UNIVERSITÀ DEGLI STUDI DI TRIESTE

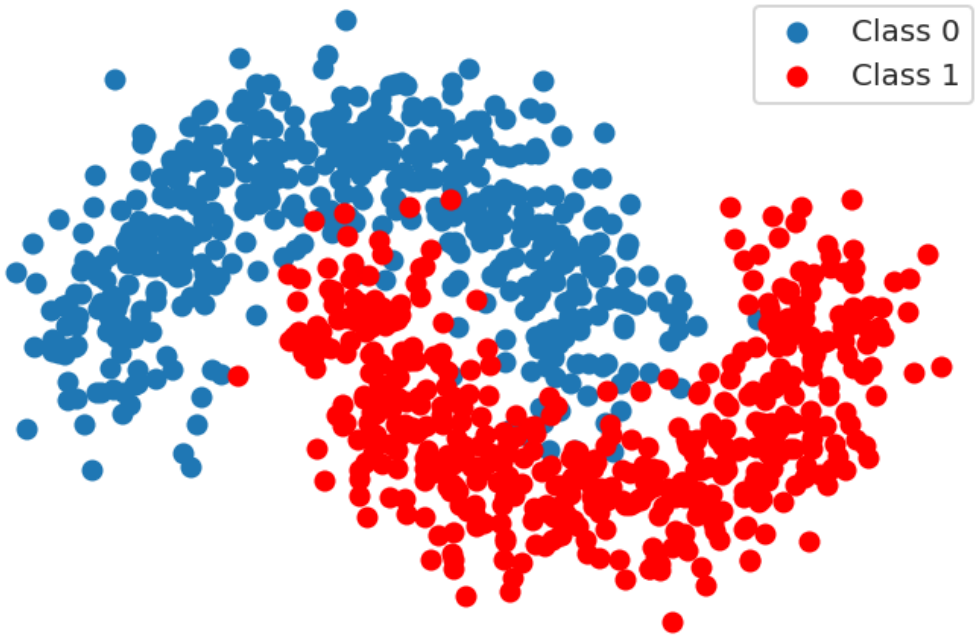
DATA SCIENCE & SCIENTIFIC COMPUTING

Predicting Smart Grid Stability with Probabilistic Machine Learning Models.

Baurice Nafack

Fatma Moustafa

July 18, 2022



Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:
  - 'tau1' to 'tau4': the reaction time of each network participant.

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:
  - 'tau1' to 'tau4': the reaction time of each network participant.
  - 'p1' to 'p4': nominal power produced (positive) or consumed (negative) by each network participant.

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:
  - 'tau1' to 'tau4': the reaction time of each network participant.
  - 'p1' to 'p4': nominal power produced (positive) or consumed (negative) by each network participant.
  - 'g1' to 'g4': price elasticity coefficient for each network participant.

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:
  - 'tau1' to 'tau4': the reaction time of each network participant.
  - 'p1' to 'p4': nominal power produced (positive) or consumed (negative) by each network participant.
  - 'g1' to 'g4': price elasticity coefficient for each network participant.
- Dependent variables:

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:
  - 'tau1' to 'tau4': the reaction time of each network participant.
  - 'p1' to 'p4': nominal power produced (positive) or consumed (negative) by each network participant.
  - 'g1' to 'g4': price elasticity coefficient for each network participant.
- Dependent variables:
  - 'stab': if positive, the system is linearly unstable; if negative, linearly stable.

Dataset Description

The dataset consists of results taken from grid stability simulations done on a 4-node network.

- It has 12 predictive features and two dependent variables.
- It has 60000 samples.
- Predictive features:
  - 'tau1' to 'tau4': the reaction time of each network participant.
  - 'p1' to 'p4': nominal power produced (positive) or consumed (negative) by each network participant.
  - 'g1' to 'g4': price elasticity coefficient for each network participant.
- Dependent variables:
  - 'stab': if positive, the system is linearly unstable; if negative, linearly stable.
  - 'stabf': a categorical (binary) label ('stable' or 'unstable').

- A Gaussian process (GP) for regression is a random process where any point $\mathrm{x} \in \mathbb{R}^d$ is assigned a random variable $f(\mathrm{x})$ and where the joint distribution of a finite number of these variables $p(f(\mathrm{x}_1), ..., f(\mathrm{x}_N))$ is itself Gaussian:

$$p(\mathrm{f} \mid \mathrm{X}) = \mathcal{N}(\mathrm{f} \mid \boldsymbol{\mu}, \mathrm{K}) \tag{1}$$

- A Gaussian process (GP) for regression is a random process where any point $\mathrm{x} \in \mathbb{R}^d$ is assigned a random variable $f(\mathrm{x})$ and where the joint distribution of a finite number of these variables $p(f(\mathrm{x}_1), ..., f(\mathrm{x}_N))$ is itself Gaussian:

$$p(\mathrm{f} \mid \mathrm{X}) = \mathcal{N}(\mathrm{f} \mid \boldsymbol{\mu}, \mathrm{K}) \tag{1}$$

- A GP is a prior over functions whose shape (smoothness, ...) is defined by $\mathrm{K} = \kappa(\mathrm{X}, \mathrm{X})$ where κ is a parametric kernel function.

Gaussian processes for classification: Regression recap

- A Gaussian process (GP) for regression is a random process where any point $\mathrm{x} \in \mathbb{R}^d$ is assigned a random variable $f(\mathrm{x})$ and where the joint distribution of a finite number of these variables $p(f(\mathrm{x}_1), ..., f(\mathrm{x}_N))$ is itself Gaussian:

$$p(\mathrm{f} \mid \mathrm{X}) = \mathcal{N}(\mathrm{f} \mid \boldsymbol{\mu}, \mathrm{K}) \tag{1}$$

- A GP is a prior over functions whose shape (smoothness, ...) is defined by $\mathrm{K} = \kappa(\mathrm{X}, \mathrm{X})$ where κ is a parametric kernel function.
- Given observed noisy function values y at points X we want to predict a noise-free function value f_* at point x_*. The joint distribution of observed values y and prediction f_* is also a Gaussian:

- $$p(\mathrm{y}, f_* \mid \mathrm{X}, \mathrm{x}_*) = \mathcal{N}\left(\begin{pmatrix}\mathrm{y} \\ f_*\end{pmatrix} \middle| \, 0, \begin{pmatrix}\mathrm{K}_y & \mathrm{k}_* \\ \mathrm{k}_*^T & k_{**}\end{pmatrix}\right) \tag{2}$$

  where $\mathrm{K}_y = \mathrm{K} + \sigma_y^2 \mathrm{I}$, $\mathrm{k}_* = \kappa(\mathrm{X}, \mathrm{x}_*)$ and $k_{**} = \kappa(\mathrm{x}_*, \mathrm{x}_*)$. σ_y^2 models noise in the observed function values y.

- $$p(\mathrm{y}, f_* \mid \mathrm{X}, \mathrm{x}_*) = \mathcal{N}\left(\begin{pmatrix}\mathrm{y} \\ f_*\end{pmatrix} \middle| \, 0, \begin{pmatrix}\mathrm{K}_y & \mathrm{k}_* \\ \mathrm{k}_*^T & k_{**}\end{pmatrix}\right) \tag{2}$$

  where $\mathrm{K}_y = \mathrm{K} + \sigma_y^2 \mathrm{I}$, $\mathrm{k}_* = \kappa(\mathrm{X}, \mathrm{x}_*)$ and $k_{**} = \kappa(\mathrm{x}_*, \mathrm{x}_*)$. σ_y^2 models noise in the observed function values y.
- Turning the joint distribution (2) into a conditional distribution we obtain a predictive distribution:

$$p(f_* \mid \mathrm{x}_*, \mathrm{X}, \mathrm{y}) = \mathcal{N}(f_* \mid \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{3}$$

  with

$$\begin{aligned} \boldsymbol{\mu}_* &= \mathrm{k}_*^T \mathrm{K}_y^{-1} \mathrm{y} \\ \boldsymbol{\Sigma}_* &= k_{**} - \mathrm{k}_*^T \mathrm{K}_y^{-1} \mathrm{k}_* \end{aligned} \tag{4}$$

Gaussian processes for classification: Binary classification

In context of binary classification,

- We have a discrete target variable $t \in \{0, 1\}$ that follows a Bernoulli distribution.

Gaussian processes for classification: Binary classification

In context of binary classification,

- We have a discrete target variable $t \in \{0, 1\}$ that follows a Bernoulli distribution.
- We are interested in the probability $p(t = 1 \mid x) = \sigma(f(x))$ where σ is the logistic sigmoid function taking logit $x \in \mathbb{R}$ as argument. $p(t = 0 \mid x)$ is given by $1 - p(t = 1 \mid x)$.

In context of binary classification,

- We have a discrete target variable $t \in \{0, 1\}$ that follows a Bernoulli distribution.
- We are interested in the probability $p(t = 1 \mid x) = \sigma(f(x))$ where σ is the logistic sigmoid function taking logit $x \in \mathbb{R}$ as argument. $p(t = 0 \mid x)$ is given by $1 - p(t = 1 \mid x)$.
- The predictive distribution can be defined as:

$$p(t_* = 1 \mid \mathrm{t}, x_*, X) = \int p(t_* = 1 \mid f_*) p(f_* \mid \mathrm{t}, x_*, X) df_* \quad (5)$$

$$p(f_* \mid \mathrm{t}, x_*, X) = \int p(f_* \mid \mathrm{f}, x_*, t) p(\mathrm{f} \mid \mathrm{t}, X) d\mathrm{f} \quad (6)$$

Where $p(\mathrm{f} \mid \mathrm{t}, X)$ can be approximated with a Gaussian distribution $q(\mathrm{f})$ using the Laplace approximation, minimising the KL divergence.

$$q(\mathsf{f}) = \mathcal{N}(\mathsf{f} \mid \hat{\mathsf{f}}, \mathsf{H}^{-1}) \tag{7}$$

where $\mathsf{H} = \mathsf{W} + \mathsf{K}_f^{-1}$. W is a diagonal matrix with elements $\sigma(f_n)(1-\sigma(f_n))$ with f_n being the elements of f. Written in vector notation the diagonal is $\boldsymbol{\sigma}(1-\boldsymbol{\sigma})$.

$$q(\mathsf{f}) = \mathcal{N}(\mathsf{f} \mid \hat{\mathsf{f}}, \mathsf{H}^{-1}) \tag{7}$$

where $\mathsf{H} = \mathsf{W} + \mathsf{K}_f^{-1}$. W is a diagonal matrix with elements $\sigma(f_n)(1-\sigma(f_n))$ with f_n being the elements of f. Written in vector notation the diagonal is $\boldsymbol{\sigma}(1-\boldsymbol{\sigma})$.

- The mean $\hat{\mathsf{f}}$ can be obtained iteratively with the following update equation:

$$\mathsf{f}^{\text{new}} = \mathsf{K}_f(\mathsf{I} + \mathsf{W}\mathsf{K}_f)^{-1}(\mathsf{t} - \boldsymbol{\sigma} + \mathsf{W}\mathsf{f}) \tag{8}$$

  At convergence $\hat{\mathsf{f}} = \mathsf{f}^{\text{new}}$.

$$q(\mathsf{f}) = \mathcal{N}(\mathsf{f} \mid \hat{\mathsf{f}}, \mathsf{H}^{-1}) \tag{7}$$

where $\mathsf{H} = \mathsf{W} + \mathsf{K}_f^{-1}$. W is a diagonal matrix with elements $\sigma(f_n)(1-\sigma(f_n))$ with f_n being the elements of f. Written in vector notation the diagonal is $\boldsymbol{\sigma}(1-\boldsymbol{\sigma})$.

- The mean $\hat{\mathsf{f}}$ can be obtained iteratively with the following update equation:

$$\mathsf{f}^{\text{new}} = \mathsf{K}_f(\mathsf{I} + \mathsf{W}\mathsf{K}_f)^{-1}(\mathsf{t} - \boldsymbol{\sigma} + \mathsf{W}\mathsf{f}) \tag{8}$$

  At convergence $\hat{\mathsf{f}} = \mathsf{f}^{\text{new}}$.

- The Gaussian approximation of $p(f_* \mid \mathsf{t})$ is therefore given by

$$p(f_* \mid \mathsf{t}) \approx \mathcal{N}(f_* \mid \mu_{f_*}, \sigma_{f_*}^2) \tag{9}$$

with

$$\begin{aligned} \mu_{f_*} &= \mathrm{k}_*^T(\mathrm{t} - \boldsymbol{\sigma}) \\ \sigma_{f_*}^2 &= k_{**} - \mathrm{k}_*^T(\mathrm{W}^{-1} + \mathrm{K}_f)^{-1}\mathrm{k}_* \end{aligned} \tag{10}$$

Finally, $p(t_* = 1 \mid f_*)$ in Equation (4) is approximate with the inverse probit function $\Phi(f_*)$ so that the predictive distribution can be approximated with:

$$p(t_* = 1 \mid \mathrm{t}) \approx \sigma(\mu_{f_*}(1 + \pi\sigma_{f_*}^2/8)^{-1/2}) \tag{11}$$

Gaussian Naive Bayes

- It is a Naive Bayes classifier with the likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(x_i - \mu_C)^2}{2\sigma_C^2}\right) \tag{12}$$

Gaussian Naive Bayes

- It is a Naive Bayes classifier with the likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(x_i - \mu_C)^2}{2\sigma_C^2}\right) \tag{12}$$

- The parameters σ_C and μ_C are estimated using maximum likelihood

Bayesian Logistic Regression

- In logistic regression, parameters (weight and bias) are learned by maximum likelihood

Bayesian Logistic Regression

- In logistic regression, parameters (weight and bias) are learned by maximum likelihood
- Prior on parameters w and b

Bayesian Logistic Regression

- In logistic regression, parameters (weight and bias) are learned by maximum likelihood
- Prior on parameters w and b
- Inference $\rightarrow$ Use Stochastic Variational Inference (SVI).

Bayesian Logistic Regression

- In logistic regression, parameters (weight and bias) are learned by maximum likelihood
- Prior on parameters w and b
- Inference $\rightarrow$ Use Stochastic Variational Inference (SVI).
- Family of distributions

Bayesian Logistic Regression

- In logistic regression, parameters (weight and bias) are learned by maximum likelihood
- Prior on parameters w and b
- Inference $\rightarrow$ Use Stochastic Variational Inference (SVI).
- Family of distributions
- Find an approximate posterior distribution from this family that has the lowest KL divergence from the true posterior.

Stochastic Variational Inference

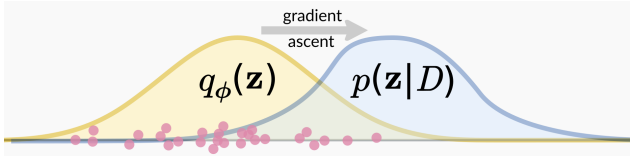
- KL divergence: Similarity between the posterior distribution and approximate posterior distribution

Stochastic Variational Inference

- KL divergence: Similarity between the posterior distribution and approximate posterior distribution
- Inference $\rightarrow$ Optimization problem

Stochastic Variational Inference

- KL divergence: Similarity between the posterior distribution and approximate posterior distribution
- Inference $\rightarrow$ Optimization problem
- Objective: minimize the KL-divergence from the approximate posterior to the true posterior



Variational Inference

$$\begin{aligned} \mathrm{KL}[\mathrm{q}_\phi(\mathrm{z})\|\mathrm{p}(\mathrm{z}|\mathrm{D})] &= \int q_\phi(\mathrm{z}) \log \frac{q_\phi(\mathrm{z})}{p(\mathrm{z}|\mathrm{D})} \\ &= \mathbb{E}_{\mathrm{q}_\phi(\mathrm{z})} \left[\log \mathrm{q}_\phi(\mathrm{z}) - \log \mathrm{p}(\mathrm{z}|\mathrm{D})\right] \end{aligned} \tag{13}$$

- We can't directly optimize (minimize) the KL divergence, instead we optimize (maximize) the ELBO: Evidence Lower Bound with respect to the variational parameters

$$\mathrm{ELBO} \equiv \mathbb{E}_{q_\phi(\mathsf{z})}\left[\log p_\theta(\mathsf{z}, \mathsf{D}) - \log q_\phi(\mathsf{z})\right] \tag{14}$$

$$\mathbb{E}_{\mathrm{q}_\phi(\mathsf{z})}\left[\log \mathrm{q}_\phi(\mathsf{z}) - \log \mathrm{p}(\mathsf{z}|\mathsf{D})\right] = \mathbb{E}_{q_\phi(\mathsf{z})}\left[\log p_\theta(\mathsf{z}, \mathsf{D}) - \log q_\phi(\mathsf{z})\right] + \left[\log p(\mathsf{D})\right] \tag{15}$$

$$\text{KL divergence} = -\mathrm{ELBO} + \mathrm{Evidence} \tag{16}$$

Bayesian Neural Networks

- Artificial Neural Network: train network parameters as point estimates

Bayesian Neural Networks

- Artificial Neural Network: train network parameters as point estimates
- Bayesian Neural Network:

Bayesian Neural Networks

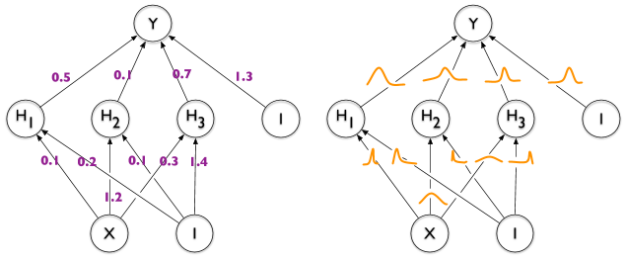
- Artificial Neural Network: train network parameters as point estimates
- Bayesian Neural Network:
  - Model uncertainty in network parameters

Bayesian Neural Networks

- Artificial Neural Network: train network parameters as point estimates
- Bayesian Neural Network:
  - Model uncertainty in network parameters
  - Prior over parameters represented with probability distribution

Bayesian Neural Networks

- Artificial Neural Network: train network parameters as point estimates
- Bayesian Neural Network:
  - Model uncertainty in network parameters
  - Prior over parameters represented with probability distribution
  - Posterior of parameters given dataset is approximated using the stochastic variational inference



Weights in ANN Vs. BNN

Experimental Methods

- we split the data into test data, training data, and validation data to train the BNN method and others. The validation data will be used to track the bias-variance tradeoff in our models.

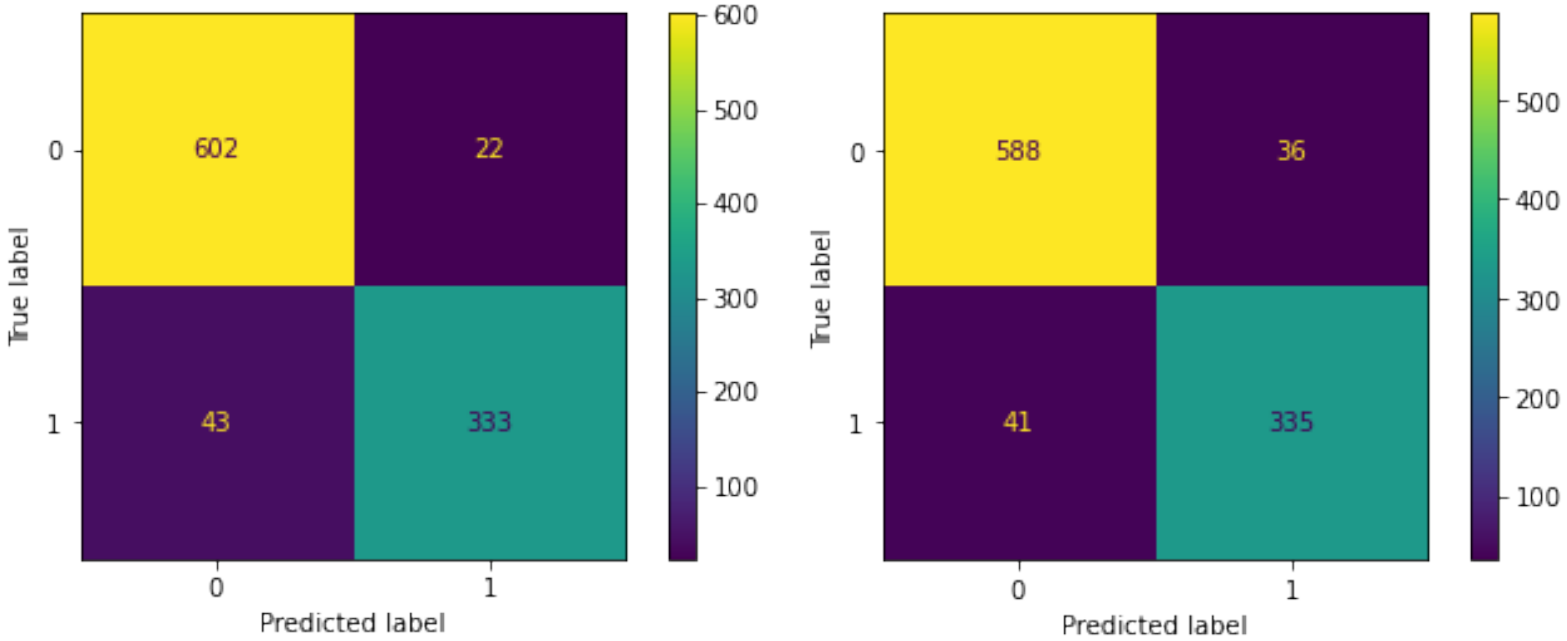
Experimental Methods

- we split the data into test data, training data, and validation data to train the BNN method and others. The validation data will be used to track the bias-variance tradeoff in our models.
- To train the Gaussian Process classifier, we select the first 1000 samples and the last 1000 samples for the test. the same test data will be used to evaluate all of our models.

Experimental Methods

- we split the data into test data, training data, and validation data to train the BNN method and others. The validation data will be used to track the bias-variance tradeoff in our models.
- To train the Gaussian Process classifier, we select the first 1000 samples and the last 1000 samples for the test. the same test data will be used to evaluate all of our models.
- We perform Cross-Validation to choose the best kernel for the Gaussian process classification.

Gaussian Process result



a) RBF kernel b) Rational Quadratic kernel

Figure: Confusion matrix

Gaussian Process Performance Metrics

```
Best Mean Accuracy: 0.937
Best Config: {'kernel': 1**2 * RationalQuadratic(alpha=1, length_scale=1)}
>0.628 with: {'kernel': 1**2 * RBF(length_scale=1)}
>nan with: {'kernel': 1**2 * DotProduct(sigma_0=1)}
>0.905 with: {'kernel': 1**2 * Matern(length_scale=1, nu=1.5)}
>0.937 with: {'kernel': 1**2 * RationalQuadratic(alpha=1, length_scale=1)}
>0.628 with: {'kernel': 1**2 * WhiteKernel(noise_level=1)}
```

a) Cross-Validation

	precision	recall	f1-score	support
unstable	0.93	0.96	0.95	624
stable	0.94	0.89	0.91	376
accuracy			0.94	1000
macro avg	0.94	0.93	0.93	1000
weighted avg	0.94	0.94	0.93	1000

b)Classification report using RBF kernel

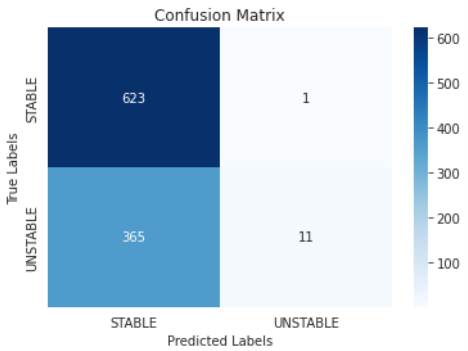
	precision	recall	f1-score	support
unstable	0.93	0.94	0.94	624
stable	0.90	0.89	0.90	376
accuracy			0.92	1000
macro avg	0.92	0.92	0.92	1000
weighted avg	0.92	0.92	0.92	1000

c)Classification report using RQ kernel

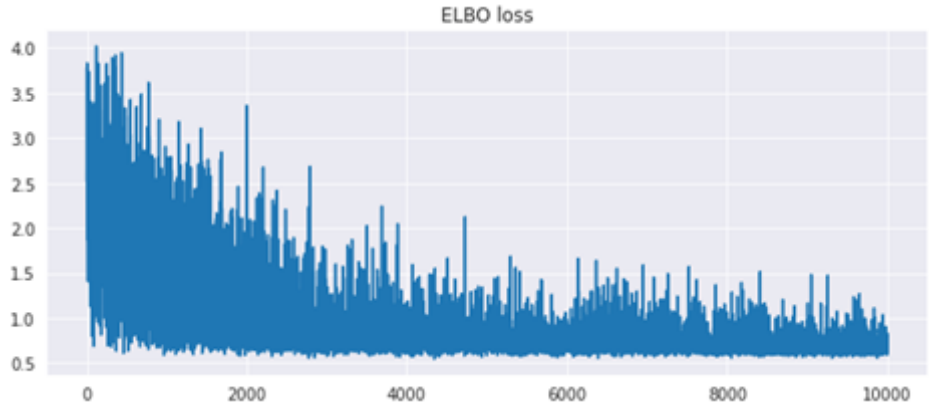
Bayesian Logistic Regression Result

```
Classification Report:
              precision    recall  f1-score   support

           1     0.9167    0.0293    0.0567       376
           0     0.6306    0.9984    0.7730       624

    accuracy                         0.6340      1000
   macro avg     0.7736    0.5138    0.4148      1000
weighted avg     0.7381    0.6340    0.5036      1000
```


a) Classification Report



b)Confusion Matrix

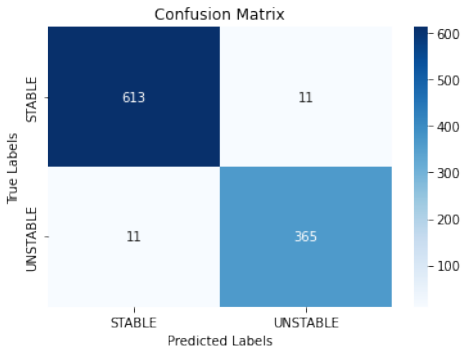


c) ELBO loss

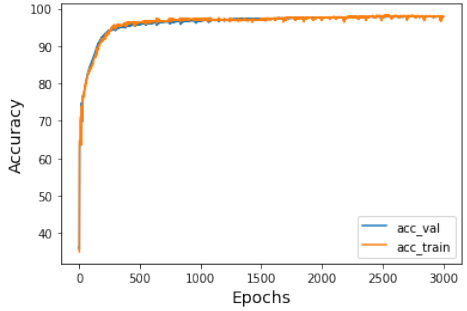
Artificial Neural Network Result

```
Classification Report:
              precision    recall  f1-score   support

           1     0.9707    0.9707    0.9707       376
           0     0.9824    0.9824    0.9824       624

    accuracy                         0.9780      1000
   macro avg     0.9766    0.9766    0.9766      1000
weighted avg     0.9780    0.9780    0.9780      1000
```

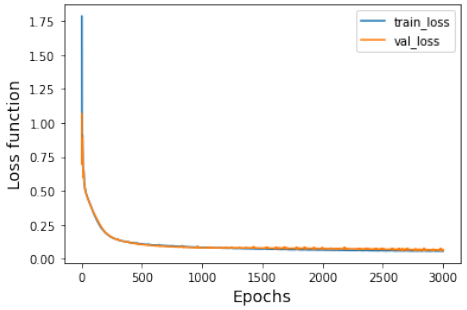
a) Classification Report



b)Confusion Matrix



c)Accuracy



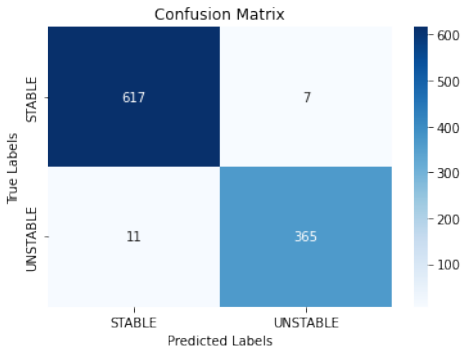
d) Loss function

Figure: 1 hidden layer

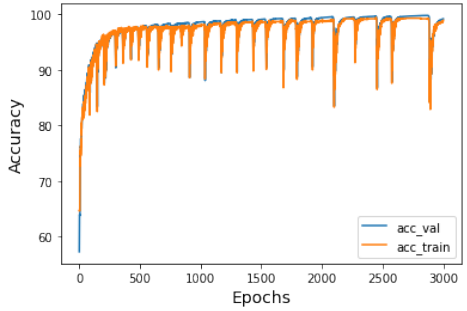
Artificial Neural Network Result

```
Classification Report:
              precision    recall  f1-score   support

           1     0.9812    0.9707    0.9759       376
           0     0.9825    0.9888    0.9856       624

    accuracy                         0.9820      1000
   macro avg     0.9818    0.9798    0.9808      1000
weighted avg     0.9820    0.9820    0.9820      1000
```

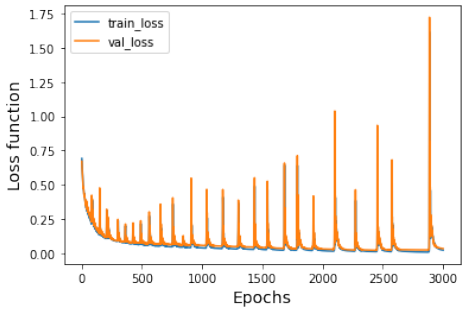
a) Classification Report



b)Confusion Matrix



c)Accuracy



d) Loss function

Figure: 3 hidden layers

Parameters of Bayesian Neural Network

```
[191…
model = nn.Sequential(
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=12, out_features=100),
    nn.ReLU(),
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=100, out_features=2),
)
```

```
[192…
ce_loss = nn.CrossEntropyLoss()
kl_loss = bnn.BKLLoss(reduction='mean', last_layer_only=False)
kl_weight = 0.01

optimizer = optim.Adam(model.parameters(), lr=0.005)
```

a) 1 hidden layer

```
6]:
model = nn.Sequential(
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=12, out_features=200),
    nn.ReLU(),
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=200, out_features=100),
    nn.ReLU(),
     bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=100, out_features=50),
    nn.ReLU(),
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=50, out_features=2),
)
```

```
7]:
ce_loss = nn.CrossEntropyLoss()
kl_loss = bnn.BKLLoss(reduction='mean', last_layer_only=False)
kl_weight = 0.01

optimizer = optim.Adam(model.parameters(), lr=0.005)
```

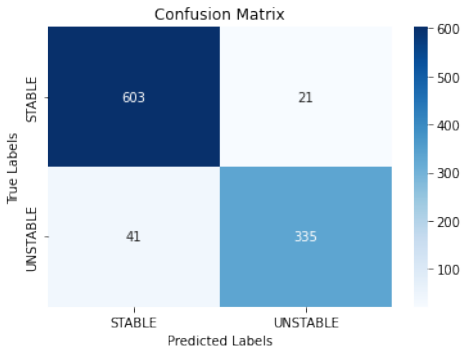
b) 3 hidden layers

Figure: BNN parameters

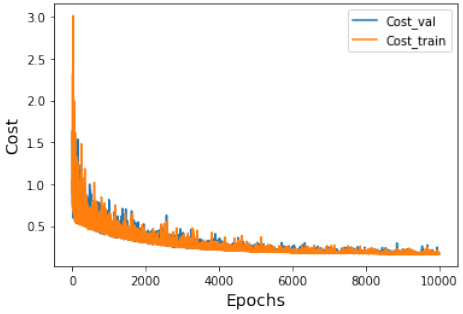
Bayesian Neural Network Result

```
Classification Report:
              precision    recall  f1-score   support

           1     0.9410    0.8910    0.9153       376
           0     0.9363    0.9663    0.9511       624

    accuracy                         0.9380      1000
   macro avg     0.9387    0.9287    0.9332      1000
weighted avg     0.9381    0.9380    0.9376      1000
```

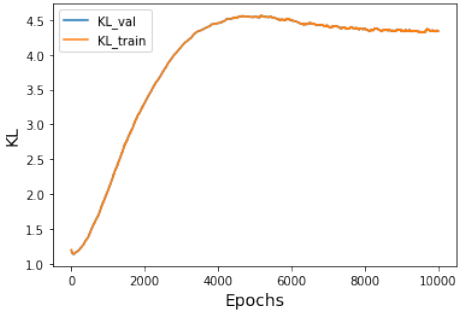
a) Classification Report



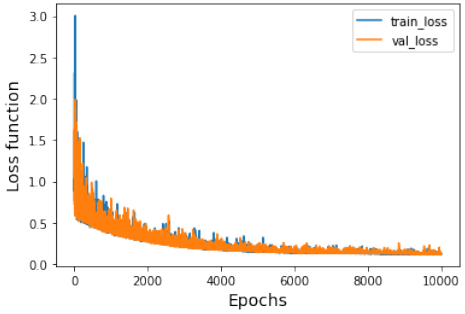
b)Confusion Matrix



c)Cost function



d)KL

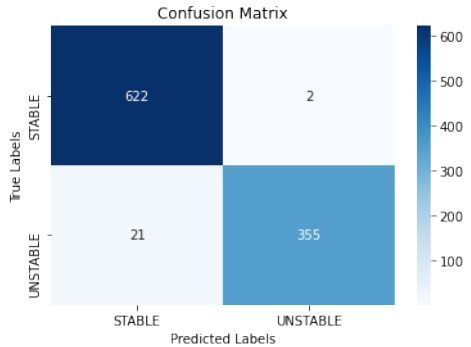


e) Loss function

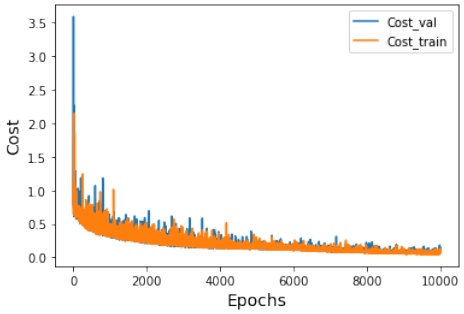
Figure: 1 hidden layer

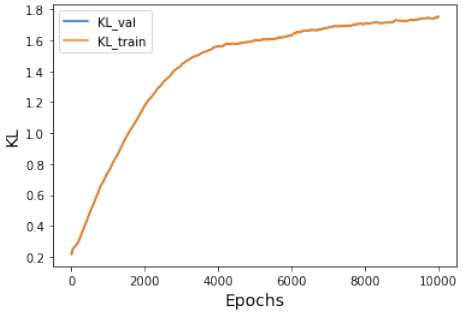
Bayesian Neural Network Result

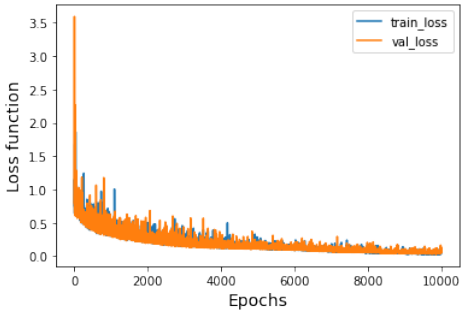
```
Classification Report:
              precision    recall  f1-score   support

           1     0.9944    0.9441    0.9686       376
           0     0.9673    0.9968    0.9818       624

    accuracy                         0.9770      1000
   macro avg     0.9809    0.9705    0.9752      1000
weighted avg     0.9775    0.9770    0.9769      1000
```



a) Classification Report b)Confusion Matrix







c)Cost function d)KL e) Loss function

Figure: 3 hidden layers

Comparison of Predictive Accuracy using Different Techniques

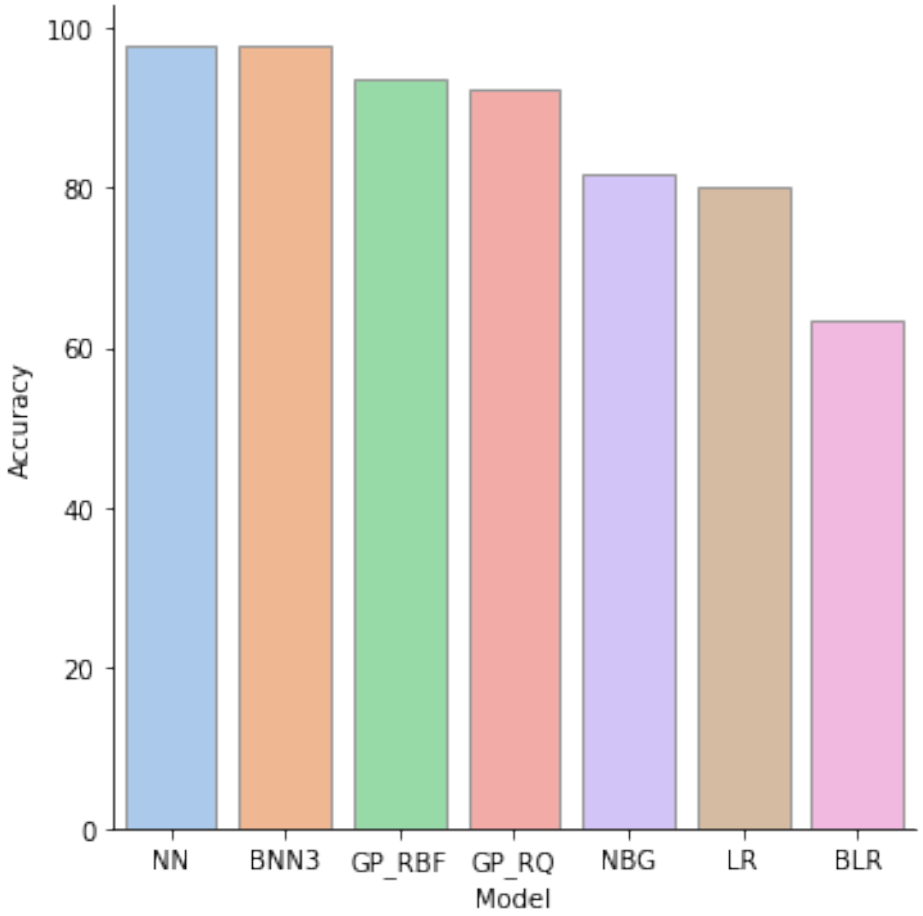


Figure: Comparison of Predictive Accuracy using Different Techniques

Comparison of Predictive Accuracy using Different Techniques

Table: Comparison of Predictive Accuracy using Different Techniques

Prediction model	Test accuracy %
GP RBF	93.5
GP RQ	92.3
NN with one hidden layer	97.8
NN with 3 hidden layer	98.2
BNN with one hidden layer	$(90.5, 95.3)$
BNN with 3 hidden layer	$(95.2, 98.6)$
NB Gaussian	81.7
LR	80
BLR	63.4

Comparison of Predictive Accuracy using Different KL weight

Table: Comparison of predictive accuracy for different KL weight (inspire from Bowman et al. 2016 work) using BNN with 1 hidden layers.

KL weight	Test accuracy %	epoch
1	$(37.6, 74.7)$	3000
0.5	$(58.3, 77.5)$	3000
0.05	$(85.5, 93.0)$	3000
0.01	$(82.8, 92.4)$	3000
0.01	$(90.5, 95.3)$	10000

- For higher values of KL weight, we obtained a decreasing KL value. However, we get a low accuracy and the model tends to overfit.

Comparison of Predictive Accuracy using Different KL weight

Table: Comparison of predictive accuracy for different KL weight (inspire from Bowman et al. 2016 work) using BNN with 1 hidden layers.

KL weight	Test accuracy %	epoch
1	$(37.6, 74.7)$	3000
0.5	$(58.3, 77.5)$	3000
0.05	$(85.5, 93.0)$	3000
0.01	$(82.8, 92.4)$	3000
0.01	$(90.5, 95.3)$	10000

- For higher values of KL weight, we obtained a decreasing KL value. However, we get a low accuracy and the model tends to overfit.
- As we decrease the KL weight , we get an increasing KL value. The accuracy gets better, and our model learns from the data.

Comparison of Predictive Accuracy to previous work

The following table done by Paulo Breviglieri 2021 present the comparison of prediction accuracy using different techniques.

Prediction method	Accuracy (%)	Used system
XGBoost [23]	97.8	IEEE 39-bus test system
Bayesian rate (BR) [24]	91.6	IEEE 39-bus test system
DT [25]	90.3	IEEE 39-bus test system
CNN [26]	89.22	IEEE 118-bus and IEEE 145-bus systems
Long Short Term Memory (LSTM) [27]	99.98	IEEE 39-bus test system and 162 and 145 bus systems
FNN-based transient stability predictor [28]	99.2	IEEE 39-bus test system
CART [4]	80.0	4-node star architecture
Proposed	99.62	4-node star architecture

Conclusion and future work

- The comparative analysis shows that the Bayesian Neural Networks performs well on the test data compared to other probabilistic machine learning models.

Conclusion and future work

- The comparative analysis shows that the Bayesian Neural Networks performs well on the test data compared to other probabilistic machine learning models.
- The Gaussian processes method is not as computationally expensive as other learning methods, however, it does not scale well with large datasets.

Conclusion and future work

- The comparative analysis shows that the Bayesian Neural Networks performs well on the test data compared to other probabilistic machine learning models.
- The Gaussian processes method is not as computationally expensive as other learning methods, however, it does not scale well with large datasets.
- The hyperparameters in the Bayesian Neural Networks plays an important role on how the model learn the data. Analysis of hyperparameters tuning would give more insight on the power of Bayesian Neural Networks.

Conclusion and future work

- The comparative analysis shows that the Bayesian Neural Networks performs well on the test data compared to other probabilistic machine learning models.
- The Gaussian processes method is not as computationally expensive as other learning methods, however, it does not scale well with large datasets.
- The hyperparameters in the Bayesian Neural Networks plays an important role on how the model learn the data. Analysis of hyperparameters tuning would give more insight on the power of Bayesian Neural Networks.
- Implementation could be done on an HPC system which could help in obtaining reliable and more accurate results.

Bibliography I

- Bowman, Samuel R. et al. (2016). "Generating Sentences from a Continuous Space". In.
- Paulo Breviglieri Türkücan Erdem, Süleyman Eken (2021). "Predicting Smart Grid Stability with Optimized Deep Models". In: *SN Computer Science*, pp. 293–298. DOI: https://doi.org/10.1007/s42979-021-00463-5.

Thank you for your kind attention