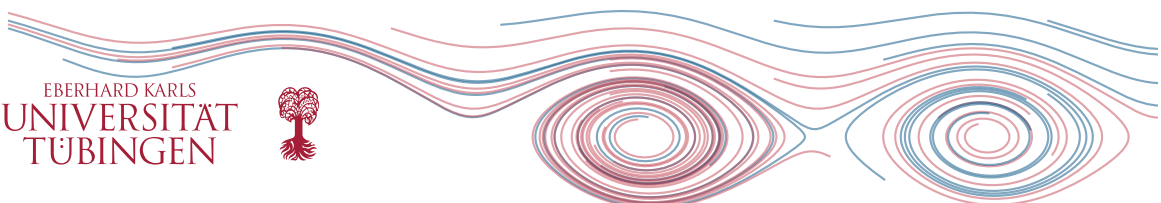
Bayesian Numerical Integration with Neural Networks

Katharina Ott

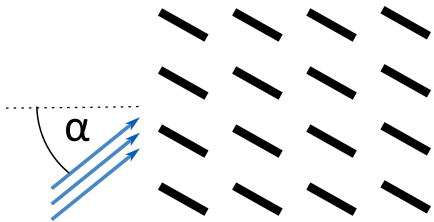
Cologne

10 April 2024



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Wind farm simulation



- f: energy output of wind farm
- π: Distribution over wind direction α

Expected energy output:

$$\Pi[f] = \int f(\alpha)\pi(\alpha)d\alpha$$

Numerical integration: Monte Carlo sampling

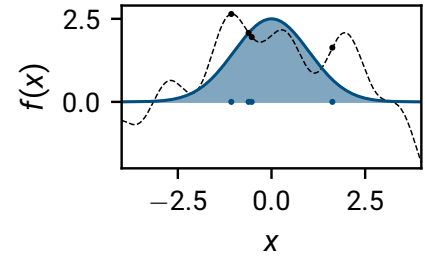
π and f are cheap

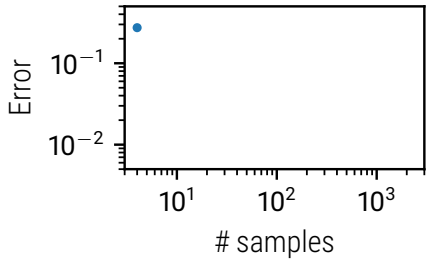
$$\Pi[f] = \int f(\alpha)\pi(\alpha)d\alpha$$

Assumption: We can sample from π

$$\Pi[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(\alpha_i), \quad \alpha_i \sim \pi(\alpha)$$

- extremly simple
- extensions: MCMC sampling





Numerical integration: Monte Carlo sampling

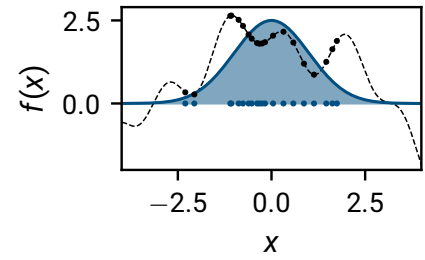
π and f are cheap

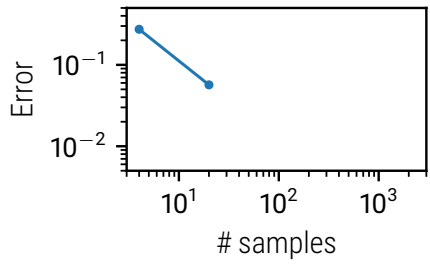
$$\Pi[f] = \int f(\alpha)\pi(\alpha)d\alpha$$

Assumption: We can sample from π

$$\Pi[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(\alpha_i), \quad \alpha_i \sim \pi(\alpha)$$

- extremly simple
- extensions: MCMC sampling





Numerical integration: Monte Carlo sampling

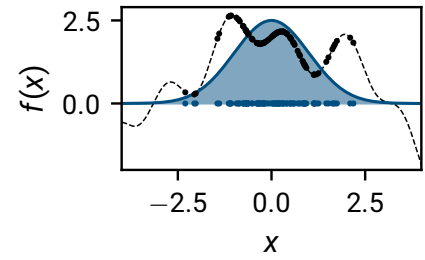
π and f are cheap

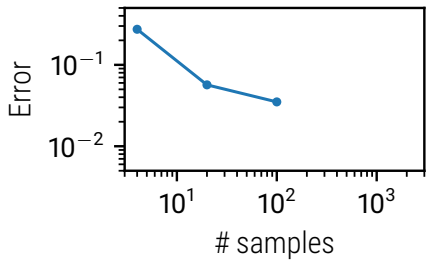
$$\Pi[f] = \int f(\alpha)\pi(\alpha)d\alpha$$

Assumption: We can sample from π

$$\Pi[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(\alpha_i), \quad \alpha_i \sim \pi(\alpha)$$

- extremly simple
- extensions: MCMC sampling





Numerical integration: Monte Carlo sampling

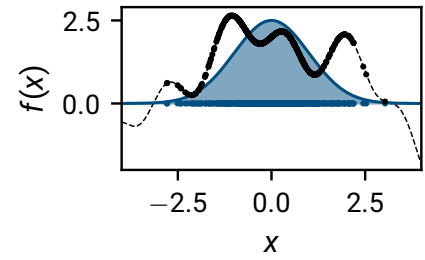
π and f are cheap

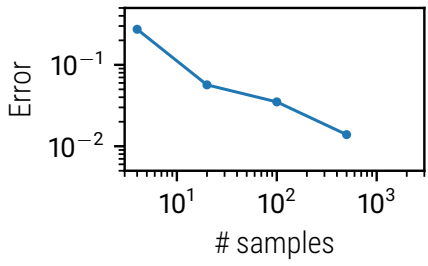
$$\Pi[f] = \int f(\alpha)\pi(\alpha)d\alpha$$

Assumption: We can sample from π

$$\Pi[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(\alpha_i), \quad \alpha_i \sim \pi(\alpha)$$

- extremly simple
- extensions: MCMC sampling





Numerical integration: Monte Carlo sampling

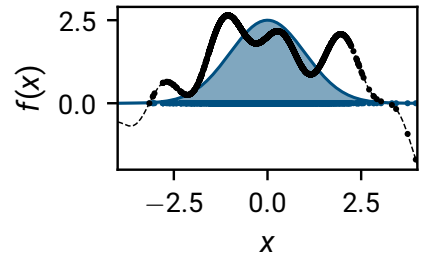
π and f are cheap

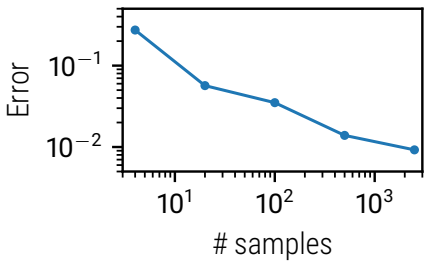
$$\Pi[f] = \int f(\alpha)\pi(\alpha)d\alpha$$

Assumption: We can sample from π

$$\Pi[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(\alpha_i), \quad \alpha_i \sim \pi(\alpha)$$

- extremly simple
- extensions: MCMC sampling





Numerical integration: Bayesian Quadrature

What if π or $\boldsymbol{f}$ are **expensive**?

- When is our approximation good enough?
- Use knowledge about function $\boldsymbol{f} \rightarrow$ Use GP as a prior for $\boldsymbol{f}$

Numerical integration: Bayesian Quadrature

Quick recap (+ my notation):
Gaussian process $g \sim \mathcal{GP}(m, k)$
Data $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$
Compute predictive posterior $g \mid \mathcal{D} = \mathcal{GP}(m_p, k_p)$,

$$m_p(x) = m(x) + k_X(x)^T \left(k_{XX} + \sigma^2 I\right)^{-1} (Y - m_X)$$
$$k_p(x, x') = k(x, x') - k_X(x)^T \left(k_{XX} + \sigma^2 I\right)^{-1} k_X(x)$$

Numerical integration: Bayesian Quadrature

Prior $g \sim \mathcal{GP}(m, k)$

$$\Lambda = \int_{\mathcal{X}} g(x)\pi(x)dx$$

Note $\Lambda \sim \mathcal{N}(\mu, \sigma)$

A. O'Hagan. Bayes–hermite quadrature. *Journal of Statisti-cal Planning and Inference*, 1991
P. Diaconis. Bayesian numerical analysis. *Statistical Deci-sion Theory and Related Topics IV*, 1988
F-X. Briol et al. Probabilistic integration: a role in statistical computation? *Statistical Science*, 2019

Numerical integration: Bayesian Quadrature

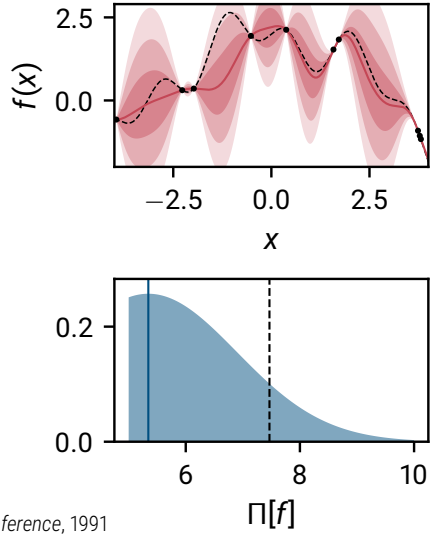
Prior $g \sim \mathcal{GP}(m, k)$

$$\Lambda = \int_{\mathcal{X}} g(x)\pi(x)dx$$

Note $\Lambda \sim \mathcal{N}(\mu, \sigma)$
Conditioning the GP on observations $\Lambda \mid \mathcal{D} \sim \mathcal{N}(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}})$

$$\mu_{\mathcal{D}} = \int_{\mathcal{X}} m_p(x)\pi(x)dx$$

$$\sigma_{\mathcal{D}} = \int_{\mathcal{X}} \int_{\mathcal{X}} k_p(x, x')\pi(x)\pi(x')dxdx'$$



A. O'Hagan. Bayes–hermite quadrature. *Journal of Statisti-cal Planning and Inference*, 1991
P. Diaconis. Bayesian numerical analysis. *Statistical Deci-sion Theory and Related Topics IV*, 1988
F-X. Briol et al. Probabilistic integration: a role in statistical computation? *Statistical Science*, 2019

Numerical integration: Bayesian Quadrature

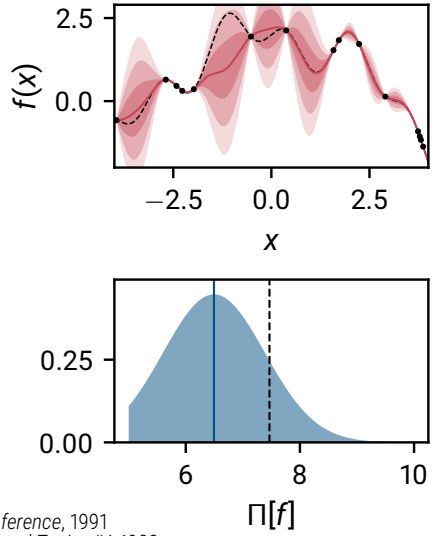
Prior $g \sim \mathcal{GP}(m, k)$

$$\Lambda = \int_{\mathcal{X}} g(x)\pi(x)dx$$

Note $\Lambda \sim \mathcal{N}(\mu, \sigma)$
Conditioning the GP on observations $\Lambda \mid \mathcal{D} \sim \mathcal{N}(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}})$

$$\mu_{\mathcal{D}} = \int_{\mathcal{X}} m_p(x)\pi(x)dx$$

$$\sigma_{\mathcal{D}} = \int_{\mathcal{X}} \int_{\mathcal{X}} k_p(x, x')\pi(x)\pi(x')dxdx'$$



A. O'Hagan. Bayes–hermite quadrature. *Journal of Statisti-cal Planning and Inference*, 1991
P. Diaconis. Bayesian numerical analysis. *Statistical Deci-sion Theory and Related Topics IV*, 1988
F-X. Briol et al. Probabilistic integration: a role in statistical computation? *Statistical Science*, 2019

Numerical integration: Bayesian Quadrature

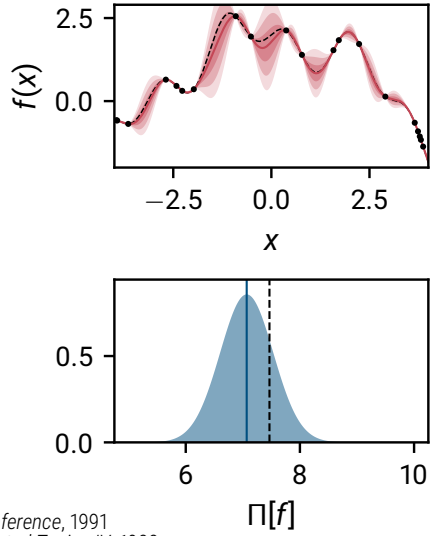
Prior $g \sim \mathcal{GP}(m, k)$

$$\Lambda = \int_{\mathcal{X}} g(x)\pi(x)dx$$

Note $\Lambda \sim \mathcal{N}(\mu, \sigma)$
Conditioning the GP on observations $\Lambda \mid \mathcal{D} \sim \mathcal{N}(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}})$

$$\mu_{\mathcal{D}} = \int_{\mathcal{X}} m_p(x)\pi(x)dx$$

$$\sigma_{\mathcal{D}} = \int_{\mathcal{X}} \int_{\mathcal{X}} k_p(x, x')\pi(x)\pi(x')dxdx'$$



A. O'Hagan. Bayes–hermite quadrature. *Journal of Statisti-cal Planning and Inference*, 1991
P. Diaconis. Bayesian numerical analysis. *Statistical Deci-sion Theory and Related Topics IV*, 1988
F-X. Briol et al. Probabilistic integration: a role in statistical computation? *Statistical Science*, 2019

Numerical integration: Bayesian Quadrature

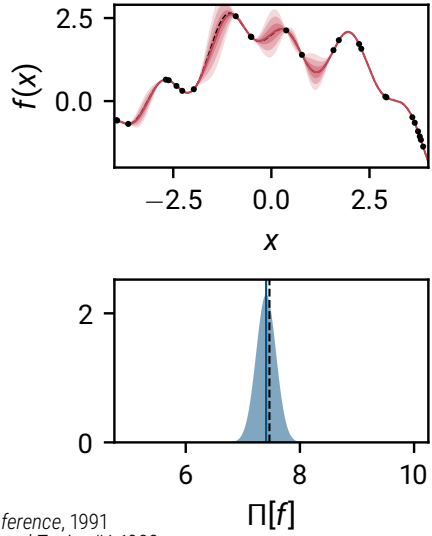
Prior $g \sim \mathcal{GP}(m, k)$

$$\Lambda = \int_{\mathcal{X}} g(x)\pi(x)dx$$

Note $\Lambda \sim \mathcal{N}(\mu, \sigma)$
Conditioning the GP on observations $\Lambda \mid \mathcal{D} \sim \mathcal{N}(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}})$

$$\mu_{\mathcal{D}} = \int_{\mathcal{X}} m_p(x)\pi(x)dx$$

$$\sigma_{\mathcal{D}} = \int_{\mathcal{X}} \int_{\mathcal{X}} k_p(x, x')\pi(x)\pi(x')dxdx'$$



A. O'Hagan. Bayes–hermite quadrature. *Journal of Statisti-cal Planning and Inference*, 1991
P. Diaconis. Bayesian numerical analysis. *Statistical Deci-sion Theory and Related Topics IV*, 1988
F-X. Briol et al. Probabilistic integration: a role in statistical computation? *Statistical Science*, 2019

Numerical integration: Bayesian Quadrature

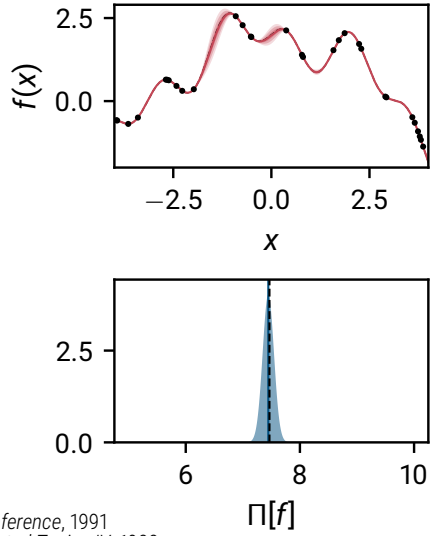
Prior $g \sim \mathcal{GP}(m, k)$

$$\Lambda = \int_{\mathcal{X}} g(x)\pi(x)dx$$

Note $\Lambda \sim \mathcal{N}(\mu, \sigma)$
Conditioning the GP on observations $\Lambda \mid \mathcal{D} \sim \mathcal{N}(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}})$

$$\mu_{\mathcal{D}} = \int_{\mathcal{X}} m_p(x)\pi(x)dx$$

$$\sigma_{\mathcal{D}} = \int_{\mathcal{X}} \int_{\mathcal{X}} k_p(x, x')\pi(x)\pi(x')dxdx'$$



A. O'Hagan. Bayes–hermite quadrature. *Journal of Statisti-cal Planning and Inference*, 1991
P. Diaconis. Bayesian numerical analysis. *Statistical Deci-sion Theory and Related Topics IV*, 1988
F-X. Briol et al. Probabilistic integration: a role in statistical computation? *Statistical Science*, 2019

Numerical integration: ?

- f and π cheap $\rightarrow$ MC-sampling
- f or π expensive $\rightarrow$ Bayesian quadrature

Wind farm example: f takes 2 min

- sampling expensive
- But: we can obtain many samples $\rightarrow$ Bayesian quadrature expensive

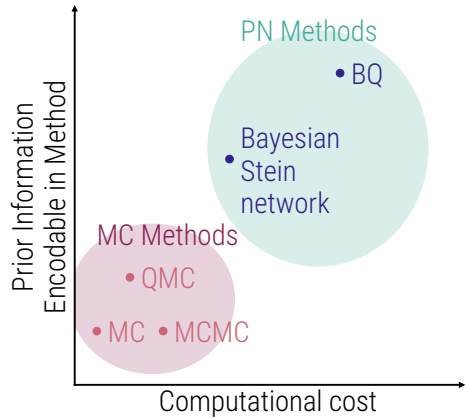
How can we handle large amounts of data + higher dimensions?

Neural networks for numerical integration

$$\Pi[f] = \int_{\mathcal{X}} f(x)\pi(x)dx.$$

Why Bayesian neural networks for integration tasks?

- Scaling to many data points
- Scaling to higher dimensions
- Memory efficient
- Neural networks interpolate well
- Uncertainty estimates



Numerical integration with a neural networks

1. Approximate f with a Bayesian neural networks $f_\theta \approx f$
2. Compute the integral (how? $\rightarrow$ later)

$$\Pi[f_\theta] = \int f_\theta(\alpha)\pi(\alpha)d\alpha \approx \Pi[f]$$

3. Compute posterior for $\Pi[f_\theta]$

Numerical integration with a neural networks

1. Approximate f with a **Bayesian neural networks** $f_\theta \approx f$
2. Compute the integral (how? $\rightarrow$ later)

$$\Pi[f_\theta] = \int f_\theta(\alpha)\pi(\alpha)d\alpha \approx \Pi[f]$$

3. Compute posterior for $\Pi[f_\theta]$

Bayesian neural networks

Neural networks

Neural network f_θ: $\mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$
Network parameters: $\theta \in \Theta \subseteq \mathbb{R}^{d_\theta}$
Dataset: $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$
Define cost function $\ell : \Theta \to \mathbb{R}$, e.g.:

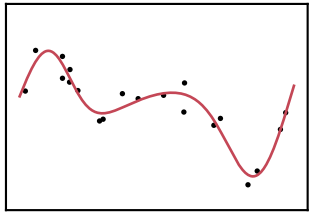
- Regression:

$$\ell(\theta) = \frac{1}{N} \sum_{n=1}^{N} ||f_\theta(x_n) - y_n||_2^2$$

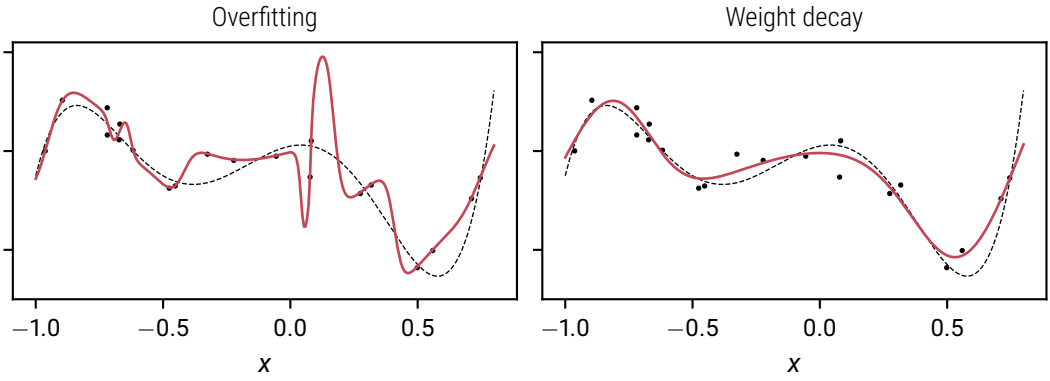
Goal:

$$\theta^* = \arg\min_\theta \ell(\theta)$$

Regression



Overfitting and weight decay



Reduce overfitting $\rightarrow$ regularize loss
e.g., via weight decay $\ell_{\text{decay}}(\theta) = \ell(\theta) + \lambda||\theta||_2^2$.

Bayesian neural networks

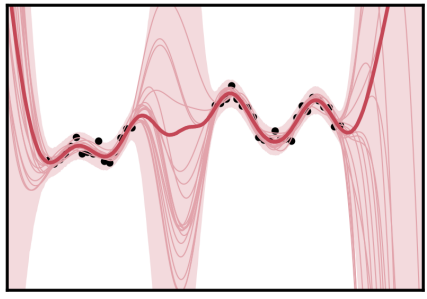
Prior: No knowledge about network parameters
$\rightarrow$ Gaussian prior $p(\theta) = \mathcal{N}(0, \sigma_0^2 \mathbb{I})$
Likelihood: $\log p(\mathcal{D} \mid \theta) \propto \ell(\theta)$
Posterior:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta) p(\theta)}{\int p(\mathcal{D} \mid \theta) p(\theta) d\theta}$$

Use MCMC to compute posterior and predictive posterior

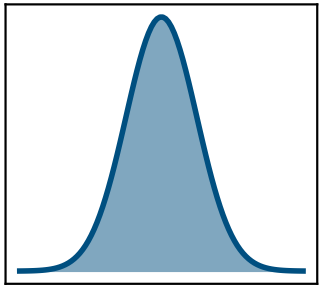
- computationally expensive
- cheap alternative $\rightarrow$ Laplace approximation

BNN



Laplace approximation

Prior $p(\theta)$

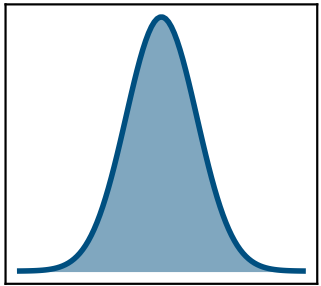


$$p(\theta \mid x, y) = \frac{p(y \mid \theta, x)p(\theta)}{\int p(y \mid \theta, x)p(\theta)d\theta}$$

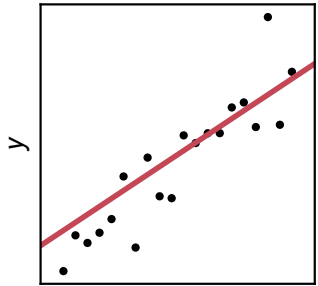
MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computation* (1992)

Laplace approximation

Prior $p(\theta)$



Likelihood $p(y \mid \theta, x)$

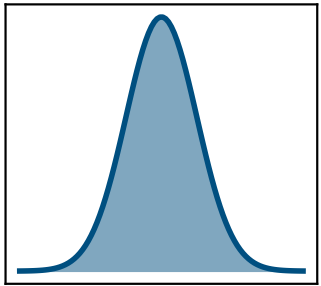


$$p(\theta \mid x, y) = \frac{p(y \mid \theta, x)p(\theta)}{\int p(y \mid \theta, x)p(\theta)d\theta}$$

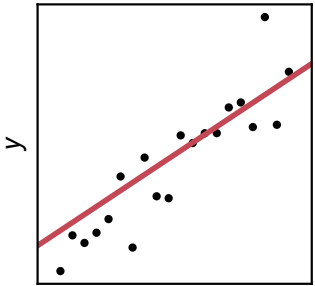
MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computation* (1992)

Laplace approximation

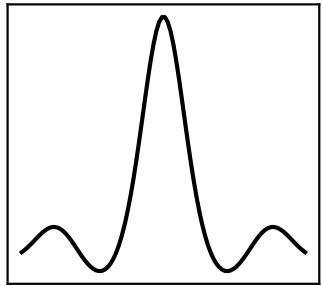
Prior $p(\theta)$



Likelihood $p(y \mid \theta, x)$



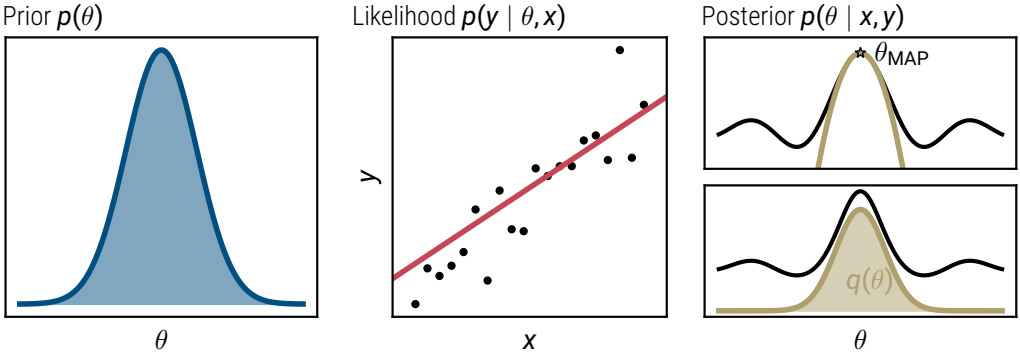
Posterior $p(\theta \mid x, y)$



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

MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computation* (1992)

Laplace approximation

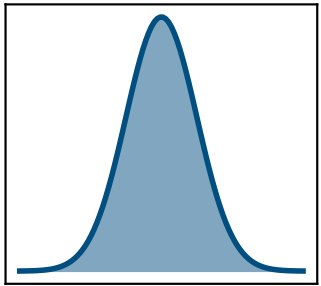


$$p(\theta \mid x, y) = \frac{p(y \mid \theta, x)p(\theta)}{\int p(y \mid \theta, x)p(\theta)d\theta}$$

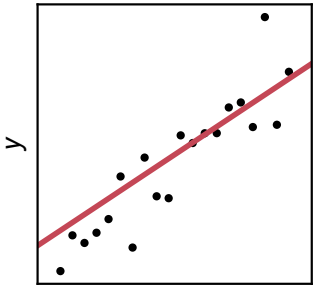
MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computation* (1992)

Laplace approximation

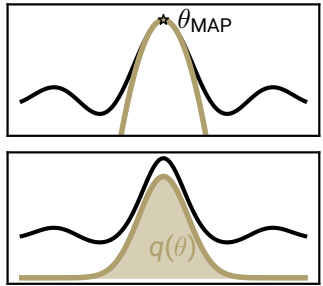
Prior $p(\theta)$



Likelihood $p(y \mid \theta, x)$



Posterior $p(\theta \mid x, y)$



$\log p(\theta)$ + $\log p(y \mid \theta, x)$ = $\log p(\theta \mid x, y) + \log Z$

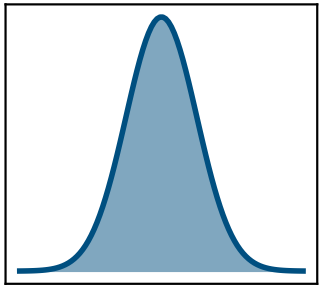
$\sim \lambda ||\theta||_2^2$ $\sim \frac{1}{N} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$

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

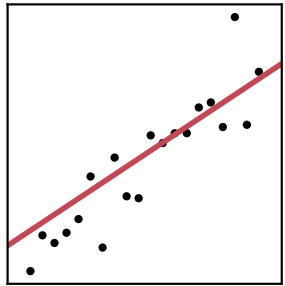
MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computation* (1992)

Laplace approximation

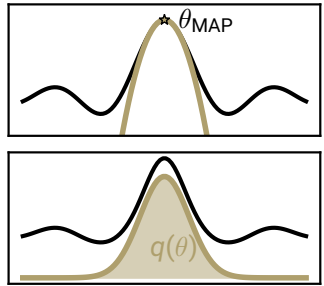
Prior $p(\theta)$



Likelihood $p(y \mid \theta, x)$



Posterior $p(\theta \mid x, y)$



$\log p(\theta)$ + $\log p(y \mid \theta, x)$ = $\log p(\theta \mid x, y) + \log Z$

$\sim \lambda ||\theta||_2^2$ $\sim \frac{1}{N} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$

Regularized loss

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

MacKay, A practical Bayesian framework for backpropagation networks, *Neural Computation* (1992)

Linearization and GGN

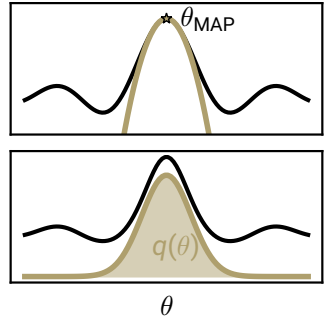
Taylor series approximation requires:

$$H = \nabla^2_\theta \log p(\theta \mid x, y)$$

- Not necessarily positive definite
- Computationally expensive

Use additional approximation of Hessian:

- Generalized Gauss Newton (GGN) approximation



Schraudolph, Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent, *Neural Computation* (2002)
Ritter et al., A scalable Laplace approximation for neural networks *ICLR* (2018)
Daxberger et al., Laplace Redux - Effortless Bayesian Deep Learning, *NeurIPS* (2021)

Integrating neural networks

Integrating neural networks

1. Approximate f with a Bayesian neural networks $f_\theta \approx f$
2. **Compute the integral**

$$\Pi[f_\theta] = \int f_\theta(\alpha)\pi(\alpha)d\alpha \approx \Pi[f]$$

3. Compute posterior for $\Pi[f_\theta]$

Stein network

Consider:

$$\mathcal{S}[u](x) := (\nabla_x \log \pi(x))^\top u(x) + \nabla_x u(x)$$

where $u : \mathbb{R}^n \to \mathbb{R}^n$
and $\nabla_x u(x) = \sum_i \partial_{x_i} u(x)_i$

Stein operator see: Anastasiou et al., *Statistical Science* (2023)

Stein network

Consider:

$$\mathcal{S}[u](x) := (\nabla_x \log \pi(x))^\top u(x) + \nabla_x u(x)$$

where $u : \mathbb{R}^n \to \mathbb{R}^n$
and $\nabla_x u(x) = \sum_i \partial_{x_i} u(x)_i$

$$\Pi\left[\mathcal{S}[u]\right] = \int_{\mathbb{R}^n} \left[(\nabla_x \log \pi(x)) \cdot u(x)\pi(x) + \nabla_x u(x) \cdot \pi(x)\right] dx$$

Stein operator see: Anastasiou et al., *Statistical Science* (2023)

Stein network

Consider:

$$\mathcal{S}[u](x) := (\nabla_x \log \pi(x))^\top u(x) + \nabla_x u(x)$$

where $u : \mathbb{R}^n \to \mathbb{R}^n$
and $\nabla_x u(x) = \sum_i \partial_{x_i} u(x)_i$

$$\begin{aligned}\Pi\left[\mathcal{S}[u]\right] &= \int_{\mathbb{R}^n} \left[(\nabla_x \log \pi(x)) \cdot u(x)\pi(x) + \nabla_x u(x) \cdot \pi(x)\right] dx \\ &= \int_{\mathbb{R}^n} \left[\frac{\nabla_x \pi(x)}{\pi(x)} \cdot u(x)\pi(x) + \nabla_x u(x) \cdot \pi(x)\right] dx\end{aligned}$$

Stein operator see: Anastasiou et al., *Statistical Science* (2023)

Stein network

Consider:

$$\mathcal{S}[u](x) := (\nabla_x \log \pi(x))^\top u(x) + \nabla_x u(x)$$

where $u : \mathbb{R}^n \to \mathbb{R}^n$
and $\nabla_x u(x) = \sum_i \partial_{x_i} u(x)_i$

$$\begin{aligned}
\Pi\left[\mathcal{S}[u]\right] &= \int_{\mathbb{R}^n} \left[(\nabla_x \log \pi(x)) \cdot u(x)\pi(x) + \nabla_x u(x) \cdot \pi(x)\right] dx \\
&= \int_{\mathbb{R}^n} \left[\frac{\nabla_x \pi(x)}{\pi(x)} \cdot u(x)\pi(x) + \nabla_x u(x) \cdot \pi(x)\right] dx \\
&= \underbrace{\left[\pi(x)u(x)\right]_{\mathbb{R}^n}}_{=0} + \int_{\mathbb{R}^n} \underbrace{\left[-\pi(x) \cdot \nabla_x u(x) + \nabla_x u(x) \cdot \pi(x)\right]}_{=0} dx \\
&= 0.
\end{aligned}$$

Stein operator see: Anastasiou et al., *Statistical Science* (2023)

Stein network

Consider:

$$\mathcal{S}[u](x) := (\nabla_x \log \pi(x))^\top u(x) + \nabla_x u(x)$$

$$\Pi\left[\mathcal{S}[u]\right] = 0$$

Stein operator see: Anastasiou et al., *Statistical Science* (2023)

Stein network

Consider:

$$\mathcal{S}[u](x) := \left(\nabla_x \log \pi(x)\right)^\top u(x) + \nabla_x u(x)$$

$$\Pi\left[\mathcal{S}[u]\right] = 0$$

Neural network:

$$f_\theta(x) := \mathcal{S}\left[u_{\theta_u}\right](x) + \theta_0$$

Stein operator see: Anastasiou et al., *Statistical Science* (2023)

Stein network

Consider:

$$\mathcal{S}[u](x) := (\nabla_x \log \pi(x))^\top u(x) + \nabla_x u(x)$$

$$\Pi\left[\mathcal{S}[u]\right] = 0$$

Neural network:

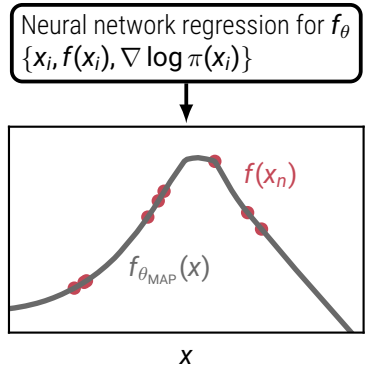
$$f_\theta(x) := \mathcal{S}\left[u_{\theta_u}\right](x) + \theta_0$$

Then

$$\Pi[f_\theta] = \Pi\left[\mathcal{S}[u_{\theta_u}]\right] + \Pi\left[\theta_0\right] = \theta_0$$

If $f_\theta \approx f$

$$\Pi[f] \approx \Pi[f_\theta] = \theta_0$$



Stein operator see: Anastasiou et al., *Statistical Science* (2023)

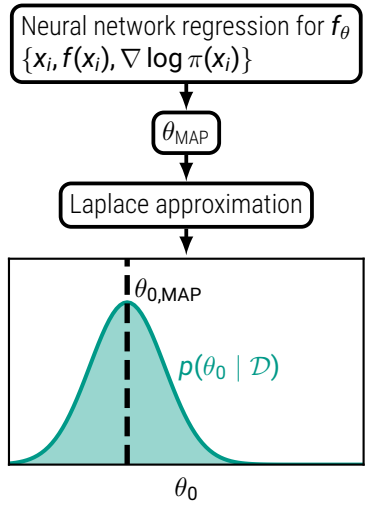
Laplace Approximation for Stein network

Via Laplace approximation:

$$p(\theta \mid \mathcal{D}) \approx \mathcal{N}(\theta \mid \theta_{\text{MAP}}, \Sigma_{\text{GGN}})$$

Then:

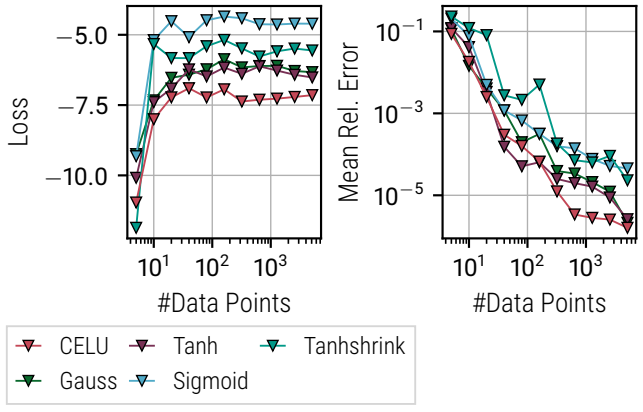
$$p(\theta_0 \mid \mathcal{D}) \approx \mathcal{N}(\theta_0 \mid \theta_{0,\text{MAP}}, (\Sigma_{\text{GGN}})_{0,0})$$



Architectural choices

Stein network requires special choices:

- **Choice of optimizer:** L-BFGS
- **Choice of activation function:** Needs to be differentiable
  - ReLU does not work
  - CELU works best

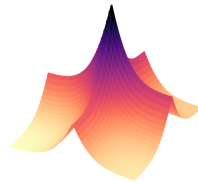


Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Does it work?

Continuous Genz dataset

- Standard integration benchmark dataset.
- Integration against a standard normal distribution $\pi(x) = \mathcal{N}(x \mid 0, 1)$.
- Dimension: $d = 20$.

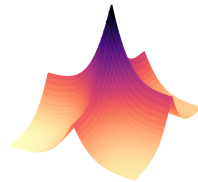


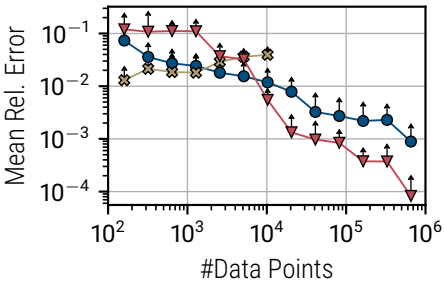
Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

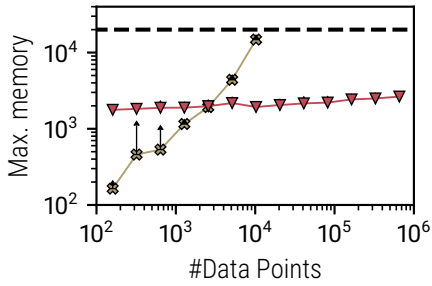
Does it work?

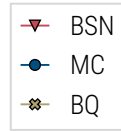
Continuous Genz dataset

- Standard integration benchmark dataset.
- Integration against a standard normal distribution $\pi(x) = \mathcal{N}(x \mid 0, 1)$.
- Dimension: $d = 20$.







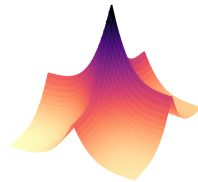


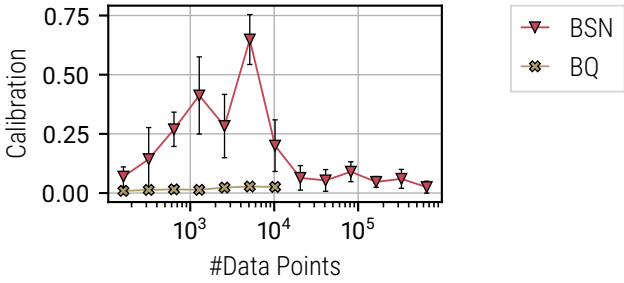
Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Does it work?

Continuous Genz dataset

- Standard integration benchmark dataset.
- Integration against a standard normal distribution $\pi(x) = \mathcal{N}(x \mid 0, 1)$.
- Dimension: $d = 20$.





Calibration:

$$\gamma = e_{\text{abs}} / \sigma_{\theta_0}$$

e_{abs}: integration error
σ_{θ_0}: standard deviation from Laplace/BQ

Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Parameter inference for ODEs

Task

- Data + ODE model for data

$$\frac{dz}{dt} = f(t, z, x)$$

- Prior for parameters $p(x)$
- Goal: compute posterior mean

$$\Pi[x] = \int x\pi(x)dx$$

here $\pi(x) = p(x \mid \mathcal{D})$.

Issues

- Sampling expensive (expensive $\nabla \log \pi$).
- π not available in normalized form $\rightarrow$ BQ impractical.

Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Parameter inference for ODEs

Task

- Data + ODE model for data

$$\frac{dz}{dt} = f(t, z, x)$$

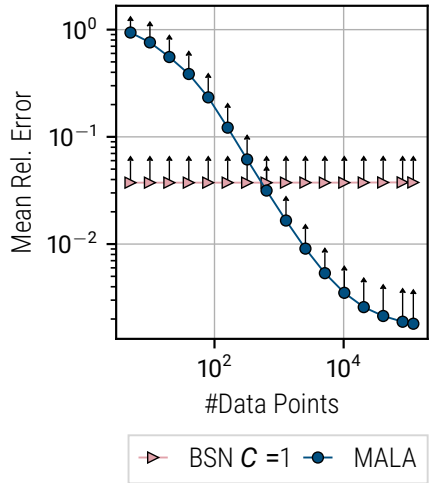
- Prior for parameters $p(x)$
- Goal: compute posterior mean

$$\Pi[x] = \int x\pi(x)dx$$

here $\pi(x) = p(x \mid \mathcal{D})$.

Issues

- Sampling expensive (expensive $\nabla \log \pi$).
- π not available in normalized form $\rightarrow$ BQ impractical.



Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Parameter inference for ODEs

Task

- Data + ODE model for data

$$\frac{dz}{dt} = f(t, z, x)$$

- Prior for parameters $p(x)$
- Goal: compute posterior mean

$$\Pi[x] = \int x\pi(x)dx$$

here $\pi(x) = p(x \mid \mathcal{D})$.

Issues

- Sampling expensive (expensive $\nabla \log \pi$).
- π not available in normalized form $\rightarrow$ BQ impractical.

Observation

$\nabla_x \log \pi(x)$ can take very large values

Solution

Diffusion Stein operator

$$\mathcal{S}_m[u](x) := \left(m(x)^\top \nabla_x \log \pi(x)\right)^\top u(x) + \nabla_x \cdot (m(x)u(x)),$$

Choice: Constant $m(x) = C^{-1}$ with

$$C = \max_{i=1,\ldots,n} \nabla_x \log \pi(x_i)$$

Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Parameter inference for ODEs

Task

- Data + ODE model for data

$$\frac{dz}{dt} = f(t, z, x)$$

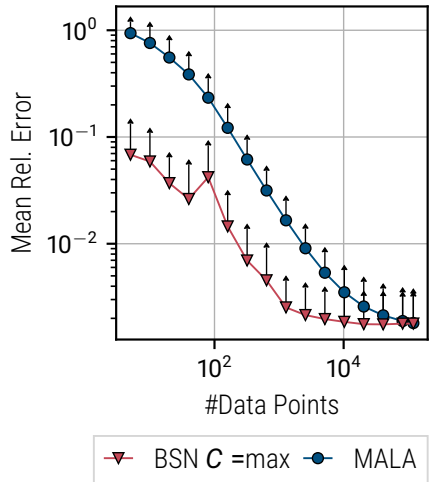
- Prior for parameters $p(x)$
- Goal: compute posterior mean

$$\Pi[x] = \int x\pi(x)dx$$

here $\pi(x) = p(x \mid \mathcal{D})$.

Issues

- Sampling expensive (expensive $\nabla \log \pi$).
- π not available in normalized form $\rightarrow$ BQ impractical.



Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Wind farm simulation

- Expected energy production of a wind farm
- Distributions over wind direction, turbulence intensity, …
- Sampling slow ($\sim$ 2 min)

Wind farm simulation

- Expected energy production of a wind farm
- Distributions over wind direction, turbulence intensity, …
- Sampling slow ($\sim$ 2 min)

Bounded domains

For the wind direction $\mathcal{X} = [a, b]$
But we need

$$\Pi\left[\mathcal{S}[u]\right] = 0$$

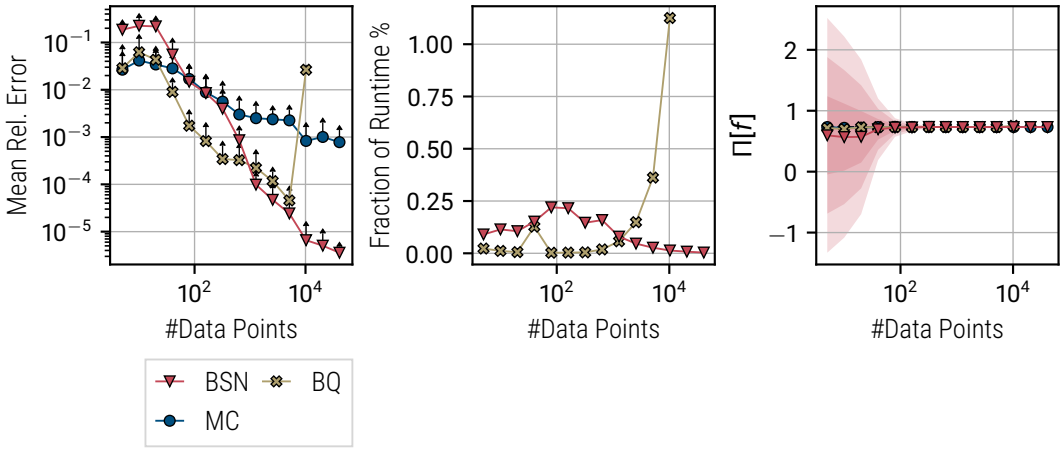
Choose

$$\tilde{u}(x) = u(x)\delta(x)$$
$$\delta(a) = 0, \quad \delta(b) = 0$$

For example

$$\delta(x) = (x - a)(x - b)$$

Wind farm simulation



Ott et al., Bayesian numerical integration with neural networks *UAI* (2023)

Limitations and discussion

Choice of prior

- Encoding more specific information hard
- Choice of Gaussian prior might be limiting

Scaling the architecture

- L-BFGS only feasible for small architectures
- To scale to high dimensions/ more datapoints
  - Batching
  - First order optimization (e.g. Adam, SGD)

Summary

Apporach

- Neural networks + Stein operator + Laplace approximation

Advantages

- Provides uncertainty estimates of good quality
- Scales well to large number of data points and higher dimensions
- Handles wide range of integration densities
  - e.g. if density only available in unnormalized form

Thank You!

Francois-Xavier Briol
University College London

Philipp Hennig
University of Tübingen

Michael Tiemann
Bosch Center for Artificial Intelligence