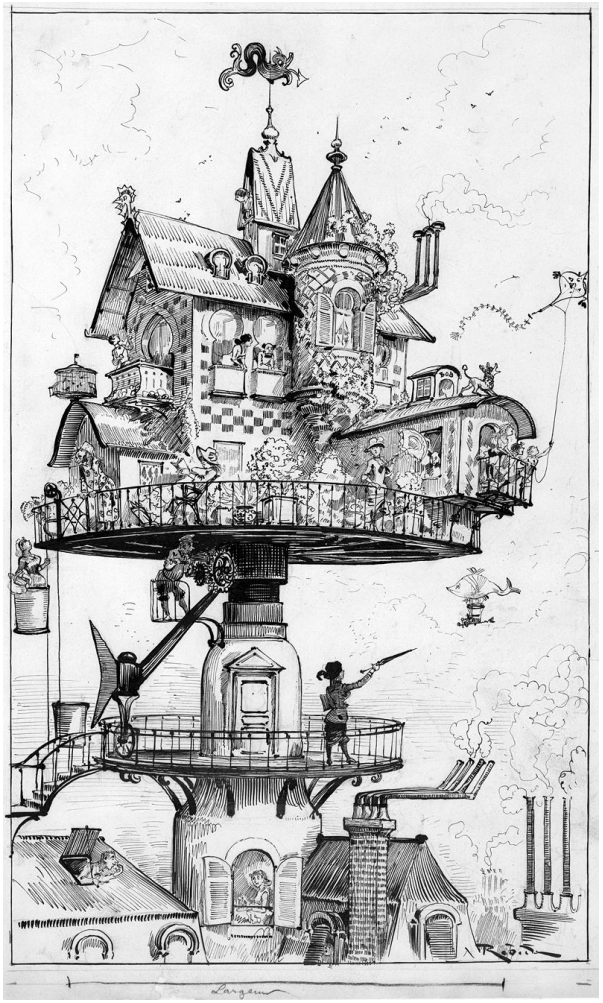
Foundations of Unsupervised Learning

Wray Buntine
Monash University
http://Bayesian-Models.org

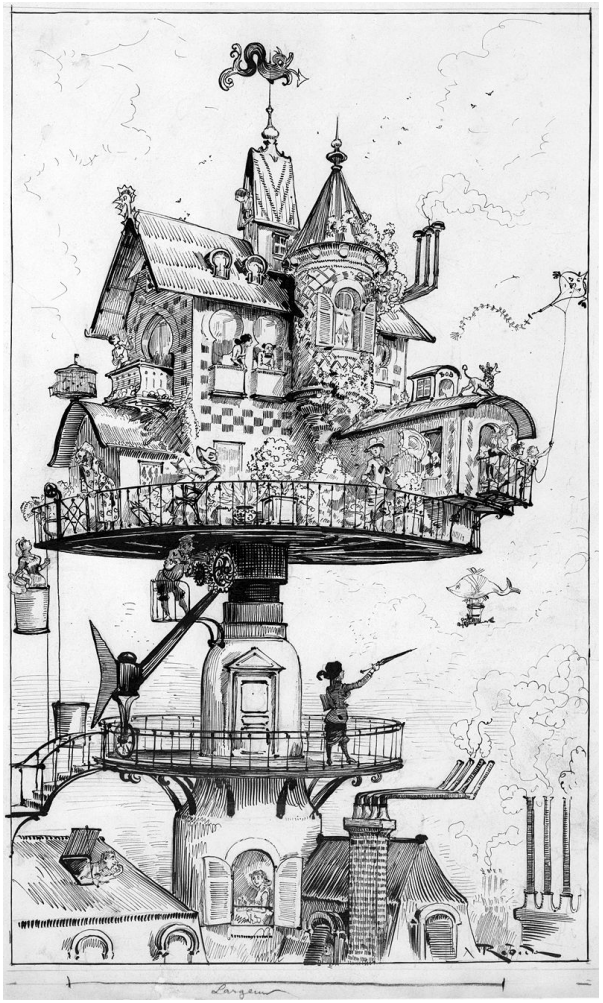
2019-07-09

Preliminaries

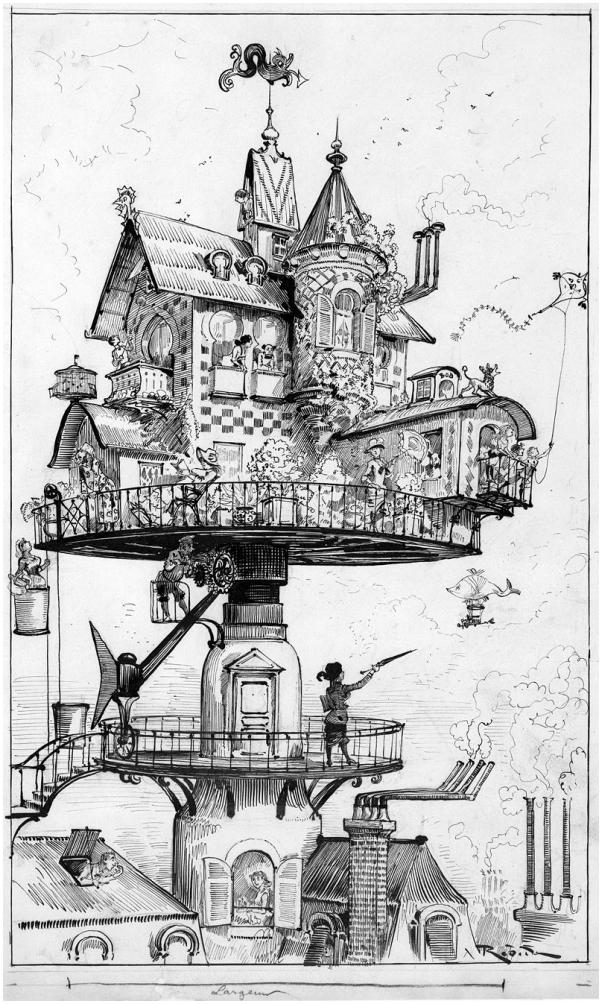


- links on handout are live
- most items in green link to Wikipedia pages
- an introduction to concepts, not math

Outline

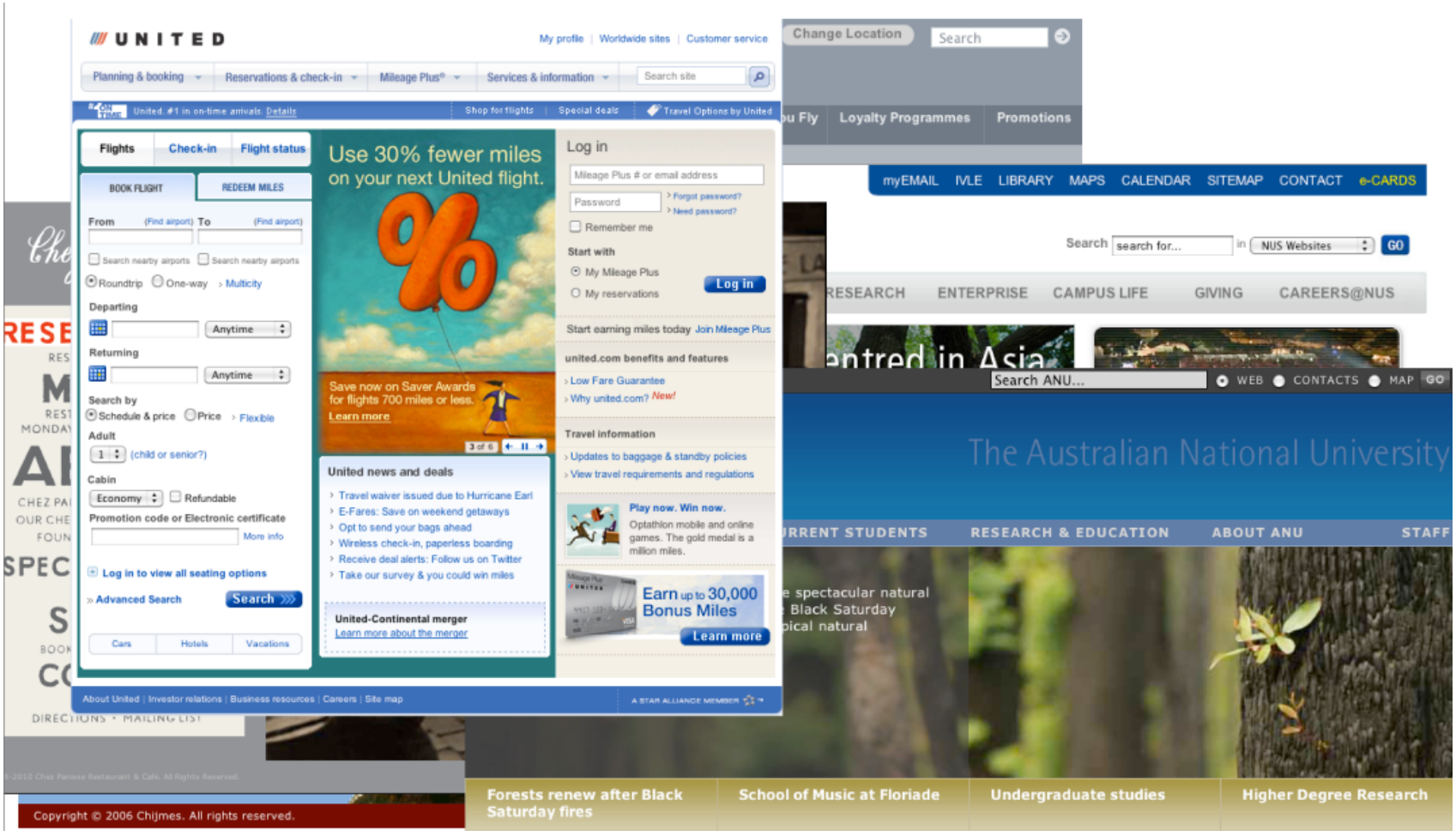


Outline



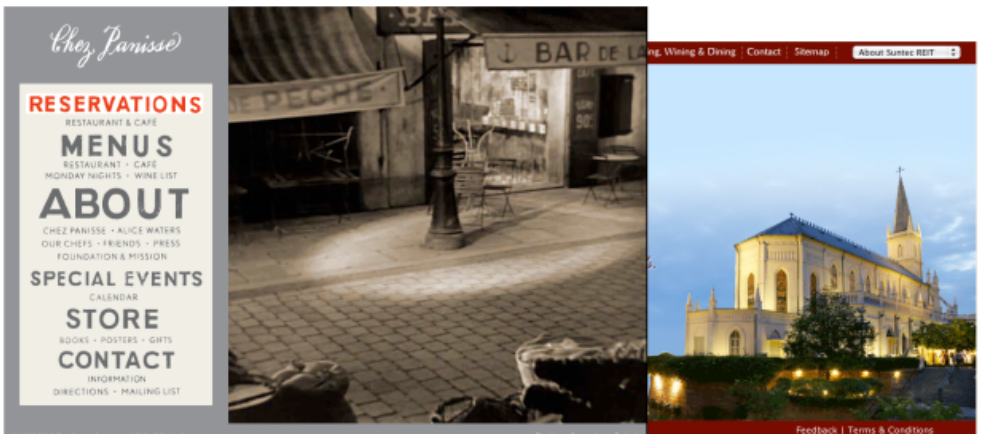
Clustering Web Pages: Data

slide from Ahmed, MLSS 2014



Clustering Web Pages: Clusters

slide from Ahmed, MLSS 2014



Clustering Web Pages: Labels

slide from Ahmed, MLSS 2014

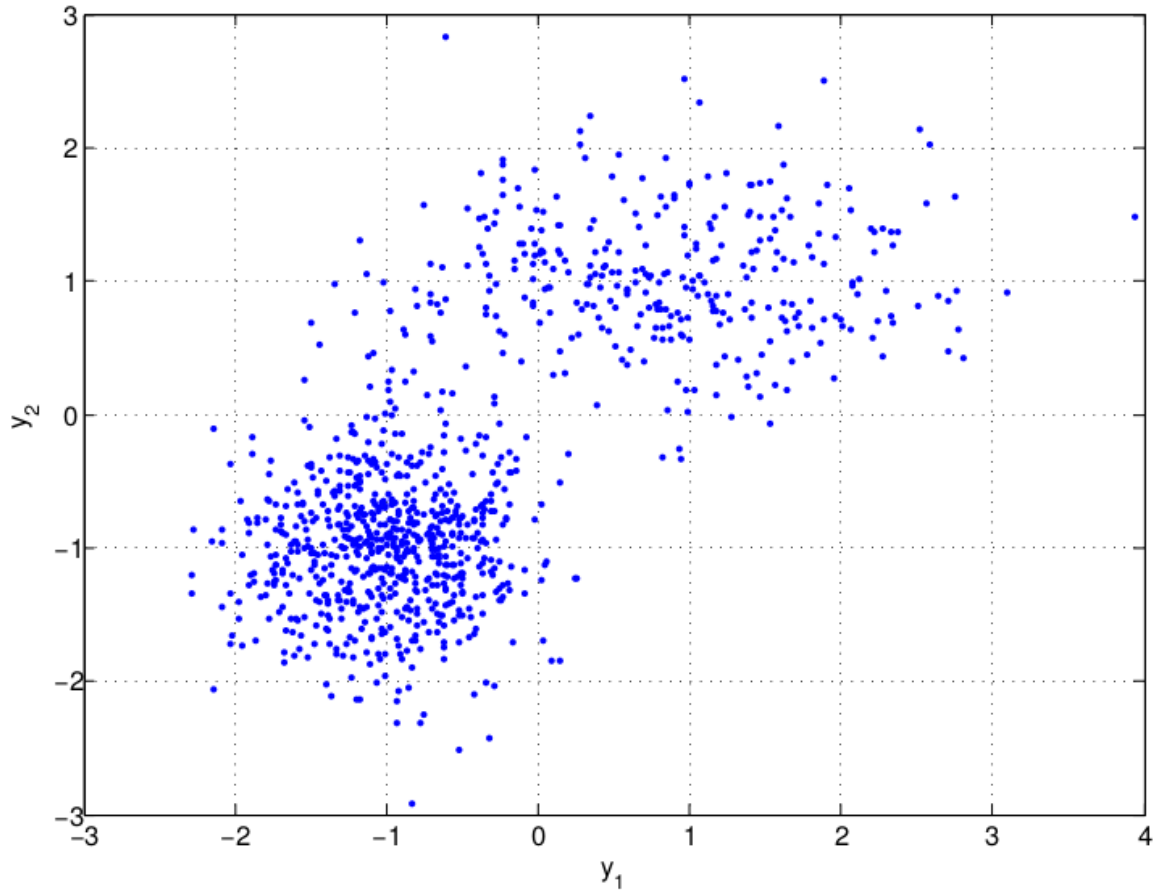
Clustering Web Pages: Questions

- How do we build the clusters?
- How many clusters (we use K to denote this)?
- How do we get the labels?
- What are the clusters used for?

NB. Amr Ahmed says (roughly) "we have huge amounts of unlabelled data, so if we can use it we should!"

Clustering Example for $K = 2$

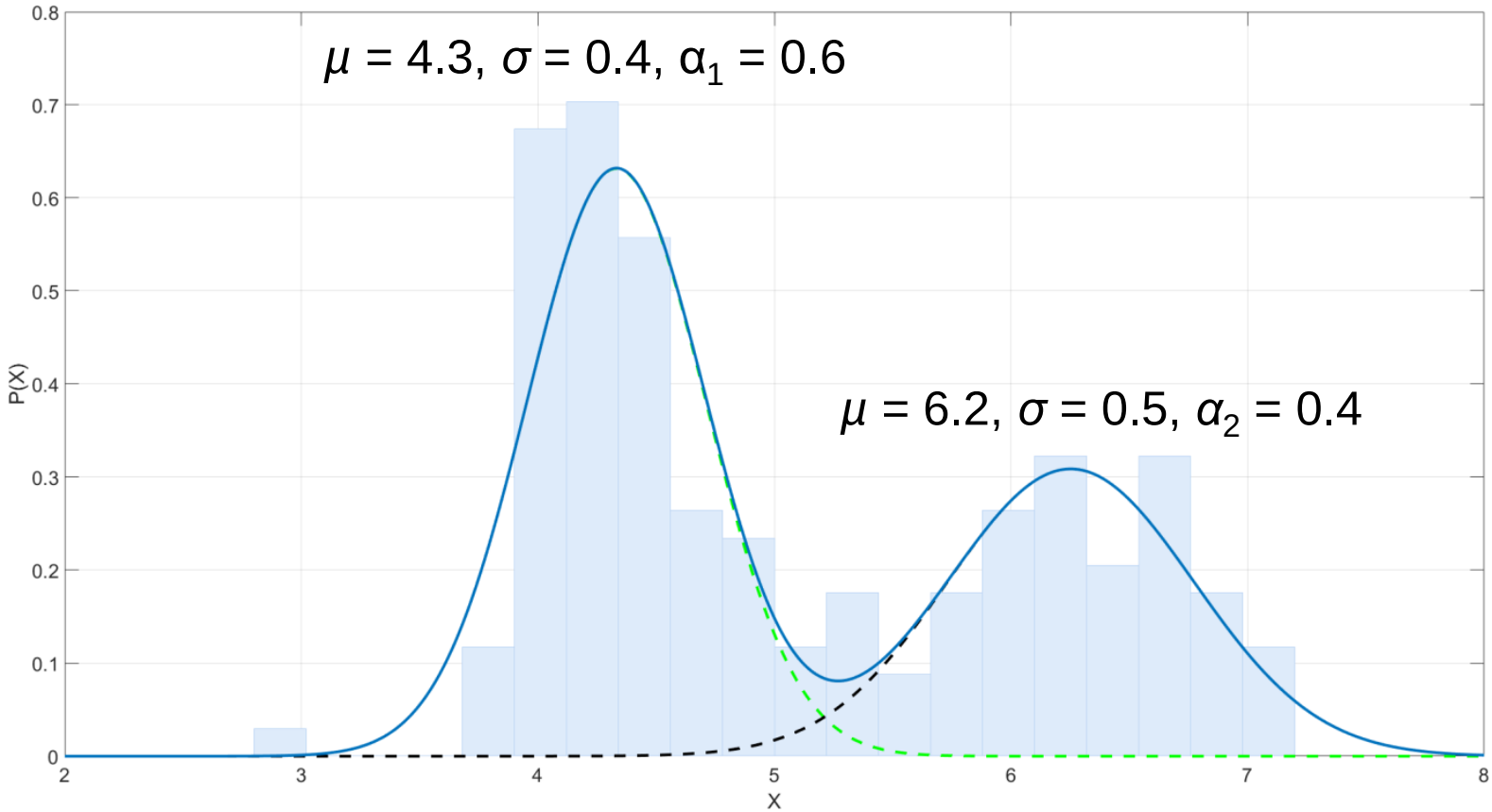
- Artificial data example



- Chosen so that the "clusters" are obvious for demonstration purposes

Mixture Modelling Example for $K = 2$

- Example: two normal distributions mixed with $\vec{\alpha} = (0.6, 0.4)$



Mixture Modelling

- Models data for each feature as a mixture of probability distributions for i-th data point $\boldsymbol{x}_i$

$$p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)$$

  where
  - K is the number of classes
  - $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$ are the mixing weights
  - $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ are the parameters of the distributions
- Has an explicit probabilistic form
  $\Longrightarrow$ allows for statistical interpretion
- As a model is usually rather artificial.
- But, it does have efficient algorithms.

Mixture Modelling, cont.

- Alternatively, expand the sum for the previous formula.
- Introduce a latent class variable c_i, and for i-th data point $\boldsymbol{x}_i$

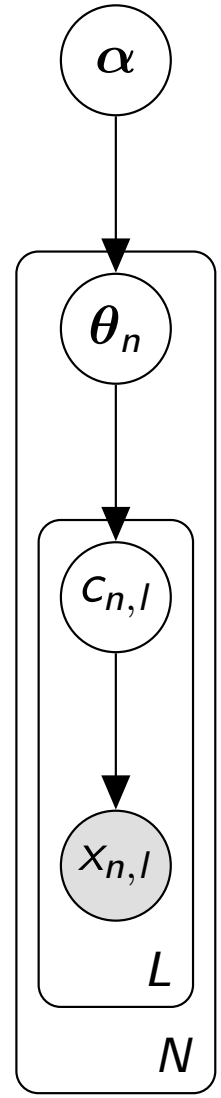
$$p(\boldsymbol{x}_i, c_i|\boldsymbol{\alpha}, \boldsymbol{\Theta}) = \alpha_{c_i} p(\boldsymbol{x}_i|\boldsymbol{\theta}_{c_i})$$

where

  - c_k plays the role of k in the previous sum
  - $p(c_i{=}k|\boldsymbol{\alpha}) = \alpha_k$ are the priors for c_i

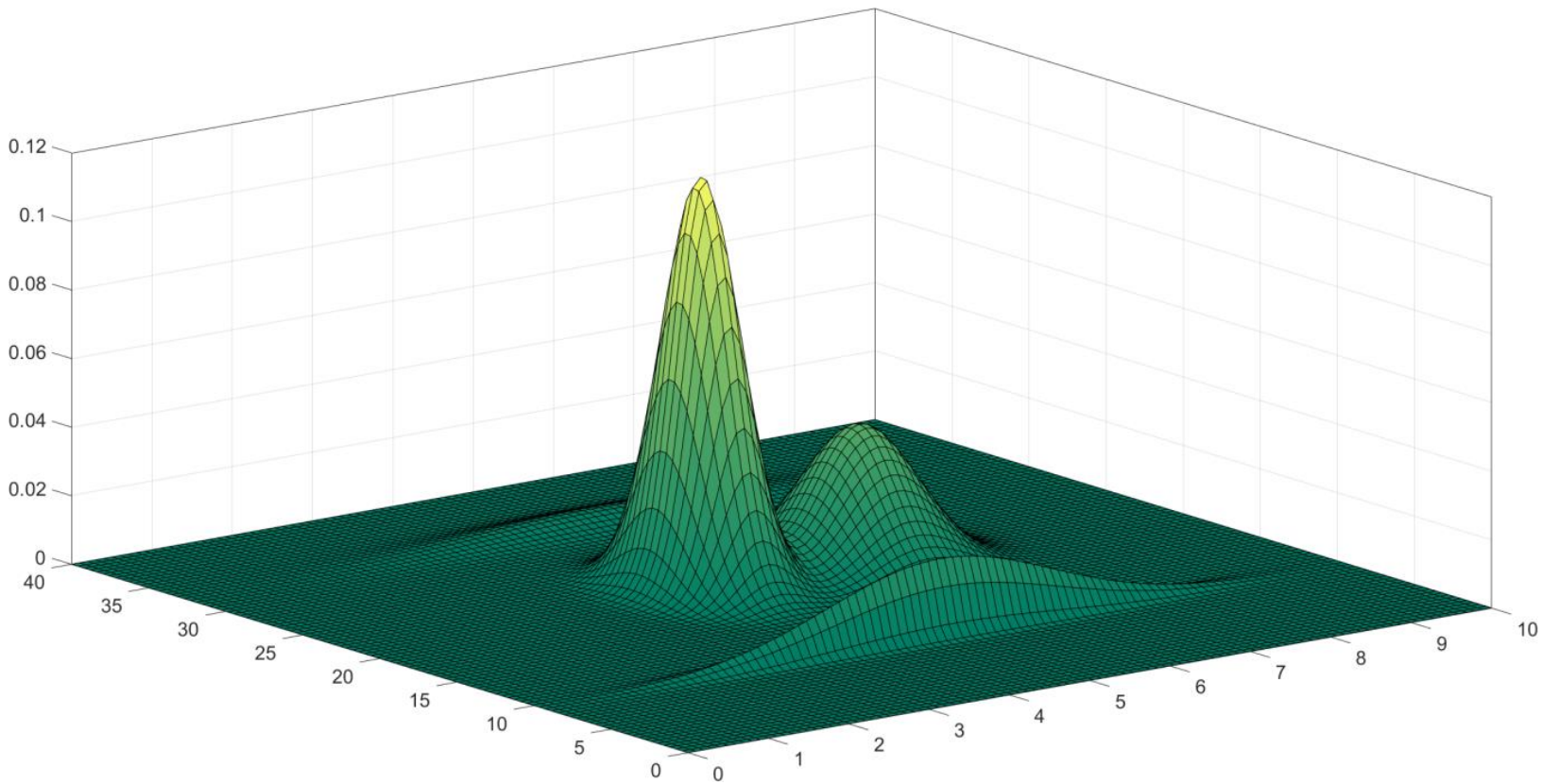
Notation

- data: variables that are observed,
  - *e.g.* matrix $\boldsymbol{X}$ with row vectors $\boldsymbol{x}_i$ with individual entries $x_{i,j}$
- latent variable: is an (always) unobserved variable attached to data
  - *e.g.* matrix $\boldsymbol{C}$ with row vectors $\boldsymbol{c}_i$ with individual entries $c_{i,j}$
- parameter: a variable not attached to data but occurs in a probability distribution for the data
  - *e.g.* a matrix $\boldsymbol{\Theta}$ with row vectors $\boldsymbol{\theta}_i$ with individual entries $\theta_{i,k}$
- hyper-parameter: a variable not attached to data, nor is it a parameter, but occurs in a probability distribution for a parameter or for another hyper-parameter
  - *e.g.* a vector $\boldsymbol{\alpha}$ with individual entries α_k



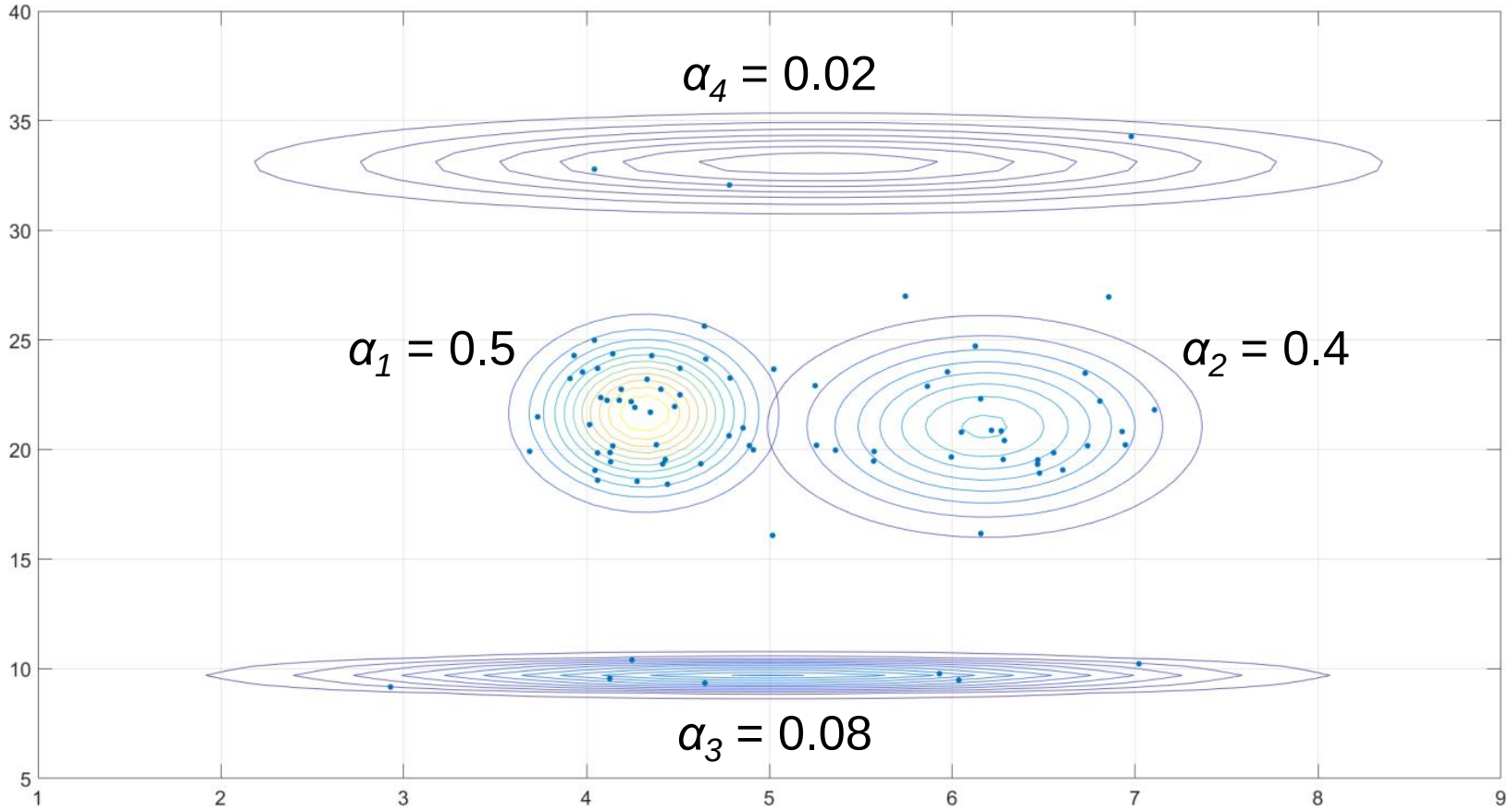
Clustering Example Model

- Plot of the Gaussian mixture model density



Clustering Example for $K = 4$

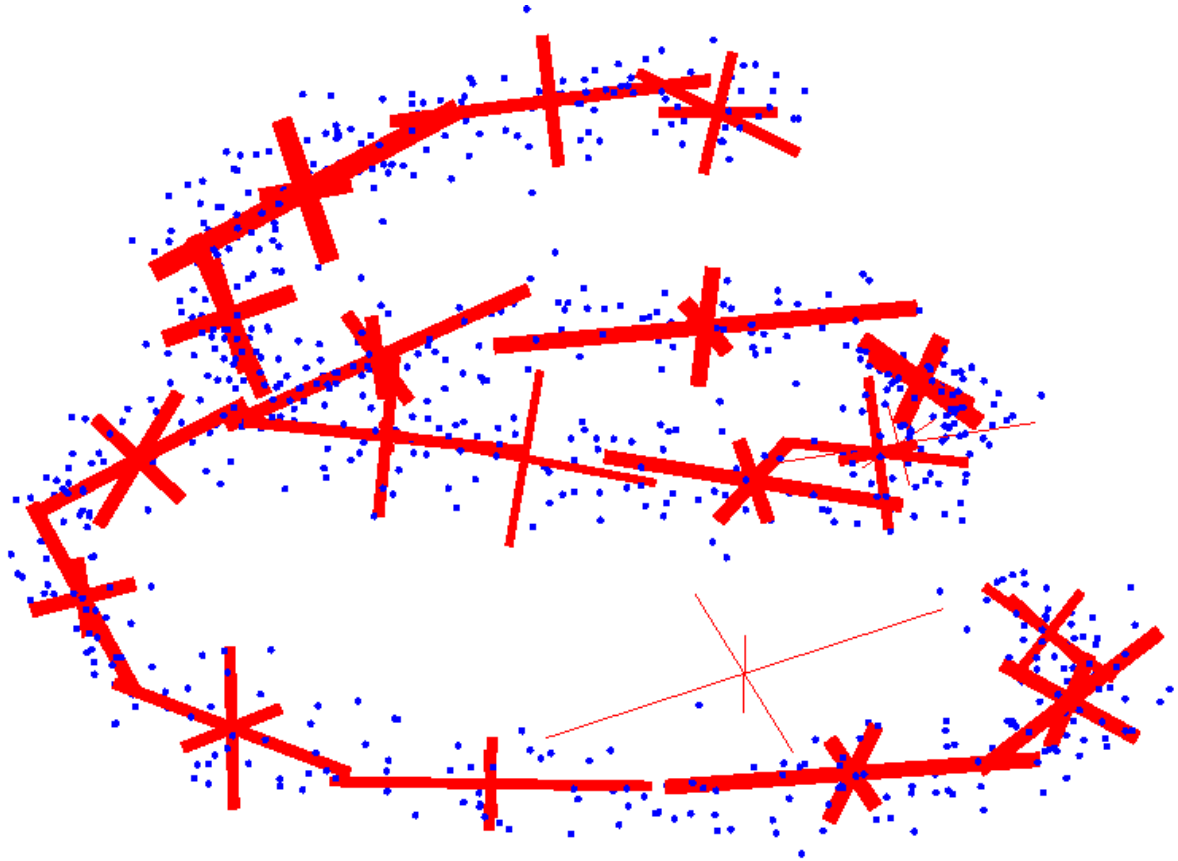
- Gaussian mixture artificial data example



- Could you reconstruct the model from so few data points?
- You really need to concurrently learn the mixing weights α!

Clustering Example for $K = \infty$

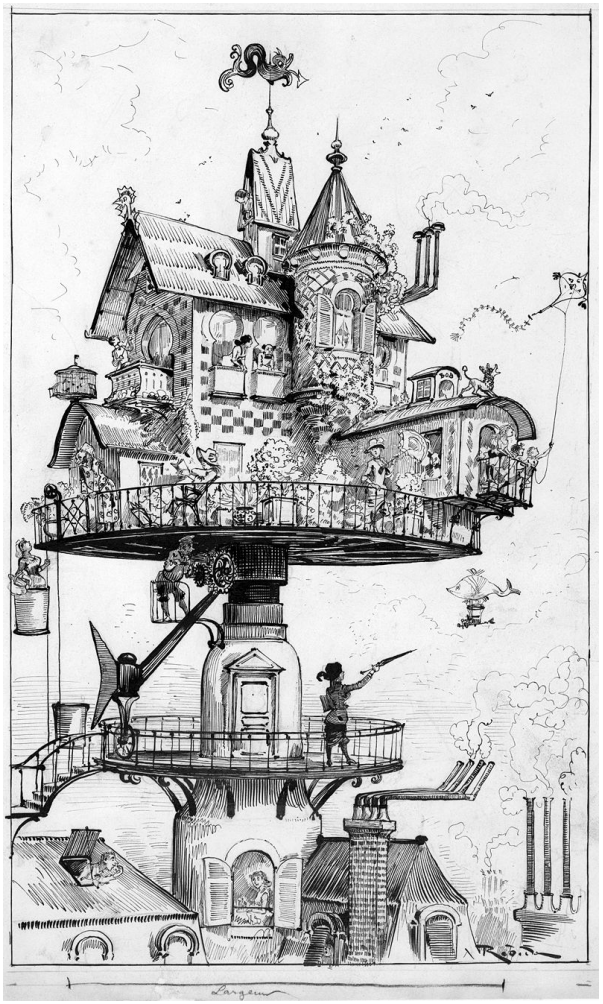
- Let's use infinite number!
  (each red cross is one Gaussian component)



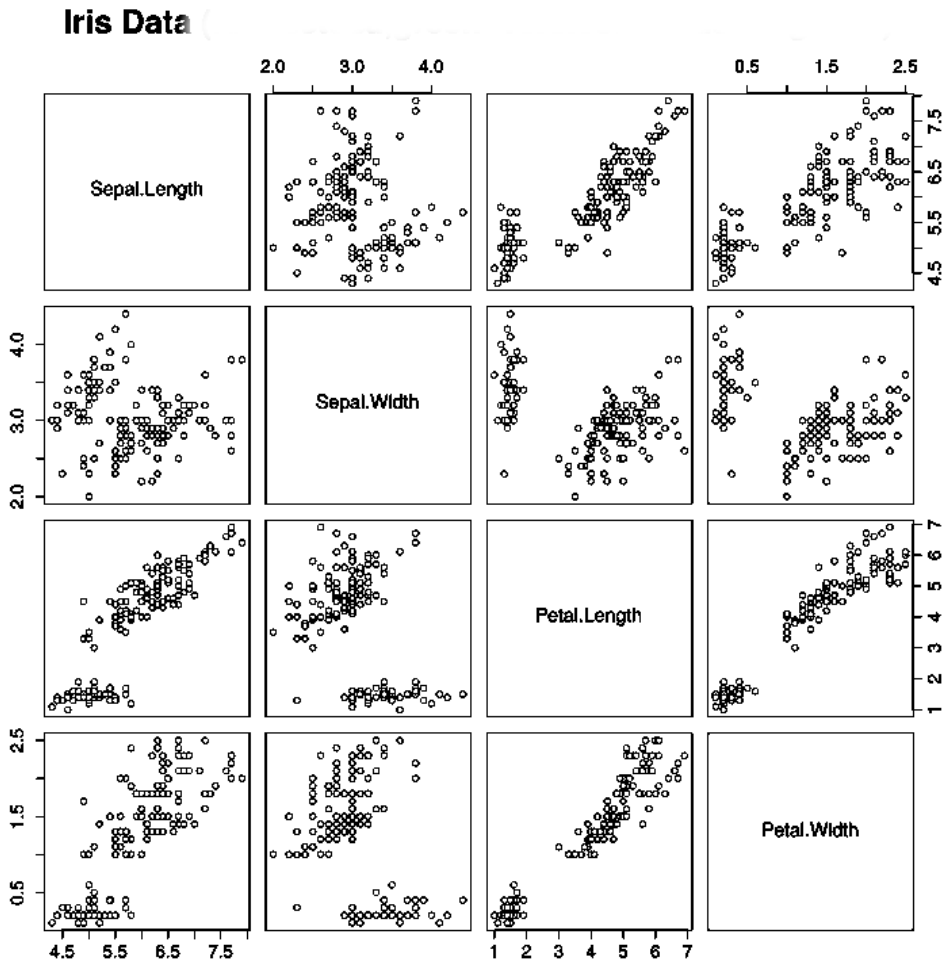
from "The Infinite Gaussian Mixture Model," Rasmussen, *NIPS*, 2000

- Before the modern era of deep neural nets, even more artificial!

Outline



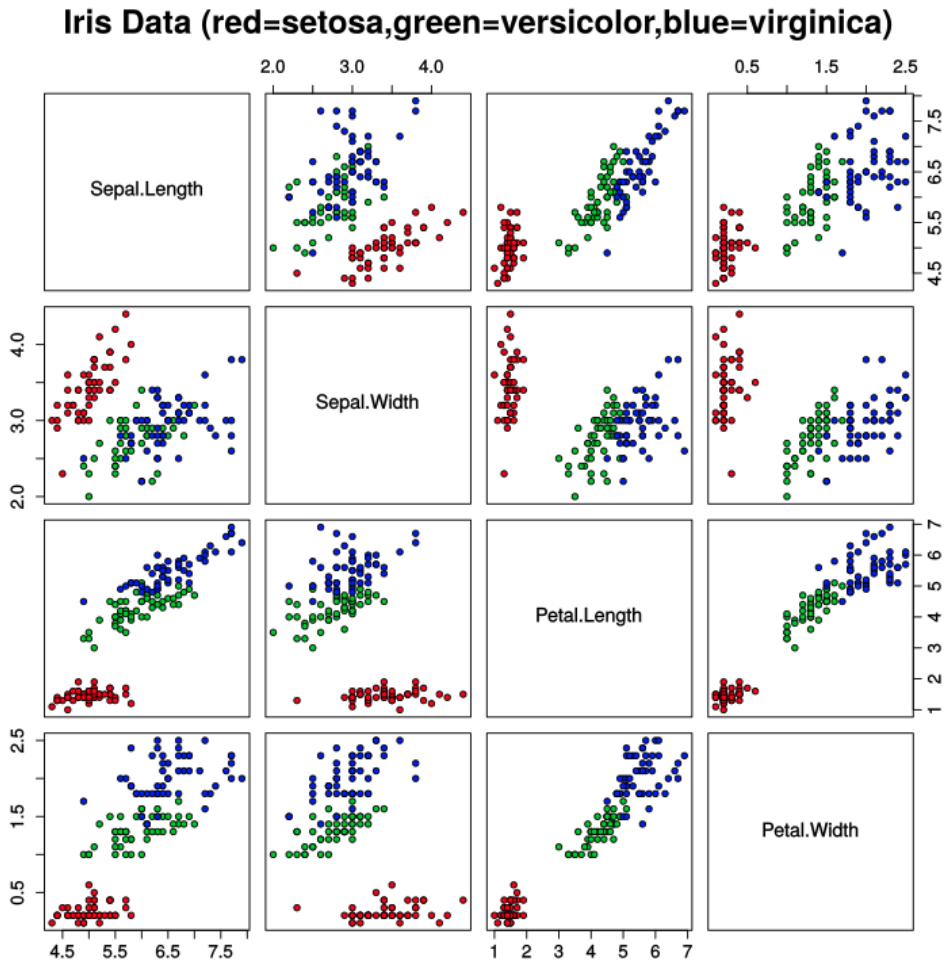
The Iris Data



by Nicoguaro [CC BY 4.0], from Wikimedia Commons

Only viewing two features at once can make it hard to see the clusters!

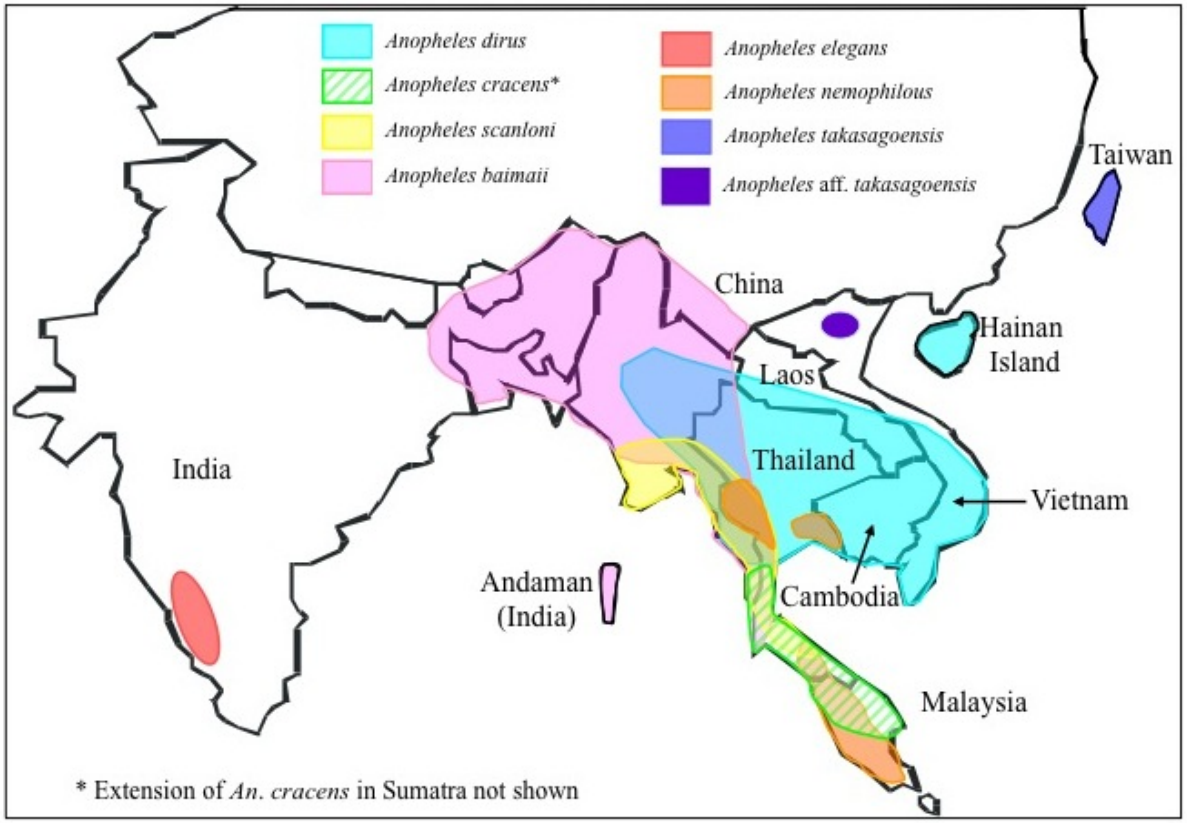
The Iris Data: Actual Clusters



by Nicoguaro [CC BY 4.0], from Wikimedia Commons

only viewing two features at once can make it hard to see the clusters

How Many Species of Mosquitoes are There?

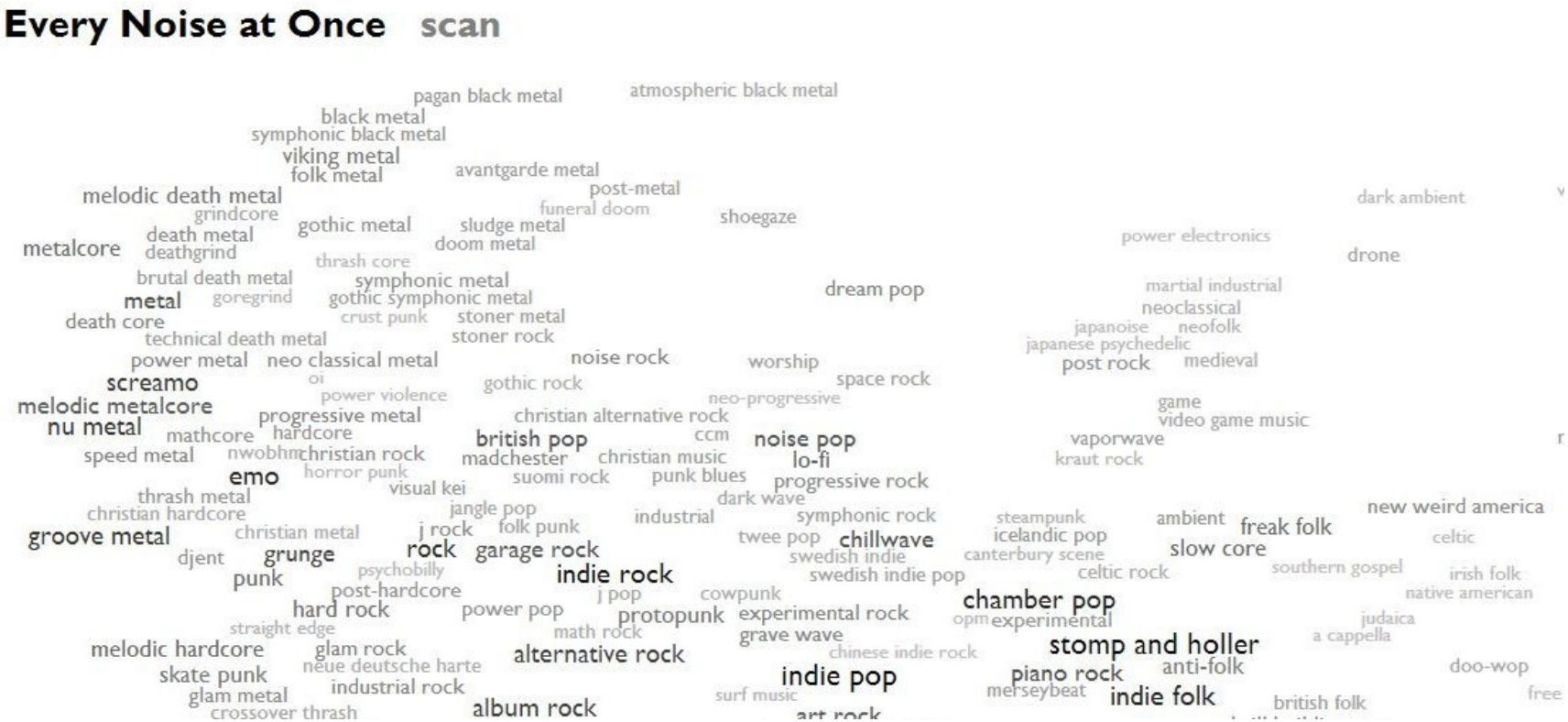


e.g. Given some measurement points about mosquitoes in Asia, how many species are there?

K=4? K=5? K=6 K=8?

It is reasonable to say there is a "true" K!

Which Music Genre's do You Listen to?



(see http://everynoise.com)

Music genre's are constantly developing: Which ones do you listen to? What is the chance that a new genre is seen?

The number of music genre's is forever expanding! Though at a given time, there is a particular number.

How Many Words in the English Language are There?

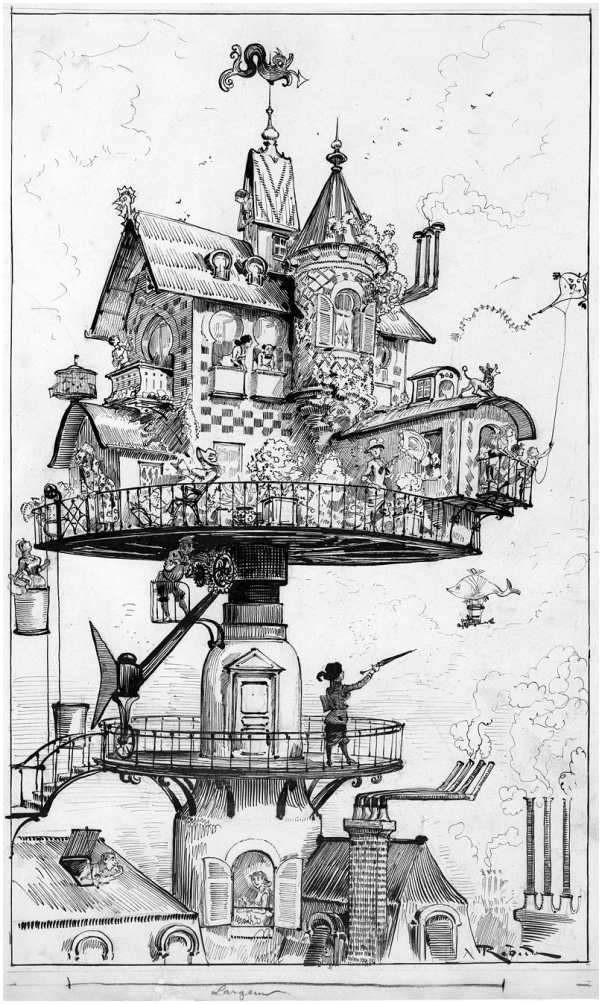
> ... lastly, she pictured to herself how this same little sister of hers would, in the after-time, be herself a grown woman; and how she would keep, through all her riper years, the simple and loving heart of her childhood: and how she would gather about her other little children, and make their eyes bright and eager with many a strange tale, perhaps even with the dream of wonderland of long ago: ...

e.g. Given 10 gigabytes of English text, how many words are there in the English language?

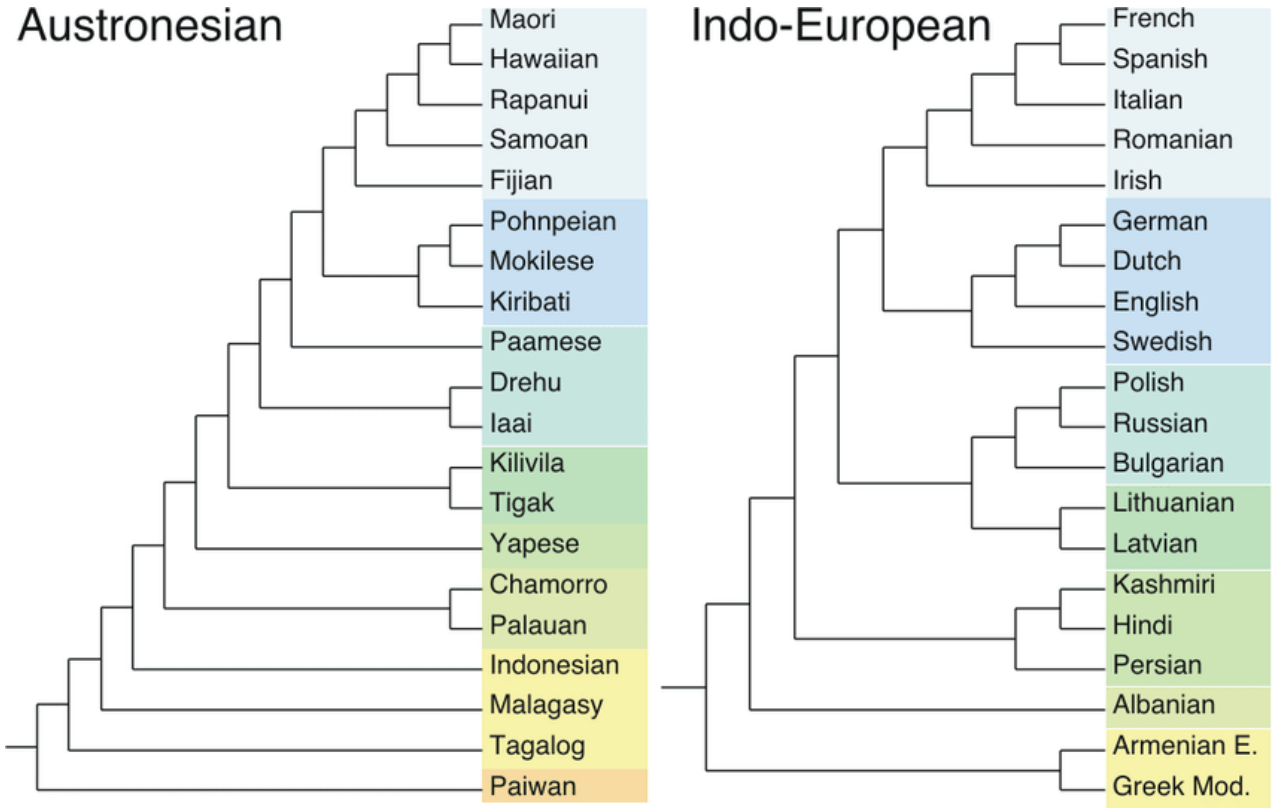
K=1,235,791? K=1,719,765? K=2,983,548?

From experience, we know count of distinct words generally grows logarithmically, or sub-linearly, with size of text.

Outline



Comparative Linguistics: Hierarchical Clustering

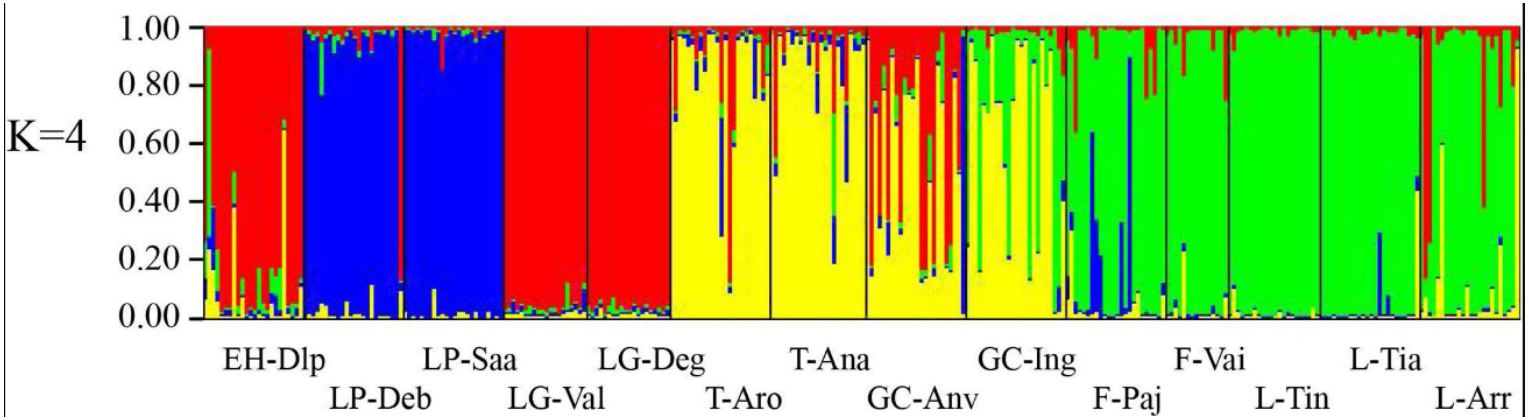


from "The shape and tempo of language evolution", Greenhill, Atkinson, Meade, Gray, *RSB: Biol*, 2010

In linguistics and genetics, one sometimes wants trees, not clusters.

Genetics: "Soft" Clustering

Kleinia neriifolia, a form of mountain grass in the Canary islands

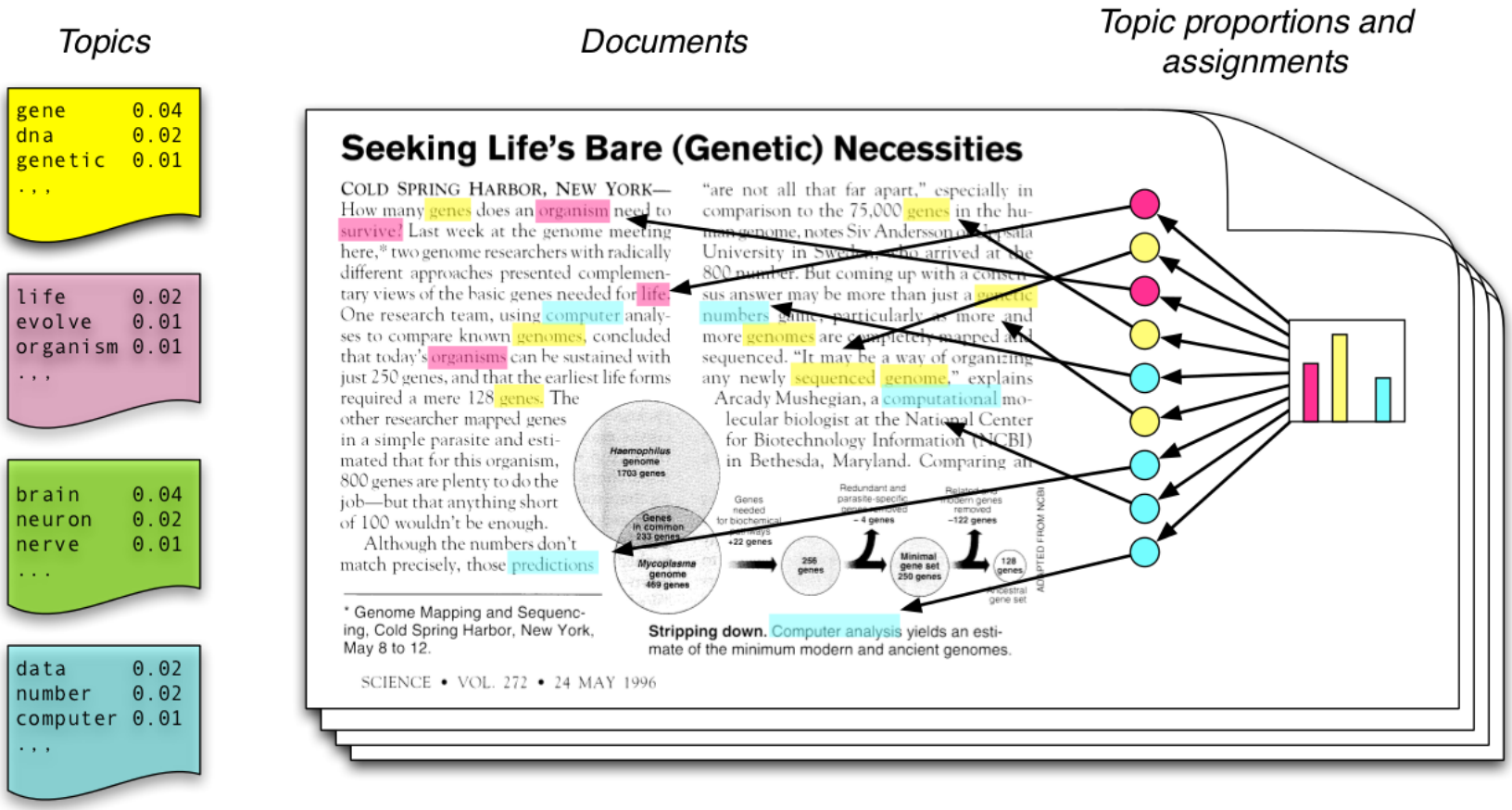


from "Population Structure, Genetic Diversity, and Evolutionary History of Kleinia neriifolia," Sun and Vargas-Mendoza, *Front. Plant Sci.*, 2017

Each vertical line is a single plant. The colors represent proportions of the plant in the soft cluster, according to the genetics. The horizontal layout is roughly the geographical spread.

In genetics and document analysis, clusters may not be "hard".

Documents: "Multi-level" Clustering

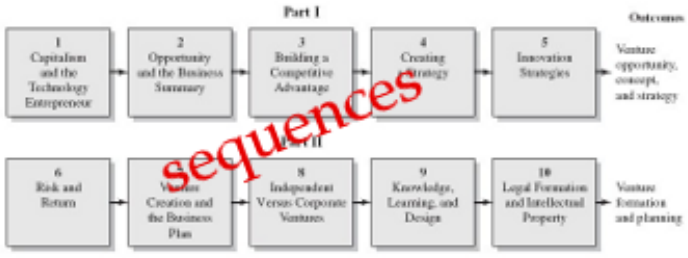


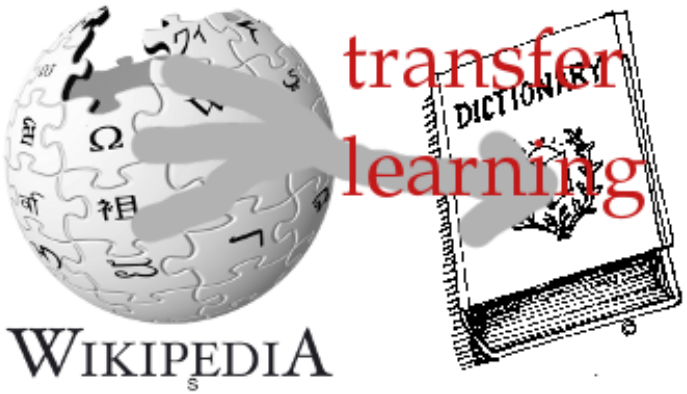
from "Latent Dirichlet Allocation" Blei, Ng, Jordan *JMLR* 2003

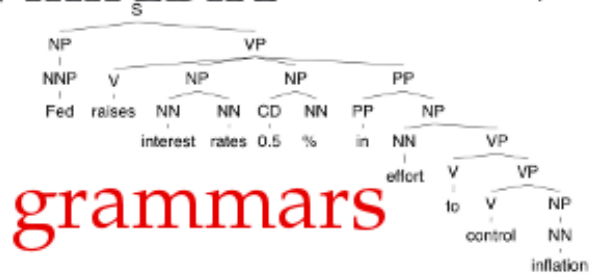
We can cluster the words while being aware of their place in documents. This can be related to soft clustering.

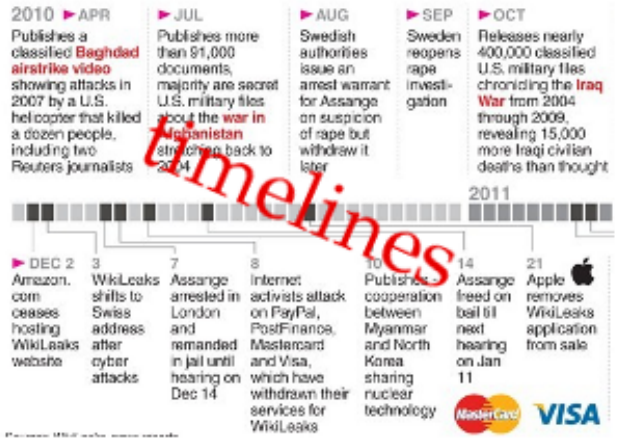
Rich Structures in Data

many opportunities for extensions to clustering or latent variable modelling

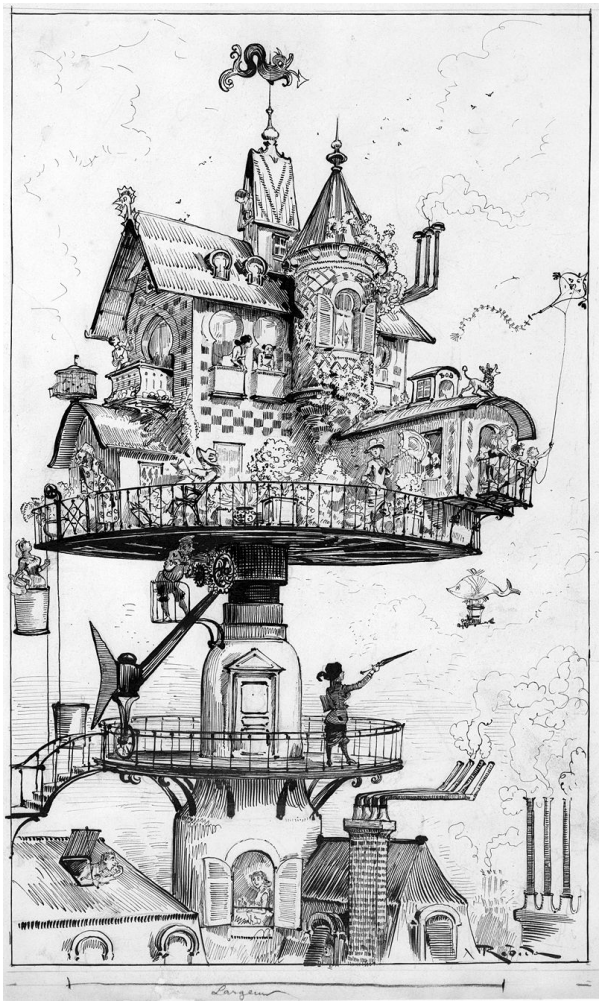








Outline



Matrix Data from Documents

Step 1: bag up your documents to get terms and their counts.

Original news article:

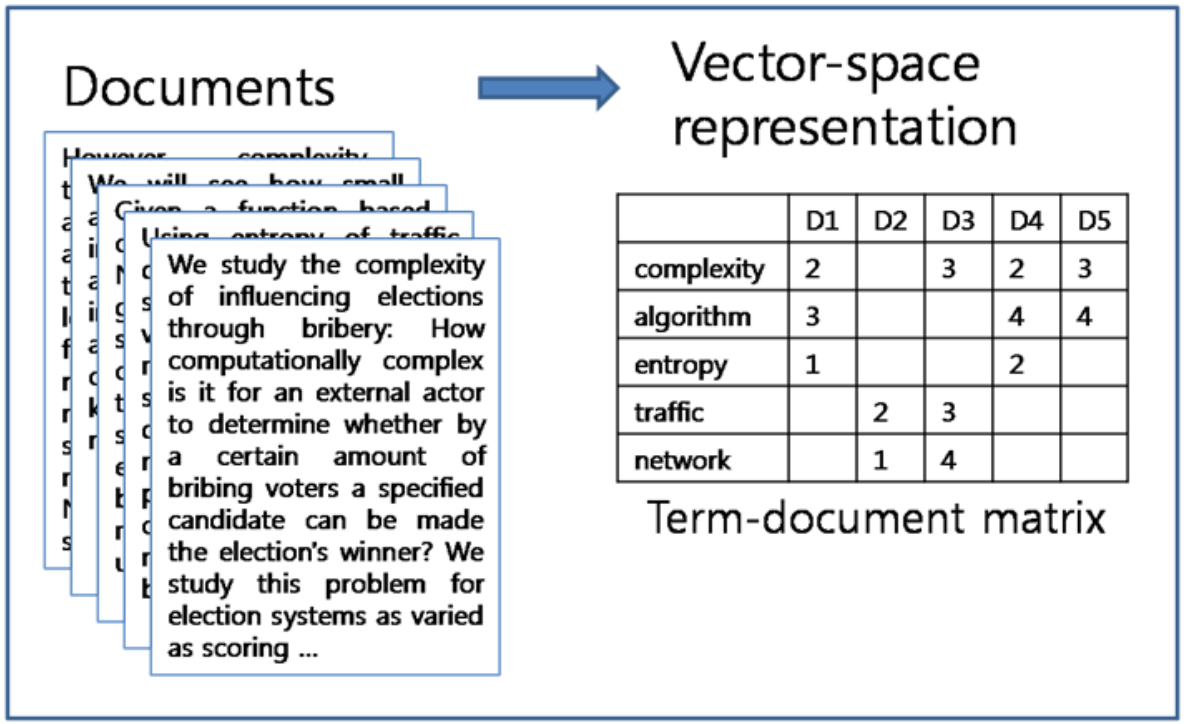
Women may only account for 11% of all Lok-Sabha MPs but they fared better when it came to representation in the Cabinet. Six women were sworn in as senior ministers on Monday, accounting for 25% of the Cabinet. They include Swaraj, Gandhi, Najma, Badal, Uma and Smriti.

Bag of words:
ordered
0-9A-Za-z

11% 25% Badal Cabinet(2) Gandhi Lok-Sabha MPs Monday Najma Six Smriti Swaraj They Uma Women account accounting all and as better but came fared for(2) in(2) include it may ministers of on only representation senior sworn the(2) they to were when women

Matrix Data from Documents, cont.

Step 2: line up all the term-count vectors to produce a document-term matrix.



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Term-document matrix

NB. similarly, tweets, graphs, road traffic, network traffic, gene data, all sorts of data can be turned into matrices!

Matrix Factorisation

- Model document collection, image collection, graph, ..., as a matrix.
- Full data view:

$$\text{(data)....} \quad \boldsymbol{X} \simeq \boldsymbol{\Theta}\boldsymbol{\Phi} \quad \text{....(model)}$$

- Individual row (document, image, node) view, showing i-th item:

$$\text{(data item)....} \quad \boldsymbol{x}_i \simeq \boldsymbol{\theta}_i\boldsymbol{\Phi} \quad \text{....(model)}$$

- Single entry view, showing j-entry of i-th item:

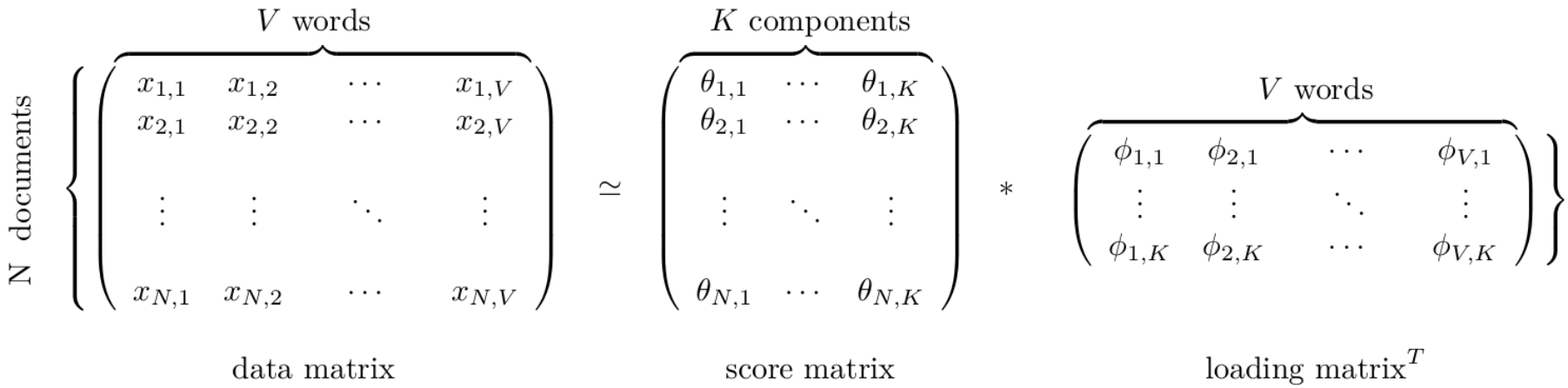
$$\text{(data entry)....} \quad x_{i,j} \simeq \boldsymbol{\theta}_i\boldsymbol{\phi}_{\cdot,j}^T \quad \text{....(model)}$$

- Variables are:

  $\boldsymbol{x}_i$: single datum though some entries can be missing
  $\boldsymbol{\theta}_i$: latent variable, used to generate $\boldsymbol{x}_i$
  $\boldsymbol{\Phi}$: model parameters

Matrix Factorisation, cont.

(data) .... $\boldsymbol{X} \simeq \boldsymbol{\Theta\Phi}$ .... (model)



Data $\boldsymbol{X}$	Components $\boldsymbol{\Theta}$	Error	Models
real valued	unconstrained	least squares	PCA and LSA
non-negative	non-negative	least squares	NMF, learning codebooks
non-neg int.	rates	cross-entropy	Poisson & Neg.Bino. MF
non-neg int.*	probabilities	cross-entropy	topic models
real valued	independent	small	ICA
non-neg int.	scores	shifted PMI	GloVe

Matrix Factorisation Terminology

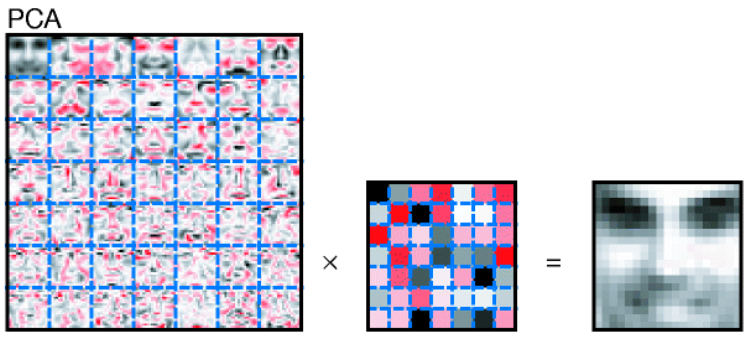
Statistics: "components"

Classical ML: "topics"

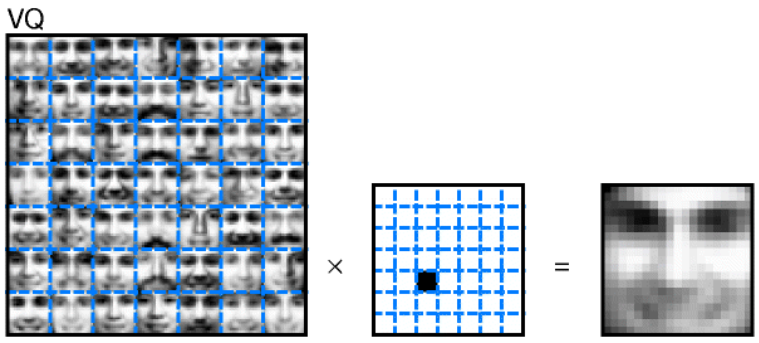
Deep NNs: "embeddings"

Variants for Images

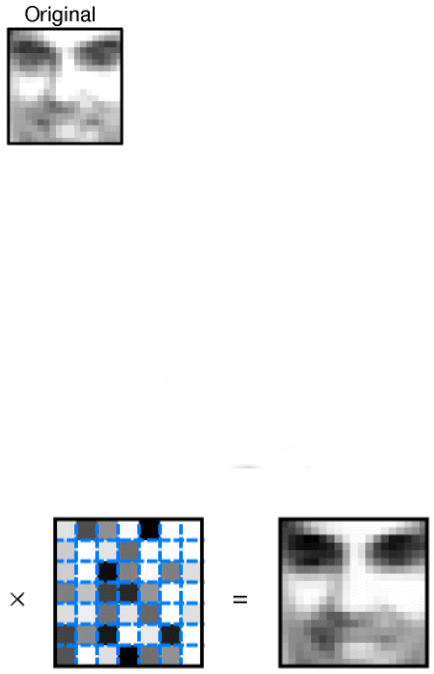
from Lee & Seung, *Nature* 1999



Principal Components Analysis



Vector Quantization



Non-negative Matrix Factorisation

Image/Vision community has moved away from matrix factorisation methods to deep neural networks.

$\Longrightarrow$ The data for images are not in "semi-semantic" tokens.

Modelling Text with Matrix Factorisation

image →

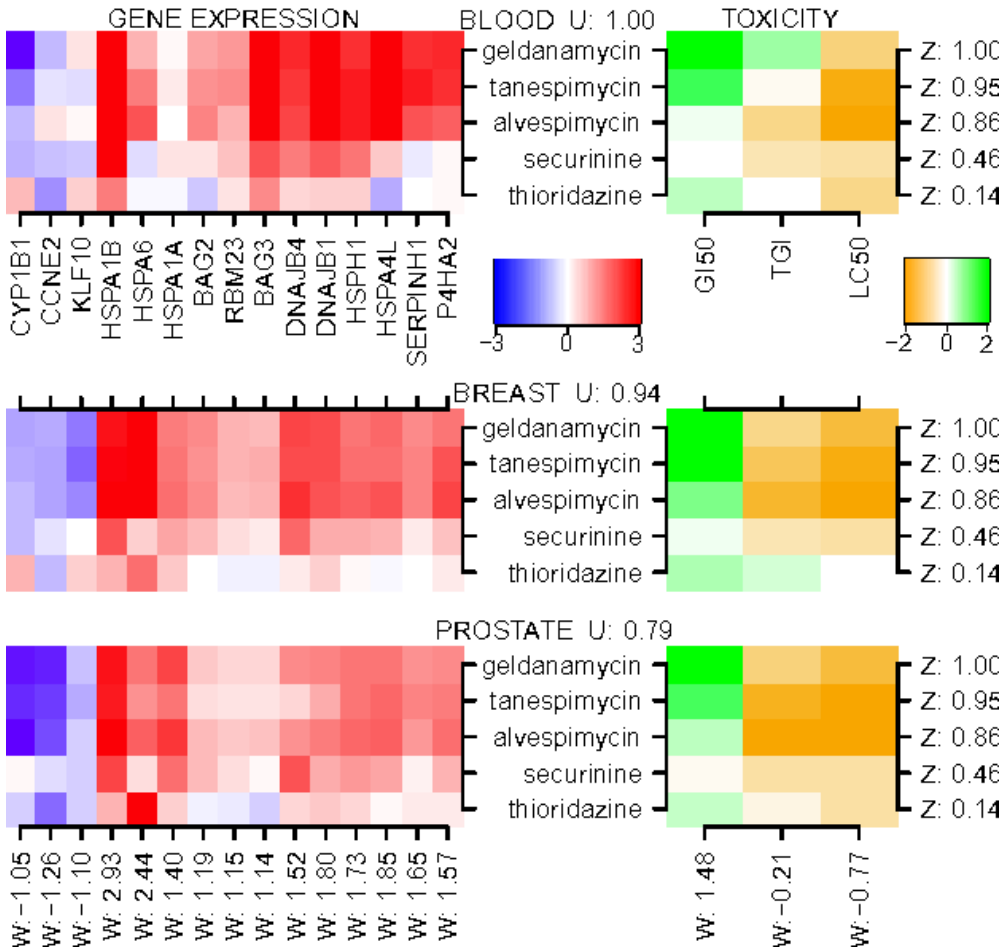
Prince, Queen, Elizabeth, title, son, ...	school, student, college, education, year, ...
John, David, Michael, Scott, Paul, ...	and, or, to , from, with, in, out, ...

text →

13 1995 accompany and(2) andrew at boys(2) charles close college day despite diana dr eton first for gayley harry here housemaster looking old on on school separation sept stayed the their(2) they to william(2) with year

Approximate faces/bag-of-words (RHS) with a linear combination of components (LHS).

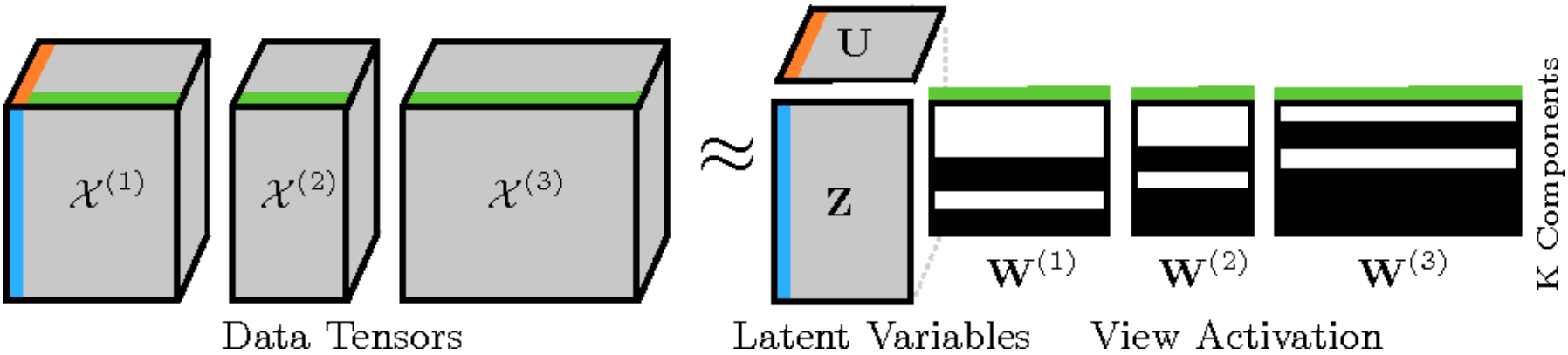
Multi-view Tensor Factorization



from "Bayesian Multi-view Tensor Factorization" Kahn and Kaski, *ECML/PKDD* 2014

We can work with multiple matrices (with common sides) at once.

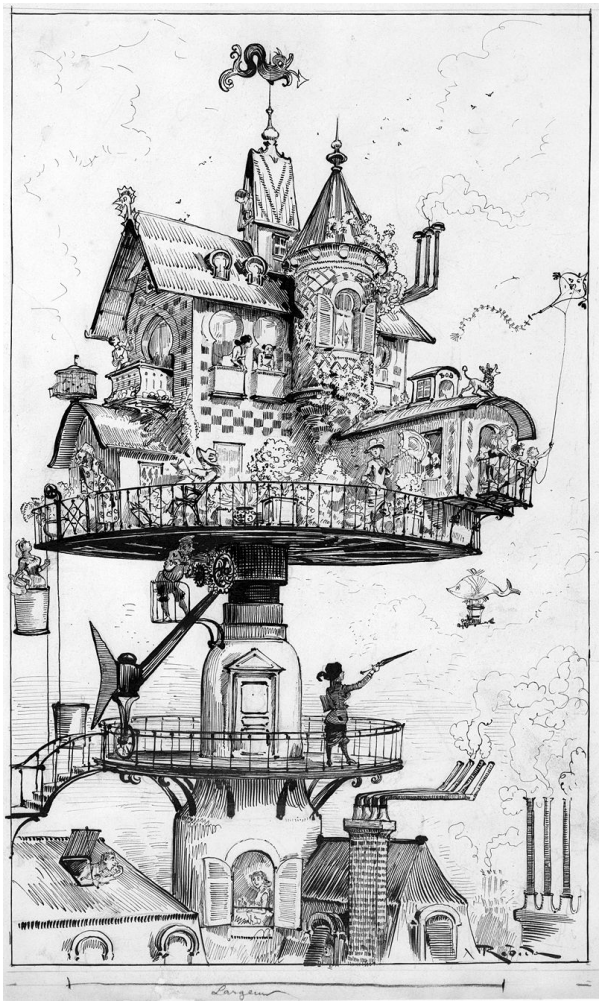
Multi-view Tensor Factorization, cont.



from "Bayesian Multi-view Tensor Factorization" Kahn and Kaski, *ECML/PKDD* 2014

- Matrices are the simplest case of multi-view tensors and heterogenous graphs.
- The methods become more complex (e.g., Tucker decompositions) but share many similarities.

Outline



Examples of Unsupervised Models

- Generative Adversarial Networks (GANs) and Variational Auto-encoders (VAEs)
- Dimensionality reduction
  - some covered with matrix factorisation
- Unsupervised anomaly detection
- Semi-supervised learning
- Generative models
- Segmentation
- Unsupervised natural language learning
- Record linkage, using blocking

These can be very different tasks, and require their own custom methods.

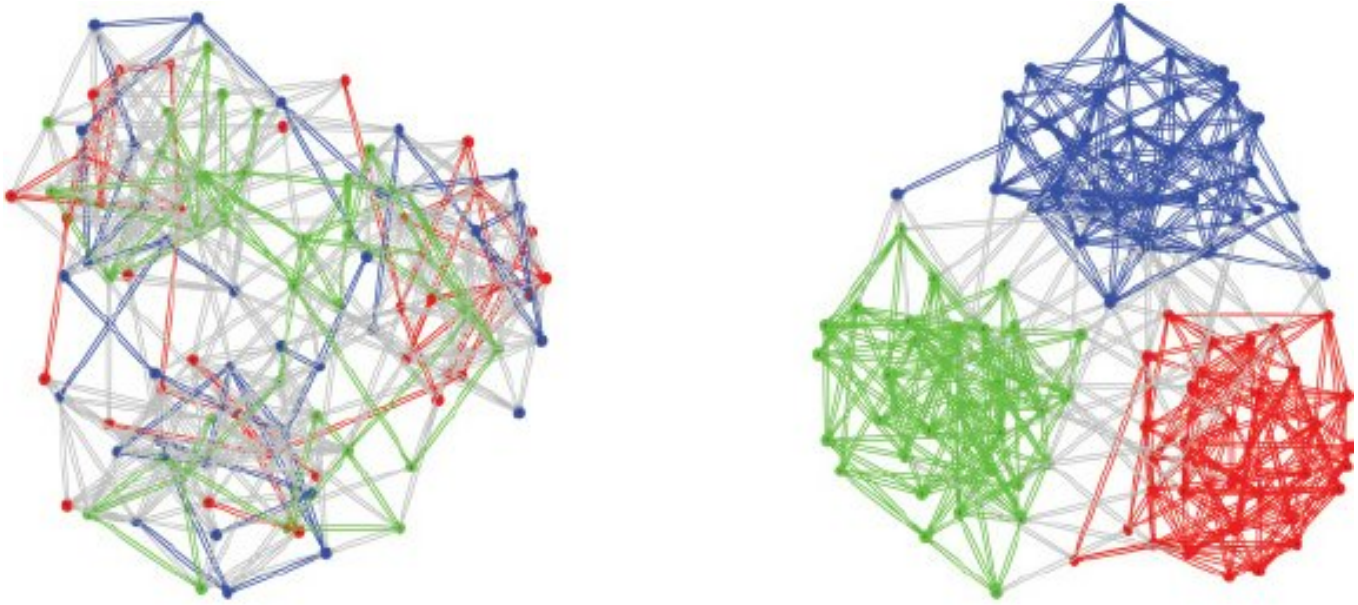
Image Segmentation

from

`https://en.wikipedia.org/wiki/Image_segmentation`

- cluster pixel colours into 16 clusters
- replace each pixel by its centroid
- very simple and naive clustering!

Graph Segmentation



(a) A poor edge-cut partitioning. Vertices are assigned to partitions at random, thus, there are many inter-partition links.

(b) A good edge-cut of the same graph, where vertices that are highly connected are assigned to the same partition.

from "A Distributed Algorithm for Large-Scale Graph Partitioning," Rahimian, Payberah and Girdzijauskas, *ACM Trans. AAS*, 2015

- *graph partitioning* "clusters" nodes of a graph
- source of well known NP-complete problems in CS theory
- initially developed for Design Automation (silicon chips) in Electrical Engineering

The Model Behind VAEs

- Matrix factorisation with noise:

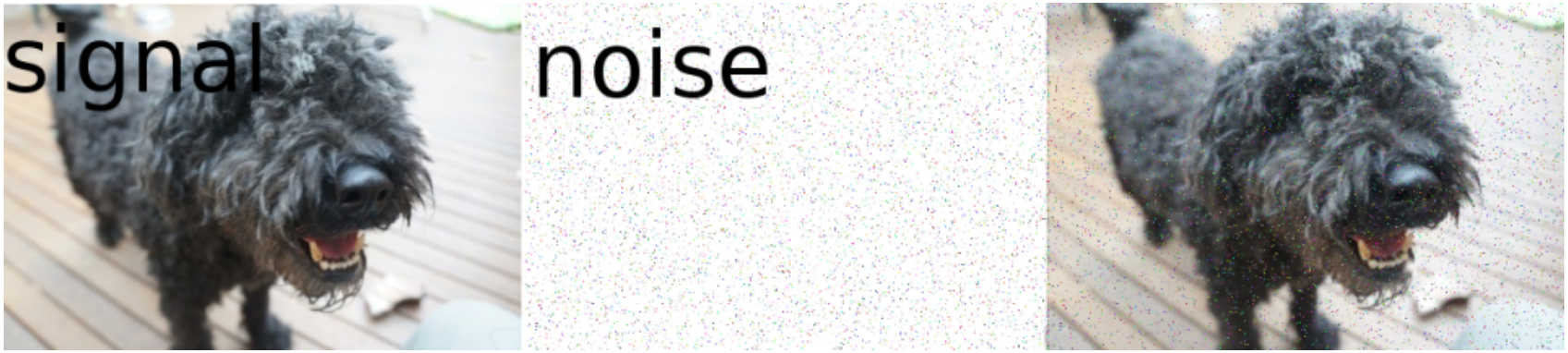
$$\boldsymbol{x}_i = \boldsymbol{\theta}_i \boldsymbol{\Phi} + \boldsymbol{\epsilon}_i \qquad \text{.....(row of matrix factorisation)}$$

- VAE model, and $\boldsymbol{\theta}_i$ is a latent Gaussian vector:

$$\boldsymbol{x}_i = f(\boldsymbol{\theta}_i, \boldsymbol{\Phi}) \qquad \text{.....(deep net } f(,) \text{ with weights } \boldsymbol{\Phi})$$

- Works well in image, speech and similar signal processing (which has no "noise" in the classic sense).

Noise Models for VAEs

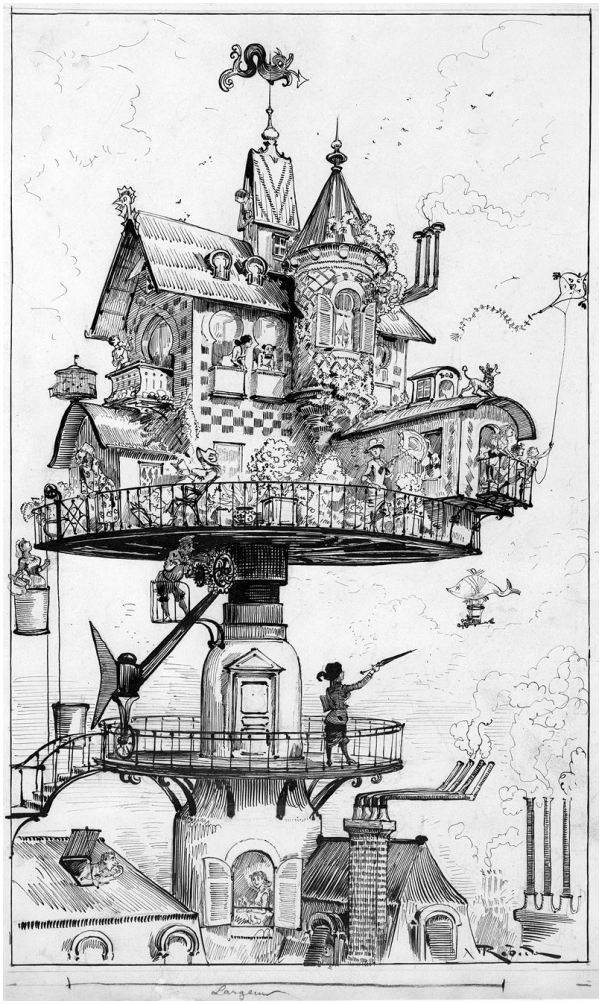


- Typical images don't have noise in the usual sense. They have semantic-level noise:
  - different locations for trees
  - different detail in grass/hair/waves/...
- For data with semi-semantic tokens, e.g., VAE-style topic modelling of documents, people can add a noise term

$$\boldsymbol{x}_i = f(\boldsymbol{\theta}_i, \boldsymbol{\Phi}) + \boldsymbol{\epsilon}_i$$

e.g., use a negative binomial to sample an integer value.

Outline



Typical Uses of Clustering

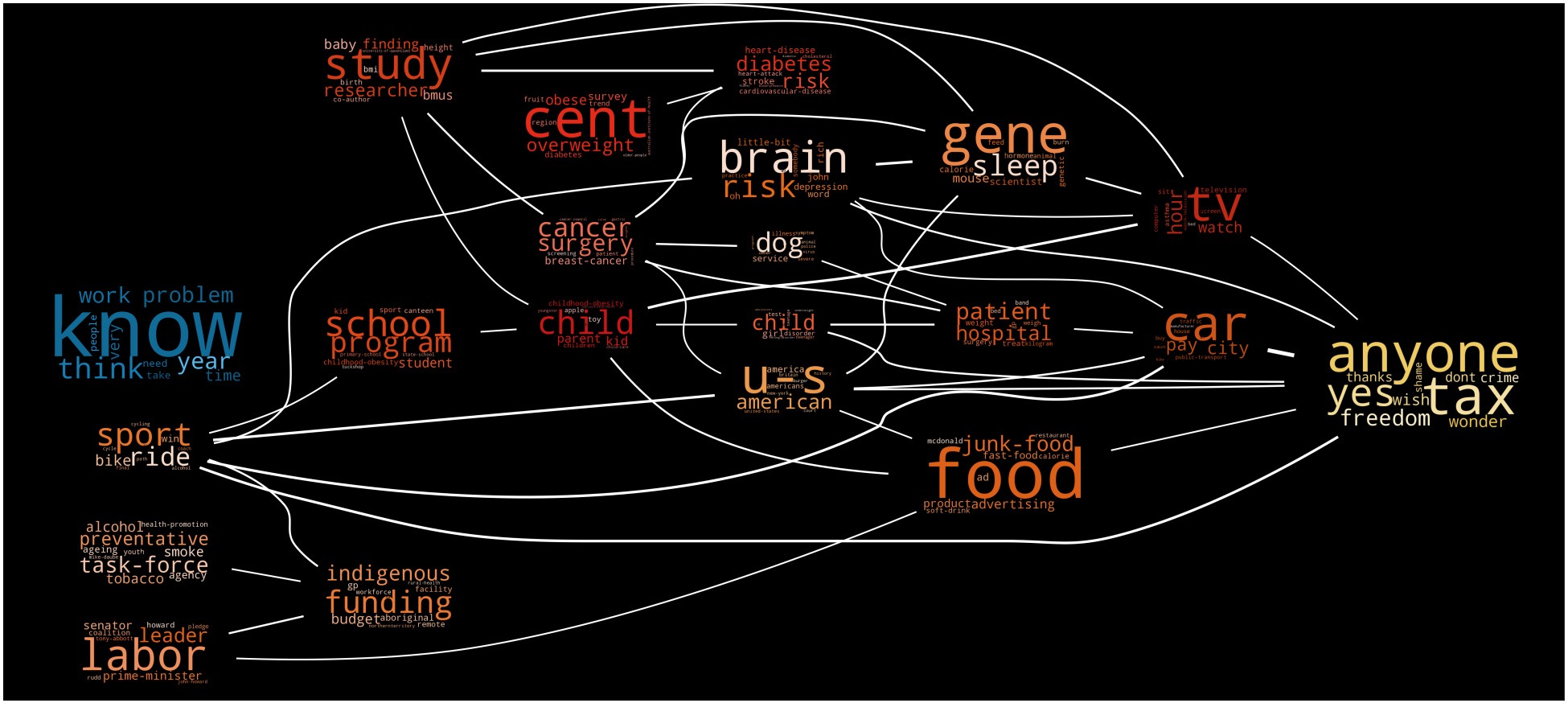
- discovery tool in general data science tasks
- generative model for tasks like anomaly detection
- data exploration/browsing
  - there is a long history of using clustering to help present results of Information Retrieval
- market segmentation

  i.e., bucketing users/clients for group actions
  - done for targetted advertising, to groups rather than individuals
- See more at *Wikipedia*.

Typical Uses of Matrix Factorisation

- dimensionality reduction, reducing much larger matrices in size
- summarisation of relational content: similar role of clustering
- recommender systems
- bioinformatics, analysis of matrix/vector gene data

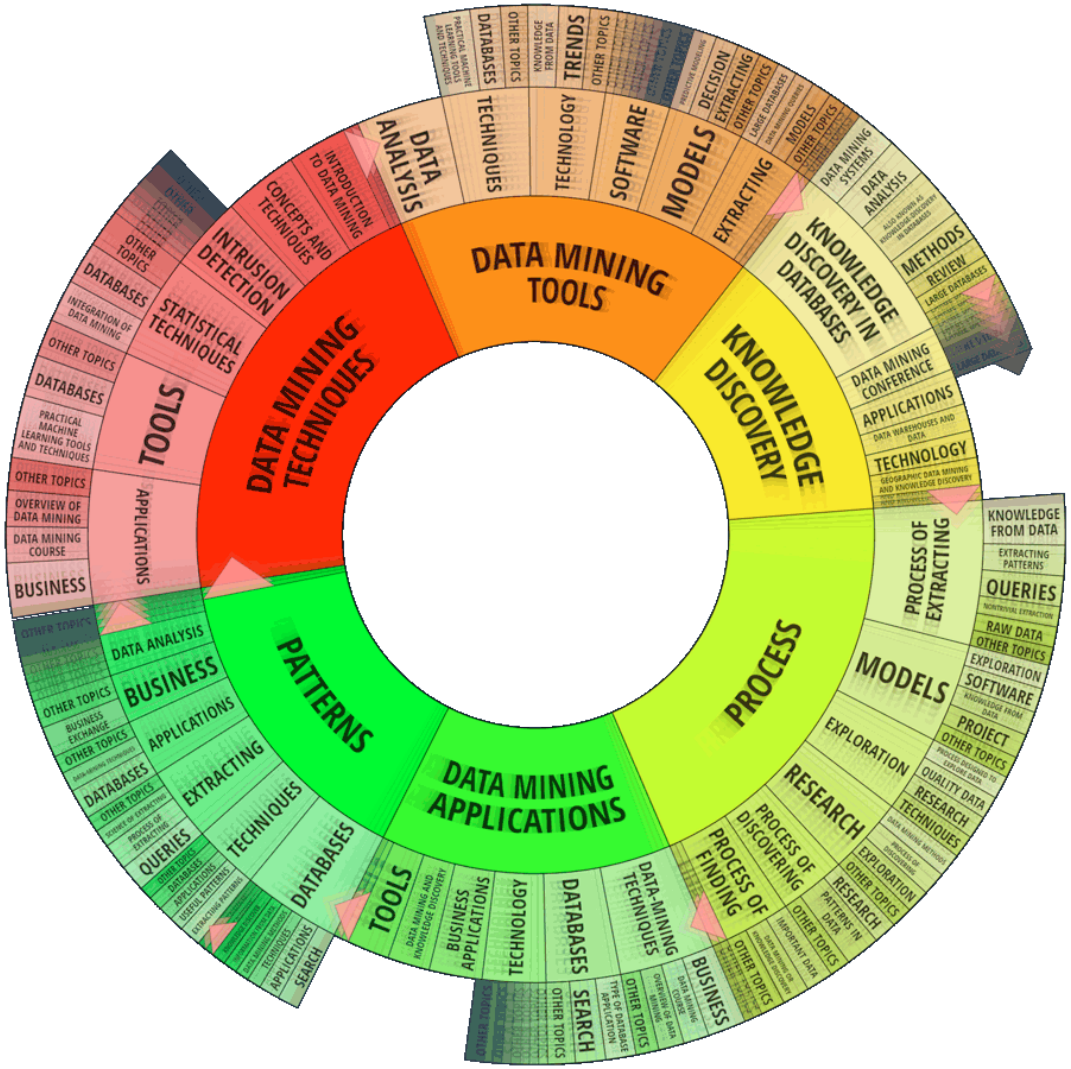
Summarising 380 News Articles on Obesity



- blue words: the background topic have non-topical words, is a prior for all other topics!
- word size: is how characteristic they are **for the topic**
- whiter words: more frequent **in the full collection**

Carrot^2 Circles

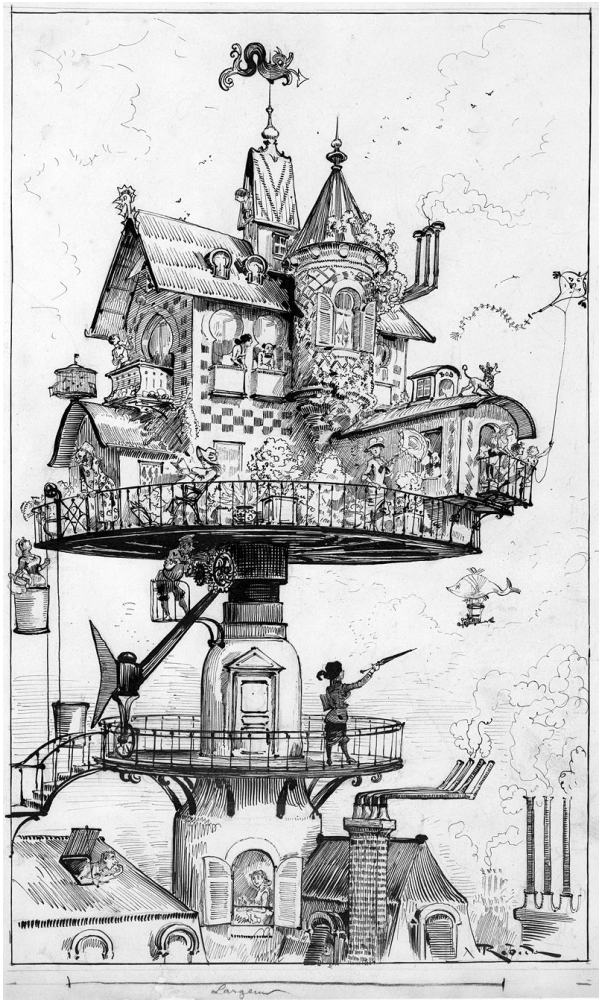
summarising document collections with hierarchical clustering



from `https://carrotsearch.com/circles/`

summary of results for query "data mining"

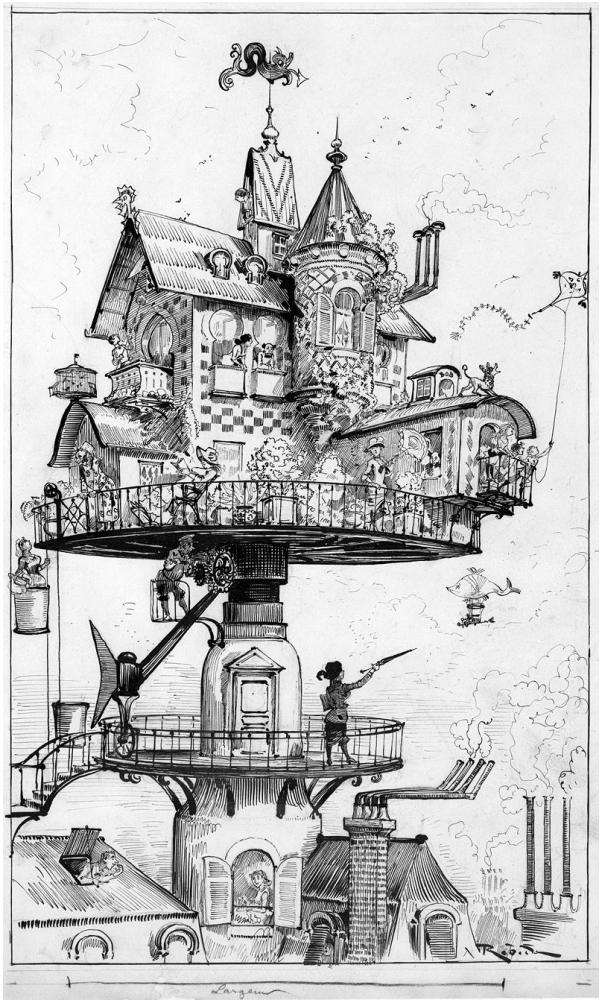
Outline



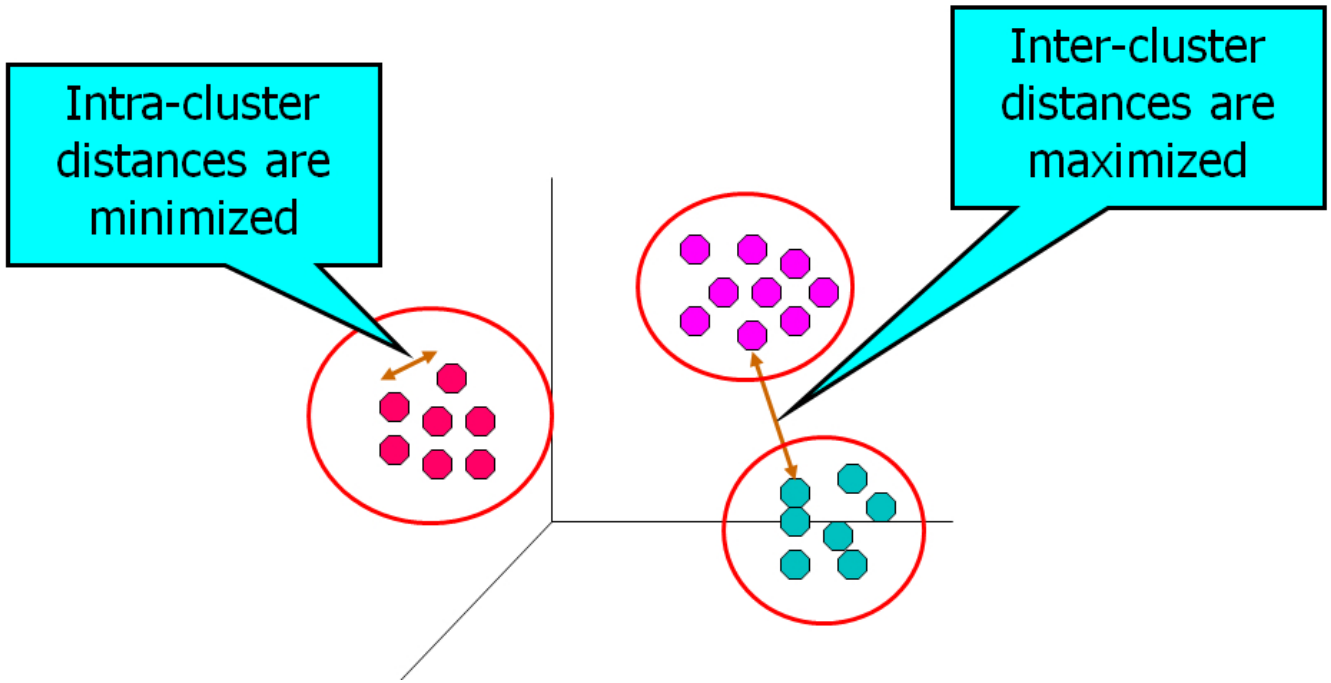
Historical Context

Early clustering algorithms like **k-means** were developed for these guys

Outline



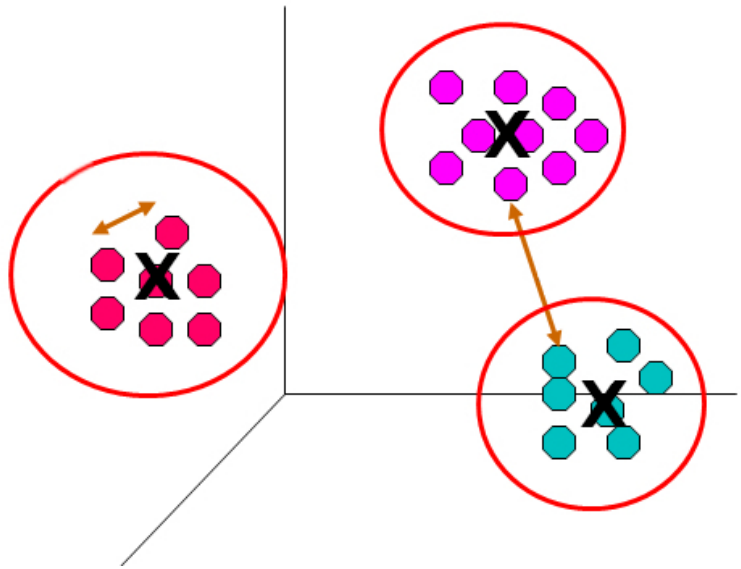
Distance-based Models



from https://apandre.wordpress.com/visible-data/cluster-analysis/

- set up a cost function to maximise inter-cluster distances and minimise intra-cluster distances
- in general case need to compute N^2 distances!

Distance-based Models



modified from https://apandre.wordpress.com/visible-data/cluster-analysis/

- k-means was developed to make this $O(KN)$ instead
- give each cluster a centroid (marked with "X")
- change the cost function to measure distance between data and centroid

K-means Definition

from *Wikipedia*, LaTeX and all!

K-means Clustering

Given a set of observations $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, where each observation is a d-dimensional, **k-means clustering** aims to partition the N observations into K ($\leq N$) sets $S = \{S_1, S_2, \ldots, S_K\}$ so as to minimize the within-cluster sum of squares (WCSS, i.e. variance).

Formally, the objective is to find:

$$\text{WCSS} = \arg\min_{\mathbf{S}} \sum_{k=1}^{K} \sum_{\mathbf{x} \in S_k} |\mathbf{x} - \boldsymbol{\mu}_k|^2 = \arg\min_{\mathbf{S}} \sum_{k=1}^{K} |S_k| \, \text{Var}(S_k)$$

where $\boldsymbol{\mu}_k$ is the mean (also called a **centroid**) of points in S_k.

- defined as finding a *partition* of the data
- algorithm recomputes the $\boldsymbol{\mu}_k$ then S iteratively

K-means is Double-Plus Good!

These guys are telling you k-means is the goto method for clustering

Towards Data Science | DATA SCIENCE | MACHINE LEARNING | PROGRAMMING | VISUALIZATION | AI | JOURNALISM | PICKS

Data Science in the Real World

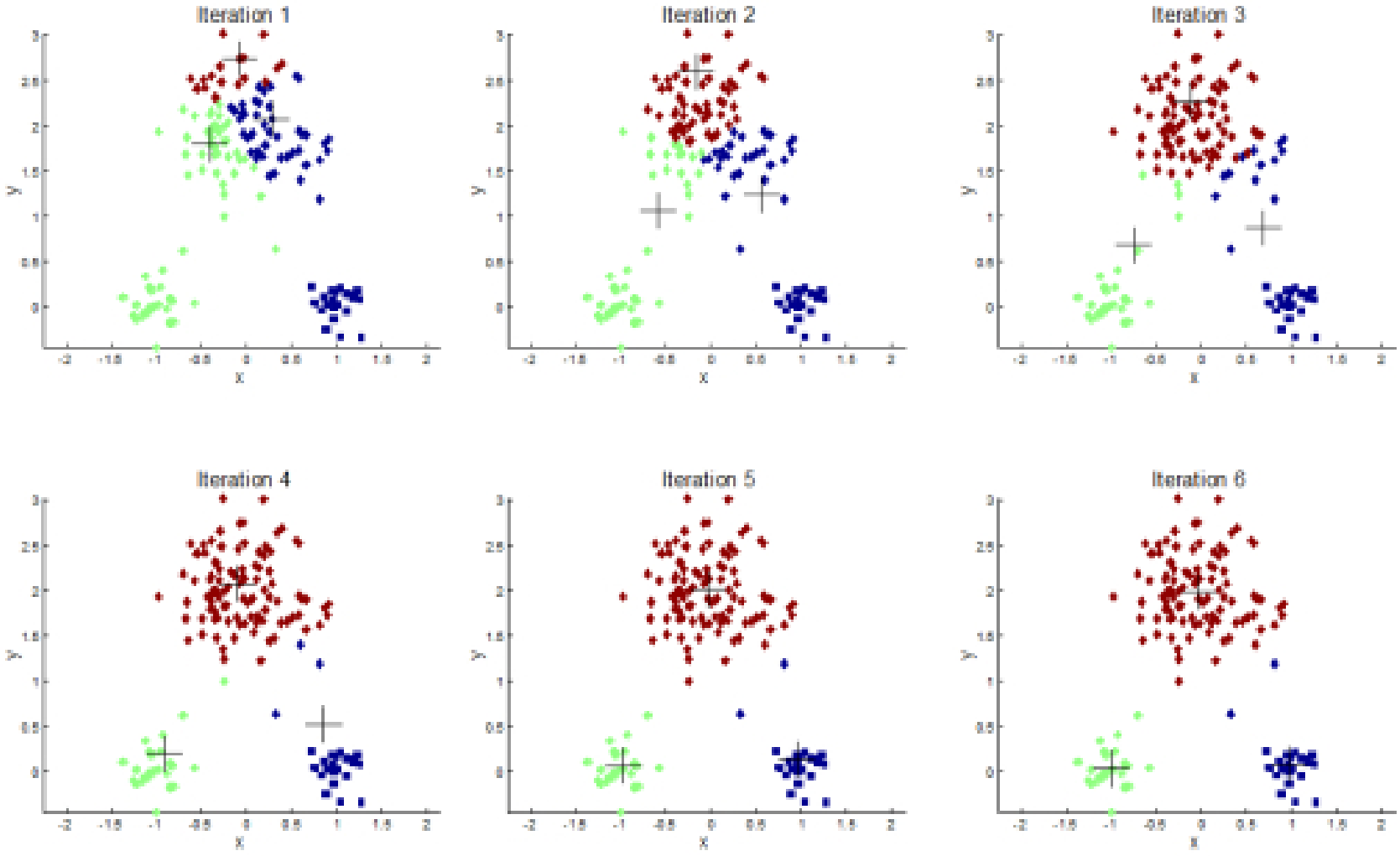
10 Machine Learning Methods that Every Data Scientist Should Know

Jump-start your data science skills

Jorge Castañón, Ph.D. Follow

May 2 · 15 min read

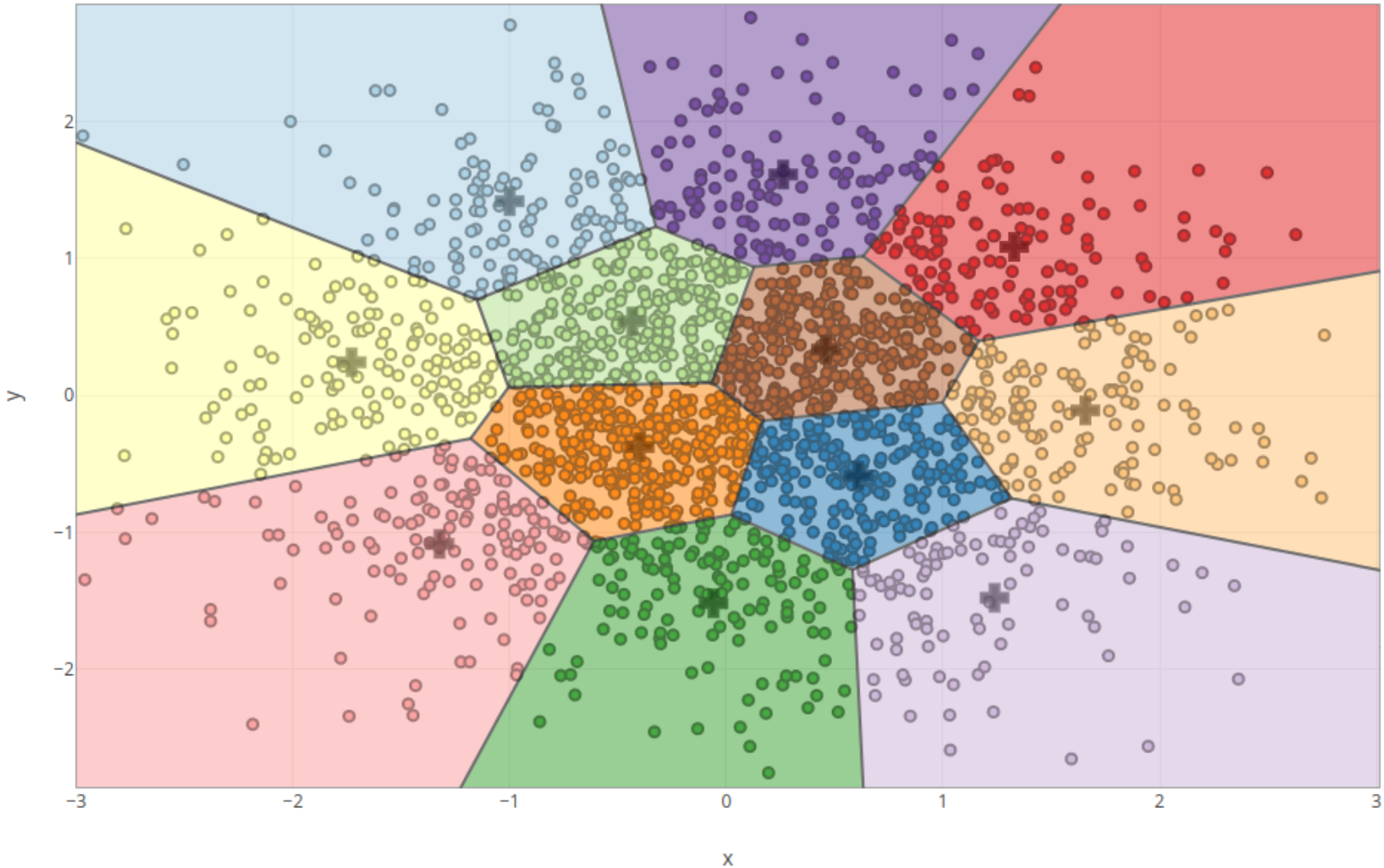
K-means Algorithm



from `https://apandre.wordpress.com/visible-data/cluster-analysis/`

- an iterative algorithm, "repeat until no change"

K-means and Voronoi Polygons



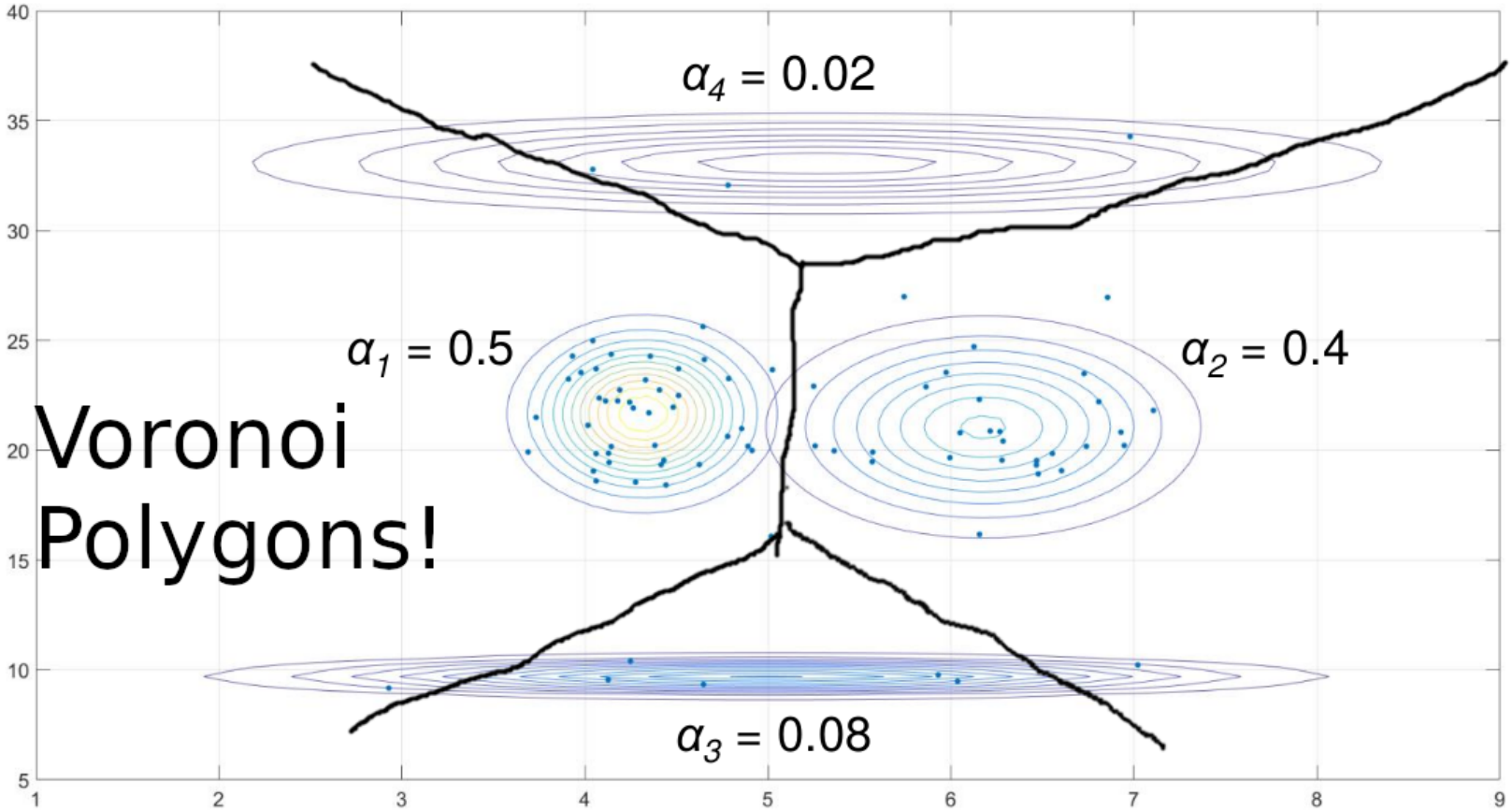
from `https://plot.ly/ MariaKu/95.embed`

K-means Computation

Some reflection on definition shows it is a greedy local search algorithm (i.e., one dimension at a time) on the space of centroids and assignments:

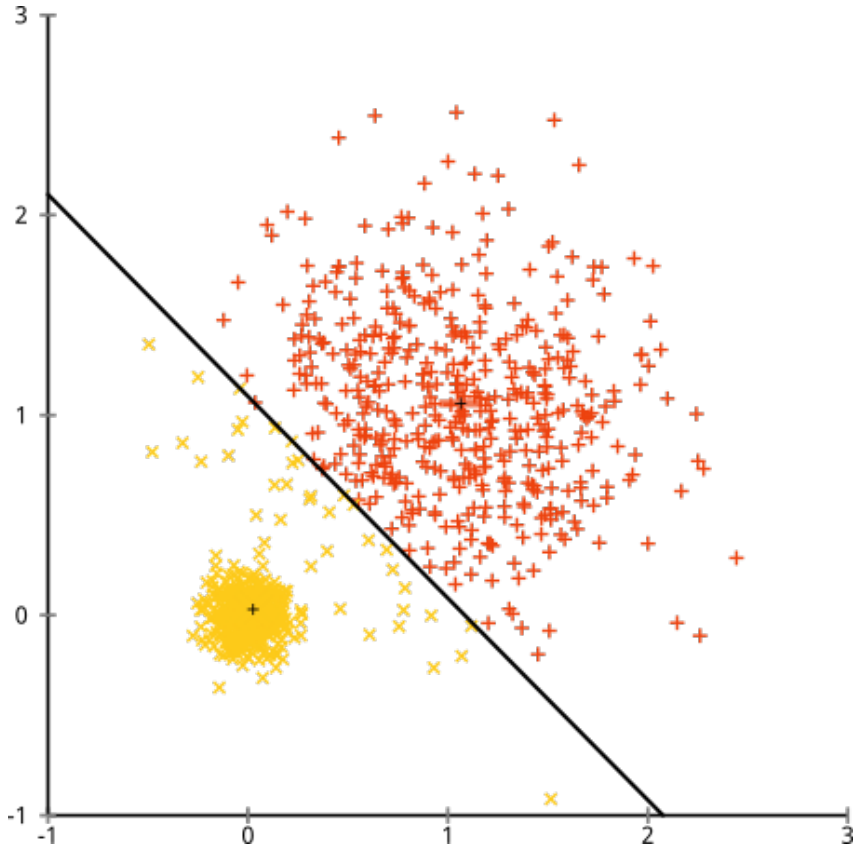
- each step prior to convergence decreases the WCSS
- on convergence the space is partitioned into *Voronoi polygons* with the centroids as the points
- and like **all** complex search problems
  - there is no global (best) solution
  - is sensitive to initialisation

K-means on Mixture Models



- some data will be assigned to the wrong cluster, causing the centroid estimates to be biased!
- cluster 1 gains some outliers from clusters 3 & 4!

K-means: Problems with Variance



see *"Understand drawbacks of K-means"* on Stackexchange

- Both clusters have 500 points but yellow has smaller variance.
- The Voronoi polygons thus don't match: some points wrongly assigned, so centroids will be biased away from the midline.

K-means Issues

- user needs to define distances and means (of points, for centroids)
  - how would you do this for text data?
- user needs to specify K
  - use the so-called *elbow method*
- specialised data structures and fast algorithms exist
  - fabulous topic for computer scientists
  - found in most scalable data science packages
  - you can implement it natively in some SQLs!
- has a simple probabilistic interpretation (we show later)
  - that proves it can yield *biased estimates*
- has no notion of cluster proportions
- suffers with different variance or non-spherical clusters

Basic Hierarchical Algorithms

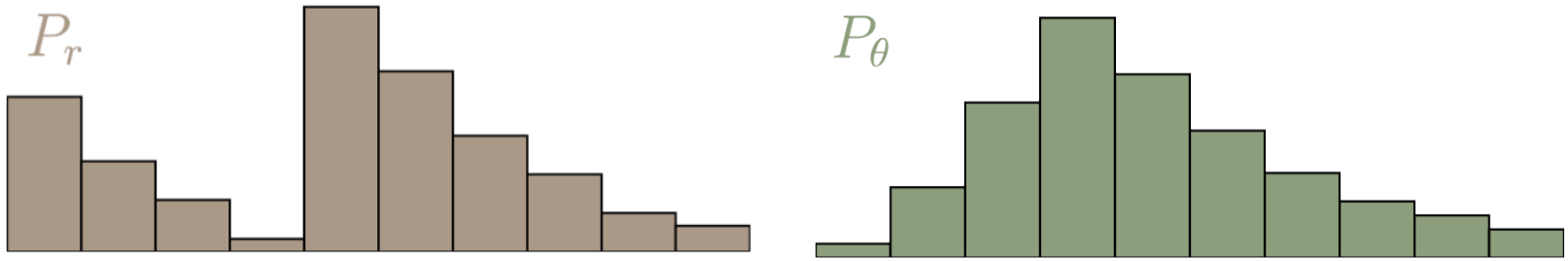
- also called *connectivity based*

  AKA numerical taxonomy

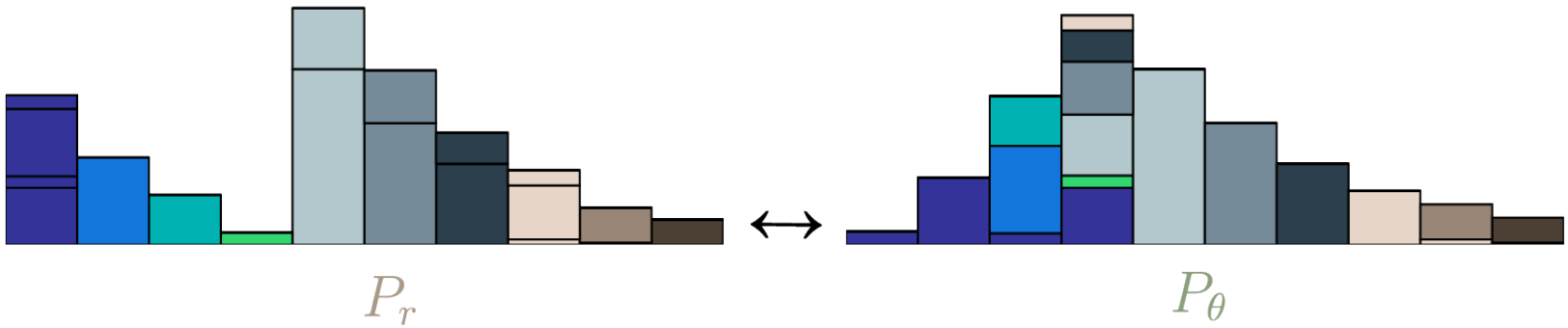
  e.g. the hierarchies in linguistics, biology and genetics
- modify the K-means cost function
- greedily build a hierarchy by either splitting or agglomerating parts of the existing partition
- no K needed!
- again, brought to you by

Wasserstein Distance

from blog of Vincent Herrmann, "Wasserstein GAN", 2017



What is the distance between these two distributions?



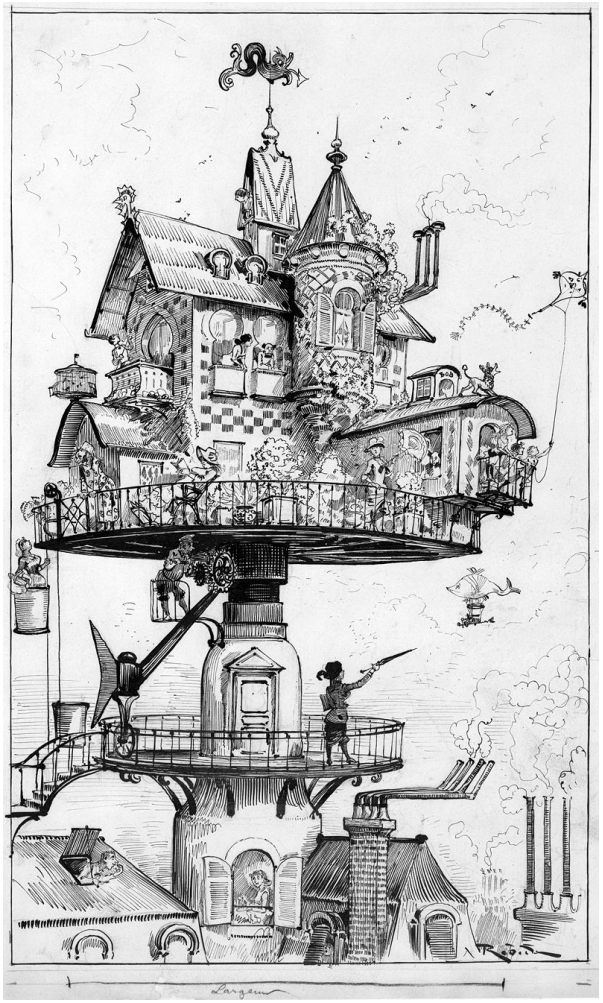
Wasserstein "earth movers" distance: whats the minimum total distance travelled to lug probability from one to the other?

Wasserstein Distance, cont.

- for discrete distributions it is a "simple" optimisation problem
- for real distributions, complex and expensive to compute
- a very intuitive distance and works exceptionally well in practice for clustering
- suits hierarchical and multi-level clustering well

NB. earlier, theoreticians also tried *Bregman divergence*, but these didn't prove as good

Outline



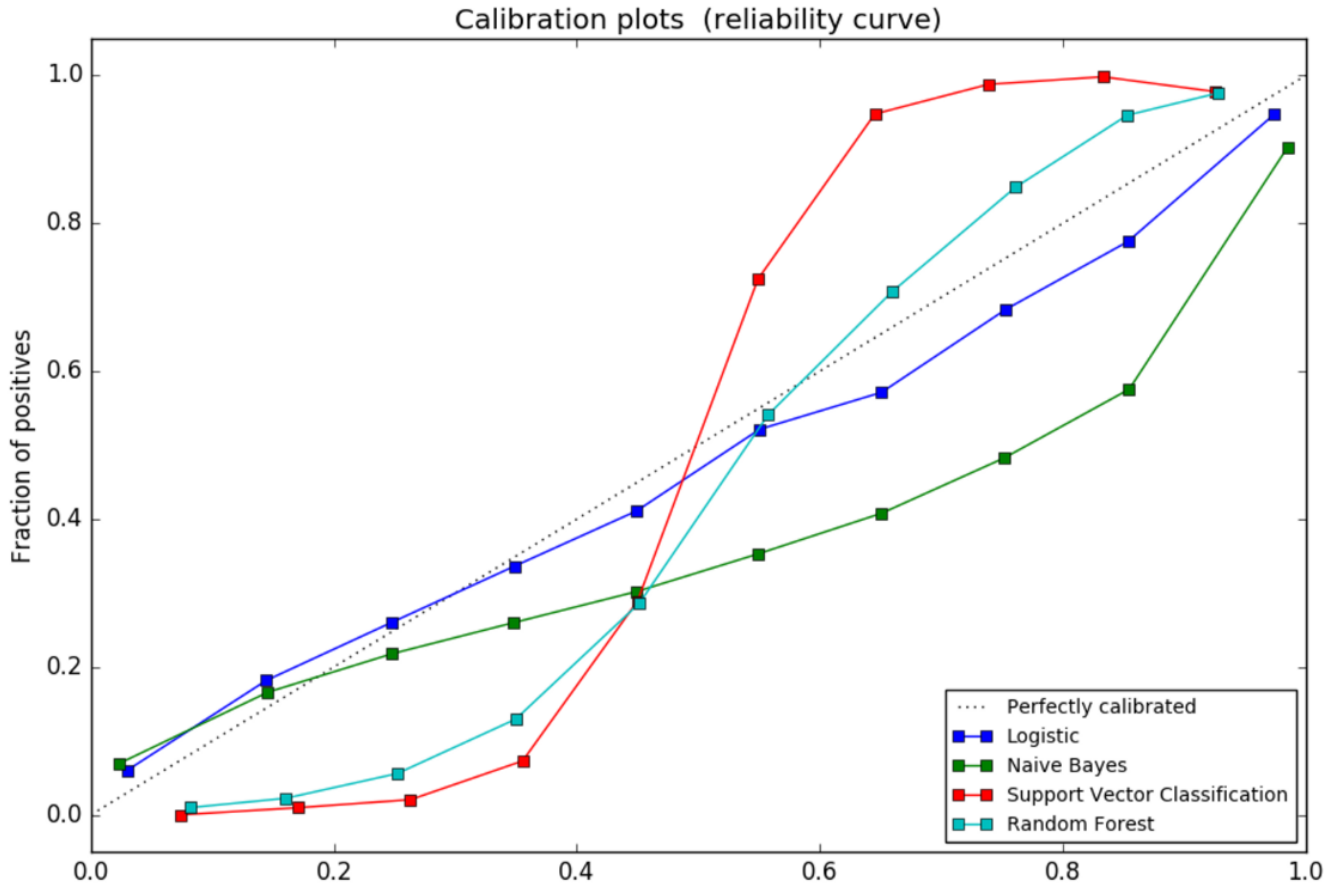
Generative Models

- Give probabilistic prescription for generating data:
  Gaussian mixture model: to generate vector $\boldsymbol{x}_i$:
  1. generate class c_i according to probability vector $\boldsymbol{\alpha}$,
  2. generate data $\boldsymbol{x}_i$ according to Gaussian with parameters $\boldsymbol{\theta}_{c_i}$

  Matrix factorisation: to generate new row $\boldsymbol{x}_i$
  1. generate latent component for a row $\boldsymbol{\theta}_i$
  2. compute mean vector $\boldsymbol{\theta}_i \boldsymbol{\Phi}$ from parameters $\boldsymbol{\Phi}$
  3. add noise, to get row $\boldsymbol{x}_i = \boldsymbol{\theta}_i \boldsymbol{\Phi} + \boldsymbol{\epsilon}_i$
- Are explicitly/implicitly probability models that generates data according to proposed model probabilities (frequencies).
- Are inherently testable:
  - you can compare observed frequencies in data to probabilities predicted by the model, via the likelihood
  - basis of the maximum likelihood principle

Calibrating Models

- Good probability models should be *well calibrated*.
- probabilities predicted by the model should be close to observed frequencies in data
- Example shown for Boolean classification error:



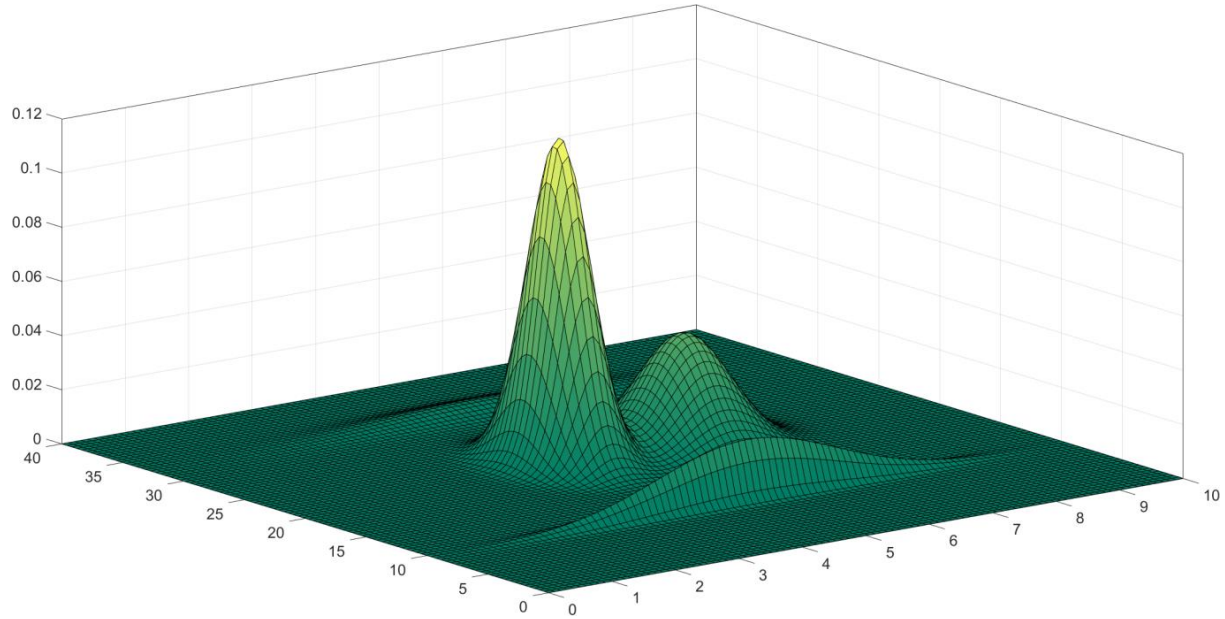
from `https://scikit-learn.org/stable/modules/calibration.html`

Calibration of Models

- A calibrated model deals well with many non-standard prediction problems, e.g.,
  - imbalanced classes
  - unequal costs
- If you train your classification model using a cost function that is a *proper scoring rule*, then it will give you a calibrated model.
  - log probability and *Brier score* are proper scoring rules
  - prediction errors is not a proper scoring rule
- In the calibration plot previously:
  - only Logistic is trained with a proper scoring function
  - naive Bayes trained on log probability for generative model, not a proper scoring function for predictive model

Gaussian Mixture Models (GMMs)

- Plot of a Gaussian mixture model density



- As a mixture model:

$$p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)$$

- Equivalently, as a latent variable model:

$$p(\boldsymbol{x}_i, c_i|\boldsymbol{\alpha}, \boldsymbol{\Theta}) = \alpha_{c_i} p(\boldsymbol{x}_i|\boldsymbol{\theta}_{c_i})$$

GMMs: Algorithms

Parameters are proportions $\boldsymbol{\alpha}$ and Gaussian parameters $\boldsymbol{\theta}_k$ for each cluster k. Optionally may include latents $\boldsymbol{c}$.

- greedy local search for $\boldsymbol{c}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ to minimise

$$\textbf{cost} = \sum_i \log \frac{1}{p(\boldsymbol{x}_i, c_i | \boldsymbol{\alpha}, \boldsymbol{\Theta})}$$

- Gibbs sampling on $\boldsymbol{c}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ from

$$\textbf{probability} = p(\boldsymbol{\alpha}) p(\boldsymbol{\Theta}) \prod_i p(\boldsymbol{x}_i, c_i | \boldsymbol{\alpha}, \boldsymbol{\Theta})$$

- gradient-based search for $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ to minimise

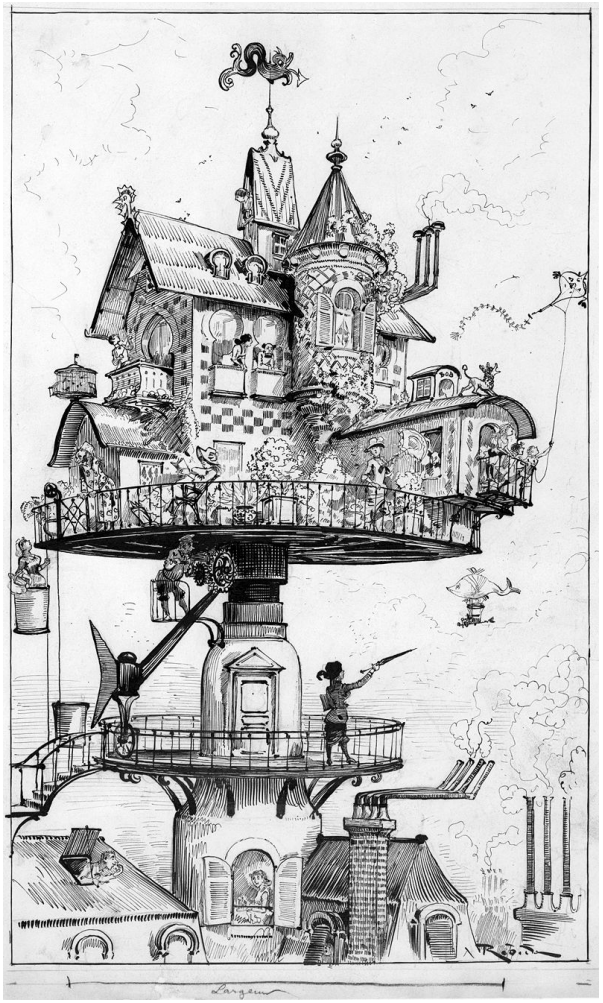
$$\textbf{cost} = \sum_i \log \frac{1}{p(\boldsymbol{x}_i | \boldsymbol{\alpha}, \boldsymbol{\Theta})}$$

- and more ...

Model-based Clustering

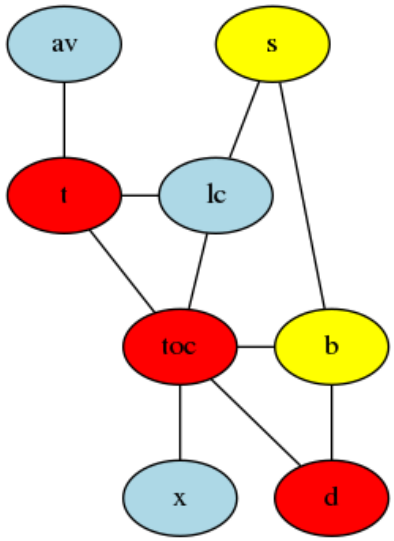
- The staticians refer to generative clustering models as model-based clustering, or *distribution-based clustering*.
- The pioneer in this area was *Adrian Raftery*.
- The framework allows the full machinery of applied statistics to be used.
- It also addresses a lot of the problems CS theoreticians have with definitions of clustering:
  - e.g. a clustering algorithm returns a partition
  - e.g. a clustering model should return a distribution over partitions
  - e.g. from which one could derive things like $\boldsymbol{x}_{10}$ and $\boldsymbol{x}_{27}$ are in the same cluster with probability 0.76
  - e.g. what constitutes a cluster can be customized based on the model families used

Outline



Reminder: Local Search

Graph colouring: try to find a colouring so no two neighbours have the same colour.



Then conflicts(x) is the number of neighbours of x with the same colour: conflicts(k) $= 0$, conflicts(t) $= 1$.

Local Search for Graph Colouring

Repeat until bored.

1. *Choose* dimension k randomly or by cycling through.
2. *Update* x_k to a colour such that conflicts(x_k) is locally minimum.

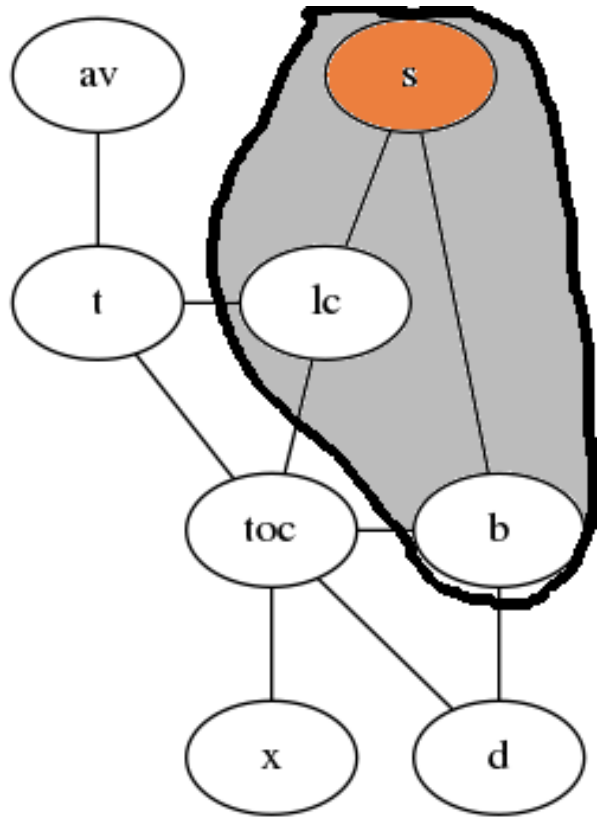
Probabilistic Local Search, example

The Grand Canyon (from Wikipedia.org).

To find the bottom (1) mostly walk down hill (2) but sometimes follow flat ridges (3) and maybe occasionally go up hill to get out of a local valley.

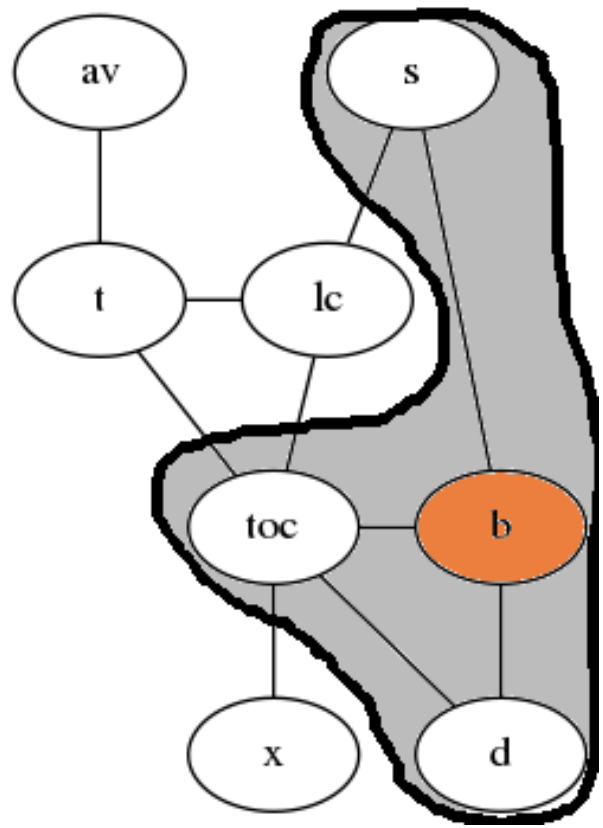
Probabilistic Local Search for Graph Colouring

Update s using $p(s|b, toc)$ where the distribution favours less conflicts.



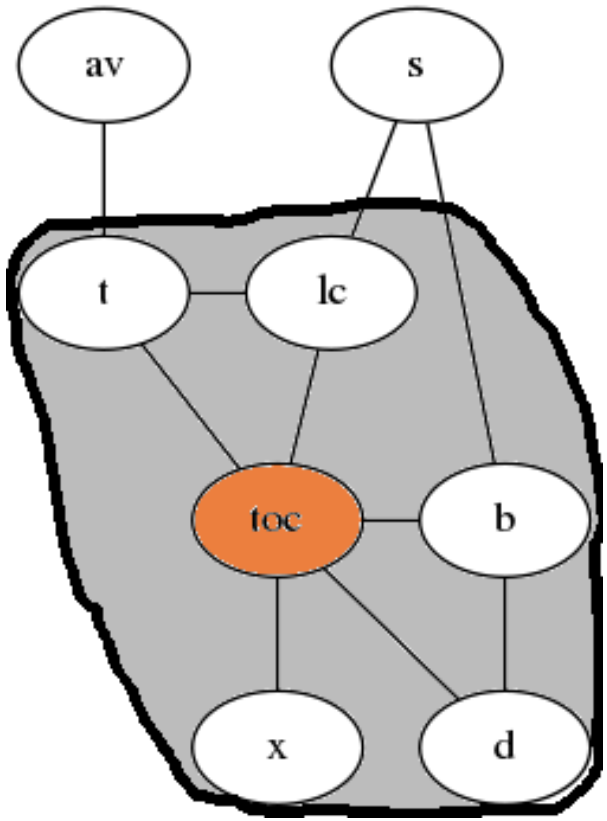
Probabilistic Local Search for Graph Colouring

Update b using $p(b|s, toc, d)$ where the distribution favours less conflicts.

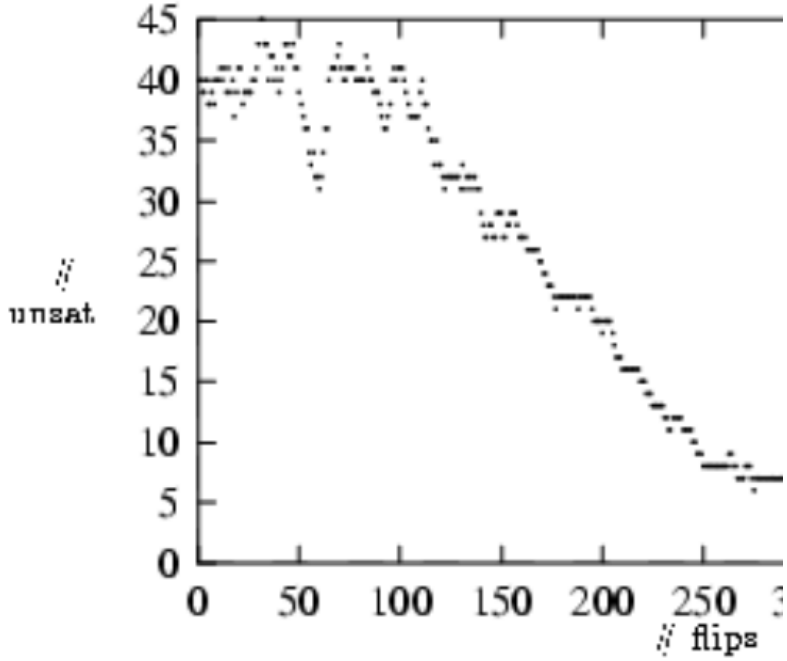


Probabilistic Local Search for Graph Colouring

Update *toc* using $p(toc|t, lc, b, x, d)$ where the distribution favours less conflicts.



Probabilistic Local Search, example



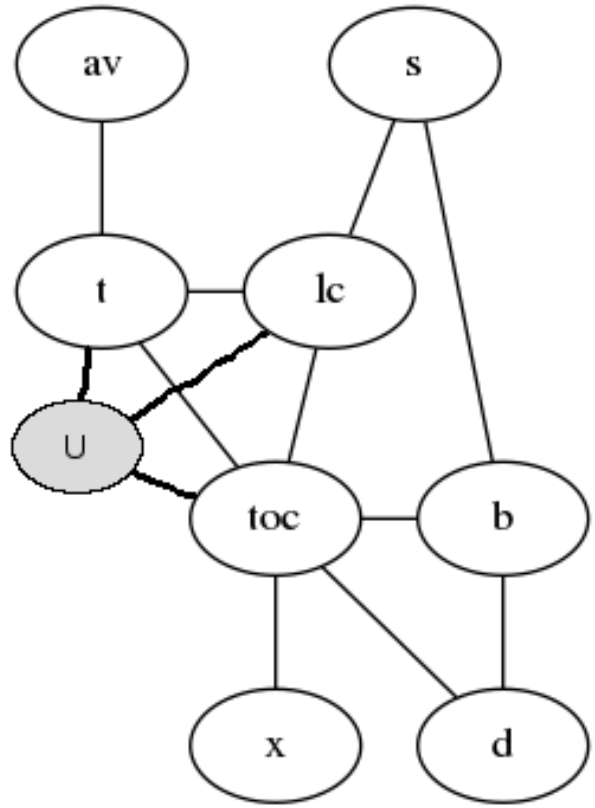
Here we have a 100 variable 3SAT problem done with "Gibbs" search. So we sample variables at each stage instead of assigning to the value maximising local probability.

Augmentation: Adding a Variable

Add a new variable (conditioned on some others).

Carefully designed to make sampling of the remainder easier.

On the left, add $p(U|t, lc, toc)$ to the distribution to get a new joint $P(U, X)$.

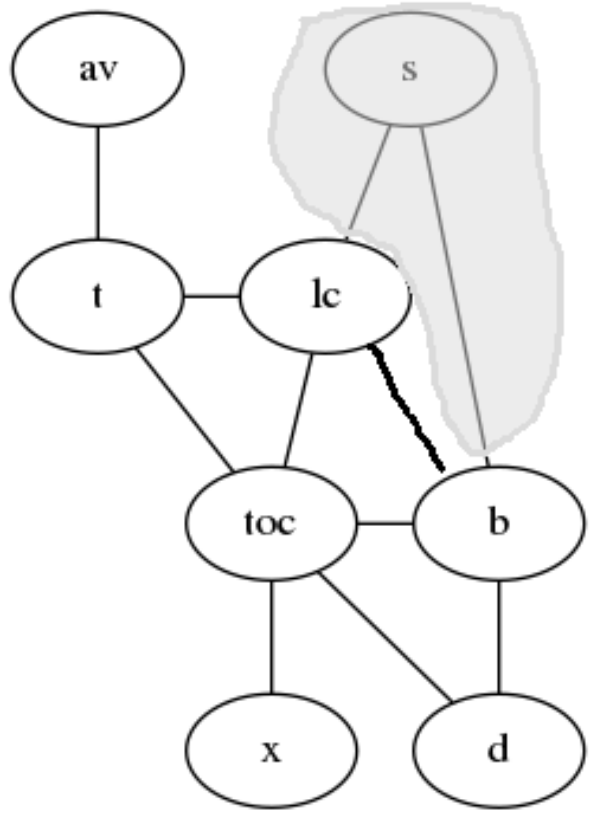


Collapsing: Removing a Variable

Marginalise out a variable.

Carefully designed to make sampling of the remainder easier.

On the left, marginalising out s from X induces a new arc between lc and b to get a new joint $P(X - \{s\})$.



General Probabilistic Local Search

Probabilistic Local Search

Input: $X = X_0$, conditional distributions

Repeat until bored.

1. *Choose* a dimension k.
2. *Update* x_k by $x_k \sim p(x_k \mid X/\{x_k\})$

- Only needs to compute/sample conditional probabilities $p(x_k \mid X/\{x_k\})$.
- So do not need the normalising constant for $p(X)$.
  - Which is the hardest thing to compute for many models!
- If there is sparse undirected graph behind the model, each computation $p(x_k \mid X/\{x_k\})$ will be sparse and thus more efficient.
  - hard for many deep neural networks!

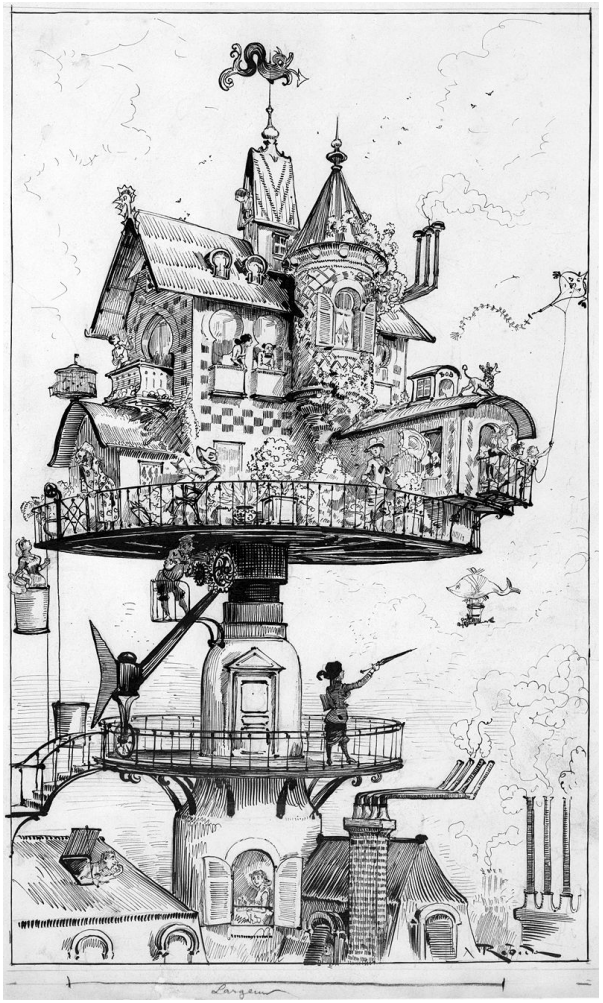
Gibbs Sampling

Gibbs Sampling

Probabilistic local search is called *Gibbs sampling*.

- The easiest and most widely used MCMC method.
- Collapsing and augmentation can make it really efficient.
- Software packages exist for doing it from model specifications:
  - *BUGS*
  - *Just Another Gibbs Sampler (JAGS)*
- These have revolutionised the application of statistics.
- With Hogwild Gibbs sampling can be made distributed, and has been ported to GPUs.

Outline



GMMs: Greedy Local Search

Local search for $\boldsymbol{c}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ to minimise

$$\textbf{cost} = \sum_i \log \frac{1}{p(\boldsymbol{x}_i, c_i | \boldsymbol{\alpha}, \boldsymbol{\Theta})} = \sum_i \log \left(\frac{1}{\alpha_{c_i} p(\boldsymbol{x}_i | \boldsymbol{\theta}_{c_i})} \right)$$

1. randomly **initialise** $\boldsymbol{c}$ (a partition to define clusters)
2. **repeat** until convergence:
   - 2.1 update $\boldsymbol{\alpha}$ by minimising

$$\textbf{cost} = \sum_i \log \frac{1}{\alpha_{c_i}} + \text{constant}$$

   - 2.2 for each k update $\boldsymbol{\theta}_k$ by minimising

$$\textbf{cost} = \sum_{\boldsymbol{x} \in S_k} \log \frac{1}{p(\boldsymbol{x} | \boldsymbol{\theta}_k)} + \text{constant}$$

   - 2.3 for each i, update c_i by maximising $p(\boldsymbol{x}_i, c_i | \boldsymbol{\alpha}, \boldsymbol{\Theta})$, then recompute the partition S

GMMs: Greedy Local Search, cont.

As before S_k is data assigned to cluster k, $\{ \boldsymbol{x}_i : c_i \equiv k \}$.

Local search for optimal $\boldsymbol{c}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ of **cost**:

1. randomly **initialise** $\boldsymbol{c}$ (a partition to define clusters)
2. **repeat** until convergence:
   2.1 update $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ using data in the cluster partitions

$$\begin{aligned} \alpha_k &\leftarrow \frac{|S_k|}{N} \\ \boldsymbol{\theta}_k &\leftarrow \text{Gaussian-MLE}(S_k) \end{aligned}$$

   2.2 for each i, update i to the most probable cluster,

$$c_i \leftarrow \operatorname*{argmax}_k \left(\alpha_k p(\boldsymbol{x}_i | \boldsymbol{\theta}_k) \right)$$

   then recompute the partition S

NB. this is an extension of k-means, but includes effects of cluster proportions $\boldsymbol{\alpha}$, and uses probability measures, not distance.

GMMs: Gibbs sampling

Gibbs sampling of $\boldsymbol{c}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ via

$$\textbf{probability} = p(\boldsymbol{\alpha})p(\boldsymbol{\Theta})\prod_i (\alpha_{c_i} p(\boldsymbol{x}_i|\boldsymbol{\theta}_{c_i}))$$

1. randomly **initialise** $\boldsymbol{c}$ (a partition to define clusters)
2. **repeat** for some time:
   - 2.1 update $\boldsymbol{\alpha}$ by sampling proportionally to

$$\textbf{probability} \propto p(\boldsymbol{\alpha})\prod_i \alpha_{c_i}$$

   - 2.2 for each k update $\boldsymbol{\theta}_k$ by sampling proportionally to

$$\textbf{probability} \propto p(\boldsymbol{\Theta})\prod_{\boldsymbol{x}\in S_k} p(\boldsymbol{x}|\boldsymbol{\theta}_k)$$

   - 2.3 for each i, update c_i by sampling proportionally to $p(\boldsymbol{x}_i, c_i|\boldsymbol{\alpha}, \boldsymbol{\Theta})$, then recompute the partition S

NB. This is a stochastic version of the previous greedy algorithm.

GMMs: Gibbs sampling, cont.

Here is an incomplete version of the Gibbs sampler:

- requires appropriate conjugate priors,
- haven't given parameter values for sampling.

1. randomly **initialise** $\boldsymbol{c}$ (a partition to define clusters)
2. **repeat** for some time:
   - 2.1 update $\boldsymbol{\alpha}$ by sampling from the posterior *Dirichlet*
   - 2.2 for each k update $\boldsymbol{\theta}_k$ by sampling from the posterior Gaussian
   - 2.3 for each i, update $c_i = k$ by sampling proportionally to $\alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)$

GMMs: Gradient-based Search

Gradient-based search for $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ on

$$\textbf{cost} = \sum_i \log \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta})} = -\sum_i \log \left(\sum_k \alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k) \right)$$

$$\begin{aligned}
\frac{\partial \textbf{cost}}{\partial \alpha_k} &= -\sum_i \frac{p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}{p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta})} \\
&= \frac{-1}{\alpha_k} \sum_i p(c_i{=}k|\boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\Theta}) \\
\frac{\partial \textbf{cost}}{\partial \boldsymbol{\theta}_k} &= \sum_i p(c_i{=}k|\boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\Theta}) \frac{\partial \log \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}}{\partial \boldsymbol{\theta}_k}
\end{aligned}$$

GMMs: Gradient-based Search

Gradient-based search for $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ on

$$\textbf{cost} = \sum_i \log \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta})} = -\sum_i \log \left(\sum_k \alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k) \right)$$

$$\begin{aligned} \frac{\partial \textbf{cost}}{\partial \alpha_k} &= -\sum_i \frac{p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}{p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta})} \\ &= \frac{-1}{\alpha_k} \sum_i p(c_i{=}k|\boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\Theta}) \\ \frac{\partial \textbf{cost}}{\partial \boldsymbol{\theta}_k} &= \sum_i p(c_i{=}k|\boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\Theta}) \frac{\partial \log \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}}{\partial \boldsymbol{\theta}_k} \end{aligned}$$

It appears we have regular looking cost function derivatives that have been weighted by $p(c_i{=}k|\boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\Theta})$.

If these weights are treated as constants, the solutions are easy.

GMMs: Averaging Gibbs Sampling

- The Gibbs step for c_i: sample $c_i = k$ proportionally to

$$q(c_i|\boldsymbol{x}_i) = p(c_i|\boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\Theta})$$

- Make a cost function the has "fractional" samples given by proportions $q(c_i{=}k|\boldsymbol{x}_i)$.
- This artificial cost function is:

$$\textbf{cost-q} = \sum_i \sum_k q(c_i{=}k|\boldsymbol{x}_i) \log \frac{1}{\alpha_{c_i=k} p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}$$

  This moves the sum outside of the log!

- We can formalise this operation using the *expectation–maximization algorithm* (EM), or alternatively the *variational methods* such as the mean field approximation.

NB. variational method commonly used with deep neural networks.

Variational Method: Framework

- The problem considered has a model with latent data $\boldsymbol{c}$.
- Have a model $p(\boldsymbol{x}_i, c_i|\boldsymbol{\Theta})$, where only $\boldsymbol{x}_i$ is observed.
  - clustering, latent Dirichlet allocation, missing values, ...
- We will develop an approximate conditional distribution for the latent $\boldsymbol{c}$

$$q(c_i|\boldsymbol{x}_i) \approx p(c_i|\boldsymbol{x}_i, \boldsymbol{\Theta})$$

- The aim is to use the approximate distribution $q()$ to simplify the log probability.
- However, $q()$ will need to be repeatedly re-estimated as $\boldsymbol{\Theta}$ changes.
- Called "variational" because *calculus of variations* is, in principle, needed to work the solutions.
  - in practice, you just need differentials (like $\partial q()$) and some common sense math.

i.e. I could teach you in 10 minutes

The KL Variational Setup

Consider the following equality:

$$\log p(\boldsymbol{x}, \boldsymbol{\Theta}) - \sum_i KL\left(q(c_i|\boldsymbol{x}_i)|p(c_i|\boldsymbol{x}_i, \boldsymbol{\Theta})\right)$$
$$= \quad \mathrm{E}_{q()}\left[\log p(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{\Theta})\right] + \sum_i I(q(c_i|\boldsymbol{x}_i))$$

- The two lines are equal so can work with either one.
- Is true for any distribution $q()$. So we can change it to help us, as needed.
- From this we also get the evidence lower bound (ELBO)

$$\log p(\boldsymbol{x}, \boldsymbol{\Theta}) \geq \mathrm{E}_{q()}\left[\log p(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{\Theta})\right] + \sum_i I(q(c_i|\boldsymbol{x}_i))$$

which becomes an equality for the EM algorithm.

The KL Variational Setup, cont.

Consider the following equality:

$$\log p(\boldsymbol{x}, \Theta) - \sum_i KL\left(q(c_i|\boldsymbol{x}_i)|p(c_i|\boldsymbol{x}_i, \Theta)\right)$$
$$= \quad \mathrm{E}_{q()}\left[\log p(\boldsymbol{x}, \boldsymbol{c}, \Theta)\right] + \sum_i I(q(c_i|\boldsymbol{x}_i))$$

- Maximise the first line w.r.t. $q()$
- So minimise just the $KL()$ terms.
- So build an approximation $q(c_i|\boldsymbol{x}_i) = p(c_i|\boldsymbol{x}_i, \Theta)$.
  - not possible for more complex models and deep neural networks, so they usually just seek to get $q(c_i|\boldsymbol{x}_i)$ closer under $KL()$

The KL Variational Setup, cont.

Consider the following equality:

$$\log p(\boldsymbol{x}, \Theta) - \sum_i KL\left(q(c_i|\boldsymbol{x}_i)|p(c_i|\boldsymbol{x}_i, \Theta)\right)$$

$$= \quad \mathrm{E}_{q()}\left[\log p(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{\Theta})\right] + \sum_i I(q(c_i|\boldsymbol{x}_i))$$

- Maximise the second line w.r.t. the $\boldsymbol{\Theta}$.
- So ignore the $I()$ terms.
- So use the approximation to more simply estimate Θ from a fractional version of the data likelihood.
  - i.e. $\mathrm{E}_{q()}\left[\log p(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{\Theta})\right]$ is simpler to fit than $\log p(\boldsymbol{x}, \boldsymbol{\Theta})$ because the sum inside the log disappears

Use on a Mixture Model

The KL variational method for a mixture model reduces to:

1. **Initialise** $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ somehow.
2. **Repeat** until convergence.
   2.1 Update the $q()$ approximations (as vectors)

$$q(c_i|\boldsymbol{x}_i) \leftarrow p(c_i|\boldsymbol{x}_i, \alpha, \boldsymbol{\Theta})$$

   2.2 Fit $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ to minimise the score, with $q()$ fixed:

$$\textbf{cost-q} = \sum_i \sum_k q(c_i|\boldsymbol{x}_i) \log \frac{1}{\alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}$$

- We are using fractional data, as suggested previously!
- This moves the sum out from inside of the log.
- In our case, this converges on a local maxima.

Use on the GMM

The KL variational method for a GMM reduces to:

1. randomly **initialise** $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$
2. **repeat** until convergence.
   - 2.1 update the $q()$ approximations (as vectors)

$$q(c_i|\boldsymbol{x}_i) \leftarrow p(c_i|\boldsymbol{x}_i, \alpha, \boldsymbol{\Theta})$$

   - 2.2 update $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ using fractional data, the $q()$:

$$\begin{aligned} \alpha_k &\leftarrow \frac{\sum_i q(c_i{=}k|\boldsymbol{x}_i)}{N} \\ \boldsymbol{\theta}_k &\leftarrow \text{Gaussian-Weighted-MLE}\left(\{x_i, q(c_i{=}k|\boldsymbol{x}_i) \ : \ i = 1, \ldots, N\}\right) \end{aligned}$$

- This is a fractional version of the k-means style algorithm.
- It also represents an averaging of the Gibbs style algorithm.

Compare with the Greedy Algorithm

1. randomly **initialise** $\boldsymbol{c}$ (a partition to define clusters)
2. **repeat** until convergence:
   2.1 update $\boldsymbol{\alpha}$, $\boldsymbol{\Theta}$ using data in the cluster partitions

$$\begin{aligned} \alpha_k &\leftarrow \frac{|S_k|}{N} \\ \boldsymbol{\theta}_k &\leftarrow \text{Gaussian-MLE}(S_k) \end{aligned}$$

   2.2 for each i, update i to the most probable cluster,

$$c_i \leftarrow \operatorname*{argmax}_k p(c_i | \boldsymbol{x}_i, \alpha, \boldsymbol{\Theta})$$

   then recompute the partition S

The KL Var. Method on Distributions

Exponential family models: there are many, including:

Name	Domain	Parameters	$p(x \mid \lambda, \cdots)$
Bernoulli(λ)	$x \in \{0, 1\}$	$\lambda \in (0, 1)$	$\lambda^x (1-\lambda)^{1-x}$
multinomial($K, \vec{\lambda}, N$)	$\vec{x} \in \mathcal{N}^K$	(as above)	$\binom{N}{\vec{x}} \prod_{k=1}^K \lambda_k^{x_k}$
Dir.multin.($K, \vec{\lambda}, N$)	$\vec{x} \in \mathcal{N}^K$	$\lambda_k > 0, \sum_{k=1}^K \lambda_k = A$	$\binom{N}{\vec{x}} \frac{\Gamma(A)}{\Gamma(N+A)} \prod_{k=1}^K \lambda_k^{(x_k)}$
Poisson(λ)	$x \in \mathcal{N}$	$\lambda > 0$	$\frac{1}{x!} \lambda^x e^{-\lambda}$
neg.binom.(λ, ρ)	$x \in \mathcal{N}$	$\lambda > 0, \rho \in (0, 1)$	$\frac{1}{x!} \lambda^{(x)} \rho^x (1-\rho)^\lambda$

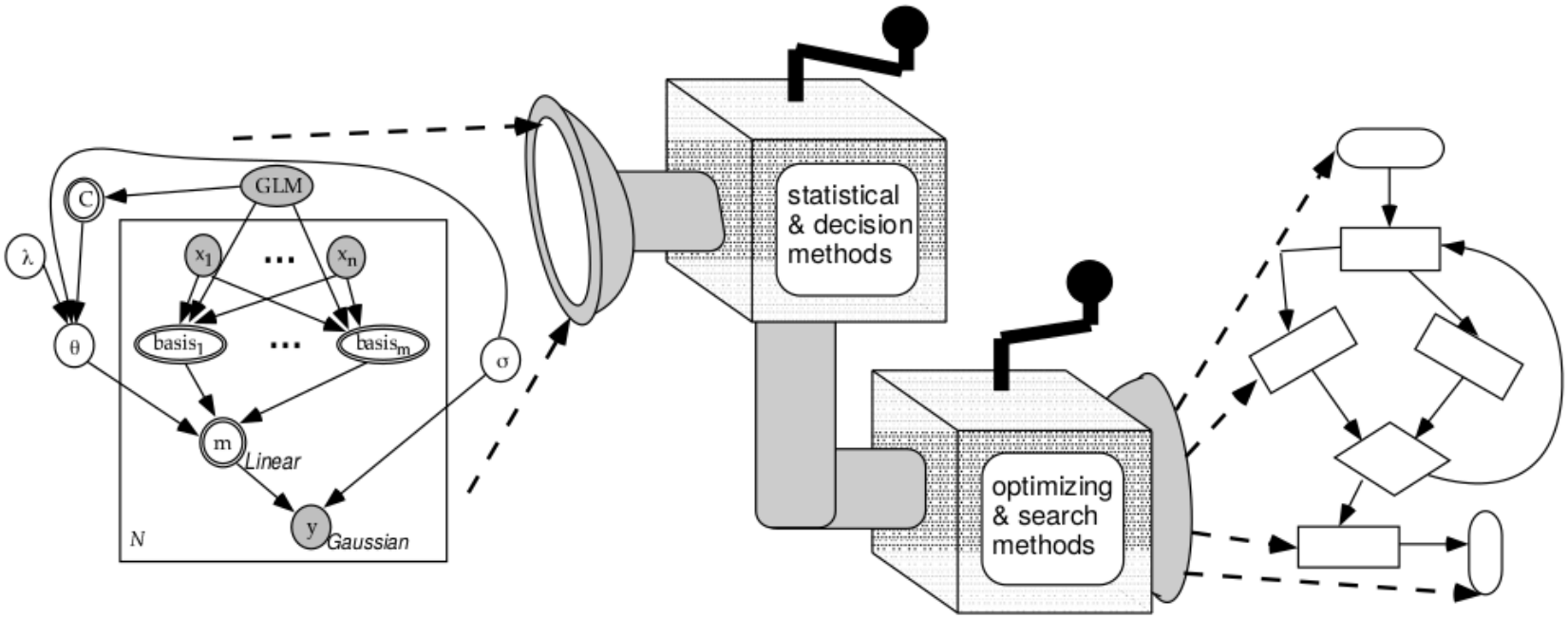
Name	Domain	Parameters	$p(\vec{\lambda} \mid \cdots)$
beta(α, β)	$\lambda \in (0, 1)$	$\alpha, \beta > 0$	$\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \lambda^{\alpha-1} (1-\lambda)^{\beta-1}$
gamma(α, β)	$\lambda > 0$	$\alpha, \beta > 0$	$\frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\lambda\beta}$
Dirichlet($\vec{\alpha}$)	$\lambda_k > 0, \sum_{k=1}^K \lambda_k = 1$	$\alpha_k > 0$	$\frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)} \prod_k \lambda_k^{\alpha_k - 1}$

- KL var. method works well on arbitrary networks of these.
- In most cases has closed form solutions to the algorithm steps.
- And is approximately a Newton-Raphson update for these.

The KL Variational Properties

- For a mixture model, the variational method becomes equivalent to the expectation–maximization (EM) algorithm.
- A much more general theory exists for arbitary network models. A very broad family!
- In neural networks, the update to $\boldsymbol{\Theta}$ is done stochastically and $q()$ never optimised.

Automating Statistical Inference

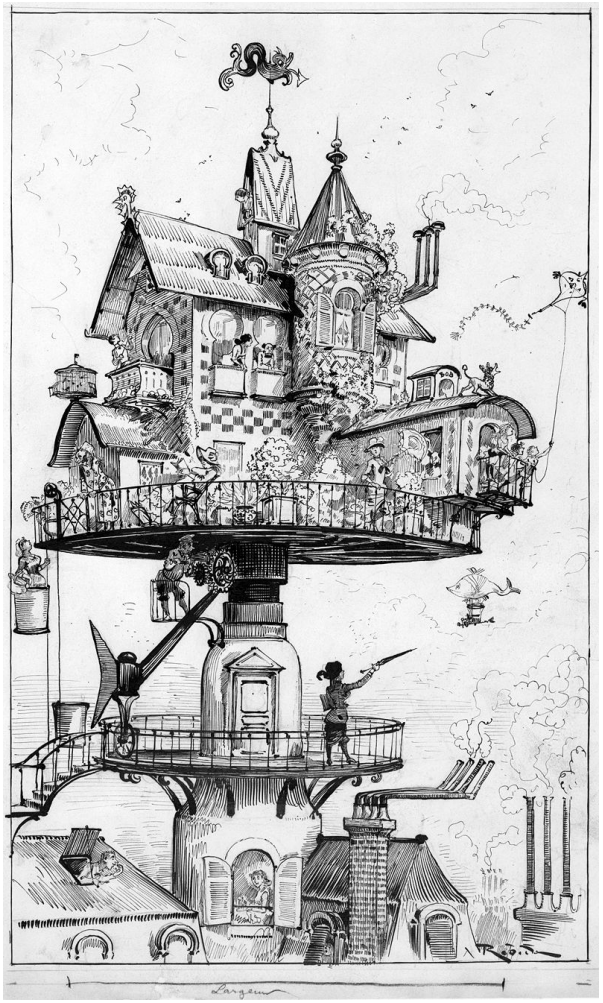


from Buntine *JAIR* **1994**

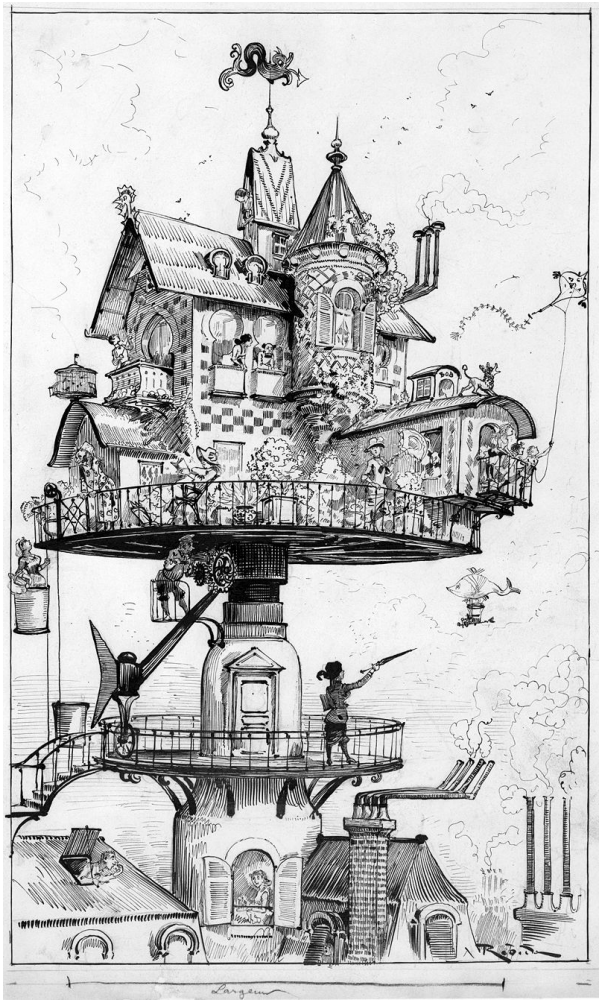
Applying Gibbs and the KL Variational method, once you now the details, is routine and we are one step away from automating a lot of it.

- initial efforts like *Stan* from Columbia underway

Outline



Outline



Cluster Indices: Is This ...

slide modified from Ahmed, MLSS 2014

Cluster Indices: The Same As This ...

slide modified from Ahmed, MLSS 2014

Cluster Indices: Or This ...

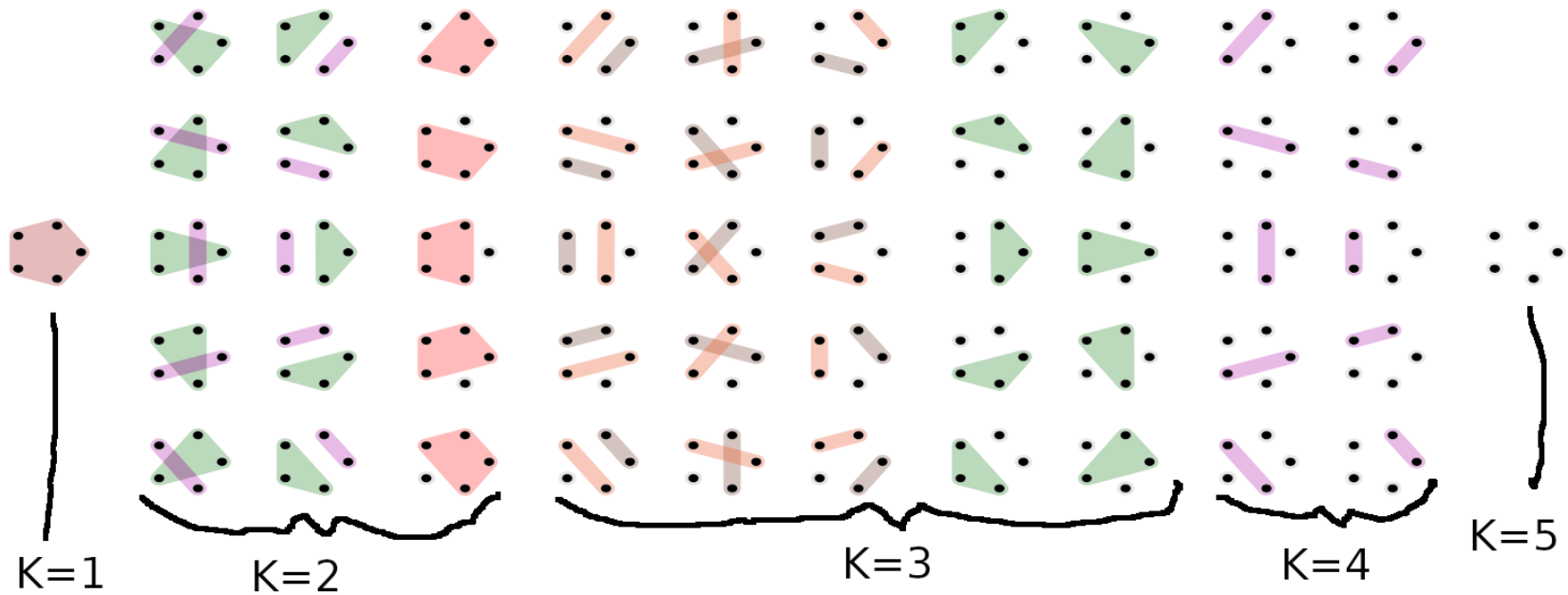
slide modified from Ahmed, MLSS 2014

Label Switching

- Clusters do not come with labels.
- *Constructing linguistic labels* can be challenging.
- After clustering, cluster details are usually stored in vectors or structures and simply indexed, maybe as 1,2,3,...
- In principle, we can randomly permute the order of the clusters (or change cluster indices), and **most** cost functions or cluster probabilities will be the same.
  - WCSS is invariant if we switch cluster indices.
- This is called the label switching problem.
  - confusingly named ... should be the "cluster index switching problem"

Partitions

- The label switching problem disappears if instead of searching
  - the **space of clustering assignments:** $\boldsymbol{c}$, mapping data indices (1,...,N) to a cluster indices (1,...,K),
  - we search the **space of partitions:** partition the set $\{1, ..., N\}$ into K separate sets.
- Usually, we assume a partition has no empty sets.
- Below, for $N = 5$ shows different partitions for $K = 1$, $K = 2$, etc.



Partitions, cont.

- For fixed K we usually ignore the theory of partitions and search in the space of clustering assignments.
- If we make K unknown, we need to be aware of the theory of distributions over partitions.
- Developed extensively by Pitman from 1995 onwards "exchangeable random partitions", for instance:
  - *Pitman-Yor process* (PYP)
  - *Dirichlet process* (DP) which is an instance of the PYP
- Similar theory applies to trees (for hierarchical clustering).
- For a more readable account see *"A Bayesian View of the Poisson-Dirichlet Process"* by Buntine and Hutter, 2010.

Fun facts about combinatorics of partitions:

- The total number of partitions of an N-element set is the *Bell number* B_n: $B_5 = 52$, $B_6 = 203$, ...
- The number of partitions of an N-element set into exactly K nonempty parts is the *Stirling number of the second kind*, S_N^K. e.g., $S_{10}^3 = 511$, $S_{10}^8 = 750$, $S_N^1 = 1$, $S_N^N = 1$.

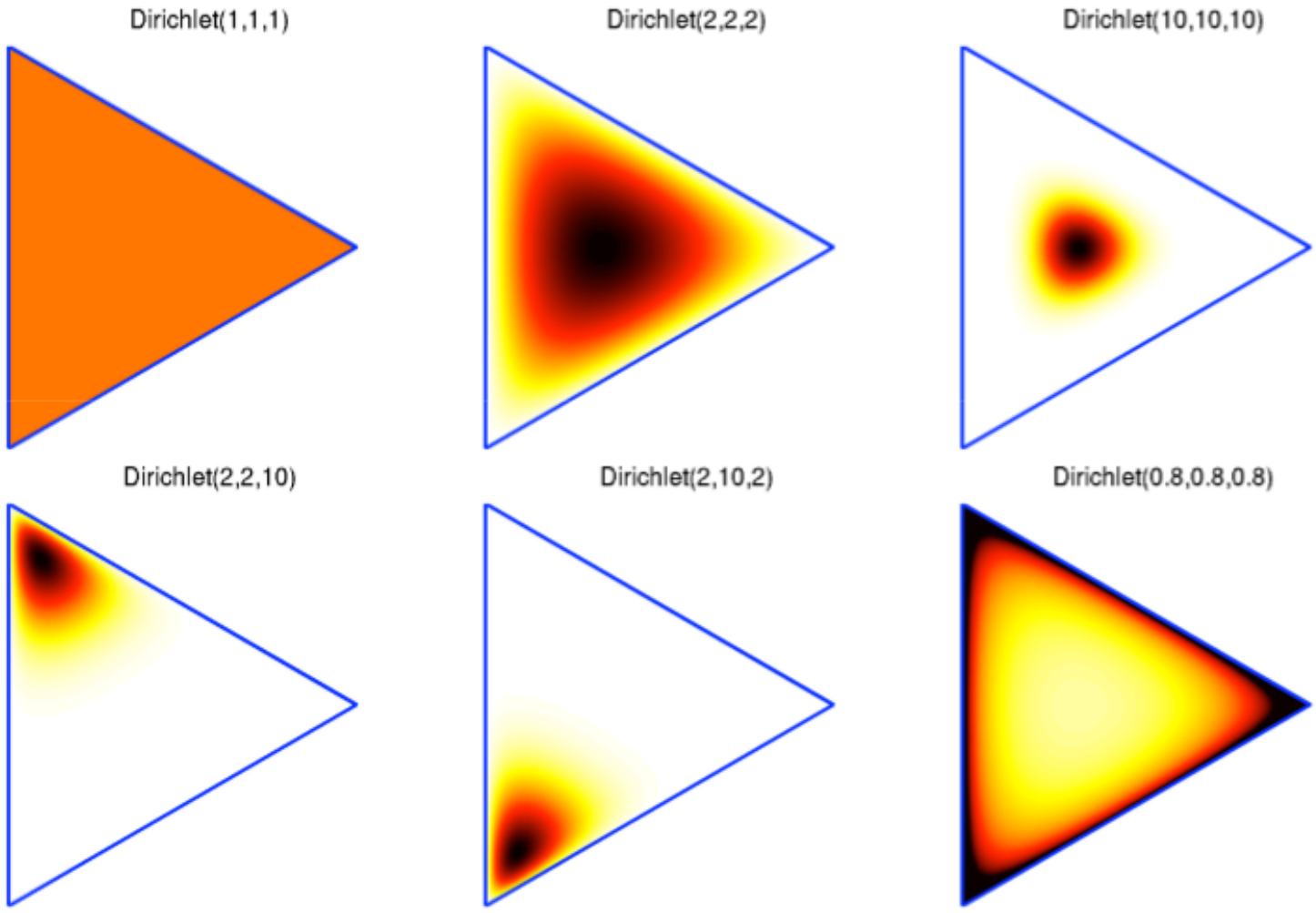
Known Finite Partitions with Unknown K

Mixture Model:

$$p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)$$

- In the "how many species of mosquitoes" case, place a *Dirichlet distribution* on $\boldsymbol{\alpha}$ and a distribution on K.
- We want to estimate $\boldsymbol{\alpha}$ for a few different K.
- Standard parametric and exponential family analysis.

3-D Dirichlet Plots



NB. Dirichlets have an inverse variance parameter α and a mean.

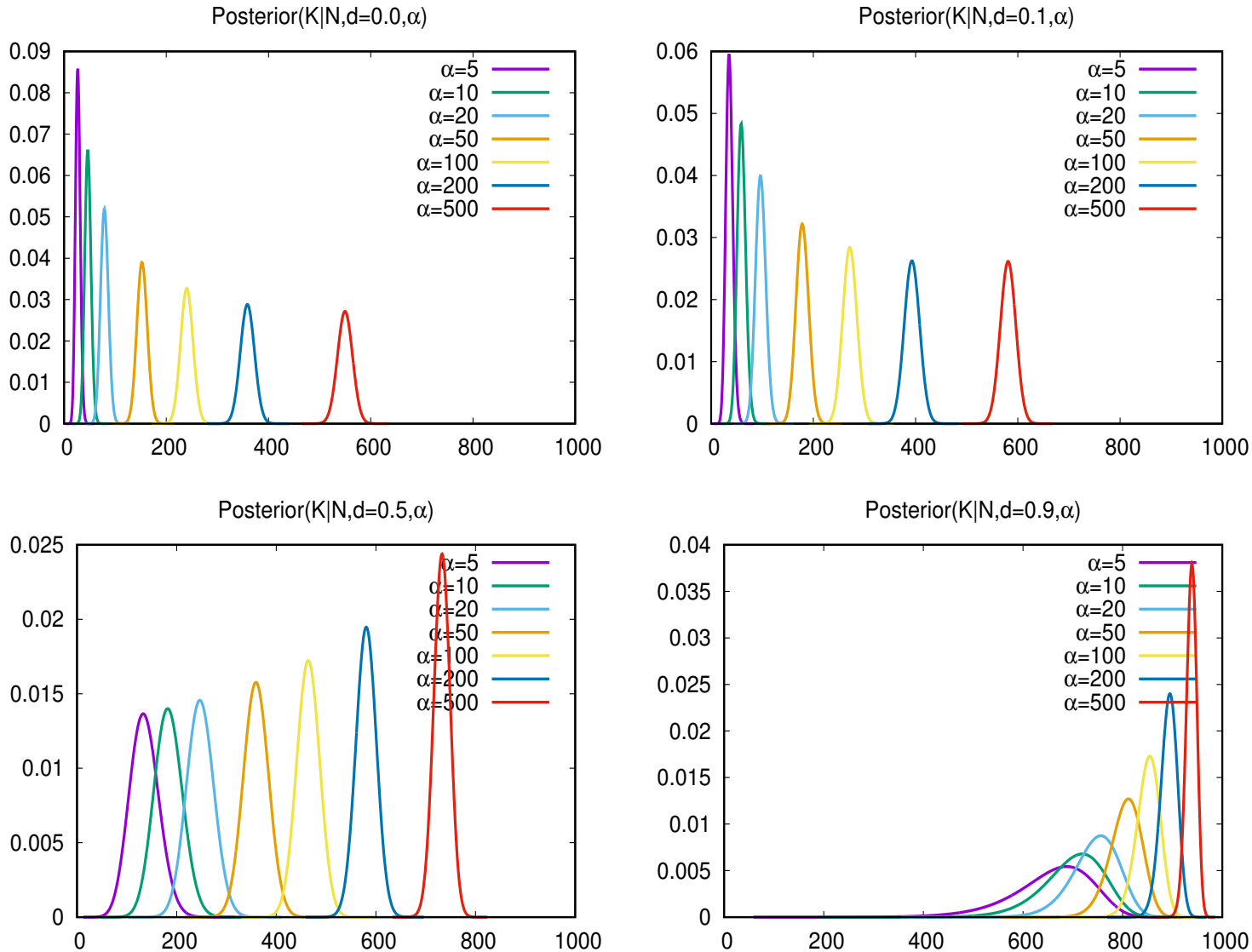
Infinite Partitions with Finite N

Infinite Mixture Model:

$$p(\boldsymbol{x}_i|\boldsymbol{\alpha}, \boldsymbol{\Theta}) = \sum_{k=1}^{\infty} \alpha_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)$$

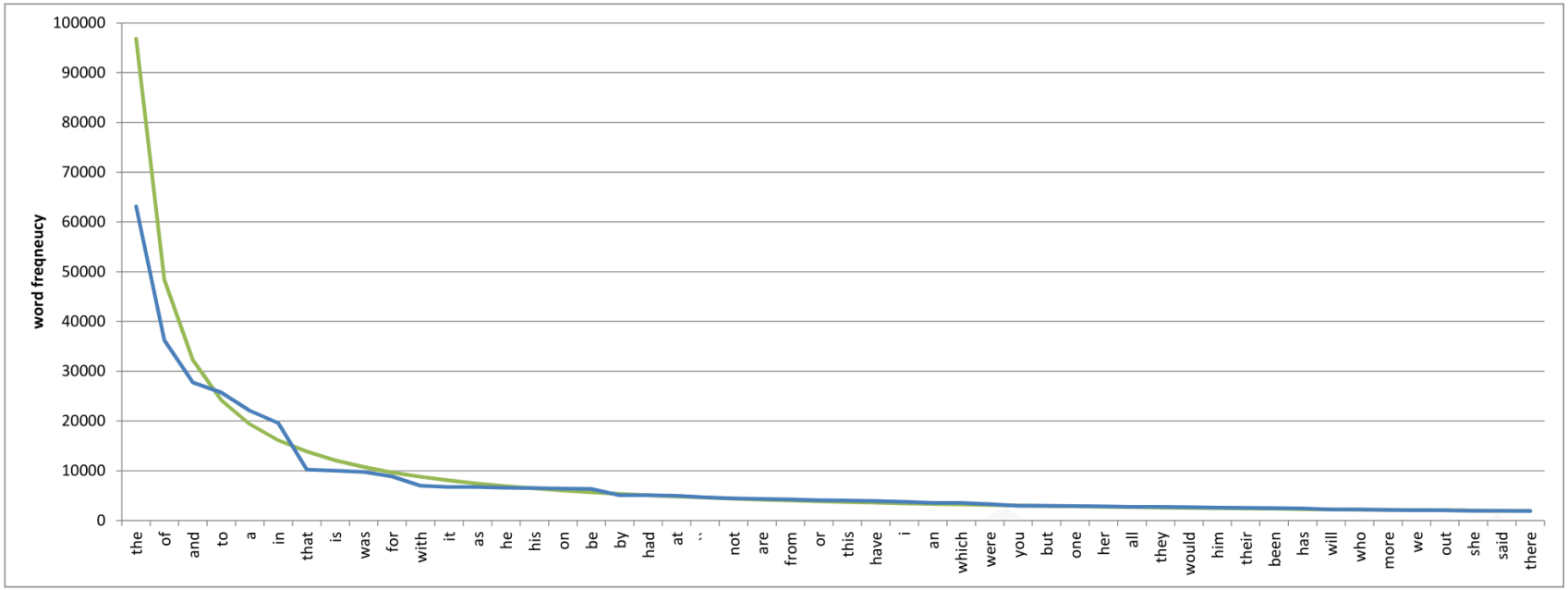
- In the "how many music genres" case, make $K = \infty$ and place a non-parametric distribution on $\boldsymbol{\alpha}$.
  e.g. Pitman-Yor process or Dirichlet process
- We want to estimate the infinite vector $\boldsymbol{\alpha}$.
- The Dirichlet process is an infinite counterpart to the Dirichlet.
- Standard non-parametric analysis largely reduces to parametric inference.
  - moderately efficient methods exist, even in the hierarchical case, see Buntine *et al.* and Mingyuan Zhou *et al.*
  - Warning: you need to ignore a lot of the ML literature on this, because the Chinese restaurant process doesn't seem to be useful for any situation, including hierarchical models!

Infinite K is Finite for Fixed N



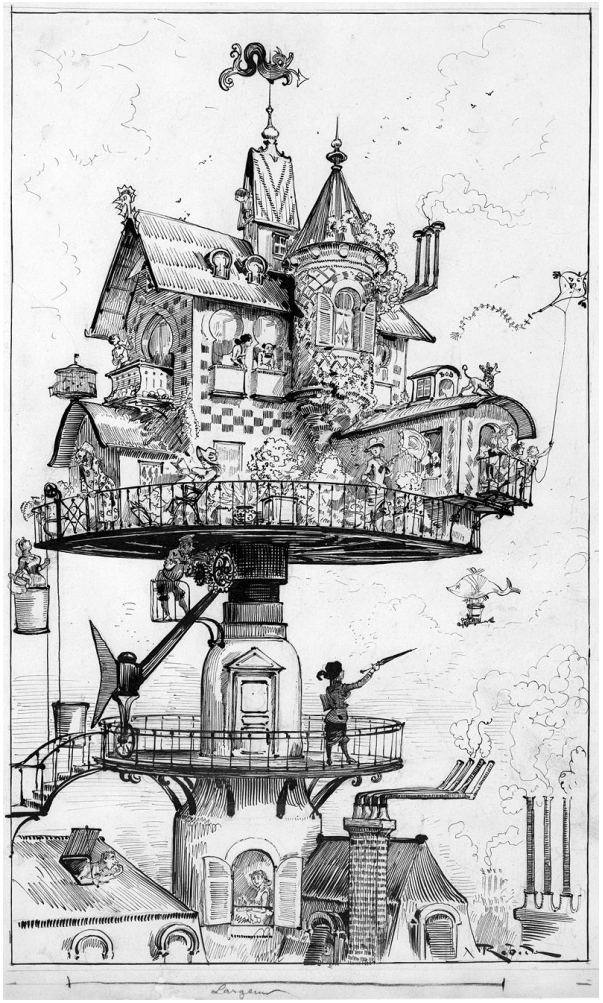
PYP: posterior probability on K given $N = 1000$ and different d, α.

Zipfian Probabilities



- we count the different words types in a large collection
- if we order words by rank, from most frequent to least frequent,
- then the proportion for words drops dramatically
- in fact log(probability(word)) roughly drops linearly
- the PYP generates Zipfian like probability vectors

Outline



Cost Functions 101

- we train a model with parameters $\boldsymbol{\Theta}$ using training set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- have a cost or *loss function* based on each data point $\boldsymbol{x}$ and the model, $l(\boldsymbol{x}, \boldsymbol{\Theta})$

e.g. squared loss, ϵ-insensitive loss, Brier score, absolute error

- one then uses cost on the data, $\textbf{cost} = \sum_{i=1}^{N} l(\boldsymbol{x}_i, \boldsymbol{\Theta})$
- cost functions are schizophrenic:

utility based: for real decisions measured in units of dollars, lives lost, or other measure of true worth

score based: e.g., proper scoring rule to get a calibrated classification model

Cost Functions 101, cont.

- can add a *regularizing term* to penalize "complex" models

e.g. in regression estimating y_i with $\boldsymbol{x}^T\boldsymbol{\theta}$, ridge regression using a loss of squared error with L_2 regularization has cost

$$\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T\boldsymbol{\theta})^2 + \lambda L_2(\boldsymbol{\theta})$$

whereas lasso regression uses L_1

- both are often poor in the general case of regression
- *Gaussian processes* effectively allow the function shape to be considered as well

- Bayesian estimation roughly corresponds to these with negative log probability:

$$\sum_i \log \frac{1}{p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})} + \log \frac{1}{p(\boldsymbol{\theta})}$$

- Generally one has a cost $\sum_i l(\boldsymbol{x}_i, \boldsymbol{\Theta}) + \lambda L(\boldsymbol{\Theta})$
  - trading complex model against loss on data
  - λ hard to know unless one is Bayesian, so use cross validation or a hold out set to estimate it

Bayesian Theory 101

Bayes Theorem for I.I.D. data is:

$$p(\boldsymbol{\theta}|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N) = \frac{p(\boldsymbol{\theta})\prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{\theta})}{p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N)}$$

$$p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N) = \int p(\boldsymbol{\theta})\prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}$$

- probabilities like $p(\boldsymbol{x}_i|\boldsymbol{\theta})$ are testable, and correspond to predictions for data frequencies
- probabilities like $p(\boldsymbol{\theta})$ are untestable, and correspond to *beliefs*, and can be considered subjective
- since they are beliefs about $\boldsymbol{\theta}$, they are beliefs about frequencies, said to *Bayesian*
- Why do we use them?

Bayesian Theory TL;DR

- there are strong theoretical justifications for using Bayesian methods
- but they ignore computational and multi-agent issues
  - of the kind that applied ML people well understand!
- and they don't say you must *use* Bayesian methods
  - but you should be reasonable consistent with them
- in practice we do hybrids, compromises, and variations, i.e., "applied Bayesian"
  - like regularisation with cross validation
  - like ensembling
  - so-called "empirical Bayes"
  - MDL and MML approaches

Bayesian Justifications (skip)

Kolmogorov's Axioms: justify probability as frequency

- based on simple counting arguments

Cox's Axioms: justify that probability as belief should be handled the same as probability as frequency

- based on simple properties of belief

Dutch Books: justify that probability as belief should be handled the same as probability as frequency

- based on gambling against a person who doesn't have underlying beliefs

von Neumann-Morgenstern Axioms: justify specifying costs and making decisions based on minimum expected cost

- based on comparing alternative rewards

But, Hidden Assumptions (skip)

Using Bayesian probabilities makes the following implicit assumptions:

- you have unlimited computational power;
- you are a single agent seeking to optimize performance in a directly measurable sense;
- you are using a model family that includes a sufficiently close approximation to the "truth."

But, Hidden Assumptions (skip)

These become dangerous when:

- seeking to model "objective" reasoning;
- using simple linear models for everything;
- have a complex model you cannot justify priors for;
- seeking to test a new drug for safety before to public release;
- using an old CPU with low power and memory.

Model Selection

TBD

- the evidence
- AIC, BIC as crude approximations

Encoding Methods (skip)

Minimum Description Length (MDL): formalised as NML, assumes a log probability utility function and does a prior-free minimax Bayesian approach with a specific coding intepretation as a 2-part code

- see *"An MDL framework for data clustering"* by Kontkanen, Myllymäki, *et al.* 2005

Minimum Message Length (MML): a Bayesian-like method that sets up a 3-part code code length for the parameter space (sometimes corresponds to using a Jeffreys prior)

- see *"Unsupervised learning using MML"* by Oliver, Baxter and Wallace, 1996.

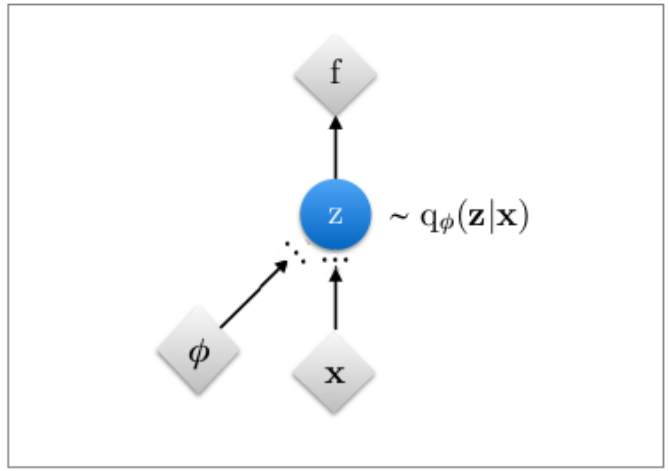
Popular MDL: a separate method based loosely on the notion of coding hypotheses and then the data

Different ways of motivating and ideating a cost function.

Cost Functions and Latent Variables

- The problem considered has a model with latent data $\boldsymbol{c}$.
- Have a model $p(\boldsymbol{x}_i, c_i|\boldsymbol{\Theta})$, where only $\boldsymbol{x}_i$ is observed.
  - clustering, latent Dirichlet allocation, missing values, ...
- Naive approach: $\textbf{cost} = \sum_{i=1}^{N} l(\boldsymbol{x}_i, c_i, \boldsymbol{\Theta})$
  - this is the simplest approach which causes estimation bias
  - can be viewed as over-fitting: room to optimise latent c_i
- Using a generative probability model and a well designed algorithm solves the problem:
  - sample $\boldsymbol{c}$ with MCMC such as Gibbs
  - use variational methods

Neural Networks and Latent Variables



modified from *"Intro to VAEs"* Kingma and Welling 2019, p.23

- for gradient descent, one would like to marginalise out z and compute $\frac{\partial \log p(f|\mathbf{x},\phi)}{\partial \phi}$
- the existence of latent z blocks any derivative of f from being computed past z

NNs and Latent Vars, cont.

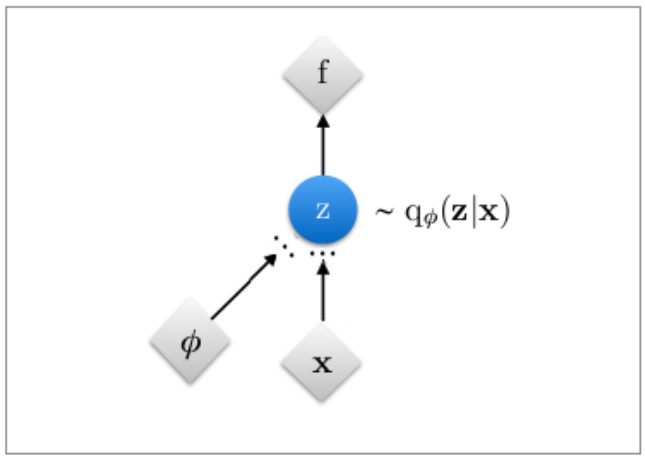
Derivatives with Latent Variables

The following identity holds:

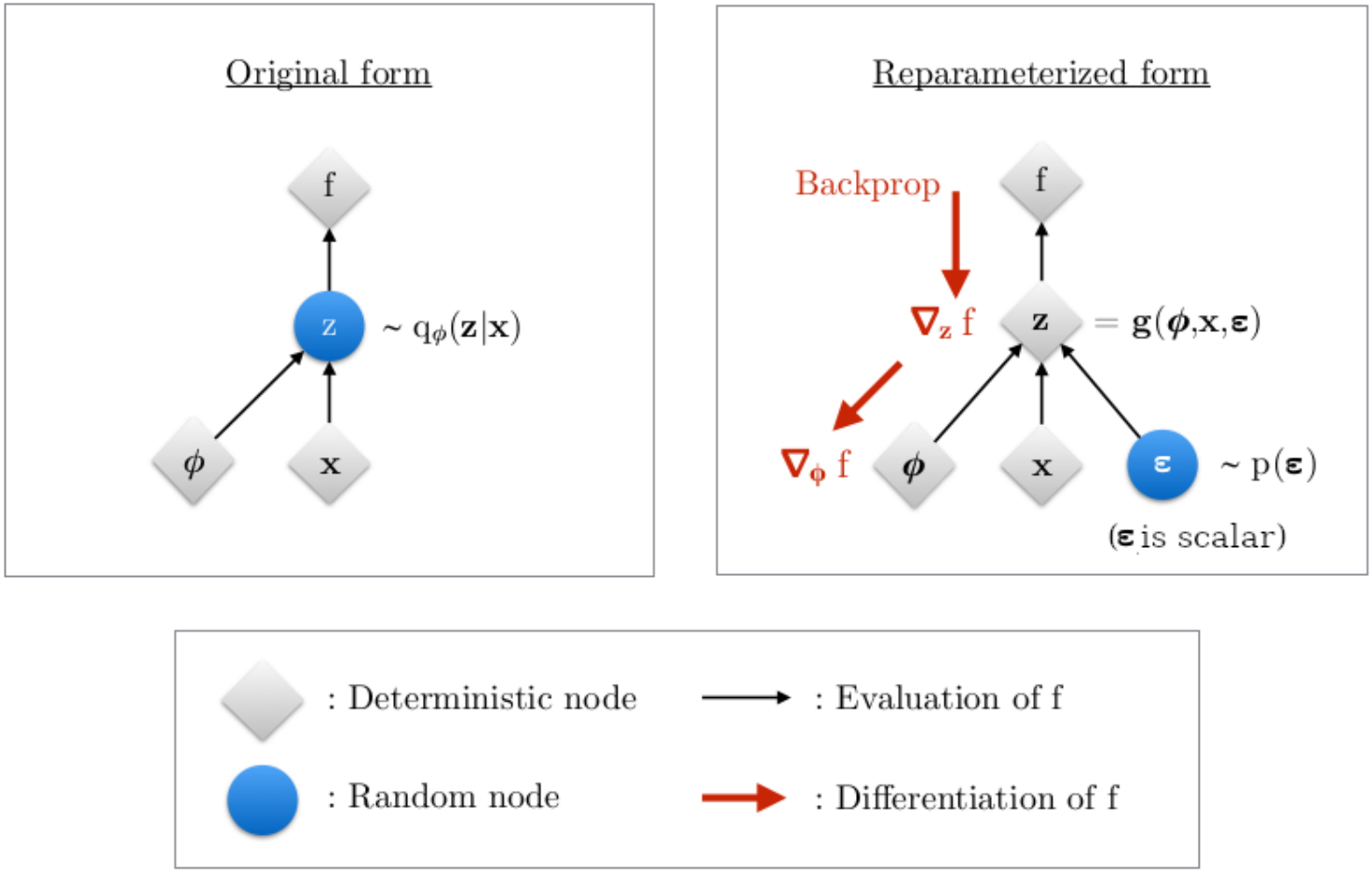
$$\frac{\partial \log p(f|\boldsymbol{x},\phi)}{\partial \phi} = \int p(z|f,\boldsymbol{y},\Theta)\frac{\partial \log p(z|\boldsymbol{x},\phi)}{\partial \phi}\mathrm{d}z \ .$$

- need a Gibbs-like step to sample z via $p(z|f,\boldsymbol{x},\phi)$
  - not currently done with neural nets, but could be
- so technique becomes

1. sample $z \sim p(z|f,\boldsymbol{x},\phi)$
2. use the stochastic approximation $\frac{\partial \log p(f|\boldsymbol{x},\phi)}{\partial \phi} \approx \frac{\partial \log p(z|\boldsymbol{x},\phi)}{\partial \phi}$



NNs and Reparameterisation Trick



from *"Intro to VAEs"* Kingma and Welling 2019, p.23

A latent variable blocks differentiation.

But we can isolate the randomness into a leaf to reinstate differentiation.

NNs and Reparameterisation Trick, cont.

Reparameterisation Trick

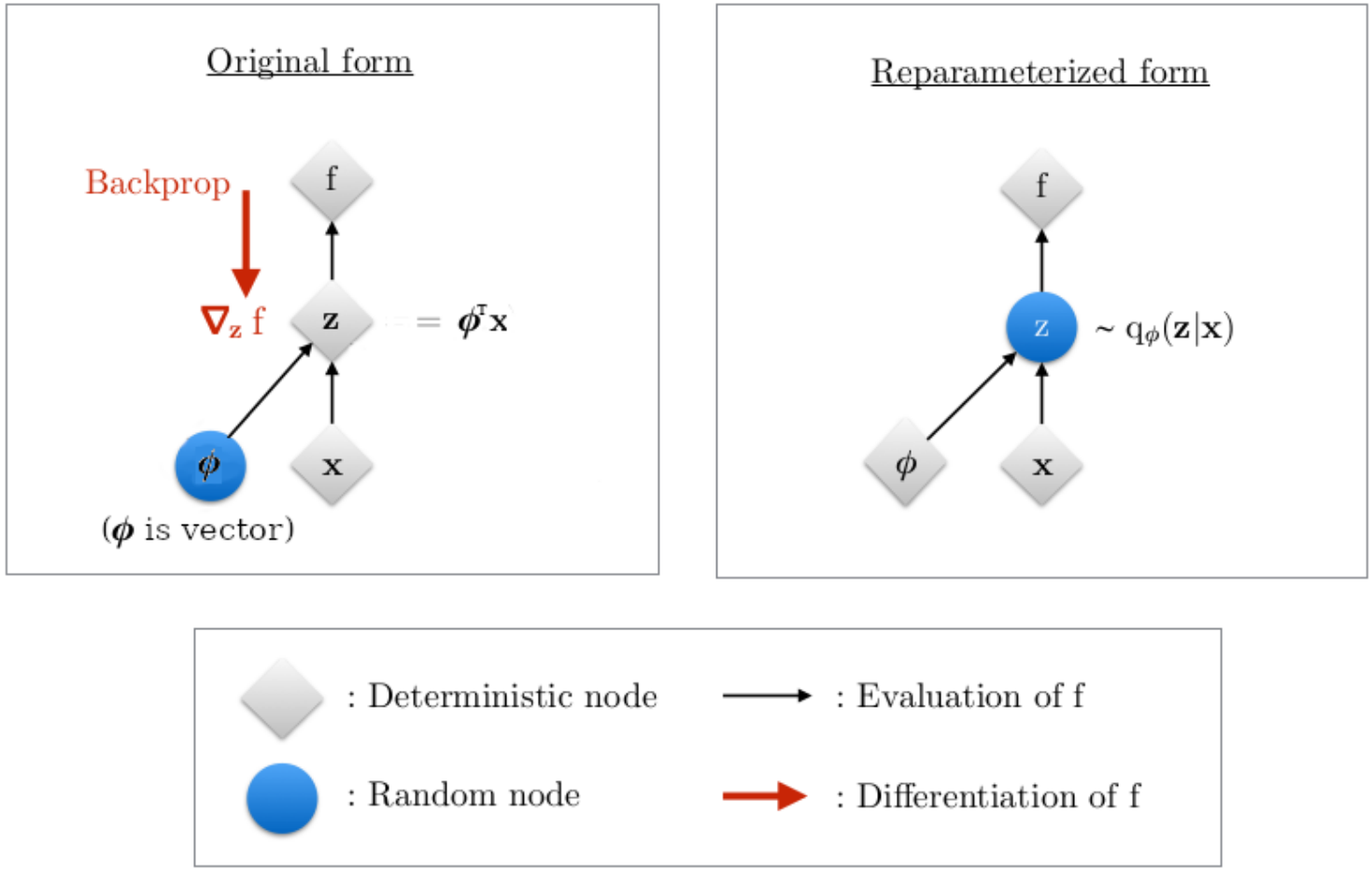
Let $g_{\boldsymbol{x},\phi}(\cdot)$ be an invertible function. A sample from the distribution $p(z|\boldsymbol{x},\phi)$ is equivalent to the function evaluation $z = g_{\boldsymbol{x},\phi}(\epsilon)$ where ϵ has the distribution $p(\epsilon)$ if and only if

$$p(z|\boldsymbol{x},\phi) = p(\epsilon{=}g^{-1}_{\boldsymbol{x},\phi}(z))\frac{\mathrm{d}g^{-1}_{\boldsymbol{x},\phi}(z)}{\mathrm{d}z} \ .$$

This limits the form used, but the recipe is simple.

1. select any real valued parameter-free distribution $p(\epsilon)$
   - Kingma and Welling used a standard normal distribution.
2. select any invertible function $g_{\boldsymbol{x},\phi}(\epsilon)$

NNs and Stochastic Weights



modified from *"Intro to VAEs"* Kingma and Welling 2019, p.23

A large stochastic vector of weights requires a lot of sampling.
But we can shift the randomness into the single real z.

NNs and Stochastic Weights, cont.

Eliminating Stochastic Weights

Let $z = \boldsymbol{\phi}^T \boldsymbol{x}$ be a node in a network where each ϕ_i is IID. Then the *characteristic function* of z, $\psi_z(t)$ is given by

$$\psi_z(t) = \prod_i \psi_{\phi_i}(tx_i)$$

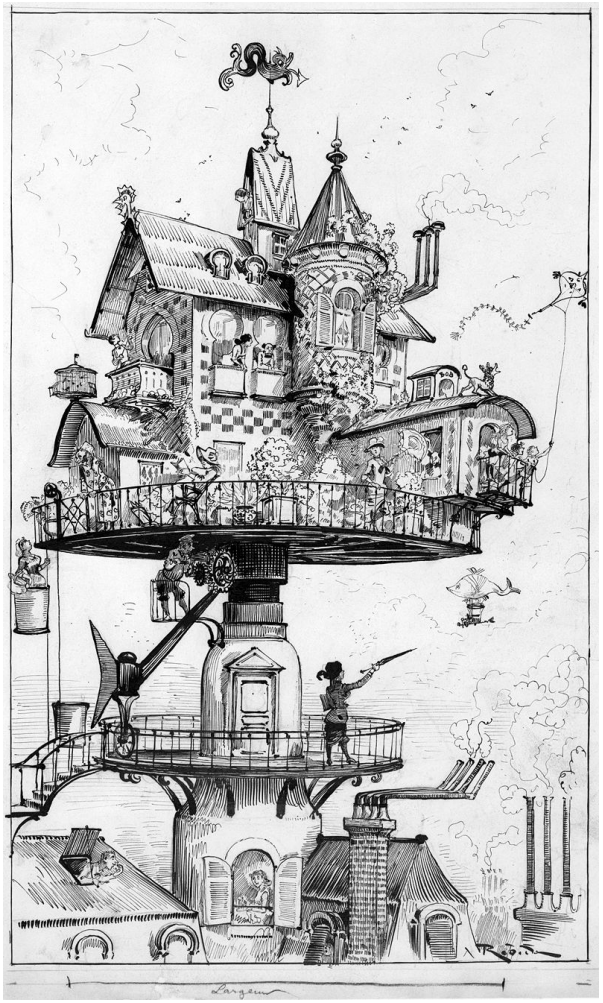
where $\psi_{\phi_i}(t)$ is the characteristic function of ϕ_i.

Suitable distributions are rare.

While you need to understand characteristic functions to apply this, for our purposes, the theorem yields

- $\phi_i \sim N(\mu_i, \sigma_i^2)$ then $z \sim N\left(\sum_i \mu_i x_i, \sum_i \sigma_i^2 x_i^2\right)$
- same for the Cauchy (which can be derived from a Gaussian)
- $\phi_i \sim \text{gamma}(\lambda_i)$ then $z \sim \text{gamma}\left(\sum_i \lambda_i x_i\right)$

Outline



Evaluation: Running on a Test Set I

Step I: Get test set with given labels and build clusters without the labels:

- Have a set of N data with **known single labels** and convert to indices $1, ..., L$, to get $l_1, ..., l_N$.
  - each datum has a single label,
  - labels should be "characteristic" of the data
- Cluster data (but ignoring labels) into K clusters numbered $1, 2, ..., K$, represented as cluster assignments $c_1, ..., c_N$.

Evaluation: Running on a Test Set II

Step 2: Evaluate by comparing labels to clusters:

- How do we match labels $1, ..., L$ to clusters $1, 2, ..., K$?
  - remember, both need to be treated as partitions
  - is a variation of the *Hungarian assignment problem*

Purity: run Hungarian assignment to get best matches for labels and clusters, then compute how pure each cluster is in terms of its label.

Entropy: Computing $I(l|c)$ gives how informative the clusters are about the labels

- treating both variables as partitions
- without requiring fixed assignments between clusters and labels

Evaluating Generative Models

- Have a training set of N data, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, and a test set of M data, $\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_M$,
- Build generative model $p(\boldsymbol{x}|\boldsymbol{\Theta})$ from the data.
- Compute average log probability on the test set with the latent variables marginalised out:

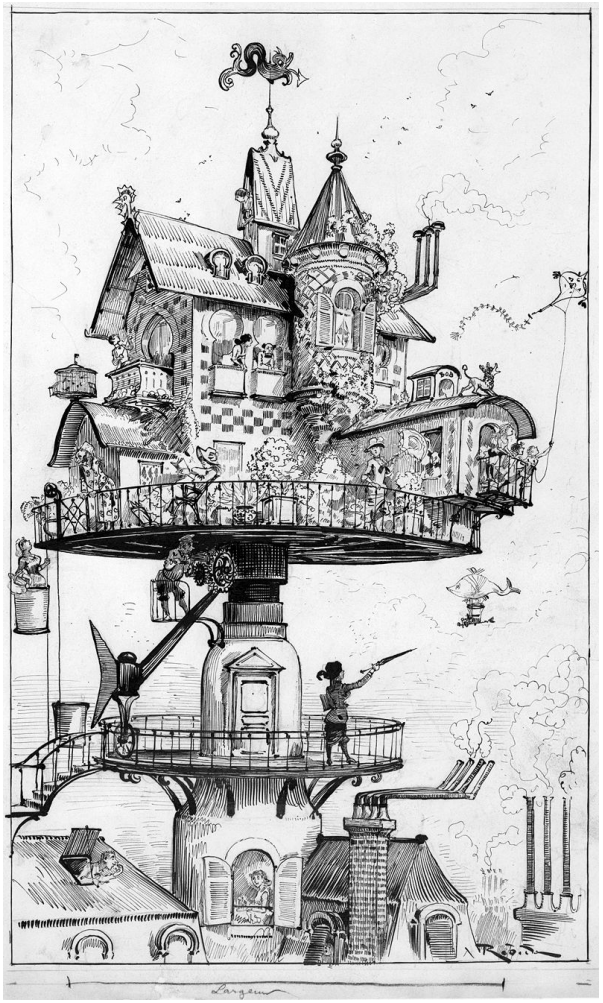
$$\textbf{score} = \frac{1}{M} \sum_{i=1}^{M} \log p(\boldsymbol{x}'_i|\boldsymbol{\Theta})$$

  - A challenge! See *"Estimating Likelihoods for Topic Models"* Buntine 2009.
- If data is multi-level, for instance each datum is a sequence of D tokens, so $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D_i})$, standardise to a **per token score** for a single test datum

$$\textbf{perplexity}_i = e^{\frac{1}{D} \log \frac{1}{p(\boldsymbol{x}'_i|\boldsymbol{\Theta})}} = \frac{1}{(p(\boldsymbol{x}'_i|\boldsymbol{\Theta}))^{\frac{1}{D}}}$$

- These scores are **not** inherently related to any real task.

Outline

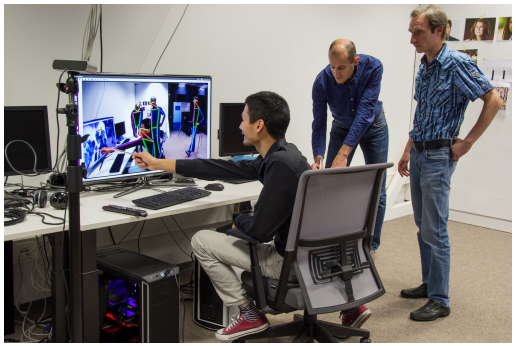


Justifications in Machine Learning

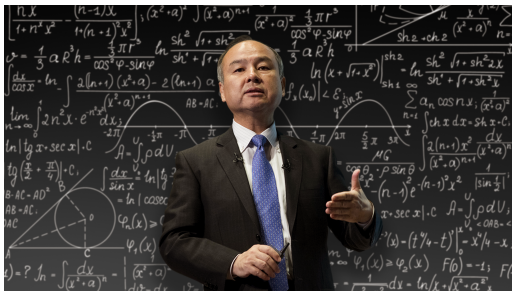
Justifications for ML methods fall into 3 camps:

- because we **can**

- because we **should**



- because we **must**



Justifications in Machine Learning, cont.

Justifications for ML methods fall into 3 camps:

Because we can: **expediency** from making huge assumptions or taking huge short cuts:

- fast algorithms (k-means),
- simple models (mixture of Gaussians),
- most early statistical methods (PCA)

Because we should: **reasonable arguments** show the method is appropriate:

- deep neural networks applied to complex problems like vision, natural language and speech
- when doing summarisation, not statistical inference, methods change

Because we must: some **theory claims** its the "right thing" to do

- early statistical methods in specific contexts
- following Bayesian-like practices in specific contexts
  - e.g. regularisation
    - Bayesian theory does not say you have to be Bayesian

Levels of Complexity in Modelling

Parametric: Exponential family and related known distributions: Gaussians, negative binomomials. **pre-1980's**

- well understood statistical inference algorithms

Non-parametric: Parametric models but with infinite vectors (e.g., Gaussian process, beta process, etc.) **1995-2000s**

- non-parametric extensions of well understood statistical inference algorithms

Latent variables and hierarchical models: Combinations of the above with decomposable models. **1990s-2000s**

- iterative approximation algorithms

Non-linear and non-exponential family: neural networks of various forms **2000s on**

- brute force inference algorithms

General hierarchical: Combinations of all the above. **2020 on**

Tutorials

Good introductory tutorials.

- Very basic tutorials I give less mathematical students from: *StatQuest*: *K-means clustering* and *hierarchical clustering*
- Good (generally basic) tutorial articles in blog form *Medium.com*
- Amr Ahmed's MLSS 2014 tutorial on parallelising LDA, big data and inference, *part 1* *part 2*

More advanced theory

- Peter Orbanz's *Lecture Notes on Bayesian Nonparametrics*, with PDF booklet from MLSS 2012
  - a good intro. for ML folks; ask me if you want more statistics
- Spectral clustering is based around using eigenvectors: *Short Introduction to Spectral Clustering*, from Hein and von Luxburg, MLSS 2007
- *"A Tutorial on NMF with Applications ...,"* Essid and Ozerov, *ICME*, 2014

Community Research Styles in ML

from Caitlin Doogan's PhD, 2019

Theoretical: concerned with proving theorems.

Methodological: concerned with general community philosophies like Deep Neural Networks, Bayesian, etc.

Empirical: concerned algorithms and their performance, often on standard test benchmarks

Applications: concerned with specific problems in practice

- there is general drift from one to another
- each community has its own special challenges and oftentimes look down on the others!
- each community has its own special publication styles
- see tutorial by Eamonn Keogh (from empirical school) *How to Do Good Data Mining Research and Get it Published*

Brief Guide to the Literature

ICML+NIPS: highbrow machine learning with less empirical work, mainly **methodological**

SIGKDD+ICDM: machine learning with more empirical work and some applications, mainly **empirical**

ICLR: SOTA deep neural nets, mainly **methodological**

various NLP + Vision + IR: diversity and depth in machine learning, **empirical** and **applications** and **methodological**

Example: The Hierarchical Dirichlet Process (HDP)

Teh, Jordan, Beal & Blei, 2006 (3000+ citations)

4.2 The Chinese restaurant franchise

In this section we describe an analog of the Chinese restaurant process for hierarchical Dirichlet processes that we refer to as the *Chinese restaurant franchise*. In the Chinese restaurant franchise, the metaphor of the Chinese restaurant process is extended to allow multiple restaurants which share a set of dishes.

The metaphor is as follows (see Figure 2). We have a restaurant franchise with a shared menu across the restaurants. At each table of each restaurant one dish is ordered from the menu by the first customer who sits there, and it is shared among all customers who sit at that table. Multiple tables in multiple restaurants can serve the same dish.

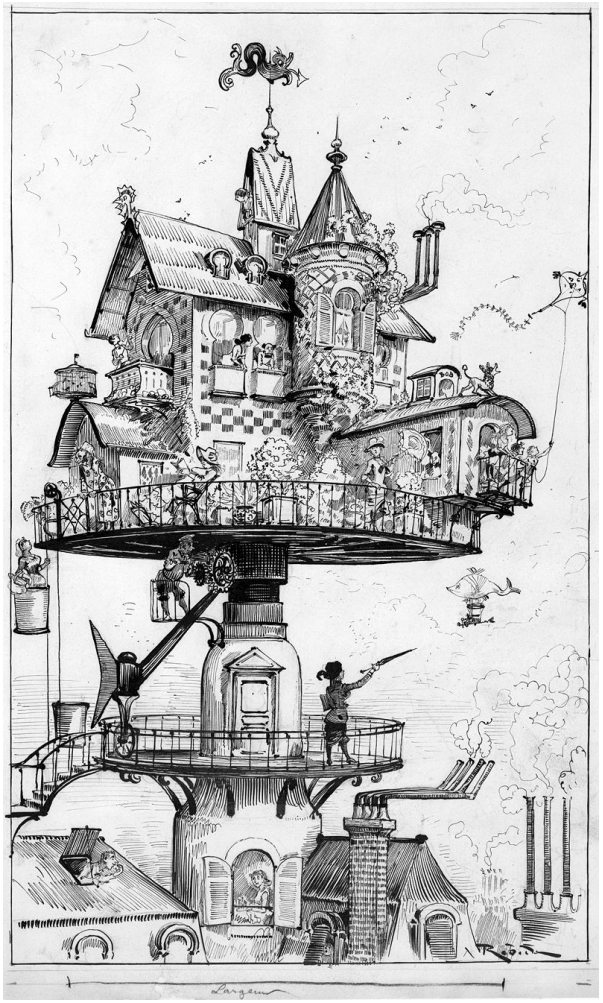
In this setup, the restaurants correspond to groups and the customers correspond to the factors θ_{ji}. We also let $\phi_1, \ldots, \phi_K$ denote K i.i.d. random variables distributed according to H; this is the global menu of dishes. We also introduce variables ψ_{jt} which represent the table-specific choice of dishes; in particular, ψ_{jt} is the dish served at table t in restaurant j.

Note that each θ_{ji} is associated with one ψ_{jt}, while each ψ_{jt} is associated with one ϕ_k. We introduce indicators to denote these associations. In particular, let t_{ji} be the index of the ψ_{jt} associated with θ_{ji}, and let k_{jt} be the index of ϕ_k associated with ψ_{jt}. In the Chinese restaurant franchise metaphor, customer i in restaurant j sat at table t_{ji} while table t in restaurant j serves dish k_{jt}.

We also need a notation for counts. In particular, we need to maintain counts of customers and counts of tables. We use the notation n_{jtk} to denote the number of customers in restaurant j at table t eating dish k. Marginal counts are represented with dots. Thus, $n_{jt\cdot}$ represents the number of customers in restaurant j at table t and $n_{j\cdot k}$ represents the number of customers in restaurant j eating dish k. The notation m_{jk} denotes the number of tables in restaurant j serving dish k. Thus, $m_{j\cdot}$ represents the number of tables in restaurant j, $m_{\cdot k}$ represents the number of tables serving dish k, and $m_{\cdot\cdot}$ the total number of tables occupied.

Let us now compute marginals under a hierarchical Dirichlet process when G_0 and G_j are

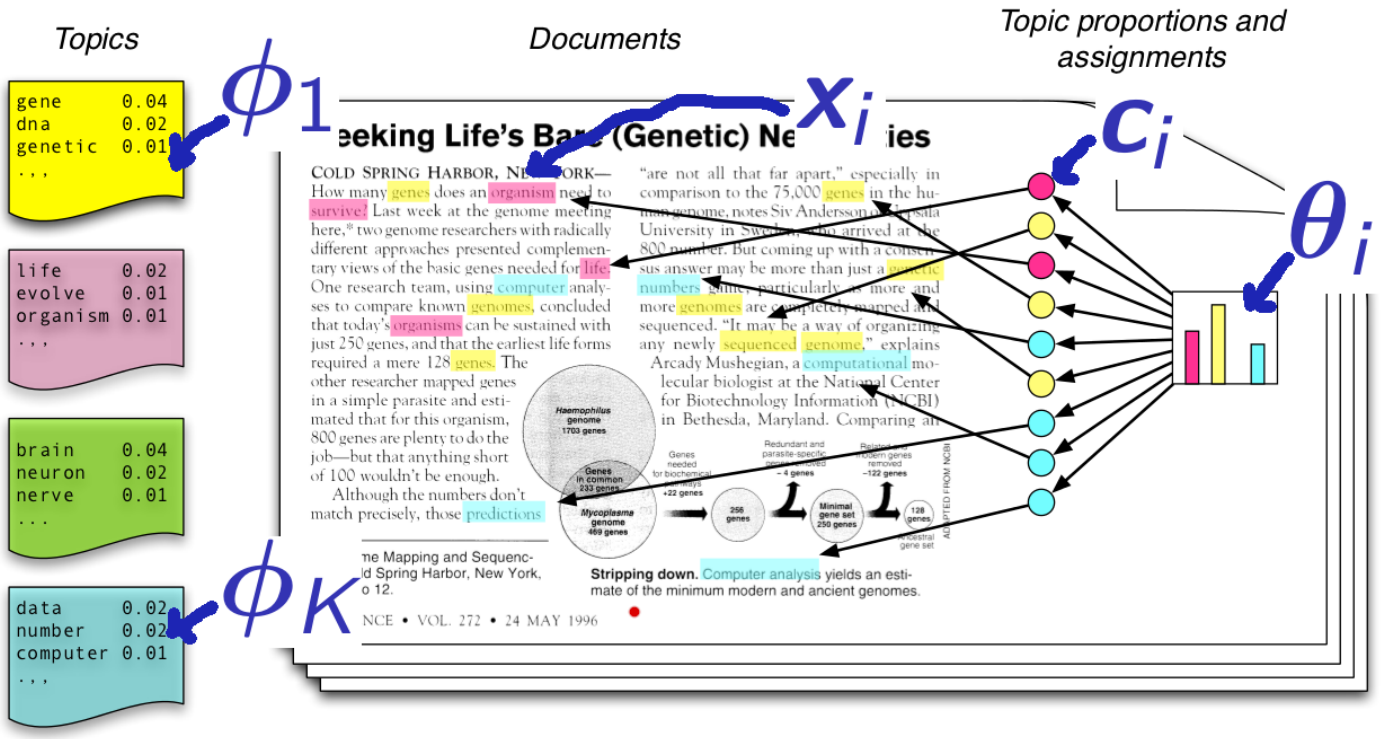
Outline



Framing the Problem

- **Data:** may have real/positive/integer/Boolean data, or rows may be normalised
- **Algorithm:** there are distributed, approximate/accurate, fast, a Python/Java/R variants of algorithms
- **Internet:** there are lots of poor algorithms and papers
- **Effects:** there are lots of sophisticated augmentations:
  - dealing with missing data
  - dealing with side information
  - dealing with sparse/short documents
  - dealing with multiple matrices

Latent Dirichlet Allocation, Description



N, K number of documents, number of topics

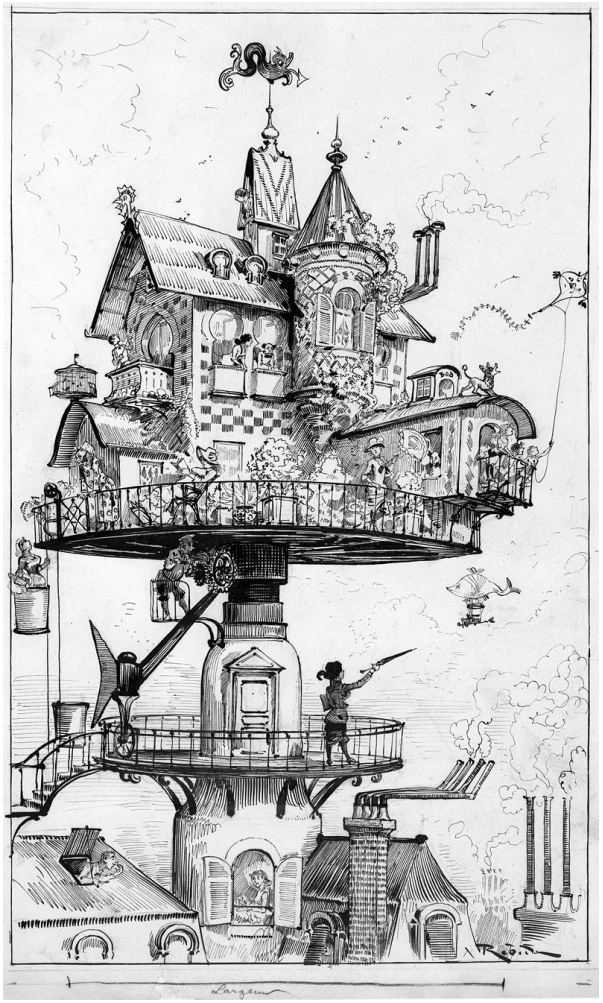
x_i list of (observed) words in document i, data

c_i list of topics for words in document i, latent variables

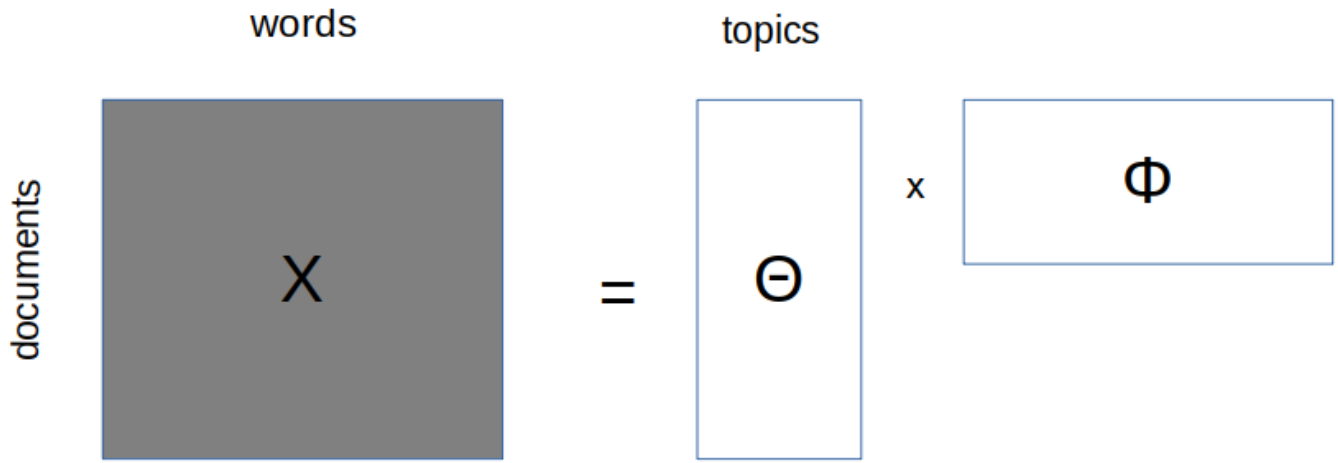
θ_i topic proportions for document i, a latent variable

ϕ_k word proportions for topic k, parameters

Outline



Starting with LDA

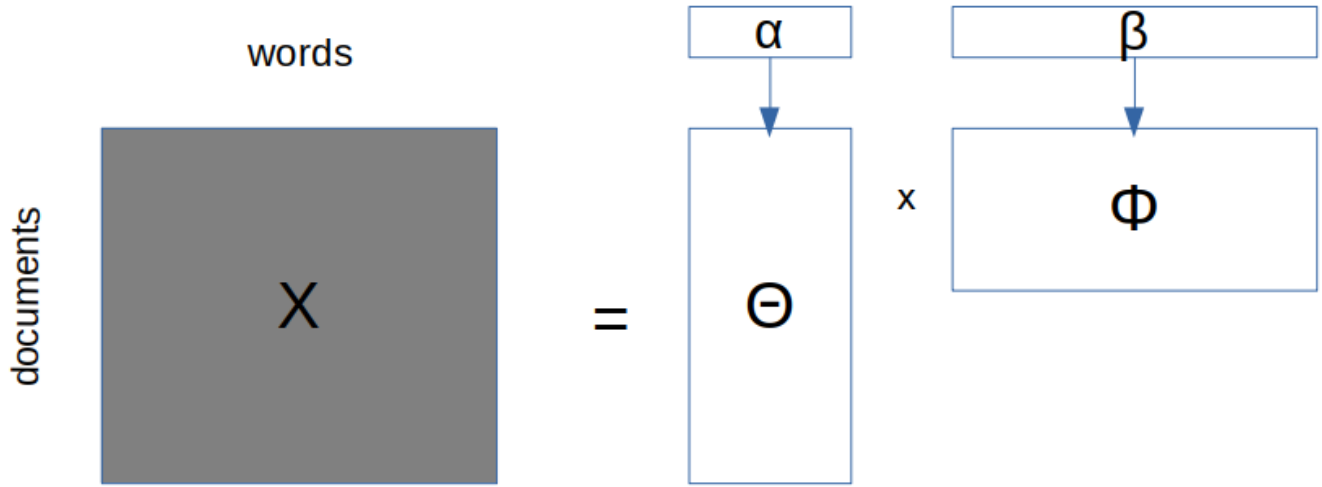


Basic LDA topic modelling with no effects (α and β are constants):

$$
\begin{aligned}
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\beta\mathbf{1}) && \text{for } k = 1, ..., K \\
\boldsymbol{\theta}_i &\sim \text{Dirichlet}(\alpha\mathbf{1}) && \text{for } i = 1, ..., N \\
\boldsymbol{x}_i &\sim \text{multinomial}\left(\boldsymbol{\theta}_i\boldsymbol{\Phi}\right) && \text{for } i = 1, ..., N
\end{aligned}
$$

NB. $\boldsymbol{c}$ has been marginalised out by the product $\boldsymbol{\theta}_i\boldsymbol{\Phi}$

Improving Topic Models: I



Add priors to $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ (γ and ψ are constants):

$$
\begin{aligned}
\boldsymbol{\beta} &\sim \text{Pitman-Yor}(\psi\mathbf{1}) \\
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\boldsymbol{\beta}) && \text{for } k = 1, ..., K \\
\boldsymbol{\alpha} &\sim \text{Dirichlet}(\gamma\mathbf{1}) \\
\boldsymbol{\theta}_i &\sim \text{Dirichlet}(\boldsymbol{\alpha}) && \text{for } i = 1, ..., N \\
\boldsymbol{x}_i &\sim \text{multinomial}\,(\boldsymbol{\theta}_i \boldsymbol{\Phi}) && \text{for } i = 1, ..., N
\end{aligned}
$$

Improving Topic Models: I

Different topics should have different base rates.

- Consider the following topics in news about "Obesity":
  - `say have obesity not health need problem issue` $\longrightarrow$ 10.7% of words
  - `christ religious faith jewish bless wesleyan` $\longrightarrow$ 0.08% of words
- Standard LDA says these two should be equally likely.

Improving Topic Models: I

Different topics should have different base rates.

- we make priors on the topic proportions asymmetric,
- done by Teh, Jordan, Beal and Blei 2006
  - spawned Hierarchical Dirichlet processes (HDP) and nested/hierarchical Chinese restaurants

Improving Topic Models: I

Different topics should have different base rates.

- we make priors on the topic proportions asymmetric,
- done by Teh, Jordan, Beal and Blei 2006
  - spawned Hierarchical Dirichlet processes (HDP) and nested/hierarchical Chinese restaurants
- done by Wallach, Mimno, McCallum 2009
  - now available in the Mallet topic modelling system

Improving Topic Models: I

Different topics should have different base rates.

- we make priors on the topic proportions asymmetric,
- done by Teh, Jordan, Beal and Blei 2006
  - spawned Hierarchical Dirichlet processes (HDP) and nested/hierarchical Chinese restaurants
- done by Wallach, Mimno, McCallum 2009
  - now available in the Mallet topic modelling system
- considerable theory and algorithms, 2009-2012
  - noteable mention: Bryant and Sudderth, 2012
  - but some implementations gave poor results

Improving Topic Models: I

Different topics should have different base rates.

- we make priors on the topic proportions asymmetric,
- done by Teh, Jordan, Beal and Blei 2006
  - spawned Hierarchical Dirichlet processes (HDP) and nested/hierarchical Chinese restaurants
- done by Wallach, Mimno, McCallum 2009
  - now available in the Mallet topic modelling system
- considerable theory and algorithms, 2009-2012
  - noteable mention: Bryant and Sudderth, 2012
  - but some implementations gave poor results
- done by Buntine and Mishra, *KDD*, 2014
  - does HDP efficiently with a fast Gibbs sampler
  - multi-core, great results
  - Gibbs sampling beats variational inference!

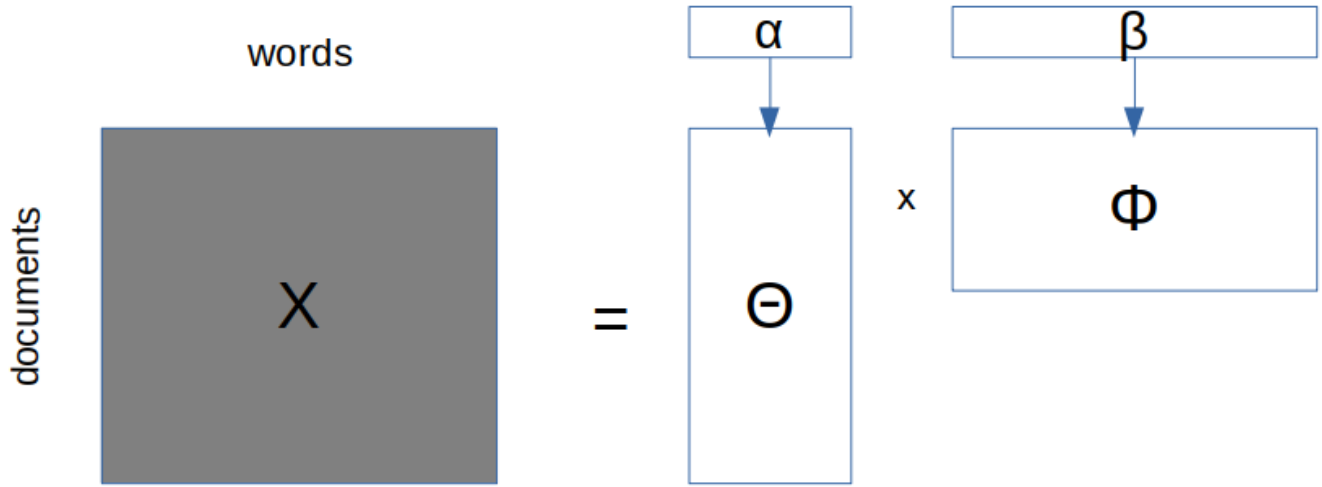
Yields High Fidelity Topics

Examples from 100 topics about "Obesity in the ABC news" from 2003-2012, from 600 news articles of average length 150 words:

rank		words
5	4.57%	study researcher finding journal publish twice university
14	1.54%	teenager boy child adults parent youngster bauer school-child
22	0.86%	doctor ambulance hospital psychiatric general-practitioner staff
42	0.43%	soft-drink instant soda carbonated fizzy beverages candy sugary
78	0.18%	olympics time second olympic pool win team freestyle gold
91	0.11%	colonel lieutenant-general afghanistan rifle stirling mission
95	0.10%	dialysis end-stage dementia kidney-disease kidney abdominal

- 100 topics for 600 documents
- most are on coherent subjects

Improving Topic Models: II



Make the generation of $\boldsymbol{x}_i$ more robust:

$$
\begin{aligned}
\boldsymbol{\beta} &\sim \text{Pitman-Yor}(\psi \mathbf{1}) \\
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\boldsymbol{\beta}) && \text{for } k = 1, ..., K \\
\boldsymbol{\alpha} &\sim \text{Dirichlet}(\gamma \mathbf{1}) \\
\boldsymbol{\theta}_i &\sim \text{Dirichlet}(\boldsymbol{\alpha}) && \text{for } i = 1, ..., N \\
\boldsymbol{x}_i &\sim \text{Dirichlet-multinomial}\,(\boldsymbol{\theta}_i \boldsymbol{\Phi}) && \text{for } i = 1, ..., N
\end{aligned}
$$

Improving Topic Models: II

Words in text are bursty: they appear in small bursts.

Original news article:

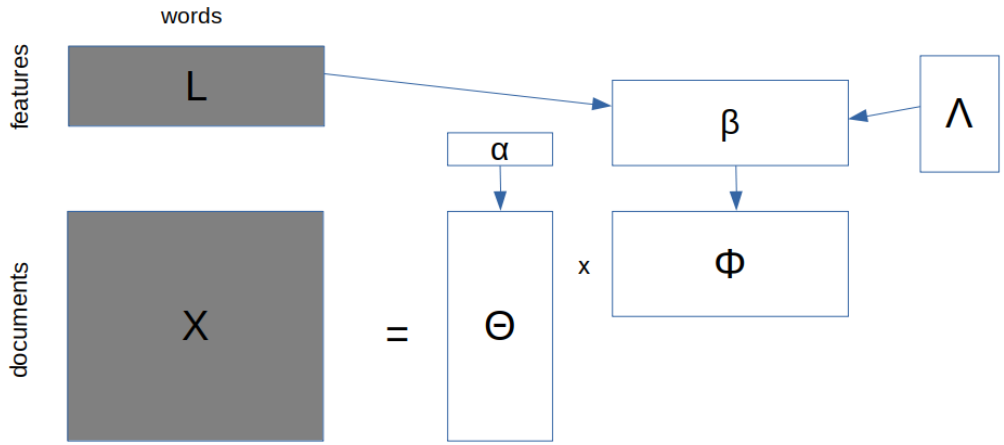
Women may only account for 11% of all Lok-Sabha MPs but they fared better when it came to representation in the Cabinet. Six women were sworn in as senior ministers on Monday, accounting for 25% of the Cabinet. ...

Bag of words:

11% 25% Cabinet(2) Lok-Sabha MPs Monday Six They Women account accounting all and as better but came fared for(2) in(2) it may ministers of on only representation senior sworn the(2) to were when women

- effect is called burstiness
- first modelled by Doyle and Elkan 2009, but intolerably slow
- done by Buntine and Mishra, KDD, 2014 using HDPs
  - only 25% (or so) penalty in memory and time
  - **huge** improvement in perplexity, and smaller one in coherence
  - but loss of fidelity ("fine" low probability topics)
    - so we usually don't use
- Also used in Poisson matrix factorisation by Mingyuan Zhou using negative binomials instead of Poisson.

Improving Topic Models: III

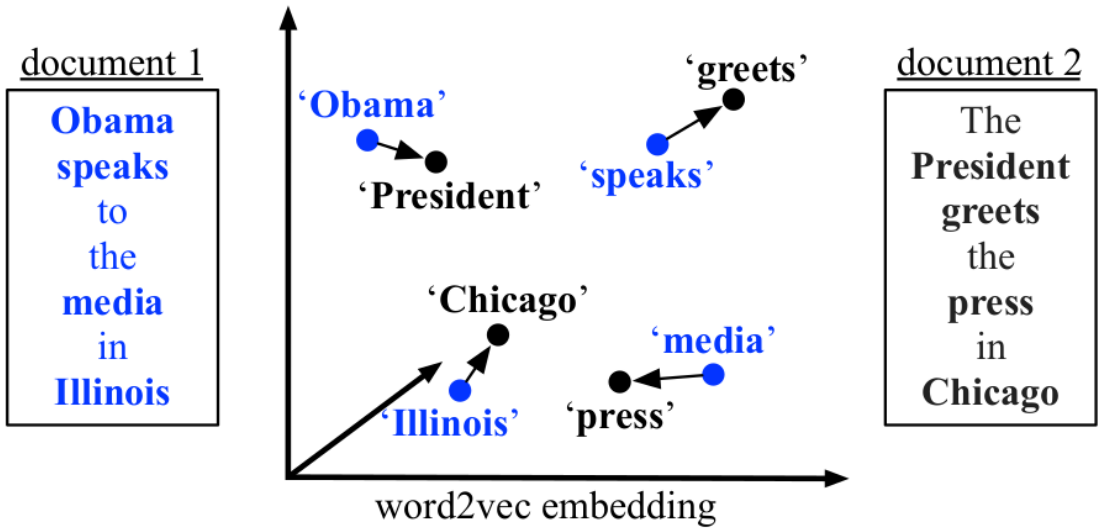


Regress word features $\boldsymbol{L}$ onto topic prior β_k to customize it for each topic.

$$
\begin{aligned}
\beta_k &\sim \text{Gamma-Regression}(\Lambda, \boldsymbol{L}) \\
\phi_k &\sim \text{Dirichlet}(\beta_k) && \text{for } k = 1, ..., K \\
\boldsymbol{\alpha} &\sim \text{Dirichlet}(\gamma \mathbf{1}) \\
\boldsymbol{\theta}_i &\sim \text{Dirichlet}(\boldsymbol{\alpha}) && \text{for } i = 1, ..., N \\
\boldsymbol{x}_i &\sim \text{multinomial}\,(\boldsymbol{\theta}_i \boldsymbol{\Phi}) && \text{for } i = 1, ..., N
\end{aligned}
$$

Improving Topic Models: III

Information about word similarity/semantics should be used when building topics.



from "An Introduction to Word Embeddings", blog by Roger Huang, 2017

- we use prior information about words from embeddings
- done recently by many in topic modelling and deep neural networks

ASIDE: Multi-Label Learning (MLL)

is the article written in Mandarin?

is the article about President Trump?

is the article about economics?

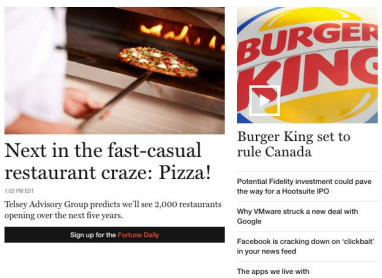
does the article have positive sentiment?

- same source data
- multiple labels
- one combined model/system to do it

ASIDE: Multi-Task Learning (MTL)

is the article written in Mandarin?

is the article about President Trump?



is the article about economics?

does the article have positive sentiment?

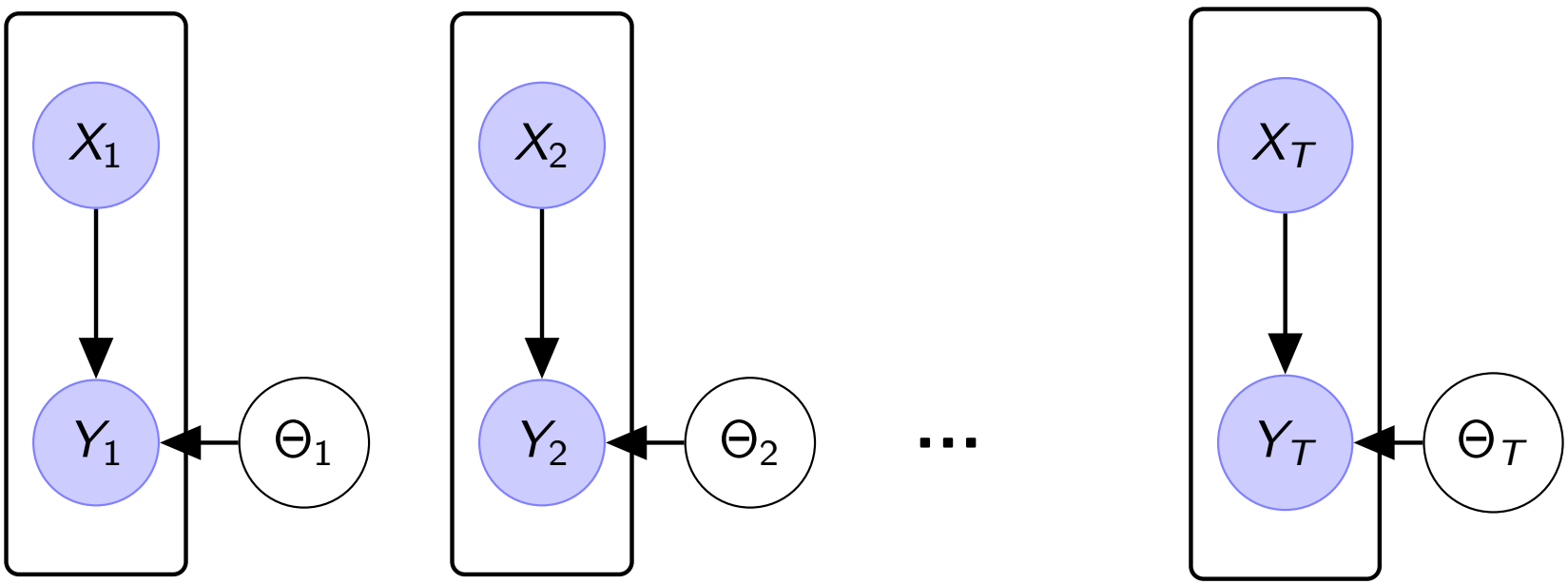
- different source data
- different labels or tasks
- one combined model/system to do it

ASIDE: Naive Multi-Task Learning

Have T somewhat related separate classification tasks.

Predict Y_t from X_t using parameters Θ_t.

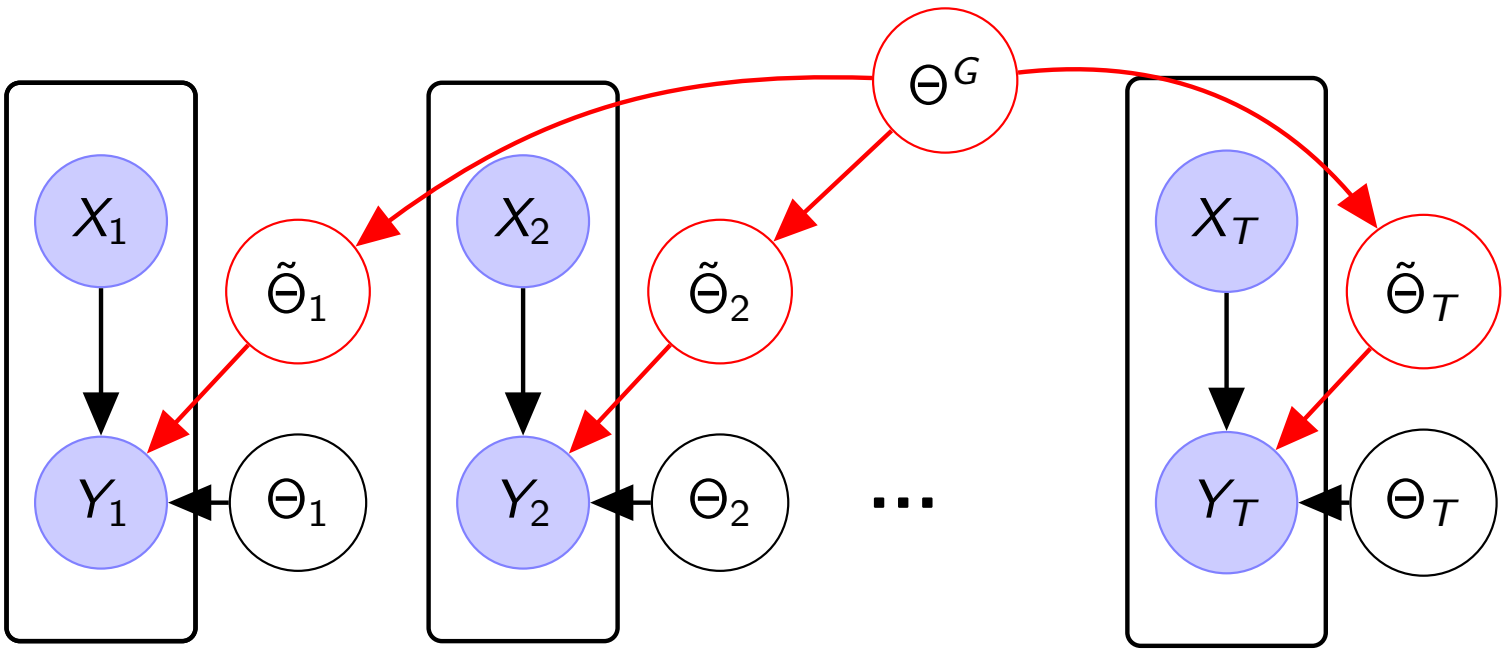
$$p(Y_t|X_t, \Theta_t) \qquad \text{for } t = 1, ..., T$$



ASIDE: Multi-Task Learning (MTL)

Add a shared parameter Θ^G which captures "common knowledge".

$$p(\tilde{\Theta}_t|\Theta^G) \qquad \text{for } t = 1, ..., T$$
$$p(Y_t|X_t, \Theta_t, \tilde{\Theta}_t) \qquad \text{for } t = 1, ..., T$$



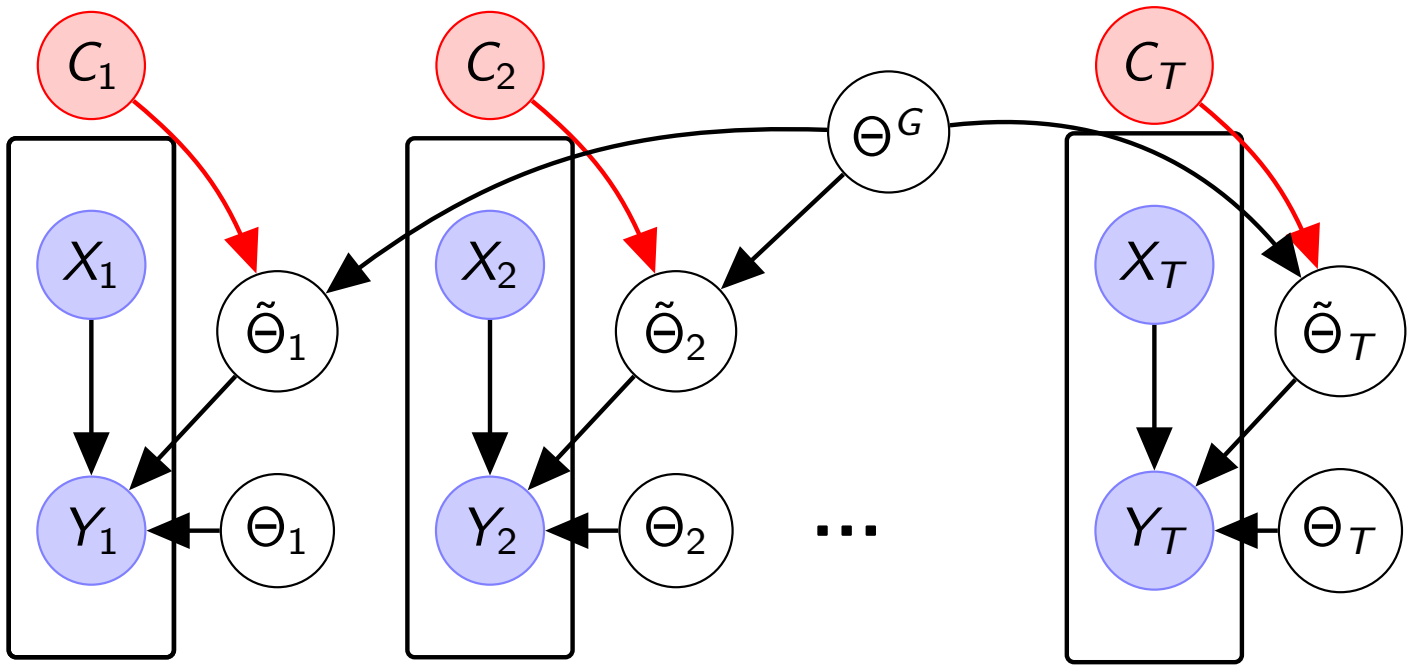
NB. another hierarchical model with Θ^G the parent node

Prior Regression for MTL

Regress from metadata C_t onto task-specific version of common knowledge $\tilde{\Theta}_t$, using parameters Θ^G.

$$p(\tilde{\Theta}_t | C_t, \Theta^G) \qquad \text{for } t = 1, ..., T$$

$$p(Y_t | X_t, \Theta_t, \tilde{\Theta}_t) \qquad \text{for } t = 1, ..., T$$



NB. in statistics, random effects models achieve this effect

Improving Topic Models: III

Information about word similarity/semantics should be used when building topics.

- we use prior information about words from embeddings
- done recently by many in topic modelling and deep neural networks
- done using prior regression by Zhao, Du, Buntine, Liu *ICDM* 2017, Zhao, Du, Buntine, *ACML* 2017
  - regress the metadata (e.g., word embeddings, document labels) onto the model parameters during learning
  - using fast "gamma regression"
  - *code available at He Zhao's GitHub repo*
  - very good results

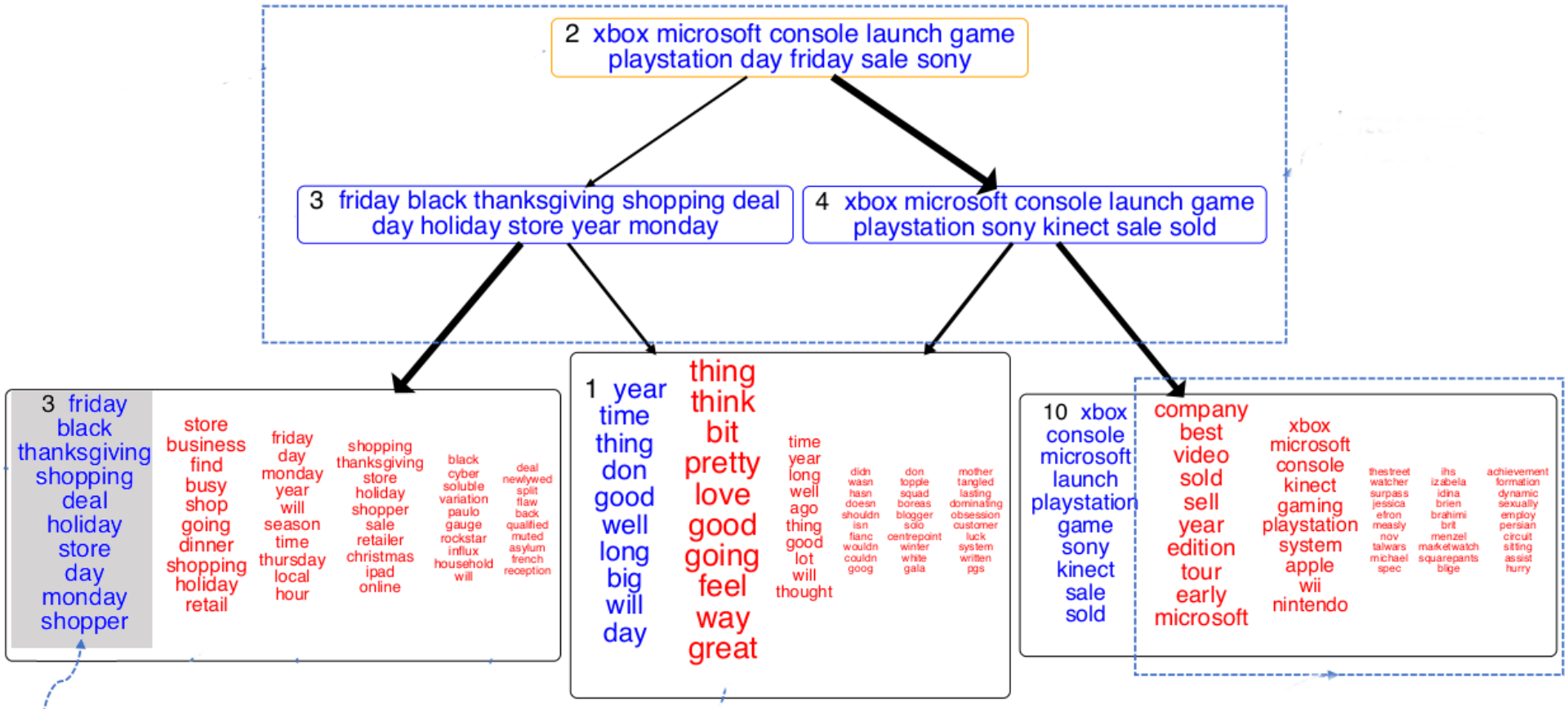
Improving Topic Models: IV

Hierarchical structure between topics should be discovered.

- once we go beyond 20 topics, this supports explanation

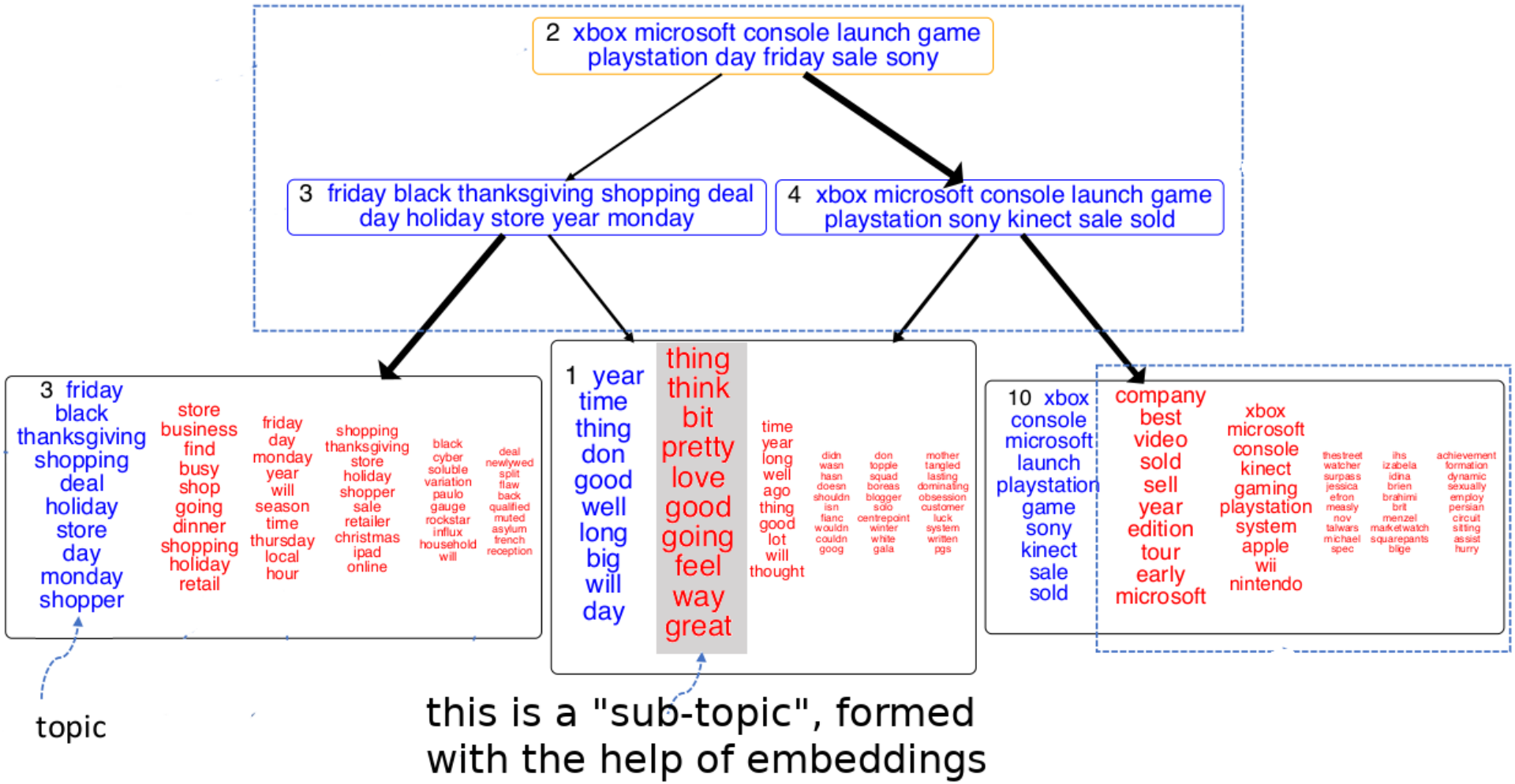
Topics Enhanced with Word Embeddings

Zhao, Du, Buntine, Zhou *ICML* 2018



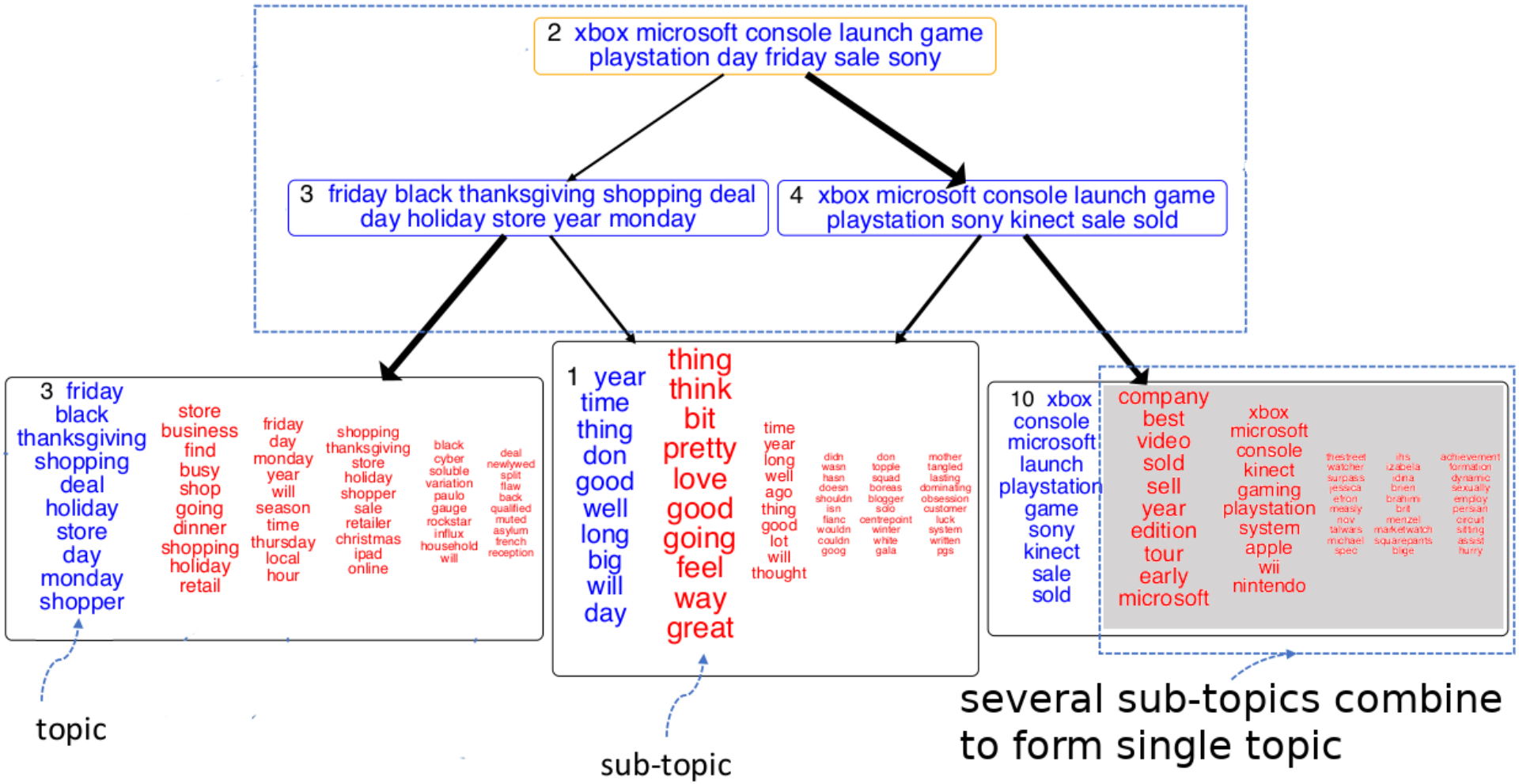
Topics Enhanced with Word Embeddings

Zhao, Du, Buntine, Zhou *ICML* 2018



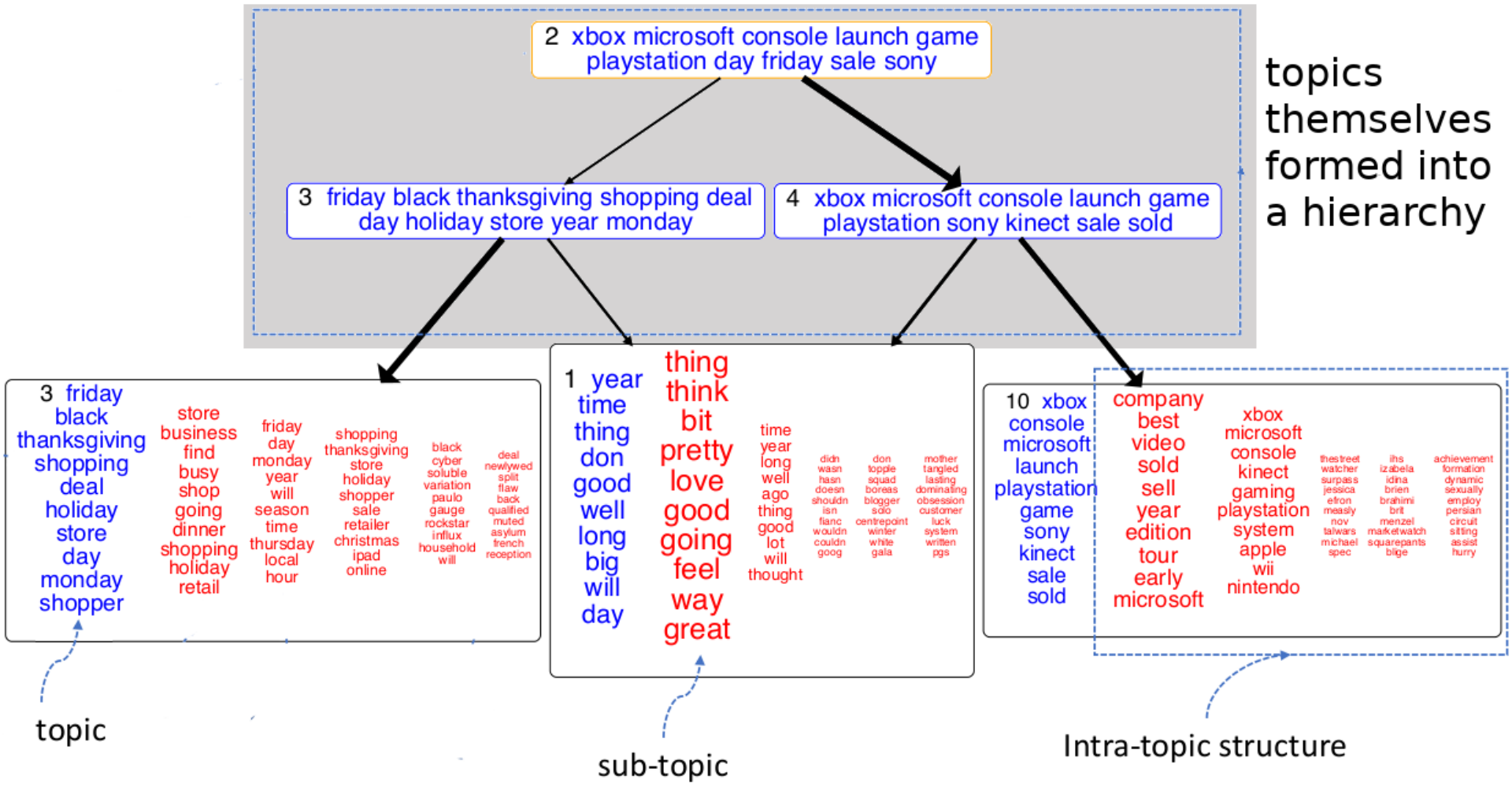
Topics Enhanced with Word Embeddings

Zhao, Du, Buntine, Zhou *ICML* 2018

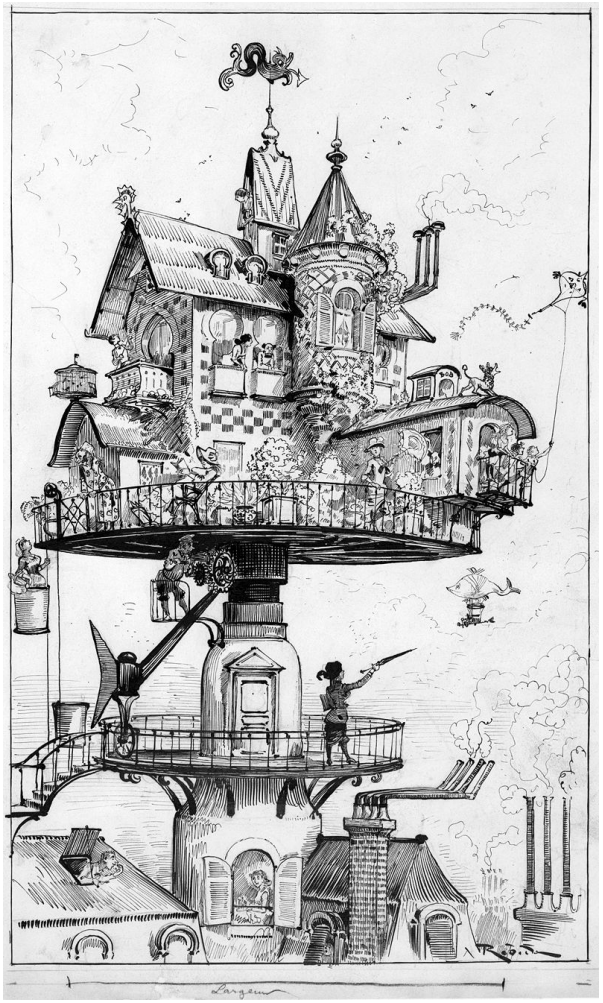


Topics Enhanced with Word Embeddings

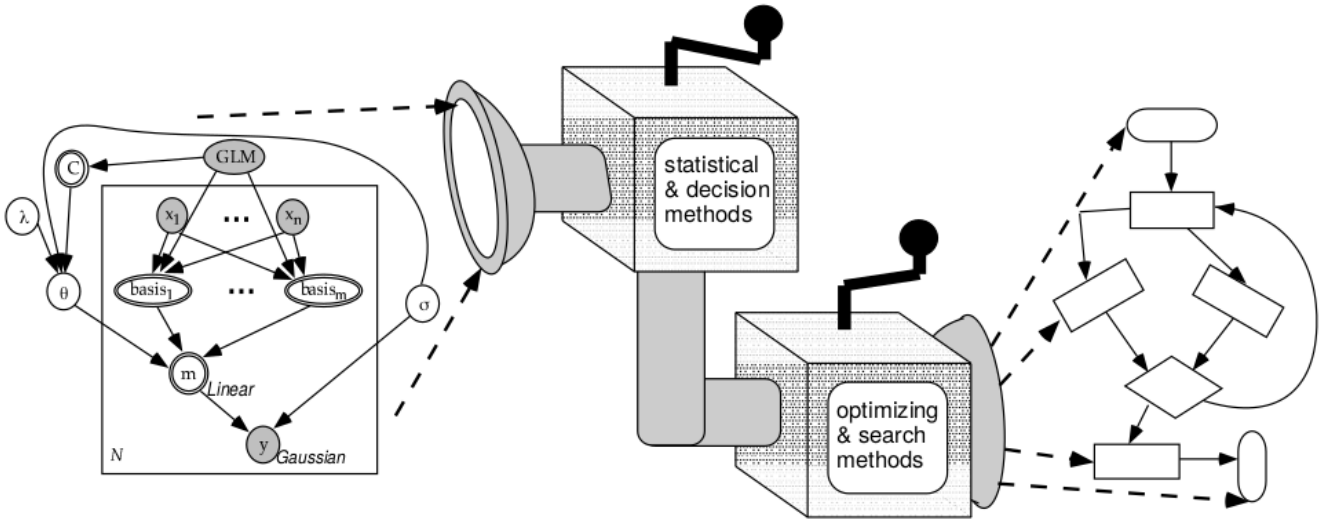
Zhao, Du, Buntine, Zhou *ICML* 2018



Outline



Putting Together an Algorithm



1. Build up a model including the various effects you may want:
   - ML folks increasingly mixing deep NNs and statistical models
2. Develop an algorithm mostly using standard/generalised techniques.
   - MCMC such as Gibbs
   - KL variational method
   - general autodiff and stochastic optimiser, etc.
3. Parallelise again mostly using standard techniques.

While its not easy, it is becoming more routine.

Varieties of Probabilistic MF

Table 3. Probabilistic models of matrix factorizations with $\mathbf{X} = \mathbf{WZ} + \mathbf{E}$.

Probabilistic Model	Random Variables	Prior Distributions	Solving Strategy
PMF [15]	$E_{ij} \sim \mathcal{N}(0, \sigma^2)$, $P(\mathbf{W}\|\sigma_{\mathbf{W}}^2) = \prod_{i=1}^{r} \mathcal{N}(\mathbf{w}_{\cdot i}\|0, \sigma_{\mathbf{W}}^2 \mathbf{I}_d)$, $P(\mathbf{Z}\|\sigma_{\mathbf{Z}}^2) = \prod_{j=1}^{N} \mathcal{N}(\mathbf{z}_{\cdot j}\|0, \sigma_{\mathbf{Z}}^2 \mathbf{I}_r)$.	-	MAP
Variational Bayesian PMF [14]	$E_{ij} \sim \mathcal{N}(0, \sigma^2)$, $W_{ik} \sim \mathcal{N}(0, \sigma_k^2)$, $Z_{kj} \sim \mathcal{N}(0, \rho_k^2)$.	-	VB
Bayesian PMF [16]	$E_{ij} \sim \mathcal{N}(0, \beta^{-1})$, $p(\mathbf{W}\|\boldsymbol{\mu}_{\mathbf{W}}, \boldsymbol{\Lambda}_{\mathbf{W}}) = \prod_{i=1}^{d} \mathcal{N}\left(\mathbf{w}_{i\cdot}\|\boldsymbol{\mu}_{\mathbf{W}}, \boldsymbol{\Lambda}_{\mathbf{W}}^{-1}\right)$, $p(\mathbf{Z}\|\boldsymbol{\mu}_{\mathbf{Z}}, \boldsymbol{\Lambda}_{\mathbf{Z}}) = \prod_{j=1}^{N} \mathcal{N}\left(\mathbf{z}_{\cdot j}\|\boldsymbol{\mu}_{\mathbf{Z}}, \boldsymbol{\Lambda}_{\mathbf{Z}}^{-1}\right)$.	$p(\Theta_{\mathbf{W}}\|\Theta_0) =$ $\mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{W}}\|\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda}_{\mathbf{W}})^{-1}\right) \mathcal{W}(\boldsymbol{\Lambda}_{\mathbf{W}}\|\mathbf{W}_0, v_0)$, $p(\Theta_{\mathbf{Z}}\|\Theta_0) =$ $\mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{Z}}\|\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda}_{\mathbf{Z}})^{-1}\right) \mathcal{W}(\boldsymbol{\Lambda}_{\mathbf{Z}}\|\mathbf{W}_0, v_0)$.	Gibbs sampling
Sparse Bayesian matrix completion [19]	$E_{ij} \sim \mathcal{N}(0, \beta^{-1})$, $p(\mathbf{W}\|\boldsymbol{\gamma}) = \prod_{i=1}^{r} \mathcal{N}\left(\mathbf{w}_{\cdot i}\|\gamma_i^{-1} \mathbf{I}_d\right)$, $p(\mathbf{Z}\|\boldsymbol{\gamma}) = \prod_{i=1}^{r} \mathcal{N}\left(\mathbf{z}_{i\cdot}\|\gamma_i^{-1} \mathbf{I}_r\right)$.	$\gamma_i \sim \mathrm{Gam}(a, b)$, $p(\beta) \propto \beta^{-1}$.	VB
Robust Bayesian matrix factorization [17]	$E_{ij} \sim \mathcal{N}(0, (\tau \alpha_i \beta_j)^{-1})$, $p(\mathbf{w}_{i\cdot}\|\alpha_i) = \mathcal{N}(\mathbf{w}_{i\cdot}\|\mathbf{0}, (\alpha_i \boldsymbol{\Lambda}_{\mathbf{w}})^{-1})$, $p(\mathbf{z}_{\cdot j}\|\beta_j) = \mathcal{N}(\mathbf{z}_{\cdot j}\|\mathbf{0}, (\beta_j \boldsymbol{\Lambda}_{\mathbf{z}})^{-1})$.	$p(\alpha_i) = \mathrm{Gam}(\alpha_i\|a_0/2, b_0/2)$, $p(\beta_j) = \mathrm{Gam}\left(\beta_j\|c_0/2, d_0/2\right)$.	VB, type II ML, empirical Bayes

from "Survey on ... Low-Rank Matrix Factorizations," Shi, Zheng & Wang, *Entropy*, 2017

Varieties of Infinite Vectors

from "Understanding Hierarchical Processes," Buntine, forthcoming

Vector Process	Normalised	Hierarchical
beta process$(M, 0, 1)$		Dickman(M)
gamma process$(M, 0, \beta)$	Dirichlet proc.	gamma(M, β)
gen. gamma proc.(M, α, β)	Pitman-Yor proc.	Tweedie$(\alpha, M^{1/\alpha}, \beta)$
stable process(M, α)	PYP end case	pstable$(\alpha, M^{1/\alpha})$
Poisson process(M)	multinomial proc.	Poisson(M)
neg. binomial proc.(M, ρ)	?	neg.bin(M, ρ)

- for most, inference generalises standard parametric methods using the hierarchical version
- for some, need generalised Chinese restaurant processes

Practical Advice

- for clustering text, TF/IDF (specifically, BM25) is a great distance, so k-means works well
- for multi-fielded record data, k-means not good, model-based theory exists, way too many statisticians projects
- for topic modelling, I recommend MetaLDA (on He Zhao's Github page), though Mallet is better supported
- there is good NMF software, but I have no experience
- embeddings, a crowded field: I use GloVe, great software and prebuilt sets
- very good heterogenous graph embedding methods in R&D now

Questions?

Figure source: http://milewalk.com/mwblog/4-worst-job-interview-questions/