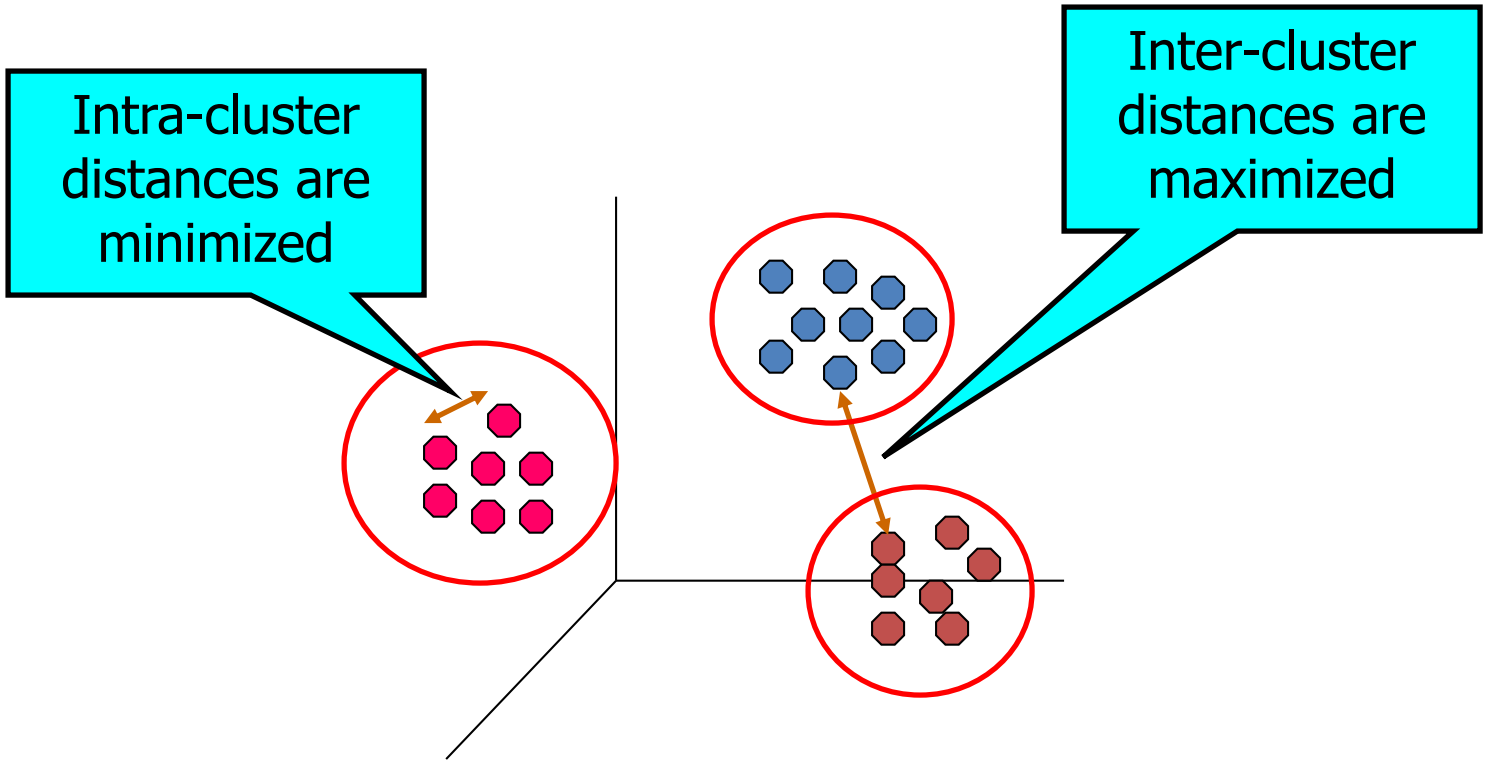
FUZZY CLUSTERING

Lecture 05.04

What is a Clustering?

- In general a grouping of objects such that the objects in a group (cluster) are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Clustering Algorithms

- K-means and its variants
- Hierarchical clustering
- DBSCAN

- All these algorithms assign each data point to a single cluster: **crisp** clustering

Fuzzy clustering

- Uses concepts from the field of ***fuzzy logic*** and ***fuzzy set theory***.

- Objects are allowed to belong to more than one cluster.

- Each object belongs to every cluster with some weight.

Why fuzzy clustering

- When clusters are well separated, a crisp classification of objects into clusters makes sense.

- But in many cases, clusters are not well separated.

  - In a crisp formulation, a borderline object ends up being assigned to a cluster in an arbitrary manner.

Based on the theory of fuzzy sets

- Introduced by Lotfi Zadeh in 1965 as a way of dealing with imprecision and uncertainty.
- Fuzzy set theory allows an object to belong to a set with a degree of membership between 0 and 1.
- Traditional set theory can be seen as a special case that restrict membership values to be either 0 or 1.

Fuzzy clusters: definition

- Assume a set of n objects $X = \{x_1, x_2, \ldots, x_n\}$, where x_i is a d -dimensional point.

- A fuzzy clustering is a collection of k clusters, C_1, $C_2, \ldots, C_k$, and a partition matrix $W = w_{i,j} \in [0, 1]$, for $i = 1 \ldots n$ and $j = 1 \ldots k$, where each element $w_{i,j}$ is a weight that represents the degree of membership of object i in cluster C_j .

To have a fuzzy pseudo-partition:

- All weights for a given point, x_i , must add up to 1.

$$\sum_{j=1}^{k} w_{i,j} = 1$$

- Each cluster C_j contains, with non-zero weight, at least one point, but does not contain, with a weight of one, all the points.

$$0 < \sum_{i=1}^{n} w_{i,j} < n$$

Fuzzy c-means (FCM) is a fuzzy version of k-means

Fuzzy c-means algorithm

1. Select an initial fuzzy pseudo-partition, i.e., assign random values to all $w_{i,j}$

2. Repeat:
   - compute the centroid of each cluster using the fuzzy partition
   - update the fuzzy partition, i.e, the $w_{i,j}$ – based on the distance from centroid

Until the centroids don't change

There's alternative stopping criteria: “change in the error is below a specified threshold", or “absolute change in any $w_{i,j}$ is below a given threshold".

FCM as an optimization problem

- As with k-means, FCM also attempts to minimize the sum of the squared error (SSE).
- In k-means:

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_i} dist(c_j, x)^2$$

- In FCM

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{n} w_{i,j}^{M} dist(x_i, c_j)^2$$

- *M* is a “fuzziness” parameter that prevents points to completely belong to one cluster

Computing centroids

- For a cluster C_j , the corresponding centroid c_j is defined as:

$$c_j = \frac{\sum_{i=1}^{n} w_{ij}^{M} x_i}{\sum_{i=1}^{n} w_{ij}^{M}}$$

- This is just an extension of the definition of centroid that we have for k-means.

$$c_j = \frac{\sum_{i=1}^{n_j} x_{ij}}{n_j}$$

- The difference is that all *n* points are considered and the contribution of each point to the centroid is weighted by its membership degree.

Formula for updating weights

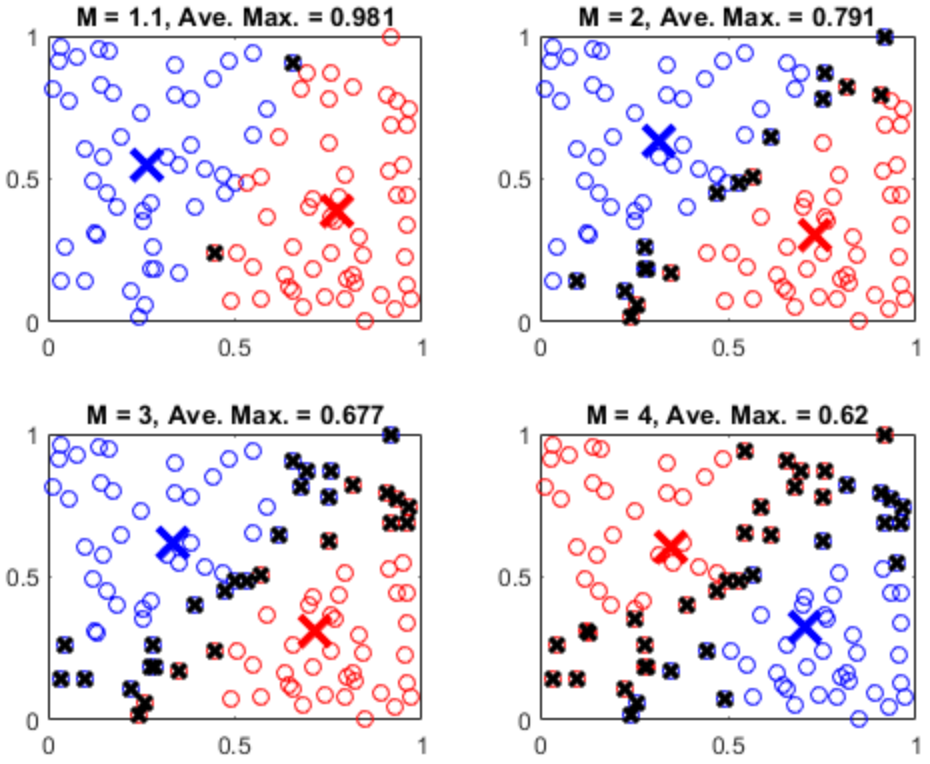
- Formula can be obtained by minimizing the SSE subject to the constraint that the weights sum to 1:

$$w_{ij} = \frac{(1/dist(x_i, c_j)^2)^{\frac{1}{M-1}}}{\sum_{q=1}^{k}(1/dist(x_i, c_q)^2)^{\frac{1}{M-1}}}$$

- Intuition: w_{ij} should be high if x_i is close to the centroid c_j, i.e., if $dist(x_i, c_j)$ is low.
- Denominator (sum of all weights for a given point) is needed to normalize weights for a point – so they sum up to 1.

Fuzziness: average maximum membership

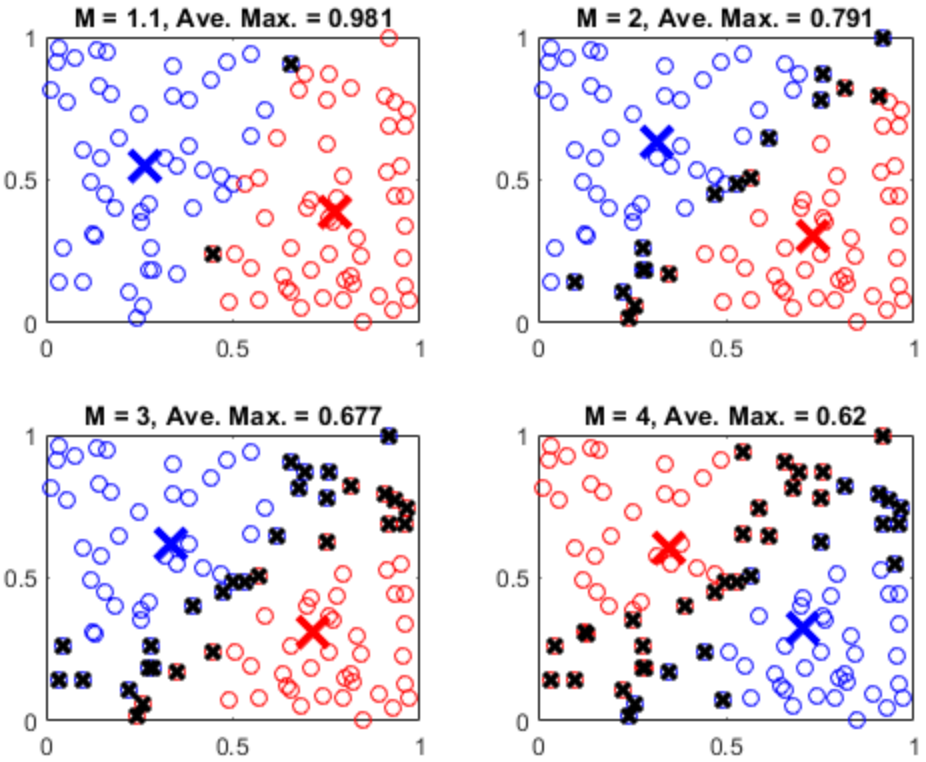
- A given data point is classified into the cluster for which it has the highest membership value.
- A maximum membership value of 0.5 indicates that the point belongs to both clusters equally.
- The data points marked with a black x have maximum membership values below 0.6. These points have a greater degree of uncertainty in their cluster membership.
- The average maximum membership value, averageMax, provides a quantitative description of the overlap.
- An averageMax value of 1 indicates crisp clusters, with smaller values indicating more overlap.



Fuzziness parameter *M*

$$w_{ij} = \frac{(1/dist(x_i, c_j)^2)^{\frac{1}{M-1}}}{\sum_{q=1}^{k}(1/dist(x_i, c_q)^2)^{\frac{1}{M-1}}}$$

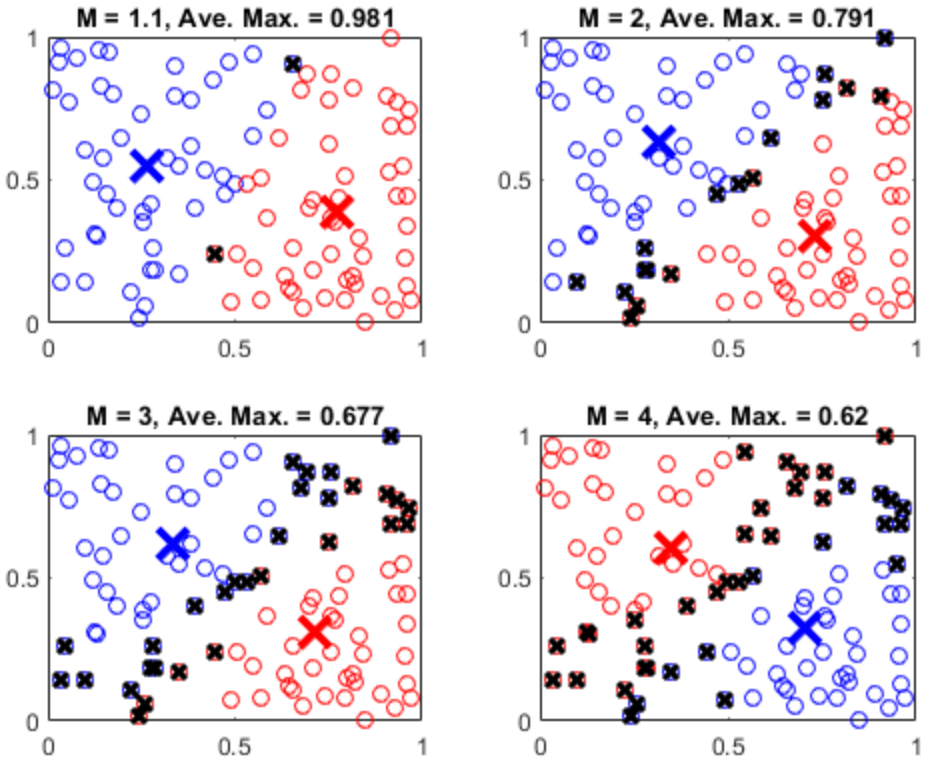
- If $M \rightarrow 1$, the exponent increases the membership weights of points to which the cluster is close. As $M \rightarrow 1$, membership$\rightarrow$1 for the closest cluster and membership$\rightarrow$0 for all the other clusters (this corresponds to k-means).



Fuzziness parameter *M*

$$w_{ij} = \frac{(1/dist(x_i, c_j)^2)^{\frac{1}{M-1}}}{\sum_{q=1}^{k}(1/dist(x_i, c_q)^2)^{\frac{1}{M-1}}}$$

- As M $\rightarrow \infty$, the exponent $\rightarrow$ 0. This implies that the weights $\rightarrow$ 1/k – each point belongs to each cluster with equal weight



As M grows, more and more uncertain memberships

Fuzzy C-means clustering algorithm

- The fuzzy c-means algorithm is very similar to the k-means algorithm:

Choose a number of clusters *c*

To each data point *i*:

assign random weights $w_1 \ldots w_c$ for being in the clusters 1…*c*

Repeat until the algorithm has converged* :

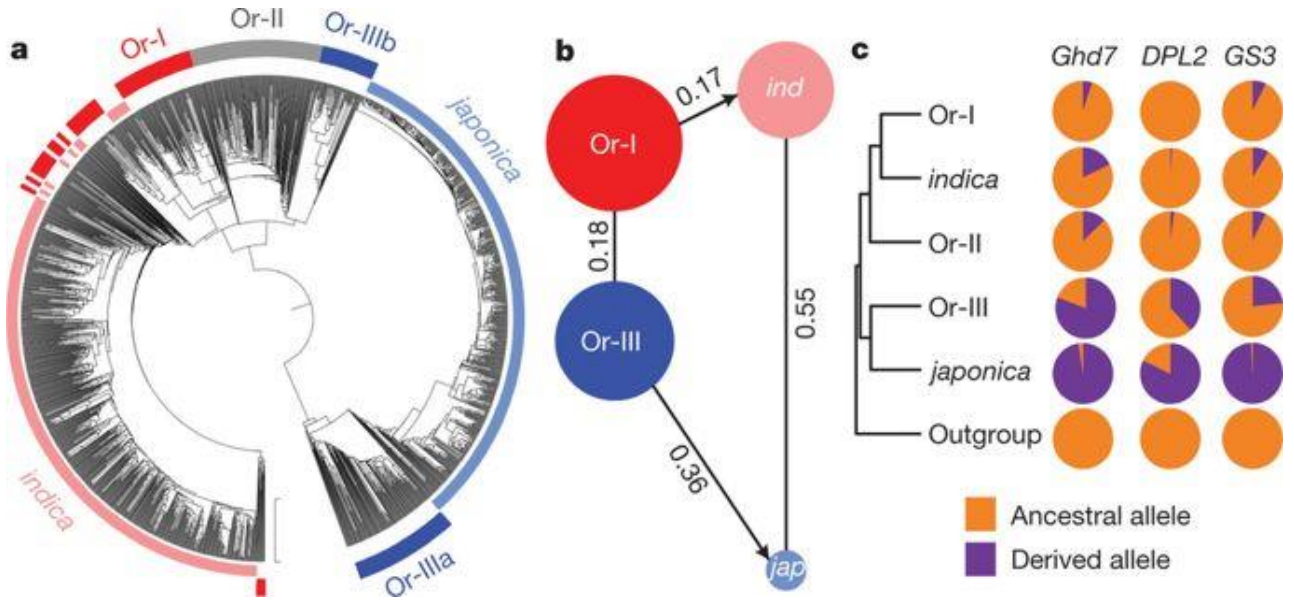
Compute the centroid for each cluster

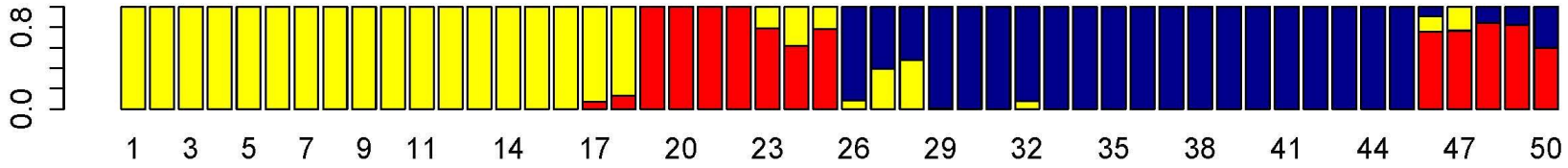
Recompute weights for each point

based on distance from *c* centroids

*that is, the coefficients' change between two iterations is no more than ε, the given sensitivity threshold

Sample application: clustering whole genomes of domesticated rice

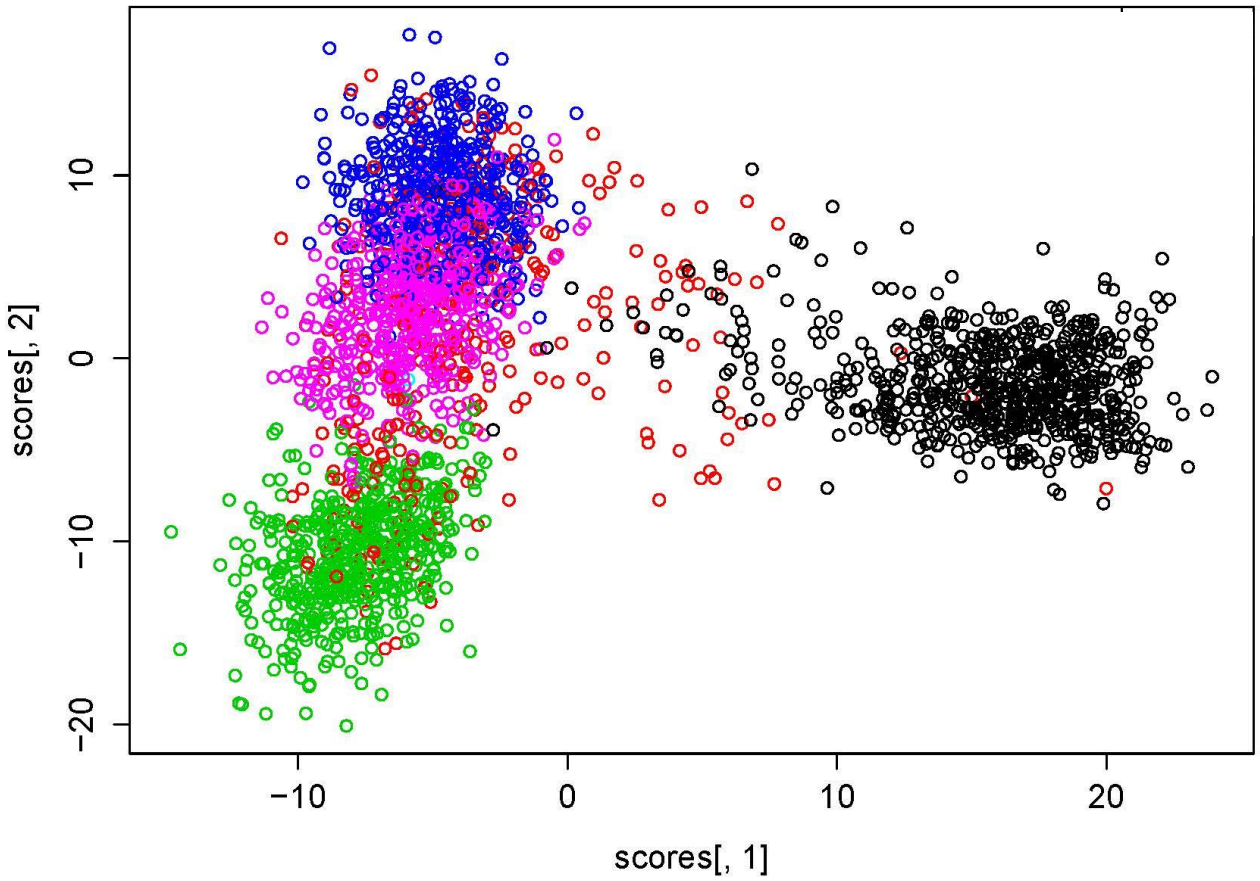




Wild ancestors of domesticated rice *Oryza Sativa*: *O. Indica, O. Japonica*, and *O. Rufipogon*
As you see - most cultivated plants belongs to more than 1 ancestral cluster

https://www.nature.com/articles/nature11532#s1

The same applies to human populations



MIXTURE MODELS AND THE EM ALGORITHM

Statistical approach to fuzzy clustering

Model-based clustering

- In order to understand our data, we will assume that there is a generative process (a model) that creates/describes the data, and we will try to find the model that best fits the data.
  - Models of different complexity can be defined, but we will assume that our model is a distribution from which data points are sampled
  - Example: the data is the height of all people in Greece: height follows Gaussian distribution

Gaussian Distribution

- Example: the data is the height of all people in Greece
  - Experience has shown that this data follows a Gaussian (Normal) distribution
  - Reminder: Normal distribution:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - μ = mean, σ = standard deviation

Gaussian Model

- What is a model?
  - A Gaussian distribution is fully defined by the mean μ and the standard deviation σ
  - We define our model as the pair of parameters $\theta = (\mu, \sigma)$
- This is a general principle: a model is defined as a vector of parameters θ

Fitting the model

- We want to find the normal distribution that best fits our data
  - Find the best values for μ and σ
  - But what does best fit mean?

Maximum Likelihood Estimation (MLE)

- Suppose that we have a vector $X = (x_1, \dots, x_n)$ of values and we want to fit a Gaussian $N(\mu, \sigma)$ model to the data
- Probability of observing point x_i:

$$P(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Probability of observing all points (assume independence)

$$P(X) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- We want to find the parameters $\theta = (\mu, \sigma)$ that maximize the probability $P(X|\theta)$

Maximum Likelihood Estimation (MLE)

- The probability $P(X|\theta)$ as a function of θ is called the Likelihood function

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- It is usually easier to work with the Log-Likelihood function

$$LL(\theta) = -\sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2} - \frac{1}{2} n \log 2\pi - n \log \sigma$$

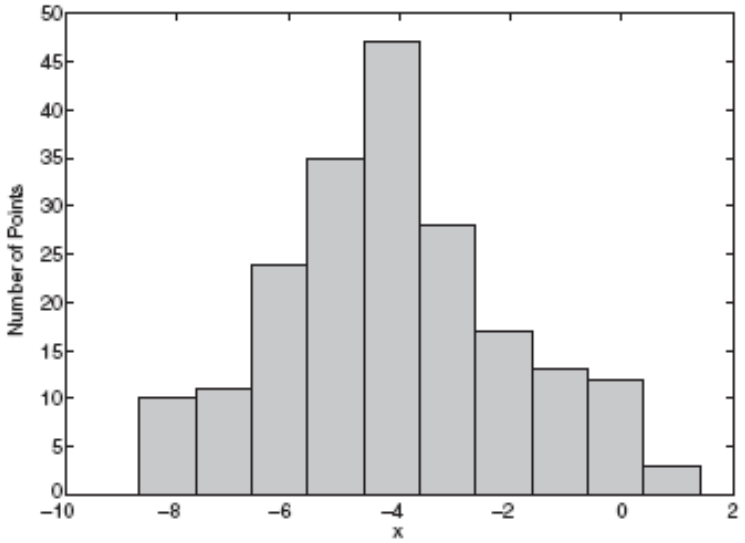
- Maximum Likelihood Estimation
  - Find parameters μ, σ that maximize $LL(\theta)$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \mu_X$$

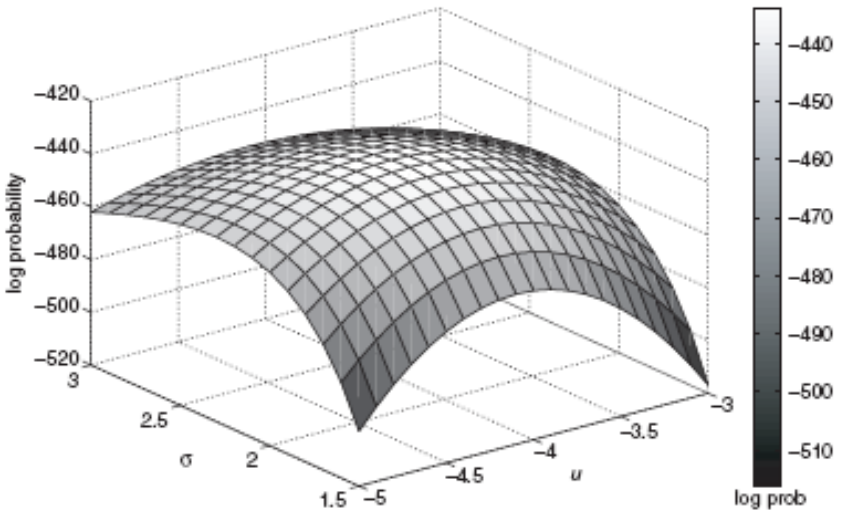
Sample Mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i-\mu)^2 = \sigma_X^2$$

Sample Variance



(a) Histogram of 200 points from a Gaussian distribution.



(b) Log likelihood plot of the 200 points for different values of the mean and standard deviation.

Figure 9.3. 200 points from a Gaussian distribution and their log probability for different parameter values.

MLE

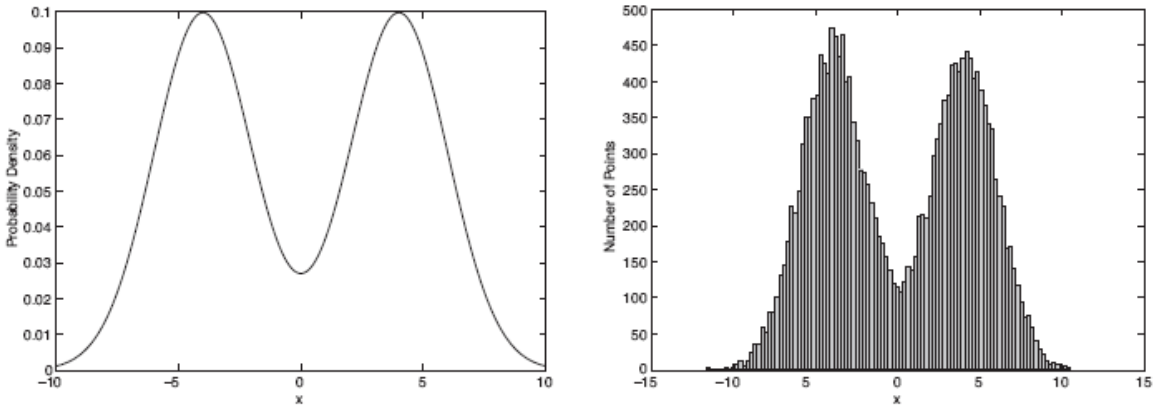
- Note: these are also the most likely parameters given the data

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

- If we have no prior information about θ, or X, then maximizing $P(X|\theta)$ is the same as maximizing $P(\theta|X)$

Mixture of Gaussians

- Suppose that you have the heights data and the distribution looks like the figure below (dramatization)



(a) Probability density function for the mixture model.

(b) 20,000 points generated from the mixture model.

Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

If you look at the mixture, you cannot see the normal distribution at all

Mixture model

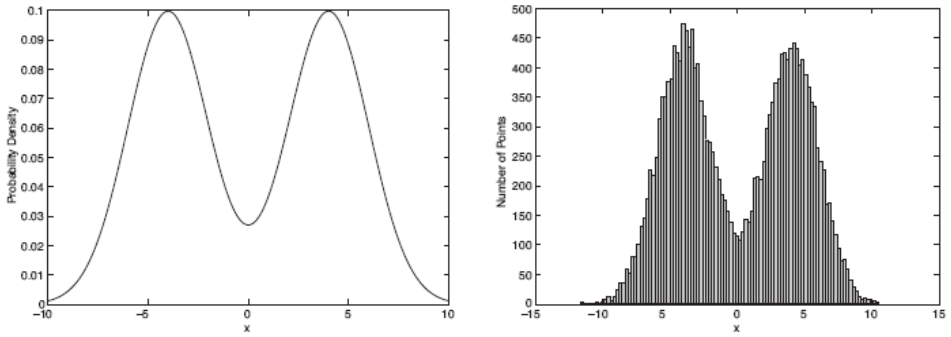
- You suspect that the data is the height of all people in Greece and China
- We define two latent (hidden) classes (variables): with probability π_G the point is drawn from the Greek distribution, with probability $\pi_C = 1 - \pi_G$ from Chinese distribution
- Different distributions correspond to different clusters in the data.

The world's tallest countries
1.Netherlands - 1.838m
2.Montenegro - 1.832m
3.Denmark - 1.826m
4.Norway - 1.824m
5.Serbia - 1.82m

The world's shortest countries
1.Indonesia - 1.58m
2.Bolivia - 1.6m
3.Philippines - 1.619m
4.Vietnam - 1.621m
5.Cambodia - 1.625m

Mixture of Gaussians

- In this case the data is the result of the mixture of two Gaussians
  - One for Greek people, and one for Chinese people
  - Identifying for each value which Gaussian is most likely to have generated it will give us the same as the weights in fuzzy clustering.



(a) Probability density function for the mixture model.

(b) 20,000 points generated from the mixture model.

Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

Generative process

- A value x_i is generated according to the following process:
  - First select the nationality
    - With probability π_G select Greece, with probability π_C select China $(\pi_G + \pi_C = 1)$

We can also think of this as a Hidden Variable Z that takes two values: Greece and China

  - Given the nationality, generate the point from the corresponding Gaussian
    - $P(x_i|\theta_G) \sim N(\mu_G, \sigma_G)$ if Greece
    - $P(x_i|\theta_C) \sim N(\mu_C, \sigma_C)$ if China

θ_G: parameters of the Greek distribution
θ_C: parameters of the China distribution

Mixture Model

- Our model has the following parameters

$$\Theta = (\pi_G, \pi_C, \mu_G, \sigma_G, \mu_C, \sigma_C)$$

Mixture probabilities

θ_G: parameters of the Greek distribution

θ_C: parameters of the China distribution

All the parameters are unknown – all we have is data points and the suspicion that there are 2 clusters here

Mixture Model

- Our model has the following parameters

$$\Theta = (\pi_G, \pi_C, \mu_G, \sigma_G, \mu_C, \sigma_C)$$

Mixture probabilities | Distribution Parameters

- For value x_i, we have:

$$P(x_i|\Theta) = \pi_G P(x_i|\theta_G) + \pi_C P(x_i|\theta_C)$$

- For all values $X = (x_1, \ldots, x_n)$

$$P(X|\Theta) = \prod_{i=1}^{n} P(x_i|\Theta)$$

- We want to estimate the parameters that maximize the Likelihood of all the data

Mixture Models

- Once we have the parameters $\Theta = (\pi_G, \pi_C, \mu_G, \mu_C, \sigma_G, \sigma_C)$ we can estimate the membership probabilities $P(G|x_i)$ and $P(C|x_i)$ for each point x_i:
  - This is the probability that point x_i belongs to the Greek or the Chinese population (cluster)

Given from the Gaussian distribution $N(\mu_G, \sigma_G)$ for Greek

$$P(G|x_i) = \frac{P(x_i|G)P(G)}{P(x_i|G)P(G) + P(x_i|C)P(C)}$$

$$= \frac{P(x_i|\theta_G)\pi_G}{P(x_i|\theta_G)\pi_G + P(x_i|\theta_C)\pi_C}$$

EM (Expectation Maximization) Algorithm

- Initialize the values of the parameters in Θ to some random values
- Repeat until convergence
  - E-Step: Given the parameters Θ estimate the membership probabilities $P(G|x_i)$ and $P(C|x_i)$
  - M-Step: Compute new parameter values that (in expectation) maximize the data likelihood

$$\pi_C = \frac{1}{n}\sum_{i=1}^{n} P(C|x_i) \qquad \pi_G = \frac{1}{n}\sum_{i=1}^{n} P(G|x_i)$$

Fraction of population in G,C

$$\mu_C = \sum_{i=1}^{n} \frac{P(C|x_i)}{n * \pi_C} x_i \qquad \mu_G = \sum_{i=1}^{n} \frac{P(G|x_i)}{n * \pi_G} x_i$$

MLE Estimates if π's were fixed

$$\sigma_C^2 = \sum_{i=1}^{n} \frac{P(C|x_i)}{n * \pi_C} (x_i - \mu_C)^2 \qquad \sigma_G^2 = \sum_{i=1}^{n} \frac{P(G|x_i)}{n * \pi_G} (x_i - \mu_G)^2$$

Relationship to FCM

- E-Step: Assignment of points to clusters with weights - probabilities
  - EM: soft assignment
- M-Step: Computation of centroids
  - Changes the mean and variance for different distributions (clusters)
  - If the variance is fixed then both minimize the same error function

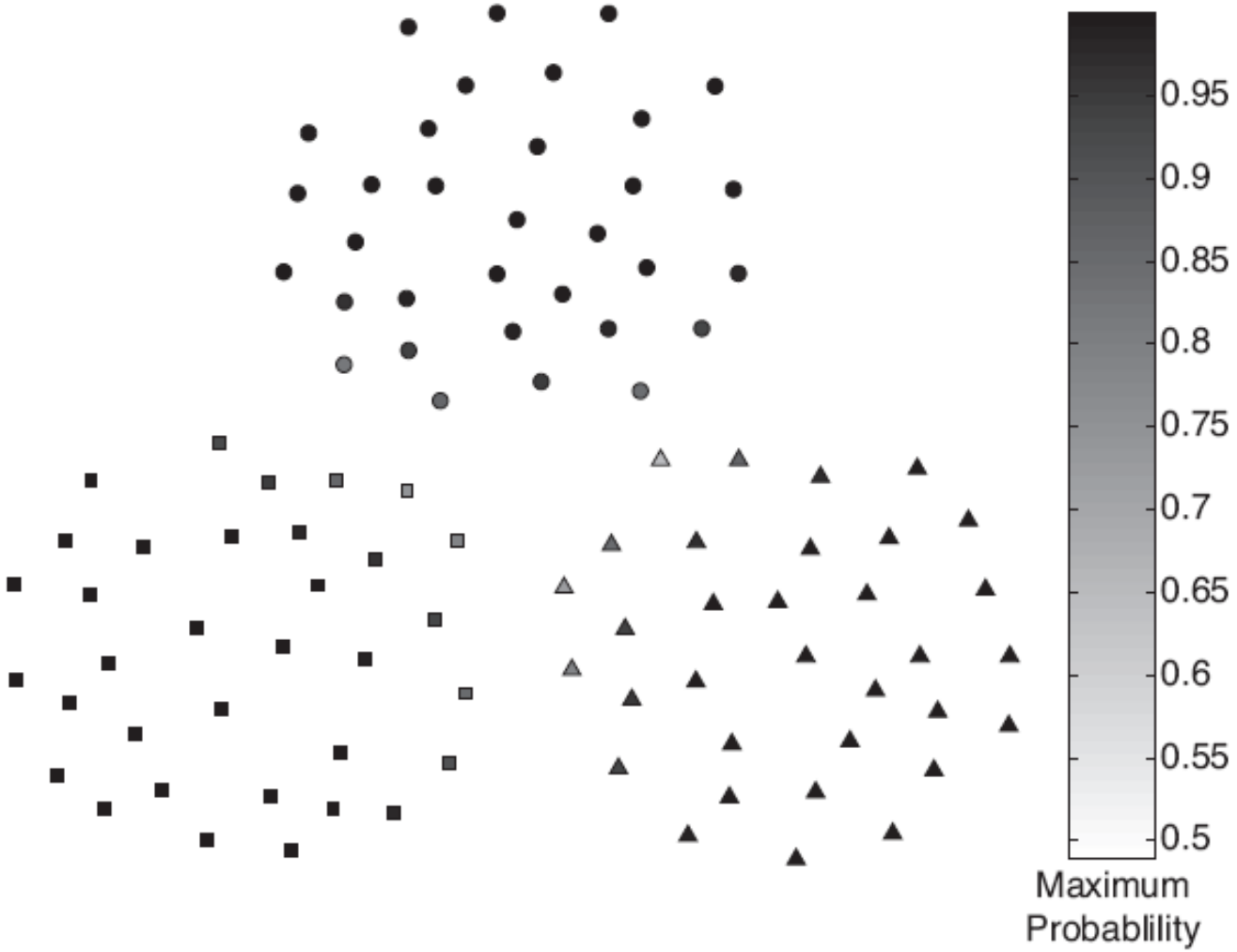


Figure 9.4. EM clustering of a two-dimensional point set with three clusters.

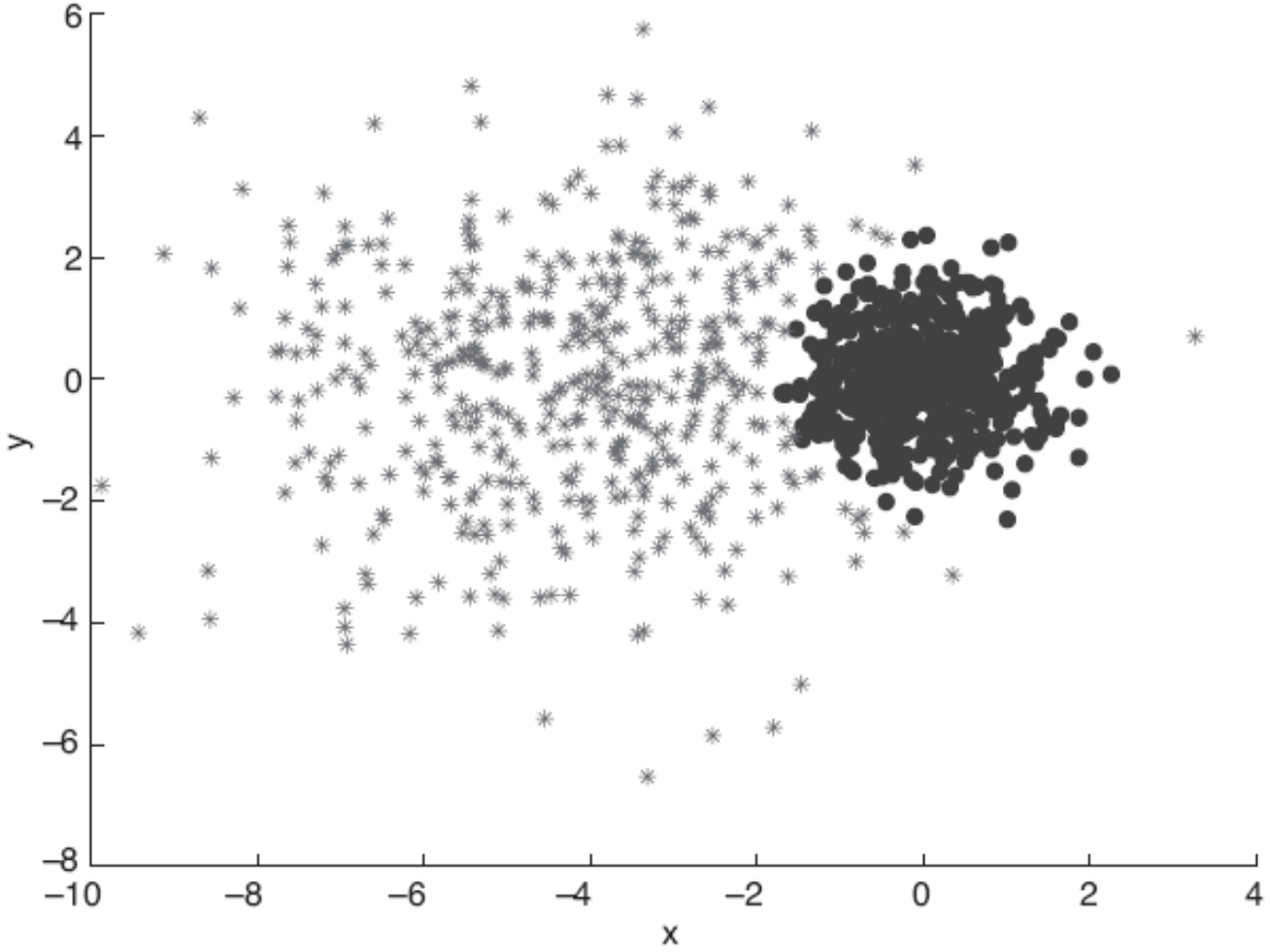


Figure 9.5. EM clustering of a two-dimensional point set with two clusters of differing density.

“All models are wrong but some are useful”
George Box

TOPIC MODELING

Latent Dirichlet Allocation

Clustering documents

- We can apply the same idea to documents and words
- Each document can refer to several topics
- We want to identify main topics and side topics for each document
- We want to learn the topics automatically from a large collection of texts

Generative probabilistic model

- We assume that there is a probability of each word belonging to a specific topic
- The document is generated by first choosing the probabilities of topics and then by choosing the words from these topics according to the probabilities of each word
- This is also a *bag of words* assumption (the order of words is not important). This may be a valid assumption, because, if I take a document, jumble the words and give it to you, you still can guess what sort of topics are discussed in the document

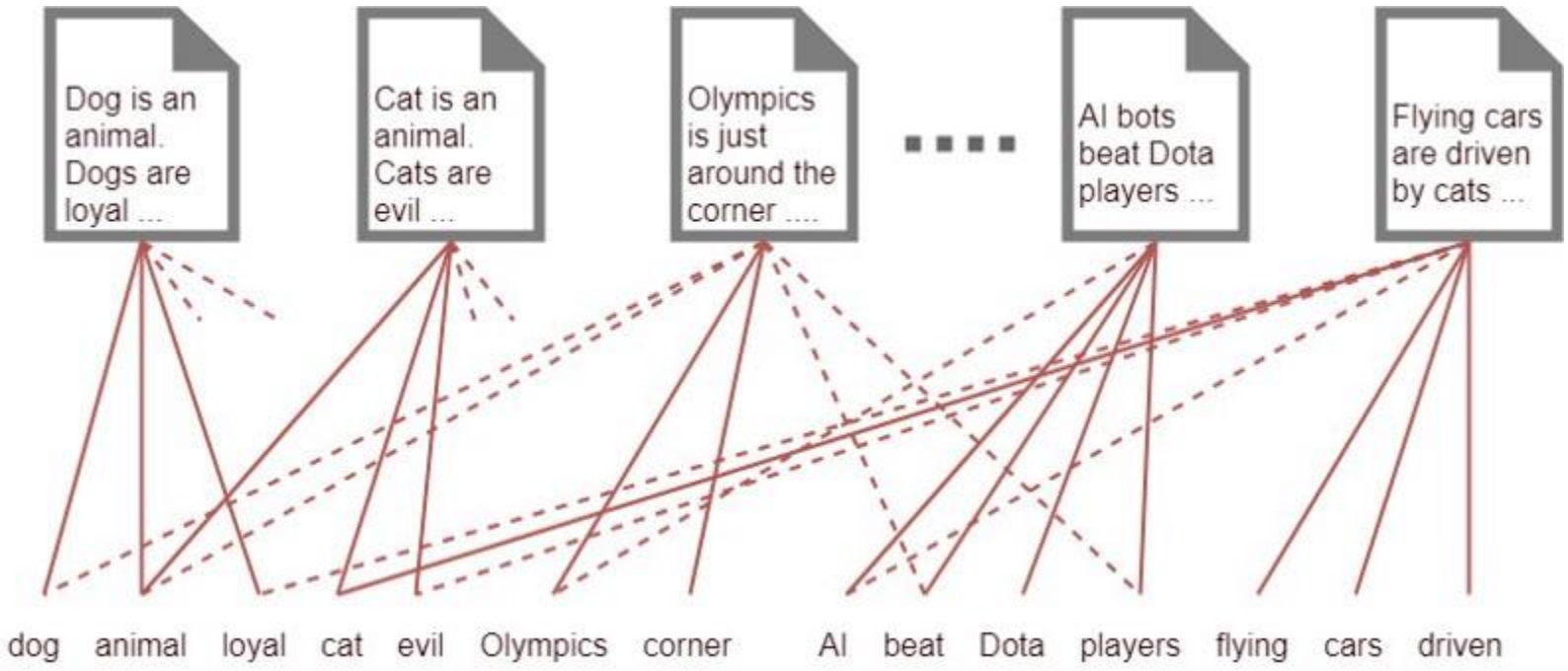
Topic modeling

- Topic modelling refers to the task of identifying topics that best describe a set of documents.
- These topics are not known in advance, and will only emerge during the topic modelling process (therefore called ***latent***).
- One popular topic modelling technique is known as *Latent Dirichlet Allocation* (LDA) – assumes Dirichlet Distributions of both topics and words inside each topic

Applications

- Historians can use LDA to *identify important events in history* by analyzing text based on year.
- Web-based libraries can use LDA to *recommend books based on the topics that you were interested in in the past*.
- News providers can use topic modelling to *understand articles quickly or cluster similar articles*.
- Another interesting application is *unsupervised clustering of images*, where each image is treated similar to a document.

Modeling documents just with words

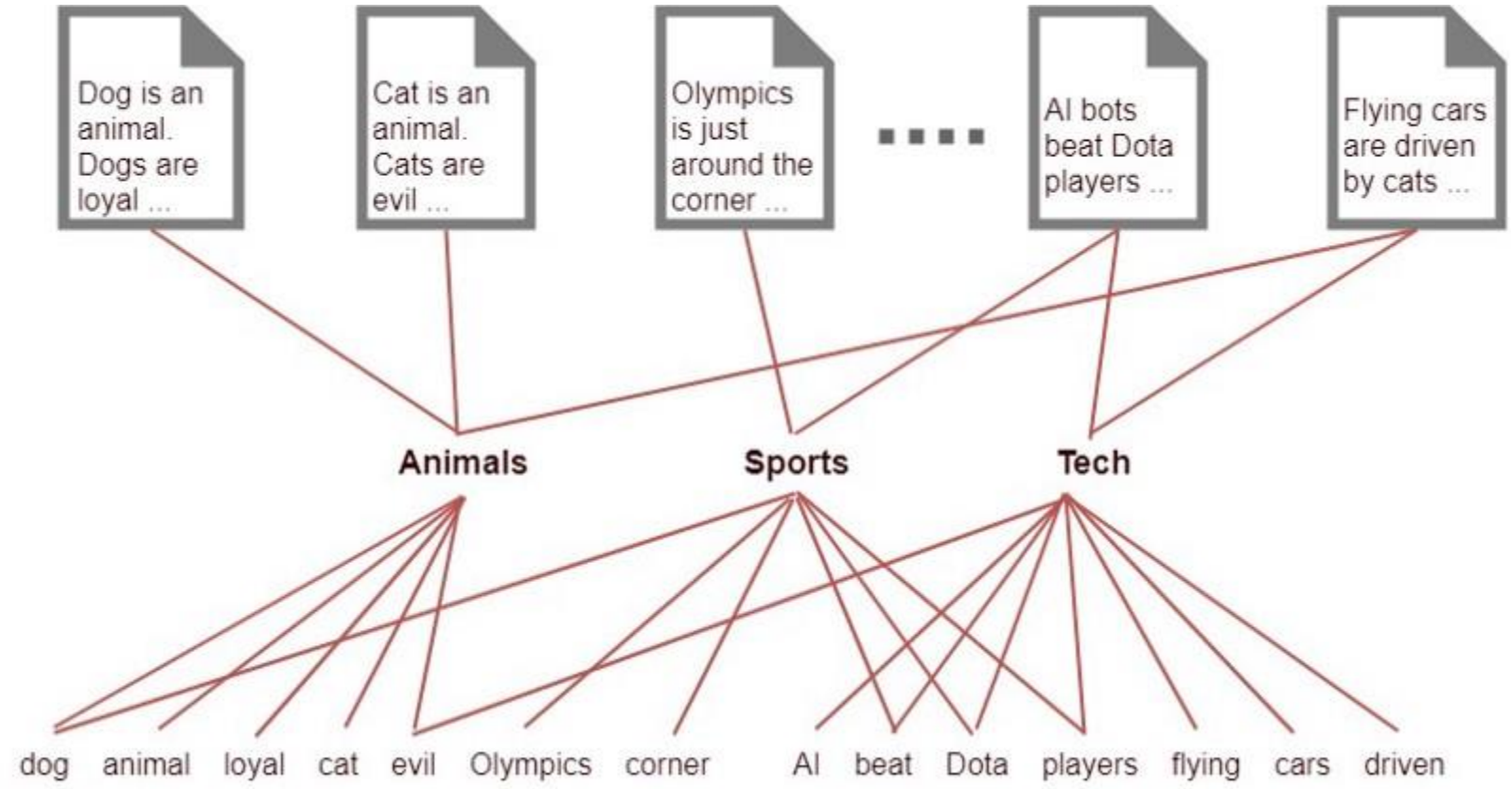


- Some documents are connected to same set of words
- You can see that we can't really infer any useful information due to the large amount of connections

Introducing a topic layer

- We can solve this problem, by introducing a latent (i.e. hidden) layer.
- Say we know that there are 3 topics/themes overall - but these topics are not observed (latent, hidden), we only observe words and documents.
- We want to utilize this information to cut down on the number of threads.
- Then we can connect the words to the topics depending on how well that word fall in that topic and then connect the topics to the documents based on what topics each document touches upon.

Example with topic layer



Each topic can be represented as a distribution of words like *(0.3*Cats,0.4*Dogs,0.2*Loyal, 0.1*Evil)* for the topic “Animals”

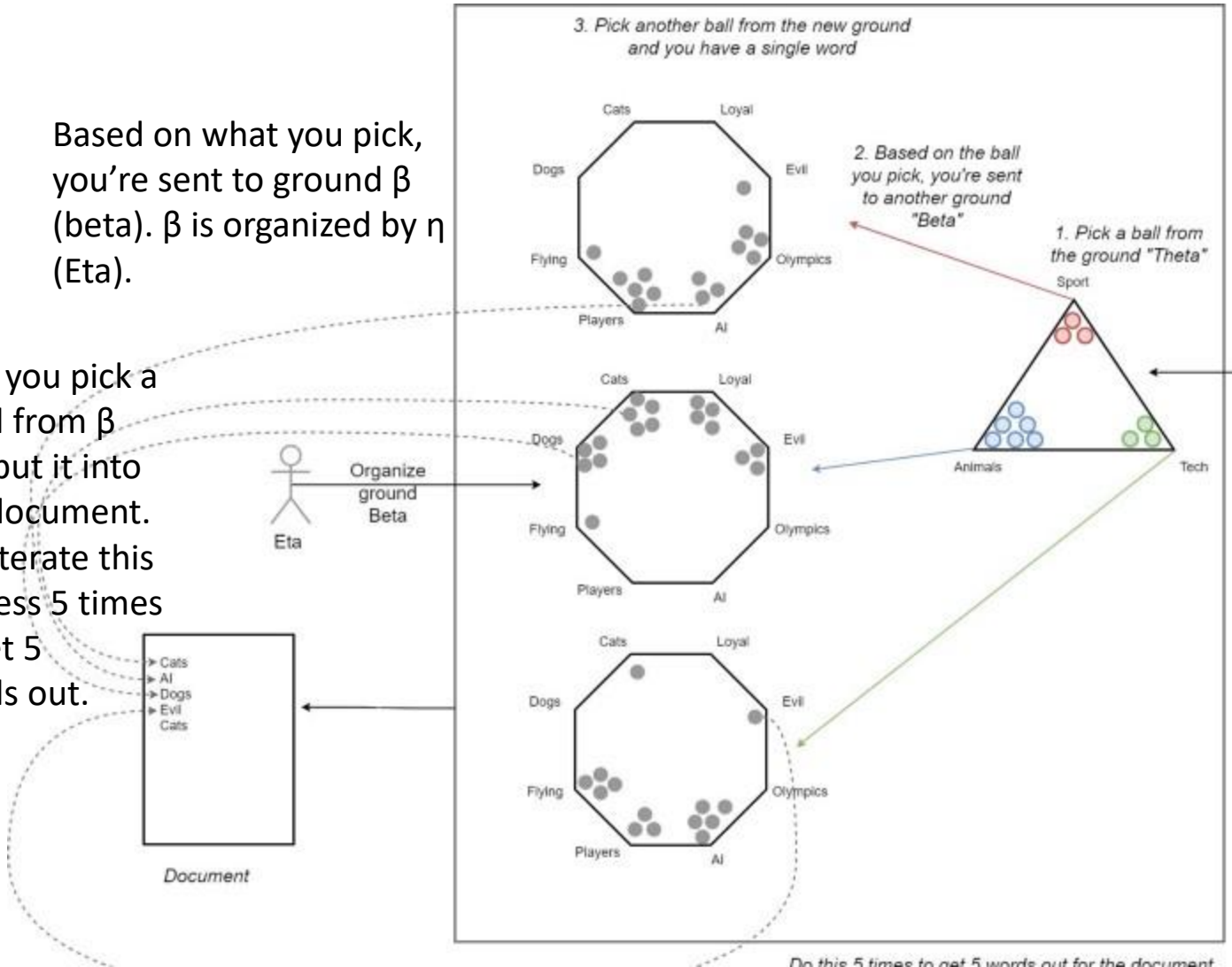
Latent Dirichlet Allocation

- Assumes that the following generative process is behind any document you see:
  - The topic is selected with some probability from the Dirichlet distribution
  - The word is selected with the probability assigned to each word to belong to the selected topic

Generating a new document: model

Based on what you pick, you're sent to ground β (beta). β is organized by η (Eta).

Now you pick a word from β and put it into the document. You iterate this process 5 times to get 5 words out.

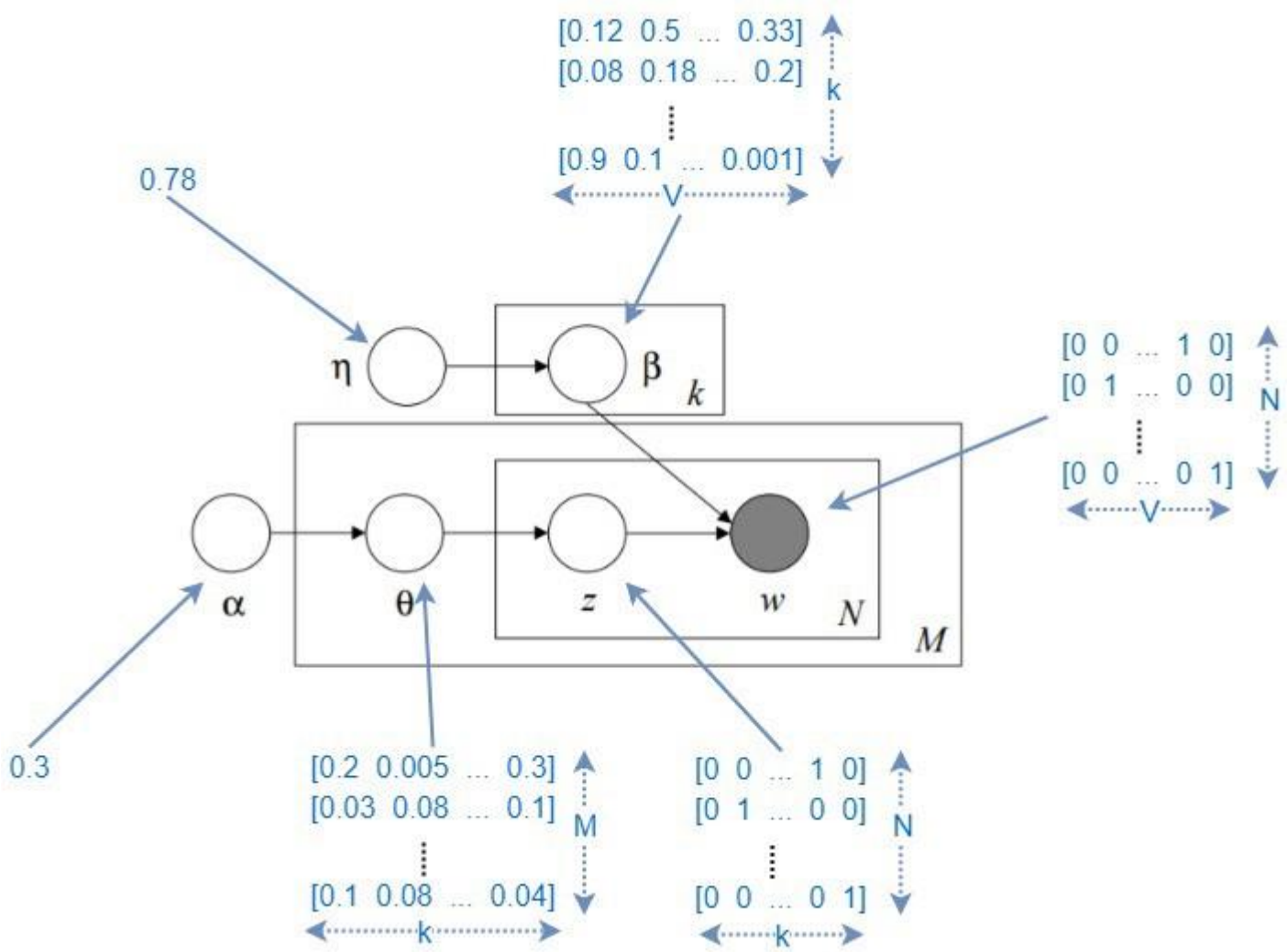


First α (alpha) organizes the ground θ (theta) and then you go and pick a ball from θ.

Definitions and notations

- k—Number of topics a document belongs to (a fixed number)
- V—Size of the vocabulary
- M—Number of documents
- N—Number of words in each document
- *w*—A word in a document. This is represented as a one hot encoded vector of size *V* (i.e. *V*—vocabulary size)
- ***w*** (bold *w*): represents a document (i.e. vector of "*w*"s) of *N* words
- *D*—Corpus, a collection of *M* documents
- z—A topic from a set of *k* topics. A topic is a distribution of words. For example *Animal = (0.3 Cats, 0.4 Dogs, 0 AI, 0.2 Loyal, 0.1 Evil)*

Graphical model of LDA



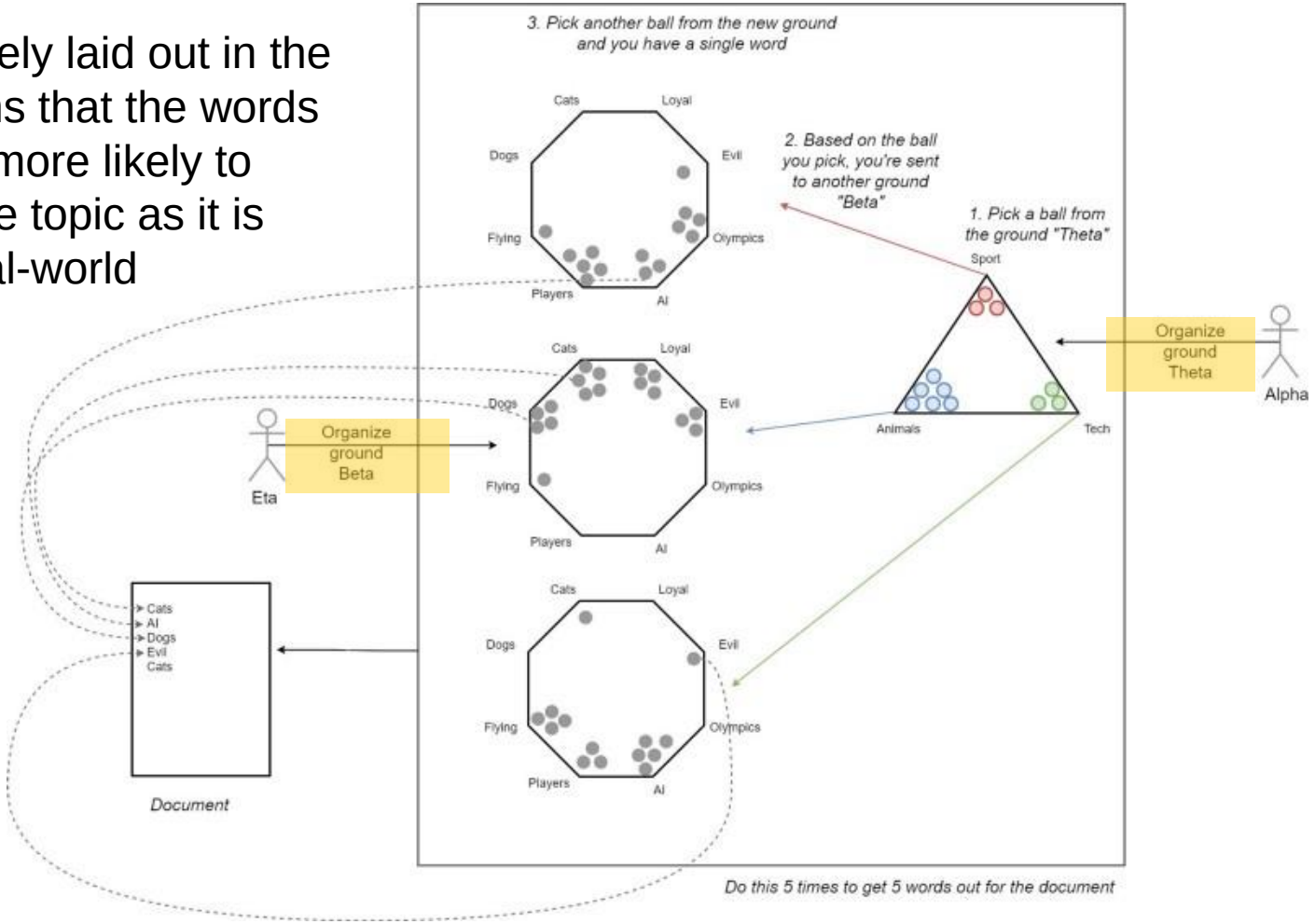
Remember that θ, z, and β are distributions, not deterministic values

Generating a new document

- We have a single α value (i.e. organizer of ground θ) which defines θ - the topic distribution for documents
- We have *M* documents and got a separate θ distribution for each such document
- Any single document has *N* words and each word is generated by a topic. You generate *N* topics to be filled in with words. These N words are still placeholders.
- Now the top plate kicks in. Based on η, β has some distribution (i.e. a Dirichlet distribution to be precise) and according to that distribution, β generates k individual words for each topic.
- Now you fill in a word to each placeholder (in the set of N placeholders), conditioned on the topic it represents.
- Voila, you got a document with *N* words now!

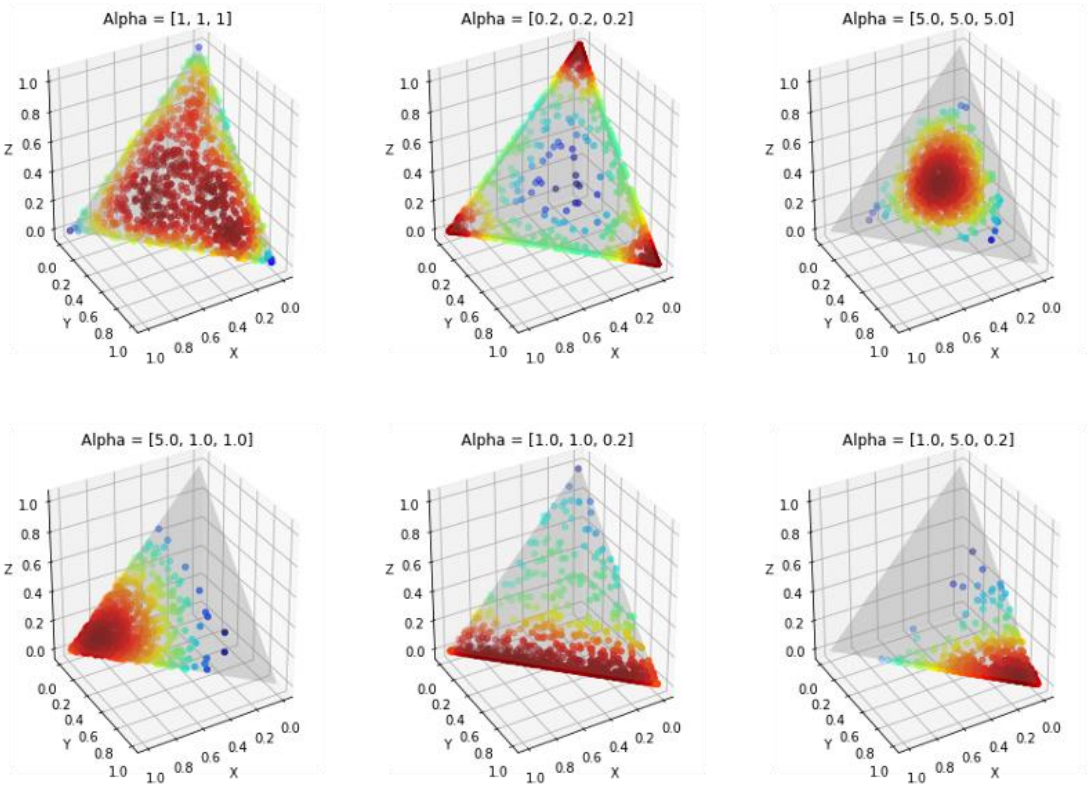
Both θ and β are modeled as a *Dirichlet distribution*

- The balls are nicely laid out in the corners - it means that the words we produce are more likely to belong to a single topic as it is normally with real-world documents.



Dirichlet distribution

- Multivariate generalization of the binary Beta distribution.



Large α values push the distribution to the middle of the triangle, where smaller α values push the distribution to the corners.

How do we find model parameters?

- We need to compute the latent (hidden) variables from a given set of documents:
- α—Distribution-related parameter that governs what the distribution of topics is for all the documents in the corpus looks like
- θ—Random matrix where θ(i,j) represents the probability of the i-th document to containing the j-th topic
- η—Distribution-related parameter that governs what the distribution of words in each topic looks like
- β—A random matrix where β(i,j) represents the probability of i-th topic containing the j-th word.

Learn the following model

$$P(\theta_{1:M}, \mathbf{z}_{1:M}, \beta_{1:k} | \mathcal{D}; \alpha_{1:M}, \eta_{1:k})$$

- Find the joint posterior probability of:
  - θ—A distribution of topics, one for each document,
  - **z**—N Topics for each document,
  - β—A distribution of words, one for each topic,
- Given:
  - D—All the data we have (i.e. the corpus),
- and using parameters:
  - α—A parameter vector for each document (document—Topic distribution)
  - η—A parameter vector for each topic (topic—word distribution)
- But we do not know any of the parameters!
- So the problem seems entirely intractable

Variational inference

- The probability we discussed above is a very messy intractable posterior (meaning we cannot calculate that on paper and have nice equations)
- The only thing we can do is to approximate that with some known probability distribution that closely matches the true posterior
- That's the idea behind variational inference.
- The way to do this is to minimize the KL divergence between the approximation and true posterior using optimization techniques.

Optimization problem

$$\gamma^\star, \phi^\star, \lambda^\star = \text{argmin}_{(\gamma,\phi,\lambda)} D(q(\theta, \mathbf{z}, \beta|\gamma, \phi, \lambda)||p(\theta, \mathbf{z}, \beta|\mathcal{D}; \alpha, \eta)$$

- γ , ϕ and λ represent the free variational parameters with which we approximate θ, **z** and β.
- Here D(q||p) represents the KL divergence between *q* and *p*. And by changing γ, ϕ and λ, we get different *q* distributions having different distances from the true posterior *p*.
- Our goal is to find the γ* , ϕ* and λ* that minimize the KL divergence between the approximation *q* and the true posterior *p*.
- Now it's just a matter of iteratively solving the above optimization problem until the solution converges.
- Once you have γ* , ϕ* and λ* you have everything you need in the final LDA model.