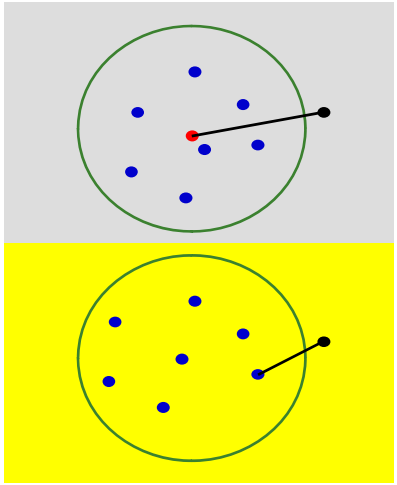
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster



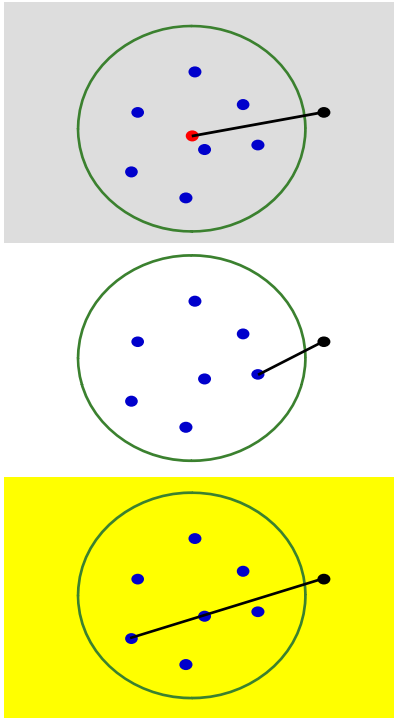
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
  - Distance from the closest point in the cluster



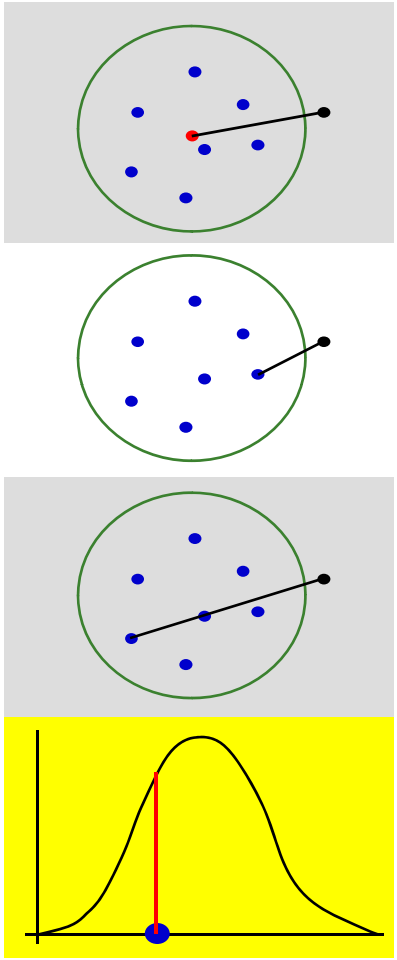
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster



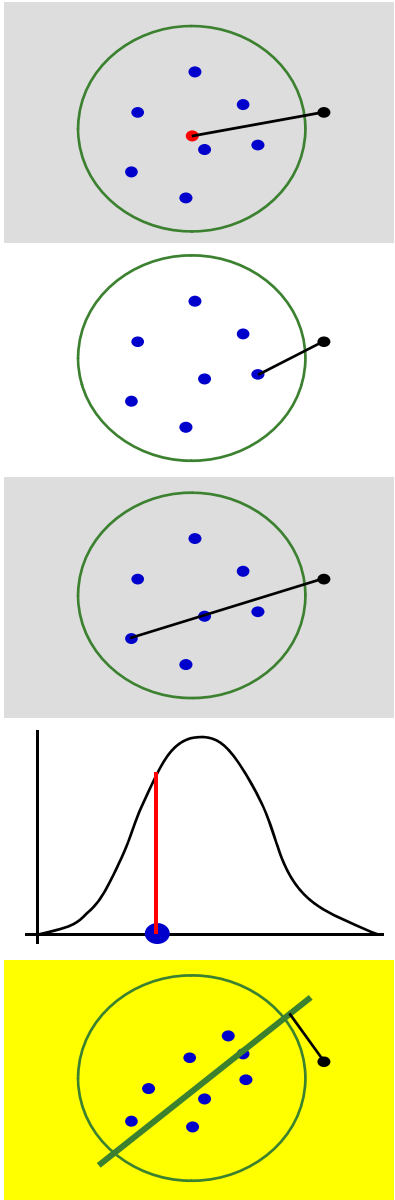
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster
  - Probability of data measured on cluster distribution



Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster
  - Probability of data measured on cluster distribution
  - Fit of data to cluster-based regression



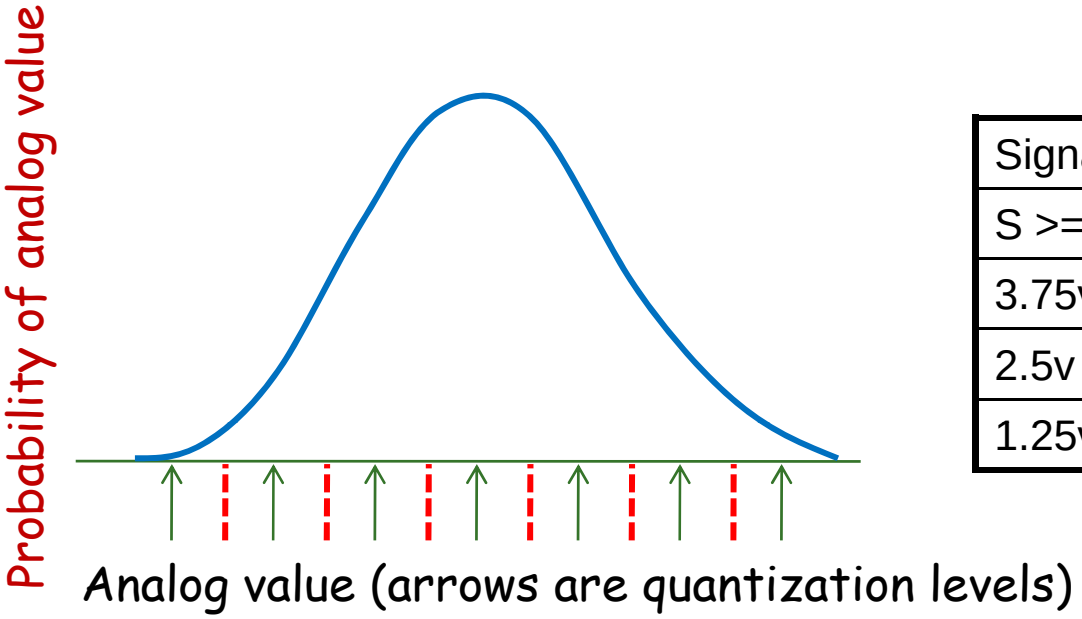
Optimal clustering: Exhaustive enumeration

- All possible combinations of data must be evaluated
  - If there are M data points, and we desire N clusters, the number of ways of separating M instances into N clusters is

$$\frac{1}{M!}\sum_{i=0}^{N}(-1)^{i}\binom{N}{i}(N-i)^{M}$$

  - Exhaustive enumeration based clustering requires that the objective function (the “Goodness measure”) be evaluated for every one of these, and the best one chosen
- This is the only correct way of optimal clustering
  - Unfortunately, it is also computationally unrealistic

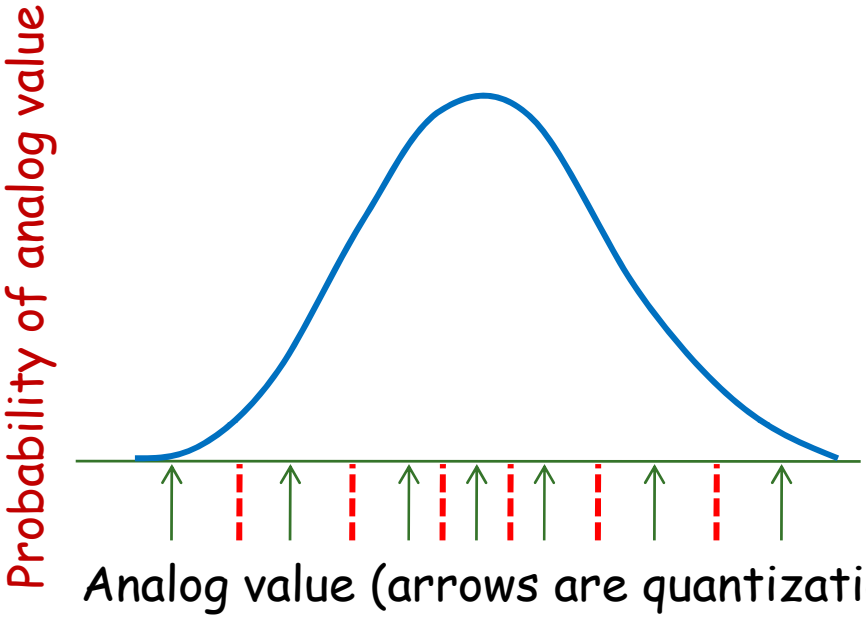
Not-quite non sequitir: Quantization



Signal Value	Bits	Mapped to
S >= 3.75v	11	3 * const
3.75v > S >= 2.5v	10	2 * const
2.5v > S >= 1.25v	01	1 * const
1.25v > S >= 0v	0	0

- Linear quantization (uniform quantization):
  - Each digital value represents an equally wide range of analog values
  - Regardless of distribution of data
  - Digital-to-analog conversion represented by a “uniform” table

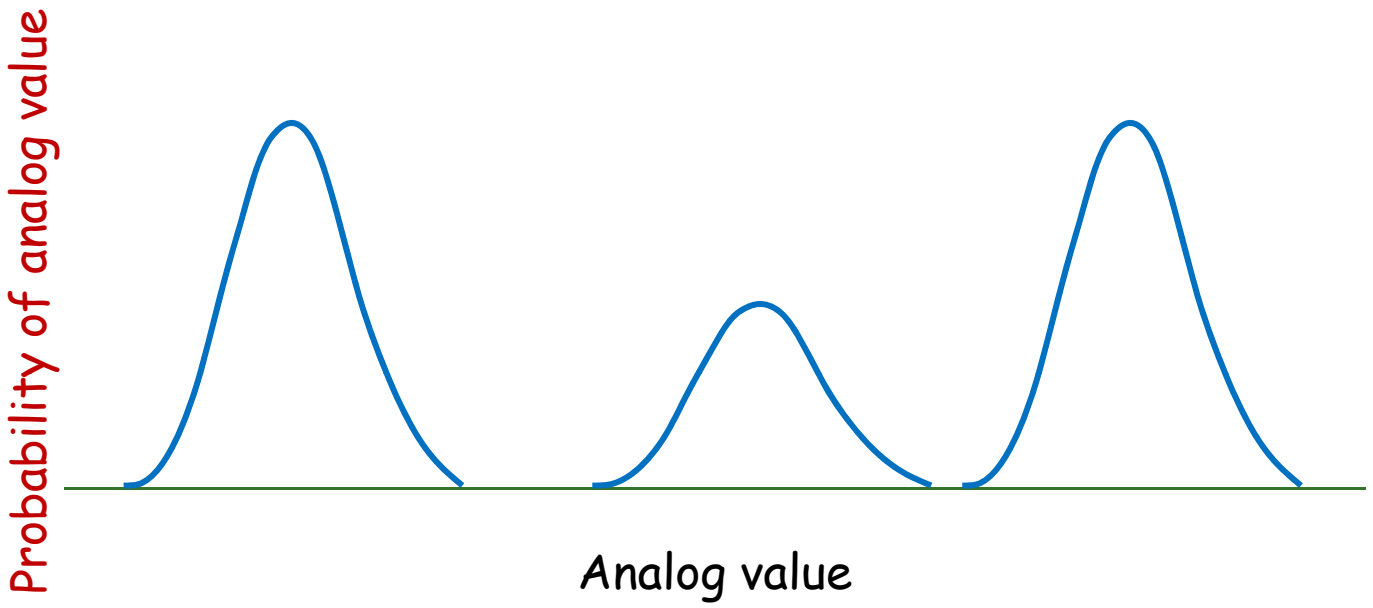
Not-quite non sequitir: Quantization



Signal Value	Bits	Mapped to
S >= 4v	11	4.5
4v > S >= 2.5v	10	3.25
2.5v > S >= 1v	01	1.25
1.0v > S >= 0v	0	0.5

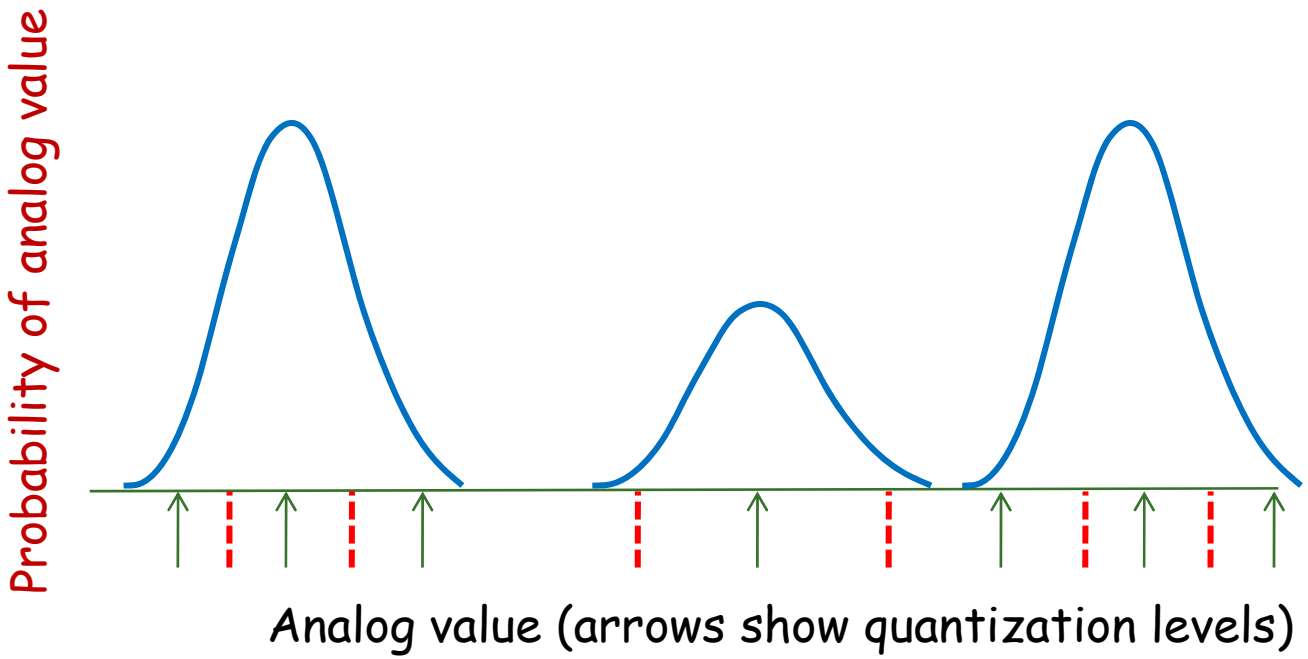
- Non-Linear quantization:
  - Each digital value represents a different range of analog values
    - Finer resolution in high-density areas
    - Mu-law / A-law assumes a gaussian-like distribution of data
  - Digital-to-analog conversion represented by a “non-uniform” table

Non-uniform quantization



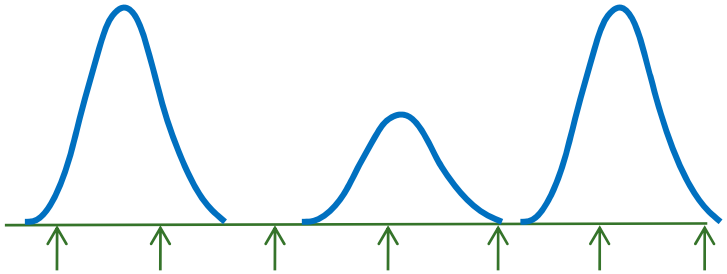
- If data distribution is not Gaussianish?
  - Mu-law / A-law are not optimal
  - How to compute the optimal ranges for quantization
    - Or the optimal table

The Lloyd Quantizer



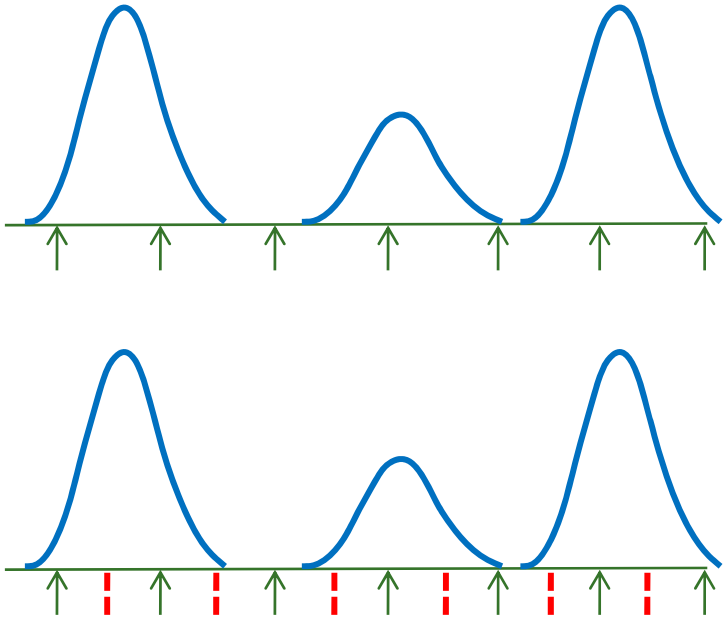
- Lloyd quantizer: An iterative algorithm for computing optimal quantization tables for non-uniformly distributed data
- **Learned from "training" data**

Lloyd Quantizer



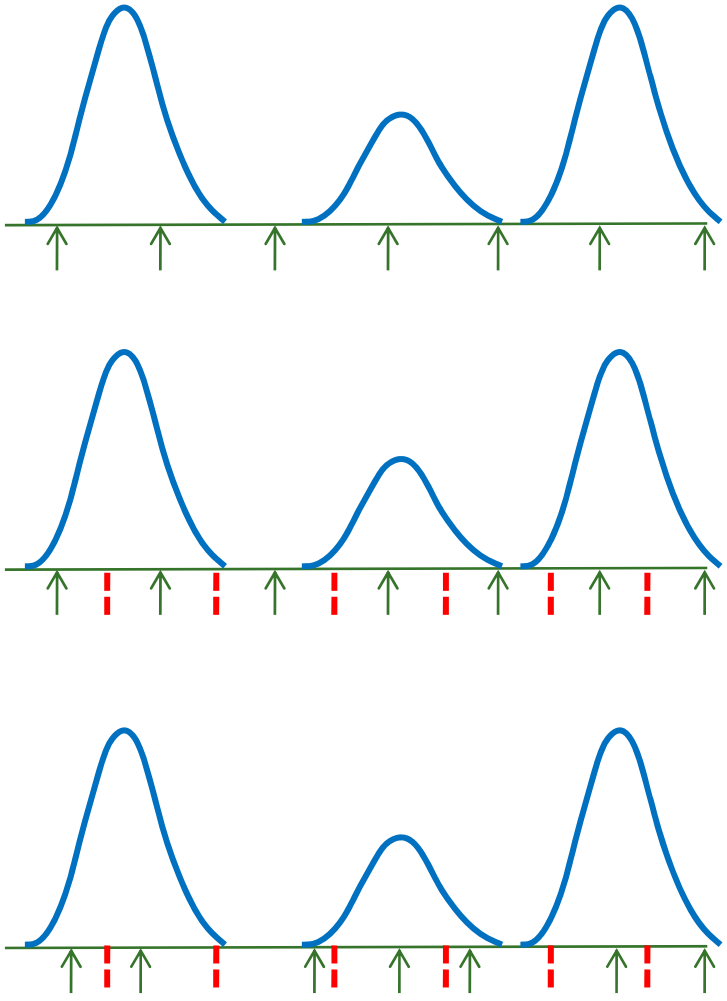
- Randomly initialize quantization points
  - Right column entries of quantization table

Lloyd Quantizer



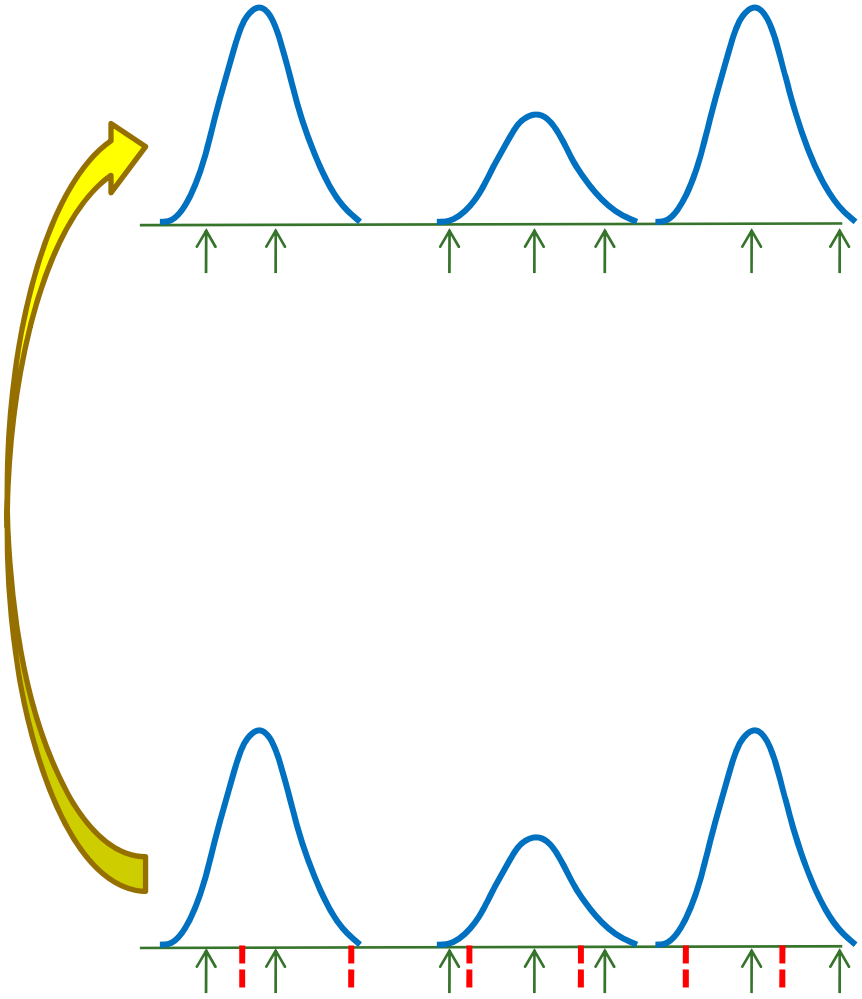
- Randomly initialize quantization points
  - Right column entries of quantization table
- Assign all training points to the nearest quantization point
  - Draw boundaries

Lloyd Quantizer



- Randomly initialize quantization points
  - Right column entries of quantization table
- Assign all training points to the nearest quantization point
  - Draw boundaries
- Reestimate quantization points

Lloyd Quantizer



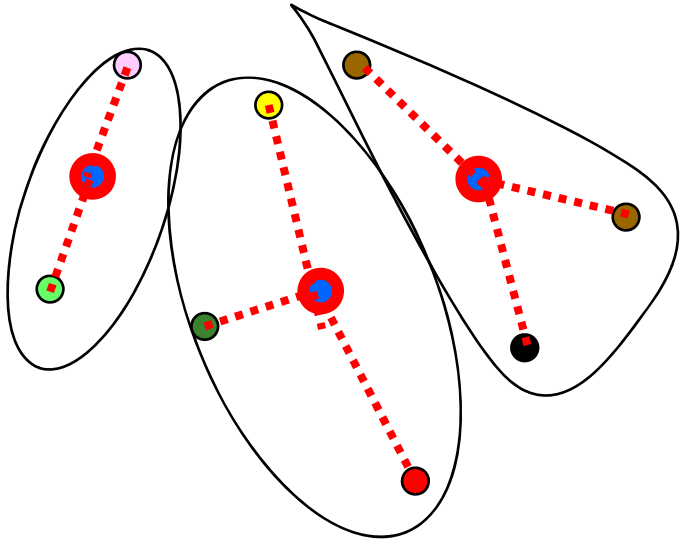
- Randomly initialize quantization points
  - Right column entries of quantization table
- Assign all training points to the nearest quantization point
  - Draw boundaries
- Reestimate quantization points
- Iterate until convergence

Generalized Lloyd Algorithm: K–means clustering

- K means is an iterative algorithm for clustering ***vector*** data
  - McQueen, J. 1967. “Some methods for classification and analysis of multivariate observations.” Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 281-297
- General procedure:
  - Initially group data into the required number of clusters somehow (initialization)
  - Assign each data point to the closest cluster
  - Once all data points are assigned to clusters, redefine clusters
  - Iterate

K–means

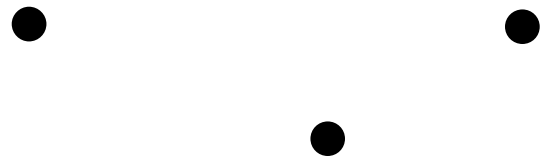
- Problem: Given a set of data vectors, find natural clusters
- Clustering criterion is **scatter**: distance from the centroid
  - Every cluster has a centroid
  - The centroid represents the cluster
- **Definition**: The **centroid** is the weighted mean of the cluster
  - Weight = 1 for basic scheme



$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

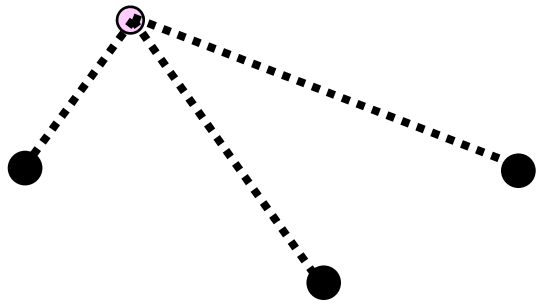
K–means

1. Initialize a set of centroids randomly



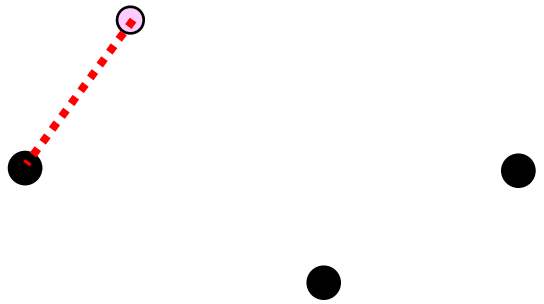
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$



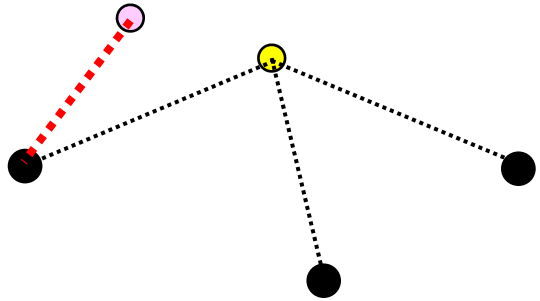
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



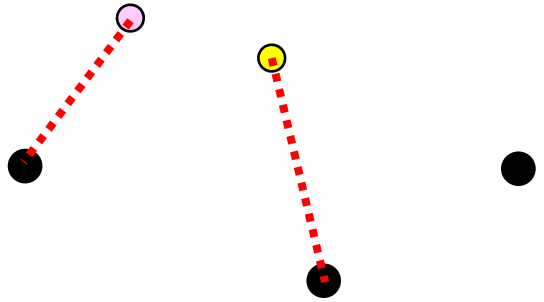
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



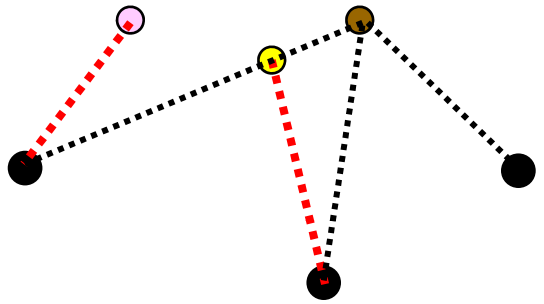
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



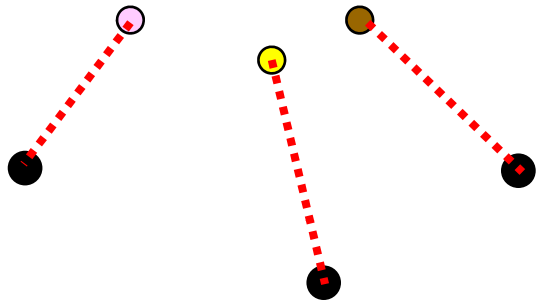
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



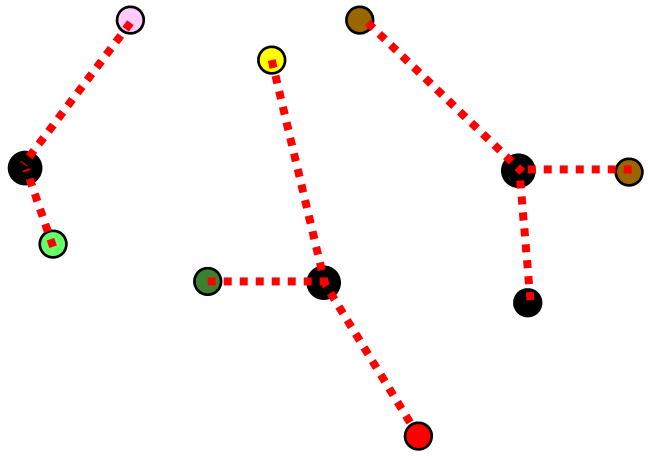
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



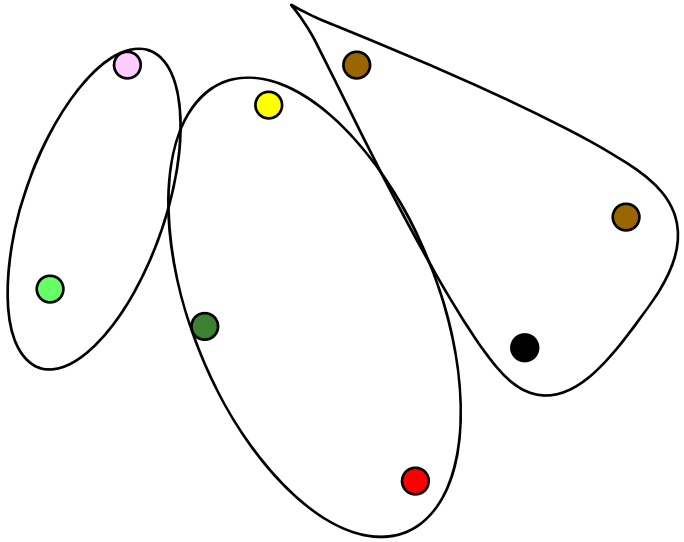
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



K–means

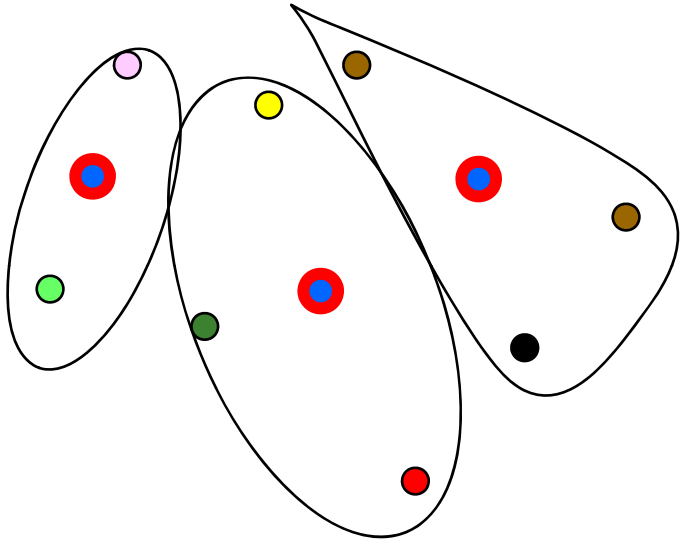
1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

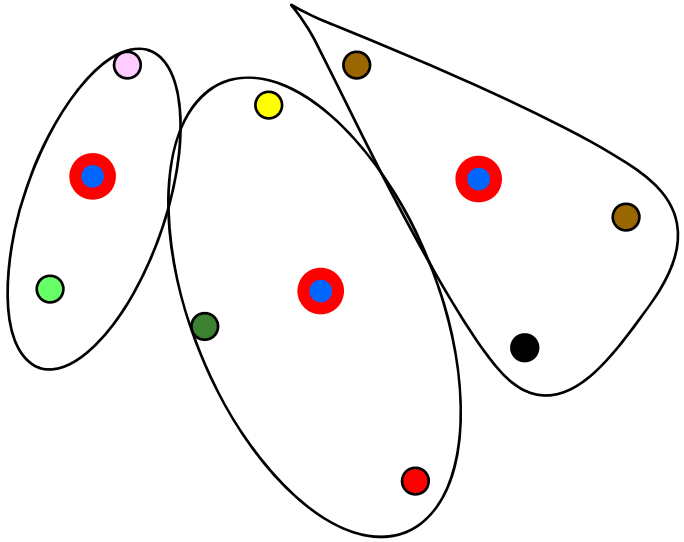


K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

5. If not converged, go back to 2



K-Means comments

- The distance metric determines the clusters
  - In the original formulation, the distance is L2 distance
    - Euclidean norm, $w_i = 1$

$$\textbf{distance}_{cluster}(x, m_{cluster}) = \| x - m_{cluster} \|_2$$

$$m_{cluster} = \frac{1}{N_{cluster}} \sum_{i \in cluster} x_i$$

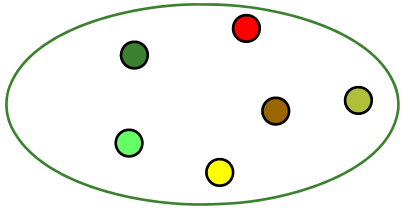
  - If we replace every x by $m_{\text{cluster}}(x)$, we get *Vector Quantization*
- K-means is an instance of *generalized* EM
- Not guaranteed to converge for all distance metrics

Initialization

- Random initialization
- Top-down clustering
  - Initially partition the data into two (or a small number of) clusters using K means
  - Partition each of the resulting clusters into two (or a small number of) clusters, also using K means
  - Terminate when the desired number of clusters is obtained

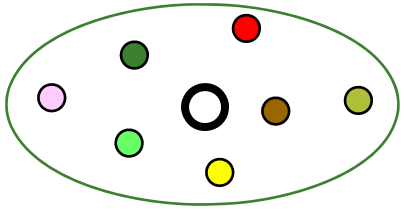
K-Means for Top–Down clustering

1. Start with one cluster



K-Means for Top–Down clustering

1. Start with one cluster

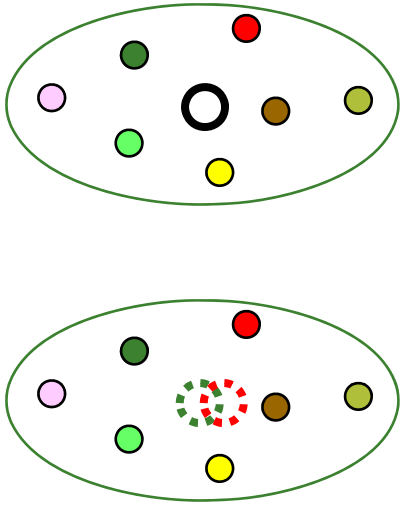


2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

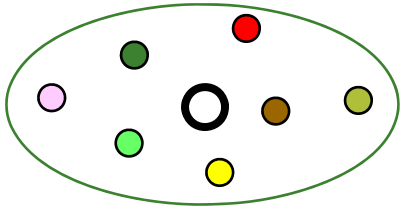


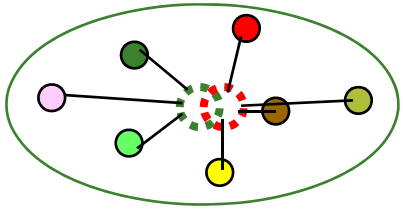
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

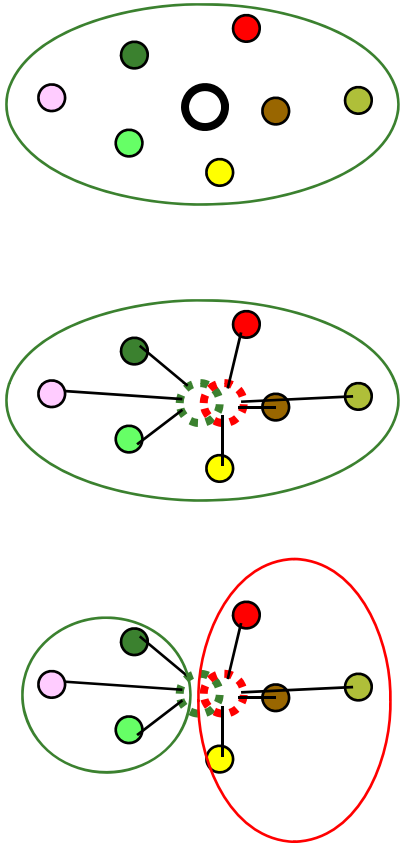
3. Initialize K means with new set of centroids





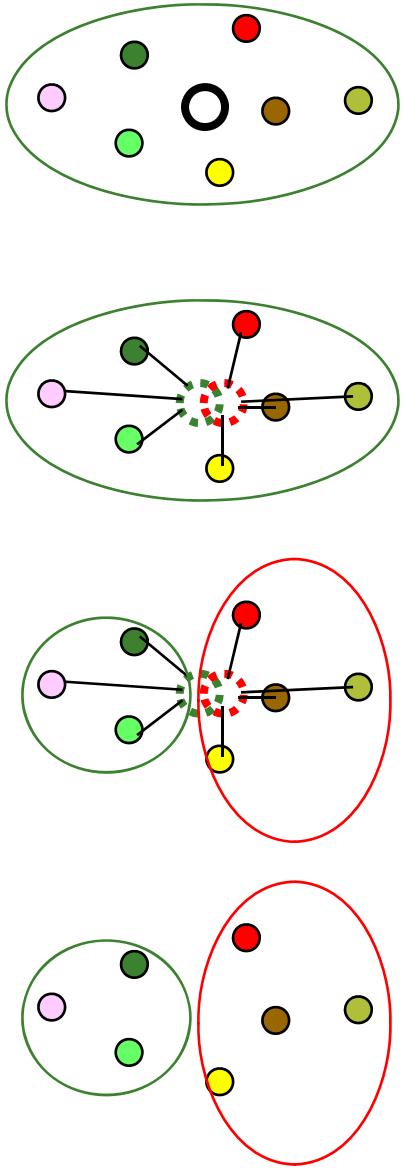
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence



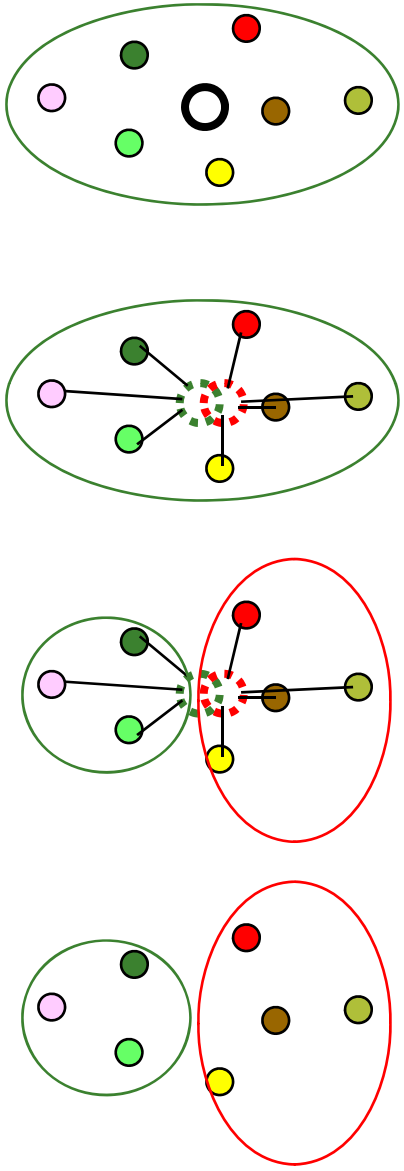
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence

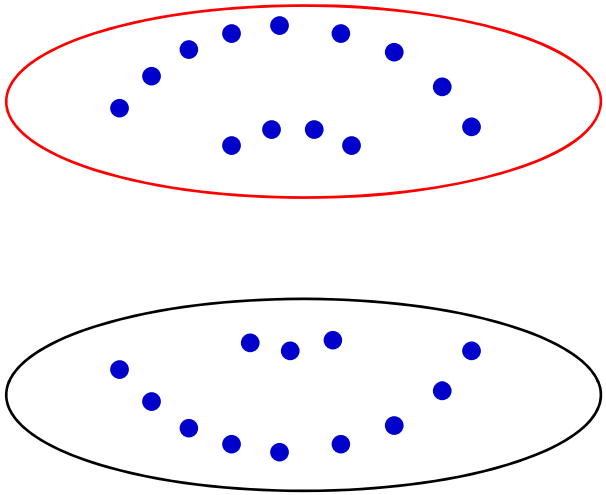


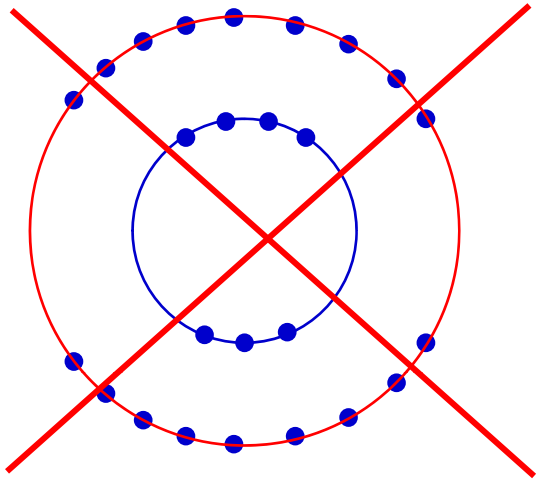
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence
5. If the desired number of clusters is not obtained, return to 2



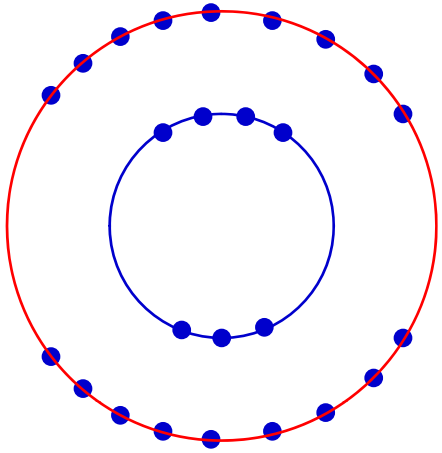
K-means, distances, kernels and spectra



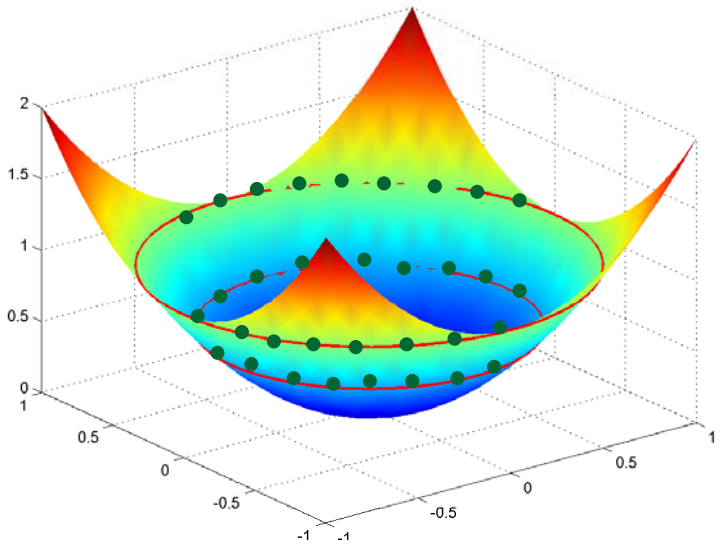


- Basic K-means results in good clusters in Euclidean spaces
  - Alternately stated, will only find clusters that are "good" in terms of Euclidean distances
- Will not find other types of clusters

Non-euclidean clusters



f([x,y]) -> [x,y,z]
x = x
y = y
$z = x^2 + y^2$



- For other forms of clusters we must modify the distance measure
  - E.g. distance from a circle
- May be viewed as a distance in a higher dimensional space
  - I.e *Kernel* distances
  - *Kernel* K-means
- Other related clustering mechansims:
  - Spectral clustering
    - Non-linear weighting of adjacency
  - Normalized cuts..