Approximate Inference in Bayesian Statistics

Maxim Panov

Skoltech

December 2021

Skoltech

Outline

Uncertainty Estimation in Machine Learning

Approximate Bayesian Inference

Markov Chain Monte Carlo

Variational Inference

Outline

Uncertainty Estimation in Machine Learning

Approximate Bayesian Inference

Markov Chain Monte Carlo

Variational Inference

Regression and Uncertainty Estimation

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Some machine learning models along with

- approximation

  $$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

  can provide

- uncertainty estimation

  $$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$

Regression and Uncertainty Estimation

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Some machine learning models along with

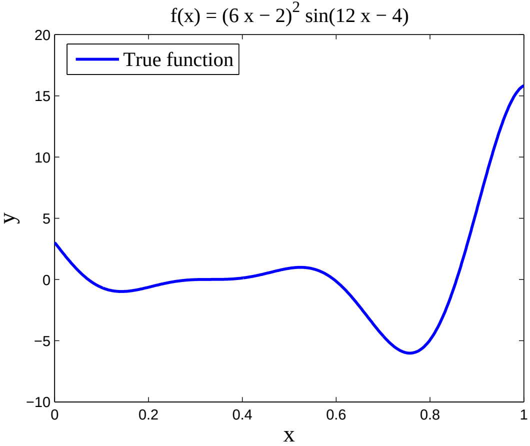
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$



Regression and Uncertainty Estimation

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Some machine learning models along with

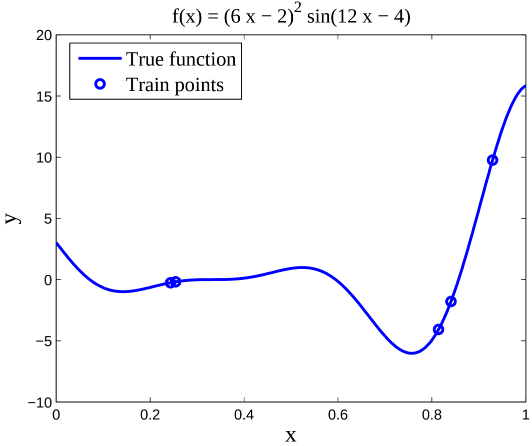
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

  can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$



Skoltech
Skolkovo Institute of Science and Technology

Regression and Uncertainty Estimation

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Some machine learning models along with

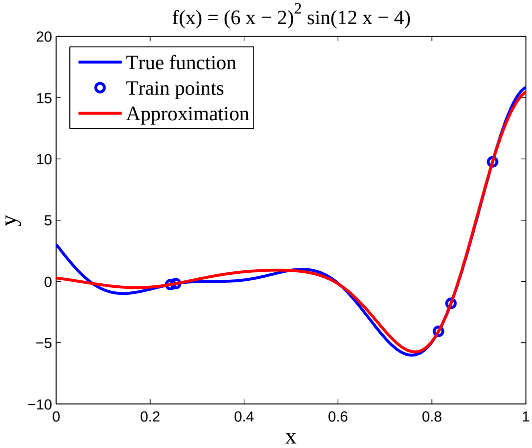
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

  can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$



Regression and Uncertainty Estimation

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Some machine learning models along with

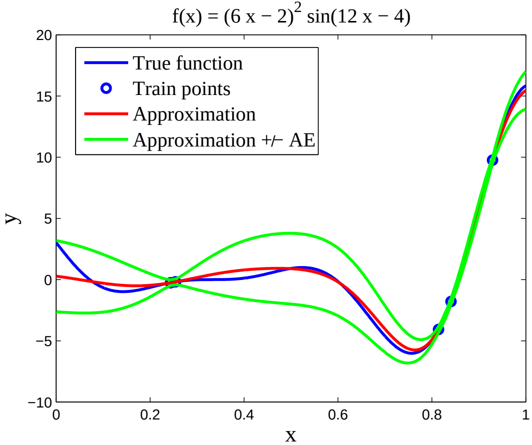
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$



Regression and Uncertainty Estimation

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Some machine learning models along with

- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}(\hat{f}(\mathbf{x}) - f(\mathbf{x}))^2.$$

Use cases:

- Possibility of rejection to predict:
  - out-of-distribution data detection
  - adversarial examples detection
- Active learning
- Bayesian optimization

Active Learning (also known as *Adaptive Design of Experiments*)

- In many applications labelled (annotated) data is very limited.
- Unlabelled data is usually widely available.
- Labelling (annotation) is often expensive.
- Thus, a clever choice of points to annotate is needed.
- **Active learning:** use machine learning model to select the points to annotate.

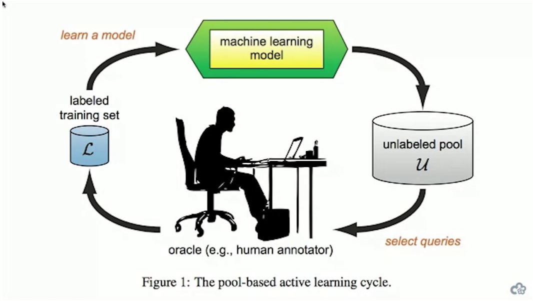
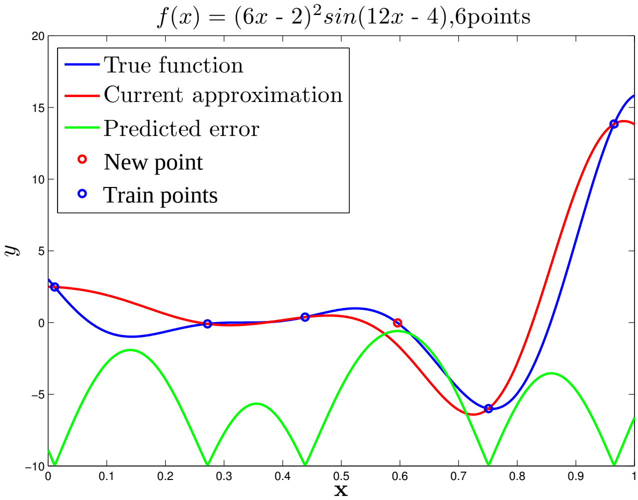


Figure 1: The pool-based active learning cycle.

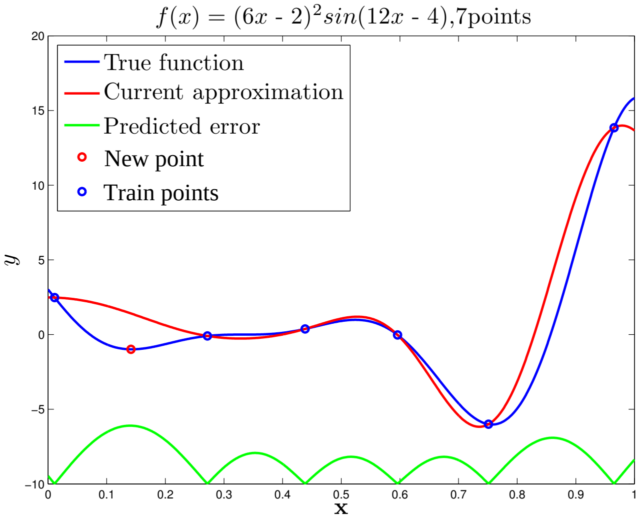
*Figure is taken from Lukas Biewald Youtube lecture

Applications: industrial design, chemoinformatics, material design, human annotation (NLP, images), . . .

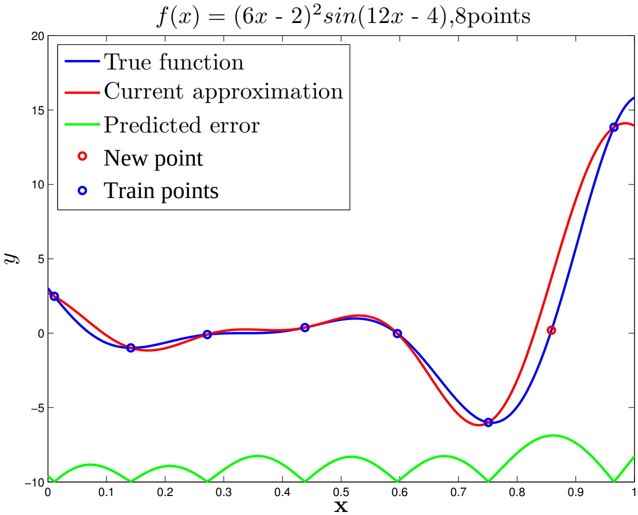
Active Learning: Example



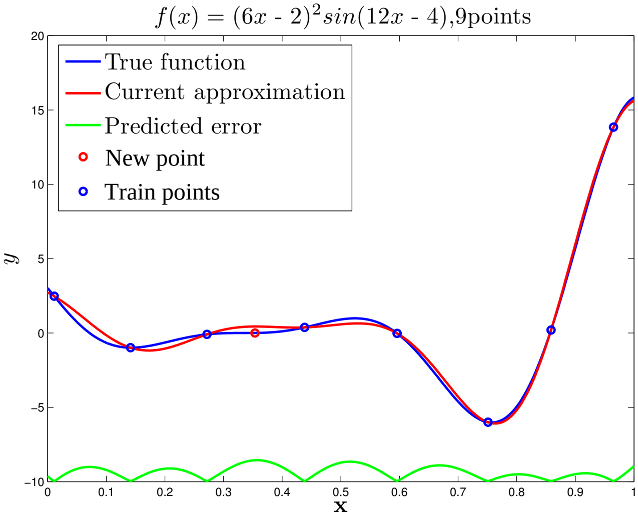
Active Learning: Example



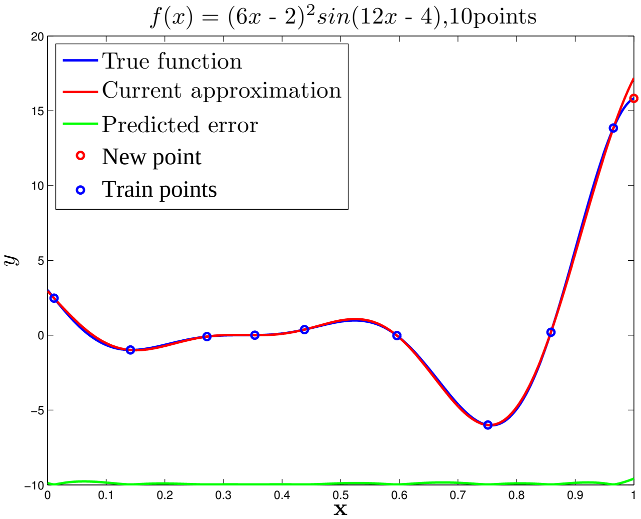
Active Learning: Example



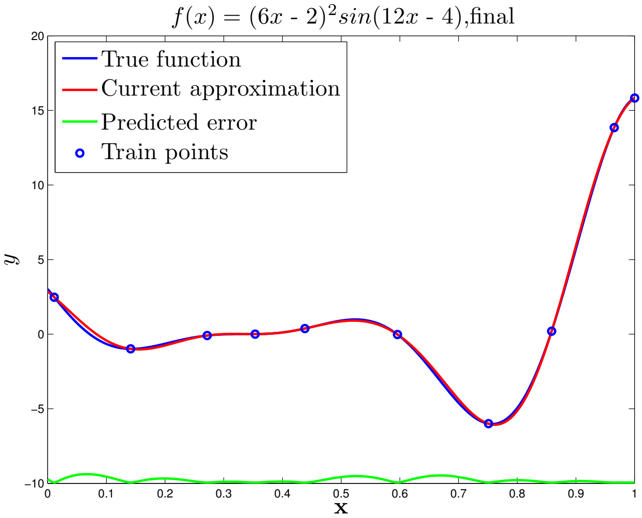
Active Learning: Example



Active Learning: Example



Active Learning: Example



Machine Learning Models and Uncertainty Estimation

- General approaches:
  - Analytic statistical approaches (variance estimates and confidence intervals based on CLT);
  - Bootstrap.
- Bayesian inference
- Model-specific approaches:
  - Gaussian processes for regression and classification;
  - Neural networks with variance-predicting subnetwork;
  - Decision trees variance estimation at leaves.

Outline

Uncertainty Estimation in Machine Learning

Approximate Bayesian Inference

Markov Chain Monte Carlo

Variational Inference

Bayesian Approach to Machine Learning

Consider a probabilistic model:

$$p(y \mid \mathbf{x}, \mathbf{w}),$$

where

- $\mathbf{x}$ is a model input;
- $\mathbf{w}$ is a vector of model parameters (i.e., linear regression weights).

Let us be given the dataset $\mathcal{D}_n = \{\mathbf{x}_i, y_i\}_{i=1}^n$. Then the likelihood of the data reads as

$$p(\mathcal{D}_n \mid \mathbf{w}) = \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}).$$

In Bayesian approach, $\mathbf{w}$ is assumed to be a random variable with some prior distribution:

$$\mathbf{w} \sim p(\mathbf{w}).$$

Bayesian Inference Problem

In Bayesian problems we are interested in posterior distribution of latent variables:

$$p(\mathbf{w} \mid \mathcal{D}_n) = \frac{p(\mathcal{D}_n \mid \mathbf{w})\ p(\mathbf{w})}{p(\mathcal{D}_n)},$$

where $\mathcal{D}_n$ – observed data, $\mathbf{w}$ – latent (unobserved) variables.

Posterior allows **to reason about the uncertainties** in latent variables.

The following distributions are involved:

- $p(\mathbf{w} \mid \mathcal{D}_n)$ – posterior (our updated knowledge about $\mathbf{w}$ after we have observed data $\mathcal{D}_n$);
- $p(\mathbf{w})$ – prior (our knowledge about $\mathbf{w}$ before we have observed data $\mathcal{D}_n$);
- $p(\mathcal{D}_n \mid \mathbf{w})$ – likelihood (probability of data $\mathcal{D}_n$ given latent variables $\mathbf{w}$);
- $p(\mathcal{D}_n)$ – normalizing constant for $p(\mathbf{w} \mid \mathcal{D}_n)$ to be a proper distribution.

Bayesian Model Averaging

Let us be given the dataset $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$.
We can compute a posterior distribution:

$$p(\mathbf{w} \mid D) = \frac{p(D \mid \mathbf{w})p(\mathbf{w})}{\int p(D \mid \mathbf{w})p(\mathbf{w})d\mathbf{w}}$$

The standard approach starts from considering the posterior predictive distribution

$$p(y \mid \mathbf{x}, D) = \int p(y \mid \mathbf{x}, \mathbf{w})\; p(\mathbf{w} \mid D)\; d\mathbf{w} = \mathbb{E}_{p(\mathbf{w}|D)}\; p(y \mid \mathbf{x}, \mathbf{w}).$$

If we can sample from posterior, then we can naturally perform Bayesian model averaging

$$\mathbb{E}_{p(\mathbf{w}|D)}\; p(y \mid \mathbf{x}, \mathbf{w}) \approx \bar{p}_T(y \mid \mathbf{x}, D) = \frac{1}{T}\sum_{t=1}^{T} p(y \mid \mathbf{x}, \mathbf{w}_t),$$

where $\mathbf{w}_t \sim p(\mathbf{w} \mid D),\; t = 1, \ldots, T$.

Approximate Bayesian Inference Problem

Posterior distribution:

$$p(\mathbf{w} \mid \mathcal{D}_n) = \frac{p(\mathcal{D}_n \mid \mathbf{w})\; p(\mathbf{w})}{p(\mathcal{D}_n)},$$

The problem with exact posterior computation comes from the denominator:

$$p(\mathcal{D}_n) = \int p(\mathcal{D}_n, \mathbf{w}) d\mathbf{w}$$

as

- generally, the complexity of this integral computation grows exponentially with dimensionality;
- an exception is the case of conjugate pairs of prior and likelihood.

That is why in practice we use approximate Bayesian inference:

- MCMC (Markov Chain Monte Carlo);
- Variational Inference.

Outline

Uncertainty Estimation in Machine Learning

Approximate Bayesian Inference

Markov Chain Monte Carlo

Variational Inference

Monte-Carlo Integration

This Lecture

We aim at computing expectation

$$\pi(f) := \mathbb{E}[f(X)] = \int_{\mathsf{X}} f(x)\pi(x)\,\mathrm{d}x, \quad f \in \mathrm{L}_2(\pi).$$

We discuss,

- Monte-Carlo method
- Rejection sampling
- Importance sampling
- MCMC

Monte-Carlo Method

- Get an i.i.d. sample $(X_k)_{k=0}^{\infty}$ from π, estimate $\pi(f)$ by

$$\pi_n(f) := \frac{1}{n}\sum_{k=0}^{n-1} f(X_k),$$

- Kolmogorov's strong law of large numbers: with probability 1

$$\lim_{n\to\infty} \pi_n(f) = \mathbb{E}[f(X_0)] = \pi(f).$$

- **Advantage** over deterministic integration: MC positions the integration grid (samples) in regions of high probability.
- **Disadvantage:** when $\pi(x)$ has standard form, e.g. Gaussian, it is straightforward to sample from it using easily available routines. However, when this is not the case, we need to introduce more sophisticated techniques.

Rejection Sampling

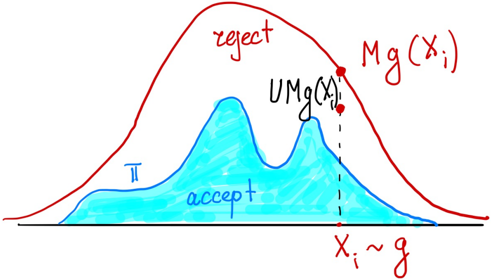
- Sample from a distribution π, which is known up to a proportionality constant, by sampling from another easy-to-sample proposal distribution g that satisfies

$$\pi(x) \leq Mg(x), M < \infty.$$

- Algorithm:
  Set $k = 0$;
  Repeat until $k = n - 1$
  1. Sample $X_i \sim g$ and independent $U \sim \mathrm{Uniform}[0, 1]$;
  2. Accept X_i and set $i := i + 1$, if

$$U < \frac{\pi(X_i)}{Mg(X_i)}.$$

Otherwise, reject.



Rejection Sampling

- **Advantage:** simple.
- Disadvantage: impractical in high-dimensional scenarios.

  It is not always possible to bound $\pi(x)/g(x)$ with a reasonable constant M over the whole space X. If M is too large,

$$\mathbb{P}(X_i \text{ accepted}) = \mathbb{P}\left(U < \frac{\pi(X_i)}{Mg(X_i)}\right) = \mathbb{E}\left[\mathbb{P}\left(U < \frac{\pi(X_i)}{Mg(X_i)}\middle| X_i\right)\right]$$
$$= \mathbb{E}\left[\frac{\pi(X_i)}{Mg(X_i)}\right] = \int_{\mathsf{X}} \frac{\pi(x)}{Mg(x)} g(x) \mathrm{d}x = \frac{1}{M}.$$

  will be too small (here we also assume $g(x) > 0, x \in \mathsf{X}$).

Importance Sampling

- Make change of measure: replace $\pi(x)$ by another easy-to-sample proposal distribution $g(x)$:

$$\pi(f) = \int_{\mathsf{X}} f(x)\pi(x)\mathrm{d}x = \int_{\mathsf{X}} f(x)w(x)g(x)\mathrm{d}x,$$

where $w(x)$ – importance weight:

$$w(x) := \frac{\pi(x)}{g(x)}.$$

- Replace $\pi_n(f)$ by $\bar{\pi}_n(f)$,

$$\bar{\pi}_n(f) := \frac{1}{n}\sum_{k=0}^{n-1} f(X_i)w(X_i),$$

where $X_i \sim g$.

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) – a family of generic methods, which have theoretical guarantees under some mild conditions of receiving **exact samples** from the posterior distribution.

Idea:

- design a Markov chain $(X_k)_{k \in N}$, whose stationary distribution is

$$\pi(x)$$

  known up to a normalization constant;
- it means, that starting from a sample from a prior $p(x)$, distribution of X_K converges to the target $\pi(x)$ as K goes to infinity.

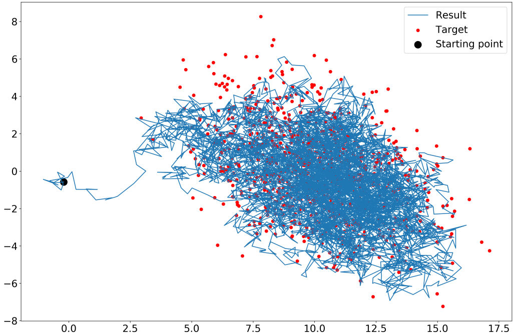
$$\pi_n(f) = \frac{1}{n} \sum_{k=0}^{n-1} f(X_k)$$

Markov Chain Monte Carlo

Metropolis Hastings (MH) algorithm is an option:

- Draw a proposal y from some transition density $q(y, x)$.
- Accept / Reject the proposal with probability

$$\alpha(x, y) = 1 \wedge \frac{\pi(y)\ q(x, y)}{\pi(x)\ q(y, x)}.$$



Example: Random Walk MH

Take $q(x,y) = \overline{q}(y-x)$, where $\overline{q}(x) = \overline{q}(-x)$. Then

$$Y_{k+1} = X_k + Z_{k+1}, \quad Z_{k+1} \sim \overline{q}.$$

In this case

$$\alpha(x,y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}.$$

Markov Chain Monte Carlo

Many recent advances for efficient MCMC methods, using Langevin dynamics, Hamiltonian Monte Carlo, ...

Advantages:

1. Generic (does not require introduction of a family of distributions).
2. Theoretical guarantees for fairly general cases.

Disadvantages:

- The rate of convergence could be really slow.
- You never know in advance how long to run a chain to receive decent samples from it.

Outline

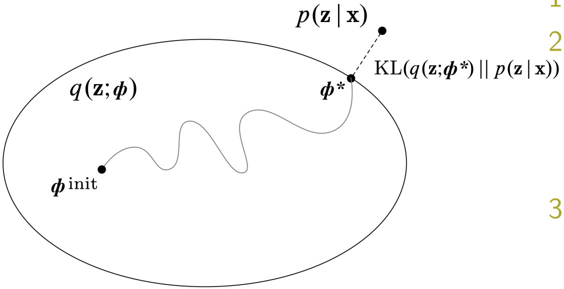
Uncertainty Estimation in Machine Learning

Approximate Bayesian Inference

Markov Chain Monte Carlo

Variational Inference

Variational Inference


$p(\mathbf{z} \mid \mathbf{x})$
$\mathrm{KL}(q(\mathbf{z};\boldsymbol{\phi}^*) \,||\, p(\mathbf{z} \mid \mathbf{x}))$
$q(\mathbf{z};\boldsymbol{\phi})$
$\boldsymbol{\phi}^*$
$\boldsymbol{\phi}$ init

1. VI turns inference into optimization.
2. Introduce a variational family of distributions over the latent variables:

$$\mathcal{Q} = \{q_\phi(z), \quad \phi \in \Phi\}.$$

3. Fit the variational parameters ϕ to become close (usually in KL) to the exact posterior.

Source: David Blei, Rajesh Ranganath, Shakir Mohamed: Variational Inference:Foundations and Modern Methods. NIPS 2016, December 5, 2016.

Evidence Lower Bound (ELBO)

Typically, in VI methods we are interested in minimizing Kullback-Leibler divergence between variational distribution and the target:

$$\mathsf{KL}(q_\phi(z) \| p(z \mid x)) = \int q_\phi(z) \log \frac{q_\phi(z)}{p(z \mid x)}.$$

But in practice it is convenient to consider the equivalent task, which is called ELBO:

$$\mathcal{L}(x; \phi) = \int q_\phi(z) \log \frac{p(z \mid x)}{q_\phi(z)} dz$$

and can be interpreted as a lower bound for the log-likelihood:

$$\log p(x) \geq \mathcal{L}(x; \phi).$$

Evidence Lower Bound (ELBO)

- A key observation here is that to compute both these formulas we still need to have $p(z \mid x)$.
- And as we discussed, it is complicated because of the normalizing constant $p(x)$.
- It can be shown, that for optimization **we do not need $p(z \mid x)$ to be normalized**:

$$\mathcal{L}(x;\phi) = \int q_\phi(z) \log \frac{p(z \mid x)}{q_\phi(z)} dz = \int q_\phi(z) \log \frac{p(x,z)}{p(x)\ q_\phi(z)} dz$$

$$= \int q_\phi(z) \log \frac{p(x,z)}{q_\phi(z)} dz - C.$$

  as $p(z,x) = p(x \mid z)p(z)$ and **the constant $C = \log p(x)$ does not depend on ϕ**.

- The integral is usually computed using MC-estimate.

Reparameterization Trick

Optimization ϕ of

$$\mathcal{L}(x;\phi) = \int \log\left(\frac{p(x,z)}{q_\phi(z)}\right) q_\phi(z)\mathrm{d}z\,.$$

- If you have an integral

  $$\phi \mapsto \int h(x,z) q_\phi(z)\mathrm{d}z,$$

  its gradient may be written as

  $$\int h(x,z) q_\phi(z) \nabla \log q_\phi(z)\mathrm{d}z\,,$$

- Monte Carlo estimation

  $$M^{-1}\sum_{i=1}^{M} h(x,Z_i)\nabla \log q_\phi(Z_i)\,, \quad Z_i \sim q_\phi(\cdot)\,.$$

Problem: the variance of the vanilla unbiased estimator of this quantity is generally very high!

Reparameterization Trick

- **Reparameterization trick:** Assume there exists a diffeomorphism V_ϕ and a distribution g easy to sample from such that

$$\epsilon \sim g, \quad z = V_\phi(\epsilon) \sim q_\phi(\cdot)\,.$$

- Using the reparameterization, the ELBO writes

$$\mathcal{L}(x;\phi) = \int \log\left(\frac{p(V_\phi(\epsilon), x)}{q_\phi(V_\phi(\epsilon))}\right) g(\epsilon)\mathrm{d}\epsilon\,.$$

- Then you can do a standard Monte-Carlo estimate.

Variational Inference

Advantages:

1. Compared to MCMC, scales faster to high dimensions.
2. Allows us to **leverage complexity and accuracy**, selecting a family of distribution.
3. Allows efficient mini-batch optimization, which scales to massive data.

Disadvantages:

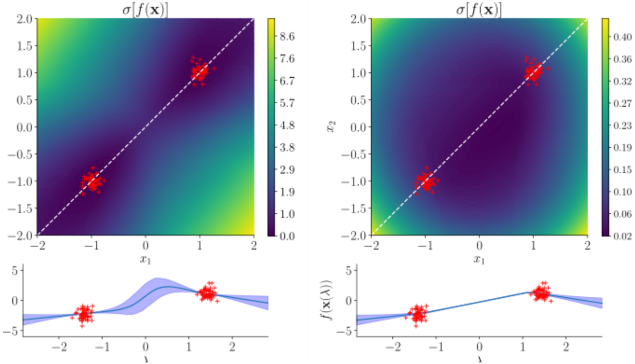
- By construction it **introduces bias**, since we are considering only a family of parametric distributions.
- It means, we in principle cannot obtain exact approximation of the posterior.

Bayesian Neural Networks

The framework above can be well-applied to ML and NNs in particular:

- $p_\theta(y \mid \mathbf{x})$ is the likelihood with θ being parameters of the model.
- $q_\phi(z)$ is usually chosen to be some simple form, i.e. factorized Gaussian.

The main issue: poor approximation of the posterior:



Foong, A. Y et all (2019). On the Expressiveness of Approximate Inference in Bayesian Neural Networks. arXiv preprint arXiv:1909.00719.

Mean Field Gaussian Variational Inference

The most straightforward choice of the Variational family is fully factorized Gaussian.

In this case,

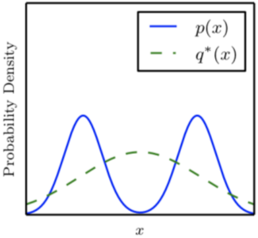
$$q_\phi = \mathcal{N}(\mu_\phi, diag\{\sigma_\phi^2\}),$$

where each μ_ϕ and σ_ϕ^2 are vectors of size $\dim(z)$.

Main advantage: there are only $2\dim(z)$ learnable parameters.

The **disadvantage** is in its expressiveness:

$q^* = \text{argmin}_q D_{\text{KL}}(p\|q)$



Normalizing Flows

- Recently, new deterministic parametric models, **Normalizing Flows**, were suggested to transform one probability density to another.
- Formally, **Normalizing Flow:**
  - invertible parametrized transformation T_ϕ;
  - both T_ϕ and T_ϕ^{-1} are differentiable.
- Moreover, determinant J_{T_ϕ} of Jacobian of T_ϕ should be easy to compute.
- Given a sample from a simple density $u \sim g(u)$, we deterministically transform it as

$$z = T_\phi(u).$$

- Using *change of variables* formula we can compute the resulting density:

$$q_\phi(z) = g(u)|J_{T_\phi}(u)|^{-1}.$$

Normalizing Flows

In case of K flows, resulting density is expressed as follows:

$$\log q_\phi^K(z) = \log g(u_0) - \sum_{i=1}^{K} \log |J_{T_i}(u_i)|.$$

This log-density is then used in the ELBO, using Monte Carlo estimation of the integral:

$$\mathcal{L}(x;\phi) = \int q_\phi^K(z) \log \frac{p(x,z)}{q_\phi(z)} dz,$$

where

$$\log \frac{p(x,z)}{q_\phi(z)} = \log p(x,z) - \log g(u_0) + \sum_{i=1}^{K} \log |J_{T_i}(u_i)|.$$

Normalizing Flows

- We can compose these transformations, and use them in variational inference framework, minimizing KL between resulting distribution $q_\phi^K(z)$ and target.
- Even simple transformation stacked on top of each other are capable to transform a simple density to a complex one:

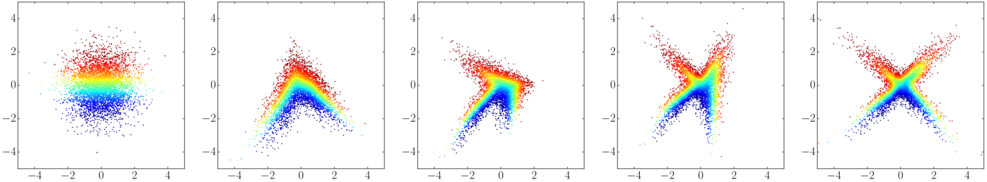


Figure 1: Example of a 4-step flow transforming samples from a standard-normal base density to a cross-shaped target density.

Source: Papamakarios et al., 2019, Normalizing flows for probabilistic modeling and inference.

Examples of Normalizing Flows

We briefly consider examples of the most known normalizing flows. Some of them are simple, yet very expressive.

1. **Planar and Radial Flows**. The pioneering work on normalizing flows were planar and radial flows were introduced. Simple form transitions, parametrized by learnable parameters.
2. **RealNVP**. The idea is to keep some part of coordinates unchanged, while transform the rest using an affine transformation. Parameters of affine transformation are usually arbitrarily complex neural networks.
3. **IAF**. Invertible Autoregressive Flow (IAF) also performs affine transformation, but in contrast to RealNVP, it changes all coordinates, without keeping unchanged. To compute determinant of Jacobian effectively, we introduce a restriction, that neural networks which perform Affine transformation should be autoregressive.

D. Rezende et al., Variational Inference with Normalizing Flows

L. Dinh et al., Density estimation using Real NVP

D. Kingma et al., Improved Variational Inference with Inverse Autoregressive Flow

Thank you for your attention![1]

[1] Slides are partially based on the material provided by Alexey Naumov.