# Bayesian Machine Learning

## Neural Networks and Gaussian Processes

Diogo Ribeiro
ESMAD – Escola Superior de Média Arte e Design
Lead Data Scientist, Mysense.ai

November 16, 2025

# Outline

# Why Bayesian Machine Learning?

**Traditional ML Limitations:**

- Point estimates: Single "best" parameters
- Overconfidence: No uncertainty quantification
- Overfitting: Limited regularization mechanisms
- Model selection: Ad-hoc validation approaches

**Bayesian Advantages:**

- Uncertainty quantification: Principled confidence intervals
- Automatic regularization: Prior knowledge integration
- Model comparison: Marginal likelihood for selection
- Sequential learning: Natural online updates

## Bayesian Paradigm

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \quad (1)$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}} \quad (2)$$

## Key Insight

Treat parameters as random variables, not fixed unknowns

# Bayesian vs Frequentist Perspectives

| **Aspect** | **Frequentist** | **Bayesian** |
|---|---|---|
| Parameters | Fixed unknown constants | Random variables with distributions |
| Uncertainty | Confidence intervals (repeated sampling) | Credible intervals (probability statements) |
| Model Selection | Cross-validation, AIC/BIC | Marginal likelihood, posterior odds |
| Regularization | $L_1/L_2$ penalties | Prior distributions |
| Prediction | Point estimates | Predictive distributions |
| Computational | Optimization-based | Integration-based (MCMC/VI) |

## Bayesian Prediction

# Computational Challenges and Solutions

**The Integration Problem:**

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

**Challenges:**

- High-dimensional integrals
- No closed-form solutions
- Computational complexity
- Scalability to big data

**Historical Solutions:**

- Conjugate priors
- Laplace approximation

**Modern Approaches:**

**Markov Chain Monte Carlo**

- Hamiltonian Monte Carlo
- No-U-Turn Sampler (NUTS)
- Exact sampling (asymptotically)

**Variational Inference**

- Mean-field approximation
- Normalizing flows
- Automatic differentiation
- Scalable to large datasets

**Trade-offs**

**MCMC:** Exact but slow

# From Neural Networks to Bayesian Neural Networks

**Standard Neural Network:**

$$y = f(\mathbf{x}; \mathbf{w}) + \epsilon$$

**Learning:** Find optimal weights $\mathbf{w}^*$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \lambda R(\mathbf{w})$$

**Prediction:** Point estimate

$$p(y^*|\mathbf{x}^*) = \delta(y^* - f(\mathbf{x}^*; \mathbf{w}^*))$$

**Issues:**

- No uncertainty quantification
- Prone to overconfidence

**Bayesian Neural Network:**

$$y = f(\mathbf{x}; \mathbf{w}) + \epsilon, \quad \mathbf{w} \sim p(\mathbf{w})$$

**Learning:** Posterior distribution

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$$

**Prediction:** Averaging over weights

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w}$$

**Benefits:**

- Uncertainty quantification
- Automatic regularization

# BNN Architecture and Prior Specification

**Network Architecture:**

$$h_1 = \sigma(W_1 \mathbf{x} + b_1) \tag{3}$$
$$h_2 = \sigma(W_2 h_1 + b_2) \tag{4}$$
$$\vdots \tag{5}$$
$$y = W_L h_{L-1} + b_L \tag{6}$$

**Prior Distributions:**

$$W_{ij}^{(l)} \sim \mathcal{N}(0, \sigma_w^2) \tag{7}$$
$$b_i^{(l)} \sim \mathcal{N}(0, \sigma_b^2) \tag{8}$$
$$\sigma_y^2 \sim \text{InverseGamma}(\alpha, \beta) \tag{9}$$

## Prior Considerations

**Weight Scale:** Controls capacity

- Small $\sigma_w$: Smooth functions
- Large $\sigma_w$: Complex functions

**Architecture Prior:**

- Number of layers
- Hidden units per layer
- Activation functions

## Practical Tip

Use empirical Bayes or cross-validation for hyperparameter selection

# Inference Methods for BNNs

1. **Hamiltonian Monte Carlo**
   - Exact sampling (asymptotically)
   - Uses gradient information
   - Handles correlations well
   - Computationally intensive
2. **Variational Inference**
   - Approximate posterior $q(\mathbf{w}; \boldsymbol{\phi})$
   - Minimize KL divergence
   - Scalable to large networks
   - Mean-field assumption
3. **Monte Carlo Dropout**
   - Approximate Bayesian inference
   - Keep dropout at test time
   - Ensemble of sub-networks
   - Computationally efficient
4. **Ensemble Methods**
   - Train multiple networks
   - Different initializations
   - Bootstrap sampling
   - Deep ensembles

## VI Objective

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{q(\mathbf{w};\boldsymbol{\phi})}[\log p(\mathcal{D}|\mathbf{w})] - \mathrm{KL}\left(q(\mathbf{w};\boldsymbol{\phi}) \| p(\mathbf{w})\right)$$

## Practical Recommendation

**Small networks:** MCMC

**Large networks:** Variational inference

# Variational Inference for BNNs: Bayes by Backprop

**Algorithm: Bayes by Backprop (Blundell et al., 2015)**

**Algorithm 1** Variational Inference for BNN

1: **Initialize:** Variational parameters $\boldsymbol{\phi}$ for $q(\mathbf{w}|\boldsymbol{\phi})$
2: **for** each iteration **do**
3: Sample weights: $\mathbf{w} \sim q(\mathbf{w}|\boldsymbol{\phi})$
4: Compute loss: $\mathcal{L} = -\log p(\mathcal{D}|\mathbf{w}) + \mathsf{KL}(q(\mathbf{w})\|p(\mathbf{w}))$
5: Compute gradients: $\nabla_{\boldsymbol{\phi}}\mathcal{L}$ using reparameterization trick
6: Update: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha\nabla_{\boldsymbol{\phi}}\mathcal{L}$
7: **end for**

**Reparameterization Trick:**

$$\mathbf{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I) \tag{10}$$

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \mathsf{diag}(\boldsymbol{\sigma}^2)) \tag{11}$$

**Key Innovation:** Gradient-based optimization of approximate posterior

# BNN Applications and Case Studies

1. **Regression with Uncertainty**
   - Heteroscedastic noise modeling
   - Confidence intervals for predictions
   - Outlier detection
   - Active learning applications
2. **Classification with Calibration**
   - Well-calibrated probabilities
   - Uncertainty in predictions
   - Out-of-distribution detection
   - Medical diagnosis applications
3. **Reinforcement Learning**
   - Exploration via uncertainty
   - Thompson sampling

## Case Study: Medical Diagnosis

**Problem:** Skin cancer classification
**Dataset:** 10,000 dermoscopy images
**BNN Results:**

- 94.2% accuracy (vs 93.8% standard NN)
- Well-calibrated confidence scores
- Identifies uncertain cases for expert review
- 15% reduction in misdiagnosis risk

**Key Insight:** Uncertainty quantification more valuable than accuracy gain

## Industrial Impact

BNNs enable **safe AI deployment** in critical applications

# Introduction to Gaussian Processes

## Definition (Gaussian Process)

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

**Key Insight:** Instead of parameterizing functions, put distributions directly over functions.

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{12}$$
$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{13}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')] \tag{14}$$

**Properties:**

- Non-parametric method
- Infinite dimensional

## Finite Dimensional Consistency

For any finite set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$:

$$\begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix} \sim \mathcal{N}\left( \begin{pmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_n) \end{pmatrix}, K \right)$$

where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

## Computational Complexity

# Kernel Functions and Prior Specification

**Popular Kernel Functions:**

**1. Squared Exponential (RBF):**

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right)$$

**2. Matérn:**

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{\ell}\right)^\nu K_\nu \left(\frac{\sqrt{2\nu}r}{\ell}\right)$$

**3. Periodic:**

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{2\sin^2(\pi|x - x'|/p)}{\ell^2}\right)$$

**4. Linear:**

**Kernel Properties:**

- **Length scale** $\ell$: Controls smoothness
- **Signal variance** $\sigma_f^2$: Output scale
- **Noise variance** $\sigma_n^2$: Observation noise

**Kernel Composition:**

- **Addition:** $k_1 + k_2$ (combining patterns)
- **Multiplication:** $k_1 \times k_2$ (conjunction)
- **Scaling:** $\alpha k$ (amplitude)

**Example: Trend + Periodic**

$$k(\mathbf{x}, \mathbf{x}') = k_{\text{linear}}(\mathbf{x}, \mathbf{x}') + k_{\text{periodic}}(\mathbf{x}, \mathbf{x}') + k_{\text{noise}}(\mathbf{x},$$

# GP Regression: Predictive Distribution

**Training Data:** $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
**Likelihood:** $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$

## Predictive Distribution

For a new input $\mathbf{x}^*$, the predictive distribution is:

$$p(f^*|\mathbf{x}^*, \mathcal{D}) = \mathcal{N}(f^*; \mu^*, (\sigma^*)^2) \tag{15}$$

$$\mu^* = \mathbf{k}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{16}$$

$$(\sigma^*)^2 = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k} \tag{17}$$

where:

- $\mathbf{k} = [k(\mathbf{x}^*, \mathbf{x}_1), \ldots, k(\mathbf{x}^*, \mathbf{x}_n)]^T$
- $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$
- $\mathbf{y} = [y_1, \ldots, y_n]^T$

**Key Properties:**

- Exact Bayesian inference (given kernel)

# Hyperparameter Learning in GPs

**Marginal Likelihood:**

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{K} + \sigma_n^2 \mathbf{I}) \tag{18}$$

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} \tag{19}$$

$$- \frac{1}{2}\log |\mathbf{K}_y| - \frac{n}{2}\log 2\pi \tag{20}$$

where $\mathbf{K}_y = \mathbf{K} + \sigma_n^2 \mathbf{I}$ and $\boldsymbol{\theta}$ are hyperparameters.

**Optimization:**

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

**Three Terms Interpretation:**

1. **Data fit:** $-\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$
2. **Complexity penalty:** $-\frac{1}{2}\log |\mathbf{K}_y|$
3. **Normalization:** $-\frac{n}{2}\log 2\pi$

**Automatic Occam's Razor**

GPs automatically balance model complexity and data fit through the marginal likelihood

**Practical Implementation**

Use gradient-based optimization (L-BFGS) with multiple random restarts

# Sparse Gaussian Processes

**Problem:** Standard GP inference scales as $O(n^3)$
**Solution:** Inducing point methods

**Key Idea:** Approximate full GP using $m \ll n$ inducing points

$$f(\mathbf{x}) \approx \tilde{f}(\mathbf{x}) = \mathbf{k}_*^T \mathbf{K}_{mm}^{-1} \mathbf{f}_m \quad (21)$$

$$\mathbf{f}_m = [f(\mathbf{z}_1), \ldots, f(\mathbf{z}_m)]^T \quad (22)$$

where $\{\mathbf{z}_i\}_{i=1}^m$ are inducing inputs.

**Variational Sparse GP:**

- Optimize inducing inputs $\{\mathbf{z}_i\}$
- Variational distribution $q(\mathbf{f}_m)$

## FITC Approximation

Fully Independent Training Conditional:

$$q(\mathbf{f}) = p(\mathbf{f}|\mathbf{f}_m) q(\mathbf{f}_m)$$

## Stochastic Variational GP

- Mini-batch training
- Natural gradients
- Scales to millions of points
- $O(m^3)$ per iteration

**Modern Extensions:**

# GP Applications and Case Studies

1. **Bayesian Optimization**
   - Expensive function optimization
   - Acquisition functions (EI, UCB, PI)
   - Hyperparameter tuning
   - Experimental design
2. **Time Series Forecasting**
   - Temporal kernels
   - Uncertainty in predictions
   - Missing data handling
   - Irregular time series
3. **Spatial Statistics**
   - Geostatistics and kriging
   - Environmental monitoring

## Case Study: Drug Discovery

**Problem:** Optimize molecular properties

**Setup:**

- 10,000 molecules tested
- Each test costs $1000
- Goal: Find top 1% molecules

**GP-based Optimization:**

- Molecular fingerprints as features
- Tanimoto kernel for similarity
- Expected improvement acquisition

**Results:**

- 95% reduction in tests needed
- Found optimal molecules in 500 tests

# Deep Gaussian Processes

**Motivation:** Combine flexibility of deep learning with uncertainty of GPs

**Architecture:**

$$\mathbf{h}_1 \sim \mathcal{GP}(\mathbf{0}, k_1(\mathbf{x}, \mathbf{x}')) \tag{23}$$
$$\mathbf{h}_2 \sim \mathcal{GP}(\mathbf{0}, k_2(\mathbf{h}_1, \mathbf{h}_1')) \tag{24}$$
$$\vdots \tag{25}$$
$$\mathbf{y} \sim \mathcal{GP}(\mathbf{0}, k_L(\mathbf{h}_{L-1}, \mathbf{h}_{L-1}')) \tag{26}$$

**Properties:**

- Non-stationary kernels
- Hierarchical feature learning
- Uncertainty propagation through layers
- Automatic relevance determination

**Inference Challenges:**

- Intractable posterior
- Doubly stochastic variational inference
- Reparameterization trick for GPs
- Computational complexity

**Variational Approach**

- Variational distribution for each layer
- Monte Carlo estimates
- Natural gradient optimization

**Applications:**

- High-dimensional regression
- Dimensionality reduction

# Neural Networks vs Gaussian Processes: The Infinite Width Limit

**Remarkable Connection:** Neural networks converge to Gaussian processes in the infinite width limit

## Theorem (Neal, 1996)

*Consider a single hidden layer neural network:*

$$f(\mathbf{x}) = \frac{1}{\sqrt{H}} \sum_{i=1}^{H} v_i \sigma(w_i^T \mathbf{x} + b_i)$$

*As $H \to \infty$ with i.i.d. weights $w_i, v_i, b_i$, the function $f(\mathbf{x})$ converges to a Gaussian process.*

**Neural Tangent Kernel:**

- Describes infinite-width NN dynamics
- Fixed kernel during training

## Modern Extensions

**Deep Neural Networks:**

- Infinite depth limit
- Bayesian neural networks at scale

# Variational Inference: Connecting BNNs and GPs

**Unified Framework:** Both BNNs and GPs can be viewed through variational inference lens

**Bayesian Neural Networks:**

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{w})}[\log p(\mathcal{D}|\mathbf{w})] - \text{KL}(q(\mathbf{w})\|p(\mathbf{w})) \tag{27}$$

**Sparse Gaussian Processes:**

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{f})}[\log p(\mathcal{D}|\mathbf{f})] - \text{KL}(q(\mathbf{f})\|p(\mathbf{f})) \tag{28}$$

**Common Techniques:**

- Reparameterization trick
- Natural gradients
- Stochastic optimization

**Modern Developments:**

**Normalizing Flows**

- Flexible posterior approximations
- Invertible transformations
- Better than mean-field

**Neural Processes**

- Combine NNs and GPs
- Amortized inference
- Meta-learning for functions

**Practical Tools:**

- PyTorch, TensorFlow Probability

# Software Ecosystem for Bayesian ML

| Framework | Strengths | Applications | Language |
|---|---|---|---|
| PyTorch | Flexible, research-friendly | Custom BNN architectures | Python |
| TensorFlow Prob. | Production-ready, scalable | Large-scale deployment | Python |
| GPyTorch | GPU acceleration for GPs | Large-scale GP inference | Python |
| GPflow | TensorFlow-based GPs | Deep GPs, sparse methods | Python |
| Stan | Probabilistic programming | Custom model specification | Multiple |
| PyMC | User-friendly Bayesian | Educational, prototyping | Python |

**Performance Considerations:** **Production Deployment:**

# Implementation Example: Simple BNN in PyTorch

## Bayesian Linear Layer

```
class BayesianLinear(nn.Module):
    def __init__(self, in_features, out_features):
        super().__init__()
        # Weight parameters
        self.weight_mu = nn.Parameter(
            torch.zeros(out_features, in_features))
        self.weight_rho = nn.Parameter(
            torch.ones(out_features, in_features) * -3)

        # Bias parameters
        self.bias_mu = nn.Parameter(
            torch.zeros(out_features))
        self.bias_rho = nn.Parameter(
            torch.ones(out_features) * -3)
```

**Key Components:**

- Variational parameters: $\mu$, $\rho$
- Reparameterization: $w = \mu + \sigma\epsilon$
- Softplus: $\sigma = \log(1 + \exp(\rho))$

**Training Loop:**

- Sample weights each forward pass
- Compute ELBO loss
- Backprop through sampling
- Update variational parameters

# Best Practices and Common Pitfalls

**Bayesian Neural Networks:**

## Best Practices

- Start with prior sensitivity analysis
- Use multiple random seeds
- Monitor KL divergence during training
- Validate uncertainty calibration
- Consider computational budget

## Common Pitfalls

- Overconfident posterior approximations
- Poor initialization of variational parameters
- Ignoring computational overhead

**Gaussian Processes:**

## Best Practices

- Choose kernels based on problem structure
- Use multiple optimization restarts
- Validate on held-out data
- Consider sparse approximations for large data
- Monitor numerical stability

## Common Pitfalls

- Poor kernel choice for the problem
- Local optima in hyperparameter optimization
- Numerical issues with matrix inversion

# Current Research Frontiers

**Methodological Advances:**

- Continual learning with uncertainty
- Federated Bayesian learning
- Physics-informed priors
- Causal discovery with GPs
- Multi-modal Bayesian models

**Application Domains:**

- Autonomous systems (vehicles, drones)
- Healthcare AI with safety guarantees
- Climate modeling and prediction
- Financial risk assessment
- Scientific discovery acceleration

**Computational Innovations:**

- Quantum-enhanced sampling
- Neuromorphic computing for BNNs
- Distributed inference at scale
- Edge deployment of Bayesian models

**Societal Impact:**

- Trustworthy AI development
- Algorithmic fairness with uncertainty
- Privacy-preserving ML
- Explainable AI through Bayesian lens

## The Next Decade

Bayesian ML will become the **standard approach** for safety-critical applications requiring

# Summary and Key Takeaways

**Bayesian Neural Networks:**

- Uncertainty quantification for deep learning
- Automatic regularization through priors
- Computational challenges in large networks
- Growing adoption in critical applications

**Gaussian Processes:**

- Non-parametric Bayesian approach
- Exact inference for regression
- Flexible through kernel design
- Scalability remains a challenge

**Practical Guidelines:**

- Start with **simple baselines**
- Validate **uncertainty calibration**
- Consider **computational constraints**
- Use **appropriate software** tools
- Focus on **problem-specific** solutions

**When to Use What?**

**BNNs:** Large datasets, complex patterns, representation learning

**GPs:** Small-medium datasets, interpretability, principled uncertainty

# Thank You

**Questions & Discussion**

**Diogo Ribeiro**
ESMAD – Escola Superior de Média Arte e Design
Lead Data Scientist, Mysense.ai

dfr@esmad.ipp.pt
https://orcid.org/0009-0001-2022-7072

*Slides and code available at:*
github.com/diogoribeiro7/academic-presentations