Bayesian Logistic Regression

ELG 5218 - Uncertainty Evaluation in Engineering Measurements and Machine Learning

Miodrag Bolic

University of Ottawa

January 28, 2026

The Logistic Regression Model

Discriminative Classification:

- We model $p(y|x, \boldsymbol{w})$ directly using the logistic sigmoid function $\sigma(\cdot)$.
- Prediction: $\mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})}$.
- Likelihood (Bernoulli):

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \prod_{n=1}^{N} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} (1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^{1-y_n}$$

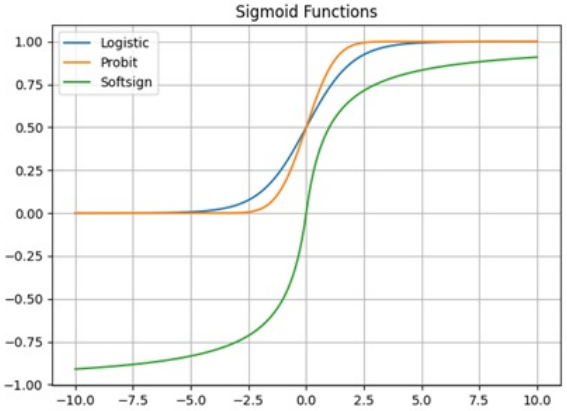
The Bayesian Goal:

- Introduce a prior on weights: $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$.
- Compute the **Posterior**: $p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})$.
- Compute the **Predictive Distribution**: $p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})$.

Sigmoid Functions

Sigmoid Options:

- **Logistic:** Simple closed form, standard choice. Heavy tails; posterior not conjugate with a Gaussian prior.
- **Probit:** Gaussian CDF; lighter tails, links to latent-Gaussian models. More expensive; posterior still intractable.
- **Softsign:** Cheap and smooth, but not a probability model and poorly calibrated.



The Intractability Problem

Why is this harder than Linear Regression?

Posterior Distribution

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) \propto \underbrace{\prod_{n=1}^{N} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n}(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^{1-y_n}}_{\text{Likelihood (Sigmoids)}} \times \underbrace{\exp\left(-\frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w}\right)}_{\text{Prior (Gaussian)}}$$

- The likelihood is a product of sigmoids; the prior is Gaussian.
- They are **not conjugate**.
- The product is **not** a Gaussian (and not a standard distribution).
- The normalization constant $Z = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}$ is analytically intractable.

Solution: We must use approximations. Today: **The Laplace Approximation**.

Why We Use Laplace Approximation

Why Not Use Standard Loss?

- Classification usually optimized via *cross-entropy* (negative log-likelihood).
- Works for point estimation (MLE/MAP), but **does not give the posterior**.
- Logistic likelihood and Gaussian prior are **not conjugate** $\rightarrow$ posterior integral is intractable.

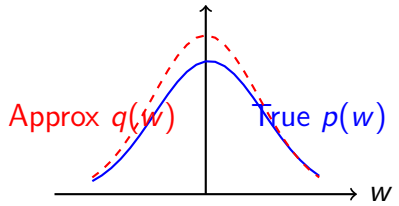
Why Laplace Approximation?

- Approximates the posterior by a Gaussian around the MAP.
- Makes Bayesian logistic regression computationally feasible.

The Laplace Approximation

Idea: Approximate the intractable posterior $p(\boldsymbol{w}|\mathcal{D})$ with a Gaussian $q(\boldsymbol{w})$.

1. Find the mode of the posterior, $\boldsymbol{w}_{MAP}$ (Maximum A Posteriori).
2. Compute the curvature (Hessian) of the log-posterior at the mode.
3. Construct a Gaussian centered at $\boldsymbol{w}_{MAP}$ with covariance determined by the curvature.



$$\ln p(\boldsymbol{w}|\mathcal{D}) \approx \ln p(\boldsymbol{w}_{MAP}|\mathcal{D}) - \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{MAP})^\top \boldsymbol{A}(\boldsymbol{w} - \boldsymbol{w}_{MAP})$$

$$q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{MAP}, \boldsymbol{A}^{-1})$$

Where $\boldsymbol{A} = -\nabla\nabla \ln p(\boldsymbol{w}|\mathcal{D})|_{\boldsymbol{w}_{MAP}}$ (Negative Hessian).

Step 1: Finding w_{MAP}

We maximize the log-posterior:

$$\ln p(\boldsymbol{w}|\mathcal{D}) = -\frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + \sum_{n=1}^{N}\{y_n \ln \mu_n + (1-y_n)\ln(1-\mu_n)\} + \text{const}$$

where $\mu_n = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)$.

Gradient (∇E):

$$\nabla \ln p(\boldsymbol{w}|\mathcal{D}) = -\alpha \boldsymbol{w} + \sum_{n=1}^{N}(y_n - \mu_n)\boldsymbol{x}_n = -\alpha \boldsymbol{w} + \boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{\mu})$$

Algorithm: Since this is convex, we use **gradient descent** .

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \boldsymbol{\eta}\nabla E(\boldsymbol{w}_{old})$$

Step 2: The Hessian Matrix

To apply Newton-Raphson and find the Laplace covariance, we need the second derivatives.

$$\boldsymbol{H} = \nabla\nabla \ln p(\boldsymbol{w}|\mathcal{D}) = -\alpha \boldsymbol{I} - \sum_{n=1}^{N} \mu_n(1-\mu_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top$$

In matrix notation:

$$\boldsymbol{H} = -(\alpha \boldsymbol{I} + \boldsymbol{X}^\top \boldsymbol{S}\boldsymbol{X})$$

where $\boldsymbol{S}$ is a diagonal weighting matrix with elements $S_{nn} = \mu_n(1-\mu_n)$.

The Laplace Approximation Result

The posterior is approximated as Gaussian $q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{MAP}, \boldsymbol{\Sigma}_N)$ with:

$$\boldsymbol{\Sigma}_N^{-1} = \alpha \boldsymbol{I} + \boldsymbol{X}^\top \boldsymbol{S}\boldsymbol{X}$$

Predictive Distribution: The Challenge

We have the posterior $q(\boldsymbol{w})$. Now we want to predict class y_* for a new input $\boldsymbol{x}_*$.

$$p(y_* = 1|\boldsymbol{x}_*, \mathcal{D}) = \int p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{w}) q(\boldsymbol{w}) d\boldsymbol{w}$$

$$= \int \sigma(\boldsymbol{w}^\top \boldsymbol{x}_*) \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{MAP}, \boldsymbol{\Sigma}_N) d\boldsymbol{w}$$

Problem: This is the convolution of a Sigmoid and a Gaussian.

- Still analytically intractable!
- However, it's a 1D integral of a projected Gaussian.

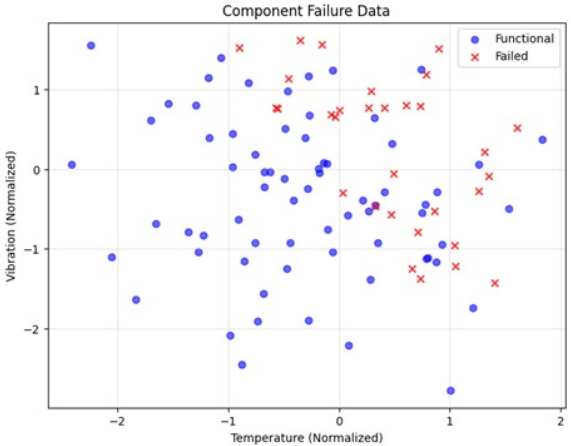
Engineering Example: Component Failure Prediction

Scenario: We want to predict mechanical component failure based on sensor data.

- x_1: Operating Temperature (Normalized)
- x_2: Vibration Frequency (Normalized)
- $y \in \{0, 1\}$: Functional vs. Failed

Why Bayesian?

- We have limited failure data (rare events).
- We need to know **when** the model is uncertain to trigger manual inspection.



Synthetic Failure Data

Implementation in NumPyro

Defining a Bayesian Logistic Regression model in probabilistic code:

Python/NumPyro Code

```
def logistic_model(X, y=None):
    # 1. Priors (Gaussian regularization)
    w = numpyro.sample("w", dist.Normal(0, 1).expand([2]))
    b = numpyro.sample("b", dist.Normal(0, 1))

    # 2. Logits
    logits = jnp.dot(X, w) + b

    # 3. Likelihood (Bernoulli)
    with numpyro.plate("data", len(X)):
        numpyro.sample("obs", dist.Bernoulli(logits=logits), obs=y)
```

Numpyro: What are `plate`, `jnp.dot`, and `numpyro.sample`?

1) `numpyro.plate("data", N)`
Declares an i.i.d. dimension of size N and vectorizes the computation. For a Bernoulli likelihood,
$\log p(\mathbf{y} \mid X, \theta) = \sum_{n=1}^{N} \log \text{Bernoulli}\big(y_n \,;\, \sigma(\mathbf{w}^\top \mathbf{x}_n + b)\big)$.

2) `jnp.dot(X, w)`
Matrix–vector product. With $X \in \mathbb{R}^{N \times D}$ and $w \in \mathbb{R}^D$, it returns logits $\in \mathbb{R}^N$ where $\text{logits}_n = \sum_{j=1}^{D} X_{nj} w_j$.

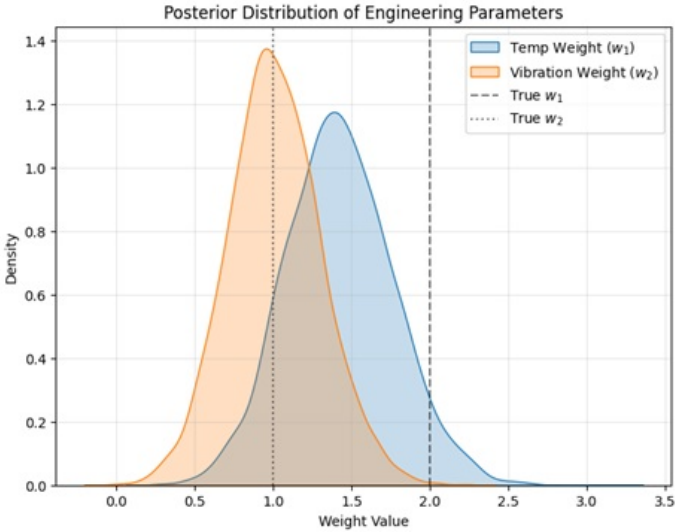
3) `numpyro.sample(name, dist, obs=None)`
Creates a random variable site from a distribution. Used for priors and likelihood. When `obs` is provided, the node is *observed* and contributes its log-probability; when omitted, NumPyro will draw from the distribution during prior/predictive simulation.

Posterior Uncertainty in Parameters

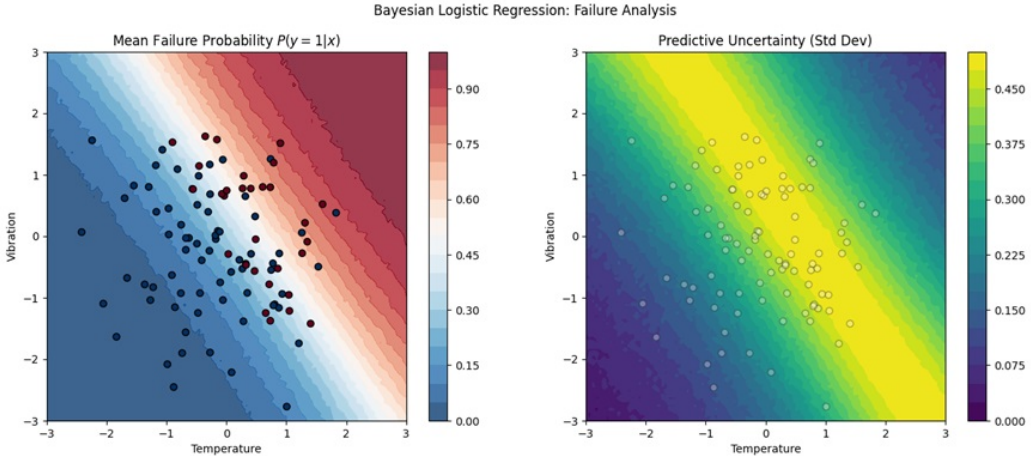
Instead of a single "best fit" weight vector $\hat{\boldsymbol{w}}_{MLE}$, we get a distribution over possible weights.

- **Blue:** Temperature Sensitivity (w_1)
- **Orange:** Vibration Sensitivity (w_2)

We are more confident about w_1 (narrower peak) than w_2.



Visualizing Predictive Uncertainty



- **Left:** Mean probability of failure (The Decision Boundary).
- **Right:** Predictive Standard Deviation (Where is the model confused?).

Reading the Posterior Predictive Maps

Left: Mean Failure Probability

- Color $= \mathbb{E}_w[\sigma(w^\top x + b)]$.
- Black contour at 0.5 is the decision boundary.
- High Temp + High Vibration $\Rightarrow$ red (high failure risk).

Right: Predictive Uncertainty (Std of y)

- Peaks near the boundary (mean ≈ 0.5) and in sparse regions.
- Decomposes into aleatoric $E[p(1-p)]$ and epistemic $\mathrm{Var}(p)$.
- Use high-uncertainty flags for additional inspection.

Summary

1. **Theory:** Logistic Regression likelihood $\times$ Gaussian Prior $\neq$ Gaussian Posterior.
2. **Approximation:** We used Laplace Approx (Taylor Series) to estimate $p(\boldsymbol{w}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{w}_{MAP}, \boldsymbol{H}^{-1})$.
3. **Implementation:** Modern tools like NumPyro allow full MCMC inference for critical engineering applications where uncertainty quantification is key.

Summary Beyond Laplace

Summary:

1. Bayesian Logistic Regression is non-conjugate.
2. **Laplace Approximation**:
   - Fit MAP via IRLS.
   - Use Hessian for Covariance $\mathbf{\Sigma}_N$.
3. **Predictive Distribution**:
   - Convolve Sigmoid with Gaussian.
   - Result: "Moderated" Sigmoid (less confident predictions when uncertain).

Other Approaches (Higher accuracy/cost):

- **Variational Inference (VI):** Optimizes a lower bound on marginal likelihood. (Jaakkola & Jordan, 2000).
- **MCMC:** Metropolis-Hastings or Hamiltonian Monte Carlo (Gold standard, slow).
- **EP (Expectation Propagation):** Often more accurate than Laplace for logistic regression.

Backup slides about Laplace approximation

Learning goals

- Understand the **Hessian** as *curvature* (1D) and *curvature* + *coupling* (multi-D).
- Understand the **Laplace approximation** as a local Gaussian approximation around the MAP.
- See when Laplace works well (unimodal, near-Gaussian) and when it fails (multimodal, skewed).
- Apply Laplace to **Bayesian logistic regression**: approximate posterior + credible intervals + predictive bands.

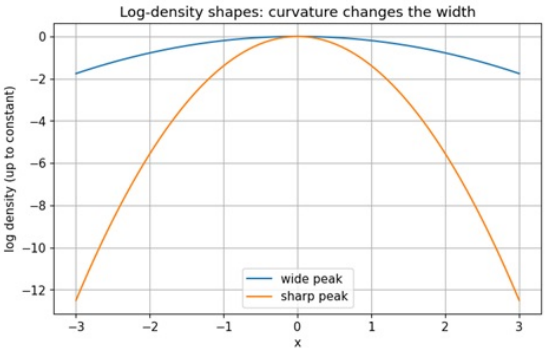
Hessian in 1D: second derivative = curvature

For a 1D function $f(x)$:

- $f'(x)$ is the **slope**.
- $f''(x)$ is how the slope changes: the **curvature**.

For a *log-density* peak:

- **More curvature** (more negative f'' at the peak) $\Rightarrow$ **narrower** uncertainty.
- **Less curvature** $\Rightarrow$ **wider** uncertainty.

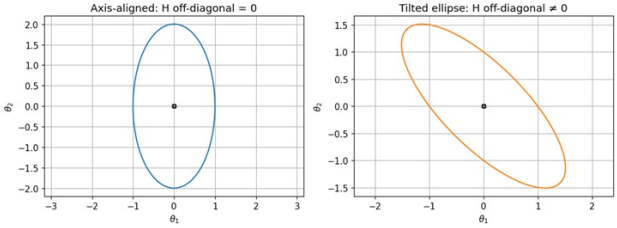


Hessian in 2D: curvature + coupling

In many dimensions, the Hessian is a matrix: $H_{ij}(\theta) = \frac{\partial^2 f}{\partial \theta_i \partial \theta_j}$.

Intuition:

- Diagonal entries: curvature in each coordinate direction.
- Off-diagonal entries: **coupling** (how directions interact) $\Rightarrow$ tilted ellipses.



Near a mode, if the negative log posterior looks like $U(\theta) \approx U(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^\top H(\theta - \hat{\theta})$, then the covariance is approximately $\Sigma \approx H^{-1}$.

Laplace approximation: the one-line story

Goal: approximate a difficult posterior $p(\theta \mid y)$ by a Gaussian.
Define the negative log posterior:

$$U(\theta) = -\log p(\theta \mid y).$$

Step 1 (MAP): find the mode / MAP

$$\hat{\theta} = \arg\min_{\theta} U(\theta).$$

Step 2 (curvature): compute the Hessian at the MAP

$$H = \nabla^2 U(\hat{\theta}).$$

Step 3 (Gaussian): approximate

$$p(\theta \mid y) \approx \mathcal{N}\big(\hat{\theta},\ H^{-1}\big).$$

Translation: *mode becomes mean; curvature becomes covariance.*

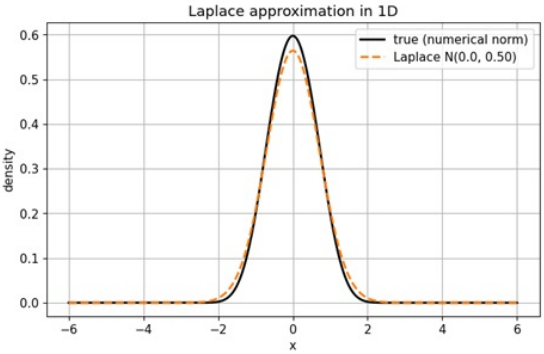
Example: a slightly non-Gaussian target

Consider

$$p(x) \propto \exp(-x^2 - 0.1x^4).$$

It is not exactly Gaussian (tails differ), but it is *approximately* Gaussian near its peak.
At the mode $\hat{x} = 0$:

$$U(x) = x^2 + 0.1x^4, \quad U''(0) = 2 \Rightarrow \sigma^2 \approx 1/2.$$



Why it becomes Gaussian: quadratic log-density

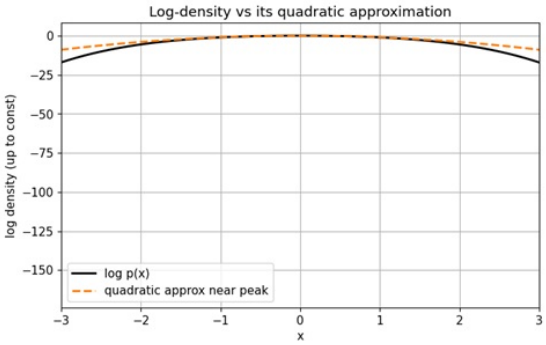
A Gaussian has a **quadratic** in the exponent:

$$\mathcal{N}(\mu, \sigma^2) \propto \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Laplace approximates the log-density by a quadratic around the peak:

$$\log p(x) \approx \log p(\hat{x}) - \tfrac{1}{2} H (x - \hat{x})^2.$$

Exponentiating this quadratic gives a Gaussian.

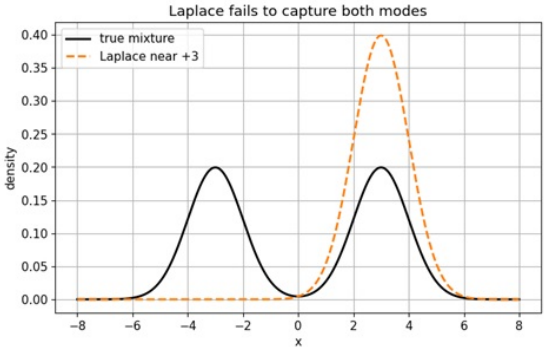


Failure case: multimodal posterior

If the target has **multiple peaks**, a single Gaussian around one peak cannot capture the whole distribution.
Example mixture:

$$p(x) = \tfrac{1}{2}\,\mathcal{N}(-3, 1) + \tfrac{1}{2}\,\mathcal{N}(3, 1).$$

Laplace around the right peak captures only that mode and ignores the other.
Takeaway: Laplace works best for *unimodal, near-Gaussian* posteriors.



Key takeaways

- **Hessian** = curvature (2nd derivatives); off-diagonals mean coupling/correlation.
- **Laplace**: approximate $p(\theta \mid y)$ by $\mathcal{N}(\hat{\theta}, H^{-1})$.
- Works best when posterior is **unimodal** and **locally Gaussian**.
- Can be misleading for **multimodal**, **skewed**, or **heavy-tailed** posteriors.
- For more accurate uncertainty, use **HMC/NUTS** (NumPyro/Stan) — Laplace is fast and interpretable.