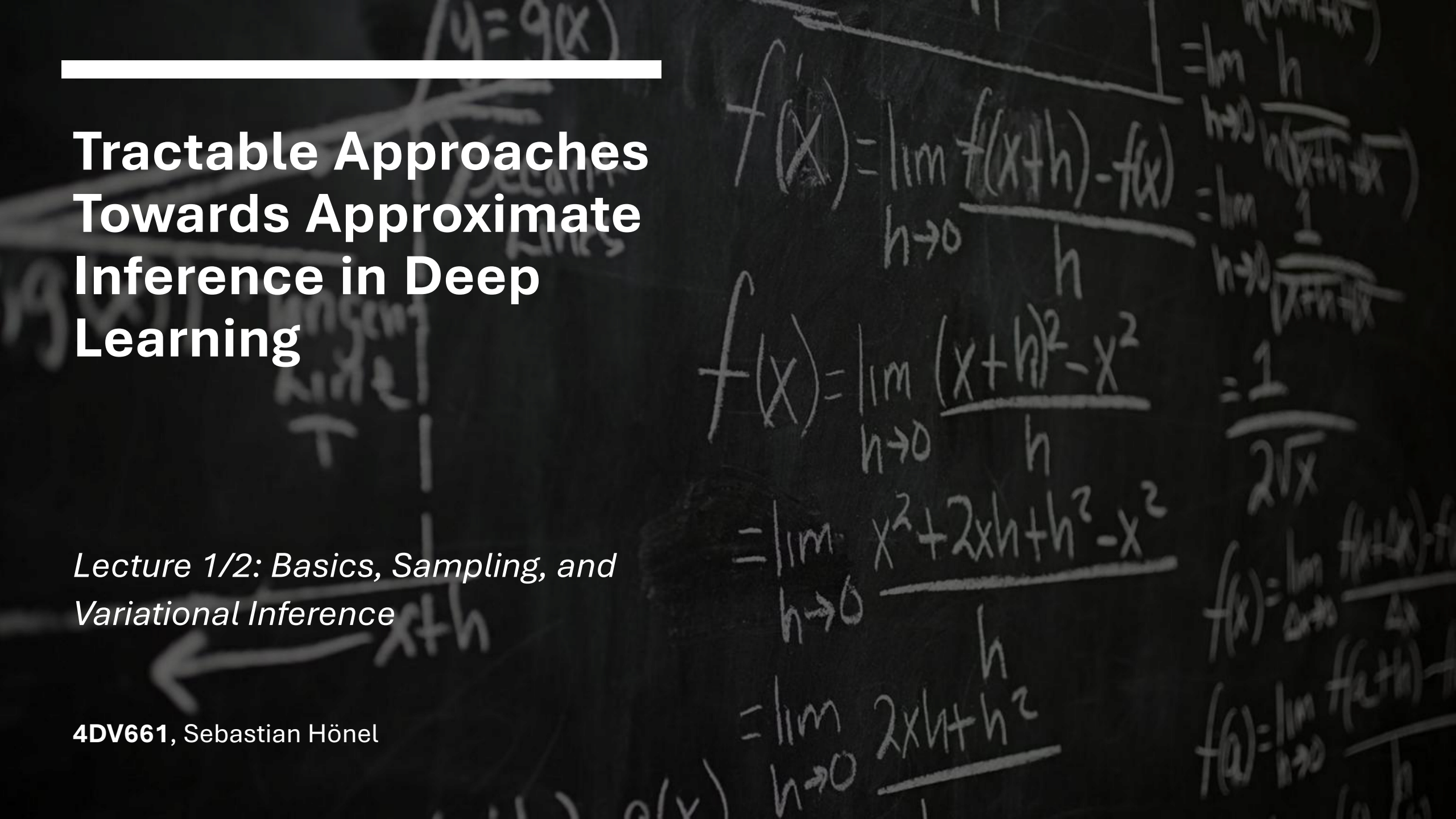


Tractable Approaches Towards Approximate Inference in Deep Learning

Lecture 1/2: Basics, Sampling, and Variational Inference

4DV661, Sebastian Hönel

Approximate Inference

Refers to any approach that simplifies complicated, hard-to-solve or ***intractable*** probabilistic computations in complex models like neural networks or *deep generative models*. Examples are:

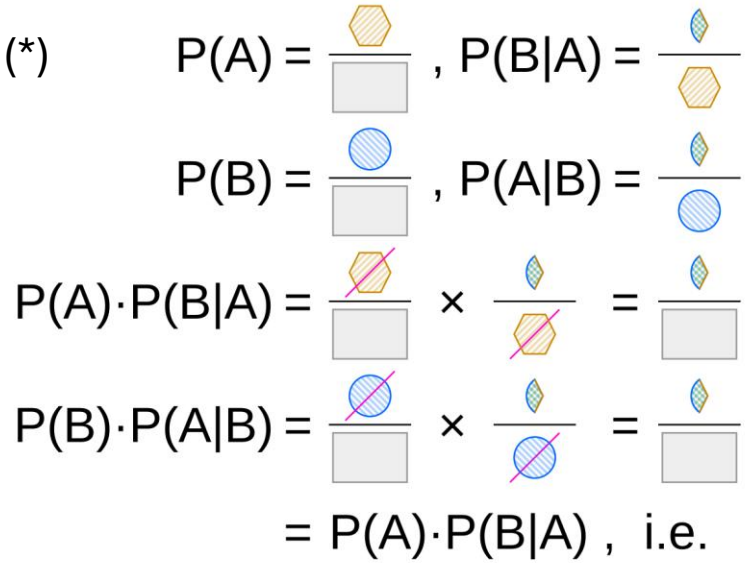
- Calculating *posterior* distributions
- **Variational inference**
- Markov Chain **Monte Carlo** (MCMC)
- **Density estimation**
- Missing-value imputation
- **(Conditional) Sample generation**
- …

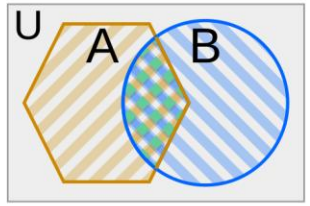
Contents – Lecture 1

1. Notation
2. Bayes‘ Theorem
3. Expected Values, Evidence
4. Sampling
   - From well-known, Rejection, Importance, Metropolis-Hastings
5. Variational Inference
   - Motivating Example: Spam/Ham Detection
6. Kullback–Leibler *f*-Divergence
7. Variational Bayes
8. Jensen‘s Inequality
9. Evidence Lower BOund (ELBO)
6. Mean-field variational inference: $q(\boldsymbol{z}) = \prod_i q(z_i)$
10. Monte Carlo Approximation of ELBO
11. Reparameterization Trick

Bayes‘ Theorem (Notation)

(*)





$$P(A|B) = \frac{P(A)\cdot P(B|A)}{P(B)}$$

$$P(B|A) = \frac{P(B)\cdot P(A|B)}{P(A)}$$

- $P(A), P(B)$ – marginal probabilities of A / B.
- $P(A, B)$ – or: $P(A \cup B)$ – the **joint** probability of A and B
  - Symmetry property: $P(A, B) = P(B, A)$
- **Sum Rule**: $P(A) = \sum_i P(A, B_i)$.
  - Here, we are summing out (**marginalizing**) B.
  - Using the product rule: $P(A) = \sum_i P(A|B_i) \cdot P(B_i)$.
- **Product (or Chain) rule**: $P(A \cup B) = P(B|A) \cdot P(A)$.
  - From the **symmetry property** it follows $P(A, B) = P(A|B) \cdot P(B)$.
  - Also, $P(B, A) = P(A|B) \cdot P(B)$ and $P(B, A) = P(B|A) \cdot P(A)$. This allows us to establish a relation between conditional probabilities.

Bayes‘ Theorem: $P(A|B) = \frac{P(A \cup B)}{P(B)} = \frac{P(B|A) \cdot P(A)}{P(B)} = \frac{P(A|B) \cdot \cancel{P(B)}}{\cancel{P(B)}}$. (**)

(*) Image taken from Wikipedia.

(**) While the last is correct, it‘s not useful for establishing a relation between $p(A|B)$ and $p(B|A)$.

Bayes‘ Theorem (Notation)

$$P(\theta|Y) = \frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}$$

A or θ – typically represents the **parameters** (the unknown quantities we are trying to infer)
B or Y – typically represents the **data** (the observed outcomes)

$P(Y|\theta)$ – Called the **LIKELIHOOD** (function). This is a conditional probability. However, it does not neccessarily need to be a proper probability density. Often, it is written as $L(\theta; Y)$ because it is viewed as a function of the parameters θ with observations Y fixed.

$P(\theta)$ – The **PRIOR**, a probability distribution that encodes our prior beliefs over what the parameters in θ are without having observed (or considering) any Y.

$P(Y)$ – The **EVIDENCE**, a constant scalar that normlizes the posterior. This may also be called partition function (we will use that term later). Measures the overall average likelihood of the data under *all possible model realizations* (see next slides).

$P(\theta|Y)$ – The **POSTERIOR**, a proper probability density (function) that contains our updated beliefs of the parameter values θ after incorporating the observations Y.

Bayes‘ Theorem (Evidence, BIC)

$$P(\theta|Y) = \frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}$$

$P(Y)$, termed **EVIDENCE** (**marginal likelihood**), defined using the sum and product rules:

$$P(Y) = \int_{\theta} P(Y|\theta) \cdot P(\theta)\, d\theta.$$

Note how this integral is a weighted sum over all possible parameters, keeping the data constant. Therefore, it tells us, on average, how well our model (*overall*) fits the data. This is actually exploited in the Bayesian Information Criterion (**BIC**). While this integral is usually **intractable**, the BIC uses an asymptotic (Laplace) approximation(*) that allows us to compare any two models parameterized by $\hat{\theta}_1$ and $\hat{\theta}_2$, respectively.

As the number of parameters in θ tends to infinity, a model will perfectly fit the data. However, the BIC penalizes larger numbers of parameters, effectively becoming an instance of **Occam‘s Razor**: Between two models that explain the data equally well, the one with less parameters is a better choice.

(*): A Gaussian approximation around the posterior mode that gives approximate evidence:

$$\mathrm{BIC} = -2\log\big(P(Y|\theta)\big) + k$$

$$\log(P(Y)) \approx \log\left(P\left(Y|\hat{\theta}\right)\right) - \frac{k}{2}\log(n)$$

Bayes‘ Theorem (Evidence, BIC [cont‘d])

$$P(\theta|Y) = \frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}$$

Why can we integrate out θ?

$$P(Y) = \int_{\theta} P(Y|\theta) \cdot P(\theta)\, d\theta$$

In probability theory, "integrating out" (or marginalizing) a random variable we do not directly observe or do not currently care about is an extremely common and valid operation. When we do it:

- We're not saying “$P(Y)$ depends on θ” directly. Instead, we say ”$P(Y)$ is obtained by considering every possible value of θ, appropriately weighting each scenario by how probable it is to begin with”.
- By performing this integration, we’re essentially *averaging* out θ’s influence. While individual terms inside the integral depend on θ, once we do the integral, the result no longer depends on θ explicitly. The parameter has been “removed” through averaging.

Expected Values (PRML 1.2)

One of the most important operations involving probabilities is that of finding weighted averages of functions. The average value of some function $f(x)$ under a probability distribution $p(X)$ is called the expectation of $f(x)$ and will be denoted by $\mathbb{E}[f] = \int p(x) \cdot f_X(x)\, dx$. We use this quite often in machine learning. Consider the following:

We are trying to predict whether an image shows a cat or a dog. We have a distribution of animal images, X, and a distribution of associated labels, Y. Our loss function $L(\cdot)$ is the binary cross-entropy, which we wish to minimize. Our dataset is somewhat imbalanced, i.e., $Y \nsim U$. We join the images with their assoc. labels in a new variable $Z = (X, Y)$.

Then the *average weighted* loss is given by:

$$\mathbb{E}_{Z \sim p_Z(z)}[L(Z)] = \int L(z) \cdot p_Z(z)\, dz.$$

We are taking the average of $L(Z)$, while averaging over Z, which is distributed according to the probability density p_Z.

In case of a finite set of N examples, the expectation can be approximated simply:

$$\mathbb{E}_{Z \sim p_Z(z)}[L(Z)] \simeq \frac{1}{N} \sum_{n=1}^{N} L(z_n).$$

This is of great importance later for sampling, computing lower bounds, and enabling MCMC approximations of otherwise intractable integrals!

The Evidence as an Expectation

$$P(\theta|Y) = \frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}$$

$$P(Y) = \int_{\theta} P(Y|\theta) \cdot P(\theta)\, d\theta,$$

Let's recognize that this looks like an **EXPECTATION**. For a random variable X with probability density $p(x)$, the expectation of the function $f(X)$ is:

$$\mathbb{E}_{X\sim p(x)}[f(X)] = \int f(x) \cdot p(x)\, dx.$$

If we choose the relevant random variable to be θ, the density to be the prior $\mathrm{p} = p(\theta)$, and the function we are taking the expectation of to be the likelihood function $\mathrm{f} = P(Y|\theta)$:

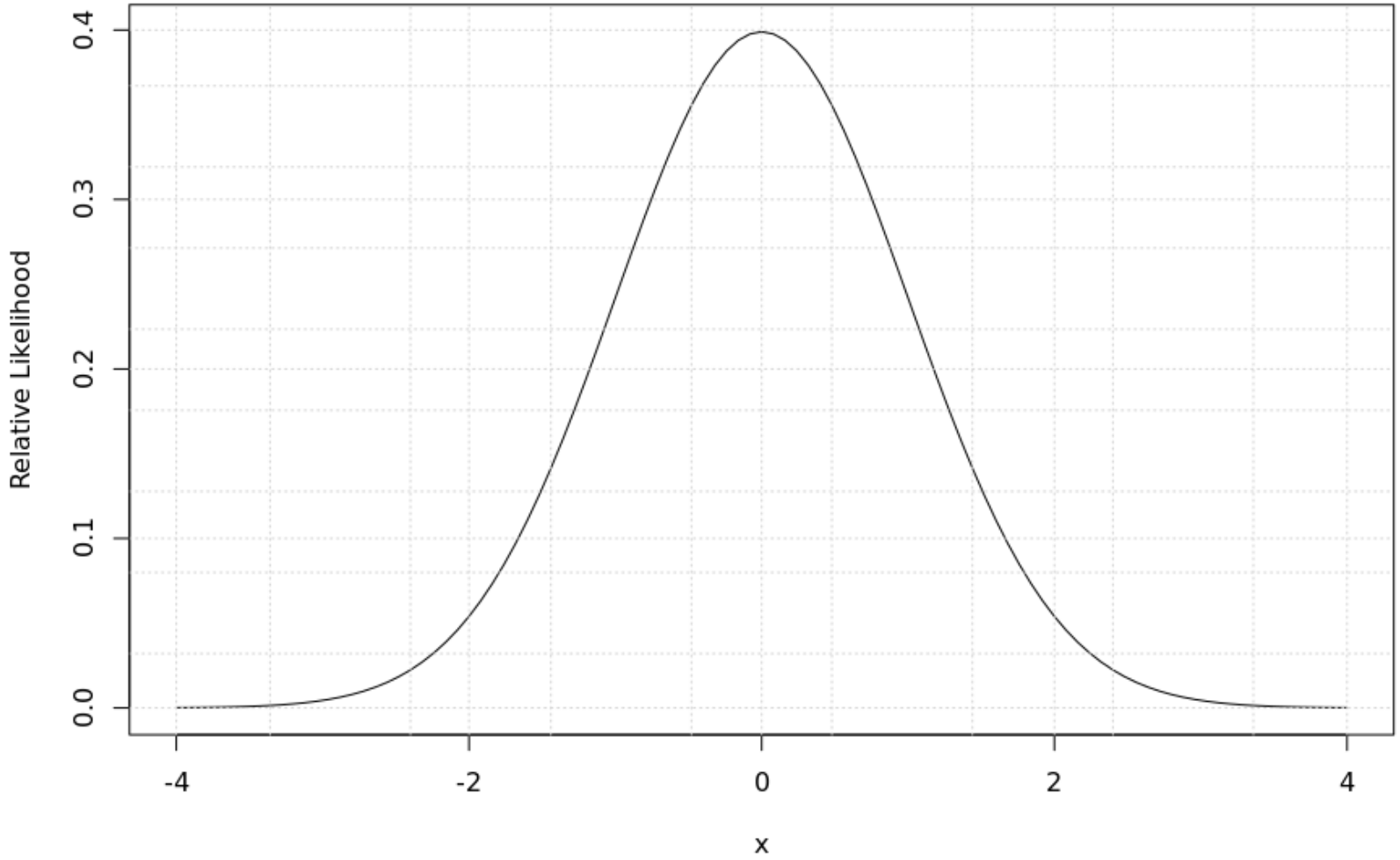
$$P(Y) = \int P(Y|\theta) \cdot P(\theta)\, d\theta = \mathbb{E}_{\theta\sim P(\theta)}[P(Y|\theta)].$$

Thus explicitly:

$$P(Y) = \mathbb{E}_{\theta\sim P(\theta)}[P(Y|\theta)].$$

Sampling

Please identify this:



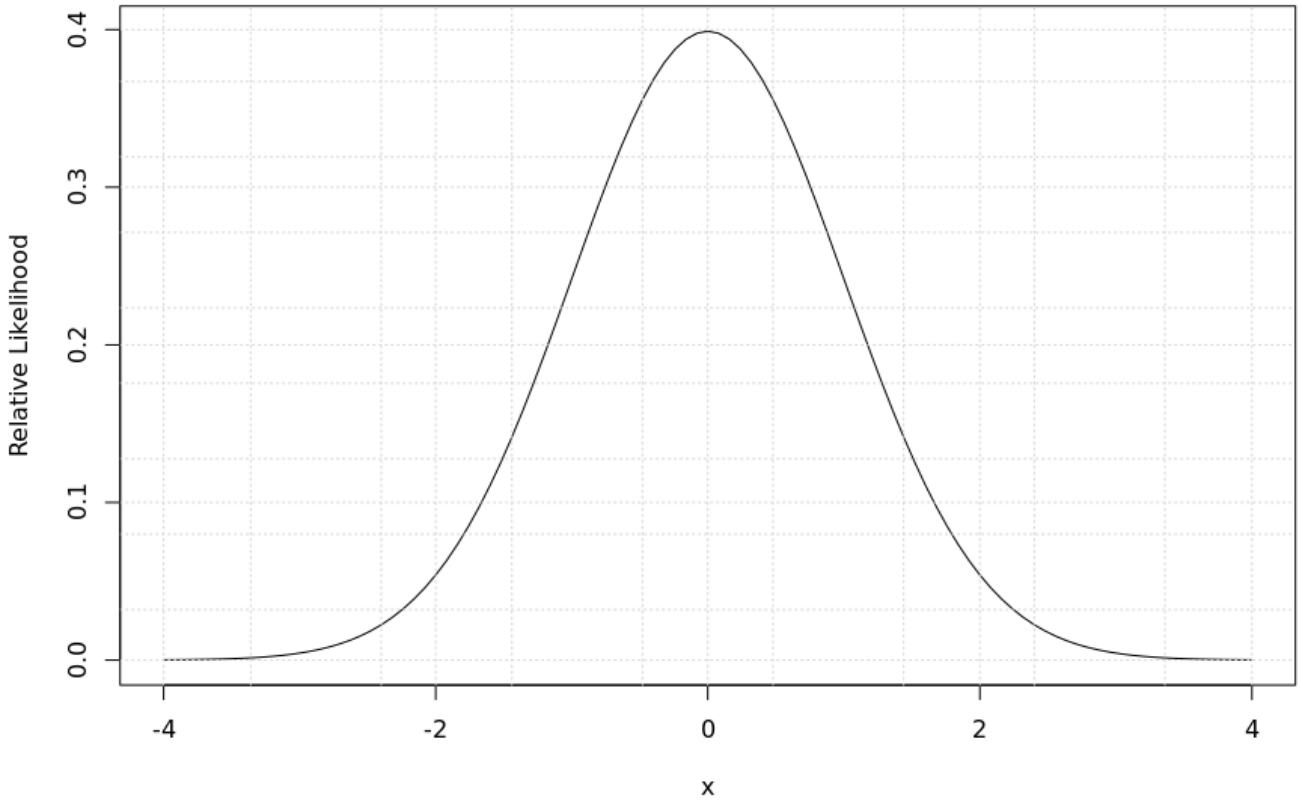
Sampling

Exactly, the probability density (function image) of a *standard normal* distribution with $\mu = 0$ and $\sigma = 1$.

The normal distribution is a well-defined and well-understood family of probability distributions.

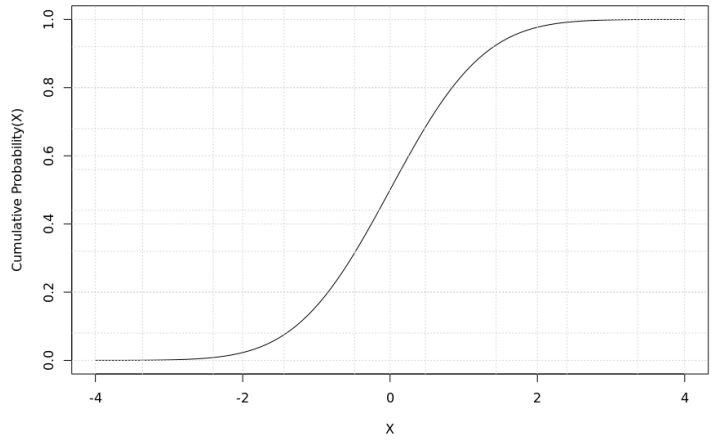
$$p(\mu, \sigma; x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

But: How do we sample from it?

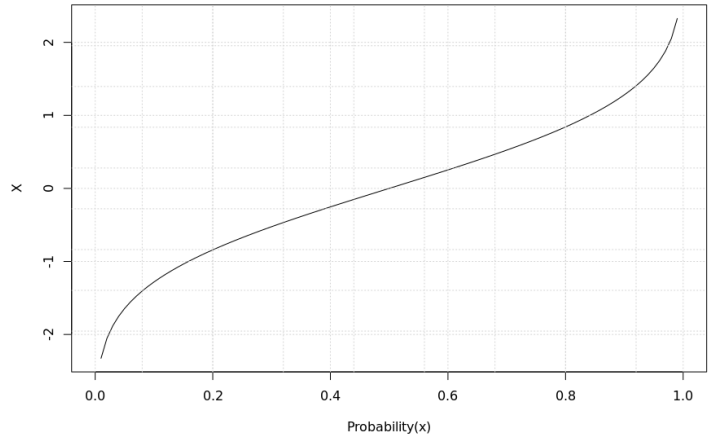


Sampling (from well-known Distributions)

CDF:



PPF:



CDF: $\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$, where $\operatorname{erf}(\cdot)$ is the error function.

For a well-known distribution, it means we have **analytical** expressions for its cumulative distribution function (**CDF**), as well as its quantile- or percent-point function (**PPF**).

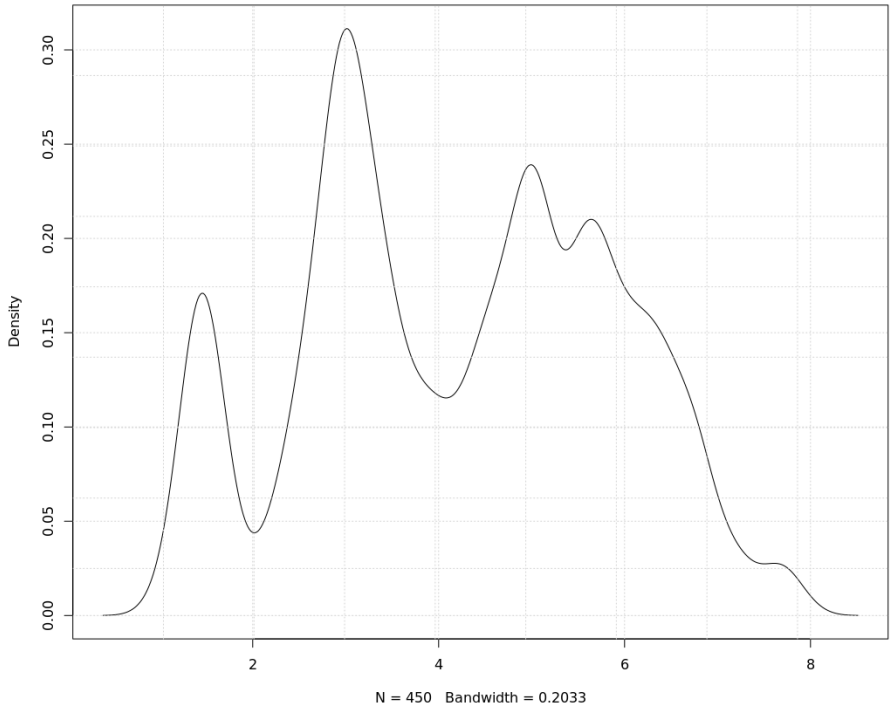
To create samples, we simply plug in numbers generated from a uniform distribution ($U \sim [0,1]$) into the PPF. This is called **inverse transform sampling**.

In sampling, If we do this often enough, the resulting sample's distribution will approach the true (but usually unknown) population density (the law of large numbers).

PPF: $\mu+\sigma\sqrt{2}\,\operatorname{erf}^{-1}(2p-1)$.

Sampling (density estimate)

How about this one here? This is a Kernel Density Estimate of some data.



This is an empirical distribution (i.e., modeled after data). What you see is a *Kernel Density Estimate* (KDE), using Gaussian Kernels and the Sheather-Jones bandwidth.

PDF: $\mathrm{p}(x) = \frac{1}{N*h}\sum_{i=1}^{N}\phi\left(\frac{x-d_i}{h}\right),$

where ϕ is the std. normal PDF and h is the bandwidth.

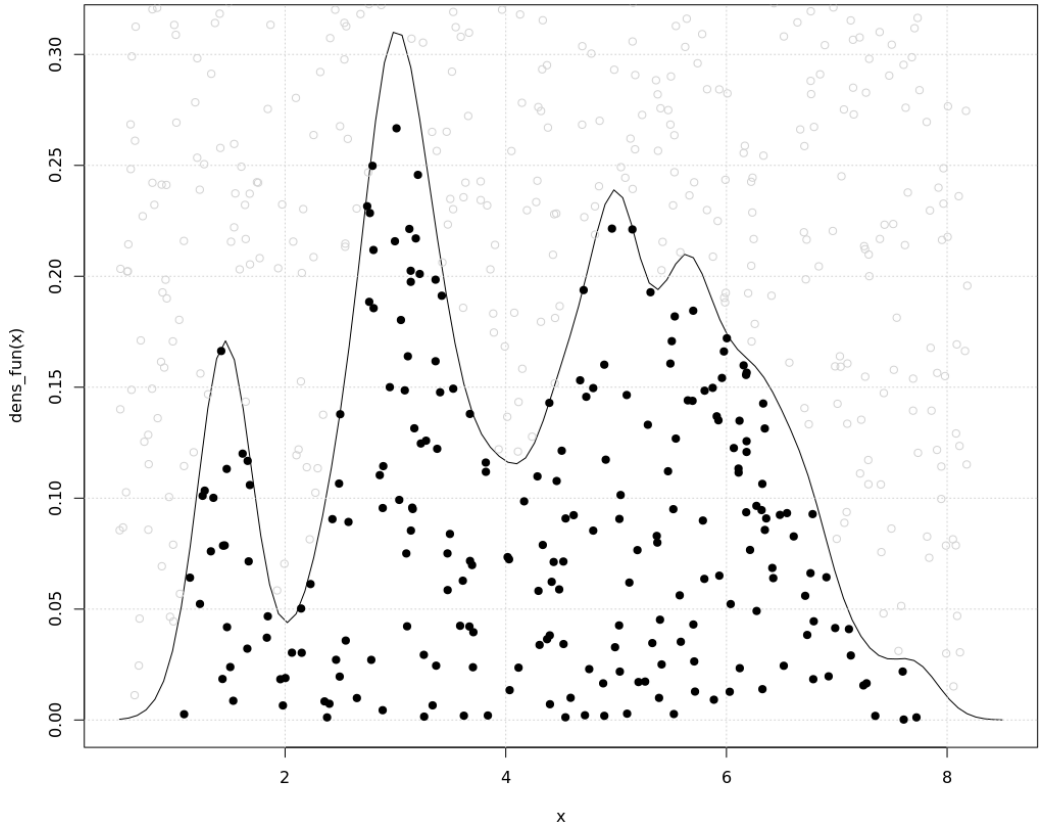
CDF: $P(x) = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\phi}\left(\frac{x-d_i}{h}\right),$

where $\boldsymbol{\Phi}$ is the std. normal CDF.

PPF: **???**

While the PPF is technically the *inverse* of the CDF, the expression for the CDF here cannot be inversed!

Rejection Sampling



Is like *uniformly* shooting darts at a $(n+1)$-dimensional dart board. For rejection sampling, we only need $p(x)$ or a function proportional to it (this is very important, because it means that we **do not** require to compute the **normalization constant**).

However, we better know the n-dimensional envelope in order not to throw too many darts that will miss (hard in practice).

RS works well in very low-dimensional problems but suffers from the **Curse of dimensionality** in higher dimensions, as it is becoming increasingly difficult to propose high-dimensional samples with $p(x) > 0$.

Importance Sampling

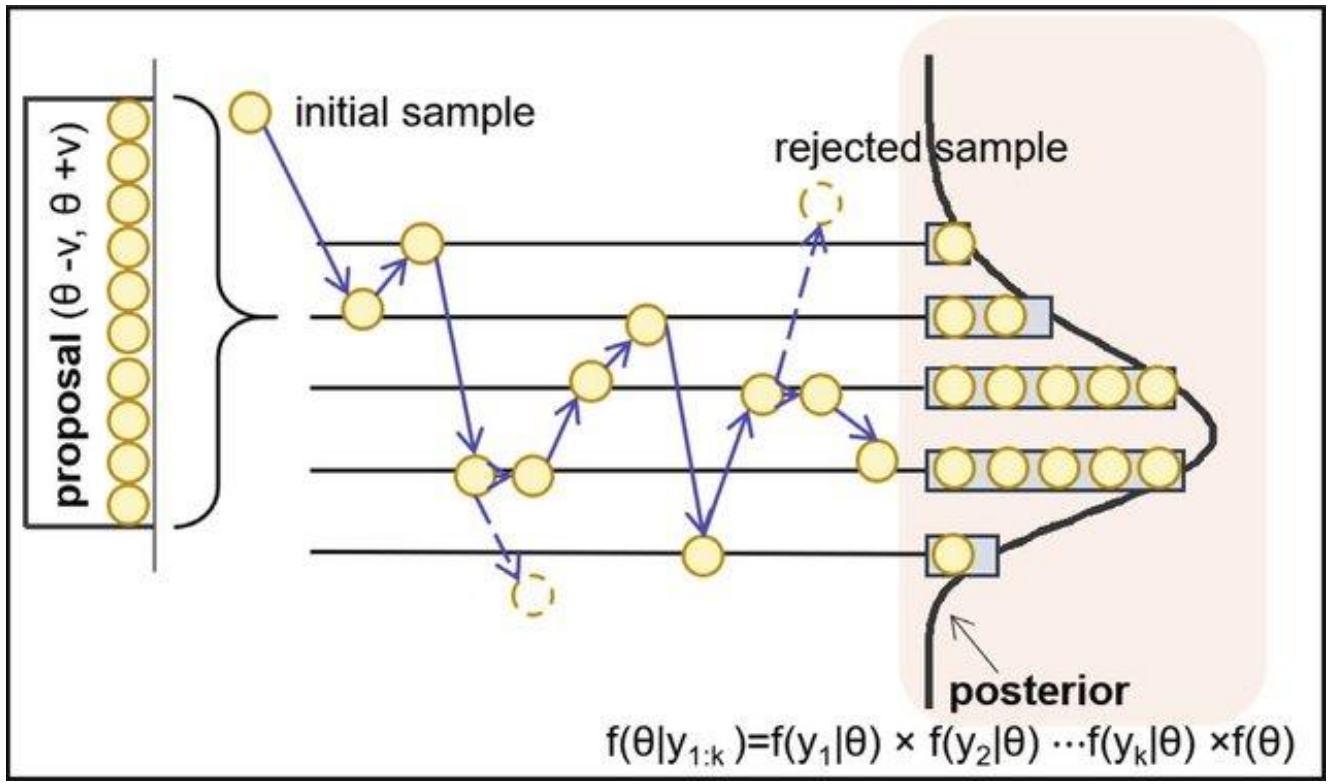
Recall $\mathbb{E}_{Z\sim p_Z(z)}[L(Z)] = \int L(z)\cdot p_Z(z)\,dz$, and its **Monte Carlo** approximation $\mathbb{E}_{Z\sim p_Z(z)}[L(Z)] \simeq \frac{1}{N}\sum_{n=1}^{N} L(z_n)$.

Now suppose it is difficult to draw samples from Z. Instead, it is straightforward to evaluate the distribution $p_Z(z)$. Similar to rejection sampling, importance sampling makes use of a **proposal distribution** $q(z)$. In contrast to $p_Z(z)$, it is easy to draw samples (**and** evaluate their likelihood) from this proposal distribution.

Note that $\text{supp}(p_Z) \subseteq \text{supp}(q)$ is a requirement for importance sampling to work correctly.

$$\begin{aligned}\mathbb{E}_{Z\sim p_Z(z)}[L(Z)] &= \int L(z)\cdot p_Z(z)\,dz \\ &= \int L(z)\cdot \frac{p_Z(z)}{q(z)}\cdot q(z)\,dz \\ &= \frac{1}{L}\sum_{l=1}^{L}\frac{p_Z\left(z^{(l)}\right)}{q\left(z^{(l)}\right)}\cdot f\left(z^{(l)}\right).\end{aligned}$$

Markov Chain Monte Carlo



(*) Image taken from Wikipedia

The idea of the Metropolis-Hastings algorithm is simple:

- Find an initial sample x with p(x)>0.
- Propose a new sample x', calculate k=p(x') / p(x).
- If k >= 1, accept the sample
- Otherwise, draw a random number u from U~[0,1]. If u > k, reject the sample.

Over time, again, if our proposals are good enough, the sample distribution will approach the population density.

There are actually clever ways to propose high-dimensional samples without being subject to the curse of dimensionality (check out the Preconditioned Crank–Nicolson algorithm).

Variational Inference (Motivation)

- As we have seen, exact computation of the posterior is often intractable, sometimes impossible (complex integrals, infeasible expectations, computationally prohibitive sampling for large data or complex models).
- While we have seen that sampling methods (rejection sampling, importance sampling, MCMC) allow us in some cases to find approximations, some drawbacks such has high variance, autocorrelated samples, scalability issues, slow mixing chains, mutlimodal distributions, etc., remain.
- Variational inference strikes a different balance – trading accuracy (no exact posterior) for computational efficiency. It is one method in a toolbox called **Approximate Inference**.
- It approximates a complex posterior distribution by using a simpler, parameterized distribution. We find an approximate solution typically by optimizing for parameters that minimize the **Kullback-Leibler** divergence between the two.

Takeaway: Variational inference is regression with a crucial twist: Instead of just predicting one number, we explicitly model the latent reason behind that number and quantify how confident we are in our prediction!

Motivating Example: Spam/Ham

- **Problem**: Spam detection often suffers from false negatives/positives. We want to attach confidence levels.
- **Limitation**: A simple probabilistic classifier may give a false impression of confidence. For example, a probability of 0.9 does not communicate our **uncertainty** about its correctness or the robustness of the model.
- Variational Approach: We would like to approximate the Bayesian posterior $p(\theta|X, y)$, where X is the message and y is the class (Spam/Ham).
  - Unobserved parameters θ have a (typically intractable) posterior: $p(\theta|X, y) \approx q_\phi(\theta)$
  - Let‘s say we want to classify a new message, $m^\star$:

$$p(y^\star = \text{Spam} \mid m^\star, X, y) = \int p(y^\star = \text{Spam} \mid m^\star, \theta) \cdot p(\theta|X, y)\, d\theta.$$

The Kullback-Leibler (*f*-)divergence

- The Kullback-Leibler (KL) divergence measures the (asymmetric) distance between two probability distributions. It is always ≥ 0. A value of 0 means the two distributions are identical.
- For two random variables Q, P with probability densities q, p, it is defined as:

$$\mathrm{KL}(Q \mid\mid P) = \mathbb{E}_{Q\sim q}\left[\log\left(\frac{q(x)}{p(x)}\right)\right] = \int_{-\infty}^{\infty} q(x) \cdot \log\left(\frac{q(x)}{p(x)}\right)\, dx\,.$$

Minimizing the KL divergence aligns our simplified inference distribution (the *variational distribution* q) more closely to the intractable posterior distribution p.

But **how** is this done in practice?

Variational Bayes

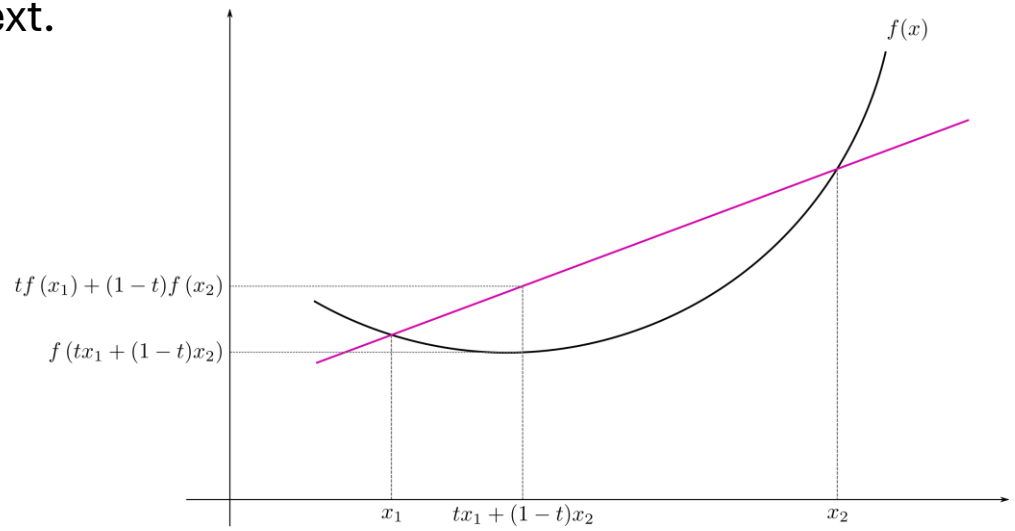
$$P(\theta|Y) = \frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}$$

In variational Bayes, we can employ something called the **Evidence Lower BOund**, or **ELBO**, for short. As we have learned, it may be intractable or impossible to estimate the **Evidence** (marginal likelihood) $P(Y)$.

By maximizing the ELBO, we raise the lower bound, bringing it closer to the true evidence, $P(Y)$.

Crucially, maximizing the ELBO also implies minimizing the KL divergence between the **variational approximation** $\boldsymbol{q(\theta)}$ and the **true** posterior $p(\theta|Y)$**.**

The ELBO is derived using **Jensen's Inequality**, we will see that next.



(*) Image taken from Wikipedia

Jensen‘s Inequality (for random variables)

For a concave function $f(\cdot)$, the inequality states that $f(\mathbb{E}[Y]) \geq \mathbb{E}[f(Y)]$ for random variables. (*)

First recall the log-**Evidence** in Bayes’ theorem. We’ll insert our approximate distribution $q(\theta)$, too:

$$\log\big(P(Y)\big) = \log\left(\int P(Y,\theta)\, d\theta\right) = \log\left(\int q(\theta) \cdot \frac{P(Y,\theta)}{q(\theta)}\, d\theta\right) = \log\left(\mathbb{E}_{q(\theta)}\left[\frac{P(Y,\theta)}{q(\theta)}\right]\right).$$

Now it becomes obvious why we have expressed the evidence as log-evidence, the logarithm is concave!

$$\log\left(\mathbb{E}_{q(\theta)}\left[\frac{P(Y,\theta)}{q(\theta)}\right]\right) \;\geq\; \mathbb{E}_{q(\theta)}\left[\log\left(\frac{P(Y,\theta)}{q(\theta)}\right)\right],$$

Giving us the key inequality:

$$\big[\log(P(Y)) = \log\big(\mathbb{E}_{\theta\sim P(\theta)}[P(Y|\theta)]\big)\big] \;\geq\; \left[\mathbb{E}_{q_\phi(\theta)}\left[\log\left(\frac{P(Y,\theta)}{q_\phi(\theta)}\right)\right] = \mathbb{E}_{q_\phi(\theta)}\big[\log(P(Y,\theta)) - \log\big(q_\phi(\theta)\big)\big]\right].$$

(*) For a convex function, the sign flips. However, we mostly use the logarithm, which is concave (upper convex).

The ELBO = $\mathbb{E}_{q_\phi(\theta)}\left[\underbrace{\log(P(Y,\theta))}_{\text{Exp. data likelihood}} - \underbrace{\log\left(q_\phi(\theta)\right)}_{\text{Entropy term}}\right] \leq \log(P(Y))$

Maximizing the ELBO is the same as minimizing the KL divergence:

$$D_{\mathrm{KL}}(q(\theta) \,||\, P(\theta|Y)) = \int q(\theta) \cdot \log\left(\frac{q(\theta)}{P(\theta|Y)}\right) d\theta = \mathbb{E}_{q(\theta)}\left[\log\left(\frac{q(\theta)}{P(\theta|Y)}\right)\right].$$

Thus, we have (notice the flip of numerator and denominator [logarithm laws]!):

$$-D_{\mathrm{KL}}(q(\theta) \,||\, P(\theta|Y)) = \mathbb{E}_{q(\theta)}\left[\log\left(\frac{P(\theta|Y)}{q(\theta)}\right)\right].$$

Now, bringing back the original log-**evidence** $\log(P(Y))$, we get:

$$\log(P(Y)) = \mathbf{ELBO}(q) + D_{\mathrm{KL}}\left(q(\theta) \,||\, P(\theta|Y)\right).$$

Mean-Field Variational Bayes

So far, we have only hinted at suitable choices for our variational distribution $q(\cdot)$. In general, **mean-field** approximation in variational inference means approximating a complex posterior distribution $p(\theta|Y)$ by *factorizing* it into simpler, **independent** factors (this is the mean-field assumption):

$$p(\theta|Y) \;\approx\; q_{\phi}(\theta) = \prod_i q_{\phi_i}(\theta_i) = \exp\left(\sum_i \log\left(q_{\phi_i}(\theta_i)\right)\right).$$

Optimization Objective (practically done using a Monte Carlo approach, next slide):

- Maximize ELBO with regard to the variational parameters ϕ, i.e., $\hat{\phi} = \phi + \eta \cdot \nabla_{\phi} \widehat{\mathrm{ELBO}}(\phi)$.

Remarks:

- The mean-field assumption makes optimization computationally efficient, but may restrict flexibility and underestimate posterior uncertainty. A much more flexible choice here are **Normalizing Flows** (soon).
- Other, richer, non-factorized approximations exist (e.g., VAE posteriors).

Maximizing ELBO (minimizing the KL divergence)

Typically, the ELBO expectation **cannot** be computed analytically because it involves some complicated distributions (such as neural-network-based, high-dimensional, or custom probability distributions).

Monte Carlo approximation of the ELBO:

1. Sample from approximate posterior distribution $q_\phi(\theta)$.
    - Direct sampling methods (e.g., if q is Gaussian: just draw normal samples): **S samples** $\{\theta^{(s)}\}_{s=1}^{S}, \forall\, \theta^{(s)} \sim q_\phi(\theta)$
    - Apply reparameterization trick to train model (next slides).

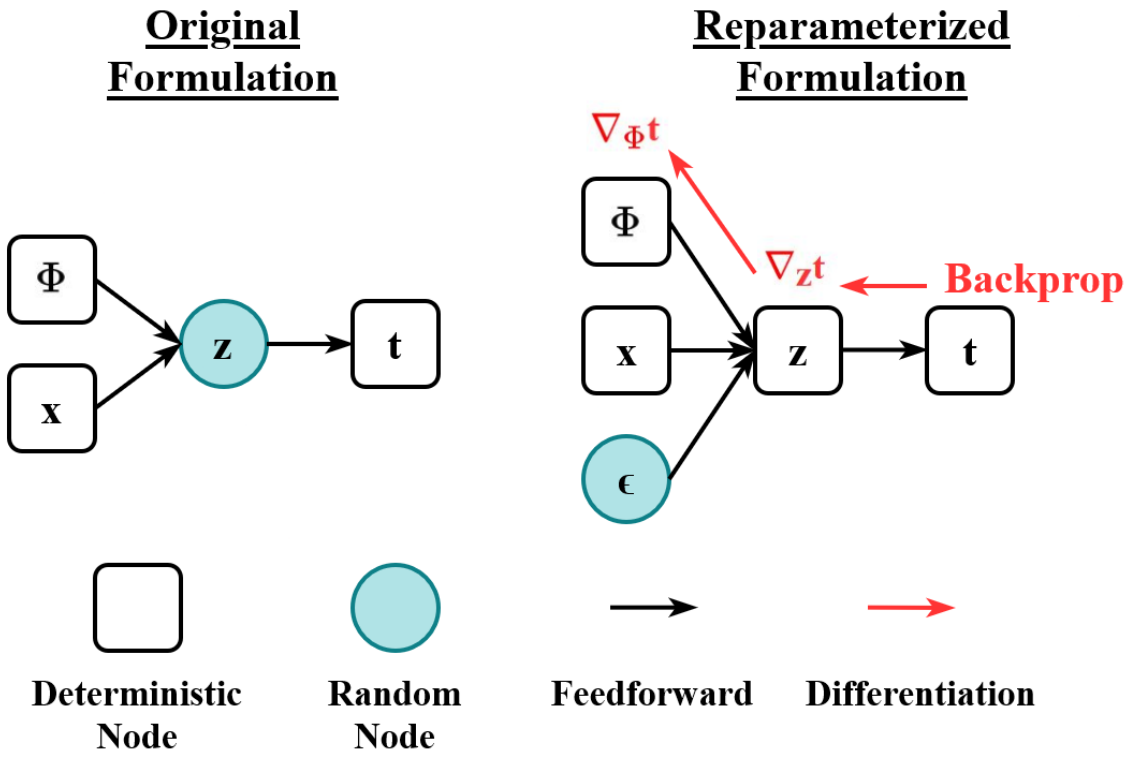
- Monte Carlo estimate of the ELBO using stochastic gradient-based optimization:
    - Using these samples, we construct a simple Monte Carlo approximation of the Expectation:

$$\mathrm{ELBO}(\phi) \approx \frac{1}{S}\sum_{s=1}^{S}\left[\log\left(P(Y,\theta^{(s)}\right) - \log\left(q_\phi\left(\theta^{(s)}\right)\right)\right]$$

– or, as KL divergence:

$$D_{KL}^{MC}\left(q_\phi(\theta)\;||\;P(\theta|Y)\right) \coloneqq \frac{1}{S}\sum_{s=1}^{S}\log\left(\frac{q_\phi\left(\theta^{(s)}\right)}{P(\theta^{(s)}|Y)}\right).$$

The Reparameterization Trick



Key Idea: Separate randomness from model parameters by injecting noise separately into the computational graph as a new input.

Suppose our model uses a Monte Carlo approximation that draws samples $z \sim \mathcal{N}(\mu, \sigma^2)$. If we wanted to optimize the parameters μ, σ, then we would move out the randomness like so:

- Sample standard normal noise: $\epsilon \sim \mathcal{N}(0,1)$ externally.
- Define $z = \mu + \sigma \cdot \epsilon$, which is fully differentiable w.r.t. μ, σ.

Sampling is non-differentiable because it is inherently a discontinuous, stochastic operation. If we were to sample directly, we would introduce random choices that would break the smooth connection between parameters and outputs, making it impossible for gradients to flow backwards through the sampling step.

(*) Image taken from Wikipedia

Assignment

Reading:

- Cursory reading of chapter 20 (Deep Learning)
- **Video** (15m)**:** ”Normalizing Flow Mechanics for Anomaly Detection in Industrial Parts through Visual Inspection”, https://play.lnu.se/media/t/0_zuy2rkxn
- “Auto-Encoding Variational Bayes”, https://doi.org/10.48550/arXiv.1312.6114
- “Variational Inference with Normalizing Flows”, http://proceedings.mlr.press/v37/rezende15.html

Practical:

- Implement notebook “Foundations of Bayes’ theorem” (20%)
- Implement notebook “Variational Inference: Spam Detection” (80%)

Deadline: Reading (before next lecture); Practical (Wed., May 21st, 23:59).

Next Lecture (From Spam Detection to Generative AI)

In the second part, we will look more closely at generative AI through the lens of Variational Auto Encoders (**VAEs**) and discrete Normalizing Flows (**NFs**).

- So far, we have effectively created statistical models describing how spam- and ham-messages look („spam/ham distribution“). We have learned approximate parameters of these distributions efficiently, while explicitly modeling uncertainty.
- To generate new data, we typically introduce latent, lower-dimensional variables that will learn to describe underlying factors, such as „writing style“, „topic/theme“, „promotion intensity“, etc.
- A generative model captures these variables and learns how messges were „generated“ in a statistical sense.
- Variational inference then helps us to create both, **realistic** and **varied** new messages.