<table>
<thead>
<tr>
<th>#</th>
<th>content</th>
<th>Ex</th>
<th>#</th>
<th>content</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
<td>15</td>
<td>Exponential Families</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Reasoning under Uncertainty</td>
<td>16</td>
<td>16</td>
<td>Graphical Models</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Continuous Variables</td>
<td>2</td>
<td>17</td>
<td>Factor Graphs</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Monte Carlo</td>
<td>18</td>
<td>18</td>
<td>The Sum-Product Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Markov Chain Monte Carlo</td>
<td>3</td>
<td>19</td>
<td>Example: Modelling Topics</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Gaussian Distributions</td>
<td>4</td>
<td>20</td>
<td>Latent Dirichlet Allocation</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>Parametric Regression</td>
<td>21</td>
<td>21</td>
<td>Mixture Models</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>Learning Representations</td>
<td>22</td>
<td>22</td>
<td>Bayesian Mixture Models</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>Gaussian Processes</td>
<td>5</td>
<td>23</td>
<td>Expectation Maximization (EM)</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>Understanding Kernels</td>
<td>24</td>
<td>24</td>
<td>Variational Inference</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>Gauss-Markov Models</td>
<td>25</td>
<td>25</td>
<td>Example: Collapsed Inference</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>An Example for GP Regression</td>
<td>26</td>
<td>26</td>
<td>Example: EM for Kernel Topic Models</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>GP Classification</td>
<td>27</td>
<td>27</td>
<td>Making Decisions</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>Generalized Linear Models</td>
<td>28</td>
<td>28</td>
<td>Revision</td>
<td>13</td>
</tr>
</tbody>
</table>
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:

\[ \int p(x_1, x_2) \, dx_2 = p(x_1) \quad p(x_1, x_2) = p(x_1 \mid x_2)p(x_2) \quad p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} \]

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

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

- Draw $K$ topic distributions $\theta_k$ over $V$ words from
- Draw $D$ document distributions over $K$ topics from
- Draw topic assignments $c_{dik}$ of word $w_{di}$ from
- Draw word $w_{di}$ from

Useful notation: $n_{dkv} = \# \{i : w_{di} = v, c_{dik} = 1\}$. Write $n_{dk} := [n_{d1k}, \ldots, n_{dVk}]$ and $n_{dkv} = \sum_v n_{dkv}$, etc.
The Joint Topic Models

\[ p(C, \Pi, \Theta, W) = \left( \prod_{d=1}^{D} D(\pi_d; \alpha_d) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \pi_{ dik}^{c_{ dik}} \right) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \theta_{ dik}^{c_{ dik}} \right) \right) \cdot \left( \prod_{k=1}^{K} D(\theta_k; \beta_k) \right) \]

\[ = \left( \prod_{d=1}^{D} D(\pi_d; \alpha_d) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \pi_{ dik}^{c_{ dik}} \right) \right) \cdot \left( \prod_{k=1}^{K} D(\theta_k; \beta_k) \right) \]

\[ \text{Probabilistic ML — P. Hennig, SS 2021 — Lecture 21: Efficient Inference & Mixture Models — © Philipp Hennig, 2021 CC BY-NC-SA 3.0} \]
Posterior distributions for Latent Dirichlet Allocation (LDA) are given by:

\[
p(C, \Pi, \Theta, W) = \left( \prod_{d=1}^{D} \mathcal{D}(\pi_d; \alpha_d) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \pi_{dik} \right) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \theta_{dkw_{di}} \right) \right) \cdot \left( \prod_{k=1}^{K} \mathcal{D}(\theta_k; \beta_k) \right)
\]

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

\[
p(C \mid \Theta, \Pi, W) = \frac{p(W, C, \Theta, \Pi)}{\sum_{C} p(W, C, \Theta, \Pi)} = \prod_{d=1}^{D} \prod_{i=1}^{l_d} \frac{\Pi_{k=1}^{K} (\pi_{dkw_{di}})^{c_{dik}}}{\sum_{k'}^{K} (\pi_{dk'w_{di}})}
\]

Note that this conditional independence can easily be read off from the above graph!
If we had \( C \) (which we don’t), then the posterior \( p(\Theta, \Pi | C, W) \) would be easy:

\[
p(\Theta, \Pi | C, W) = \frac{p(C, W, \Pi, \Theta)}{\int p(\Theta, \Pi, C, W) \, d\Theta \, d\Pi} = \frac{(\prod_{d} D(\pi_d; \alpha_d)) (\prod_{k} \pi_{dk}^{n_{dk}})) (\prod_{k} D(\theta_k; \beta_k)) (\prod_{v} \theta_{kv}^{n_{kv}}))}{p(C, W)}
\]

\[
= \left( \prod_{d} D(\pi_d; \alpha_d; + n_{d:.}) \right) \left( \prod_{k} D(\theta_k; \beta_k; + n_{.k}) \right)
\]

► 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 $n_{dkv} = \# \{i : w_{di} = v, c_{ijk} = 1 \}$)

$$\Theta \sim p(\Theta | C, W)$$

$$\Pi \sim p(\Pi | C, W)$$

$$C \sim p(C | \Theta, \Pi, W)$$

$$= \prod_k D(\theta_k; \beta_k + n_{.k:})$$

$$= \prod_d D(\pi_d; \alpha_d + n_{d:})$$

$$= \prod_{d=1}^D \prod_{i=1}^{I_d} \prod_{k=1}^K \frac{\prod_{k'=1}^K (\pi_{dk} \theta_{k'w_{di}})^{c_{ dik}}}{\sum_{k''} (\pi_{dk''} \theta_{k''w_{di}})}$$

▶ 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 $\Theta, \Pi$ (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 $\theta, \pi$ and vice versa.

▶ properly vectorizing the code is important for speed
Consider the exponential family 
\[ p_w(x \mid w) = \exp \left[ \phi(x)^T w - \log Z(w) \right] \]
its conjugate prior is the exponential family 
\[ F(\alpha, \nu) = \int \exp(\alpha^T w - \nu^T \log Z(w)) \, dw \]

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

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

and the predictive is 
\[ p(x) = \int p_w(x \mid w)p_\alpha(w \mid \alpha, \nu) \, dw = \int \exp(\phi(x)(\alpha) + (\nu + 1) \log Z(w) - \log F(\alpha, \nu)) \, dw \]
\[ = \frac{F(\phi(x) + \alpha, \nu + 1)}{F(\alpha, \nu)} \]

**Exponential Families**, among other things (see also last lecture) provide **conjugate priors** for standard distributions (Lectures 2,15)
Consider the exponential family \( p(c \mid \pi) = \exp \left[ c^T (\log \pi) - \log \sum_k \pi_k \right] \)

its conjugate prior is the exponential family

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

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

and the predictive is

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

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

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

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

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

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

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

$$p(c_{dik} = 1 | C^{\backslash \text{di}}, W) = \frac{(\alpha_{dk} + n_{dk})^{\beta_{kw}^{\text{di}} + n_{kw}^{\text{di}} - 1} \Gamma\left(\sum_{v} \beta_{kv} + n_{kv}\right)^{-1}}{\sum_{k'}^{\text{di}} (\alpha_{dk'} + n_{dk'}) \cdot \sum_{v}^{\text{di}} (\beta_{kw'} + n_{kw'}) \cdot \sum_{v}^{\text{di}} (\beta_{kv'} + n_{kv'})^{-1}}$$
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

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

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

1. procedure LDA(W, \(\alpha\), \(\beta\))
2. \(\gamma_{dkv} \leftarrow 0 \ \forall d, k, v\) \hfill // initialize counts
3. while true do
4.     for \(d = 1, \ldots, D; i = 1, \ldots, I_d\) do
5.         \[c_{di} \propto (\alpha_{dk} + n_{dk,di})(\beta_{kw_{di}} + n_{kw_{di}})(\sum_v \beta_{kv} + n_{kv})^{-1}\] \hfill // can be parallelized
6.         \(n \leftarrow \text{UPDATECOUNTS}(c_{di})\) \hfill // sample assignment
7.     end for
8. end while
9. end procedure
Collapsed Sampling is quite efficient

The Mean Field argument

The collapsed sampler operates on the mean field

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

The expected value of the variables $\Theta, \Pi$ 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?
Mixture Models

a **supervised** problem
Mixture Models

an unsupervised problem

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

Mixture Models

A clustering

![Graph showing a clustering with waiting time vs. duration in minutes. The graph has a scatter plot with two distinct clusters.]
a **supervised** problem that can be solved **discriminatively** in a **linear** fashion
a supervised problem that can be solved discriminatively in a nonlinear fashion
A Typography of Machine Learning Problems

nb: this list is not complete!

Task types

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

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

Unsupervised given collection \([x_i \in X]_{i=1,\ldots,n}\)

- Generative Modelling assume \(x_i \sim p\). Make more \(x_j \sim p\)
- Clustering assign a class \(c_i \in [1, \ldots, C]\) 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.
Given \( \{x_i\}_{i=1, \ldots, n} \)

**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

\[
m_k \leftarrow \frac{1}{R_k} \sum_{i} r_{ki} x_i \quad \text{where } R_k := \sum_{i} r_{ki}
\]

**Repeat** until the assignments do not change.
Pseudocode

$k$-means

1 \textbf{procedure} $k$-MEANS($x \!, k$)  
2 \hspace{1em} $m \leftarrow \text{RAND}(k)$  
3 \hspace{1em} \textbf{while} not converged \textbf{do}  
4 \hspace{2em} $r \leftarrow \text{FIND}(\min(||m - x||^2))$  
5 \hspace{2em} $m \leftarrow r x \ominus r1$  
6 \hspace{1em} \textbf{end while}$  
7 \hspace{1em} \textbf{return} $m$  
8 \textbf{end procedure}$
k-Means Clustering
Example on Old Faithful

[Figure after in C. Bishop, made by Ann-Kathrin Schalkamp]
"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 $k$ ...

Probabilistic ML — P. Hennig, SS 2021 — Lecture 21: Efficient Inference & Mixture Models — © Philipp Hennig, 2021 CC BY-NC-SA 3.0
*k*-means has pathologies

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

...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 ...

```
procedure k-MEANS(x, k)
    \[ m \leftarrow \text{RAND}(k) \] // initialize
    \[ \text{while not converged do} \]
    \[ r \leftarrow \text{FIND}(\min(||m - x||^2)) \] // set responsibilities
    \[ m \leftarrow rx \otimes r1 \] // set means
    end while
    return m
end procedure
```

Consider \[ J(r, m) := \sum_{i} \sum_{k} r_{ik} ||x_i - m_k||^2 \]

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

\[
\frac{\partial}{\partial m_k} J(r, m) = -2 \sum_{i} r_{ik}(x_i - m_k) = 0 \quad \Rightarrow \quad m_k = \frac{\sum_{i} r_{ik} x_i}{\sum_{i} r_{ik}}
\]

\[
\frac{\partial^2 J(r, m)}{\partial m_k^2} = 2 \sum_{i} r_{ik} > 0
\]
- $k$-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!