Max-plus algebra in deep neural networks
[Talk submission]

Ikhlas Enaieh, Olivier Fercoq

Télécom Paris - LTCI Laboratory/ S2A Team

Abstract. Deep Neural Networks (DNNs) are powerful models for solving machine learning problems due to their ability to summarize complex information from large datasets. These models consist of elementary components known as neurons, organized in layers.
Despite their modularity, only a limited number of neuron types are commonly used. Traditionally, DNNs compute their outputs using standard neurons, where the output is a weighted sum of the inputs. Recently, max-plus neurons have emerged as a novel paradigm, replacing addition with maximum and multiplication with summation. The formula for the output is thus $\hat{y} = \max_j\{X_{i,j} + w_j\}$.
We obtain sparse subgradients from the max-plus structure, a situation that does not happen for smooth functions. Current backpropagation algorithms cannot take advantage of this feature and make many unnecessary computations. We developed a novel sparse subgradient algorithm tailored for non-convex, non-smooth optimization problems in DNNs. Initial experiments in binary classification have demonstrated promising results, motivating further investigation into its efficacy in multiclass classification scenarios.

Keywords: Deep Neural Networks · Max-Plus Operator · Nonsmooth optimization · Nonconvex optimization · Sparse Subgradient method.

1 Empirical evaluation of subgradient sparsity

We conducted an experiment based on the model from [1], which minimizes the average Categorical Cross Entropy (CCE) loss to solve a multiclass classification problem using the MNIST dataset. In this experiment, we measured the sparsity of the subgradients of the average loss and the loss for a single random sample in each iteration using SGD. We observed that while the average loss gradients are dense, the subgradients computed with SGD are very sparse (Table 1).
We quantify the level of sparsity of the sub-gradient [2] using a metric denoted as γ, where:

$$\gamma(x) = \frac{\text{number of non-zero elements in } x}{\dim(x)}$$

Consequently, we modified the model to select the sample that yields the worst loss, defined as the maximum of the loss instead of the average. The sparsity of the subgradient is a characteristic feature of non-smooth functions like the maximum function. This sparsity affects the backpropagation step in the Max-Plus Neural Network.

Table 1. Sparsity level of the average CCE loss and max CCE loss

Model	$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=0}^{D} y_{i,j}\log(\hat{y}_{i,j})$	$L = \max_{1\leq i\leq N} -\sum_{j=0}^{D} y_{i,j}\log(\hat{y}_{i,j})$
$\gamma(\frac{\partial L}{\partial w})$	0.8187597457118868	0.0034634760705289673

On the one hand, the benefit of this sparsity is that the weight update step only involves the non-zero elements in the subgradient. On the other hand, recomputing the maximum can be computationally expensive. In [2] they proposed a fast method for recomputing the maximum using a Short Computational Tree (SCT) format, and an algorithm to solve convex non-smooth problems.

2 Max-Plus Neural Network model

Building on these concepts, we developed a sparse gradient algorithm to tackle non-convex, non-smooth optimization problems, particularly focusing on minimizing the loss in neural networks composed of multiple functions. Our sparse subgradient algorithm has shown promising results in binary classification tasks.

Below, we present our Max-Plus Neural Network model, designed without a hidden layer, for multi-class classification using the MNIST dataset (classify digits 0 to 9). Since the true labels are in integer format, we use Sparse Categorical Cross-Entropy (SCCE) instead of Categorical Cross-Entropy, which is more suited for one-hot encoded labels.

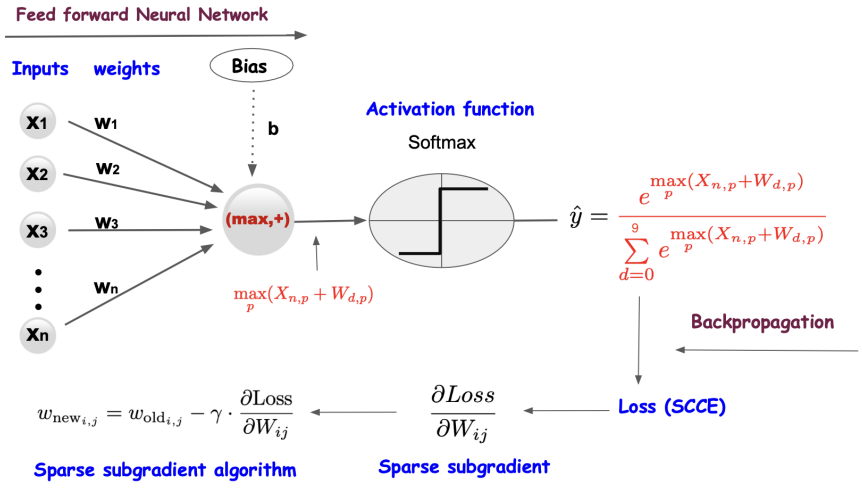

Fig. 1. Max-Plus Neural Network structure

The model that we are working on is:

$$\min_{w} \max_{1\leq n\leq N} SCCE = \min_{w} \max_{1\leq n\leq N} -\log(\hat{y}_{n,y_n})$$

where $\hat{y}_{n,y_n} = \dfrac{e^{\max_p(X_{n,p}+W_{d,p})}}{\sum_{d=0}^{9} e^{\max_p}(X_{n,p}+W_{d,p})}$

After simplification, we are solving the optimization problem

$$\min_{w} \max_{n} \left(-\max_{p}(X_{n,p}+W_{y_{(n)},p}) + \log \sum_{d=0}^{9} \exp\left(\max_{p}(X_{n,p}+W_{d,p})\right)\right)$$

Where :

- X: is the input matrix where the rows are the images and the columns are the corresponding pixels for each image.
- W: is the weight matrix where the rows represent the digits and the columns represent the weights of the pixels corresponding to each digit.
- $y(n)$: is the true label for image n.
  - n : images, where $1 \leq n \leq 60000$
  - p : pixels, where $1 \leq p \leq 784$
  - d : digits, where $0 \leq d \leq 9$

3 Sparse Subgradient algorithm

Assume we have the following optimization problem

$$\min_{x\in Q} f(x)$$

where Q is a simple closed convex set in R^n and f is a non-smooth, non-convex function, then the proposed sparse subgradient algorithm is:

$$x_0 \in Q, \quad x_{k+1} = \pi_Q\left(x_k - \frac{f(x_k)-f^*}{\|f'(x_k)\|^2} f'(x_k)\right), \quad k \geq 0.$$

where $f'(x_k)$ is a sparse subgradient which is an element of a conservative field [3].

In our upcoming work, we plan to evaluate the performance of our sparse subgradient algorithm on multi-class classification tasks, interpreting our model as a SCT structure. We will also focus on enhancing the convergence rate of the algorithm. Furthermore, we intend to explore various configurations of Max-Plus Neural Networks, starting with a model that has no hidden layers, then progressing to a model with one hidden layer, and finally experimenting with combinations of different neuron types, such as max-plus and max-minus neurons.

References

1. Yunxiang Zhang, Samy Blusseau, Santiago Velasco-Forero, Isabelle Bloch, and Jesus Angulo. Max-plus operators applied to filter selection and model pruning in neural networks,2019.
2. Yu Nesterov. Subgradient methods for huge-scale optimization problems. Mathematical Programming,2014,(146-275).
3. Jérôme Bolte, Edouard Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient method and deep learning. Mathematical Programming, 2020, 188 (19-51)