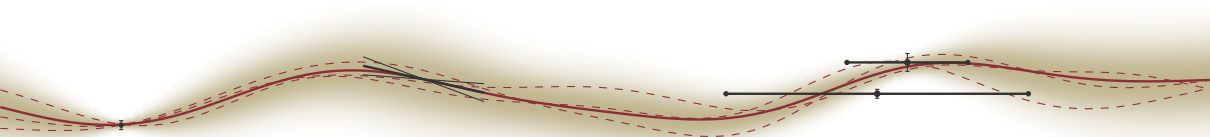
Probabilistic Machine Learning
Lecture 27
Revision

Philipp Hennig

27 July 2021

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning



#	date	content	Ex	#	date	content	Ex
1	20.04.	Introduction	1	14	09.06.	Generalized Linear Models	
2	21.04.	Reasoning under Uncertainty		15	15.06.	Exponential Families	8
3	27.04.	Continuous Variables	2	16	16.06.	Graphical Models	
4	28.04.	Monte Carlo		17	22.06.	Factor Graphs	9
5	04.05.	Markov Chain Monte Carlo	3	18	23.06.	The Sum-Product Algorithm	
6	05.05.	Gaussian Distributions		19	29.06.	Example: Modelling Topics	10
7	11.05.	Parametric Regression	4	20	30.06.	Mixture Models	
8	12.05.	Learning Representations		21	06.07.	EM	11
9	18.05.	Gaussian Processes	5	22	07.07.	Variational Inference	
10	19.05.	Understanding Kernels		23	13.07.	Tuning Inference Algorithms	12
11	26.05.	Gauss-Markov Models		24	14.07.	Kernel Topic Models	
12	25.05.	An Example for GP Regression	6	25	20.07.	Outlook	
13	08.06.	GP Classification	7	26	21.07.	Revision	

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:

Computation:

The Rules of Probability:

- the **Sum Rule:**

$$P(A) = P(A, B) + P(A, \neg B)$$

- the **Product Rule:**

$$P(A, B) = P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A)$$

- **Bayes' Theorem:**

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B, A) + P(B, \neg A)}$$

Bayes' Theorem

Inverting Probabilities

$$\underbrace{P(X \mid D)}_{\text{posterior for } X \text{ given } D} = \frac{\overbrace{P(X)}^{\text{prior for } X} \cdot \overbrace{P(D \mid X)}^{\text{likelihood for } X}}{\underbrace{P(D)}_{\text{evidence for the model}}} = \frac{P(X) \cdot P(D \mid X)}{\sum_{x \in \mathcal{X}} P(D \mid x) P(x)}$$

Bayes' Theorem tells us how to update the *belief* in a *hypothesis X* when observing *data D*.

- $P(D \mid X)$ is *the likelihood of X*, but the *(conditional) probability for D (given X)*
- the **model** is the entire thing – prior *and* likelihood
- despite the name, the prior is not necessarily what you know *before* seeing the data, but the marginal distribution $P(X) = \sum_{d \in \mathcal{D}} P(X, d)$ under *all* possible data.

$A =$ "it will begin to rain by 6pm"
$B =$ "the sky will become cloudy before 6pm"

$$A \Rightarrow B$$

if A is true, the B is true

Assume: if A is true, then B is true ($A \Rightarrow B$)

A is true	thus	B is true	(modus ponens)
B is false	thus	A is false	(modus tollens)
B is true	thus	A becomes more plausible	
A is false	thus	B becomes less plausible	

if A is true, B becomes more plausible ($P(B \mid A) > P(B)$)

A is true	thus	B becomes more plausible
B is false	thus	A becomes less plausible
B is true	thus	A becomes more plausible
A is false	thus	B becomes less plausible

Computational Difficulties of Probability Theory

Uncertainty is a global notion

- The joint distribution of $n = 26$ propositional variables $A, B, \ldots, Z$ has 2^n free parameters

$$
\begin{aligned}
[1] \quad & P(A, B, \ldots, Z) = \ldots \\
[2] \quad & P(\neg A, B, \ldots, Z) = \ldots \\
[3] \quad & P(A, \neg B, \ldots, Z) = \ldots \\
\vdots \quad & \vdots \\
[67\,108\,863] \quad & P(\neg A, \neg B, \ldots, Z) = \ldots \\
[67\,108\,864] \quad & P(\neg A, \neg B, \ldots, \neg Z) = 1 - \sum P(\ldots)
\end{aligned}
$$

- requires not just large memory, but computing marginals like $P(A)$ is also very expensive
- nb: just committing to a single guess is **much** (exponentially in n) cheaper
- can we specify the joint distribution with fewer numbers?

Conditional Independence

Chiefly a computational concept

[Example: Stefan Harmeling]

Definition (conditional independence)

Two variables A and B are ***conditionally independent*** *given variable C, if and only if their conditional distribution factorizes,*

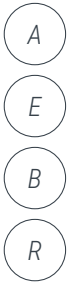
$$P(A, B|C) = P(A|C)\, P(B|C)$$

In that case we have $P(A|B, C) = P(A|C)$, i.e. in light of information C, B provides no (further) information about A. Notation: $A \perp\!\!\!\perp B \mid C$

Parameter Counting

a simple example

[adapted from Pearl, 1988 / MacKay, 2003 §21]



$A =$ the alarm was triggered

$E =$ there was an earthquake

$B =$ there was a break-in

$R =$ an announcement is made on the radio

Joint probability distribution has $2^4 - 1 = 15 = 8 + 4 + 2 + 1$ parameters

$$P(A, E, B, R) = P(A \mid R, E, B) \cdot P(R \mid E, B) \cdot P(E \mid B) \cdot P(B).$$

Removing irrelevant conditions (domain knowledge!) reduces to $8 = 4 + 2 + 1 + 1$ parameters:

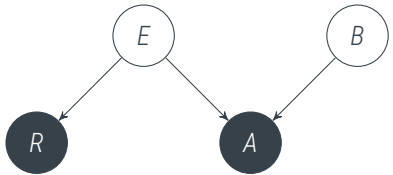
$$P(A, E, B, R) = P(A \mid E, B) \cdot P(R \mid E) \cdot P(E) \cdot P(B)$$

A Graphical Representation

Our first Bayesian network.

[adapted from Pearl, 1988 / MacKay, 2003 §21]

$$P(A, E, B, R) = P(A \mid E, B) \cdot P(R \mid E) \cdot P(E) \cdot P(B)$$



$A =$ the alarm was triggered
$E =$ there was an earthquake
$B =$ there was a break-in
$R =$ an announcement is made on the radio

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

Computation:

Constructing Directed Graphs

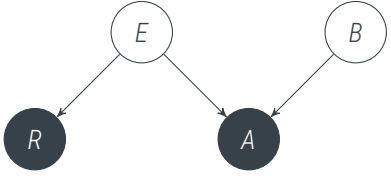
Conditional Independence Affects Computational Complexity

Joint probability distribution has
$2^4 - 1 = 15 = 8 + 4 + 2 + 1$ parameters

$$p(A, E, B, R) = p(A \mid R, E, B) \cdot p(R \mid E, B) \cdot p(E \mid B) \cdot p(B)$$

Removing irrelevant conditions (domain knowledge!) reduces to $8 = 4 + 2 + 1 + 1$ parameters:

$$p(A, E, B, R) = p(A \mid E, B) \cdot p(R \mid E) \cdot p(E) \cdot p(B)$$



Procedural construction of **directed graphical model**

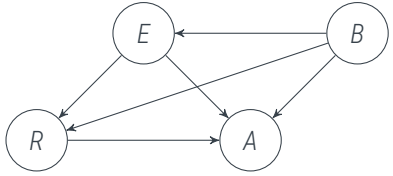
1. For each variable in the joint distribution, draw a circle
2. For each term $p(x_1, \dots \mid y_1, \dots)$ in the factorized joint distribution, draw an arrow *from* every **parent** (right side) node y_i *to* every **child** (left side) node x_i.
3. fill in all **observed** variables (variables on which we want to *condition*).

Every Probability Distribution is a DAG

It's just not always a helpful concept

By the Product Rule, every joint can be factorized into a (dense) DAG.

$$p(A, E, B, R) = p(A \mid E, B, R) \cdot p(R \mid E, B) \cdot p(E \mid B) \cdot p(B)$$



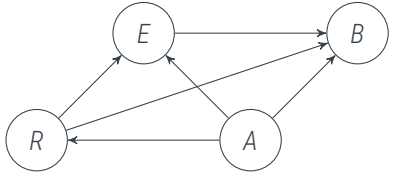
$A =$ the alarm was triggered
$E =$ there was an earthquake
$B =$ there was a break-in
$R =$ an announcement is made on the radio

Every Probability Distribution is a DAG

It's just not always a helpful concept

The direction of the arrows is **not** a causal statement.

$$p(A, E, B, R) = p(B \mid A, E, R) \cdot p(E \mid A, R) \cdot p(R \mid A) \cdot p(A)$$



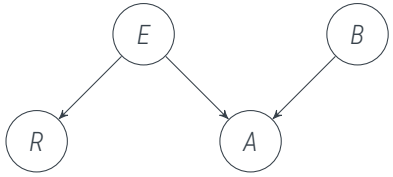
$A =$ the alarm was triggered
$E =$ there was an earthquake
$B =$ there was a break-in
$R =$ an announcement is made on the radio

Every Probability Distribution is a DAG

It's just not always a helpful concept

But the representation is particularly interesting when it reveals **independence**.

$$p(A, E, B, R) = p(A \mid E, B) \cdot p(R \mid E) \cdot p(E) \cdot p(B)$$



$A =$ the alarm was triggered
$E =$ there was an earthquake
$B =$ there was a break-in
$R =$ an announcement is made on the radio

Directed Graphs are an Imperfect Representation

The Graph for Two Coins and a Bell

example by Stefan Harmeling

$$P(A=1)=0.5 \qquad P(C=1 \mid A=1,B=1)=1 \qquad P(C=1 \mid A=1,B=0)=0$$
$$P(B=1)=0.5 \qquad P(C=1 \mid A=0,B=1)=0 \qquad P(C=1 \mid A=0,B=0)=1$$

These CPTs imply $P(A|B)=P(A)$, $P(B|C)=P(B)$ and $P(C|A)=P(C)$ and $P(C \mid B)=P(C)$.

Directed Graphs are an Imperfect Representation

The Graph for Two Coins and a Bell

example by Stefan Harmeling

$$P(A=1)=0.5 \qquad P(C=1 \mid A=1,B=1)=1 \qquad P(C=1 \mid A=1,B=0)=0$$
$$P(B=1)=0.5 \qquad P(C=1 \mid A=0,B=1)=0 \qquad P(C=1 \mid A=0,B=0)=1$$

These CPTs imply $P(A|B)=P(A)$, $P(B|C)=P(B)$ and $P(C|A)=P(C)$ and $P(C \mid B)=P(C)$.

We thus have three factorizations:

1. $P(A,B,C)=P(C|A,B)\cdot P(A|B)\cdot P(B)=P(C|A,B)\cdot P(A)\cdot P(B)$
2. $P(A,B,C)=P(A|B,C)\cdot P(B|C)\cdot P(C)=P(A|B,C)\cdot P(B)\cdot P(C)$
3. $P(A,B,C)=P(B|C,A)\cdot P(C|A)\cdot P(A)=P(B|C,A)\cdot P(C)\cdot P(A)$

Directed Graphs are an Imperfect Representation

The Graph for Two Coins and a Bell

example by Stefan Harmeling

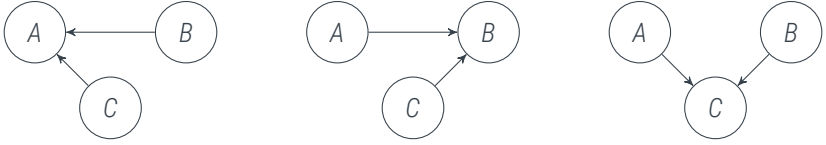
$$P(A=1)=0.5 \qquad P(C=1 \mid A=1,B=1)=1 \qquad P(C=1 \mid A=1,B=0)=0$$
$$P(B=1)=0.5 \qquad P(C=1 \mid A=0,B=1)=0 \qquad P(C=1 \mid A=0,B=0)=1$$

These CPTs imply $P(A|B)=P(A)$, $P(B|C)=P(B)$ and $P(C|A)=P(C)$ and $P(C \mid B)=P(C)$.

We thus have three factorizations:

1. $P(A,B,C)=P(C|A,B)\cdot P(A|B)\cdot P(B)=P(C|A,B)\cdot P(A)\cdot P(B)$
2. $P(A,B,C)=P(A|B,C)\cdot P(B|C)\cdot P(C)=P(A|B,C)\cdot P(B)\cdot P(C)$
3. $P(A,B,C)=P(B|C,A)\cdot P(C|A)\cdot P(A)=P(B|C,A)\cdot P(C)\cdot P(A)$

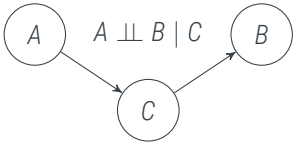
Each corresponds to a graph. Note that each can only express some of the independencies:

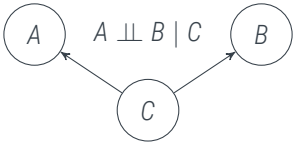


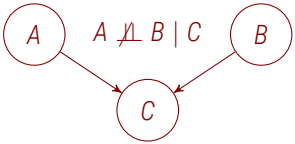
d-separation

A Generalization of the Atomic Structures above

[J. Pearl, *Probabilistic Reasoning in Intelligent Systems* 1988]







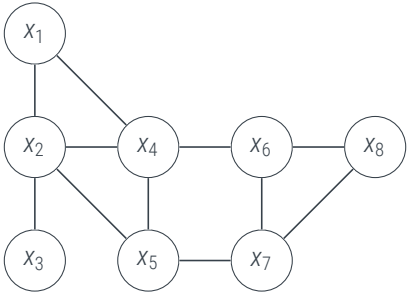
Theorem (d-separation, Pearl, 1988. Formulation taken from Bishop, 2006)

Consider a general directed acyclic graph, in which A, B, C are nonintersecting sets of nodes whose union may be smaller than the complete graph. To ascertain whether $A \perp\!\!\!\perp B \mid C$, consider all possible paths (connections along lines in the graph, regardless of the direction) from any node in A to any node in B. Any such path is considered blocked if it includes a node such that either

- *the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in C, or*
- *the arrows meet head-to-head at the node, and neither the node, nor any of its descendants is in C.*

*If all paths are blocked, then A is said to be d-**separated** from B by C, and $A \perp\!\!\!\perp B \mid C$.*

[example from Bishop, PRML, 2006]



Definition (Markov Random Field)

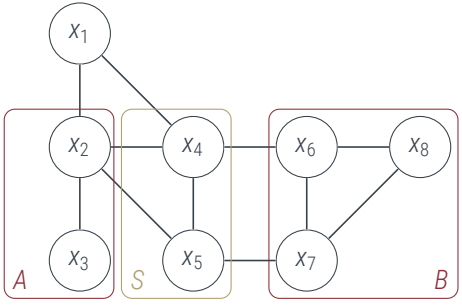
An *undirected Graph* $G = (V, E)$ is a set V of nodes and edges E. An undirected graph G and a set of random variables $X = \{X_v\}_{v \in V}$ is a **Markov Random Field** if, for any subsets $A, B \subset V$ and a *separating set* S (i.e. a set such that every path from A to B passes through S), $X_A \perp\!\!\!\perp X_B \mid X_S$.

The above definition is known as the *global Markov property*. It implies the weaker *pairwise Markov property*: Any two nodes u, v that do not share an edge are conditionally independent given all other variables: $X_u \perp\!\!\!\perp X_v \mid X_{V \setminus \{u,v\}}$.

Undirected Graphical Models

aka. Markov Random Fields

[example from Bishop, PRML, 2006]



Definition (Markov Random Field)

An *undirected Graph* $G = (V, E)$ is a set V of nodes and edges E. An undirected graph G and a set of random variables $X = \{X_v\}_{v \in V}$ is a **Markov Random Field** if, for any subsets $A, B \subset V$ and a *separating set* S (i.e. a set such that every path from A to B passes through S), $X_A \perp\!\!\!\perp X_B \mid X_S$.

The above definition is known as the *global Markov property*. It implies the weaker *pairwise Markov property*: Any two nodes u, v that do not share an edge are conditionally independent given all other variables: $X_u \perp\!\!\!\perp X_v \mid X_{V \setminus \{u,v\}}$.

Potentials

the price of dropping direction from edges

[derivation adapted from Bishop, PRML, 2016]

Any distribution $p(\boldsymbol{x})$ that satisfies the conditional independence structures of the graph G can be written as a factorization over all cliques, and thus also just over all *maximal* cliques (since any clique is part of at least one maximal clique).

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{c \in C} \psi_c(\boldsymbol{x}_c) \qquad (\star)$$

- in directed graphs, each factor $p(x_{\text{ch}} \mid x_{\text{pa}})$ had to be a probability distribution of the children (but not of the parents!). But in MRFs there is no distinction between parents and children. So we only know that each **potential function** $\psi_c(x_c) \geq 0$. For simplicity, we will restrict $\psi_c(x_c) > 0$.
- The normalization constant Z is the **partition function**

$$Z := \sum_{\boldsymbol{x}} \prod_{c \in C} \psi_c(\boldsymbol{x}_c).$$

Because of the loss of structure from directed to undirected graphs, we have to explicitly compute Z. This can be NP-hard, and is the primary downside of MRFs. (e.g. consider n discrete variables with k states each, then computing Z may require summing k^n terms).

Definition (Borel algebra)

Let (Ω, τ) be a topological space. The **Borel σ-algebra** is the σ-algebra *generated* by τ. That is by taking τ and completing it to include infinite intersections of elements from τ and all complements in Ω to elements of τ.

Definition (Probability Density Functions (pdf's))

Let $\mathfrak{B}$ be the Borel σ-algebra in $\mathbb{R}^d$. A probability measure P on $(\mathbb{R}^d, \mathfrak{B})$ has a **density** p if p is a non-negative (Borel) measurable function on $\mathbb{R}^d$ satisfying, for all $B \in \mathfrak{B}$

$$P(B) = \int_B p(x)\, dx =: \int_B p(x_1, \ldots, x_d)\, dx_1 \ \ldots \ dx_d$$

Densities Satisfy the Laws of Probability Theory

because integrals are linear operators

without proof. Cf. Matthias Hein's lecture

- For probability densities p on $(\mathbb{R}^d, \mathfrak{B})$ we have

$$P(E) \overset{(IV)}{=} 1 = \int_{\mathbb{R}^d} p(x)\, dx.$$

- Let $X = (X_1, X_2) \in \mathbb{R}^2$ be a random variable with density p_X on $\mathbb{R}^2$. Then the **marginal densities** of X_1 and X_2 are given by the **sum rule**

$$p_{X_1}(x_1) = \int_{\mathbb{R}} p_X(x_1, x_2)\, dx_2, \qquad p_{X_2}(x_2) = \int_{\mathbb{R}} p_X(x_1, x_2)\, dx_1$$

- The **conditional density** $p(x_1 \mid x_2)$ (for $p(x_2) > 0$) is given by the **product rule**

$$p(x_1 \mid x_2) = \frac{p(x_1, x_2)}{p(x_2)}$$

- **Bayes' Theorem** holds:

$$p(x_1 \mid x_2) = \frac{p(x_1) \cdot p(x_2 \mid x_1)}{\int p(x_1) \cdot p(x_2 \mid x_1)\, dx_1}.$$

Theorem (Transformation Law, general)

Let $X = (X_1, \dots, X_d)$ have a joint density p_X. Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be continously differentiable and injective, with non-vanishing Jacobian J_g. Then $Y = g(X)$ has density

$$p_Y(y) = \begin{cases} p_X(g^{-1}(y)) \cdot |J_{g^{-1}}(y)| & \text{if } y \text{ is in the range of } g, \\ 0 & \text{otherwise.} \end{cases}$$

The Jacobian J_g is the $d \times d$ matrix with

$$[J_g(x)]_{ij} = \frac{\partial g_i(x)}{\partial x_j}.$$

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
- Markov Chains

Computation:

- Monte Carlo

$$F := \int f(x)p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) =: \hat{F} \qquad \text{if } x_i \sim p$$

$$\mathbb{E}_p(\hat{F}) = F \qquad\qquad \mathrm{var}_p(\hat{F}) = \frac{\mathrm{var}_p(f)}{N}$$

- Random numbers can be used to estimate integrals ⇾ **Monte Carlo** algorithms
- although the concept of randomness is fundamentally unsound, Monte Carlo algorithms *are* competitive in high dimensional problems (primarily because the advantages of the alternatives degrade rapidly with dimensionality)
- direct sampling is not possible in general. Practical MC algorithms only use the unnormalized density $\tilde{p}$ in

$$p(x) = \frac{\tilde{p}(x)}{Z}$$

- but even this is not easy, because independent sampling requires access to global structure

The Metropolis-Hastings* Method

* Authorship controversial. Likely inventors: M. Rosenbluth, A. Rosenbluth & E. Teller, 1953

we want to find representers (**samples**) of $\tilde{p}(x)$

- given current sample x_t
- **draw proposal** $x' \sim q(x' \mid x_t)$ (for example, $q(x' \mid x_t) = \mathcal{N}(x'; x_t, \sigma^2)$)
- **evaluate**

$$a = \frac{\tilde{p}(x')}{\tilde{p}(x_t)} \frac{q(x_t \mid x')}{q(x' \mid x_t)}$$

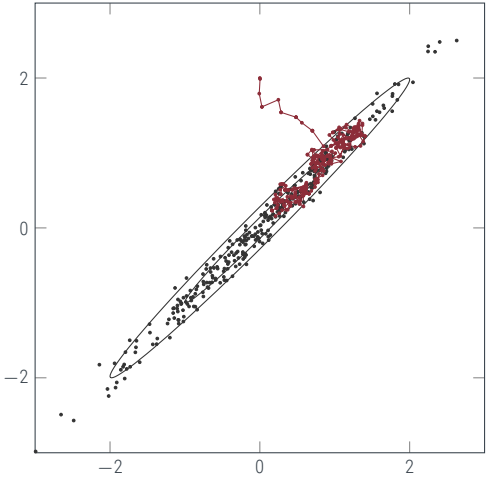
- if $a \geq 1$, **accept**: $x_{t+1} \leftarrow x'$
- else
  - **accept** with probability a: $x_{t+1} \leftarrow x'$
  - **stay** with probability $1 - a$: $x_{t+1} \leftarrow x_t$

Usually, assume symmetry $q(x_t \mid x') = q(x' \mid x_t)$ (the Metropolis method)

- no rejection. Every sample counts!
- like optimization, but with a chance to move downhill

Metropolis-Hastings performs a (biased) random walk

hence diffuses $\mathcal{O}(s^{1/2})$



Rule of Thumb: [MacKay, (29.32)]

- Metropolis-Hastings, in its basic form, performs a random walk, so that the time (number of steps) to draw an independent sample scales like $(L/\varepsilon)^2$, where L is the largest, ε the smallest length-scale of the distribution
- Algorithms that try to correct this behaviour include, for example
  - Gibbs-sampling (drawing exact along the axes)
  - Hamiltonian MC (higher-order dynamics to create smooth exploration

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

Computation:

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

Gaussians provide the linear algebra of inference

if all joints are Gaussian and all observations are linear, all posteriors are Gaussian

- products of Gaussians are Gaussians

$$\begin{aligned}&\mathcal{N}(x;a,A)\mathcal{N}(x;b,B)\\&=\mathcal{N}(x;c,C)\mathcal{N}(a;b,A+B)\\C&:=(A^{-1}+B^{-1})^{-1}\quad c:=C(A^{-1}a+B^{-1}b)\end{aligned}$$

- linear projections of Gaussians are Gaussians

$$\begin{aligned}&p(z)=\mathcal{N}(z;\mu,\Sigma)\\\Rightarrow\quad&p(Az)=\mathcal{N}(Az,A\mu,A\Sigma A^\intercal)\end{aligned}$$

- marginals of Gaussians are Gaussians

$$\int\mathcal{N}\left[\begin{pmatrix}x\\y\end{pmatrix};\begin{pmatrix}\mu_x\\\mu_y\end{pmatrix},\begin{pmatrix}\Sigma_{xx}&\Sigma_{xy}\\\Sigma_{yx}&\Sigma_{yy}\end{pmatrix}\right]dy=\mathcal{N}(x;\mu_x,\Sigma_{xx})$$

- (linear) conditionals of Gaussians are Gaussians

$$\begin{aligned}p(x\mid y)&=\frac{p(x,y)}{p(y)}\\&=\mathcal{N}\left(x;\mu_x+\Sigma_{xy}\Sigma_{yy}^{-1}(y-\mu_y),\Sigma_{xx}-\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)\end{aligned}$$

Bayesian inference becomes linear algebra

$$\text{If } p(x)=\mathcal{N}(x;\mu,\Sigma)\qquad\text{and}\qquad p(y\mid x)=\mathcal{N}(y;A^\intercal x+b,\Lambda),\ \text{then}$$

$$p(B^\intercal x+c\mid y)=\mathcal{N}[B^\intercal x+c;B^\intercal\mu+c+B^\intercal\Sigma A(A^\intercal\Sigma A+\Lambda)^{-1}(y-A^\intercal\mu-b),B^\intercal\Sigma B-B^\intercal\Sigma A(A^\intercal\Sigma A+\Lambda)^{-1}A^\intercal\Sigma B]$$

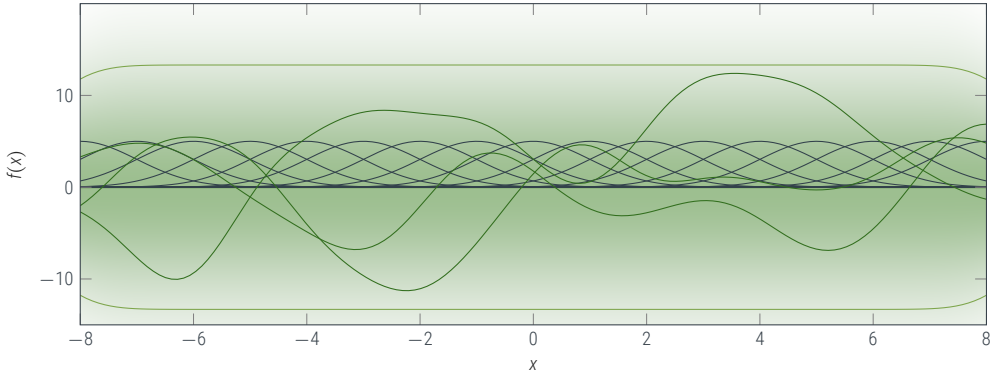
$$f(x) = w_1 + w_2 x = \phi_x^\mathsf{T} w$$

$$\phi_x := \begin{bmatrix} 1 \\ x \end{bmatrix}$$

Learning a Function, with Gaussian algebra

example: Gaussian features

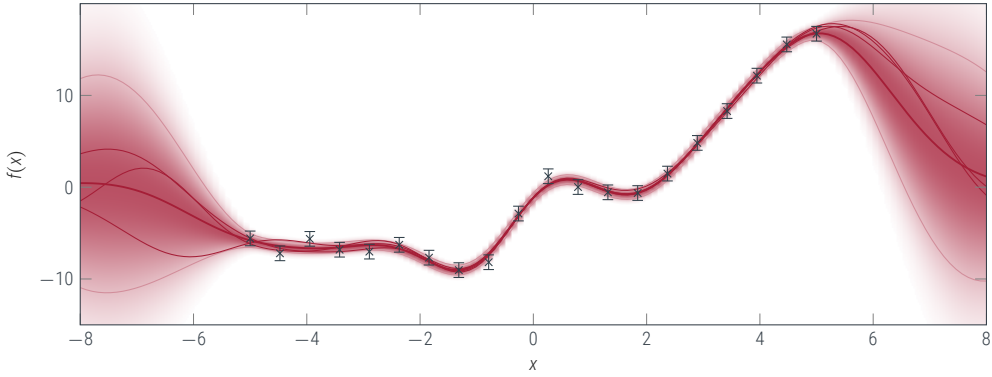
$$\phi(x) = \begin{bmatrix} e^{-\frac{1}{2}(x-8)^2} & e^{-\frac{1}{2}(x-7)^2} & e^{-\frac{1}{2}(x-6)^2} & \dots \end{bmatrix}^\intercal$$



Learning a Function, with Gaussian algebra

example: Gaussian features

$$\phi(x) = \begin{bmatrix} e^{-\frac{1}{2}(x-8)^2} & e^{-\frac{1}{2}(x-7)^2} & e^{-\frac{1}{2}(x-6)^2} & \dots \end{bmatrix}^\intercal$$



It's all just (painful) linear algebra!

Gaussian Inference on a linear function

$$
\begin{aligned}
\text{prior} \quad p(w) &= \mathcal{N}(w; \mu, \Sigma) \quad \Rightarrow \quad p(f) = \mathcal{N}(f_x; \phi_x^\intercal \mu, \phi_x \Sigma \phi_x) \\
\text{likelihood} \quad p(y \mid w, \phi_x) &= \mathcal{N}(y; \phi_X^\intercal w, \sigma^2 I) = \mathcal{N}(y; f_X, \sigma^2 I) \\
\text{posterior on } \boldsymbol{w} \quad p(w \mid \boldsymbol{y}, \phi_X) &= \mathcal{N}(w; \mu + \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(\boldsymbol{y} - \phi_X^\intercal \mu), \\
&\qquad \Sigma - \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}\phi_X^\intercal \Sigma) \\
&= \mathcal{N}\Big(w; (\Sigma^{-1} + \sigma^{-2}\phi_X^\intercal \phi_X)^{-1}\left(\Sigma^{-1}\mu + \sigma^{-2}\phi_X \boldsymbol{y}\right), \\
&\qquad (\Sigma^{-1} + \sigma^{-2}\phi_X^\intercal \phi_X)^{-1}\Big) \\
\text{posterior on } f \quad p(f_x \mid y, \phi_X) &= \mathcal{N}(f_x; \phi_x^\intercal \mu + \phi_x^\intercal \Sigma \phi_X (\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(y - \phi_X^\intercal \mu), \\
&\qquad \phi_x^\intercal \Sigma \phi_x - \phi_x^\intercal \Sigma \phi_X (\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}\phi_X^\intercal \Sigma \phi_x) \\
&\qquad \mathcal{N}\Big(f_x; \phi_x(\Sigma^{-1} + \sigma^{-2}\phi_X^\intercal \phi_X)^{-1}\left(\Sigma^{-1}\mu + \sigma^{-2}\phi_X \boldsymbol{y}\right), \\
&\qquad \phi_x(\Sigma^{-1} + \sigma^{-2}\phi_X^\intercal \phi_X)^{-1}\phi_x^\intercal\Big)
\end{aligned}
$$

$$p(f \mid \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta}) \, df} = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}$$

- **Model parameters** like θ are also known as **hyper-parameters**.
- This is largely a computational, practical distinction:

data	are observed	⇾ **condition**
variables	are the things we care about	⇾ **full probabilistic treatment**
parameters	are the things we have to deal with to get the model right	⇾ **integrate out**
hyper-parameters	are the top-level, too expensive to properly infer	⇾ **fit**

The **model evidence** in Bayes' Theorem is the (marginal) **likelihood** for the model. So we would like

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid \boldsymbol{\theta}') p(\boldsymbol{\theta}') \, d\boldsymbol{\theta}'}$$

$$p(f \mid \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta}) \, df} = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}$$

- For Gaussians, die evidence has **analytic form**:

$$\underbrace{\mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal} w, \Lambda)}_{p(\boldsymbol{y}|f,\boldsymbol{x},\boldsymbol{\theta})} \cdot \underbrace{\mathcal{N}(w, \mu, \Sigma)}_{p(f)} = \underbrace{\mathcal{N}(w; m_{\text{post}}^{\boldsymbol{\theta}}, V_{\text{post}}^{\boldsymbol{\theta}})}_{p(f|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta})} \cdot \underbrace{\mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal} \mu, \phi_X^{\boldsymbol{\theta}\intercal} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda)}_{p(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{x})}$$

- **BUT:** It's not a linear function of $\boldsymbol{\theta}$, so analytic Gaussian inference is not available!

Computational complexity is *the* principal challenge of probabilistic reasoning.

ML / MAP in Practice

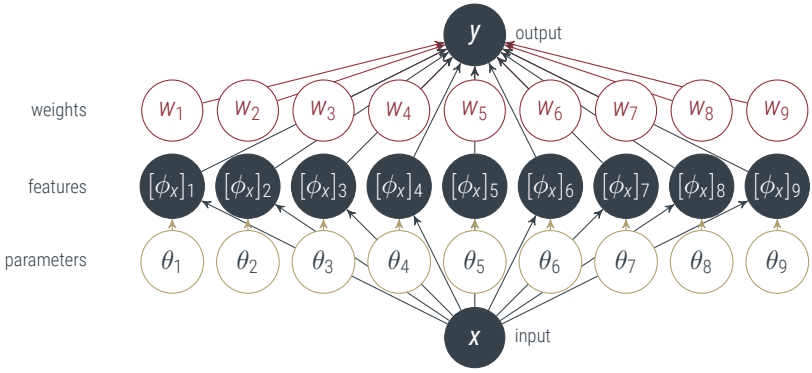
Finding the "best fit" θ in Gaussian models

[e.g. DJC MacKay, *The evidence framework applied to classification networks*, 1992]

$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \int p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})\, df \\
&= \arg\max_{\boldsymbol{\theta}} \mathcal{N}(\boldsymbol{y};\quad \phi_X^{\boldsymbol{\theta}\mathsf{T}}\mu,\quad \phi_X^{\boldsymbol{\theta}\mathsf{T}}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda) \\
&= \arg\max_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{y};\quad \phi_X^{\boldsymbol{\theta}\mathsf{T}}\mu,\quad \phi_X^{\boldsymbol{\theta}\mathsf{T}}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda) \\
&= \arg\min_{\boldsymbol{\theta}} -\log \mathcal{N}(\boldsymbol{y};\quad \phi_X^{\boldsymbol{\theta}\mathsf{T}}\mu,\quad \phi_X^{\boldsymbol{\theta}\mathsf{T}}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \left(\underbrace{(\boldsymbol{y} - \phi_X^{\boldsymbol{\theta}\mathsf{T}}\mu)^\intercal \left(\phi_X^{\boldsymbol{\theta}\mathsf{T}}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda\right)^{-1} (\boldsymbol{y} - \phi_X^{\boldsymbol{\theta}\mathsf{T}}\mu)}_{\text{square error}} + \underbrace{\log \left|\phi_X^{\boldsymbol{\theta}\mathsf{T}}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda\right|}_{\text{model complexity / Occam factor}} \right) + \frac{N}{2}\log 2\pi
\end{aligned}
$$

The Connection to Deep Learning

Representation Learning



A linear Gaussian regressor is a **single hidden layer** neural network, with quadratic output loss, and fixed input layer. Hyperparameter-fitting corresponds to training the input layer. The usual way to train such network, however, does not include the Occam factor.

What are we actually doing with those features?

let's look at that algebra again

$$
\begin{aligned}
p(f_x \mid y, \phi_X) &= \mathcal{N}(f_x; \phi_x^\intercal \mu + \phi_x^\intercal \Sigma \phi_X (\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(y - \phi_X^\intercal \mu), \\
&\qquad \phi_x^\intercal \Sigma \phi_x - \phi_x^\intercal \Sigma \phi_X (\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^\intercal \Sigma \phi_x) \\
&= \mathcal{N}(f_x; m_x + k_{xX}(k_{XX} + \sigma^2 I)^{-1}(y - m_X), \\
&\qquad k_x x - k_{xX}(k_{XX} + \sigma^2 I)^{-1} k_{Xx})
\end{aligned}
$$

using the abstraction / encapsulation

$m_x := \phi_x^\intercal \mu$	$m : \mathbb{X} \to \mathbb{R}$	mean function
$k_{ab} := \phi_a^\intercal \Sigma \phi_b$	$k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$	covariance function, aka. **kernel**

Definition (kernel)

$k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is a (Mercer / positive definite) **kernel** if, for any finite collection $X = [x_1, \dots, x_N]$, the matrix $k_{XX} \in \mathbb{R}^{N \times N}$ with $[k_{XX}]_{ij} = k(x_i, x_j)$ is **positive semidefinite**.

```
def kernel (f) : λ (a,b) -> [[f(a[i],b[j]) for j=1:length(b)] for i=1:length(a) ]
```

actually, in python:

```
def kernel (f) : return lambda a,b : np.array([ [np.float64(f(a[i],b[j])) for j in range(b.size) ] for i in range(a.size) ])
```

Definition

Let $\mu : \mathbb{X} \to \mathbb{R}$ be any function, $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a Mercer kernel. A **Gaussian process** $p(f) = \mathcal{GP}(f; \mu, k)$ is a probability distribution over the function $f : \mathbb{X} \to \mathbb{R}$, such that every finite restriction to function values $f_X := [f_{x_1}, \dots, f_{x_N}]$ is a Gaussian distribution $p(f_X) = \mathcal{N}(f_X; \mu_X, k_{XX})$.

Gaussian Processes

- Sometimes it is possible to consider **infinitely many** features at once, by extending from a sum to an integral. This requires some regularity assumption about the features' locations, shape, etc.
- The resulting **nonparametric model** is known as a **Gaussian process**
- Inference in GPs is tractable (though at polynomial cost $\mathcal{O}(N^3)$ in the number N of datapoints)
- There is no unique kernel. In fact, there are quite a few! E.g.

$$k(a,b) = \exp(-(a-b)^2)$$ Gaussian / Square Exponential / RBF kernel

$$k(a,b) = \min(a - t_0, b - t_0)$$ Wiener process

$$k(a,b) = \frac{1}{3}\min^3(a - t_0, b - t_0) + \frac{1}{2}|a-b| \cdot \min^2(a - t_0, b - t_0)$$ cubic spline kernel

$$k(a,b) = \frac{2}{\pi}\sin^{-1}\left(\frac{2a^\intercal b}{\sqrt{(1+2a^\intercal a)(1+2b^\intercal b)}}\right)$$ Neural Network kernel (Williams, 1998)

Making New Kernels from Old

the space of kernels is large

Theorem:

Let $\mathbb{X}, \mathbb{Y}$ be index sets and $\phi : \mathbb{Y} \rightarrowtriangle \mathbb{X}$. If $k_1, k_2 : \mathbb{X} \times \mathbb{X} \rightarrowtriangle \mathbb{R}$ are Mercer kernels, then the following functions are also Mercer kernels (up to minor regularity assumptions)

- $\alpha \cdot k_1(a, b)$ for $\alpha \in \mathbb{R}_+$ (proof: trivial)
- $k_1(\phi(c), \phi(d))$ for $c, d \in \mathbb{Y}$ (proof: by **Mercer's theorem**, next lecture)
- $k_1(a, b) + k_2(a, b)$ (proof: trivial)
- $k_1(a, b) \cdot k_2(a, b)$ **Schur product theorem** (proof involved. E.g. Bapat, 1997. Million, 2007)

- **Gaussian process regression** is closely related to **kernel ridge regression**.
  - the posterior mean is the kernel ridge / regularized kernel least-squares estimate in the RKHS $\mathcal{H}_k$.

$$m(x) = k_{xX}(k_{XX} + \sigma^2 I)^{-1} \boldsymbol{y} = \operatorname*{arg\,min}_{f \in \mathcal{H}_k} \|\boldsymbol{y} - f_X\|^2 + \|f\|^2_{\mathcal{H}_k}$$

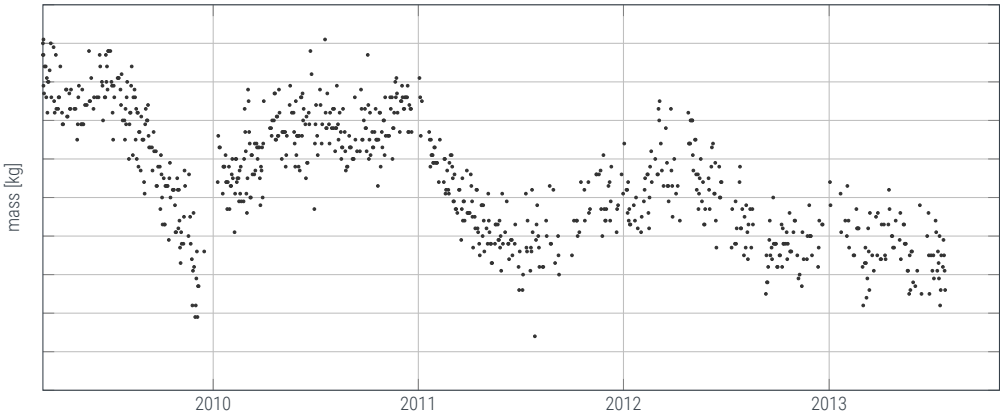
  - the posterior variance (**expected square error**) is the **worst-case square error** for bounded-norm RKHS elements.

$$v(x) = k_{xx} - k_{xX}(k_{XX})^{-1} k_{Xx} = \operatorname*{arg\,max}_{f \in \mathcal{H}_k, \|f\|_{\mathcal{H}_k} \leq 1} \|f(x) - m(x)\|^2$$

- Similar connections apply for most **kernel methods**.
- GPs are quite powerful: They can learn any function in the RKHS (a large, generally infinite-dimensional space!)
- GPs are quite limited: If $f \notin \mathcal{H}_k$, they may converge **very** (e.g. exponentially) slowly to the truth.
- But if we are willing to be cautious enough (e.g. with a rough kernel whose RKHS is a Sobolev space of low order), then polynomial rates are achievable. (Unfortunately, exponentially slow in the dimensionality of the input space)

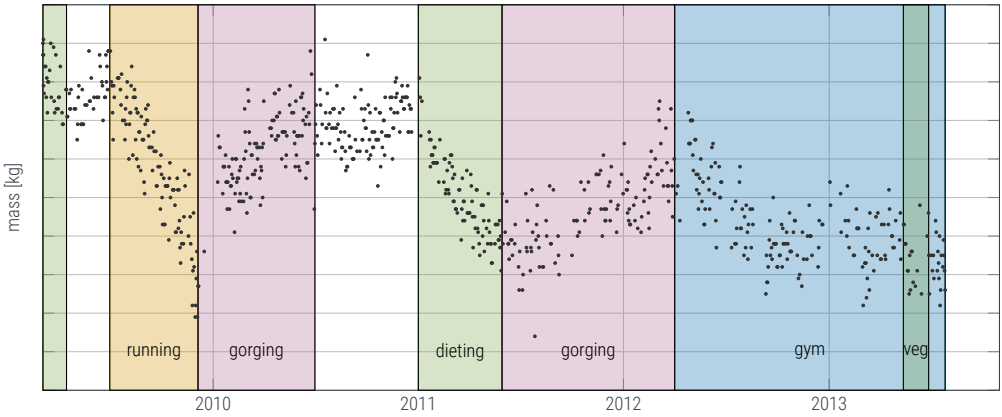
Gaussian Regression is a Powerful Tool for Everyday Use!

(c) P. Hennig, 2007–2013



Gaussian Regression is a Powerful Tool for Everyday Use!

(c) P. Hennig, 2007–2013



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

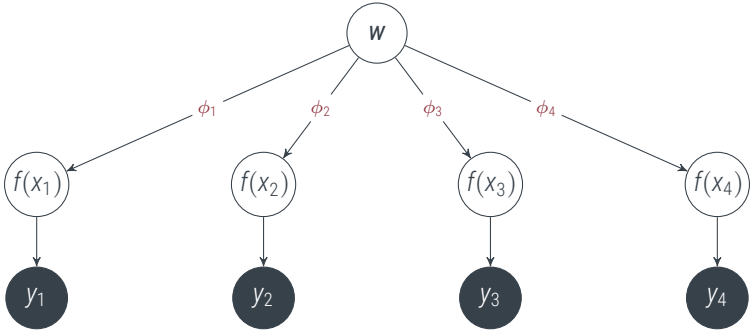
Computation:

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

Graphical View: Parametric Model

Conditional independence of data given model weights

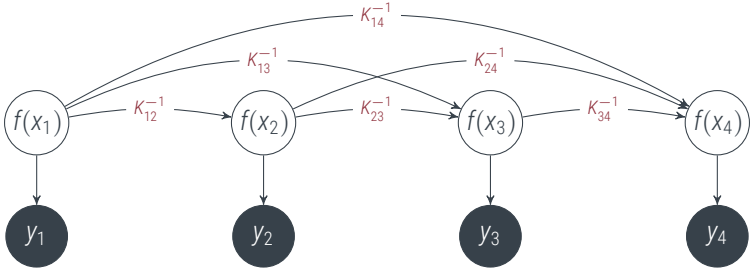
$$p(f) = \mathcal{GP}(f; 0, \Phi_X^\intercal \Sigma \Phi_X) \quad p\left(\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \middle| \, \boldsymbol{w} \right) = \prod_i \delta(f_i - \phi_i^\intercal \boldsymbol{w}) \quad p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_i \mathcal{N}(y_i; f_i, \sigma^2)$$



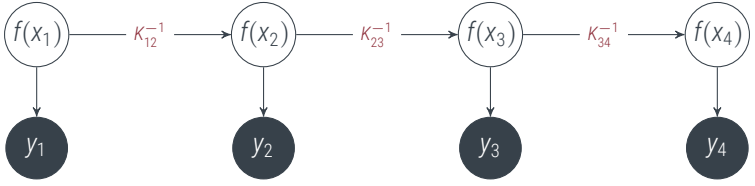
Nonparametric Model

Fully connected graph

$$p(f) = \mathcal{GP}(f; 0, k) \quad p\left(\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}\right) = \mathcal{N}\left(0, \begin{bmatrix} K_{11}^{-1} & K_{12}^{-1} & K_{13}^{-1} & K_{14}^{-1} \\ & K_{22}^{-1} & K_{23}^{-1} & K_{24}^{-1} \\ & & K_{33}^{-1} & K_{34}^{-1} \\ & & & K_{44}^{-1} \end{bmatrix}^{-1}\right) \quad p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_i \mathcal{N}(y_i; f_i, \sigma^2)$$



$$p(f) = \mathcal{GP}(f; 0, k) \quad p\left(\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}\right) = \mathcal{N}\left(0, \begin{bmatrix} K_{11}^{-1} & K_{12}^{-1} & 0 & 0 \\ K_{12}^{-1} & K_{22}^{-1} & K_{23}^{-1} & 0 \\ 0 & K_{23}^{-1} & K_{33}^{-1} & K_{34}^{-1} \\ 0 & 0 & K_{34}^{-1} & K_{44}^{-1} \end{bmatrix}^{-1}\right) \quad p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_i \mathcal{N}(y_i; f_i, \sigma^2)$$



Time Series:

- **Markov Chains** formalize the notion of a stochastic process with a *local finite memory*
- Inference over Markov Chains separates into three operations, that can be performed in *linear* time:

Filtering: $\mathcal{O}(T)$

predict: $$p(x_t \mid Y_{0:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid Y_{0:t-1})\, dx_{t-1}$$ (Chapman-Kolmogorov Eq.)

update: $$p(x_t \mid Y_{0:t}) = \frac{p(y_t \mid x_t) p(x_t \mid Y_{0:t-1})}{p(y_t)}$$

Smoothing: $\mathcal{O}(T)$

smooth: $$p(x_t \mid Y) = p(x_t \mid Y_{0:t}) \int p(x_{t+1} \mid x_t) \frac{p(x_{t+1} \mid Y)}{p(x_{t+1} \mid Y_{0:t})}\, dx_{t+1}$$

Time Series:

- **Markov Chains** formalize the notion of a stochastic process with a *local finite memory*
- Inference over Markov Chains separates into three operations, that can be performed in *linear* time.
- If all relationships are *linear* and *Gaussian*,

$$p(x(t_i) \mid x(t_{i-1})) = \mathcal{N}(x_i; Ax_{i-1}, Q) \qquad p(y_t \mid x_t) = \mathcal{N}(y_t; Hx_t, R)$$

then inference is analytic and given by the **Kalman Filter** and the **Rauch-Tung-Striebel Smoother**:

(Kalman) Filter:

$$\begin{aligned}
p(x_t) &= \mathcal{N}(x_t; m_t^-, P_t^-) && \text{with} \\
m_t^- &= Am_{t-1} && \text{predictive mean} \\
P_t^- &= AP_{t-1}A^\intercal + Q && \text{predictive covariance} \\
p(x_t \mid y_t) &= \mathcal{N}(x_t; m_t, P_t) && \text{with} \\
z_t &= y_t - Hm_t^- && \text{innovation residual} \\
S_t &= HP_t^- H^\intercal + R && \text{innovation covariance} \\
K_t &= P_t^- H^\intercal S^{-1} && \text{Kalman gain} \\
m_t &= m_t^- + Kz_t && \text{estimation mean} \\
P_t &= (I - KH)P_t^- && \text{estimation covariance}
\end{aligned}$$

(Rauch Tung Striebel) Smoother:

$$\begin{aligned}
p(x_t \mid Y) &= \mathcal{N}(x_t; m_t^s, P_t^s) && \text{with} \\
G_t &= P_t A^\intercal (P_{t+1}^-)^{-1} && \text{RTS gain} \\
m_t^s &= m_t + G_t(m_{t+1}^s - m_{t+1}^-) && \text{smoothed mean} \\
P_t^s &= P_t + G_t(P_{t+1}^s - P_{t+1}^-)G^\intercal && \text{smoothed covariance}
\end{aligned}$$

Classification vs. Regression

Two types of supervised learning problems

Regression:

Given supervised *data* (special case $d = 1$: univariate regression)

$$(X, Y) := (x_i, y_i)_{i=1,\ldots,n} \text{ with } x_i \in \mathbb{X}, y_i \in \mathbb{R}^d$$

find function $f : \mathbb{X} \to \mathbb{R}^d$ such that f "models" $Y \approx f(X)$.

Classification:

Given supervised *data* (special case $d = 2$: binary classification)

$$(X, Y) := (x_i, c_i)_{i=1,\ldots,n} \text{ with } x_i \in \mathbb{X}, c_i \in \{1, \ldots, d\}$$

find probability $\pi : \mathbb{X} \to U^d$ ($U^d = \{p \in [0,1]^d : \sum_{i=1}^d p_i = 1\}$) such that π "models" $y_i \sim \pi_{x_i}$.

Regression predicts a **function**, *classification* predicts a **probability**.

$$p(f) = \mathcal{GP}(f; m, k)$$

$$p(y \mid f_x) = \sigma(y f_x) = \begin{cases} \sigma(f) & \text{if } y = 1 \\ 1 - \sigma(f) & \text{if } y = -1 \end{cases} \qquad \text{using } \sigma(x) = 1 - \sigma(-x).$$

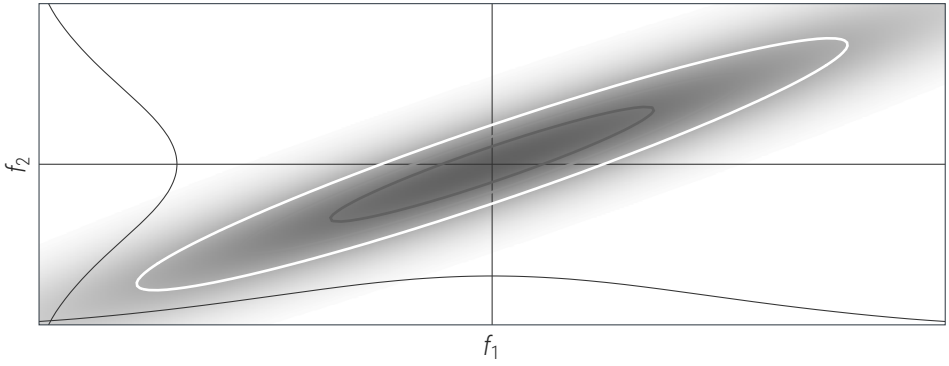
The problem: The posterior is not Gaussian!

$$p(f_X \mid Y) = \frac{p(Y \mid f_X) p(f_X)}{p(Y)} = \frac{\mathcal{N}(f_X; m, k) \prod_{i=1}^n \sigma(y_i f_{x_i})}{\int \mathcal{N}(f_X; m, k) \prod_{i=1}^n \sigma(y_i f_{x_i}) \, df_X}$$

$$\log p(f_X \mid Y) = -\frac{1}{2} f_X^\intercal k_{XX}^{-1} f_X + \sum_{i=1}^n \log \sigma(y_i f_{x_i}) + \text{const.}$$

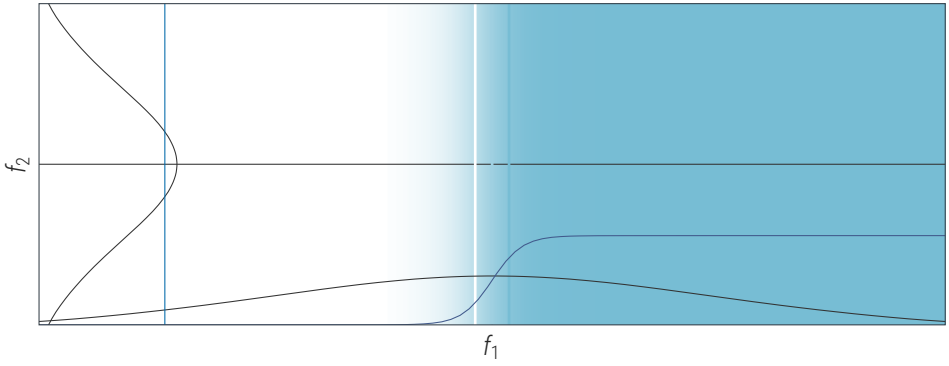
Logistic Regression is non-analytic

We'll have to break out the toolbox



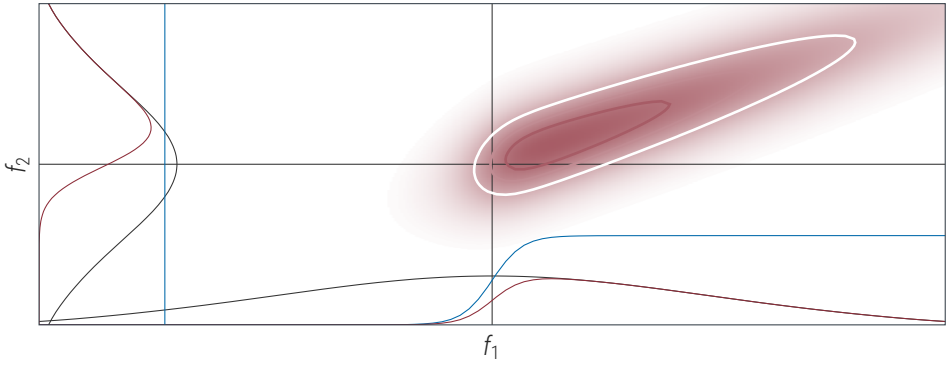
Logistic Regression is non-analytic

We'll have to break out the toolbox



Logistic Regression is non-analytic

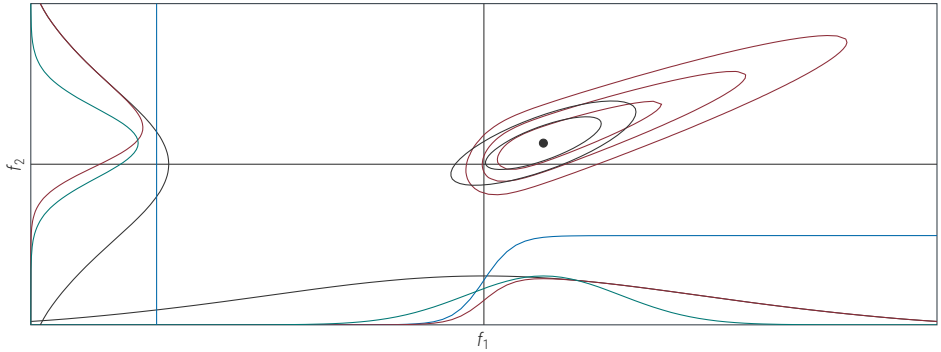
We'll have to break out the toolbox



The Laplace Approximation

A local Gaussian approximation

Pierre Simon M. de Laplace, 1814



The Laplace Approximation

formally

- Consider a probability distribution $p(\theta)$ (may be a posterior $p(\theta \mid D)$ or something else)
- find a (local) **maximum** of $p(\theta)$ or (equivalently) $\log p(\theta)$

$$\hat{\theta} = \arg\max \log p(\theta) \qquad \Rightarrow \qquad \nabla \log p(\hat{\theta}) = 0$$

- perform **second order Taylor expansion** around $\theta = \hat{\theta} + \delta$ in log space

$$\log p(\delta) = \log p(\hat{\theta}) + \frac{1}{2}\delta^\intercal \left(\underbrace{\nabla\nabla^\intercal \log p(\hat{\theta})}_{=:\Psi} \right) \delta + \mathcal{O}(\delta^3)$$

- define **the Laplace approximation** q **to** p

$$q(\theta) = \mathcal{N}(\theta; \hat{\theta}, -\Psi^{-1})$$

- Note that, if $p(\theta) = \mathcal{N}(\theta; m, \Sigma)$, then $p(\theta) = q(\theta)$

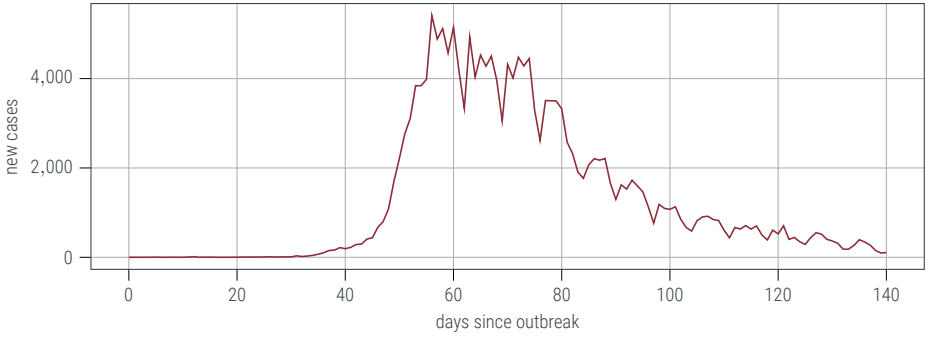
Generalized Linear Models

- extend the idea discussed for *classification* in the previous lecture to general *link functions*. That is, *non-Gaussian likelihoods* of general form.
- a simple (approximate) probabilistic version can be constructed by analogously extending the *Laplace approximation* from the previous lecture
- note that, for arbitrary link functions, the Laplace approximation may well be quite bad

A recent example

count data

data: Robert Koch Institut, 22 May 2020

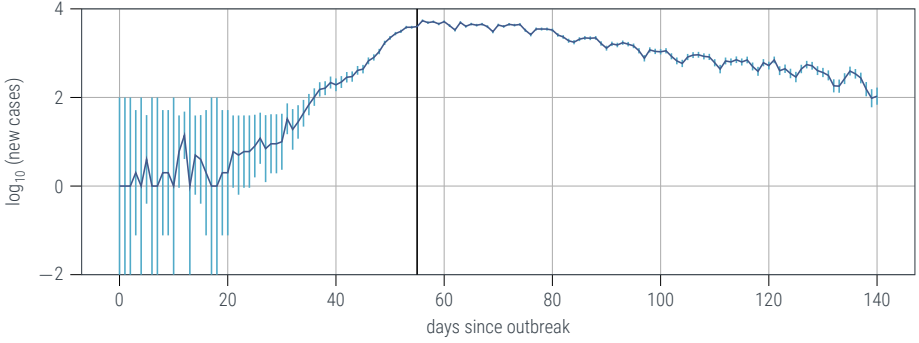


$$p(\boldsymbol{y} \mid f_T) = \mathcal{N}(\boldsymbol{y}; f_T, \sigma^2 I) \qquad p(f) = \mathcal{GP}(f; 0, k)$$

A recent example

count data

data: Robert Koch Institut, 22 May 2020



$$p(\boldsymbol{y} \mid f_T) = \mathcal{N}(\boldsymbol{y}; \exp(f_T), \sigma^2 I) \approx q(\boldsymbol{y} \mid f_T) = \mathcal{N}(\log \boldsymbol{y}; f_T, \sigma^2 \operatorname{diag}(1/\boldsymbol{y})) \quad \text{because}$$

$$\left.\frac{\partial \log p(\boldsymbol{y} \mid f_T)}{\partial f_T}\right|_{f_T=\hat{f}_T} = 0 \quad \Rightarrow \quad \hat{f}_T = \log \boldsymbol{y} \qquad \text{and} \qquad \left.\frac{\partial^2 \log p(\boldsymbol{y} \mid f_T)}{\partial^2 f_T}\right|_{f_t=\hat{f}_T} = \frac{\boldsymbol{y}^2}{\sigma^2}$$

Be Bayesian, even in Deep Learning!

Laplace approximations for deep nets

- A strong point estimate doesn't matter if it's uncertain
- replace $p(y = 1 \mid x) = \sigma(f_W(x))$ with the *marginal*

$$p(y = 1 \mid x) = \int \sigma(f_W(x))\, p(W \mid \boldsymbol{y})\, dW$$

- approximate posterior on W by Laplace as

$$p(W \mid \boldsymbol{y}) \approx \mathcal{N}(W; W^*, -(\nabla\nabla^\intercal J(W))^{-1}) =: \mathcal{N}(W; W^*, \Psi)$$

- and on f by linearizing with $G(x) = \frac{df_{W^*}(x)}{dW}$ as $f_W(x) \approx f_{W^*}(x) + G(x)(W - W^*)$, thus

$$p(f_W(x)) = \int p(f \mid W) p(W)\, dW \approx \mathcal{N}(f(x); f_{W^*}(x), G(x)\Psi G(x)^\intercal) =: \mathcal{N}(f(x); m(x), v(x))$$

- and approximate the marginal (MacKay, 1992) as

$$p(y = 1 \mid x) \approx \sigma\left(\frac{m(x)}{\sqrt{1 + \pi/8\, v(x)}}\right).$$

Problem solved (asymptotically)!

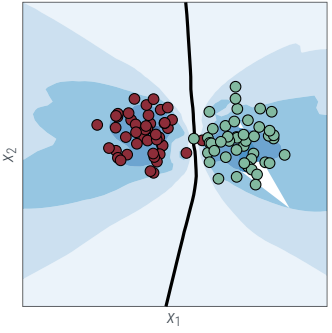
Kristiadi et al. 2020 https://arxiv.org/abs/2002.10118

Theorem (Kristiadi et al., 2020)

Let $f_W : \mathbb{R}^n \to \mathbb{R}$ be a binary ReLU classification network parametrized by $W \in \mathbb{R}^p$ with $p \geq n$, and let $\mathcal{N}(W|W^*, \Psi)$ be the approximate posterior. Then for any input $\boldsymbol{x} \in \mathbb{R}^n$, there exists an $\alpha > 0$ such that for any $\delta \geq \alpha$, the confidence $\sigma(|z(\delta \boldsymbol{x})|)$ is bounded from above by the limit $\lim_{\delta \to \infty} \sigma(|z(\delta \boldsymbol{x})|)$. Furthermore,

$$\lim_{\delta \to \infty} \sigma(|z(\delta \boldsymbol{x})|) \leq \sigma\left(\frac{|\boldsymbol{u}|}{s_{min}\,(\boldsymbol{J})\,\sqrt{\pi/8\,\lambda_{min}(\Psi)}}\right),$$

where $\boldsymbol{u} \in \mathbb{R}^n$ is a vector depending only on W and the $n \times p$ matrix $\boldsymbol{J} := \frac{\partial \boldsymbol{u}}{\partial W}\big|_{W^*}$ is the Jacobian of $\boldsymbol{u}$ w.r.t. W at W^*.



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

Definition (Conjugate Prior)

Let D and x be a data-set and a variable to be inferred, respectively, connected by the likelihood $p(D \mid x) = \ell(D; x)$. A **conjugate prior to ℓ for x** is a probability measure with pdf $p(x) = \pi(x; \theta)$ of functional form π, such that

$$p(x \mid D) = \frac{\ell(D; x)\pi(x; \theta)}{\int \ell(D; x)\pi(x; \theta)\, dx} = \pi(x; \theta').$$

That is, such that the posterior arising from ℓ is of the same functional form as the prior, with updated parameters.

Definition (Exponential Family, simplified form)

Consider a random variable X taking values $x \in \mathbb{X} \subset \mathbb{R}^n$. A probability distribution for X with pdf of the functional form

$$p_w(x) = h(x) \exp\left[\phi(x)^\intercal w - \log Z(w)\right] = \frac{h(x)}{Z(w)} e^{\phi(x)^\intercal w} = p(x \mid w)$$

is called an **exponential family** of probability measures. The function $\phi : \mathbb{X} \to \mathbb{R}^d$ is called the **sufficient statistics**. The parameters $w \in \mathbb{R}^d$ are the **natural parameters** of p_w. The normalization constant $Z(w) : \mathbb{R}^d \to \mathbb{R}$ is the **partition function**. The function $h(x) : \mathbb{X} \to \mathbb{R}_+$ is the **base measure**.

A Family Meeting

incomplete list of exponential families

Name	sufficient stats	domain	use case
Bernoulli	$\phi(x) = [x]$	$\mathbb{X} = \{0; 1\}$	coin toss
Poisson	$\phi(x) = [x]$	$\mathbb{X} = \mathbb{R}_+$	emails per day
Laplace	$\phi(x) = [1, x]^\intercal$	$\mathbb{X} = \mathbb{R}$	floods
Helmert (χ^2)	$\phi(x) = [x, -\log x]$	$\mathbb{X} = \mathbb{R}$	variances
Dirichlet	$\phi(x) = [\log x]$	$\mathbb{X} = \mathbb{R}_+$	class probabilities
Euler (Γ)	$\phi(x) = [x, \log x]$	$\mathbb{X} = \mathbb{R}_+$	variances
Wishart	$\phi(X) = [X, \log \lvert X \rvert]$	$\mathbb{X} = \{X \in \mathbb{R}^{N \times N} \mid v^\intercal X v \geq 0 \forall v \in \mathbb{R}^N\}$	covariances
Gauss	$\phi(X) = [X, XX^\intercal]$	$\mathbb{X} = \mathbb{R}^N$	functions
Boltzmann	$\phi(X) = [X, \text{triag}(XX^\intercal)]$	$\mathbb{X} = \{0; 1\}^N$	thermodynamics

Full Bayesian Regression on Distributions!

Fitting distributions with exponential families

- Given $[x_i]_{i=1,\dots,n}$ with $x_i \sim p(x)$, assume

$$p(x) \approx p_w(x \mid w) = \exp(\phi(x)^\intercal w - \log Z(w)) \quad \text{and} \quad p_F(w \mid \alpha, \nu) = \exp(w^\intercal \alpha - \nu \log Z(w) - \log F(\alpha, \nu))$$

- compute the posterior on w, using the conjugate prior

$$p(w \mid x, \alpha, \nu) = \frac{\prod_{i=1}^n p_w(x_i \mid w) p_F(w \mid \alpha, \nu)}{\int p(x \mid w) p(w \mid \alpha, \nu)\, dx} = p_F\left(w \mid \alpha + \sum_i \phi(x_i), \nu + n\right)$$

- note that $\nabla\nabla p_F(w \mid \alpha, \nu)|_{w_* = \arg\max p(w|\alpha,\nu)} = -\nu p(w_* \mid \alpha, \nu) \nabla_w \nabla_w^\intercal \log Z(w_*)$
- In the limit $n \to \infty$, posterior concentrates at w_* with

$$\nabla_w \log Z(w_*) = \frac{\alpha}{n} + \frac{1}{n} \sum_{i=1}^n \phi(x_i) = \mathbb{E}_p(\phi(x)) \quad \text{thus} \quad p_w(x \mid w_*) = \arg\min_w D_{\mathrm{KL}}(p(x) \| p_w(x \mid w))$$

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:

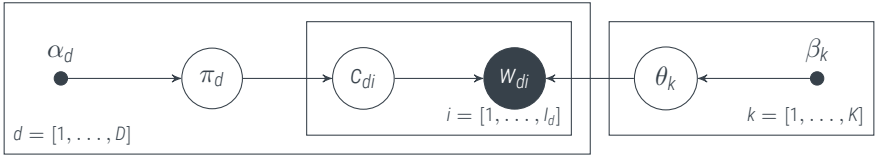
- graphical models
- Gaussian distributions
- (deep) learnt representations
- Kernels
- Markov Chains
- Exponential Families / Conjugate Priors
- Factor Graphs & Message Passing

Computation:

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

Maximum Likelihood?

unfortunately, not always an analytic option



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

$$p(W \mid \Pi, \Theta) = \sum_{d,i,k} \left(\prod_{d=1}^{D}\prod_{i=1}^{I_d}\prod_{k=1}^{K} \pi_{dk}\theta_{kw_{di}}\right) \qquad \log p(W \mid \Pi, \Theta) = \log \sum (\dots) \neq \sum \log (\dots)$$

Maximizing the likelihood for Θ, Π is difficult because it does not factorize along documents or words.

The EM algorithm:

- to find *maximum likelihood* (or MAP) estimate for a model involving a **latent** variable

$$\theta_* = \arg\max_\theta \left[\log p(x \mid \theta)\right] = \arg\max_\theta \left[\log \left(\int p(x, z \mid \theta)\, dz\right)\right]$$

- Initialize θ_0, then iterate between

E Compute $p(z \mid x, \theta_{\text{old}})$, thereby setting $D_{\text{KL}}(q \| p(z \mid x, \theta) = 0$

M Set θ_{new} to the **Maximize** the **Expectation Lower Bound**

$$\theta_{\text{new}} = \arg\max_\theta \mathcal{L}(q, \theta) = \arg\max_\theta \int q(z) \log \left(\frac{p(x, z \mid \theta)}{q(z)}\right) dz$$

- Check for convergence of either the log likelihood, or θ.

The EM algorithm:

- to find *maximum likelihood* (or MAP) estimate for a model involving a **latent** variable

$$\theta_* = \arg\max_\theta \left[\log p(x \mid \theta)\right] = \arg\max_\theta \left[\log \left(\int p(x, z \mid \theta)\, dz\right)\right]$$

- Initialize θ_0, then iterate between

E Compute $p(z \mid x, \theta_{\text{old}})$, thereby setting $D_{\text{KL}}(q \| p(z \mid x, \theta) = 0$

M Set θ_{new} to the **Maximize** the **Expectation Lower Bound / minimize the Variational Free Energy**

$$\theta_{\text{new}} = \arg\max_\theta \mathcal{L}(q, \theta) = \arg\max_\theta \int q(z) \log \left(\frac{p(x, z \mid \theta)}{q(z)}\right) dz$$

- Check for convergence of either the log likelihood, or θ.

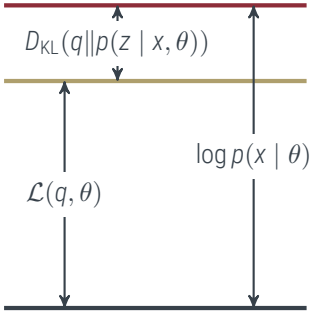
EM maximizes the ELBO / minimizes Free Energy

a more general view

$$\log p(x \mid \theta) = \mathcal{L}(q, \theta) + D_{\mathrm{KL}}(q \| p(z \mid x, \theta))$$

$$\mathcal{L}(q, \theta) = \int q(z) \log \left(\frac{p(x, z \mid \theta)}{q(z)} \right) dz$$

$$D_{\mathrm{KL}}(q \| p(z \mid x, \theta)) = -\int q(z) \log \left(\frac{p(z \mid x, \theta)}{q(z)} \right) dz$$



EM maximizes the ELBO / minimizes Free Energy

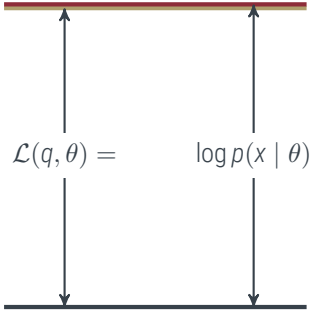
a more general view

$$\log p(x \mid \theta) = \mathcal{L}(q, \theta) + D_{\mathrm{KL}}(q \| p(z \mid x, \theta))$$

$$\mathcal{L}(q, \theta) = \int q(z) \log \left(\frac{p(x, z \mid \theta)}{q(z)} \right) dz$$

$$D_{\mathrm{KL}}(q \| p(z \mid x, \theta)) = - \int q(z) \log \left(\frac{p(z \mid x, \theta)}{q(z)} \right) dz$$

E -step: $q(z) = p(z \mid x, \theta_{\mathrm{old}})$, thus $D_{\mathrm{KL}}(q \| p(z \mid x, \theta_i)) = 0$



EM maximizes the ELBO / minimizes Free Energy

a more general view

$$\log p(x \mid \theta) = \mathcal{L}(q, \theta) + D_{\mathrm{KL}}(q \| p(z \mid x, \theta))$$

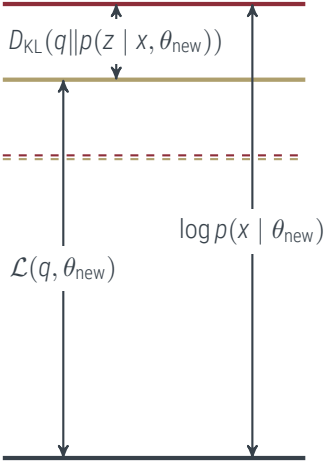
$$\mathcal{L}(q, \theta) = \int q(z) \log \left(\frac{p(x, z \mid \theta)}{q(z)} \right) dz$$

$$D_{\mathrm{KL}}(q \| p(z \mid x, \theta)) = -\int q(z) \log \left(\frac{p(z \mid x, \theta)}{q(z)} \right) dz$$

E -step: $q(z) = p(z \mid x, \theta_{\mathrm{old}})$, thus $D_{\mathrm{KL}}(q \| p(z \mid x, \theta_i)) = 0$

M -step: **Maximize ELBO**

$$\begin{aligned} \theta_{\mathrm{new}} &= \arg\max_{\theta} \int q(z) \log p(x, z \mid \theta) \, dz \\ &= \arg\max_{\theta} \mathcal{L}(q, \theta) + \int q(z) \log q(z) \, dz \end{aligned}$$



Variational Inference

- is a general framework to construct approximating **probability distributions** $q(z)$ to non-analytic posterior distributions $p(z \mid x)$ by minimizing the **functional**

$$q^* = \arg\min_{q \in \mathcal{Q}} D_{KL}(q(z) || p(z \mid x)) = \arg\max_{q \in \mathcal{Q}} \mathcal{L}(q)$$

- the beauty is that we get to *choose* q, so one can nearly always find a tractable approximation.
- If we impose the *mean field approximation* $q(z) = \prod_i q(z_i)$, get

$$\log q_j^*(z_j) = \mathbb{E}_{q, i \neq j}(\log p(x, z)) + \text{const}..$$

- for Exponential Family p things are particularly simple: we only need the expectation under q of the sufficient statistics.

Variational Inference is an extremely flexible and powerful approximation method. Its downside is that constructing the bound and update equations can be tedious. For a quick test, variational inference is often not a good idea. But for a deployed product, it can be the most powerful tool in the box.

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

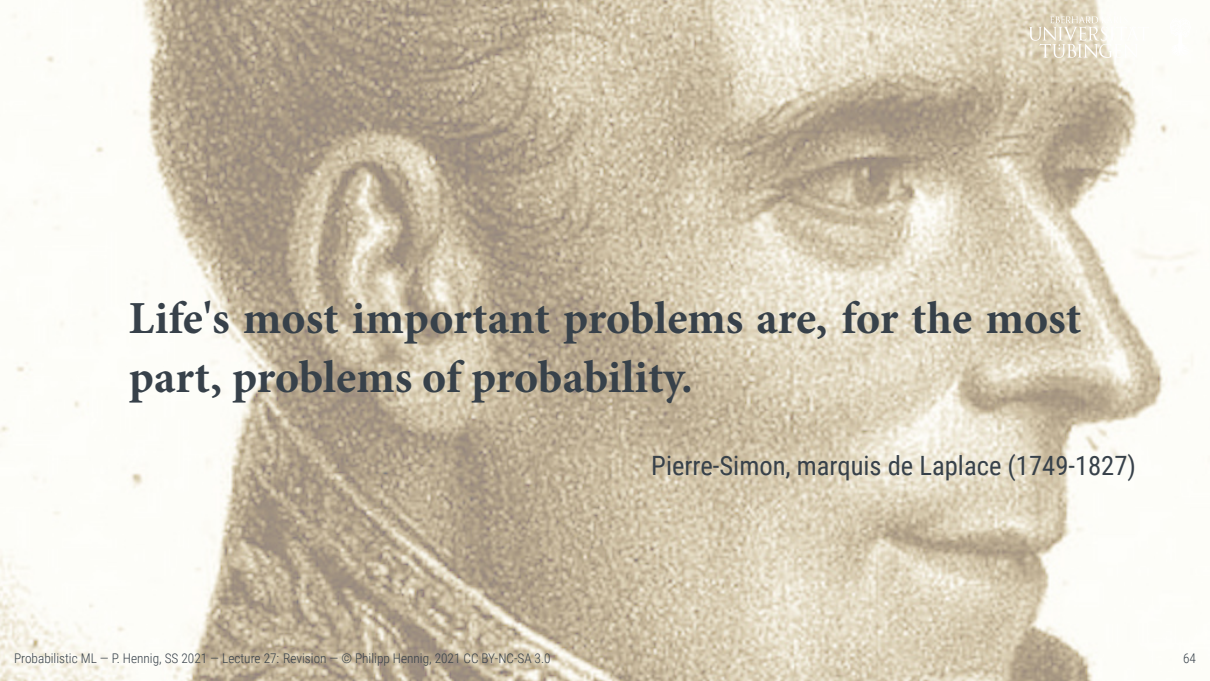
Computation:

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

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

Packaged solutions can give great first solutions, fast.
Building a tailormade solution requires creativity and mathematical stamina.



Life's most important problems are, for the most part, problems of probability.

Pierre-Simon, marquis de Laplace (1749-1827)