<table>
<thead>
<tr>
<th>#</th>
<th>date</th>
<th>content</th>
<th>Ex</th>
<th>#</th>
<th>date</th>
<th>content</th>
<th>Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20.04.</td>
<td>Introduction</td>
<td>1</td>
<td>14</td>
<td>09.06.</td>
<td>Generalized Linear Models</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21.04.</td>
<td>Reasoning under Uncertainty</td>
<td>2</td>
<td>15</td>
<td>15.06.</td>
<td>Exponential Families</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>27.04.</td>
<td>Continuous Variables</td>
<td>3</td>
<td>16</td>
<td>16.06.</td>
<td>Graphical Models</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>28.04.</td>
<td>Monte Carlo</td>
<td>4</td>
<td>17</td>
<td>22.06.</td>
<td>Factor Graphs</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>04.05.</td>
<td>Markov Chain Monte Carlo</td>
<td>5</td>
<td>18</td>
<td>23.06.</td>
<td>The Sum-Product Algorithm</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>05.05.</td>
<td>Gaussian Distributions</td>
<td>6</td>
<td>19</td>
<td>29.06.</td>
<td>Example: Modelling Topics</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>11.05.</td>
<td>Parametric Regression</td>
<td>7</td>
<td>20</td>
<td>30.06.</td>
<td>Mixture Models</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12.05.</td>
<td>Learning Representations</td>
<td>8</td>
<td>21</td>
<td>06.07.</td>
<td>EM</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>18.05.</td>
<td>Gaussian Processes</td>
<td>9</td>
<td>22</td>
<td>07.07.</td>
<td>Variational Inference</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>19.05.</td>
<td>Understanding Kernels</td>
<td>10</td>
<td>23</td>
<td>13.07.</td>
<td>Fast Variational Inference</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>25.05.</td>
<td>An Example for GP Regression</td>
<td>12</td>
<td>25</td>
<td>20.07.</td>
<td>Outlook</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>08.06.</td>
<td>GP Classification</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>09.06.</td>
<td>Generalized Linear Models</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>15.06.</td>
<td>Exponential Families</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>16.06.</td>
<td>Graphical Models</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>22.06.</td>
<td>Factor Graphs</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>23.06.</td>
<td>The Sum-Product Algorithm</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>29.06.</td>
<td>Example: Modelling Topics</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>30.06.</td>
<td>Mixture Models</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>06.07.</td>
<td>EM</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>07.07.</td>
<td>Variational Inference</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>13.07.</td>
<td>Fast Variational Inference</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>20.07.</td>
<td>Outlook</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>21.07.</td>
<td>Revision</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></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
A Mixture of Probabilities?

Desiderata

- Topics should be probabilities: 
  \[ p(x_d | k) = \prod_{v=1}^{V} \prod_{i=1}^{x_{dv}} \theta_{k(i),v} \]

- But documents should have several topics! Let \( \pi_{dk} \) be the probability to draw a word from topic \( k \)

**Doc Sparsity** each document \( d \) should only contain a small number of topics

**Word Sparsity** each topic \( k \) should only contain a small number of the words \( v \) in the vocabulary.
A Discrete Topic Model

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

- Draw topic assignments $c_{di} = [c_{d1i}, \ldots, c_{dki}] \in \{0; 1\}^K$, $\sum_k c_{dik} = 1$ of word $w_{id}$ from $C \in \{0; 1\}^{D \times l_d \times K}$ with $p(C | \Pi) = \prod_{d=1}^{D} \prod_{i=1}^{l_d} \prod_{k=1}^{K} \pi_{dik}^{c_{dik}}$

- Draw word $w_{di}$ from $p(w_{di} = v | c_{di}, \Theta) = \prod_k \theta_{kv}^{c_{dik}}$

But we need priors for $\Pi, \Theta$. And we’d like them to be sparse!
Reminder: The Dirichlet Distribution

a sparsity prior for probability distribution

\[ p(x | \pi) = \prod_{i=1}^{n} \pi_{x_i} \quad x \in \{1; \ldots, K\} \]

\[ = \prod_{k=1}^{K} \pi_k^{n_k} \]

\[ n_k := |\{x_i | x_i = k\}| \]

\[ p(\pi | \alpha) = D(\alpha) = \frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} = \frac{1}{B(\alpha)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} \]

\[ p(\pi | x) = D(\alpha + n) \]
Dirichlets can encode sparsity for $\alpha \sim 0.01, 0.1, 0.5$ (100 samples each)
To draw $I_d$ 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 $c_{ik}$ of word $w_{di}$ from
- Draw word $w_{di}$ from

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

To draw \( I_d \) words \( w_{di} \) 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 \( c_{ik} \) of word \( w_{di} \) from
- Draw word \( w_{di} \) from

Useful notation: \( n_{dkv} = \#\{i: w_{di} = v, c_{ dik} = 1\} \). Write \( n_{dk}: = \left[n_{dk1}, \ldots, n_{dkV}\right] \) and \( n_{dk} = \sum_v n_{dkv} \), etc.
(Directed) Graphical Models provide a visual language to represent ideas and structure. They also allow immediate mathematical insight into conditional independence.

$$p(C, \Pi, \Theta, W) = p(\Pi \mid \alpha) \cdot p(C \mid \Pi) \cdot p(\Theta \mid \beta) \cdot p(W \mid C, \Theta)$$

$$= \left( \prod_{d=1}^{D} p(\pi_d \mid \alpha_d) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} p(c_{di} \mid \pi_d) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} p(w_{di} \mid c_{di}, \Theta) \right) \cdot \left( \prod_{k=1}^{K} p(\theta_k \mid \beta_k) \right)$$

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

Exponential Family Distributions are a collection of “standard” distributions for certain data types. Here: for discrete probability distributions \((\theta, \pi)\), the Dirichlet distributions. Exponential families simplify maximum likelihood, MAP and full Bayesian inference by mapping integration to differentiation and Bayesian inference (through conjugate priors) to addition of sufficient statistics.
\[ p(C, \Pi, \Theta, W) = \left( \prod_{d=1}^{D} \mathcal{D}(\pi_d; \alpha_d) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \pi_c^{\alpha_{dik}} \right) \right) \cdot \left( \prod_{d=1}^{D} \prod_{i=1}^{l_d} \left( \prod_{k=1}^{K} \theta_c^{\alpha_{dik}} \right) \right) \cdot \left( \prod_{k=1}^{K} \mathcal{D}(\theta_k; \beta_k) \right) \]

Exponential Family Distributions are a collection of "standard" distributions for certain data types. Here: for discrete probability distributions \((\theta, \pi)\), the Dirichlet distributions. Exponential families simplify maximum likelihood, MAP and full Bayesian inference by mapping integration to differentiation and Bayesian inference (through conjugate priors) to addition of sufficient statistics.

- Here, the Dirichlet exponential family (the conjugate prior for the multinominal distributions \(\pi, \theta\)) simplifies the analysis by collapsing \(C\) out of the joint.
- The Dirichlet is not a "perfectly" expressive language on probabilities (more later). But it allows encoding sparsity, which is crucial for this application.
- Nevertheless, as we already know from the graph, inference on \(\theta, \pi\) is tricky, because they are dependent (the posteriors both depend on \(n!\))
Until now we have only built a model. Time to build an algorithm.
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
Let’s look at the structure again

Latent Dirichlet Allocation

\[
\begin{align*}
\alpha_d &\quad \pi_d \\
\theta_k &\quad \beta_k
\end{align*}
\]

\[
d = [1, \ldots , D]
\]

\[
i = [1, \ldots , I_d]
\]

\[
k = [1, \ldots , K]
\]

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

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

\[
p(C | \Theta, \Pi, W) = \frac{p(W, C, \Theta, \Pi)}{\sum_C p(W, C, \Theta, \Pi)} = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \prod_{k=1}^{K} \frac{\left(\pi_{dk} \theta_{kw_d}\right)^{c_{dk}}}{\sum_{k'} \left(\pi_{dk'} \theta_{k'w_d}\right)^{c_{dk'}}}
\]

▶ note that this conditional independence can easily be read off from the above graph!
Let’s look at the structure again

Latent Dirichlet Allocation

\[ \alpha_d \rightarrow \pi_d \rightarrow C_{di} \rightarrow W_{di} \rightarrow \theta_k \rightarrow \beta_k \]

\[ d = [1, \ldots, D] \]

\[ i = [1, \ldots, I_d] \]

\[ k = [1, \ldots, K] \]

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

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

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

▶ note that this conditional independence can easily be read off from the above graph!
Let’s look at the structure again

Latent Dirichlet Allocation

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

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

\[
\frac{p(C, W, \Pi, \Theta)}{\int p(\Theta, \Pi, C, W) \, d\Theta \, d\Pi} = \left( \prod_{d} \mathcal{D}(\pi_d; \alpha_d) \right) \left( \prod_{k} \mathcal{D}(\theta_k; \beta_k) \right)
\]

► note that this conditional independence can not easily be read off from the above graph!
Conditional independence structure can inform the choice of algorithm. In particular: If the entire model separate into independent, tractable parts when conditioned on one latent variable, and vice versa, *Gibbs sampling* is a contender.
$x_t \leftarrow x_{t-1}; x_{ti} \sim p(x_{ti} \mid x_{t1}, x_{t2}, \ldots, x_{t(i-1)}, x_{t(i+1)}, \ldots)$

a special case of Metropolis-Hastings:

- $q(x' \mid x_t) = \delta(x'_i - x_{t_i})p(x'_i \mid x_{t\setminus i})$
- $p(x') = p(x'_i \mid x'_{\setminus i})p(x'_{\setminus i}) = p(x'_i \mid x_{t\setminus i})p(x_{t\setminus i})$

acceptance rate: $a = \frac{p(x')}{{p(x_t)}} \cdot \frac{q(x_t \mid x')}{{q(x' \mid x_t)}} = \frac{p(x'_i \mid x_{t\setminus i})}{p(x_{t_i} \mid x_{t\setminus i})} \cdot \frac{p(x_{t_i} \mid x_{t\setminus i})}{p(x'_i \mid x_{t\setminus i})} = 1$

Markov Chain Monte Carlo Methods provide a relatively simple way to construct approximate posterior distributions. They are asymptotically exact. Compared to other approximate inference methods, like variational inference, they tend to be easier to implement but harder to monitor, and may also be more computationally expensive.

$$\int f(x)p(x) \, dx \approx \frac{1}{S} \sum_{s=1}^{S} f(x_s) \quad \text{if} \quad x_s \sim p$$
A Gibbs Sampler

Simple Markov Chain Monte Carlo inference

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

\[
\Theta \sim p(\Theta | C, W) = \prod_k D(\theta_k; \beta_k: + n_{k:})
\]

\[
\Pi \sim p(\Pi | C, W) = \prod_d D(\pi_d; \alpha_d: + n_{d:})
\]

\[
C \sim p(C | \Theta, \Pi, W) = \prod_{d=1}^{D} \prod_{i=1}^{l_d} \prod_{k=1}^{K} \frac{\prod_{k=1}^{K} (\pi_d \theta_k w_{di})^{c_{dik}}}{\sum_{k'} (\pi_d \theta_{k'} w_{di})}
\]

- 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
Designing a Probabilistic Machine Learning Model

1. get the data
   1.1 try to collect as much meta-data as possible
   1.2 take a close look at the data

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

4. Test the Setup

5. Revisit the Model and try to improve it, using creativity