Probability Theory Review

Table of Contents

Challenge: Disease Diagnosis

Problem

- Given is a disease with a prevalence of `1%` and a test procedure with sensitivity ('true positive' rate) of `95%`, and specificity ('true negative' rate) of `85%`. What is the chance that somebody who tests positive actually has the disease?

Solution

- Later in this lecture, making use (only) of the sum and product rules of probability theory.

Probability Theory as Rational Reasoning

Propositional (Boolean) Logic

Define an **event** (or "proposition") A as a statement that can be considered for its truth by a person. For instance,

$$A = \text{“}\texttt{there is life on Mars}\text{”}$$

Boolean logic (or propositional logic) is a formal system of logic based on binary truth values: every proposition is either true (with assigned value 1) or false (with assigned value 0). It is named after George Boole, who developed the algebraic formulation of logic in the mid-19th century.

With Boolean operators ($\vee$, $\wedge$, $\implies$, etc.), we can create and evaluate compound propositions, e.g.,

- Given two events A and B, the **conjunction** (logical-and)

$$A \wedge B$$

is true if-and-only-if both A and B are true. We write $A \wedge B$ also shortly as AB (or use a comma as in a joint probability distribution $p(A, B)$).

- The **disjunction** (logical-or)

$$A \vee B$$

is true if either A or B is true or both A and B are true. We write $A \vee B$ also as $A + B$ (Note that the plus-sign is here not an arithmetic operator, but rather a logical operator to process truth values.)

- The denial of A, i.e., the event **not**-A, is written as

$$\bar{A}.$$

Boolean logic provides the rules of inference for **deductive reasoning** and underpins all formal reasoning systems in mathematics and philosophy.

Probability as a Degree-of-Belief

In the real world, we are rarely completely certain about anything. Rather than assigning a binary truth value to a proposition A, we associate it with a degree of belief

$$0 \leq p(A) \leq 1\,,$$

which quantifies how likely we believe A is to be true.

Consider the truth value of the proposition

$$A = \text{“}\texttt{there is life on Mars}\text{”}$$

with

$$I = \text{“}\texttt{All known life forms require water}\text{”}$$

as background information. Now, assume that a new piece of information

$$B = \text{“}\texttt{There is water on Mars}\text{”}$$

becomes available, how **should** our degree of belief in event A be affected *if we were rational*?

Richard Cox and the Calculus of Rational Reasoning

Richard T. Cox (1946) developed a **calculus for rational reasoning** about how to represent and update the **degree-of-belief** about the truth value of an event when faced with new information.

In developing this calculus, only some very agreeable assumptions were made, including:

- Degrees of belief are represented by real numbers.
- Plausibility assessments are consistent, e.g.,
    - if A becomes more plausible under new information B, the assigned degree-of-belief should increase accordingly.
    - If the belief in A is greater than the belief in B, and the belief in B is greater than the belief in C, then the belief in A must be greater than the belief in C.
- Logical equivalences are preserved, e.g.,
    - If the belief in an event can be inferred in two different ways, for example, either by first updating on information I_1 and then I_2 or the other way around, then the two

ways must agree on the resulting belief.

Under these assumptions, Cox showed that any consistent system of reasoning about uncertainty must obey the **rules of probability theory** (see Cox theorem, 1946, and Caticha, 2012, pp.7-26). These rules are the sum and product rules.

The Sum and Product Rules

The sum rule

- The degree of belief in the disjunction of two events A and B, with given background information I, is evaluated as

$$p(A + B|I) = p(A|I) + p(B|I) - p(A, B|I)$$

The product rule

- The degree of belief in the conjunction of two events A and B, with given background information I, is evaluated as

$$p(A, B|I) = p(A|B, I)\, p(B|I)$$

Cox's Theorem derives the rules of probability theory from first principles, not as arbitrary postulates but as consequences of rational reasoning. In other words: **probability theory = extended logic**.

> 🎯 **Key concept**
>
> When real numbers are used to represent **degrees of belief**, logical consistency enforces the sum and product rules of probability theory, making probability theory the optimal framework for information processing under uncertainty.

Probability Theory Calculus

True and False Events

In probability theory, events that are certainly true or certainly false are treated as special cases of general events, and they correspond to probabilities of 1 and 0, respectively.

Let Ω be the sample space, i.e., the set of all possible outcomes of an experiment. Then:

- The true event corresponds to the entire sample space Ω.
- It always happens, regardless of the outcome.
    - For example, for the outcome of a throw of a dice, the sample space is $\Omega = \{1, 2, 3, 4, 5, 6\}$.
- Its probability is

$$p(\Omega) = 1\,.$$

The false event corresponds to the empty set $\emptyset$.

- It never happens (no possible outcomes).
- Its probability is

$$p(\emptyset) = 0\,.$$

Independent, Exclusive, and Exhaustive Events

It will be helpful to introduce some terms concerning special relationships between events.

Joint events

The expression $p(A, B)$ for the probability of the conjuction $A \wedge B$ is also called the **joint probability** of events A and B. Note that

$$p(A, B) = p(B, A)\,,$$

since $A \wedge B = B \wedge A$. Therefore, the order of arguments in a joint probability distribution does not matter: $p(A, B, C, D) = p(C, A, D, B)$, etc.

Independent events

Two events A and B are said to be **independent** if the probability of one event is not altered by information about the truth of the other event, i.e.,

$$p(A|B) = p(A)\,.$$

It follows that, if A and B are independent, then the product rule simplifies to

$$p(A,B) = p(A)p(B)\,.$$

A and B with given background I are said to be **conditionally independent** for given I, if

$$p(A|B,I) = p(A|I)\,.$$

In that case, the product rule simplifies to $p(A,B|I) = p(A|I)p(B|I)$.

Mutually exclusive events

Two events A_1 and A_2 are said to be **mutually exclusive** ('disjoint') if they cannot be true simultaneously, i.e., if

$$p(A_1,A_2) = 0\,.$$

For mutually exclusive events, probabilities add (this follows from the sum rule), hence

$$p(A_1 + A_2) = p(A_1) + p(A_2)$$

Collectively exhaustive events

A set of events $A_1, A_2, \ldots, A_N$ is said to be **collectively exhaustive** if one of the statements is necessarily true, i.e., $A_1 + A_2 + \cdots + A_N = \mathrm{TRUE}$, or equivalently

$$p(A_1 + A_2 + \cdots + A_N) = 1\,.$$

Partitioning the universe

If a set of events $A_1, A_2, \ldots, A_n$ is both **mutually exclusive** and **collectively exhaustive**, then we say that they **partition the universe**. Technically, this means that

$$\sum_{n=1}^{N} p(A_n) = p(A_1 + \ldots + A_N) = 1$$

We mentioned before that every inference problem in PT can be evaluated through the sum and product rules. Next, we present two useful corollaries: (1) *Marginalization* and (2) *Bayes rule*.

Marginalization

Let A and $B_1, B_2, \ldots, B_n$ be events, where $B_1, B_2, \ldots, B_n$ partitions the universe. Then

$$\sum_{i=1}^{n} p(A, B_i) = p(A)\,.$$

This rule is called the law of total probability.

▼ **Click to see proof**

$$\begin{aligned}
\sum_i p(A, B_i) &= p\big(\sum_i AB_i\big) && \textbf{(since all } AB_i \textbf{ are disjoint)} \\
&= p\big(A, \sum_i B_i\big) \\
&= p(A, \Omega) && \textbf{(}\Omega \textbf{ is true event, since } B_i \textbf{ are exhaustive)} \\
&= p(A)
\end{aligned}$$

A very practical application of this law is to get rid of a variable that we are not interested in. For instance, if X and $Y \in \{y_1, y_2, \ldots, y_n\}$ are discrete variables, then

$$p(X) = \sum_{i=1}^{n} p(X, Y = y_i)\,.$$

Summing Y out of a joint distribution $p(X, Y)$ is called **marginalization** and the result $p(X)$ is sometimes referred to as the **marginal probability** of X.

Note that marginalization can be understood as applying a "generalized" sum rule. Bishop (p.14) and some other authors also refer to this as the sum rule, but we do not follow that terminology.

Bayes Rule

Consider two variables D and θ. It follows from symmetry arguments that

$$p(D, \theta) = p(\theta, D)\,,$$

and hence that

$$p(D|\theta)p(\theta) = p(\theta|D)p(D)\,,$$

or equivalently,

$$p(\theta|D) = \frac{p(D|\theta)}{p(D)}p(\theta)\,. \qquad \text{(Bayes rule)}$$

This last formula is called **Bayes rule**, named after its inventor Thomas Bayes (1701-1761). While Bayes rule is always true, a particularly useful application occurs when D refers to an observed data set and θ is set of unobserved model parameters. In that case,

- the **prior** probability $p(\theta)$ represents our **state-of-knowledge** about proper values for θ, before seeing the data D.
- the **posterior** probability $p(\theta|D)$ represents our state-of-knowledge about θ after we have seen the data.

Bayes rule tells us how to update our knowledge about model parameters when facing new data. Hence,

> 🎯 **Key concept**
>
> Bayes rule is the fundamental rule for learning from data!

Probabilistic Inference

Probabilistic inference refers to computing

$$p(\,\textbf{whatever-we-want-to-know}\,|\,\textbf{whatever-we-already-know}\,)$$

For example:

$$p(\,\textbf{Mr.S.-killed-Mrs.S.}\,|\,\textbf{he-has-her-blood-on-his-shirt}\,)$$
$$p(\,\textbf{transmitted-codeword}\,|\,\textbf{received-codeword}\,)$$

This can be accomplished by repeatedly applying the sum and product rules.

In particular, consider a joint distribution $p(X, Y, Z)$. Assume we are interested in $p(X|Z)$:

$$p(X|Z) \overset{p}{=} \frac{p(X,Z)}{p(Z)} \overset{s}{=} \frac{\sum_Y p(X,Y,Z)}{\sum_{X,Y} p(X,Y,Z)},$$

where the s and p above the equality sign indicate whether the sum or product rule was used.

In the rest of this course, we'll encounter many lengthy probabilistic derivations. For each manipulation, you should be able to associate an 's' (for sum rule), a 'p' (for product or Bayes rule) or an 'm' (for a simplifying model assumption like conditional independency) above any equality sign.

🎯 Key concept

All valid probabilistic relations can be derived from just two fundamental principles: the **sum rule** and the **product rule**. These two rules form the foundation of probability theory, from which more complex constructs such as conditional probabilities, Bayes' theorem, and marginalization naturally follow.

Examples

Inference Exercise: Which color does the ball have?

Problem

- A bag contains one ball, known to be either white or black. A white ball is put in , and the bag is shaken. Next, a ball is drawn out, which proves to be white. If we now take out another ball, what is the probability it will be white?

▼ **Click for the solution**

There are two hypotheses: let $H = 0$ mean that the original ball in the bag was white and $H = 1$ that it was black. Assume the prior probabilities are equal, i.e.,

$$P(H = 0) = 1/2, \quad P(H = 1) = 1/2\,.$$

The data is that when a randomly selected ball was drawn from the bag, which contained a white one and the unknown one, it turned out to be white. The probability of this result according to each hypothesis is:

$$P(D|H = 0) = 1, \quad P(D|H = 1) = 1/2\,.$$

So by Bayes theorem,

$$\begin{aligned} P(H = 0|D) &= \frac{P(H = 0, D)}{P(D)} \\ &= \frac{P(D|H = 0)P(H = 0)}{P(D|H = 0)P(H = 0) + P(D|H = 1)P(H = 1)} \\ &= \frac{1 \cdot \frac{1}{2}}{1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2}} \\ &= \frac{2}{3} \end{aligned}$$

and consequently,

$$P(H=1|D) = 1 - P(H=0|D) = \frac{1}{3}$$

Note that the physical state of the bag—either containing one black ball or one white ball—remains unchanged after the "put one in, take one out" procedure. Yet, the probability we assign to the color of the ball in the bag changes. This illustrates a key point: probabilities describe a person's state of knowledge, not an intrinsic property of nature.

Inference Exercise: Causality?

Problem

- A dark bag contains five red balls and seven green ones.
  - (a) What is the probability of drawing a red ball on the first draw?
- Balls are not returned to the bag after each draw.
  - (b) If you know that on the second draw the ball was a green one, what is now the probability of drawing a red ball on the first draw?

▼ **Click for the solution**

(a) $p(S_1 = R) = \frac{N_{\text{red}}}{N_{\text{red}}+N_{\text{green}}} = \frac{5}{12}$

(b) The outcome of the n-th draw is referred to by S_n. Use Bayes rule to get,

$$\begin{aligned} p(S_1 = \mathrm{R}|S_2 = \mathrm{G}) &= \frac{p(S_2 = \mathrm{G}|S_1 = \mathrm{R})p(S_1 = \mathrm{R})}{p(S_2 = \mathrm{G}|S_1 = \mathrm{R})p(S_1 = \mathrm{R}) + p(S_2 = \mathrm{G}|S_1 = \mathrm{G})p(S_1 = \mathrm{G})} \\ &= \frac{\frac{7}{11} \cdot \frac{5}{12}}{\frac{7}{11} \cdot \frac{5}{12} + \frac{6}{11} \cdot \frac{7}{12}} \\ &= \frac{5}{11} \end{aligned}$$

In this case, knowledge about the future influences our state of knowledge about the present. Once again, we see that conditional probabilities capture implications for a state of knowledge, not temporal causality.

Challenge Revisited: Disease Diagnosis

Problem

- Given a disease $D \in \{0, 1\}$ with prevalence (overall occurence percentage) of `1%` and a test procedure $T \in \{0, 1\}$ with sensitivity (true positive rate) of `95%` and specificity (true negative' rate) of `85%`, what is the chance that somebody who tests positive actually has the disease?

The percentages are interactive! ***Click and drag*** *to change the values.*

Show solution? ☐

Summary

Key concept ↑ Jump to source

When real numbers are used to represent **degrees of belief**, logical consistency enforces the sum and product rules of probability theory, making probability theory the optimal framework for information processing under uncertainty.

Key concept ↑ Jump to source

Bayes rule is the fundamental rule for learning from data!

Key concept ↑ Jump to source

All valid probabilistic relations can be derived from just two fundamental principles: the **sum rule** and the **product rule**. These two rules form the foundation of probability theory, from which more complex constructs such as conditional probabilities, Bayes' theorem, and marginalization naturally follow.

← Previous lecture

Next lecture →

Code

```
using BmlipTeachingTools
```

```
using Plots, LaTeXStrings
```

```
using Distributions
```

Disease diagnosis implementation

```
md"""
##### Disease diagnosis implementation
"""
```

```
begin
    prevalence_bond = @bind prevalence Scrubbable(0:0.01:1; format=".0%",
    default=0.01)
    sensitivity_bond = @bind sensitivity Scrubbable(0:0.01:1; format=".0%",
    default=0.95)
    specificity_bond = @bind specificity Scrubbable(0:0.01:1; format=".0%",
    default=0.85)
end;
```

```
result = 0.060126582278481
```

```
# The Disease Diagnosis example uses a combination of:
# - PlutoUI.Scrubbable for the interactive input percentages
# - MarkdownLiteral to be able to interpolate numbers into markdown math
# - PrintF for consistent formatting
using MarkdownLiteral: @mdx
```

```
using Printf
```

```
n (generic function with 1 method)
```

```
n(x; digits=2) = @sprintf("%.*f", digits, x)
```

Bayesian Machine Learning

Table of Contents

Challenge: Predicting a Coin Toss

Problem

We observe the following sequence of heads (outcome $= 1$) and tails (outcome $= 0$) when tossing the same coin repeatedly.

Generate a sample

7

$$D = \{0100001\}\,.$$

What is the probability that heads comes up next?

Solution

Later in this lecture.

The Bayesian Modeling Approach

Suppose that your application is to predict a future observation $\boldsymbol{x}$, based on N past observations $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$.

The **Bayesian modeling** approach to solving this task involves four stages:

```
REPEAT
    1. Model Specification
    2. Parameter Estimation
    3. Model Evaluation
UNTIL model performance is satisfactory
    4. Apply Model
```

In principle, based on the model evaluation results, you may want to re-specify your model and *repeat* the design process (a few times), until model performance is acceptable.

Crucially, **all information processing in Bayesian machine learning is governed strictly by probability theory**. Consequently, parameter estimation, model evaluation, and model application are all carried out through probabilistic inference.

> 🎯 **Key concept**
>
> Bayesian machine learning is a subfield of machine learning that commits entirely to probability theory as the framework for all information processing tasks. This is well justified, because probability theory is the optimal calculus for representing and manipulating states of knowledge.

Next, we discuss these four stages in a bit more detail.

1. Model Specification

Your first task is to propose a probabilistic model for generating the observations $\boldsymbol{x}$.

A probabilistic model m consists of a joint distribution $p(\boldsymbol{x}, \theta|m)$ that relates observations $\boldsymbol{x}$ to model parameters θ. Usually, the model is proposed in the form of a data-generating distribution

$p(x|\theta, m)$ and a prior $p(\theta|m)$,

$$\underbrace{p(x, \theta|m)}_{\text{model}} = \underbrace{p(x|\theta, m)}_{\substack{\text{data} \\ \text{generation}}} \underbrace{p(\theta|m)}_{\text{prior}} .$$

You are responsible to choose the data generating distribution $p(x|\theta)$ based on your physical understanding of the data generating process. (For brevity, if we are working on one given model m with no alternative models, we usually drop the given dependency on m from the notation).

You must also choose the prior $p(\theta)$ to reflect what you know about the parameter values before you see the data D.

2. Parameter Estimation

You must now specify a likelihood function for the parameters from the data-generating distribution. Note that, for a given (i.e., *observed*) data set $D = \{x_1, x_2, \ldots, x_N\}$ with *independent* observations x_n (drawn from the same model θ), the likelihood factorizes as

$$p(D|\theta) = \prod_{n=1}^{N} p(x_n|\theta) .$$

The likelihood and prior both contain information about the model parameters. Next, you use Bayes rule to fuse these two information sources into a posterior distribution for the parameters:

$$\underbrace{p(\theta|D)}_{\text{posterior}} = \frac{\overbrace{p(D|\theta)}^{\text{likelihood}} \overbrace{p(\theta)}^{\text{prior}}}{\underbrace{\int p(D|\theta) p(\theta) \mathrm{d}\theta}_{p(D)\text{ (evidence)}}}$$

Note that there's **no need for you to design some clever parameter estimation algorithm**. Bayes rule *is* the parameter estimation algorithm, which can be entirely expressed in terms of the likelihood and prior. The only complexity lies in the computational issues (in particular, the computational load of computing the evidence)!

3. Model Evaluation

Let's assume that we have more candidate models, say $\mathcal{M} = \{m_1, \ldots, m_K\}$ where each model relates to a specific prior $p(\theta|m_k)$ and likelihood $p(D|\theta, m_k)$? Can we evaluate the relative performance of a model against another model from the set?

Start again with **model specification**. *You* must now specify a *model* prior $p(m_k)$ (next to the likelihood $p(D|\theta, m_k)$ and *parameter* prior $p(\theta|m_k)$) for each of the models to get a new model specification that includes the model m_k as a parameter:

$$p(D, \theta, m_k) = p(D|\theta, m_k)p(\theta|m_k)p(m_k)$$

Then, solve the desired inference problem for the posterior over the model m_k:

$$\underbrace{p(m_k|D)}_{\substack{\text{model} \\ \text{posterior}}} \propto \underbrace{p(m_k)}_{\substack{\text{model} \\ \text{prior}}} \underbrace{\int_\theta \underbrace{p(D|\theta, m_k)}_{\text{likelihood}} \underbrace{p(\theta|m_k)}_{\substack{\text{parameter} \\ \text{prior}}} \mathrm{d}\theta}_{\substack{\text{evidence } p(D|m_k) \\ = \text{ model likelihood}}}$$

▼ **Prove this yourself, and click for solution**

$$\begin{aligned} p(m_k|D) &= \frac{p(m_k, D)}{p(D)} \\ &\propto p(m_k, D) \\ &= \int_\theta p(D, \theta, m_k)\, \mathrm{d}\theta \\ &= p(m_k) \int_\theta p(D|\theta, m_k)\, p(\theta|m_k)\, \mathrm{d}\theta \end{aligned}$$

Again, **no need to invent a special algorithm for estimating the performance of your model**. Straightforward application of probability theory takes care of all that.

🎯 **Key concept**

In a Bayesian modeling framework, **model evaluation** follows the same recipe as parameter estimation; it just works at one higher hierarchical level.

4. Apply Model (Prediction)

Given the data D, our knowledge about a yet unobserved datum x is captured by the following inference problem (where everything is conditioned on the selected model):

$$p(x|D) = \int \underbrace{p(x|\theta)}_{\substack{\text{data} \\ \text{generating}}} \underbrace{p(\theta|D)}_{\text{posterior}} \mathrm{d}\theta$$

▼ **Prove this yourself, and click for solution**

$$\begin{aligned} p(x|D) &\overset{s}{=} \int p(x, \theta|D)\, \mathrm{d}\theta \\ &\overset{p}{=} \int p(x|\theta, D) p(\theta|D)\, \mathrm{d}\theta \\ &\overset{m}{=} \int p(x|\theta)\, p(\theta|D)\, \mathrm{d}\theta \end{aligned}$$

In the last equation, the simplification $p(x|\theta, D) = p(x|\theta)$ follows from our model specification. In particular, we assumed a *parametric* data generating distribution $p(x|\theta)$ with no explicit dependency on the data set D. Technically, in our model specification, we assumed that x is conditionally independent from D, given the parameters θ, i.e., we assumed $p(x|\theta, D) = p(x|\theta)$. The information from the data set D has been absorbed in the posterior $p(\theta|D)$, so all information from D is passed to a new observation x through the (posterior distribution over the) parameters θ.

We're DONE! Again, there is no need to invent a special prediction algorithm. Probability theory takes care of all that. Your problems are only of computational nature. Perhaps the integral to compute the evidence may not be analytically tractable, how to carry out the marginalization over θ, etc.

🎯 **Key concept**

Bayesian Machine learning is EASY, apart from computational details :)

Bayesian Evidence as a Model Performance Criterion

I'd like to convince you that Bayesian model evidence $p(D|m)$ is an excellent criterion for assessing your model's performance. To do so, let us consider a decomposition that relates model evidence to other highly-valued criteria such as **accuracy** and **model complexity**.

Consider a model $p(x, \theta|m)$ and a data set $D = \{x_1, x_2, \ldots, x_N\}$. Given the data set D, the log-evidence for model m decomposes as

$$\underbrace{\log p(D|m)}_{\text{log-evidence}} = \underbrace{\int p(\theta|D,m) \log p(D|\theta,m) \mathrm{d}\theta}_{\text{accuracy (a.k.a. data fit)}} - \underbrace{\int p(\theta|D,m) \log \frac{p(\theta|D,m)}{p(\theta|m)} \mathrm{d}\theta}_{\text{complexity}}.$$

▼ Click to see proof

$$\begin{aligned}
\log p(D|m) &= \log p(D|m) \cdot \underbrace{\int p(\theta|D,m) \mathrm{d}\theta}_{\text{evaluates to 1}} \\
&= \int p(\theta|D,m) \log p(D|m) \mathrm{d}\theta \qquad (\text{move } \log p(D|m) \text{ into the integral}) \\
&= \int p(\theta|D,m) \log \underbrace{\frac{p(D|\theta,m)p(\theta|m)}{p(\theta|D,m)}}_{\text{by Bayes rule}} \mathrm{d}\theta \\
&= \underbrace{\int p(\theta|D,m) \log p(D|\theta,m) \mathrm{d}\theta}_{\text{accuracy (a.k.a. data fit)}} - \underbrace{\int p(\theta|D,m) \log \frac{p(\theta|D,m)}{p(\theta|m)} \mathrm{d}\theta}_{\text{complexity}}
\end{aligned}$$

accuracy

The "accuracy" term (also known as data fit) measures how well the model predicts the data set D. We want this term to be high because good models should predict the data D well. Indeed, higher

accuracy leads to higher model evidence. To achieve high accuracy, applying Bayes' rule will shift the posterior $p(\theta|D)$ away from the prior towards the likelihood function $p(D|\theta)$.

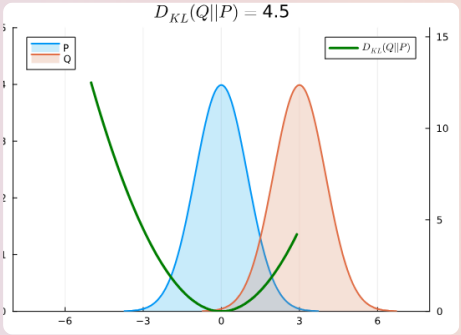
complexity

The second term ("complexity", also known as "information gain") is technically a Kullback-Leibler divergence (KLD) between the posterior and prior distributions, see this mini lecture. The KLD is an information-theoretic quantity that can be interpreted as a "distance" measure between two distributions. In other words, the complexity term measures how much the beliefs about θ changed, due to learning from the data D. Generally, we like the complexity term to be low, because moving away means forgetting previously acquired information represented by the prior. Indeed, lower complexity leads to higher model evidence.

Focussing only on accuracy maximization could lead to *overfitting* of the data set D. Focussing only on complexity minimization could lead to *underfitting* of the data. Bayesian ML attends to both terms and avoids both underfitting and overfitting.

🎯 Key concept

Bayesian learning automatically leads to models that generalize well. There is **no need for early stopping or validation data sets**. There is also **no need for tuning parameters** in the learning process. Just learn on the full data set and all behaves well.



Mini: The Kullback-Leibler Divergence

Introduction to the Kullback-Leibler Divergence with an interactive example.

Read article →

Challenge Revisited: Predicting a Coin Toss

Data Generation

Let's generate a sequence of N coin tosses $D = \{x_1, \ldots, x_N\}$, where each throw is drawn from a Bernoulli distribution

$$p(x_n|\mu = 0.4) = 0.4^{x_n} \cdot 0.6^{1-x_n}\,,$$

and where x_n denotes outcomes by

$$x_n = \begin{cases} 1 & \text{if heads comes up} \\ 0 & \text{otherwise (tails)} \end{cases}$$

So, this coin is biased!

What is the probability that heads comes up next? We solve this in the next slides ...

1. Model Specification for Coin Toss

Likelihood

Assume a **Bernoulli** distributed variable $p(x_k = 1|\mu) = \mu$ for a single coin toss, leading to

$$p(x_k|\mu) = \mu^{x_k}(1-\mu)^{1-x_k}\,.$$

Assume n times heads were thrown out of a total of N throws. The likelihood function then follows a **binomial** distribution :

$$p(D|\mu) = \prod_{k=1}^{N} p(x_k|\mu) = \mu^n(1-\mu)^{N-n}$$

Prior

Assume the prior beliefs for μ are governed by a **beta distribution**

$$p(\mu) = \mathrm{Beta}(\mu|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\mu^{\alpha-1}(1-\mu)^{\beta-1}$$

where the Gamma function is sort-of a generalized factorial function. In particular, if α, β are integers, then

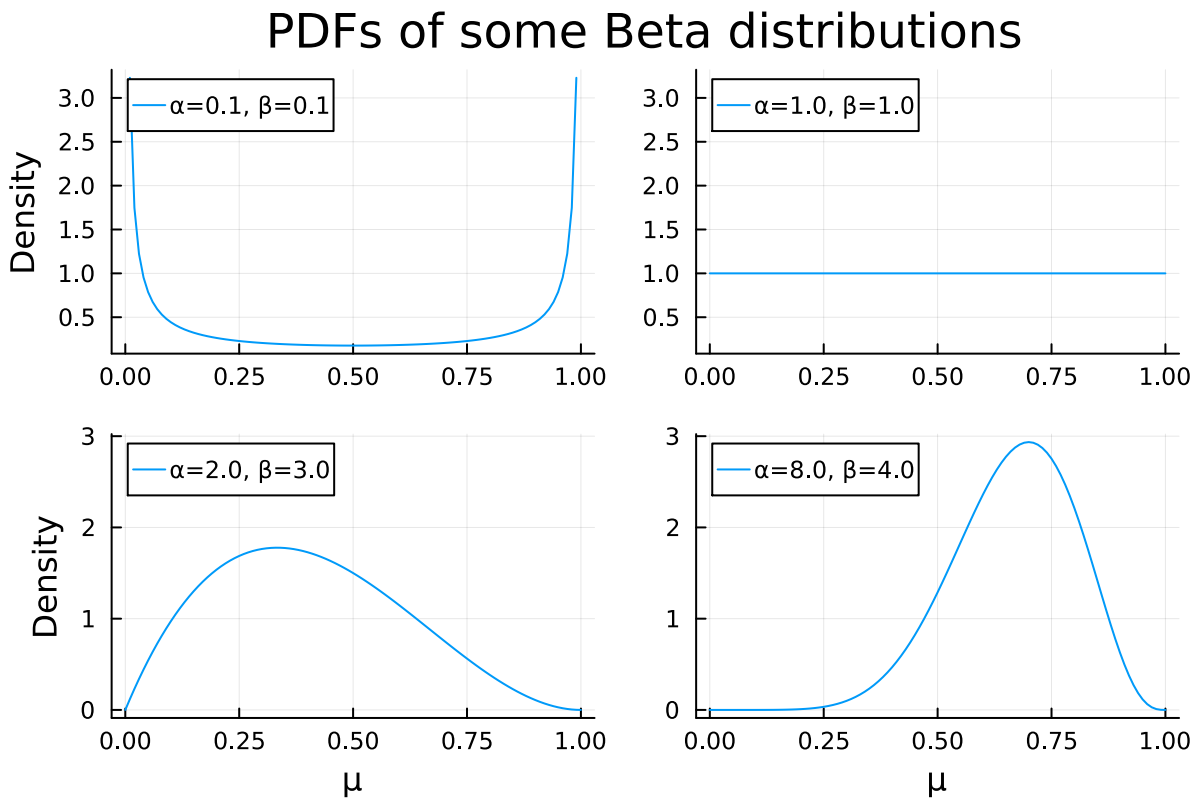
$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} = \frac{(\alpha+\beta-1)!}{(\alpha-1)!\,(\beta-1)!}$$

A *what* distribution? Yes, the **beta distribution** is a **conjugate prior** for the binomial distribution, which means that

$$\underbrace{\mathrm{beta}}_{\text{posterior}} \propto \underbrace{\mathrm{binomial}}_{\text{likelihood}} \times \underbrace{\mathrm{beta}}_{\text{prior}}$$

so we get a closed-form posterior.

α and β are called **hyperparameters**, since they parameterize the distribution for another parameter (μ). E.g., $\alpha = \beta = 1$ leads to a uniform prior for μ. Below, we visualize some priors $\mathrm{Beta}(\mu|\alpha,\beta)$ for different values of α, β.



Concretely, let's compare two models m_1, and m_2 with the same likelihood function, but with different priors:

$$
\begin{aligned}
p(\mu|m_1) &= \text{Beta}(\mu|\alpha=100, \beta=500) \\
p(\mu|m_2) &= \text{Beta}(\mu|\alpha=8, \beta=13).
\end{aligned}
$$

We can already guess which one is better!

You can verify that model m_2 has the best prior, since

$$
\begin{aligned}
p(x_n=1|m_1) &= \frac{\alpha}{\alpha+\beta}\bigg|_{m_1} = 100/600 \approx 0.17 \\
p(x_n=1|m_2) &= \frac{\alpha}{\alpha+\beta}\bigg|_{m_2} = 8/21 \approx 0.38\,,
\end{aligned}
$$

(but you are not supposed to know that the real coin has a probability 0.4 for heads.)

2. Parameter Estimation for Coin Toss

Next, infer the posterior PDF over μ (and evidence) through Bayes rule,

$$
\begin{aligned}
&p(D|\mu) \cdot p(\mu) \\
&= \underbrace{\left(\frac{B(n+\alpha, N-n+\beta)}{B(\alpha,\beta)} \right)}_{\text{evidence } p(D)} \cdot \underbrace{\left(\frac{1}{B(n+\alpha, N-n+\beta)} \mu^{n+\alpha-1} (1-\mu)^{N-n+\beta-1} \right)}_{\text{posterior } p(\mu|D) = \text{Beta}(\mu|n+\alpha, N-n+\beta)}
\end{aligned}
$$

where $B(\alpha,\beta) \triangleq \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$.

▼ **Click for proof**

$$
\begin{aligned}
&p(D|\mu) \cdot p(\mu) \\
&= \underbrace{\left(\mu^n (1-\mu)^{N-n} \right)}_{\text{likelihood}} \cdot \underbrace{\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1} (1-\mu)^{\beta-1} \right)}_{\text{prior}} \\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{n+\alpha-1} (1-\mu)^{N-n+\beta-1} \\
&= \frac{1}{B(\alpha,\beta)} \mu^{n+\alpha-1} (1-\mu)^{N-n+\beta-1} \\
&= \underbrace{\left(\frac{B(n+\alpha, N-n+\beta)}{B(\alpha,\beta)} \right)}_{\text{evidence } p(D)} \cdot \underbrace{\left(\frac{1}{B(n+\alpha, N-n+\beta)} \mu^{n+\alpha-1} (1-\mu)^{N-n+\beta-1} \right)}_{\text{posterior } p(\mu|D) = \text{Beta}(\mu|n+\alpha, N-n+\beta)}
\end{aligned}
$$

In the final equation, we included the term $\frac{1}{B(n+\alpha, N-n+\beta)}$ to normalize the posterior $p(\mu|D)$, and we compensated for this normalization in the evidence factor.

Hence, the posterior is also beta-distributed as

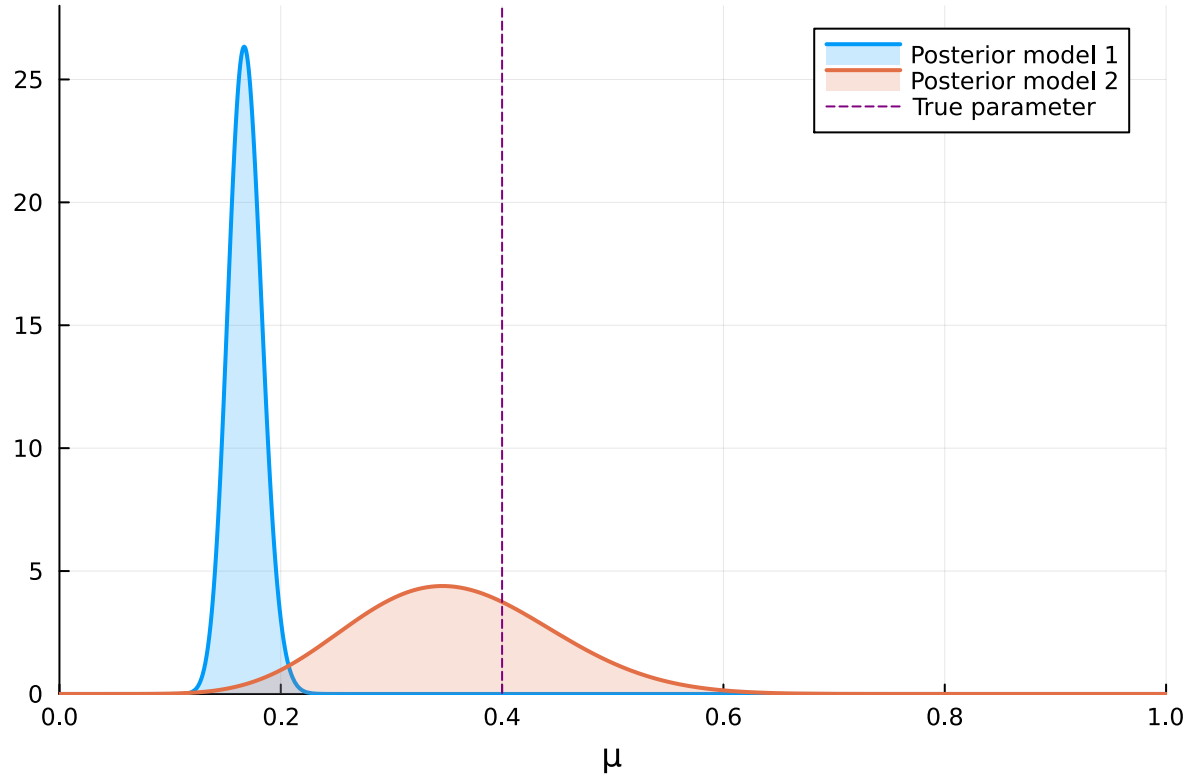
$$
p(\mu|D) = \text{Beta}(\mu|\, n+\alpha, N-n+\beta)
$$

For both models m_1 and m_2, we plot below the **parameter posteriors** $p(\mu|D_n, m_\bullet)$, computed after N coin tosses. Use the slider to change N.

Generate a sample

7

$$
D = \{0100001\}\,.
$$



3. Model Evaluation for Coin Toss

It follows from the above calculation that the evidence for model m can be analytically expressed as

$$p(D|m) = \frac{B(n+\alpha, N-n+\beta)}{B(\alpha,\beta)}$$

$$\left(= \frac{\Gamma(n+\alpha)\Gamma(N-n+\beta)}{\Gamma(N+\alpha+\beta)} \Big/ \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \cdot \right)$$

The model evidence is a scalar.

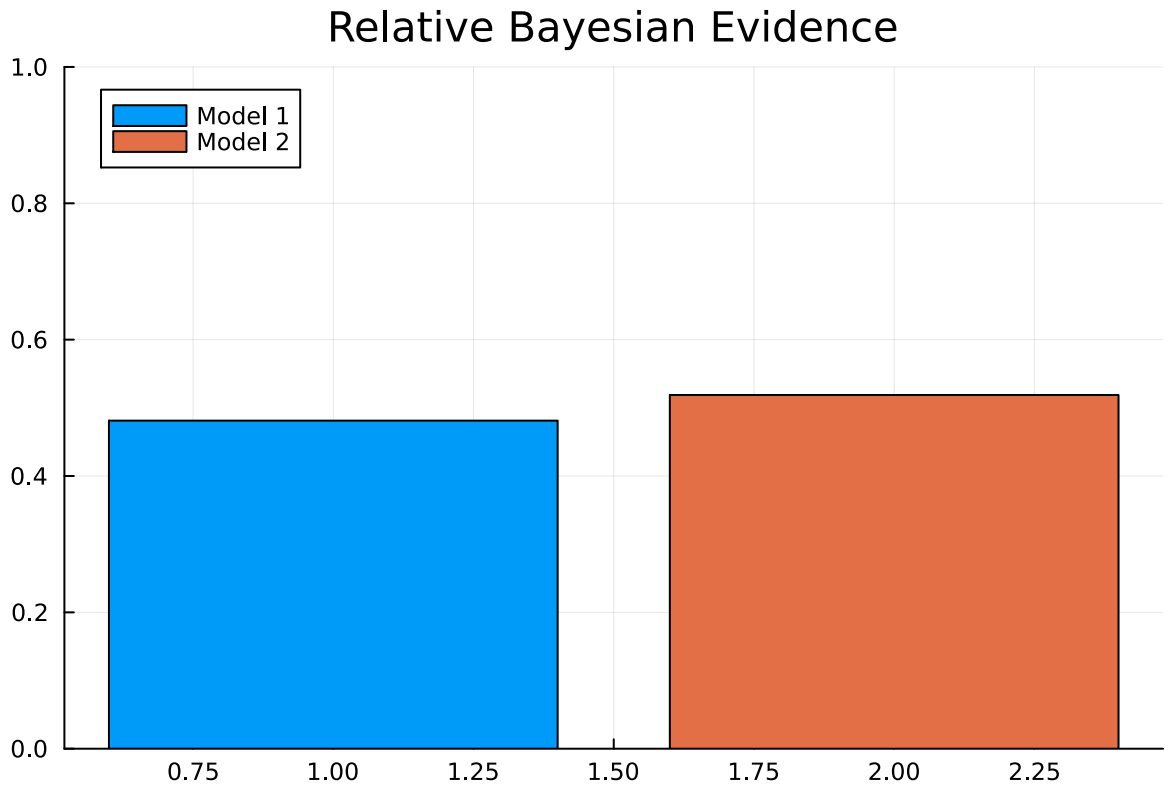
Let's check this by plotting over time the relative Bayesian evidences for each model:

$$\frac{p(D_n|m_i)}{\sum_{i=1}^{2} p(D_n|m_i)}$$

Generate a sample

7

$D = \{0100001\}\,.$



4. Prediction (Application) for Coin Toss

Once we have accepted a model, let's apply it to the application, in this case, predicting future observations.

Marginalize over the parameter posterior to get the predictive PDF for a new coin toss $x_\bullet$, given the data D,

$$\begin{aligned} p(x_\bullet = 1|D) &= \int_0^1 p(x_\bullet = 1|\mu)\, p(\mu|D)\, \mathrm{d}\mu \\ &= \int_0^1 \mu \times \mathrm{Beta}(\mu|\, n+\alpha, N-n+\beta)\, \mathrm{d}\mu \\ &= \frac{n+\alpha}{N+\alpha+\beta} \end{aligned}$$

This result is known as **Laplace's rule of succession**.

The above integral computes the mean of a beta distribution, which is given by $\mathbb{E}[x] = \frac{a}{a+b}$ for $x \sim \mathrm{Beta}(a, b)$, see wikipedia.

Generate a sample

7

$$D = \{0100001\}\,.$$

Finally, we're ready to solve our challenge: for the generated D, for $n = 2$ and $N = 7$, we get

$$p(x_\bullet = 1|D, m_1) = \frac{n + 100}{N + 100 + 500} \qquad \approx 0.168$$

$$p(x_\bullet = 1|D, m_2) = \frac{n + 8}{N + 8 + 13} \qquad \approx 0.357$$

Be aware that there is no such thing as an "objective" or "correct" prediction. Every prediction is conditional on the selected model and the used data set.

All Learning is Interpretable as Correcting Prediction Errors

What did we learn from the data? Before seeing any data, we stated that the probability of throwing heads is

$$p(x_\bullet = 1|D)|_{n=N=0} = \left.\frac{n + \alpha}{N + \alpha + \beta}\right|_{n=N=0} = \frac{\alpha}{\alpha + \beta}\,.$$

Hence, α and β can be interpreted as prior pseudo-counts for heads and tails, respectively.

If we were to assume zero pseudo-counts, i.e. $\alpha = \beta \to 0$, then our prediction for throwing heads after N coin tosses is completely based on the data, given by

$$p(x_\bullet = 1|D)|_{\alpha=\beta\to 0} = \left.\frac{n + \alpha}{N + \alpha + \beta}\right|_{\alpha=\beta\to 0} = \frac{n}{N}\,.$$

Note the following decomposition

$$
\begin{aligned}
p(x_\bullet = 1 | \, D) &= \frac{n+\alpha}{N+\alpha+\beta} \\
&= \underbrace{\frac{\alpha}{\alpha+\beta}}_{\substack{\text{prior} \\ \text{prediction}}} + \underbrace{\frac{N}{N+\alpha+\beta}}_{\text{gain}} \cdot \underbrace{\underbrace{\left(\underbrace{\frac{n}{N}}_{\substack{\text{data-based} \\ \text{prediction}}} - \underbrace{\frac{\alpha}{\alpha+\beta}}_{\substack{\text{prior} \\ \text{prediction}}} \right)}_{\text{prediction error}}}_{\text{correction}}
\end{aligned}
$$

▼ **Prove this yourself, and click for solution**

$$
\begin{aligned}
p(x_\bullet = 1 | \, D) &= \frac{n+\alpha}{N+\alpha+\beta} \\
&= \frac{\alpha}{N+\alpha+\beta} + \frac{n}{N+\alpha+\beta} \\
&= \frac{\alpha}{N+\alpha+\beta} \cdot \frac{\alpha+\beta}{\alpha+\beta} + \frac{n}{N+\alpha+\beta} \cdot \frac{N}{N} \\
&= \frac{\alpha}{\alpha+\beta} \cdot \frac{\alpha+\beta}{N+\alpha+\beta} + \frac{N}{N+\alpha+\beta} \cdot \frac{n}{N} \\
&= \frac{\alpha}{\alpha+\beta} \cdot \left(1 - \frac{N}{N+\alpha+\beta}\right) + \frac{N}{N+\alpha+\beta} \cdot \frac{n}{N} \\
&= \underbrace{\frac{\alpha}{\alpha+\beta}}_{\substack{\text{prior} \\ \text{prediction}}} + \underbrace{\frac{N}{N+\alpha+\beta}}_{\text{gain}} \cdot \underbrace{\underbrace{\left(\underbrace{\frac{n}{N}}_{\substack{\text{data-based} \\ \text{prediction}}} - \underbrace{\frac{\alpha}{\alpha+\beta}}_{\substack{\text{prior} \\ \text{prediction}}} \right)}_{\text{prediction error}}}_{\text{correction}}
\end{aligned}
$$

Let's interpret this decomposition of the posterior prediction. Before the data D was observed, our model generated a *prior prediction* $p(x_\bullet = 1) = \frac{\alpha}{\alpha+\beta}$. Next, the mismatch between the actually observed data and this prediction is represented by the *prediction error* $\frac{n}{N} - \frac{\alpha}{\alpha-\beta}$. The prior prediction is then updated to a *posterior prediction* $p(x_\bullet = 1 | D)$ by adding a fraction

$0 \leq \frac{N}{N+\alpha+\beta} < 1$ of the prediction error to the prior prediction. Hence, the **prediction error** is used to "correct" the prior prediction.

More Examples

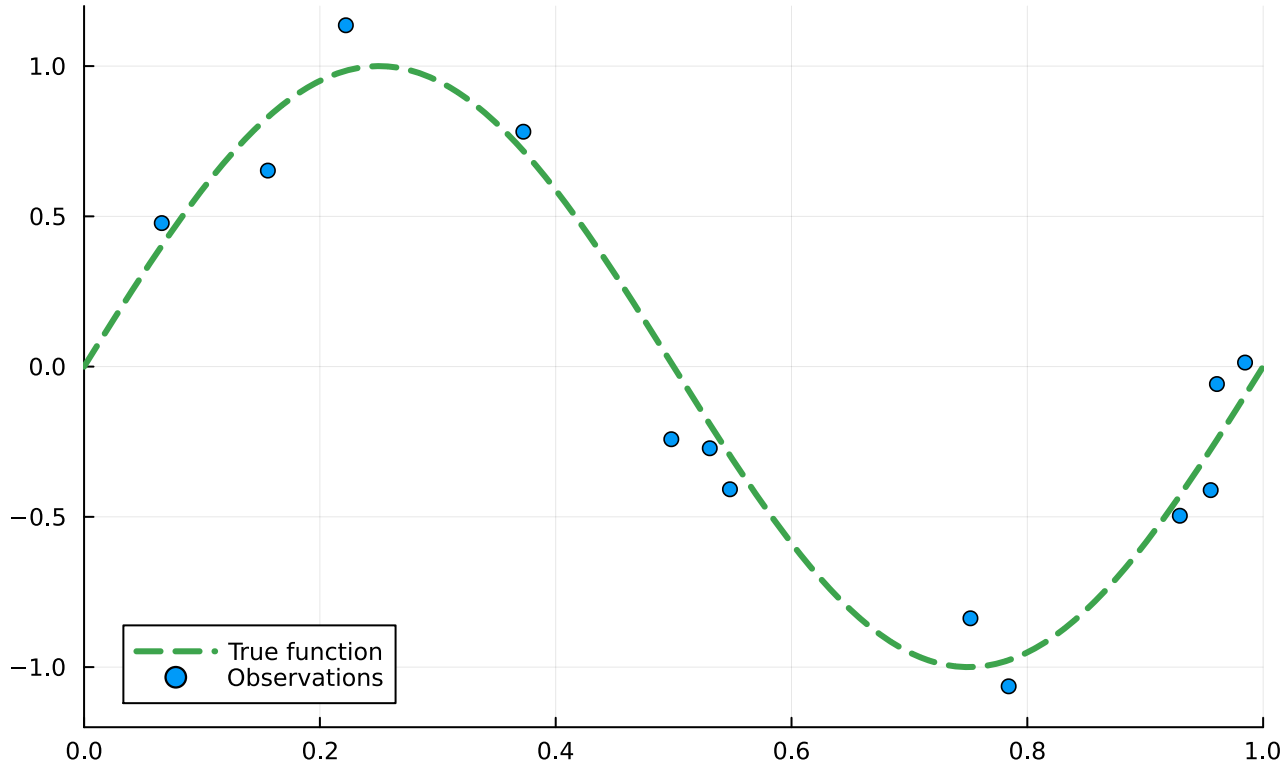
Example: Bayesian Linear Regression

Data

Assume a set of noisy observations $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where each observation satisfies $y_n = f(x_n) + \varepsilon_n$.

secret_function = sin(x * 2π)

N = 13



Challenge

The goal is to infer the underlying function f from the observed data points.

Model Specification

We assume that the data are generated according to the following model (see the Regression lecture for details):

$$\phi_k(x_n) = \exp\left(-\frac{(x_n - \mu_k)^2}{\sigma_\phi^2}\right)$$
$$p(y_n|x_n, w) = \mathcal{N}\left(y_n \middle| \sum_k w_k \phi_k(x_n), \sigma_y^2\right)$$
$$p(w) = \mathcal{N}\left(w|0, \sigma_w^2 I\right)$$

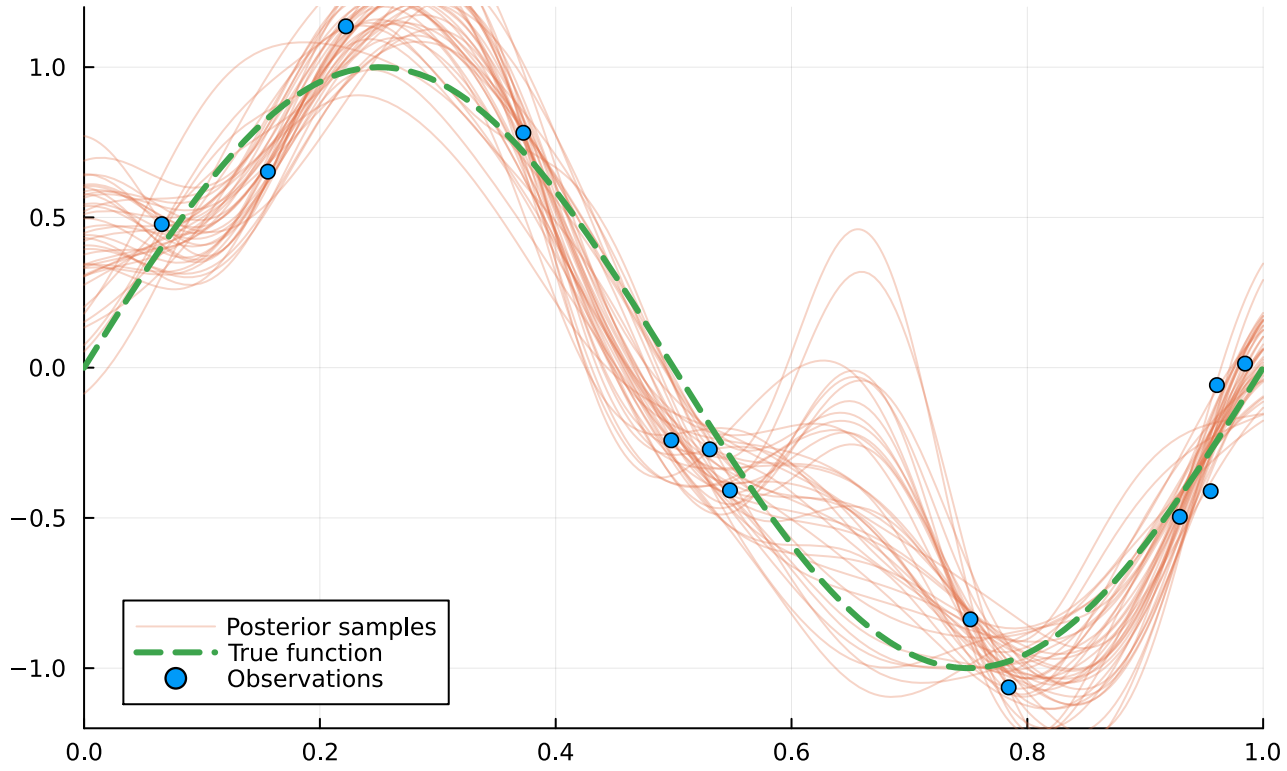
σ_data_noise = 0.12

σ_w_prior = 0.5

Results

Below, we plot 40 candidate functions, where each candidate corresponds to a draw from the posterior distribution $p(w|D)$.

N = 13

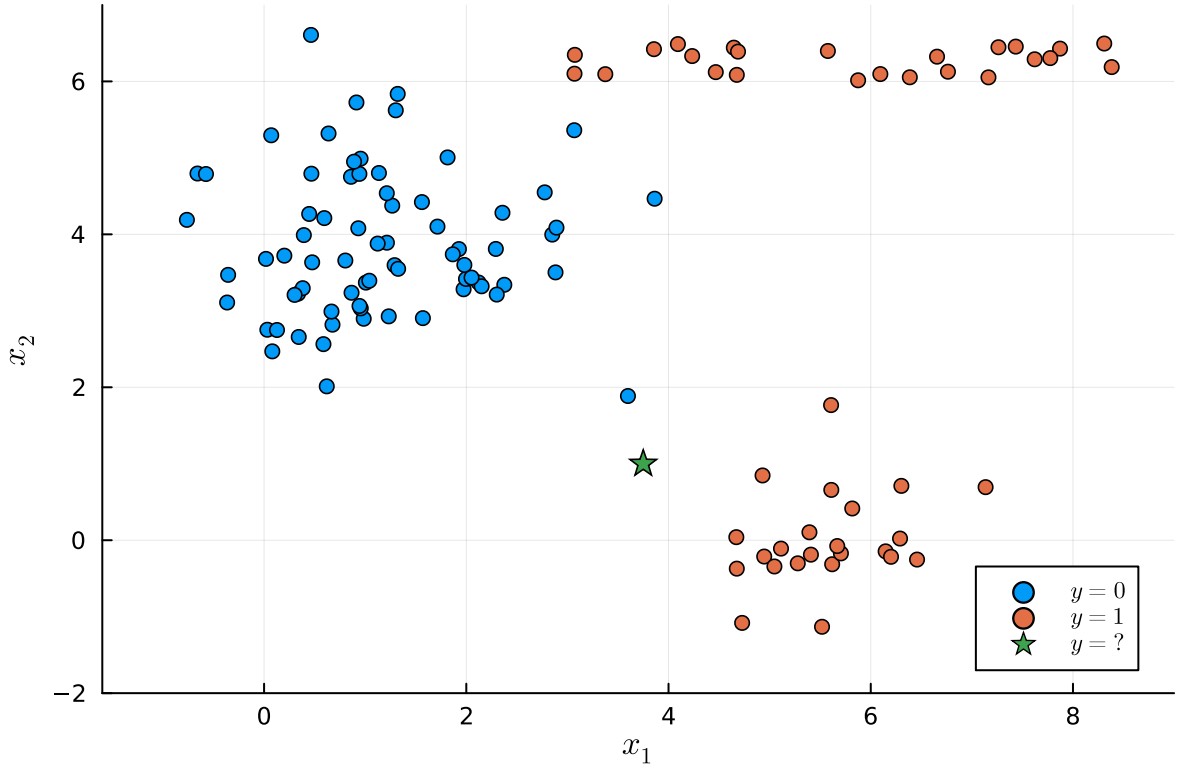


Example: Bayesian Logistic Regression (Classification)

Data

Assume a set of observations $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where each observation (x_n, y_n) holds a 2-dimensional **feature** vector $x_n \in \mathbb{R}^2$ and a **class label** $y_n \in \{0, 1\}$. Move the slider to generate more (or fewer) samples.

N2 = 120



Challenge

The (classification) challenge is to predict the class label for a new (unlabelled) sample $x_\bullet$ that is drawn from the same underlying process.

Model Specification

In a **logistic regression** model, we assume that the class labels are generated from given features by the following model (see the Discriminative Classification lecture for more details):

$$p(y_n = 1|x_n, w) = \sigma(w^T x_m)$$
$$p(w) = \mathcal{N}(w|m_0, S_0)$$

where $\sigma(a) = \frac{1}{1+e^{-a}}$ is the logistic function.

Results

After incorporating the data set D into the posterior (through Bayes rule)

$$p(w|D) = \mathcal{N}(w|m_N, S_N),$$

the predicted class label $y_\bullet$, given a new feature vector $x_\bullet$ and the data set D, can be worked out to (as shown in this reference)

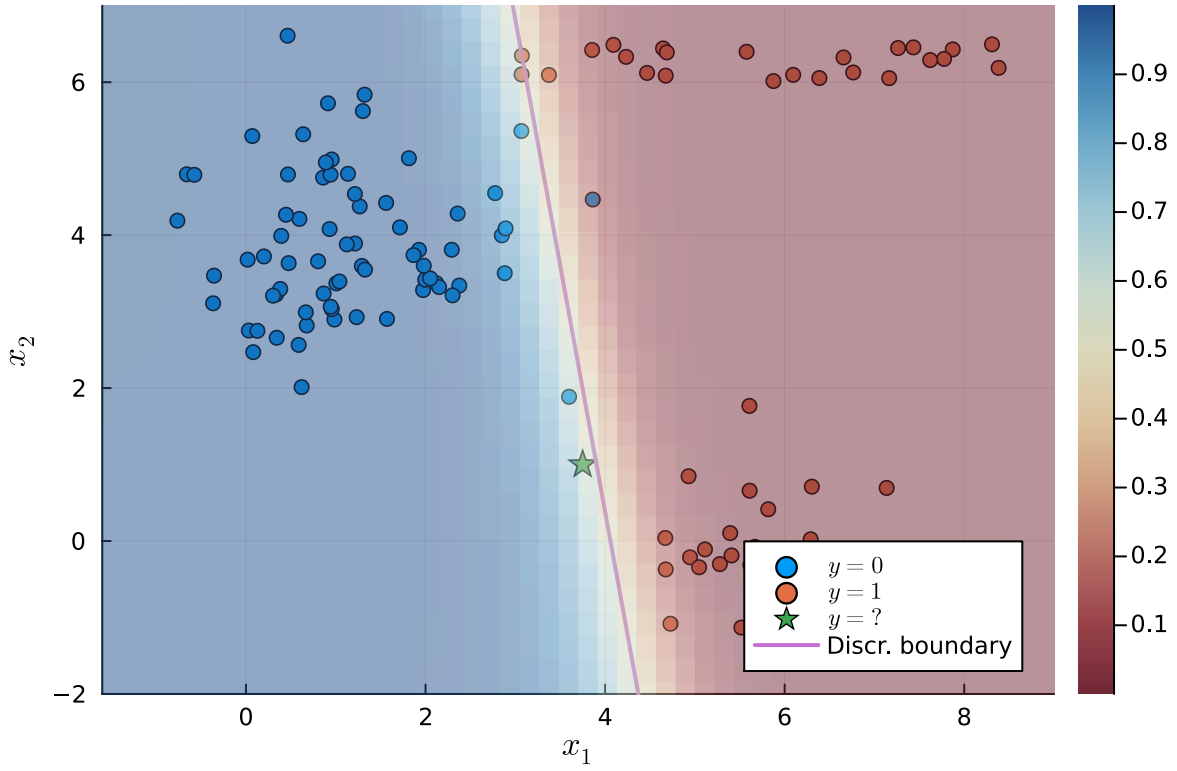
$$p(y_\bullet = 1|x_\bullet, D) = \int p(y_\bullet = 1|x_\bullet, w)p(w|D)\mathrm{d}w$$
$$\approx \Phi\left(\frac{m_N^T x_\bullet}{\sqrt{(8/\pi) + x_\bullet^T S_N x_\bullet}}\right)$$

where $\Phi(\cdot)$ is the cumulative Gaussian distribution.

The predicted class label distribution is shown in the heatmap below. The line indicates the discrimination boundary where

$$p(y_\bullet = 1|x_\bullet, D) = p(y_\bullet = 0|x_\bullet, D) = 0.5$$

N2 = 120


6
4
2
0
−2
x_2
0
2
4
6
8
x_1
$y = 0$
$y = 1$
$y = ?$
Discr. boundary
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

Summary

🎯 **Key concept** ↑ Jump to source

Bayesian machine learning is a subfield of machine learning that commits entirely to probability theory as the framework for all information processing tasks. This is well justified, because probability theory is the optimal calculus for representing and manipulating states of knowledge.

🎯 **Key concept** ↑ Jump to source

In a Bayesian modeling framework, **model evaluation** follows the same recipe as parameter estimation; it just works at one higher hierarchical level.

🎯 **Key concept** ↑ Jump to source

Bayesian Machine learning is EASY, apart from computational details :)

🎯 **Key concept** ↑ Jump to source

Bayesian learning automatically leads to models that generalize well. There is **no need for early stopping or validation data sets**. There is also **no need for tuning parameters** in the learning process. Just learn on the full data set and all behaves well.

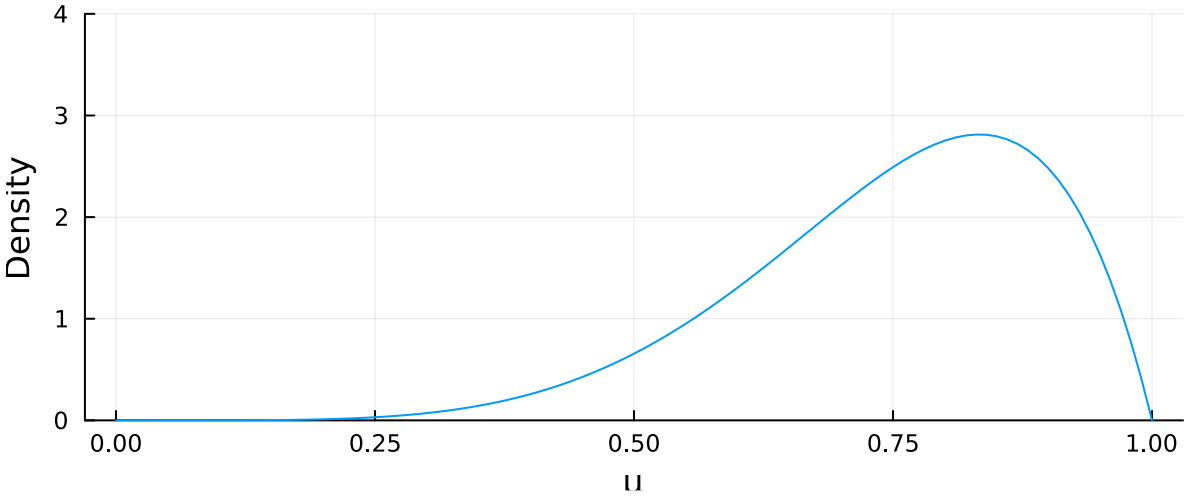
← Previous lecture

Next lecture →

Optional Slides

Pick Your Own Parameters for the Beta distribution

α = 6,0, β = 2,0



Code

```
using BmlipTeachingTools
```

```
using MarkdownLiteral: @mdx
```

```
using Distributions, Random, ExponentialFamily, LinearAlgebra, LogExpFunctions,
StatsFuns, BayesBase, Optim, SpecialFunctions, StableRNGs, Printf
```

```
using Plots, StatsPlots, LaTeXStrings, Plots.PlotMeasures
```

```
wideq (generic function with 1 method)
```

Coin Toss example code

```
secret_distribution = Bernoulli(0.4);
```

```
priors = [
    Beta(100., 500.),
    Beta(8., 13.)
];
```

```
precomputed_tosses =
▸ [false, true, false, false, false, false, true, true, true, true, false, false, true, false
```

```
precomputed_tosses = rand(StableRNG(234), secret_distribution, 500)
```

```
begin
    # save a sequence of posterior distributions for every prior, starting with the
    prior itself
    prior_distributions = [d for d in priors]
    posterior_distributions = [[d] for d in priors]
    log_evidences = [[] for _ in priors]


    # for every sample we want to update our posterior
    for (N, sample) in enumerate(precomputed_tosses)
        # at every sample we want to update all distributions
        for (i, prior) in enumerate(prior_distributions)

            # do bayesian updating
            posterior = handle_coin_toss(prior, sample)

            # add posterior to vector of posterior distributions
            push!(posterior_distributions[i], posterior)

            # compute log evidence and add to vector
            log_evidence = log_evidence_prior(prior, N,
            sum(@view(precomputed_tosses[1:N])))
            push!(log_evidences[i], log_evidence)

            # the prior for the next sample is the posterior from the current sample
            prior_distributions[i] = posterior
        end
    end
end;
```

```
function handle_coin_toss(prior::Beta, observation::Bool)
    posterior = Beta(prior.α + observation, prior.β + (1 - observation))
    return posterior
end;
```

```
function log_evidence_prior(prior::Beta, N::Int64, n::Int64)::Real
    log10gamma(prior.α + prior.β) -
    log10gamma(prior.α) -
    log10gamma(prior.β) +
    log10gamma(n+prior.α) +
    log10gamma((N-n)+prior.β) -
    log10gamma(N+prior.α+prior.β)
end;
```

```
log10gamma (generic function with 1 method)
```

```
# computes log10 of Gamma function
function log10gamma(num::Real)::Real
    num = convert(BigInt, num)
    return log10(gamma(num))
end
```

Coin toss sample controls

```
md"""
### Coin toss sample controls
"""
```

```
N_tosses_bond = @bind N_tosses Slider([1:50..., 100:50:500...]; default=7,
show_value=true);
```

```
tosses = [false, true, false, false, false, false, true]
```

```
tosses = precomputed_tosses[1:N_tosses]
```

```
tosses_latex = @mdx """

```math
D=\\{$(Int.(tosses))\\}\\,.
```

"""
```

```
# a simpler (less pretty) display that can automatically wrap when the line gets
too long
tosses_html = """
D = {$(join(Int.(tosses), " "))}.

""" |> HTML;
```

```
D_sample_controls = PlutoUI.ExperimentalLayout.Div(
    [
        @htl("<h4 style='margin-bottom: .4em;'>💣 Generate a sample</h4>"),
    PlutoUI.ExperimentalLayout.Div(
        [
            PlutoUI.ExperimentalLayout.Div(
            [
                N_tosses_bond,

            ];
                style="""
                flex: 0 0 auto;
                """
            ),
            PlutoUI.ExperimentalLayout.Div(
                [
                    N_tosses >= 50 ? tosses_html : tosses_latex
                ];
                style="""
                    display: flex;
                    width: 300px;
                    height: 50px;
                    overflow: hidden;
                    /* Make the font-size smaller when the number of tosses
                    increases, to make sure everything is visible. */
                    font-size: $(1.2 * 300 / clamp(N_tosses, 25, 100))px;
                    """)

        ];
        style="""
        display: flex;
        flex-direction: row;
        gap: 1em;
        align-items: center;
        justify-content: space-evenly;
        """
    )
    ];
    style="""
        padding: 1em;
        background: #efbaab33;
        border-radius: 1em;


    """
);
```

Bayesian Linear regression code

```
md"""
### Bayesian Linear regression code
"""
```

```
const Layout = PlutoUI.ExperimentalLayout
```

```
const Layout = PlutoUI.ExperimentalLayout
```

```
const deterministic_randomness = MersenneTwister
```

```
const deterministic_randomness = MersenneTwister
```

```
σ_data_noise² = 0.0144
```

```
σ_data_noise² = σ_data_noise^2
```

```
σ_w_prior² = 0.25
```

```
σ_w_prior² = σ_w_prior^2
```

```
const μ_basis = range(0.0, 1.0; length=10);
```

```
const σ_basis² = 0.01;
```

```
D =
▸ [(0.96079, -0.0580549), (0.54779, -0.408251), (0.222016, 1.13581), (0.984644, 0.0134769),
```

```
D = let
    xs = rand(deterministic_randomness(19), Uniform(0,1), N)

    ys_exact = secret_function.(xs)

    rng = deterministic_randomness(37)
    ys = [
        rand(rng, Normal(y, sqrt(σ_data_noise²)))
        for y in ys_exact
     ]

    collect(zip(xs, ys))
end
```

```
baseplot (generic function with 1 method)
```

```
baseplot(args...; kwargs...) = plot(args...; size=(650,400), xlim=(-0.0, 1.0), ylim=
(-1.2,1.2), kwargs...)
```

```
plot_data! (generic function with 1 method)
```

```
function plot_data!(D)
    plot!(; legend=:bottomleft)
    plot!(secret_function;
          label="True function",
          color=3,
          lw=3,
            linestyle=:dash,
         )
    scatter!(
        D;
        label="Observations",
        color=1,
        # markerstrokewidth=0,
    )
end
```

```
ϕ(μ, x) = exp(-(x - μ)^2 / σ_basis²);
```

```
function f(w, x)
    sum(enumerate(μ_basis)) do (i, μ)
        w[i] * ϕ(μ, x)
    end
end;
```

```
# This is called the "design matrix"
Φ = [
    ϕ(μ, datum[1])
    for datum in D, μ in μ_basis
];
```

```
weights_posterior = MvNormalCanon(
    # Posterior potential vector
    Φ' * last.(D) / σ_data_noise²,
    # Posterior precision matrix (inverse covariance)
    Φ' * Φ / σ_data_noise² + I / σ_w_prior²
);
```

Discriminative classification

```
generate_dataset (generic function with 1 method)
```

```
X, y = generate_dataset(N2); # Generate data set, collect in matrix X and vector y
```

```
X_c1 = 72×2 adjoint(::Matrix{Float64}) with eltype Float64:
 2.37585    3.34038
 0.859544   4.75254
 0.931862   4.07941
 1.3216     5.8365
 0.018782   3.679
 0.0323811  2.75244
 0.985239   2.89621
 ⋮
 1.12348    3.87913
 0.342695   2.65799
 0.666544   2.99033
 0.943866   3.06345
 1.86646    3.73893
 0.301837   3.20804
```

```
X_c1 = X[:,findall(.!y)]' # Split X based on class label
```

```
X_c2 = 48×2 adjoint(::Matrix{Float64}) with eltype Float64:
 6.28972   0.0219173
 3.07004   6.10065
 6.4579   -0.252939
 4.64578   6.44116
 3.07336   6.34726
 7.87204   6.42865
 5.11242  -0.109727
 ⋮
 3.37472   6.09446
 8.383     6.18726
 5.60987   0.657917
 4.68727   6.38955
 7.77538   6.30464
 7.62106   6.28921
```

```
X_c2 = X[:,findall(y)]'
```

```
X_test = [3.75; 1.0]; # Features of 'new' data point
```

```
plot_dataset (generic function with 1 method)
```

`bayesian_discrimination_boundary`

This function computes the posterior distribution over regression weights using the Laplace Approximation. We use `logσ` as a numerically stable alternative to `logistic`, and we avoid matrix inversions by computing the precision matrix of the posterior distribution instead of the covariance.

The math in this function corresponds to eq. B-4.143

`predictive_posterior`

Computes the predictive posterior eq. B-4.152 using the given approximation to the sigmoid function.

```
logσ (generic function with 1 method)
```

```
log_likelihood (generic function with 1 method)
```

Variational Inference

Table of Contents

Variational Inference

The Variational Free Energy Functional

Consider a model $p(x, z) = p(x|z)p(z)$, where x and z are observed and latent variables, respectively. z may include parameters but also observation-dependent latent variables.

The goal of Bayesian inference is to transform the (known) *likelihood-times-prior* factorization of the full model to a *posterior-times-evidence* decomposition:

$$\underbrace{p(x|z)p(z)}_{\text{what we know}} \rightarrow \underbrace{p(z|x)p(x)}_{\text{what we want}}$$

Remember from the Bayesian machine learning lesson that negative log-evidence ("surprisal") can be decomposed as "complexity" minus "accuracy" terms (the CA decomposition):

$$\underbrace{-\log p(x)}_{\text{surprisal}} = \underbrace{\int p(z|x) \log \frac{p(z|x)}{p(z)} \mathrm{d}z}_{\text{complexity}} - \underbrace{\int p(z|x) \log p(x|z) \mathrm{d}z}_{\text{accuracy}}$$

The CA decomposition cannot be evaluated because it depends on the posterior $p(z|x)$, which cannot be evaluated since it is the objective of the inference process.

Let's now introduce a distribution $q(z)$ (called the "variational" or "approximate" posterior distribution) that we will use to *approximate* the posterior $p(z|x)$. Since we propose the distribution $q(z)$ ourselves, we will assume that $q(z)$ can be evaluated.

If we substitute $q(z)$ for $p(z|x)$ in the CA decomposition, then we obtain

$$F[q] \triangleq \underbrace{\int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z}_{\text{complexity}} - \underbrace{\int q(z) \log p(x|z) \mathrm{d}z}_{\text{accuracy}}$$

This expression is called the **Variational Free Energy** (VFE), represented by the symbol F. We treat F as a function of the posterior $q(z)$. Technically, a function of a function is called a functional, and we write square brackets (e.g., $F[q]$) to differentiate functionals from functions (e.g., $q(z)$).

Note that all factors in the CA decomposition of VFE (i.e., $q(z)$, $p(z)$, and $p(x|z)$) can be evaluated as a function of z (and x is observed), and therefore the VFE can be evaluated. This is important: log-evidence $\log p(x)$ cannot be evaluated, but $F[q]$ *can* be evaluated!

The Global VFE Minimum Recovers Bayes Rule

It turns out that we can perform (approximate) Bayesian inference by minimizing the Variational Free Energy (VFE) with respect to the variational distribution q.

To explain inference by VFE minimization (abbreviated as VFEM), we first rewrite the VFE in terms of "inference bound" minus "log-evidence" terms (the bound-evidence (BE) decomposition):

$$\begin{aligned} F[q] &= \underbrace{\int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z}_{\text{complexity}} - \underbrace{\int q(z) \log p(x|z) \mathrm{d}z}_{\text{accuracy}} \\ &= \underbrace{\int q(z) \log \frac{q(z)}{p(z|x)} \mathrm{d}z}_{\text{inference bound} \geq 0} - \underbrace{\log p(x)}_{\text{log-evidence}} \end{aligned}$$

▼ **Click to see proof**

$$\begin{aligned} F[q] &= \underbrace{\int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z}_{\text{complexity}} - \underbrace{\int q(z) \log p(x|z) \mathrm{d}z}_{\text{accuracy}} \\ &= \int q(z) \log \frac{q(z)}{p(x|z)p(z)} \mathrm{d}z \\ &= \int q(z) \log \frac{q(z)}{p(z|x)p(x)} \mathrm{d}z \quad (\text{ since } p(x|z)p(z) = p(z|x)p(x) \text{ }) \\ &= \underbrace{\int q(z) \log \frac{q(z)}{p(z|x)} \mathrm{d}z}_{\text{inference bound} \geq 0} - \underbrace{\log p(x)}_{\text{log-evidence}} \end{aligned}$$

Note that the inference bound is a Kullback-Leibler (KL) divergence (see also mini- notebook) between an (approximate) posterior $q(z)$ and the (perfect) Bayesian posterior $p(z|x)$. To learn more:

Since the second term (log-evidence) does not involve $q(z)$, VFEM over q will bring $q(z)$ closer to the Bayesian posterior $p(z|x)$.

Since $D_{\mathrm{KL}}[q(z), p(z|x)] \geq 0$ for any $q(z)$, and $D_{\mathrm{KL}}[q(z), p(z|x)] = 0$ only if $q(z) = p(z|x)$, the VFE is always an upper-bound on (minus) log-evidence, i.e.,

$$F[q] \geq -\log p(x)\,.$$

As a result, global minimization of VFE leads to

$$q^*(z) = \arg\min_q F[q]$$

where

$$\begin{aligned} q^*(z) &= p(z|x) && \text{(posterior)} \\ F[q^*] &= -\log p(x) && \text{(evidence)} \end{aligned}$$

Approximate Bayesian Inference by VFE Minimization

In practice, even if we cannot attain the global minimum of VFE, we can still use a local minimum,

$$\hat{q}(z) \approx \arg\min_q F[q]$$

to accomplish **approximate Bayesian inference** by:

$$\begin{aligned} \hat{q}(z) &\approx p(z|x) \\ F[\hat{q}] &\approx -\log p(x) \end{aligned}$$

Executing inference by minimizing the VFE functional is called **Variational Bayes** (VB) or **Variational Inference** (VI).

🎯 Key concept

For a model $p(x, z) = p(x|z)p(x)$, the **variational free energy** functional

$$F[q] = \underbrace{\int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z}_{\text{complexity}} - \underbrace{\int q(z) \log p(x|z) \mathrm{d}z}_{\text{accuracy}}$$

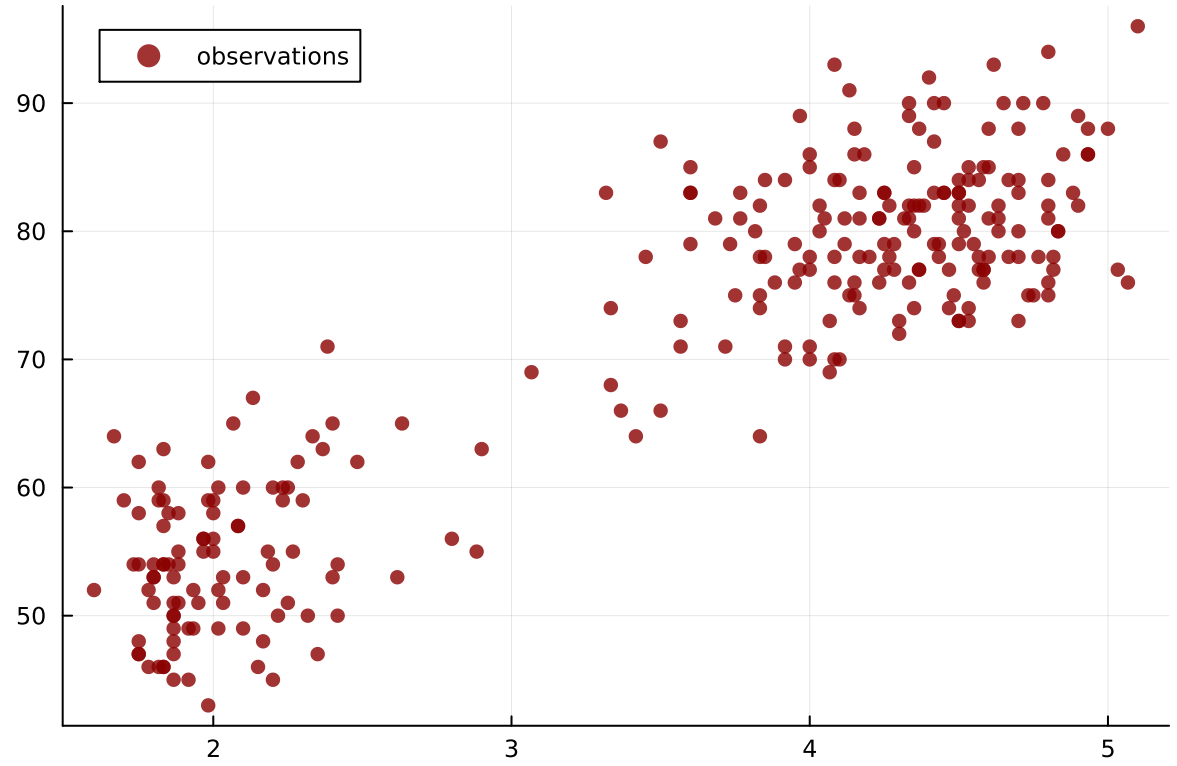
is considered a functional of a variational posterior $q(z)$ over the latent variables z. Minimizing $F[q]$ accomplishes **approximate Bayesian inference** by

$$\hat{q}(z) \approx p(z|x)$$
$$F[\hat{q}] \approx -\log p(x)$$

VFE minimization transforms a Bayesian inference problem (that involves integration) into an optimization problem! Generally, optimization problems are easier to solve than integration problems.

Code Example: VFEM for GMM on Old Faithfull data set

You are challenged to develop a model for the Old Faithful data set.



The Gaussian Mixture Model

The distribution of the observed data $\{x_n\}_{n=1}^N$ looks like it could be modeled by two Gaussians. Let's develop a model for this data set.

Likelihood

We associate a one-hot coded hidden class label z_n with each observation x_n

$$z_{nk} = \begin{cases} 1 & \text{if } x_n \in \mathcal{C}_k \text{ (the } k\text{-th cluster)} \\ 0 & \text{otherwise} \end{cases}$$

For a given class label z_n, let the observations be distributed as a Gaussian, i.e.,

$$p(x_n|z_{nk}=1) = \mathcal{N}\left(x_n|\mu_k, \Sigma_k\right)$$

which, due to the one-hot coding scheme for z_n, leads to

$$p(x_n|z_n) = \prod_{k=1}^{K} \mathcal{N}(x_n|\mu_k, \Sigma_k)^{z_{nk}} \tag{1}$$

Prior on the Clusters

Let the prior on the cluster labels $\{z_n\}_{n=1}^K$ be given by a **Categorical** distribution, i.e.,

$$p(z_n) = \prod_{k=1}^{K} \pi_k^{z_{nk}} \tag{2}$$

> **Key concept**
>
> A **Gaussian Mixture Model** is specified as
>
> $$\begin{aligned} p(x_n|z_{nk}=1) &= \mathcal{N}\left(x_n|\mu_k, \Sigma_k\right) \\ p(z_{nk}=1) &= \pi_k \end{aligned}$$
>
> where the class labels z_{nk} (and parameters) are *unobserved* in the data set.

Prior on the Parameters

We choose the following priors for the parameters:

$$p(\pi) = \mathrm{Dir}(\pi|\alpha_0) = C(\alpha_0) \prod_k \pi_k^{\alpha_0 - 1} \tag{3}$$

$$p(\mu|\Lambda) = \prod_k \mathcal{N}\left(\mu_k|m_0, (\beta_0 \Lambda_k)^{-1}\right) \tag{4}$$

$$p(\Lambda) = \prod_k \mathcal{W}\left(\Lambda_k|W_0, \nu_0\right) \tag{5}$$

where $\mathcal{W}(\cdot)$ is a Wishart distribution (i.e., a multi-dimensional Gamma distribution).

The full generative model is now specified by

$$p(x, z, \pi, \mu, \Lambda) = \underbrace{p(\pi)}_{(3)}\underbrace{p(\mu|\Lambda)}_{(4)}\underbrace{p(\Lambda)}_{(5)} \prod_n \underbrace{p(x_n|z_n, \mu, \Lambda)}_{(1)}\underbrace{p(z_n|\pi)}_{(2)}$$

with hyperparameters $\{\alpha_0, m_0, \beta_0, W_0, \nu_0\}$.

Variational Inference

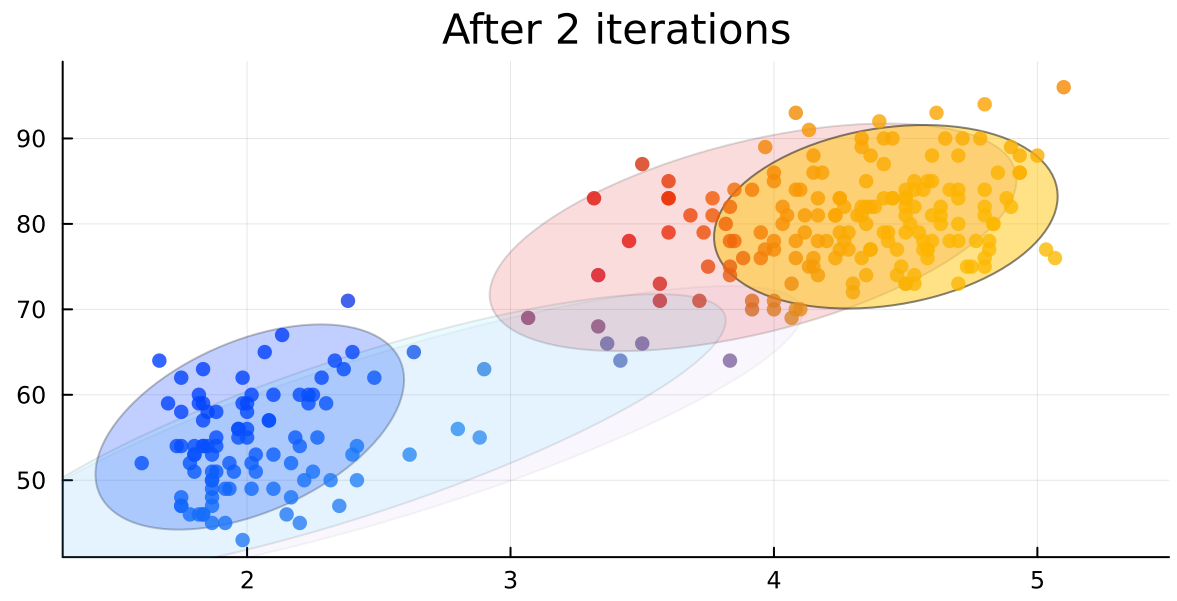
Assume that we have observed $D = \{x_1, x_2, \ldots, x_N\}$ and are interested in inferring a posterior distribution for the cluster labels z_n and the parameters π, μ, and Λ.

We will approximate Bayesian inference by VFE minimization. For the specified model, this leads to VFE minimization with respect to the hyperparameters, i.e., we need to minimize the function

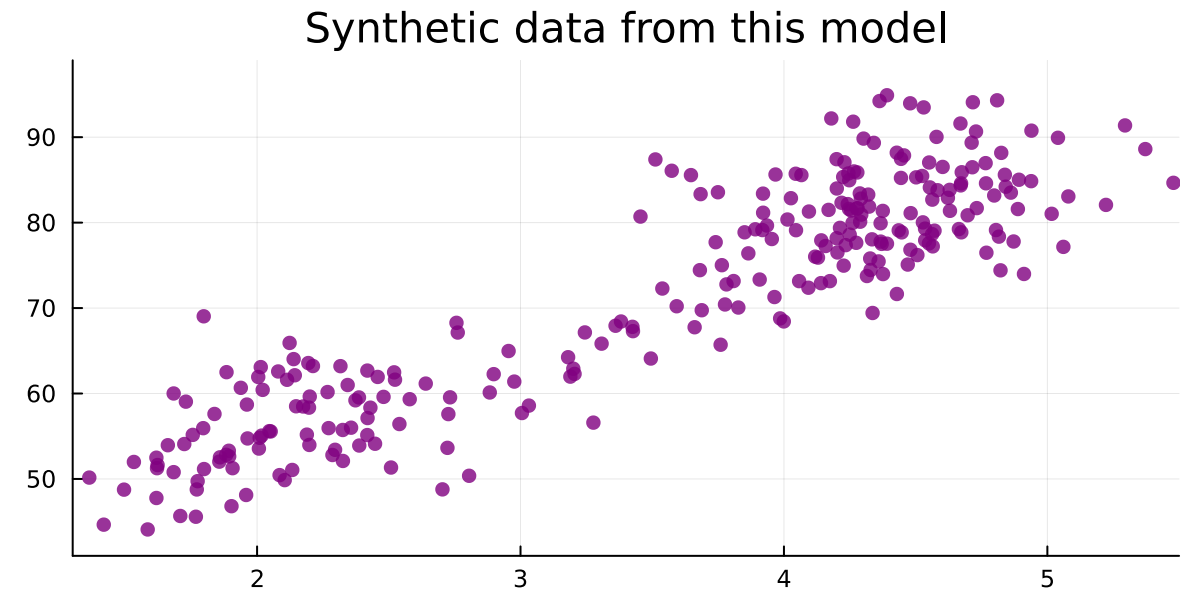
$$F(\alpha_0, m_0, \beta_0, W_0, \nu_0)\,.$$

Below we exemplify training of a Gaussian Mixture Model on the Old Faithful data set by VFE minimization.

`iteration_vfem =` 2



The generated figure resembles Figure 10.6 in Bishop. The plots show VFEM results for a GMM of $K = 6$ Gaussians applied to the Old Faithful data set. The ellipses denote the two standard-deviation density contours for each of the components, and the color coding of the data points reflects the "soft" class label assignments.



Show synthetic label colors: ☐

Theoretical Underpinnings of Variational Inference

Variational Inference and The Maximum Entropy Principle

In Caticha (2010) (based on earlier work by Shore and Johnson (1980)), the Principle of Maximum (Relative) Entropy is developed as a method for rational updating of priors to posteriors when faced with new information in the form of constraints.

Caticha's argumentation is as follows:

- Consider prior beliefs (i.e., a generative model) $p(x, z)$ about observed and latent variables x and z. Assume that new information in the form of constraints is obtained, and we are interested in the "best update" to posterior beliefs $q(x, z)$.
- In order to define what "best update" means, Caticha assumed a ranking function $S[q]$ that generates a preference score for each candidate posterior q for a given prior p. The best update from p to q is then identified as

$$q^* = \arg\max_q S[q] \,, \quad \text{subject to all constraints.}$$

Similarly to Cox' method for deriving Probability Theory from a set of sensical axioms, Caticha then introduced the following axioms, based on a rational principle (the **principle of minimal updating**, see Caticha 2010), that the ranking function needs to adhere to:

1. *Locality*: local information has local effects.
2. *Coordinate invariance*: the system of coordinates carries no information.
3. *Independence*: When systems are known to be independent, it should not matter whether they are treated separately or jointly.

It turns out that these three criteria **uniquely identify the Relative Entropy** as the proper ranking function:

$$S[q] = -\sum_z q(x, z) \log \frac{q(x, z)}{p(x, z)}$$

This procedure for finding the variational posterior q is called the **Principle of Maximum (Relative) Entropy** (PME). Note that, since $S[q] = -F[q]$, constrained Relative Entropy maximization is equivalent to constrained VFE minimization!

Therefore, when information is supplied in the form of constraints on the posterior (such as form/factorization constraints and new observations as data constraints), we *should* select the posterior that minimizes the constrained Variational Free Energy. **Constrained FE minimization is the proper method for inference!**

Bayes rule is the global solution of constrained VFEM when all constraints are data constraints, ie, delta distributions on $q(x)$. Hence, Bayes rule is a special case of constrained VFEM. Bayes rule only applies to updating beliefs on the basis of new observations.

Summary

Key concept ↑ Jump to source

For a model $p(x, z) = p(x|z)p(x)$, the **variational free energy** functional

$$F[q] = \underbrace{\int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z}_{\text{complexity}} - \underbrace{\int q(z) \log p(x|z) \mathrm{d}z}_{\text{accuracy}}$$

is considered a functional of a variational posterior $q(z)$ over the latent variables z. Minimizing $F[q]$ accomplishes **approximate Bayesian inference** by

$$\begin{aligned} \hat{q}(z) &\approx p(z|x) \\ F[\hat{q}] &\approx -\log p(x) \end{aligned}$$

VFE minimization transforms a Bayesian inference problem (that involves integration) into an optimization problem! Generally, optimization problems are easier to solve than integration problems.

Key concept ↑ Jump to source

A **Gaussian Mixture Model** is specified as

$$\begin{aligned} p(x_n|z_{nk} = 1) &= \mathcal{N}\left(x_n|\mu_k, \Sigma_k\right) \\ p(z_{nk} = 1) &= \pi_k \end{aligned}$$

where the class labels z_{nk} (and parameters) are *unobserved* in the data set.

← Previous lecture

Next lecture →

Code

```
using BmlipTeachingTools
```

```
using LinearAlgebra, PDMats, SpecialFunctions, Random
```

```
using Distributions, Plots, StatsPlots
```

```
X =
2×272 Matrix{Float64}:
  3.6   1.8   3.333   2.283   4.533   2.883  …   4.117   2.15   4.417   1.817   4.467
 79.0  54.0  74.0    62.0    85.0    55.0       81.0    46.0   90.0    46.0    74.0
```

```
N = size(X, 2);
```

```
K = 6;
```

```
max_iterations = 120;
```

```
vfem_result =
▸ (clusters_vb = 6×120 Matrix{Distribution}:                    , R = 6×272×120 Array{Float64, 3}:
                 FullNormal(                                          [:, :, 1] =
```

```
sufficientStatistics (generic function with 1 method)
```

```
updateMeanPrecisionPi (generic function with 1 method)
```

```
updateR (generic function with 1 method)
```

```
plotGMM (generic function with 1 method)
```

```
const plot_lims = ▸ (xlim = (1.3, 5.5), ylim = (41, 99))
```

```
const cluster_colors = color_list(:seaborn_bright6);
```

```
get_color (generic function with 1 method)
```

```
const data_plot_kwargs =
▸ (markersize = 4, markerstrokewidth = 0, color = :red4, opacity = 0.8)
```

```
old_faithful = 272×2 Matrix{Float64}:
 3.6    79.0
 1.8    54.0
 3.333  74.0
 2.283  62.0
 4.533  85.0
 2.883  55.0
 4.7    88.0
 ⋮
 4.75   75.0
 4.117  81.0
 2.15   46.0
 4.417  90.0
 1.817  46.0
 4.467  74.0
```

Intelligent Agents and Active Inference

Table of Contents

Materials

- Noumenal labs (2025), [WTF is the FEP? A short explainer on the free energy principle](https://www.noumenal.ai/post/wtf-is-the-fep-a-short-explainer-on-the-free-energy-principle
    - A concise, accessible introduction to the Free Energy Principle, aimed at demystifying it for a broader audience—matching the tone and intent suggested by the playful but

clear title.

- De Vries et al. (2025), Expected Free Energy-based Planning as Variational Inference
  - On minimizing expected free energy by variational free energy minimization.
- Friston et al. (2023), Path integrals, particular kinds, and strange things
  - The most recent formal developments of the Free Energy Principle (FEP). This paper frames FEP as a principle of least action over trajectories.
- Bert de Vries, Tim Scarfe, and Keith Duggar (2023), Podcast on Active Inference. Machine Learning Street Talk podcast
  - Quite extensive discussion on many aspects regarding the Free Energy Principle and Active Inference, in particular relating to its implementation.
- Friston et al. (2022), Designing Ecosystems of Intelligence from First Principles
  - Friston's vision on the future of AI.
- Bert de Vries (2021), Presentation on Beyond deep learning: natural AI systems (video)
  - 30-minute introduction to active inference from an engineering point of view.
- Raviv (2018), The Genius Neuroscientist Who Might Hold the Key to True AI.
  - Interesting article on Karl Friston, who is a leading theoretical neuroscientist working on a theory that relates life and intelligent behavior to physics (and Free Energy minimization). (**highly recommended**)
- Karl Friston (2011), What Is Optimal about Motor Control?, Neuron 72-3, p488-498
  - This work critiques classical optimal control theory for being an insufficient model of biological motor control, and instead advocates active inference as a more suitable framework.

Agents

In the previous lessons, we assumed that a data set was given.

In this lesson, we consider *agents*. An agent is a system that *interacts* with its environment through both sensors and actuators.

Crucially, by acting on the environment, the agent is able to affect the data that it will sense in the future.

- As an example, by changing the direction where I look, I can affect the (visual) data that will be sensed by my retina.

With this definition of an agent, (biological) organisms are agents, and so are robots, self-driving cars, etc.

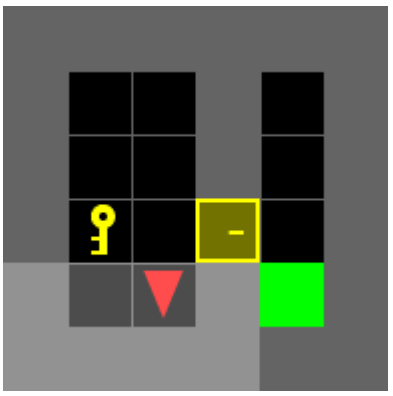
In an engineering context, we are particularly interested in agents that behave with a *purpose*, that is, with a specific goal in mind, such as driving a car or trading in financial markets.

In this lesson, we will describe how **goal-directed behavior** by biological (and synthetic) agents can also be interpreted as the minimization of a free energy functional.

Challenge: The Door-Key MiniGrid Problem

Problem

In this example, we consider the Door-Key MiniGrid problem, which is part of the challenging MiniGrid environment family. In these environments, an agent is positioned in a gridworld and has to solve a particular task. The red triangle indicates the agent's location and viewing direction, while the green square marks the target location.



At each timestep, the agent observes the portion of the environment contained within the shaded rectangle in front of its viewing direction. The agent's task is to find the key, use it to open the door (yellow square in the third column), and then navigate to the target square.

We assume that the agent has complete knowledge of the environmental process dynamics (e.g., how to walk or which action corresponds to picking up a key). However, the locations of the key and door are randomized and therefore unknown.

The challenge is to design an agent that autonomously navigates to the target square. The agent should be defined as a probabilistic model, with its control signals obtained by casting action selection as a Bayesian inference problem.

Solution

At the end of this lesson.

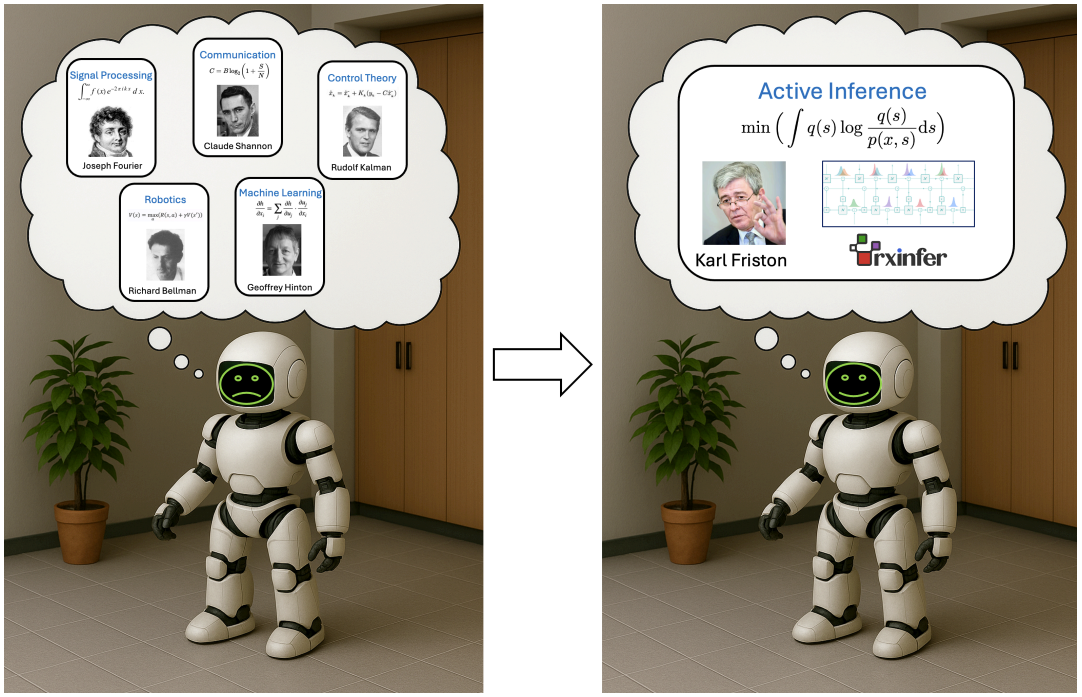
The Free Energy Principle

Motivation

The human brain is the most complex control system we know. It processes inputs from about 10 million sensory neurons through roughly 10^{11} ($= 100$ billion) neurons, and drives action via about half a million motor neurons. None of these neurons "know" anything about Fourier transforms, dynamic programming, or backpropagation, they simply minimize variational free energy (VFE).

Remarkably, this autonomous process is sufficient to create a control system that outperforms anything we have ever engineered with our hand-crafted theories of control and signal processing. As engineers, this motivates our work on active inference agents: if nature can produce such superior systems solely through autonomous VFE minimization, perhaps this is also the right path forward in engineering. To achieve truly high-performant adaptive control in volatile environments, it may be necessary to let go of hand-crafted algorithms and build systems that function solely by VFE minimization.

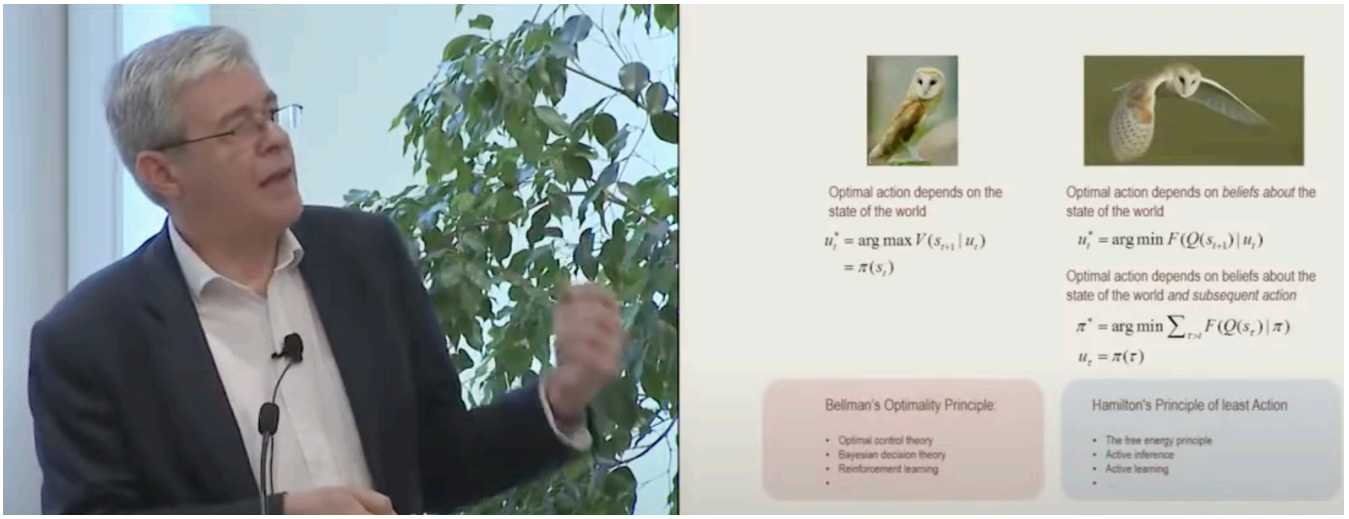
This lecture explores that idea.



What Drives Intelligent Behavior?

We begin with an example that requires "intelligent" decision-making. Assume that you are an owl and that you're hungry. What are you going to do?

Have a look at Prof. Karl Friston's answer in this video segment on the cost function for intelligent behavior. (**Do watch the video!**)



In his answer, Friston emphasizes that the first step is to search for food, for instance, a mouse. You cannot eat the mouse unless you know where it is, so the first imperative is to reduce your uncertainty about the location of the mouse. In other words, purposeful behavior begins with epistemic behavior: searching to resolve uncertainty.

This stands in contrast to more traditional approaches to intelligent behavior, such as reinforcement learning, where the objective is to maximize a value function of future states, e.g., $V(s)$, where s might encode how hungry the agent is. However, this paradigm falls short in scenarios where the optimal next action is to gather information, because uncertainty is not an attribute of states themselves, but of *beliefs* over states, which are expressed as probability distributions.

Therefore, Friston argues that intelligent behavior requires us to optimize a functional $F[q(s|u)]$, where $q(s|u)$ is a probability distribution over (future) states s for a given action sequence u, and F evaluates the quality of this belief.

Later in his lectures and papers, Friston expands on this belief-based objective F and formalizes it as a variational free energy functional—laying the foundation for the **Free Energy Principle**. This principle offers a unifying framework that connects biological (or "intelligent") decision-making and behavior directly to Bayesian inference.

> 🎯 **Key concept**
>
> Friston's key insight is that intelligent behavior necessarily involves uncertainty-reducing behavior, which should be framed as the optimization of a functional of beliefs (i.e., probabilities) over future states.

The Free Energy Principle

The Free Energy Principle (FEP) is neither a model nor a theory. Rather, it is a principle, that is, a **methodological framework** for describing the information-processing dynamics that *must* unfold in living systems to keep them within viable (i.e., livable) states over extended periods of time.

- Think of the processes continuously occurring in our bodies to maintain an internal temperature between approximately $36°C$ and $37°C$, regardless of the surrounding ambient temperature.

The literature on the FEP is widely regarded as difficult to access. It was first formally derived as a specific case of the Least Action Principle by Friston in his monograph Friston (2019), A Free Energy Principle for a Particular Physics (2019), and more recently presented in a more accessible form in Friston et al. (2023), Path integrals, particular kinds, and strange things. For a concise and approachable introduction, the explainer by Noumenal Labs (2025), WTF is the FEP? is currently the most accessible resource I am aware of.

In this lecture, we present only a simplified account. According to the FEP, the brain is a generative model for its sensory inputs, such as visual and auditory signals, and **continuously minimizes variational free energy** (VFE) in that model to stay aligned with these observations. Crucially, VFE minimization is the *only* ongoing process, and it underlies perception, learning, attention, emotions, consciousness, intelligent decision-making, etc.

To illustrate the idea that perception arises from a (variational) inference process—driven by top-down predictions from the brain and corrected by bottom-up sensory inputs—consider the following figure: "The Gardener" by Giuseppe Arcimboldo (ca. 1590).

On the left, you'll likely perceive a bowl of vegetables. However, when the same image is turned upside down, most people first see a gardener's face.

This perceptual flip arises because the brain's generative model assigns a much higher probability to being in an environment with upright human faces than with inverted bowls of vegetables. While the sensory input is consistent with both interpretations, the brain's prior beliefs drive our perception toward seeing upright faces (and upright bowls of vegetables).

In short, the FEP characterizes "intelligent" behavior as the outcome of a VFE minimization process. Next, we derive the dynamics of an *Active Inference* agent—an agent whose behavior is entirely governed by VFE minimization. We will demonstrate that minimizing VFE within a generative model constitutes a sufficient mechanism for producing basic intelligent behavior.

🎯 Key concept

The **Free Energy Principle** (FEP) can be seen as a specific instance of the Least Action Principle applied to biological systems, which are technically systems that act to preserve their functional and structural integrity.

The Expected Free Energy Theorem

Setup of Prior Beliefs

Let's make the above notions more concrete. We consider an agent that interacts with its environment. At the current time t, the agent holds a generative model to predict its future observations,

$$p(y, x, u, \theta)\,, \tag{P1}$$

where y denotes future observations, x refers to internal (hidden) future states, u represents the agent's future actions, and θ are model parameters.

Since model (P1) is designed to predict how the future is expected to unfold, we refer to (P1) as the **predictive model**. A typical example is a rollout to the future of a state-space model,

$$p(y, x, u, \theta) = p(x_t)p(\theta) \underbrace{\prod_{k=t+1}^{T} p(y_k|x_k, \theta)p(x_k|x_{k-1}, u_k)p(u_k)}_{\text{rollout to the future}}\,.$$

In addition to the predictive model, we assume that the agent holds beliefs $\hat{p}(x)$ about the *desired* future states. For example, the owl in our earlier example holds the belief that it will not be hungry in the future. We refer to $\hat{p}(x)$ as the **goal prior**.

Finally, we assume that the agent also maintains **epistemic** (= information-seeking) prior beliefs, denoted by $\tilde{p}(u)$, $\tilde{p}(x)$, and $\tilde{p}(y, x)$, which will be further specified below.

The predictive model, together with the goal and epistemic priors, constitutes the agent's complete set of prior beliefs about the future.

The Expected Free Energy Theorem

We now state the Expected Free Energy theorem. Let the variational free energy functional $F[q]$ be defined as

$$F[q] = \mathbb{E}_{q(y,x,u,\theta)}\left[\log \frac{q(y,x,u,\theta)}{\underbrace{p(y,x,u,\theta)}_{\text{predictive}}\underbrace{\hat{p}(x)}_{\text{goal}}\underbrace{\tilde{p}(u)\tilde{p}(x)\tilde{p}(y,x)}_{\text{epistemics}}}\right]. \tag{F1}$$

Let the agent's epistemic priors be defined as

$$\tilde{p}(u) = \exp\left(H[q(x|u)]\right) \tag{E1}$$
$$\tilde{p}(x) = \exp\left(-H[q(y|x)]\right) \tag{E2}$$
$$\tilde{p}(y,x) = \exp\left(D[q(\theta|y,x), q(\theta|x)]\right) \tag{E3}$$

where $H[q] = \mathbb{E}_q\left[-\log q\right]$ is the entropy functional, and $D[q,p] = \mathbb{E}_q\left[\log q - \log p\right]$ is the Kullback–Leibler divergence.

Then, the variational free energy $F[q]$ decomposes as

$$F[q] = \underbrace{\mathbb{E}_{q(u)}\left[G(u)\right]}_{\substack{\text{expected policy}\\ \text{costs}}} + \underbrace{\mathbb{E}_{q(y,x,u,\theta)}\left[\log \frac{q(y,x,u,\theta)}{p(y,x,u,\theta)}\right]}_{\text{complexity}}, \tag{F2}$$

where the function $G(u)$, known as the **Expected Free Energy** (EFE) cost function, is given by

$$G(u) = \underbrace{\underbrace{\mathbb{E}_q\left[\log \frac{q(x|u)}{\hat{p}(x)}\right]}_{\text{risk}}}_{\text{scores goal-driven behavior}} + \underbrace{\underbrace{\mathbb{E}_q\left[\log \frac{1}{q(y|x)}\right]}_{\text{ambiguity}} - \underbrace{\mathbb{E}_q\left[\log \frac{q(\theta|y,x)}{q(\theta|x)}\right]}_{\text{novelty}}}_{\text{scores information-seeking behavior}}. \tag{G1}$$

▼ **Click for proof of the EFE Theorem**

For the following proof, see also Appendix A in De Vries et.al., Expected Free Energy-based Planning as Variational Inference (2025).

$$
\begin{aligned}
F[q] &= E_{q(yxu\theta)}\left[\log \frac{q(yxu\theta)}{p(yxu\theta)\hat{p}(x)\tilde{p}(u)\tilde{p}(x)\tilde{p}(yx)}\right] \\
&= E_{q(u)}\left[\log \frac{q(u)}{p(u)} + \underbrace{E_{q(yx\theta|u)}\left[\log \frac{q(yx\theta|u)}{p(yx\theta|u)\hat{p}(x)\tilde{p}(u)\tilde{p}(x)\tilde{p}(yx)}\right]}_{B(u)}\right] \qquad \text{(C1)} \\
&= E_{q(u)}\left[\log \frac{q(u)}{p(u)} + \underbrace{G(u) + E_{q(yx\theta|u)}\left[\log \frac{q(yx\theta|u)}{p(yx\theta|u)}\right]}_{=B(u) \text{ if conditions (E1), (E2) and (E3) hold}}\right] \qquad \text{(C2)} \\
&= E_{q(u)}\left[G(u)\right] + E_{q(yxu\theta)}\left[\log \frac{q(yxu\theta)}{p(yxu\theta)}\right],
\end{aligned}
$$

if the conditions in Eqs. $(\mathbf{E1})$, $(\mathbf{E2})$, and $(\mathbf{E3})$ hold.

In the above derivation, we still need to prove the equivalence of $B(u)$ in Eqs. $(\mathbf{C1})$ and $(\mathbf{C2})$, which we address next. In the following, all expectations are with respect to $q(y, x, \theta|u)$ unless otherwise indicated.

$$
\begin{aligned}
B(u) &= E\left[\log \frac{\overbrace{q(yx\theta|u)}^{\text{posterior}}}{\underbrace{p(yx\theta|u)\hat{p}(x)\tilde{p}(u)\tilde{p}(x)\tilde{p}(yx)}_{\text{predictive}\quad\text{goals}\quad\text{epistemic priors}}}\right] \qquad \text{(C3)}\\
&= \underbrace{E\left[\log\left(\underbrace{\frac{q(x|u)}{\hat{p}(x)}}_{\text{risk}}\cdot\underbrace{\frac{1}{q(y|x)}}_{\text{ambiguity}}\cdot\underbrace{\frac{q(\theta|x)}{q(\theta|yx)}}_{-\text{novelty}}\right)\right]}_{G(u)=\text{Expected Free Energy}}+\\
&\quad +E\left[\log\left(\underbrace{\frac{\hat{p}(x)q(y|x)q(\theta|yx)}{q(x|u)q(\theta|x)}}_{\text{inverse factors from } G(u)}\cdot\underbrace{\frac{q(yx\theta|u)}{p(yx\theta|u)\hat{p}(x)\tilde{p}(u)\tilde{p}(x)\tilde{p}(yx)}}_{\text{leftover factors from (C3)}}\right)\right]\\
&= G(u)+\underbrace{E\left[\log\frac{q(yx\theta|u)}{p(yx\theta|u)}\right]}_{=C(u)}+\underbrace{E\left[\log\frac{q(y|x)q(\theta|yx)}{q(x|u)q(\theta|x)\tilde{p}(u)\tilde{p}(x)\tilde{p}(yx)}\right]}_{\text{choose epistemic priors to let this vanish}}\\
&= G(u)+C(u)+\\
&\quad +E\left[\log\frac{1}{q(x|u)\tilde{p}(u)}\right]+E\left[\log\frac{q(y|x)}{\tilde{p}(x)}\right]+E\left[\log\frac{q(\theta|yx)}{q(\theta|x)\tilde{p}(yx)}\right]\\
&= G(u)+C(u)+\\
&\quad +\sum_{y\theta}q(y\theta|x)\underbrace{\left(\underbrace{-\sum_x q(x|u)\log q(x|u)}_{=H[q(x|u)]}-\sum_x q(x|u)\log\tilde{p}(u)\right)}_{=0\text{ if }\tilde{p}(u)=\exp(H[q(x|u)])}\\
&\quad +\sum_x q(x|u)\underbrace{\left(\underbrace{\sum_y q(y|x)\log q(y|x)}_{=-H[q(y|x)]}-\sum_y q(y|x)\log\tilde{p}(x)\right)}_{=0\text{ if }\tilde{p}(x)=\exp(-H[q(y|x)])}\\
&\quad +\sum_{yx}q(yx|u)\underbrace{\left(\underbrace{\sum_\theta q(\theta|yx)\log\frac{q(\theta|yx)}{q(\theta|x)}}_{D[q(\theta|yx),q(\theta|x)]}-\sum_\theta q(\theta|yx)\log\tilde{p}(yx)\right)}_{=0\text{ if }\tilde{p}(yx)=\exp(D[q(\theta|yx),q(\theta|x)])}\\
&= G(u)+E_{q(yx\theta|u)}\left[\log\frac{q(yx\theta|u)}{p(yx\theta|u)}\right],
\end{aligned}
$$

if Eqs. (E1), (E2), and (E3) hold.

Interpretation of Expected Free Energy $G(u)$

$G(u)$ is a cost function defined over a sequence of future actions $u = (u_{t+1}, u_{t+2}, \ldots, u_T)$, commonly referred to as a **policy**. $G(u)$ decomposes into three distinct components:

risk

- The risk term is the KL divergence between $q(x|u)$, the *predicted* future states under policy u, and $\hat{p}(x)$, the *desired* future states (the goal prior). As a result, $G(u)$ penalizes policies that lead to expectations which diverge from the agent's preferences — that is, from what the agent wants to happen.

ambiguity

- Ambiguity can be expressed as $\mathbb{E}_{q(x|u)}\left[H[q(y|x)]\right]$, which quantifies the expected entropy of future observations y, under policy u. It measures how ambiguous or noisy the relationship is between hidden states x and observations y. Policies with low ambiguity are preferable because they lead to observations that are more informative about the hidden state, thus facilitating more accurate inference and better decision-making.
- ▼ **Click to show derivation of ambiguity as an expected entropy**

  Starting from Eq.(G1),

$$\begin{aligned}\mathbb{E}_{q(y,x|u)}\left[\log \frac{1}{q(y|x)}\right] &= \mathbb{E}_{q(x|u)}\left[\mathbb{E}_{q(y|x)}\left[\log \frac{1}{q(y|x)}\right]\right] \\ &= \mathbb{E}_{q(x|u)}\left[H[q(y|x)]\right]\end{aligned}$$

novelty

- The novelty term can be worked out to $\mathbb{E}_{q(x|u)}\left[I[\theta, y\,|x]\right]$, where $I[\theta, y\,|x]$ is the mutual information between parameters θ and observations y, given states x. Novelty complements the ambiguity term. While ambiguity scores information-seeking behavior aimed at reducing uncertainty about hidden states x, the novelty term extends this idea to parameters θ of the generative model. It encourages policies that are expected to lead to observations that reduce uncertainty about θ, i.e., learning about the structure or dynamics of the environment itself.
- ▼ **Click to show derivation of novelty in terms of mutual information**

  Starting from Eq.(G1),

$$
\begin{aligned}
\mathbb{E}_{q(y,x,\theta|u)}\left[\log \frac{q(\theta|y,x)}{q(\theta|x)}\right] &= \mathbb{E}_{q(y,\theta|x)q(x|u)}\left[\log \frac{q(\theta|y,x)}{q(\theta|x)}\right] \\
&= \mathbb{E}_{q(x|u)}\left[\mathbb{E}_{q(y,\theta|x)}\left[\log \frac{q(\theta|y,x)}{q(\theta|x)}\right]\right] \\
&= \mathbb{E}_{q(x|u)}\left[\underbrace{\mathbb{E}_{q(y,\theta|x)}\left[\log \frac{q(\theta,y|x)}{q(\theta|x)q(y|x)}\right]}_{I[\theta,y\,|x]}\right] \\
&= \mathbb{E}_{q(x|u)}\left[I[\theta,y\,|x]\right]
\end{aligned}
$$

Clearly, policies with lower Expected Free Energy are preferred. Such policies strike a balance between goal-directed behavior—by minimizing risk—and information-seeking behavior—by minimizing ambiguity (to infer hidden states) and maximizing novelty (to learn about model parameters). This unified objective naturally promotes both exploitation and exploration.

Active Inference

Optimal Planning by Variational Inference

Assume that our agent is continually engaged in minimizing its variational free energy $F[q]$, defined in Eq. (F2). This process tracks the following optimal posterior beliefs over policies,

$$\begin{aligned} q^*(u) &\triangleq \arg\min_q F[q] \\ &= \sigma\left(-G(u) - C(u) - P(u)\right), \end{aligned} \tag{Q*}$$

where

- $\sigma(\cdot)$ denotes the softmax function,
- $G(u)$ is the expected free energy, defined in Eq. (G1), scoring both goal-directed and epistemic value of each policy,
- $C(u) = \mathbb{E}_{q(y,x,\theta|u)}\left[\log \frac{q(y,x,\theta|u)}{p(y,x,\theta|u)}\right]$ is a complexity term, capturing divergence between the variational posterior and prior beliefs for a given policy u,
- $P(u) = -\log p(u)$ reflects prior preferences over policies from the generative model.

▼ **Click for proof of $q^*(u)$**

Starting from Eq. (F2),

$$F[q] = \mathbb{E}_{q(u)}\left[G(u)\right] + \mathbb{E}_{q(y,x,u,\theta)}\left[\log \frac{q(y,x,u,\theta)}{p(y,x,u,\theta)}\right] \tag{F2}$$

$$\begin{aligned} &= \mathbb{E}_{q(u)}\left[G(u) + \underbrace{\mathbb{E}_{q(y,x,\theta|u)}\left[\log \frac{q(y,x,\theta|u)}{p(y,x,\theta|u)}\right]}_{C(u)} + \log \frac{q(u)}{p(u)}\right] \\ &= \mathbb{E}_{q(u)}\left[\log \frac{1}{\exp(-G(u))} + \log \frac{1}{\exp(-C(u))} + \log \frac{q(u)}{\exp(-P(u))}\right] \\ &= \mathbb{E}_{q(u)}\left[\log \frac{q(u)}{\exp(-G(u) - C(u) - P(u))}\right] \end{aligned}$$

which is (proportional to) a Kullback-Leibler divergence that is minimized for

$$q^*(u) = \sigma\left(-G(u) - C(u) - P(u)\right).$$

An Active Inference Agent!

Eq. (Q*) marks a central result: an agent that minimizes the variational free energy $F[q]$, as defined in Eq.(F2), naturally selects policies that are goal-directed, epistemically valuable, and computationally parsimonious.

- Goal-directed policies **minimize risk** by steering predicted future states toward preferred or desired outcomes.
- Epistemically valuable policies reduce uncertainty by favoring informative observations (**low ambiguity**) and supporting model learning (**high novelty**).
- Computationally parsimonious policies **minimize complexity**, ensuring that posterior beliefs remain close to prior expectations. This limits the extent of belief updating, thereby *conserving computational resources* and reducing inference overhead.

The process of minimizing $F[q]$ is called an **Active Inference** (AIF) process, and an agent that realizes this process is referred to as an **active inference agent**. The “active” aspect highlights that an AIF agent does not passively consume a fixed data set, but instead actively selects its own data set through purposeful interaction with the environment.

From an engineering perspective, if one accepts that effective decision-making systems should exhibit goal-directed behavior, epistemic exploration, and computational efficiency, then an AIF agent can be viewed as an "intelligent" controller. Given a well-defined set of predictive, goal-oriented, and epistemic prior beliefs, the agent’s behavior follows directly from the minimization of variational free energy. In this sense, the agent acts rationally—or Bayes-optimally—with respect to its design objectives and internal model.

> 🎯 **Key concept**
>
> The process underlying naturally intelligent behavior is termed **Active Inference** (AIF). It can be described as the minimization of a variational free energy under a generative model that incorporates a rollout into the future.

Interpretation of the Epistemic Priors

In the formulation introduced in Eq. (E1), the epistemic prior $\tilde{p}(u) = \exp\left(H[q(x|u)]\right)$ biases the agent toward selecting policies u that maximize the entropy of the predicted future states x.

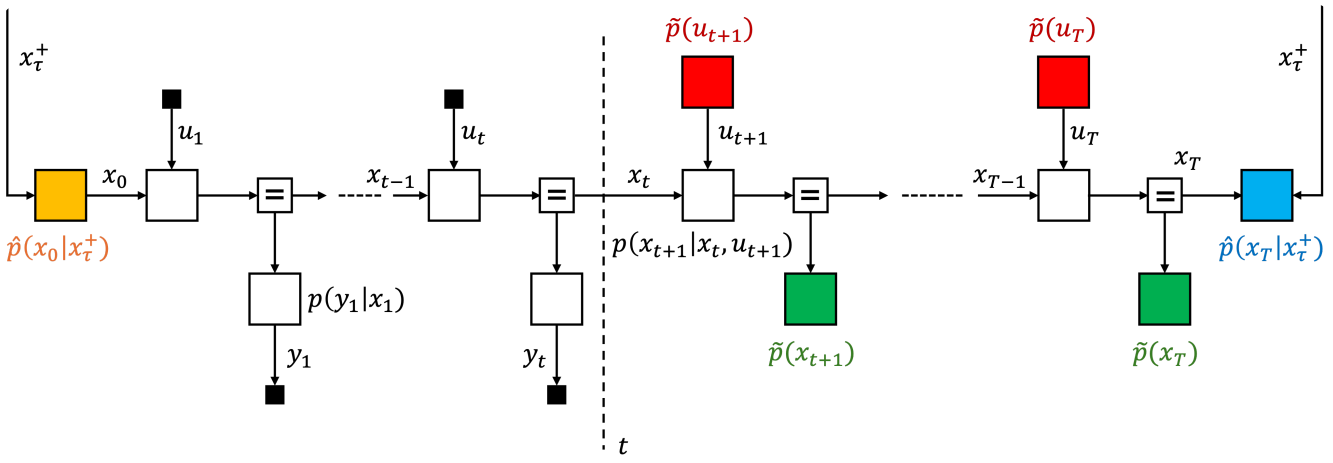
This reflects an information-seeking preference: high entropy over future states implies that the agent is actively maintaining flexibility and postponing premature commitment. Rather than treating uncertainty as something to avoid, this formulation encourages the agent to seek out policies that enable adaptation as new observations arrive.

Additionally, the epistemic prior $\tilde{p}(x) = \exp(-H[q(y|x)])$ in (E2), favors policies that reduce uncertainty about future states by selecting observations that are informative about them. Together, $\tilde{p}(u)$ and $\tilde{p}(x)$ induce a **bias toward ambiguity-minimizing behavior**.

Similarly, the epistemic priors $\tilde{p}(u)$ and $\tilde{p}(y, x)$ from (E1) and (E3), jointly shape a **preference for policies that maximize novelty**, i.e., that are expected to be informative about the parameters of the generative model.

Realization by Reactive Message Passing

An AIF agent can be efficiently realized by an autonomous reactive message passing process in a Forney-style Factor Graph (FFG) representation of (a rollout to the future of) the generative model, augmented with goal and epistemic priors.



In the above figure, the agent's generative (predictive) model

$$\prod_{k=1}^{T} p(y_k|x_k)p(x_k|x_{k-1},u_k)\,,$$

is represented by the white nodes in the factor graph. The initial and desired final states are constrained by initial and goal priors $\hat{p}(x_0|x^+)$ and $\hat{p}(x_T|x^+)$, which are typically generated by a higher-level state x^+ and shown here as orange and blue nodes, respectively.

At time $k = 0$, the agent is tasked to infer a future action sequence (a "policy") $u_{1:T}$ such that the posterior $q(x_T|y_{1:T})$ matches the goal prior $\hat{p}(x_T|x^+)$ as closely as possible. Inference proceeds entirely via reactive message passing in the factor graph, with no external control.

The figure shows the state of the system at time t, after having executed actions $u_{1:t}$ and having observed $y_{1:t}$. The future rollout for steps $t+1$ to T terminates the predictive model (white) with both epistemic priors (green and red nodes) and the goal prior (blue node). As new actions are selected and new observations are sensed, the epistemic priors are replaced by posteriors (small black boxes), enabling an ongoing free energy minimization process.

Challenge Revisited: The Door-Key MiniGrid Problem

We now return to the Door-Key MiniGrid problem from the beginning of this lesson. Below, we present RxInfer pseudocode for the augmented generative model of the MiniGrid agent. The key insight from this code fragment is that there is no explicit "algorithmic" planning code: in an AIF agent, all processes reduce to VFE minimization (of Eq. (F1)), which is carried out automatically by the RxInfer inference engine.

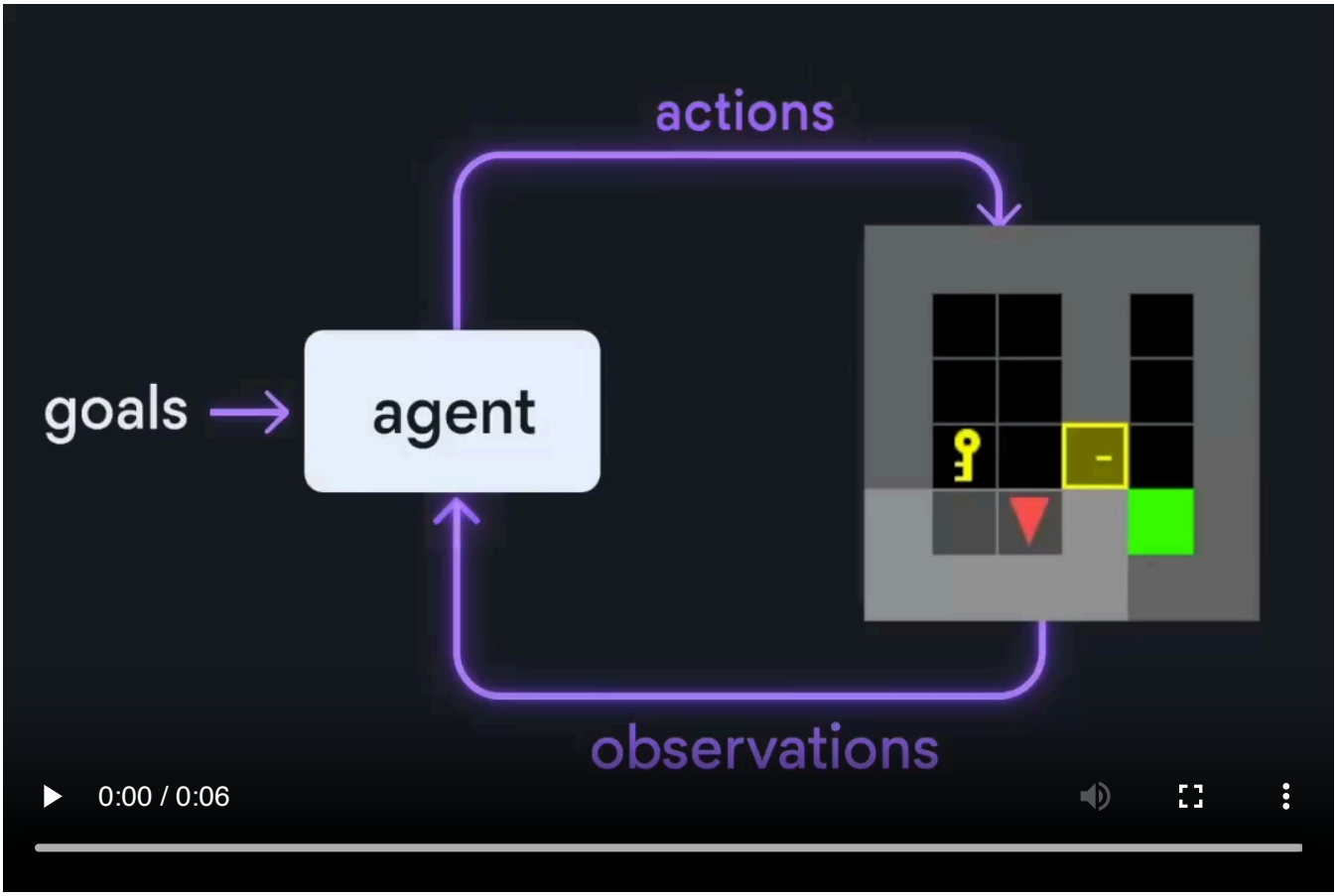
NB: A complete implementation of this solution is available here. Please note that the full code is a research prototype and not yet production-ready.

```
using RxInfer
```

```
@model function minigrid_aif(prior_state, prior_key, prior_door, target_location, horizon, transition_params, observation_params)
    initial_state ~ prior_state
    key_location ~ prior_key
    door_location ~ prior_door

    prev_state = initial_state
    for t in 1:horizon
        action[t] ~ ExplorationPrior()
        state[t] ~ AmbiguityPrior()
        state[t] ~ DiscreteTransition(prev_state, transition_params, key_location, door_location, action)
        observation[t] ~ DiscreteTransition(state[t], observation_params, key_location, door_location)
    end
    state[end] ~ Goal(target_location)
end
```

The animation below is a recording of representative agent behavior during the VFE minimization process. Initially, the agent rotates to search for the key—an uncertainty-reducing strategy **driven by the ambiguity term** of the Expected Free Energy. Once the key's location is revealed, the agent retrieves it and proceeds to the goal location. This constitutes goal-directed behavior that **minimizes the risk term** in the EFE. Because the environmental dynamics are assumed to be known, no novelty-driven exploration occurs in this example.


actions
goals
agent
observations
0:00 / 0:06

Discussion

The Free Energy Principle and active inference are deep and fast-moving areas of research. They bring fresh ideas to intelligent reasoning, control, and AI, with exciting applications in robotics, adaptive systems, cognitive modeling, and more. There's a lot to unpack, but in this lecture, we've only had time to scratch the surface. To wrap up, we'll end with a few closing thoughts.

Comparison Decision-theoretic vs Active Inference Agents

The idea of framing decision-making and planning as the minimization of expected cost over future states has become foundational across many disciplines, including machine learning (e.g., reinforcement learning), control theory (e.g., model-predictive and optimal control), and economics (e.g., utility theory and operations research). In what follows, we will refer to such systems collectively as *decision-theoretic* (DT) agents.

AIF agents fundamentally differ from DT agents in that variational free energy (VFE) minimization is the sole underlying process. As a result, policies are evaluated based on a function of beliefs about states, rather than directly on the states themselves. The FEP formally captures this distinction.

From an engineering perspective, what is gained by moving from DT to AIF agents? While our treatment here is necessarily brief and not intended as a comprehensive academic assessment, several key advantages already stand out:

- **A principled grounding in fundamental physics**
    - If we aim to understand how brains—human or animal—give rise to intelligent behavior, we must start from the premise that they operate entirely within the laws of physics. The FEP is consistent with this physical grounding.
- **Balanced goal-directed and information-seeking behavior**
    - The epistemic components that emerge naturally from the EFE functional often need to be added through ad hoc mechanisms in decision-theoretic frameworks that do not explicitly score beliefs about future states.
- **No need for task-specific reward (or value) functions**
    - In DT agents, a recurring question is: where do the reward functions come from? These functions are typically hand-crafted. In an AIF agent, preferences are encoded as prior

distributions over desired outcomes. These priors can be parameterized and updated through hyper-priors and Bayesian learning at higher levels of the generative model, allowing agents to adapt their preferences on the fly, rather than relying on externally specified reward functions.

- **A universal cost function for all problems**
  - A related limitation of the DT systems is that the value function must be specified anew for each problem. Brains, however, cannot afford to construct a different value function for every task, given that thousands of novel problems are encountered daily. In contrast, AIF agents employ the same free-energy functional F that measures the quality of the beliefs, for all problems. The structure of F (complexity minus accuracy) does not change and applies universally.
- **AIF agents are explainable and trustworthy by nature**
  - Explainability and trustworthiness are critical concerns in AI, for instance, in medical AI applications. An AIF agent's reasoning process is Bayes-optimal, and therefore logically consistent and inherently *trustworthy*. Crucially, domain-specific knowledge and inference are cleanly separated: all domain-specific assumptions reside in the model. As a result, the agent's behavior can be *explained* as the logical (Bayesian) consequence of its generative model.
- **Robustness by realization as a reactive message passing process!**
  - In contrast to decision-theoretic (DT) agents, an active inference (AIF) agent can be fully realized as a reactive variational message passing (RMP) process, since variational free energy (VFE) minimization is the only ongoing process. RMP is an event-driven, fully distributed process—both in time and space—that exhibits robustness to fluctuating computational resources. It "lands softly" when resources such as power, data, or time become limited. As a result, an AIF agent continues to function during power dips, handles missing or noisy observations gracefully, and can be interrupted at any time during decision-making without catastrophic failure, making it naturally suited for real-world, resource-constrained environments.
- **Easy to code**
  - Since VFE minimization can be automated by a toolbox, the engineer's main task is to specify the generative model and priors, typically achievable within a single page of code. However, because a professional-level automated VFE minimization toolbox is not (yet) available, this "easy-to-code" feature has not (yet) been realized.
- **Other advantages**
  - Additional advantages include the potential for scalability, particularly in real-time applications. Realizing this potential will require further research into efficient, real-time message passing capabilities that are difficult to match in frameworks that cannot be implemented as reactive message passing processes.

While the advantages listed above hold great promise for the future of synthetic AIF agents in solving complex engineering problems, it's important to acknowledge current limitations. The vast majority of engineers and scientists have been trained within DT frameworks, and the **tooling and methodologies for DT agents are far more mature**. For many practical problems, several of the above-mentioned advantages of AIF agents have yet to be conclusively demonstrated in real-world applications.

> 🎯 **Key concept**
>
> In principle, active inference offers a unified way to address several long-standing challenges faced by current approaches to agentic AI. In practice, however, the available toolboxes for active inference remain less mature than those for more conventional AI paradigms, such as deep learning, large language models, and reinforcement learning.

The FEP: A New Frontier for Understanding Intelligent Behavior

The FEP is often misunderstood as a scientific theory that counterexamples can falsify. In reality, the **FEP is a principle**, a general modeling framework for describing the dynamics of systems that exhibit ("life-like") attractor dynamics, repeatedly returning to states that preserve their functional organization and structural integrity. According to the FEP, such systems can be interpreted as performing variational Bayesian inference in a generative model, where the goal priors correspond to the system's attractors.

In this lecture, we've seen how the FEP can be used to describe the dynamics of rational AI agents, but its reach goes far beyond AI and control. As a unifying framework for understanding adaptive self-organization, the FEP touches neuroscience, biology, cognition, and even physics.

For example, the Expected Free Energy used to evaluate policies is not just an arbitrary cost function—it follows naturally from common assumptions in fundamental physics. EFE-based policy scoring also makes sense from a philosophical standpoint: if minimizing variational free energy is the only process driving the system, then it is a logical consequence to rank policies by how much VFE we expect them to minimize in the future.

Looking ahead to the future of artificial intelligence, adaptive robotics, and agentic AI, the Free Energy Principle stands out as a framework with the potential to transform not only how we build

intelligent systems, but how we fundamentally understand their nature, purpose, and place within the broader landscape of self-organizing life.

Summary

🎯 **Key concept** ↑ Jump to source

Friston's key insight is that intelligent behavior necessarily involves uncertainty-reducing behavior, which should be framed as the optimization of a functional of beliefs (i.e., probabilities) over future states.

🎯 **Key concept** ↑ Jump to source

The **Free Energy Principle** (FEP) can be seen as a specific instance of the Least Action Principle applied to biological systems, which are technically systems that act to preserve their functional and structural integrity.

🎯 **Key concept** ↑ Jump to source

The process underlying naturally intelligent behavior is termed **Active Inference** (AIF). It can be described as the minimization of a variational free energy under a generative model that incorporates a rollout into the future.

🎯 **Key concept** ↑ Jump to source

In principle, active inference offers a unified way to address several long-standing challenges faced by current approaches to agentic AI. In practice, however, the available toolboxes for active inference remain less mature than those for more conventional AI paradigms, such as deep learning, large language models, and reinforcement learning.

← Previous lecture

Next lecture →

Code

```
1 using BmlipTeachingTools
```

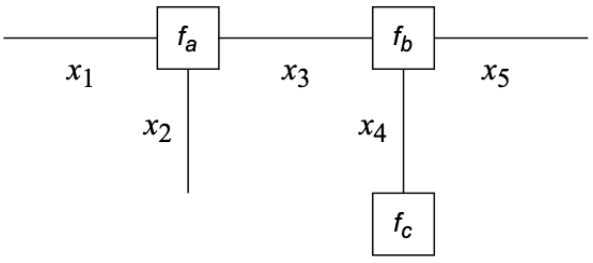
Factor Graphs

Table of Contents

Forney-style Factor Graph Construction Rules

Consider a function

$$f(x_1,x_2,x_3,x_4,x_5) = f_a(x_1,x_2,x_3)\cdot f_b(x_3,x_4,x_5)\cdot f_c(x_4)\,.$$



The factorization of this function can be graphically represented by a **Forney-style Factor Graph** (FFG), see image on the right.

An FFG is an **undirected** graph subject to the following construction rules (Forney, 2001)

1. A **node** for every factor;
2. An **edge** (or **half-edge**) for every variable;

3. Node $f_\bullet$ is connected to edge x **iff** variable x appears in factor $f_\bullet$.

A **configuration** is an assignment of values to all variables. A configuration $\omega = (x_1, x_2, x_3, x_4, x_5)$ is said to be **valid** iff $f(\omega) \neq 0$

Equality Nodes for Branching Points

Consider the factorization (where x_2 appears in three factors)

$$f(x_1, x_2, x_3, x_4) = f_a(x_1, x_2) \cdot f_b(x_2, x_3) \cdot f_c(x_2, x_4)$$

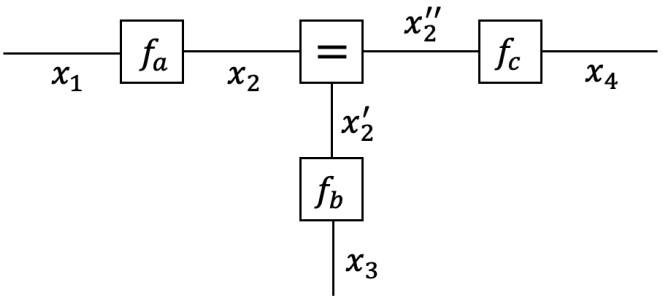
For the factor graph representation, we will instead consider the function g, defined as

$$g(x_1, x_2, x_2', x_2'', x_3, x_4) = f_a(x_1, x_2) \cdot f_b(x_2', x_3) \cdot f_c(x_2'', x_4) \cdot f_=(x_2, x_2', x_2'')\,,$$

where

$$f_=(x_2, x_2', x_2'') \triangleq \delta(x_2 - x_2')\,\delta(x_2 - x_2'')$$

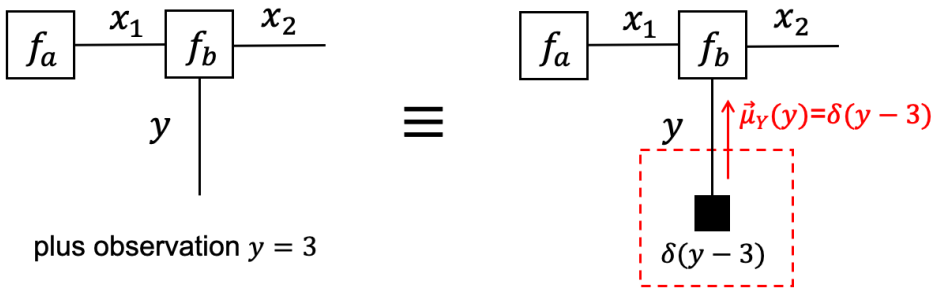
is a so-called **equality** (or branching) node. The constraint $f_=(x, x', x'')$ enforces that $x = x' = x''$ **for every valid configuration**.



Note that through introduction of auxiliary variables x_2' and x_2'' and a factor $f_=(x_2, x_2', x_2'')$, each variable in g appears in maximally two factors.

Representing Observations

An observation, say $y = 3$, can be represented by a **delta node** $f(y) = \delta(y - 3)$ to terminate the half-edge for variable y. In an FFG, we visualize a delta node by a small black box,



Key concept

Any factorized probabilistic model, including a set of observations for that model, can be represented by a Terminated Forney-style factor graph.

Message Passing-based Inference

Inference in Factorized Models

Factorizations offer opportunities to reduce the computational cost of inference by exploiting the conditional independence structure of the model.

Assume we wish to compute the marginal

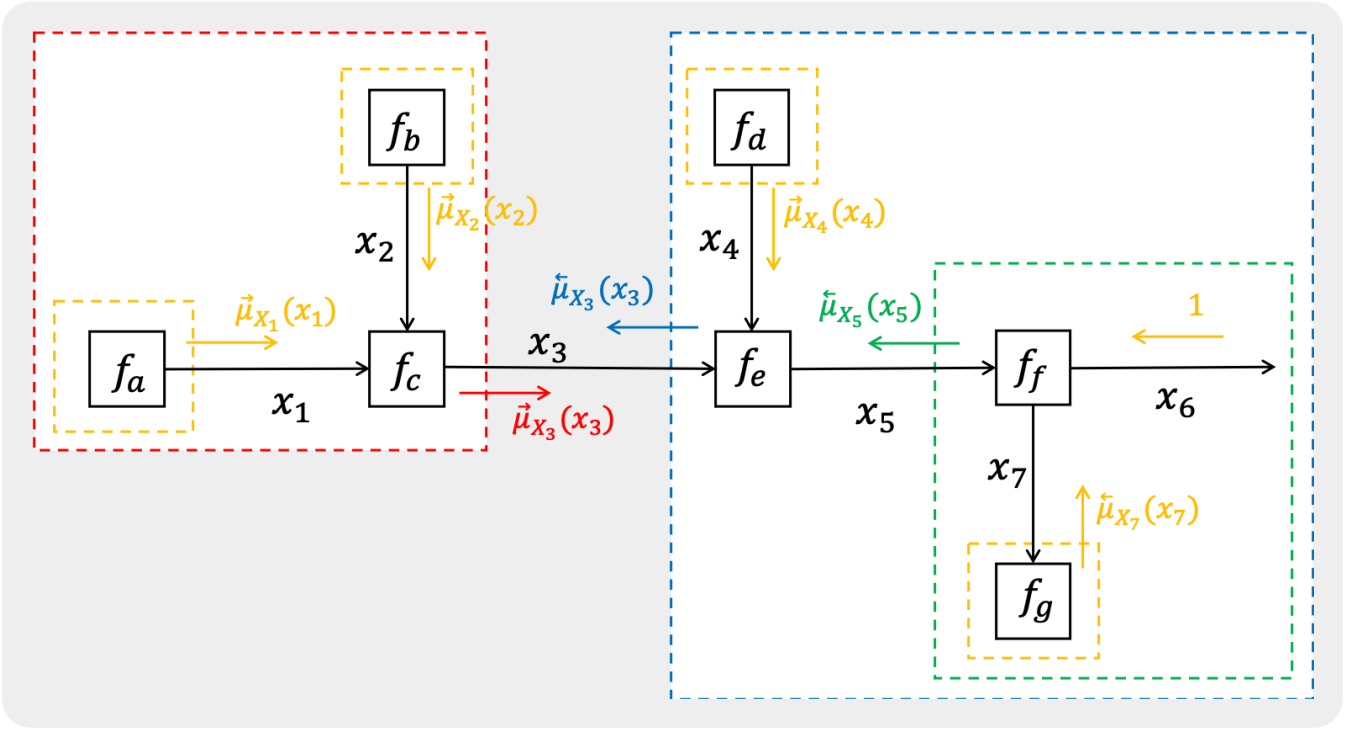
$$\bar{f}(x_3) \triangleq \sum_{x_1,x_2,x_4,x_5,x_6,x_7} f(x_1, x_2, \ldots, x_7)$$

for a model f with given factorization

$$f(x_1, x_2, \ldots, x_7) = f_a(x_1) f_b(x_2) f_c(x_1, x_2, x_3) f_d(x_4) f_e(x_3, x_4, x_5) f_f(x_5, x_6, x_7) f_g(x_7)$$

Note that, if each variable x_i can take on 10 values, then computing the marginal $\bar{f}(x_3)$ takes about 10^6 (=1 million) additions.

We draw here the FFG for the factorized distribution:



Due to the factorization of $f(x_1, x_2, \ldots, x_7)$ and the Generalized Distributive Law, we can decompose the marginalization operation to the following product-of-sums:

$$\bar{f}(x_3) = \underbrace{\left(\sum_{x_1, x_2} \underbrace{f_a(x_1)}_{\vec{\mu}_{X_1}(x_1)} \underbrace{f_b(x_2)}_{\vec{\mu}_{X_2}(x_2)} f_c(x_1, x_2, x_3) \right)}_{\vec{\mu}_{X_3}(x_3)} \cdot \underbrace{\left(\sum_{x_4, x_5} \underbrace{f_d(x_4)}_{\vec{\mu}_{X_4}(x_4)} f_e(x_3, x_4, x_5) \cdot \underbrace{\left(\sum_{x_6, x_7} f_f(x_5, x_6, x_7) \underbrace{f_g(x_7)}_{\overleftarrow{\mu}_{X_7}(x_7)} \right)}_{\overleftarrow{\mu}_{X_5}(x_5)} \right)}_{\overleftarrow{\mu}_{X_3}(x_3)}$$

which, in case x_i has 10 values, **requires a few hundred additions and is therefore computationally (much!) lighter** than executing the whole sum $\sum_{x_1, \ldots, x_7} f(x_1, x_2, \ldots, x_7)$

> 🎯 **Key concept**
>
> By naturally exploiting the distributive law, message-passing inference in factor graphs can drastically reduce the computational complexity of Bayesian inference in sparse models.

Closing-the-Box and Message Passing Interpretation

Note that the intermediate result $\vec{\mu}_{X_3}(x_3)$ is obtained by multiplying all enclosed factors (f_a, f_b, f_c) by the red dashed box, followed by marginalization (summing) over all enclosed variables (x_1, x_2),

$$\vec{\mu}_{X_3}(x_3) = \underbrace{\sum_{x_1} \sum_{x_2}}_{\text{enclosed variables}} \underbrace{f_a(x_1) f_b(x_2) f_c(x_1, x_2, x_3)}_{\text{enclosed factors}}$$

This operation is known as **Closing-the-Box**. The result is a new **composite node** that holds the factor $\vec{\mu}_{X_3}(x_3)$, and is visually represented by the red dashed box in the factor graph. The

composite node $\overrightarrow{\mu}_{X_3}(x_3)$ depends only on the variable(s) that cross the boundary of the box (in this case x_3) and effectively replaces the internal subgraph contained within the red box.

The Closing-the-box operation can alternatively be interpreted as **passing a message** from the newly created composite node to the rest of the graph.Closing the red box around f_a, f_b and f_c leads to an outgoing message $\overrightarrow{\mu}_{X_3}(x_3)$ for node f_c, given by

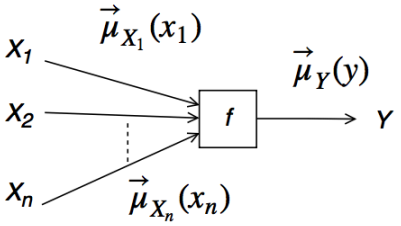
$$\underbrace{\overrightarrow{\mu}_{X_3}(x_3)}_{\substack{\text{outgoing}\\ \text{message}}} = \sum_{x_1}\sum_{x_2} \underbrace{\overrightarrow{\mu}_{X_1}(x_1)\overrightarrow{\mu}_{X_2}(x_2)}_{\substack{\text{incoming}\\ \text{messages}}} \underbrace{f_c(x_1, x_2, x_3)}_{\text{factor}}$$

Sum-Product Messages

This recipe holds generally. For a node $f(y, x_1, \ldots, x_n)$ with incoming messages $\overrightarrow{\mu}_{X_1}(x_1)$, $\overrightarrow{\mu}_{X_2}(x_2)$, $\ldots$,$\overrightarrow{\mu}_{X_n}(x_n)$, the outgoing message is given by (Loeliger (2007), pg.1299):

$$\underbrace{\overrightarrow{\mu}_Y(y)}_{\substack{\text{outgoing}\\ \text{message}}} = \sum_{x_1,\ldots,x_n} \underbrace{\overrightarrow{\mu}_{X_1}(x_1)\cdots\overrightarrow{\mu}_{X_n}(x_n)}_{\substack{\text{incoming}\\ \text{messages}}} \cdot \underbrace{f(y, x_1, \ldots, x_n)}_{\substack{\text{node}\\ \text{function}}} \qquad \text{(SP)}$$

Equation (SP) is called a **Sum-Product** message, so named because the computation involves evaluating a sum-of-products. Note that all SP messages in an FFG can be computed from information that is **locally available** at each node.



If the factor graph for the whole model has no cycles, i.e., the FFG is a tree, then passing SP messages from the terminal nodes to the internal (latent) variables yields exact Bayesian marginals for all hidden variables. This inference method is known as the **Sum-Product** (SP) algorithm.

🎯 Key concept

For a node $f(y, x_1, \ldots, x_n)$ with incoming messages $\overrightarrow{\mu}_{X_1}(x_1)$, $\overrightarrow{\mu}_{X_2}(x_2)$, $\ldots$,$\overrightarrow{\mu}_{X_n}(x_n)$, the outgoing message is given by the **sum-product rule**:

$$\overrightarrow{\mu}_Y(y) = \sum_{x_1,\ldots,x_n} \overrightarrow{\mu}_{X_1}(x_1) \cdots \overrightarrow{\mu}_{X_n}(x_n) \cdot f(y, x_1, \ldots, x_n)$$

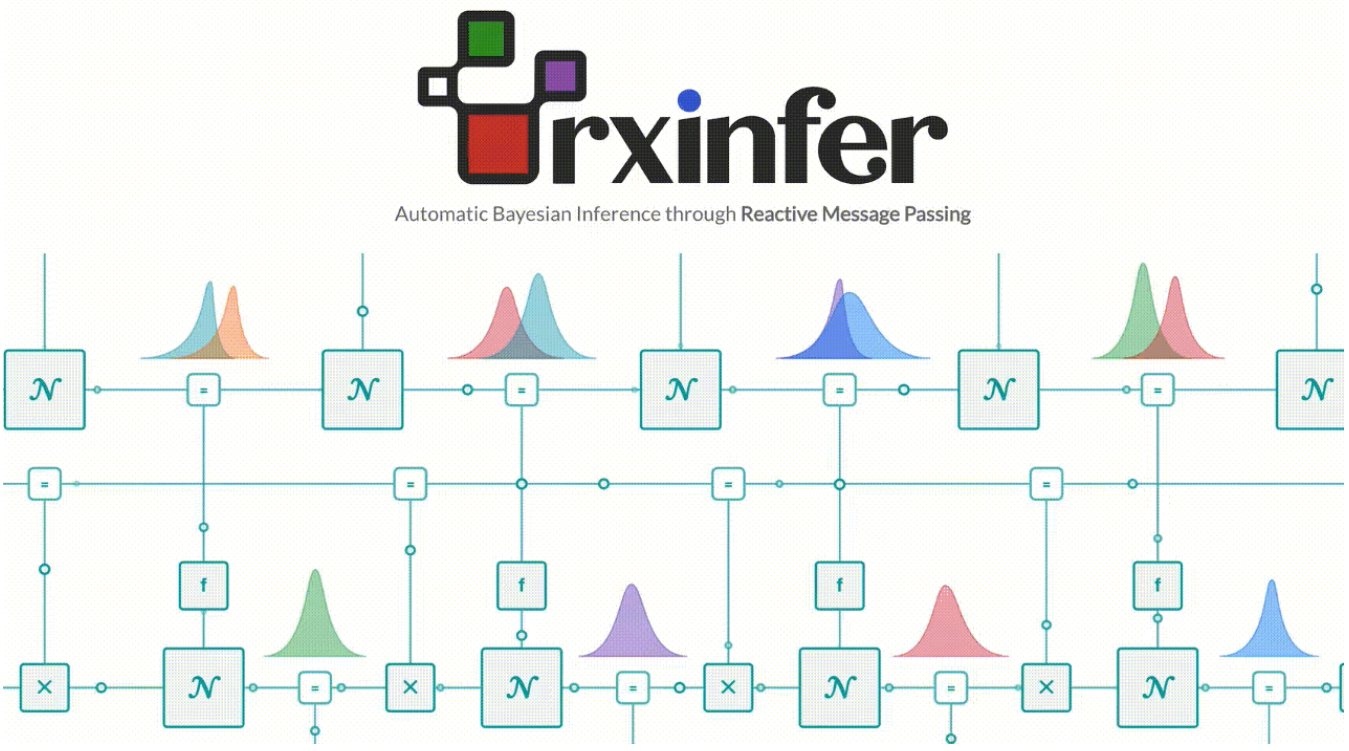
RxInfer: A Toolbox for Automated Bayesian inference

Automating Bayesian Inference by Message Passing

If the update rules for common node types in a graph have been tabulated, then inference by message passing comes down to executing a set of table-lookup operations, thus creating a completely **automatable Bayesian inference framework**.

In our research lab BIASlab at TU Eindhoven, we are developing RxInfer, which is a (Julia) toolbox for automating Bayesian inference by message passing in a factor graph.

In general, a code package that automates Bayesian inference is called a Probabilistic Programming Language (PPL). RxInfer is a PPL that automates inference through message passing-based inference in a factor graph.



The figure above (a screen recording from the RxInfer webpage) is an animated GIF illustrating how RxInfer operates. The model is represented as a graph in which each node passes messages to its neighbors. When messages meet on an edge, the belief about the variable associated with that edge is updated.

> 🎯 **Key concept**
>
> "In `RxInfer`, once the model is specified and the observations are provided, Bayesian inference is carried out automatically via the `infer()` function."

Although RxInfer is still under active development, my prediction is that within 5–10 years, RxInfer, or a comparable toolbox, will be able to automate Bayesian inference for virtually any interesting probabilistic model you can conceive. In principle, you will then have all the tools needed to implement the four-step Bayesian ML recipe—model specification, parameter learning, model evaluation, and application, for any (Bayesian) information processing problem.

Summary

Key concept ↑ Jump to source

Any factorized probabilistic model, including a set of observations for that model, can be represented by a Terminated Forney-style factor graph.

Key concept ↑ Jump to source

By naturally exploiting the distributive law, message-passing inference in factor graphs can drastically reduce the computational complexity of Bayesian inference in sparse models.

Key concept ↑ Jump to source

For a node $f(y, x_1, \ldots, x_n)$ with incoming messages $\overrightarrow{\mu}_{X_1}(x_1)$, $\overrightarrow{\mu}_{X_2}(x_2)$, $\ldots$, $\overrightarrow{\mu}_{X_n}(x_n)$, the outgoing message is given by the **sum-product rule**:

$$\overrightarrow{\mu}_Y(y) = \sum_{x_1,\ldots,x_n} \overrightarrow{\mu}_{X_1}(x_1) \cdots \overrightarrow{\mu}_{X_n}(x_n) \cdot f(y, x_1, \ldots, x_n)$$

Key concept ↑ Jump to source

"In `RxInfer`, once the model is specified and the observations are provided, Bayesian inference is carried out automatically via the `infer()` function."

← Previous lecture

Next lecture →

```
navigate_prev_next(
    "https://bertdv.github.io/mlss-2026/lectures/Intelligent%20Agents%20and%20Active%20Inference.html",
    nothing
)
```

Code

```
using BmlipTeachingTools
```

```
using Plots, LinearAlgebra, LaTeXStrings
```