Bayesian Perceptron:
Towards fully Bayesian Neural Networks

Marco F. Huber

Abstract

- NNs usually only provide point estimates without systematically quantifying corresponding uncertainties.
- In this paper a novel approach towards fully Bayesian NNs is proposed, where training and predictions of a perceptron are performed within the Bayesian inference framework in closed-form. **The weights and the predictions of the perceptron are considered Gaussian random variables**.
- Analytical expressions for predicting the perceptron's output and for learning the weights are provided for commonly used activation functions like sigmoid or ReLU. This approach requires **no computationally expensive** gradient calculations and further allows **sequential learning.**

Introduction

The weight distribution represented by means of a Gaussian. An approach for estimating the parameters of this distribution, i.e., its mean and covariance

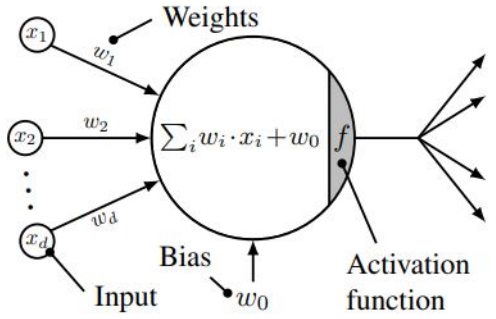


Fig. 1: Perceptron, building block of artificial NNs.

• Closed-form propagation of the parameters of the perceptron's output distribution for commonly used activation functions like sigmoid or ReLU.

• Closed-form estimation of the parameters of the weight distribution for given training data without the need of GD or MC sampling. Instead, the Bayesian inference paradigm is strictly adhered to.

• Training data can be processed sequentially, while common Bayesian NN approaches require batch processing.

Problem Formulation

For training purposes a training dataset $D = \{x_i , y_i\}^n_{i=1}$

comprising n i.i.d. training instances (x_i , y_i), with inputs/features $x_i = [x_{i,1} \ldots x_{i,d}]^T \in R^d$ and outputs $y_i \in R$, is given.

Two Goals:

1. Predictive P(y|x,D)
2. Weight training P(w|D)

Problem setting:

$$a = \underline{x}^{\mathrm{T}} \cdot \underline{w} + w_0$$
$$y = f(a)$$

$$f(a) = s(a) \triangleq \frac{1}{1 + \mathrm{e}^{-a}}$$
$$f(a) = \tanh(a) = 2 \cdot s(a) + 1$$
$$f(a) = \max(\alpha \cdot x, \beta \cdot x)$$

Bayesian Perceptron - Forward Pass

Assumption: Predictive PDF can be approximated well by means of a parametric distribution, particularly a Gaussian, i.e.,

$$p(y|\underline{x}, \mathcal{D}) \approx \mathcal{N}(y; \mu_y, \sigma_y^2)$$

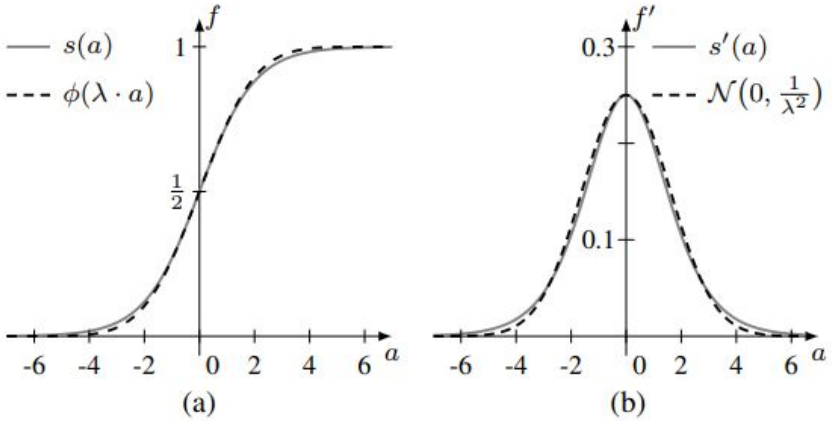
i.e., only need to calculate:

$$\mu_y = \mathrm{E}\{y\} = \mathrm{E}\{f(a)\} \,, \tag{4}$$
$$\sigma_y^2 = \mathrm{E}\left\{(y - \mu_y)^2\right\} = \mathrm{E}\left\{f(a)^2\right\} - \mu_y^2 \,. \tag{5}$$

For w being Gaussian, a also gaussian:

$$\mu_a = \underline{x}^{\mathrm{T}} \cdot \underline{\mu}^w \,, \tag{6}$$
$$\sigma_a^2 = \underline{x}^{\mathrm{T}} \cdot \mathbf{C}^w \cdot \underline{x} \,, \tag{7}$$

Forward Pass - Sigmoid Case



$$\mu_y = \mathrm{E}\{s(a)\} = \int_{\mathbb{R}} s(a) \cdot \mathcal{N}\left(a; \mu_a, \sigma_a^2\right) \mathrm{d}a$$
$$\approx \int_{\mathbb{R}} \phi(\lambda \cdot a) \cdot \mathcal{N}\left(a; \mu_a, \sigma_a^2\right) \mathrm{d}a$$
$$\overset{(a)}{=} \phi\left(\frac{\lambda \cdot \mu_a}{t}\right) \overset{(b)}{\approx} s\left(\frac{\mu_a}{t}\right) \ , \qquad (8)$$

Forward Pass - Sigmoid Case Cont'

$$\begin{aligned}\sigma_y^2 &= \mathrm{E}\left\{s(a)^2\right\} - \mu_y^2 \\ &= \mathrm{E}\left\{s(a)^2 + s(a) - s(a)\right\} - \mu_y^2 \\ &\overset{(c)}{=} \mathrm{E}\left\{s(a) - \underbrace{s(a)\cdot(1-s(a))}_{=s'(a)}\right\} - \mu_y^2 \\ &= \mu_y - \mu_y^2 - \mathrm{E}\left\{s'(a)\right\}\,, \end{aligned} \tag{9}$$

$$\sigma_y^2 \approx \mu_y \cdot (1-\mu_y)\cdot\left(1-\tfrac{1}{t}\right)\,, \tag{11}$$

$$t \triangleq \sqrt{1+\lambda^2\cdot\sigma_a^2}$$

$$\begin{aligned}\mathrm{E}\left\{s'(a)\right\} &\approx \mathrm{E}\left\{\mathcal{N}\left(a;0,\tfrac{1}{\lambda^2}\right)\right\} \\ &= \int_{\mathbb{R}} \mathcal{N}\left(a;0,\tfrac{1}{\lambda^2}\right)\cdot\mathcal{N}\left(a;\mu_a,\sigma_a^2\right)\mathrm{d}a \\ &\overset{(d)}{=} \mathcal{N}\left(0;\mu_a,\tfrac{1}{\lambda^2}+\sigma_a^2\right) \\ &= \tfrac{1}{t}\cdot\mathcal{N}\left(\tfrac{\mu_a}{t};0,\tfrac{1}{\lambda^2}\right) \overset{(e)}{\approx} \tfrac{1}{t}\cdot s'\left(\tfrac{\mu_a}{t}\right)\,, \end{aligned} \tag{10}$$

Forward Pass - Piece-wise Linear Case

$p_x(a) \triangleq p(a|\underline{x}, \mathcal{D}) = \mathcal{N}(a; \mu_a, \sigma_a^2)$ it holds that

$$
\begin{aligned}
\mu_y &= \mathrm{E}\{f(a)\} \\
&= \int_{\mathbb{R}} \max(\alpha \cdot a, \beta \cdot a) \cdot p_x(a)\, \mathrm{d}a \\
&\overset{(f)}{=} \alpha \cdot \int_{-\infty}^{0} a \cdot p_x(a)\, \mathrm{d}a + \beta \cdot \int_{0}^{\infty} a \cdot p_x(a)\, \mathrm{d}a \\
&= \alpha \cdot \underbrace{\int_{\mathbb{R}} a \cdot p(a|\underline{x})\, \mathrm{d}a}_{=\mu_a \triangleq \mathrm{E}_1} + (\beta - \alpha) \cdot \int_{0}^{\infty} a \cdot p_x(a)\, \mathrm{d}a \\
&= \alpha \cdot \mathrm{E}_1 + (\beta - \alpha) \cdot \left(\mathrm{E}_1 \cdot \phi\left(\frac{\mu_a}{\sigma_a}\right) + p_a \right), \qquad (12)
\end{aligned}
$$

Forward Pass - Piece-wise Linear Case Cont'

$$
\begin{aligned}
\sigma_y^2 &= \mathrm{E}\{f(a)^2\} - \mu_y^2 \\
&= \int_{\mathbb{R}} \left(\max(\alpha \cdot a, \beta \cdot a)\right)^2 \cdot p_x(a)\,\mathrm{d}a - \mu_y^2 \\
&= \alpha^2 \cdot \int_{-\infty}^{0} a^2 \cdot p_x(a)\,\mathrm{d}a + \beta^2 \cdot \int_{0}^{\infty} a^2 \cdot p_x(a)\,\mathrm{d}a - \mu_y^2 \\
&= \alpha^2 \cdot \underbrace{\int_{\mathbb{R}} a^2 \cdot p_x(a)\,\mathrm{d}a}_{=\mu_a^2+\sigma_a^2 \triangleq \mathrm{E}_2} + c \cdot \int_{0}^{\infty} a^2 \cdot p_x(a)\,\mathrm{d}a - \mu_y^2 \\
&= \alpha^2 \cdot \mathrm{E}_2 + c \cdot \left(\mathrm{E}_2 \cdot \phi\left(\tfrac{\mu_a}{\sigma_a}\right) + \mu_a \cdot p_a\right) - \mu_y^2 \,, \qquad (13)
\end{aligned}
$$

with $c \triangleq \left(\beta^2 - \alpha^2\right)$ and E_2 being the second raw Gaussian moment.

Bayesian Perceptron - Backward Pass

each training instance $(x_i , y_i) \in D$ is processed sequentially. In doing so, there is no need for iterative batch processing being common in training NNs.

Given the prior distribution:

$$p_{i-1}(\underline{w}) \triangleq p(\underline{w}|\mathcal{D}_{i-1}) = \mathcal{N}(\underline{w}, \underline{\mu}^w_{i-1}, \mathbf{C}^w_{i-1})$$

We will calculate the posterior distribution:

$$\begin{aligned} p_i(\underline{w}) &= \int_{\mathbb{R}} p(a, \underline{w}|\mathcal{D}_i)\, \mathrm{d}a \\ &= \int_{\mathbb{R}} \underbrace{p(\underline{w}|a, \mathcal{D}_i)}_{\text{(I)}} \cdot \underbrace{p(a|\mathcal{D}_i)}_{\text{(II)}} \mathrm{d}a \ , \end{aligned} \qquad (14)$$

Bayesian Perceptron - Backward Pass

For (I) :
$\underline{w}$ and $\underline{a}$ are jointly Gaussian due to the linear mapping.

$$p(\underline{w}|a, \mathcal{D}_i) = \mathcal{N}\left(\underline{w}; \underline{\mu}^w_{i-1} + \underline{l}_i \cdot (a - \mu_a), \mathbf{C}^w_{i-1} - \underline{l}_i \cdot \underline{\sigma}^{\mathrm{T}}_{wa}\right) \tag{15}$$

with gain vector $\underline{l}_i \triangleq \underline{\sigma}_{wa}/\sigma^2_a$.

$$\begin{aligned}\underline{\sigma}_{wa} &= \mathrm{E}\{(\underline{w} - \underline{\mu}^w_{i-1}) \cdot (a - \mu_a)\} \\ &= \mathrm{E}\left\{(\underline{w} - \underline{\mu}^w_{i-1}) \cdot (\underline{w} - \underline{\mu}^w_{i-1})^{\mathrm{T}}\right\} \cdot \underline{x}_i = \mathbf{C}^w_{i-1} \cdot \underline{x}_i\end{aligned}$$

Bayesian Perceptron - Backward Pass

For (II) :
from Bayes' rule

$$p(a|\mathcal{D}_i) = \tfrac{1}{c} \cdot p(y_i|a) \cdot p(a|\underline{x}_i, \mathcal{D}_{i-1}) \qquad (16)$$

with normalization constant $c = \int p(y_i|a) \cdot p(a|\underline{x}_i, \mathcal{D}_{i-1})\, \mathrm{d}a.$

Evaluating (16) in closed-form is not possible in general. However, assuming that y and a are jointly Gaussian yields a Gaussian approximation

$$p(a|\mathcal{D}_i) \approx \mathcal{N}(a; \mu_i, \sigma_i^2) \qquad \begin{aligned} \mu_i &= \mu_a + k_i \cdot (y_i - \mu_y) \\ \sigma_i^2 &= \sigma_a^2 - k_i \cdot \sigma_{ya}^2 \end{aligned} \qquad (17)$$

with gain $k_i \triangleq \sigma_{ya}^2 / \sigma_y^2$.

This corresponds to the measurement update step of the famous Kalman filter.

Backward Pass - Sigmoid Case

$$\begin{aligned}\sigma_{ya}^2 &= \mathrm{E}\{(y-\mu_y)\cdot(a-\mu_a)\} \\ &= \mathrm{E}\{a\cdot f(a)\} - \mu_y\cdot\mu_a\ . \end{aligned} \tag{18}$$

$$\begin{aligned}\sigma_{ya}^2 &= \mathrm{E}\{a\cdot s(a)\} - \mu_y\cdot\mu_a \approx \mathrm{E}\{a\cdot\phi(\lambda\cdot a)\} - \underbrace{\mu_y\cdot\mu_a}_{\triangleq\mu_{ya}} \\ &= \int_{\mathbb{R}} a\cdot\phi(\lambda\cdot a)\cdot p_x(a)\,\mathrm{d}a - \mu_{ya} \\ &= \int_{\mathbb{R}} a\cdot\phi(\lambda\cdot a)\cdot\tfrac{1}{\sigma_a}\mathcal{N}\left(\tfrac{a-\mu_a}{\sigma_a};0,1\right)\mathrm{d}a - \mu_{ya} \\ &\overset{(g)}{=} \sigma_a\cdot\int_{\mathbb{R}} z\cdot\phi(\lambda\cdot(\sigma_a\cdot z+\mu_a))\cdot\mathcal{N}(z;0,1)\,\mathrm{d}z + \ldots \\ &\quad \mu_a\cdot\underbrace{\int_{\mathbb{R}}\phi(\lambda\cdot(\sigma_a\cdot z+\mu_a))\cdot\mathcal{N}(z;0,1)\,\mathrm{d}z}_{=\mu_y} - \mu_{ya} \\ &= \sigma_a\cdot\int_{\mathbb{R}} z\cdot\phi(\lambda\cdot(\sigma_a\cdot z+\mu_a))\cdot\mathcal{N}(z;0,1)\,\mathrm{d}z\ , \end{aligned} \tag{19}$$

$$\longrightarrow \quad \sigma_{ya}^2 \approx \tfrac{\lambda\cdot\sigma_a^2}{t}\cdot\mathcal{N}\left(\tfrac{\lambda\cdot\mu_a}{t};0,1\right)\ . \tag{20}$$

Backward Pass - Piece-wise Linear Case

$$
\begin{aligned}
\sigma_{ya}^2 &= \mathrm{E}\{a \cdot f(a)\} - \mu_{ya} \\
&= \int_{\mathrm{R}} a \cdot \max(\alpha \cdot a, \beta \cdot a) \cdot p_x(a)\,\mathrm{d}a - \mu_{ya} \\
&= \alpha \cdot \int_{-\infty}^{0} a^2 \cdot p_x(a)\,\mathrm{d}a + \beta \cdot \int_{0}^{\infty} a^2 \cdot p_x(a)\,\mathrm{d}a - \mu_{ya} \\
&= \alpha \cdot \mathrm{E}_2 + (\beta - \alpha) \cdot \left(\mathrm{E}_2 \cdot \phi\left(\tfrac{\mu_a}{\sigma_a}\right) + \mu_a \cdot p_a\right) - \mu_{ya}
\end{aligned}
$$

Summary

$$\underline{\mu}_i^w = \underline{\mu}_{i-1}^w + \underline{l}_i \cdot (\mu_i - \mu_a) \ ,$$
$$\mathbf{C}_i^w = \mathbf{C}_{i-1}^w + \underline{l}_i \cdot (\sigma_i^2 - \sigma_a^2) \cdot \underline{l}_i^{\mathrm{T}}$$
$$\underline{l}_i = (\mathbf{C}_{i-1}^w \cdot \underline{x}_i)/\sigma_a^2$$

Algorithm 1 Forward Pass for test input $\underline{x}$

1: Calculate mean μ_a via (6) and variance σ_a^2 via (7)
2: **switch** activation function
3: *sigmoid:* Calculate mean μ_y and variance σ_y^2 of output $\boldsymbol{y}$ according to (8) and (11)
4: *pwl:* Calculate mean μ_y and variance σ_y^2 of output $\boldsymbol{y}$ according to (12) and (13)
5: **end switch**
6: Return $(\mu_y, \sigma_y^2, \mu_a, \sigma_a^2)$

Algorithm 2 Backward Pass for training the proposed BP with data $\mathcal{D}$

1: Initialize weight distribution with mean vector $\underline{\mu}_0^w$ and covariance matrix $\mathbf{C}_0^w$
2: **for** each training instance $(\underline{x}_i, y_i) \in \mathcal{D}$ **do**
3: $(\mu_y, \sigma_y^2, \mu_a, \sigma_a^2) \leftarrow$ ForwardPass$(\underline{x}_i)$ ▷ *Algorithm 1*
4: **switch** activation function
5: *sigmoid:* Calculate covariance σ_{ya}^2 via (20)
6: *pwl:* Calculate covariance σ_{ya}^2 via (21)
7: **end switch**
8: Calculate mean μ_i and variance σ_i^2 according to (17)
9: Calculate gain vector $\underline{l}_i = (\mathbf{C}_{i-1}^w \cdot \underline{x}_i)/\sigma_a^2$
10: Update mean vector $\underline{\mu}_i^w$ and covariance matrix $\mathbf{C}_i^w$ of the weights according to (22)
11: **end for**
12: Return $(\underline{\mu}^w, \mathbf{C}^w) \leftarrow (\underline{\mu}_n^w, \mathbf{C}_n^w)$

Validation

Comparison to ground truth

To compare the results of the BP with the ground truth generated by numerically solving (16), we vary the values of the mean μ_a and variance σ_a^2 over a wide range. More precisely, μ_a takes values from the set $\{-3, -2.9, -2.8, \ldots, 3\}$ while $\sigma_a^2 \in \{0, 0.2, 0.4, \ldots, 2\}$. The output $y \in \{0, 1\}$ is determined based on the mean μ_a and the Heaviside step function according to

$$y = \begin{cases} 1 & , \ \mu_a > 0 \\ 0 & , \ \text{otherwise} \end{cases} .$$

of the cases. Accordingly, the mean absolute error (mae) for the mean μ_a is 0.0952 ± 0.0708. In case of the variance σ_a^2 the mae is slightly higher with 0.1349 ± 0.1291. These close approximations are obtained with a runtime being two orders of magnitude smaller than the ground truth calculations. This

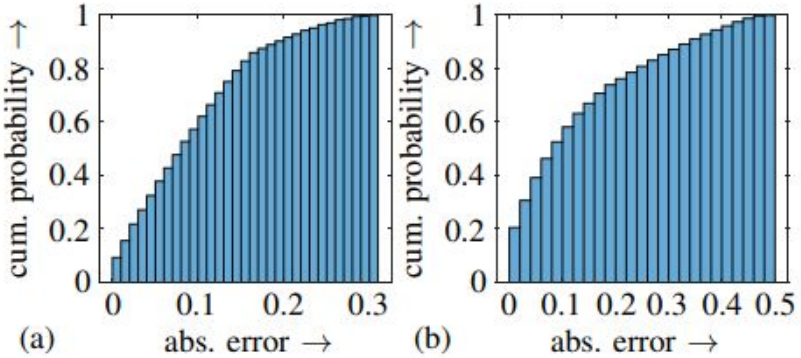


Fig. 3: Cumulative distribution of the errors in posterior mean (a) and variance (b) between ground truth and proposed solution.

Validation

Linear Binary Classification

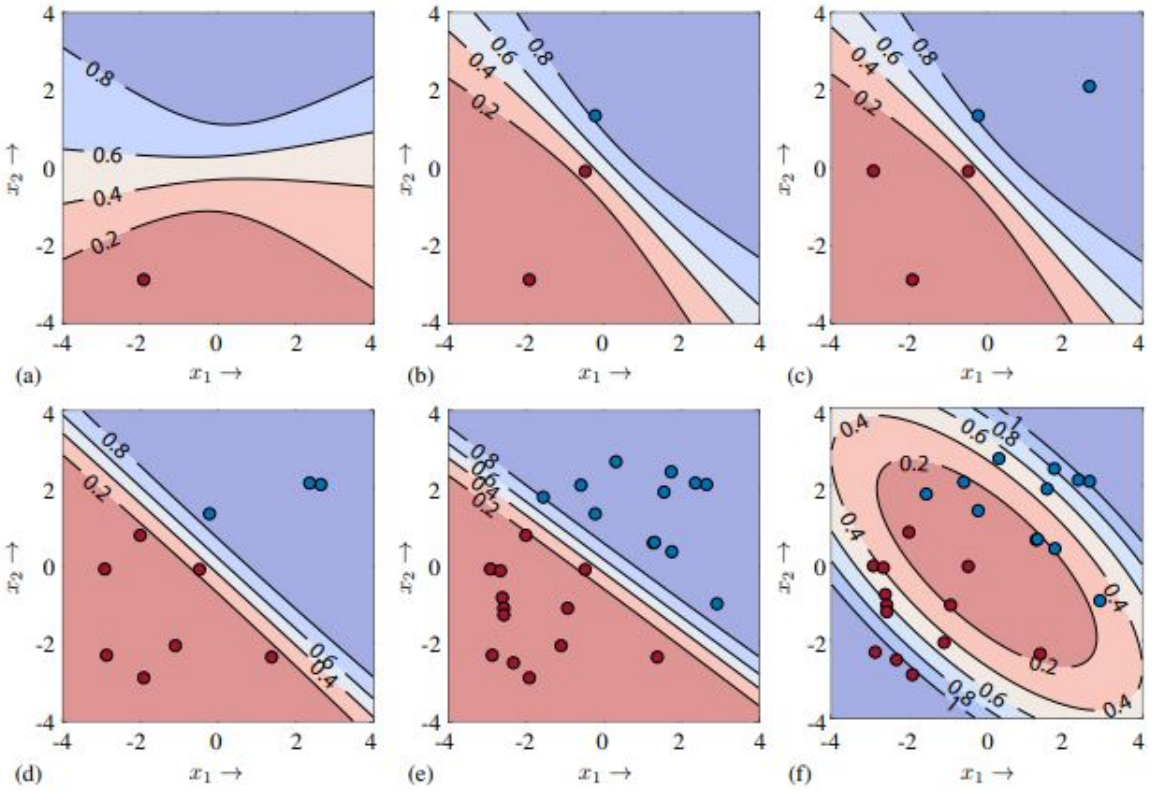


Fig. 4: Evolution of the predicted mean $\mu_y = \text{Prob}(y = 1|\underline{x}, \mathcal{D})$ for (a) one, (b) three, (c) five, (d) ten, and (e) 25 data instances. The variance σ_a^2 is shown in (f). It is important to note that the variance is not limited to one. It continues growing but higher values are not plotted in different colors in order to keep the visualization simple. Red dots indicate data instances belonging to class $y = 0$ and blue dots belong to class $y = 1$.

Validation

Non Linear Regression

Perceptrons are used for linear classification problems only, but depending on the activation function used, also simple regression problems can be tackled. Here, we consider a regression problem where the data is generated by means of a noisy *softplus* function

$$\boldsymbol{y} = \log\left(1 + \mathrm{e}^{\gamma \cdot x + \delta}\right) + \boldsymbol{v} \ , \ \boldsymbol{v} \sim \mathcal{N}(0, 0.01) \ . \quad (26)$$

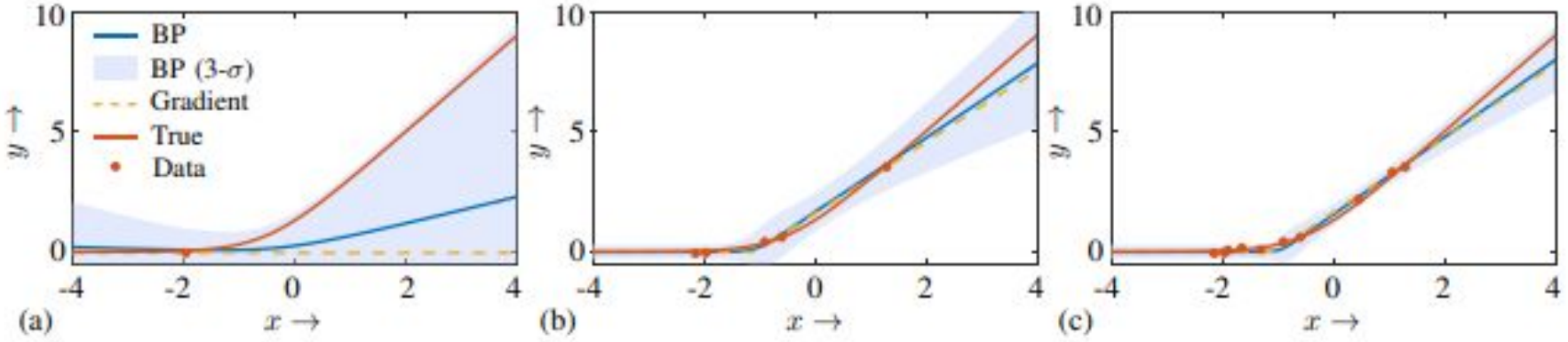


Fig. 5: Sequentially learning the softplus function (26) (red) with (a) one, (b) five, and (c) ten data instances.