Probabilistic Machine Learning
Lecture 21
Efficient Inference & Mixture Models

Philipp Hennig

5 July 2021

Eberhard Karls Universität Tübingen
Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Designing a probabilistic machine learning method:

1. get the **data**
   1.1 try to collect as much meta-data as possible
2. build the **model**
   2.1 identify quantities and datastructures; assign names
   2.2 design a generative process (graphical model)
   2.3 assign (conditional) distributions to factors/arrows (use exponential families!)
3. design the **algorithm**
   3.1 consider conditional independence
   3.2 try standard methods for early experiments
   3.3 run unit-tests and sanity-checks
   3.4 identify bottlenecks, find customized approximations and refinements

The Toolbox

Framework:

$$\int p(x_1, x_2)\, dx_2 = p(x_1) \qquad p(x_1, x_2) = p(x_1 \mid x_2)p(x_2) \qquad p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}$$

Modelling:

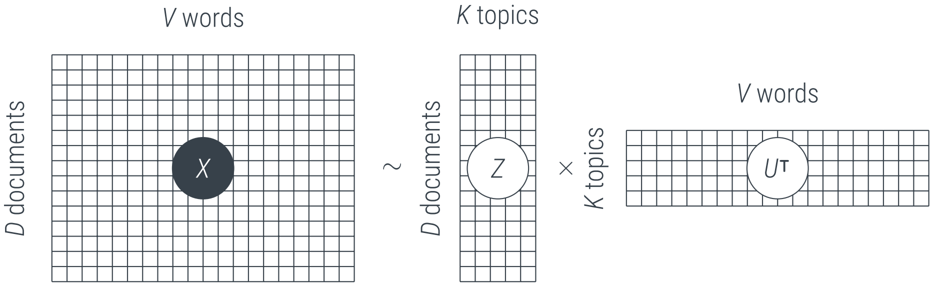
- graphical models
- Gaussian distributions
- (deep) learnt representations
- Kernels
- Markov Chains
- Exponential Families / Conjugate Priors
- Factor Graphs & Message Passing

Computation:

- Monte Carlo
- Linear algebra / Gaussian inference
- maximum likelihood / MAP
- Laplace approximations
- EM / variational approximations

Making Assumptions

Our Data, our model

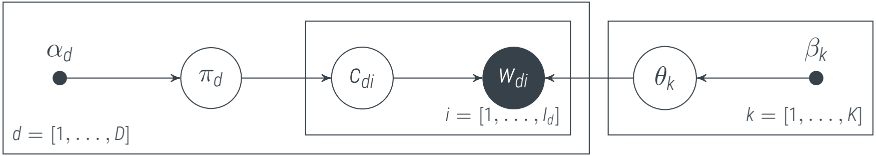


- a corpus of D documents
- each containing I_d words from a vocabulary of V words
- assumed to consist of K topics

Latent Dirichlet Allocation

Topic Models

[Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003) JMLR 3, 993–1022]



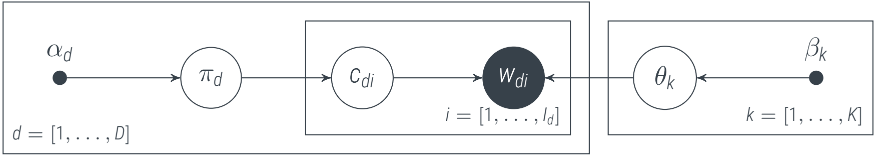
To draw I_d words $w_{di} \in [1, \dots, V]$ of document $d \in [1, \dots, D]$:

- Draw K topic distributions θ_k over V words from $p(\Theta \mid \boldsymbol{\beta}) = \prod_{k=1}^{K} \mathcal{D}(\theta_k; \beta_k)$
- Draw D document distributions over K topics from $p(\Pi \mid \boldsymbol{\alpha}) = \prod_{d=1}^{D} \mathcal{D}(\pi_d; \alpha_d)$
- Draw topic assignments c_{dik} of word w_{di} from $p(C \mid \Pi) = \prod_{i,d,k} \pi_{dk}^{c_{dik}}$
- Draw word w_{di} from $p(w_{di} = v \mid c_{di}, \Theta) = \prod_k \theta_{kv}^{c_{dik}}$

Useful notation: $n_{dkv} = \#\{i : w_{di} = v, c_{dik} = 1\}$. Write $n_{dk:} := [n_{dk1}, \dots, n_{dkV}]$ and $n_{dk\cdot} = \sum_v n_{dkv}$, etc.

The Joint

[Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003) JMLR 3, 993–1022]



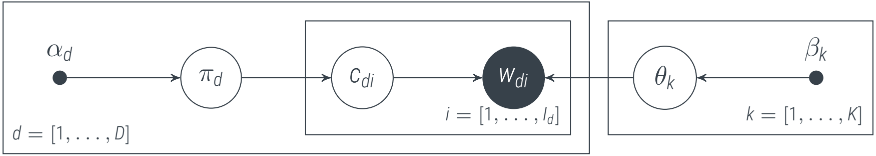
$$p(C, \Pi, \Theta, W) = \underbrace{\left(\prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\pi}_d; \boldsymbol{\alpha}_d)\right)}_{p(\Pi|\boldsymbol{\alpha})} \cdot \underbrace{\left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \pi_{dk}^{c_{dik}}\right)\right)}_{p(C|\Pi)} \cdot \underbrace{\left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \theta_{kw_{di}}^{c_{dik}}\right)\right)}_{p(W|C,\Theta)} \cdot \underbrace{\left(\prod_{k=1}^{K} \mathcal{D}(\boldsymbol{\theta}_k; \boldsymbol{\beta}_k)\right)}_{p(\Theta|\boldsymbol{\beta})}$$

$$= \underbrace{\left(\prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\pi}_d; \boldsymbol{\alpha}_d)\right)}_{p(\Pi|\boldsymbol{\alpha})} \cdot \underbrace{\left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} (\pi_{dk}\theta_{kw_{di}})^{c_{dik}}\right)\right)}_{p(W,C|\Theta,\Pi)} \cdot \underbrace{\left(\prod_{k=1}^{K} \mathcal{D}(\boldsymbol{\theta}_k; \boldsymbol{\beta}_k)\right)}_{p(\Theta|\boldsymbol{\beta})}$$

$$= \left(\prod_{d=1}^{D} \frac{\Gamma(\sum_k \alpha_{dk})}{\prod_k \Gamma(\alpha_{dk})} \prod_{k=1}^{K} \pi_{dk}^{\alpha_{dk}-1+n_{dk\cdot}}\right) \cdot \left(\prod_{k=1}^{K} \frac{\Gamma(\sum_v \beta_{kv})}{\prod_v \Gamma(\beta_{kv})} \prod_{v=1}^{V} \theta_{kv}^{\beta_{kv}-1+n_{\cdot kv}}\right)$$

Posteriors

Latent Dirichlet Allocation



$$p(C, \Pi, \Theta, W) = \left(\prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\pi}_d; \boldsymbol{\alpha}_d) \right) \cdot \left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \pi_{dk}^{c_{dik}} \right) \right) \cdot \left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \theta_{kw_{di}}^{c_{dik}} \right) \right) \cdot \left(\prod_{k=1}^{K} \mathcal{D}(\boldsymbol{\theta}_k; \boldsymbol{\beta}_k) \right)$$

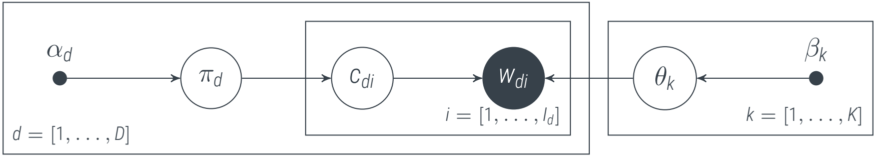
- If we had Π, Θ (which we don't), then the posterior $p(C \mid \Theta, \Pi, W)$ would be easy:

$$p(C \mid \Theta, \Pi, W) = \frac{p(W, C, \Theta, \Pi)}{\sum_C p(W, C, \Theta, \Pi)} = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \frac{\prod_{k=1}^{K} (\pi_{dk} \theta_{kw_{di}})^{c_{dik}}}{\sum_{k'} (\pi_{dk'} \theta_{k'w_{di}})}$$

- note that this conditional independence can easily be read off from the above graph!

Posteriors

Latent Dirichlet Allocation



$$p(C, \Pi, \Theta, W) = \left(\prod_{d=1}^{D} \frac{\Gamma(\sum_k \alpha_{dk})}{\prod_k \Gamma(\alpha_{dk})} \prod_{k=1}^{K} \pi_{dk}^{\alpha_{dk} - 1 + n_{dk\cdot}} \right) \cdot \left(\prod_{k=1}^{K} \frac{\Gamma(\sum_v \beta_{kv})}{\prod_v \Gamma(\beta_{kv})} \prod_{v=1}^{V} \theta_{kv}^{\beta_{kv} - 1 + n_{\cdot kv}} \right)$$

- If we had C (which we don't), then the posterior $p(\Theta, \Pi \mid C, W)$ would be easy:

$$p(\Theta, \Pi \mid C, W) = \frac{p(C, W, \Pi, \Theta)}{\int p(\Theta, \Pi, C, W) \, d\Theta \, d\Pi} = \frac{\left(\prod_d \mathcal{D}(\pi_d; \alpha_d) \left(\prod_k \pi_{dk}^{n_{dk\cdot}}\right)\right) \left(\prod_k \mathcal{D}(\theta_k; \beta_k) \left(\prod_v \theta_{kv}^{n_{\cdot kv}}\right)\right)}{p(C, W)}$$

$$= \left(\prod_d \mathcal{D}(\pi_d; \alpha_{d:} + n_{d:\cdot}) \right) \left(\prod_k \mathcal{D}(\theta_k; \beta_{k:} + n_{\cdot k:}) \right)$$

- note that this conditional independence **can not** easily be read off from the above graph!

The Algorithms

A Gibbs Sampler

Simple Markov Chain Monte Carlo inference

Iterate between (recall $n_{dkv} = \#\{i : w_{di} = v, c_{ijk} = 1\}$)

$$\Theta \sim p(\Theta \mid C, W) = \prod_k \mathcal{D}(\theta_k; \beta_{k:} + n_{.k:})$$

$$\Pi \sim p(\Pi \mid C, W) = \prod_d \mathcal{D}(\pi_d; \alpha_{d:} + n_{d:.})$$

$$C \sim p(C \mid \Theta, \Pi, W) = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \frac{\prod_{k=1}^{K} (\pi_{dk}\theta_{kw_{di}})^{c_{dik}}}{\sum_{k'} (\pi_{dk'}\theta_{k'w_{di}})}$$

- This is *comparably* easy to implement because there are libraries for sampling from Dirichlet's, and discrete sampling is trivial. All we have to keep around are the counts n (which are sparse!) and Θ, Π (which are comparably small). Thanks to factorization, much can also be done in parallel!
- Unfortunately, this sampling scheme is relatively slow to move out of initialization, because z depends strongly on θ, π and vice versa.
- properly vectorizing the code is important for speed

- Consider the exponential family $p_w(x \mid w) = \exp\left[\phi(x)^\intercal w - \log Z(w)\right]$
- its conjugate prior is the exponential family $F(\alpha, \nu) = \int \exp(\alpha^\intercal w - \nu^\intercal \log Z(w))\, dw$

$$p_\alpha(w \mid \alpha, \nu) = \exp\left[\begin{pmatrix} w \\ -\log Z(w) \end{pmatrix}^\intercal \begin{pmatrix} \alpha \\ \nu \end{pmatrix} - \log F(\alpha, \nu)\right]$$

$$\text{because } p_\alpha(w \mid \alpha, \nu) \prod_{i=1}^{n} p_w(x_i \mid w) \propto p_\alpha\left(w \,\middle|\, \alpha + \sum_i \phi(x_i), \nu + n\right)$$

- and the predictive is

$$p(x) = \int p_w(x \mid w) p_\alpha(w \mid \alpha, \nu)\, dw = \int e^{(\phi(x)+\alpha)^\intercal w - (\nu+1)\log Z(w) - \log F(\alpha,\nu)}\, dw$$

$$= \frac{F(\phi(x) + \alpha, \nu + 1)}{F(\alpha, \nu)}$$

Exponential Families, among other things (see also last lecture) provide **conjugate priors** for standard distributions (Lectures 2,15)

- Consider the exponential family $p(c \mid \pi) = \exp\left[c^\intercal(\log \pi) - \log \sum_k \pi_k\right]$
- its conjugate prior is the exponential family $B(\alpha) = \int \exp(\alpha^\intercal \log \pi - \nu \cdot 0)\, d\pi$

$$\mathcal{D}(\pi \mid \alpha) = \exp\left[\log \pi^\intercal \alpha - \log B(\alpha)\right]$$

$$\text{because } \mathcal{D}(\pi \mid \alpha) \prod_{i=1}^n \pi^{c_i} \propto \mathcal{D}\left(\pi \mid \alpha + \sum_i c_i\right)$$

- and the predictive is

$$p(c) = \int p(c \mid \pi)\mathcal{D}(\pi \mid \alpha)\, d\pi = \int e^{(c+\alpha)^\intercal(\log \pi) + \log B(\alpha)}\, d\pi = \frac{B(c+\alpha)}{B(\alpha)}$$

Exponential Families, among other things (see also last lecture) provide **conjugate priors** for standard distributions (Lectures 2,15)

Collapsing (marginalizing) latent structure

It pays off to look closely at the math!

T. L. Griffiths & M. Steyvers, *Finding scientific topics*, PNAS **101**/1 (4/2004), 5228–5235

Recall $\Gamma(x+1) = x \cdot \Gamma(x) \; \forall x \in \mathbb{R}_+$

$$p(C,\Pi,\Theta,W) = \left(\prod_{d=1}^{D} \frac{\Gamma(\sum_k \alpha_{dk})}{\prod_k \Gamma(\alpha_{dk})} \prod_{k=1}^{K} \pi_{dk}^{\alpha_{dk}-1+n_{dk\cdot}}\right) \cdot \left(\prod_{k=1}^{K} \frac{\Gamma(\sum_v \beta_{kv})}{\prod_v \Gamma(\beta_{kv})} \prod_{v=1}^{V} \theta_{kv}^{\beta_{kv}-1+n_{\cdot kv}}\right)$$

$$= \left(\prod_{d=1}^{D} \frac{B(\alpha_d + n_{d:\cdot})}{B(\alpha_d)} \mathcal{D}(\pi_d; \alpha_d + n_{d:\cdot})\right) \cdot \left(\prod_{k=1}^{K} \frac{B(\beta_k + n_{\cdot k:})}{B(\beta_k)} \mathcal{D}(\theta_k; \beta_k + n_{\cdot k:})\right)$$

$$p(C,W) = \left(\prod_{d=1}^{D} \frac{B(\alpha_d + n_{d:\cdot})}{B(\alpha_d)}\right) \cdot \left(\prod_{k=1}^{K} \frac{B(\beta_k + n_{\cdot k:})}{B(\beta_k)}\right)$$

$$= \left(\prod_{d} \frac{\Gamma(\sum_{k'} \alpha_{dk'})}{\Gamma(\sum_{k'} \alpha_{dk'} + n_{dk'\cdot})} \prod_k \frac{\Gamma(\alpha_{dk}+n_{dk\cdot})}{\Gamma(\alpha_{dk})}\right) \left(\prod_{k} \frac{\Gamma(\sum_v \beta_{kv})}{\Gamma(\sum_v \beta_{kv} + n_{\cdot kv})} \prod_v \frac{\Gamma(\beta_{kv}+n_{\cdot kv})}{\Gamma(\beta_{kv})}\right)$$

$$p(c_{dik} = 1 \mid C^{\setminus di}, W) = \frac{(\alpha_{dk} + n_{dk\cdot}^{\setminus di})(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di})(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di})^{-1}}{\sum_{k'}(\alpha_{dk'} + n_{dk'\cdot}^{\setminus di}) \cdot \sum_{w'}(\beta_{kw'} + n_{\cdot kw'}^{\setminus di}) \cdot \sum_{v'}(\beta_{kv'} + n_{\cdot kv'}^{\setminus di})^{-1}}$$

A Collapsed Gibbs Sampler for LDA

It pays off to look closely at the math!

T. L. Griffiths & M. Steyvers, *Finding scientific topics*, PNAS **101**/1 (4/2004), 5228–5235

$$p(C,W) = \left(\prod_d \frac{\Gamma(\sum_k \alpha_{dk})}{\Gamma(\sum_k \alpha_{dk} + n_{dk\cdot})} \prod_k \frac{\Gamma(\alpha_{dk}+n_{dk\cdot})}{\Gamma(\alpha_{dk})}\right) \left(\prod_k \frac{\Gamma(\sum_v \beta_{kv})}{\Gamma(\sum_v \beta_{kv} + n_{\cdot kv})} \prod_v \frac{\Gamma(\beta_{kv}+n_{\cdot kv})}{\Gamma(\beta_{kv})}\right)$$

A **collapsed** sampling method can converge much faster by eliminating the latent variables that mediate between individual data.

procedure LDA(W, α, β)
- $\gamma_{dkv} \leftarrow 0 \;\forall d,k,v$ // initialize counts
- **while** true **do**
  - **for** $d = 1,\dots,D; i = 1,\dots,I_d$ **do** // can be parallelized
    - $c_{di} \propto (\alpha_{dk} + n_{dk\cdot}^{\setminus di})(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di})(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di})^{-1}$ // sample assignment
    - $n \leftarrow$ UPDATECOUNTS(c_{di}) // update counts (check whether first pass or repeat)
  - **end for**
- **end while**

end procedure

Collapsed Sampling is quite efficient

The Mean Field argument [figure: T. L. Griffiths & M. Steyvers, *Finding scientific topics*, PNAS **101**/1 (4/2004), 5228–5235]

Thomas Griffiths

image: Princeton U

Mark Steyvers

image: UC Irvine

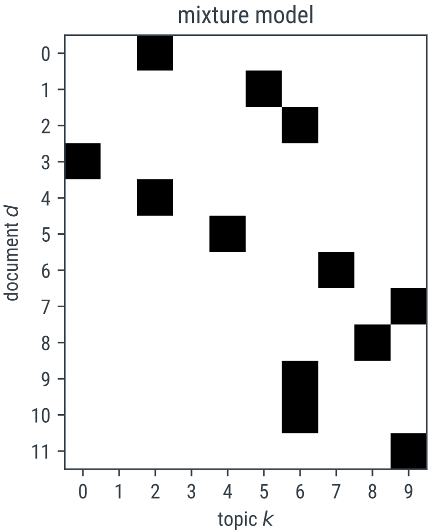
The collapsed sampler operates on the **mean field**

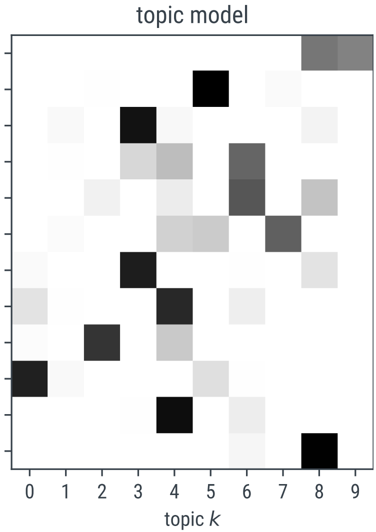
$$p(C \mid W) = \int p(C \mid \Theta, \Pi, W) p(\Theta, \Pi \mid W) \, d\Theta \, d\Pi$$

The *expected* value of the variables Θ, Π that mediate between the "particles" (words). This works well because each word's topic is approximately independent of all individual other words' topics (but together they create the whole thing).

Mixture Models

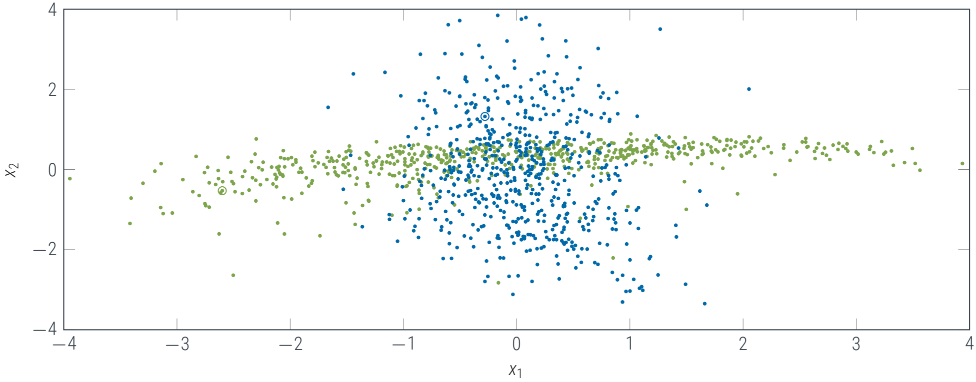
what if each document consists of only one topic?





EBERHARD KARLS UNIVERSITÄT TÜBINGEN

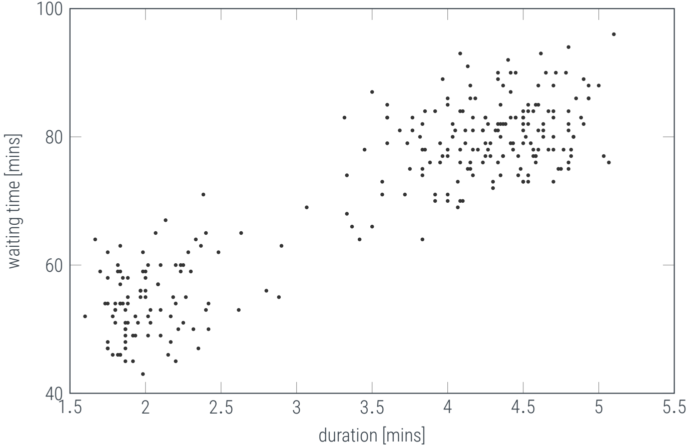
a **supervised** problem



an **unsupervised** problem

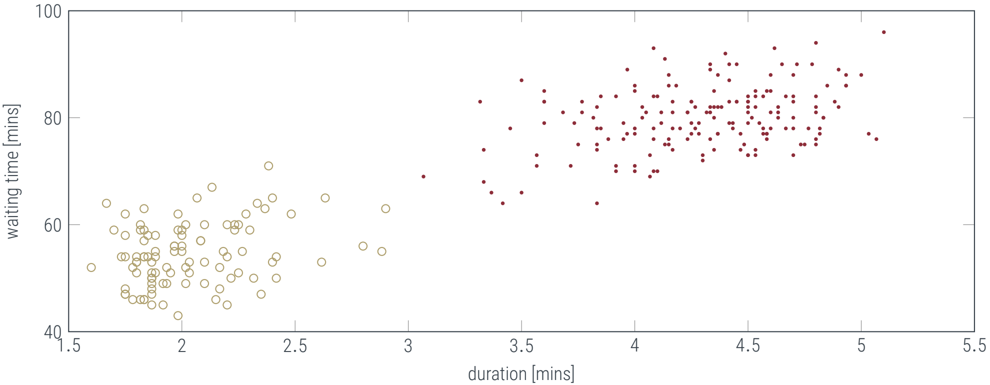
https://www.stat.cmu.edu/~larry/all-of-statistics/=data/faithful.dat

Azzalini, A. and Bowman, A. W. (1990). *A look at some data on the Old Faithful geyser.* Applied Statistics 39, 357-365.



EBERHARD KARLS UNIVERSITÄT TÜBINGEN

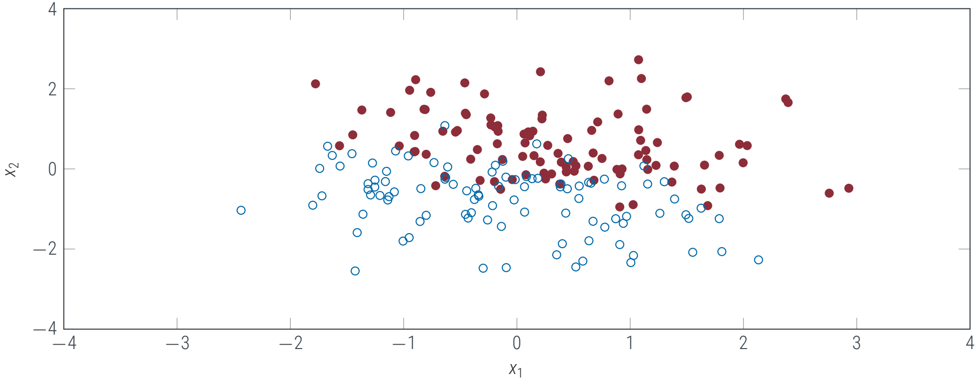
a **clustering**



A Typography of Machine Learning Problems

Unsupervised, Supervised, Generative, Discriminative

a **supervised** problem that can be solved **discriminatively** in a *linear* fashion

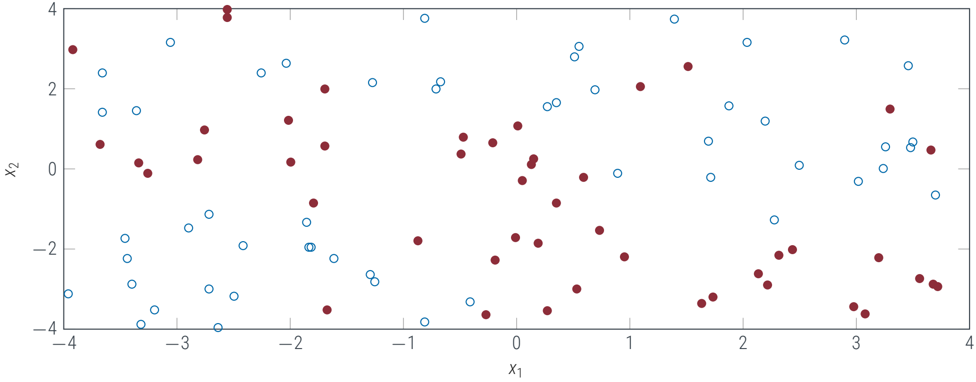


A Typography of Machine Learning Problems

Unsupervised, Supervised, Generative, Discriminative

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

a **supervised** problem that can be solved **discriminatively** in a *nonlinear* fashion



A Typography of Machine Learning Problems

nb: this list is not complete!

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Task types

Supervised given **input-output pairs** $[x_i \in \mathbb{X}, y_i \in \mathbb{Y}]_{i=1,\dots,n} = (X_{\text{train}}, Y_{\text{train}})$, predict $y_{\text{test}}(x_{\text{test}})$

- Regression $\mathbb{Y} = \mathbb{R}^d$
- Classification $\mathbb{Y} \subset \mathbb{N} = \sigma(\mathbb{R}^d)$
- Structured Output $\mathbb{Y} \simeq f(\mathbb{R}^d)$
- Time Series $\mathbb{X} = \mathbb{R}$

Unsupervised given collection $[x_i \in \mathbb{X}]_{i=1,\dots,n}$

- Generative Modelling assume $x_i \sim p$. Make more $x_j \sim p$
- **Clustering** assign a class $c_i \in [1, \dots, C]$ for each x_i (why?)

Note: there are many more task types and sub-types (semi-supervised, dimensionality reduction, matrix factorization, causal inference, …)

We will see that **Clustering** is a subtype of (or even the same thing as?) Generative Modelling.
Clustering is also primarily a way to reduce dimensionality/complexity;
it should be used carefully if the goal is to “discover” structure.

k-Means Clustering

Steinhaus, H. (1957). *Sur la division des corps matériels en parties*. Bull. Acad. Polon. Sci. 4 (12): 801–804.

Given $\{x_i\}_{i=1,\dots,n}$

Init Set k means $\{m_k\}$ to random values

Assign each datum x_i to its *nearest mean*. One could denote this by an integer variable

$$k_i = \arg\min_k \|m_k - x_i\|^2$$

or by binary responsibilities

$$r_{ki} = \begin{cases} 1 & \text{if } k_i = k \\ 0 & \text{else} \end{cases}$$

Update set the means to the sample mean of each cluster

$$m_k \leftarrow \frac{1}{R_k} \sum_i^n r_{ki} x_i \qquad \text{where } R_k := \sum_i r_{ki}$$

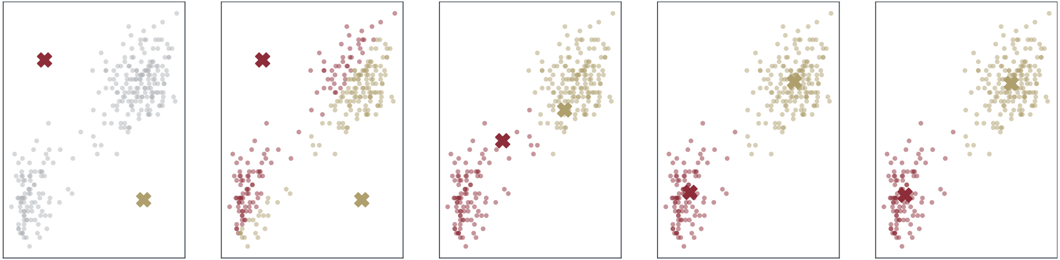
Repeat until the assignments do not change

Hugo Steinhaus
1887–1972

Pseudocode

k-means

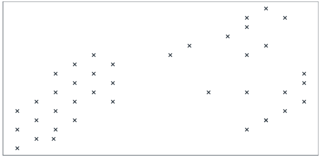
```
procedure k-MEANS(x, k)
    m ← RAND(k)                          // initialize
    while not converged do
        r ← FIND(min(||m − x||^2))       // set responsibilities
        m ← rx ⊘ r1                      // set means
    end while
    return m
end procedure
```



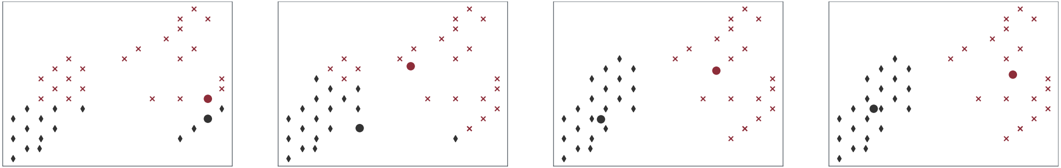
k-means has pathologies

figures after DJC MacKay, ITILA, 2003, recreated by Ann-Kathrin Schalkamp

data from David JC MacKay's book:



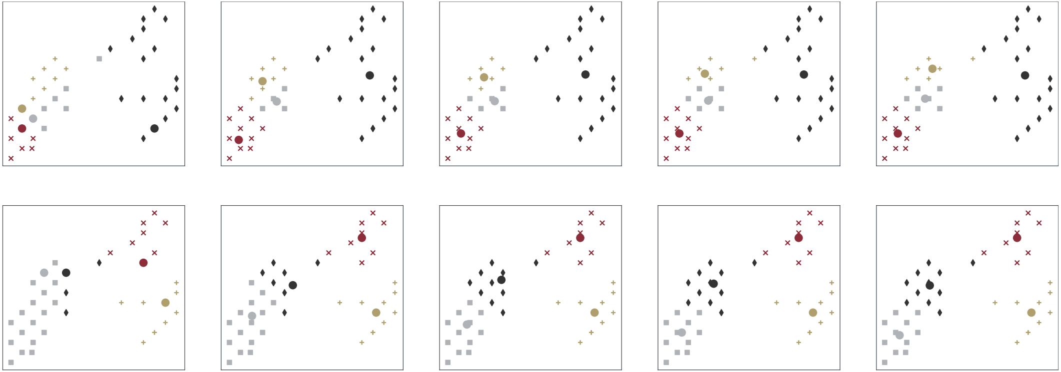
k-means can work well …



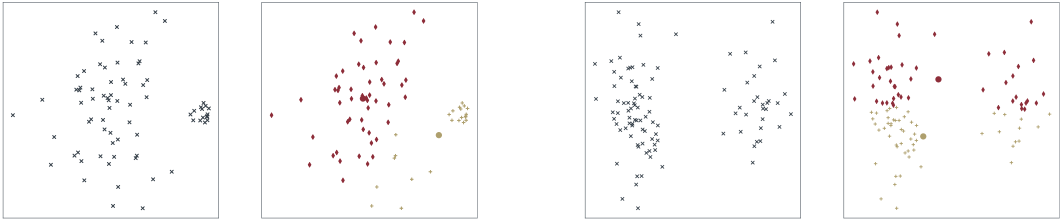
k-means has pathologies

figures after DJC MacKay, ITILA, 2003, recreated by Ann-Kathrin Schalkamp

…but it has no way to set *k* …



…or to set the *shape* of the clusters!



k-means always converges

for an interesting reason …

Definition (Lyapunov Function)

In the context of iterative algorithms, a *Lyapunov Function* J is a positive function of the algorithm's state variables that decreases in each step of the algorithm.

The existence of a Lyapunov function means that one can think about the algorithm in question as an optimization routine for J. It also guarantees convergence of the algorithm at a *local* (not necessarily global!) minimum of J

Aleksandr M. Lyapunov
(1857—1918)

k-means always converges …

for an interesting reason …

1 **procedure** k-MEANS($\boldsymbol{x}, k$)

2 $\quad \boldsymbol{m} \leftarrow \text{RAND}(k)$ // initialize

3 $\quad$ **while** not converged **do**

4 $\quad\quad \boldsymbol{r} \leftarrow \text{FIND}(\min(\|\boldsymbol{m} - x\|^2))$ // set responsibilities

5 $\quad\quad \boldsymbol{m} \leftarrow \boldsymbol{rx} \oslash \boldsymbol{r1}$ // set means

6 $\quad$ **end while**

7 $\quad$ **return** $\boldsymbol{m}$

8 **end procedure**

Consider $\quad J(r, m) := \sum_i^n \sum_k^K r_{ik} \|x_i - m_k\|^2$

- step 4 always decreases J (by definition)
- step 5 always decreases J, because

$$\frac{\partial}{\partial m_k} J(r, m) = -2 \sum_i^n r_{ik}(x_i - m_k) = 0 \quad \Rightarrow \quad m_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}} \qquad \frac{\partial^2 J(r, m)}{\partial m_k^2} = 2 \sum_i r_{ik} > 0$$

- *k*-means is a simple algorithm that always finds **a** stable clustering
- the resulting clusterings can be unintuitive. They do not capture shape of clusters or their number, and are subject to random fluctuations

a probabilistic interpretation of *k*-means yields clarity and allows fitting all parameters. As a neat side-effect, it leads to a final entry to our toolbox!