Gradient Based Training of Non-smooth Neural Networks

Xiaoyu Wang (University of Cambridge), Martin Benning (Queen Mary University of London)

Introduction

In this work, we introduce a novel mathematical formulation for the training of neural networks with non-smooth activation functions. Instead of obtaining parameter estimates via subgradient-based algorithm, we use a tailored Bregman alternating direction method of multipliers (BADMM) approach to learn the network parameters.

The advantage of this new formulation is that its partial derivatives with respect to the network's weight and bias terms do not require the computation of subdifferentials of the activation functions.

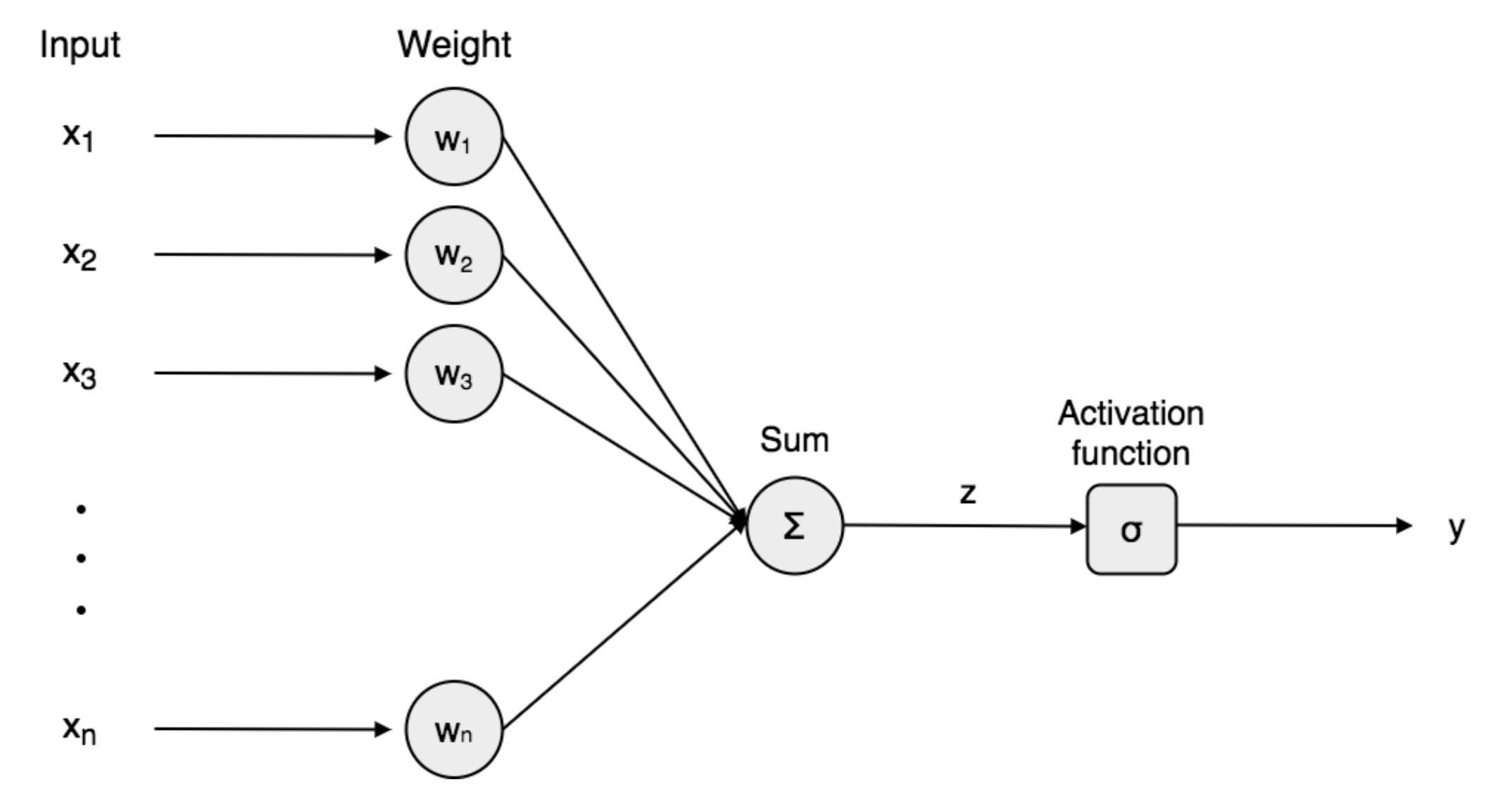
A Motivating Example

Feedforward neural network is based around a particular class of hypothesis with the following general form, with d indicates the depth of the network:

$$f(x) = A^{(d)} \circ g \circ A^{(d-1)} \circ g \circ \cdots \circ g \circ A^{(1)}(x)\,,$$

where $A^{(d)}(x) = W^{(d)}x + b^{(d)}$ defines an affine transform and g applies an activation function elementwise.

As an initial example, we consider a perceptron architecture with ReLU activation function: $f_{w,b}(x) = \max(w^\top x + b, 0)$, for input data $x \in \mathbb{R}^n$, output data $y \in \mathbb{R}_{\geq 0}$ and weights $w \in \mathbb{R}^n$ and bias $b \in \mathbb{R}$.



A standard procedure to train the perceptron is by minimizing the empirical risk: $\ell(y, \max(w^\top x + b, 0))$ via (stochastic) subgradient descent (SGD) with respect to w and b. However, one downside of this approach is the slow guaranteed convergence rate when using SGD or its variants.

Learning the Perceptron

We introduce auxiliary variable z and reformulate the empirical risk minimization problem to the following equality constrained form:

$$\min_{z,w,b} \ell(y, \max(z,0)) \quad \text{subject to} \quad z = w^\top x + b\,.$$

Inspired by the work (Geiping and Möller, 2019), we then further relax to the following problem:

$$\min_{z,w,b} D_{E_z}(y, \max(z,0)) \quad \text{subject to} \quad z = w^\top x + b\,.$$

Here D_{E_z} denotes the Bregman distance between two arguments with respect to a function E_z, i.e.

$$D_{E_z}(u,v) = E_z(u) - E_z(v) - \langle \nabla E_z(v), u - v\rangle\,,$$

where $E_z(x) := \frac{1}{2}|x - z|^2 + \chi_{\geq 0}(x)$, with

$$\chi_{\geq 0}(x) := \begin{cases} 0 & x \geq 0 \\ \infty & x < 0 \end{cases}$$

Since $0 \in \partial E_z(\max(z,0))$, the Bregman distance reads

$$D_{E_z}(y, \max(z,0)) = E_z(y) - E_z(\max(z,0))\,.$$

Notice that $E_z(\max(z,0))$ is the Moreau-Yosida regularisation of $\chi_{\geq 0}$, i.e.

$$G(z) := E_z(\max(z,0)) = \inf_x \{\chi_{\geq 0}(x) + \frac{1}{2}|x - z|^2\}.$$

From a basic property we know,

$$\nabla G(z) = z - \max(z,0)\,.$$

And therefore,

$$\nabla D_{E_z}(y, \max(z,0)) = \max(z,0) - y$$

The advantage of working with this formulation is that $\nabla D_{E_z}(y, \max(z,0))$ is Lipschitz-continuous with constant one and evaluate this gradient does not require us to differentiate the non-smoothness.

We solve this problem using a generalized Bregman alternating direction method of multipliers (Wang and Banerjee, 2014).

Generalize to Multi-layer Architecture

We further generalize this method to train a multi-layer architecture.

- First introduce auxiliary variables X_l for each hidden layer.
- Then relax the equality constraints on hidden layers with Bregman distances.

The constrained minimization problem for an L layer feedforward neural network is therefore formulated as following (bias term omitted for simplicity):

$$\min_{Z,W,X} \ell(Y, Z_L) + \sum_{l=1}^{L-1} D_{E_Z}(X_l, \sigma(Z_l))$$

$$\text{subject to} \qquad \{Z_l = W_l X_{l-1}\}_{l=1}^{L}\,.$$

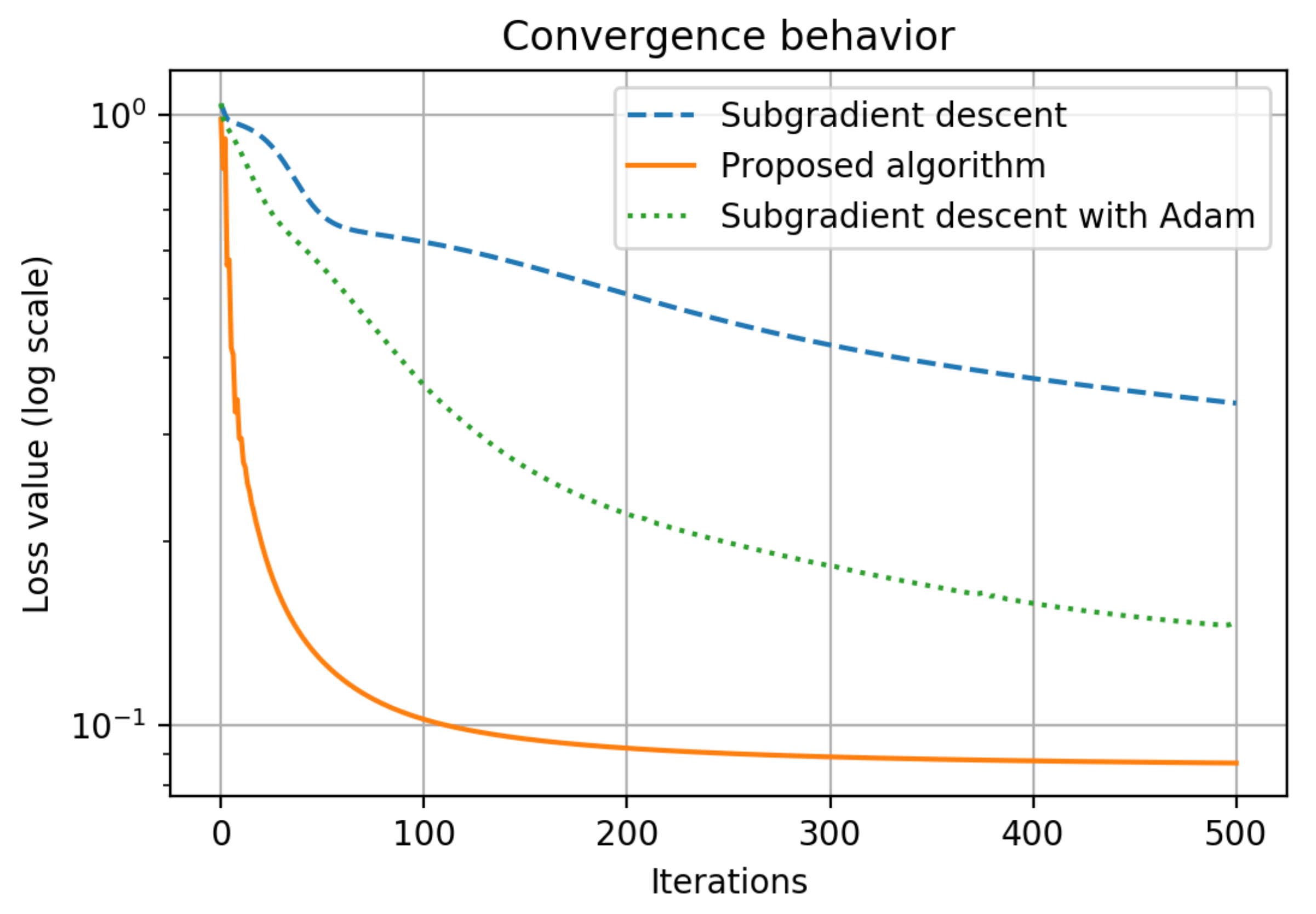
Follow the BADMM approach, with the positive step-size parameters $\tau_Z < 2, \tau_{X_l} < 2/\|W_{l+1}\|^2$ and τ_M, this yields the following algorithm:

Algorithm 1 Train a multi-layer perceptron

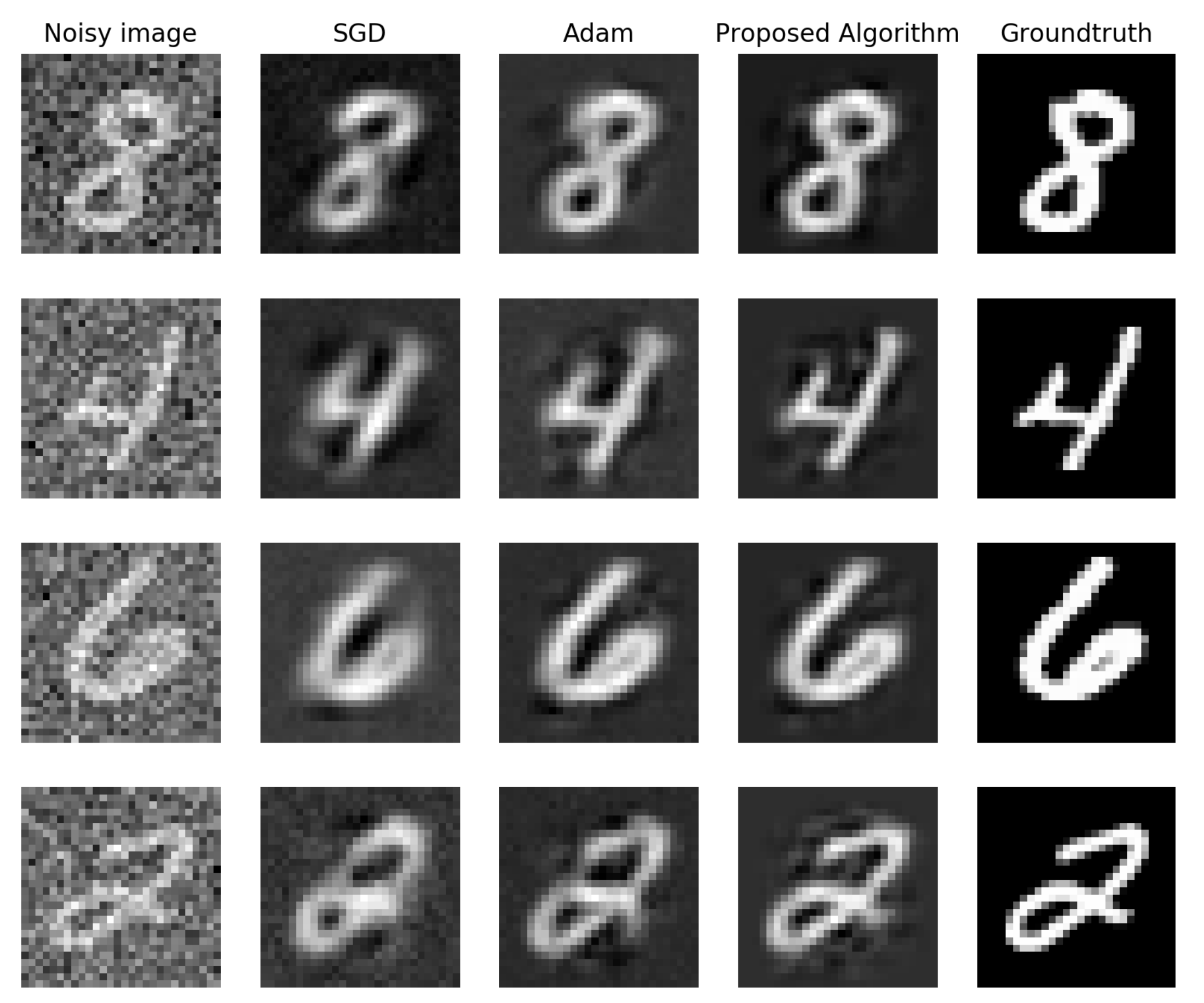
```
Initialize    $\{Z_l\}_{l=1}^L, \{W_l\}_{l=1}^L, \{X_l\}_{l=0}^{L-1}, \{M_l\}_{l=1}^L$
while $k \leq$ maximum # iterations do
    $Z_l^{k+1} = \frac{(Z_l^k - \tau_{z_l}(\sigma(Z_l^k) - X_l + M_l - \delta(W_l X_{l-1})))}{1+\tau_{z_l}\delta}$
    $W_l^{k+1} = (Z_l + \frac{1}{\delta}M_l)X_{l-1}^T(X_{l-1}X_{l-1}^T + \lambda I)^{-1}$
    $M_l^{k+1} = M_l^k + \tau_M(Z_l - W_l X_{l-1})$
    $X_l^{k+1} = \max(0, \frac{\tau_{X_l}}{\tau_{X_l}+1}(-\delta W_{l+1}^T(W_{l+1}X_l^k -$
    $Z_{l+1} - \frac{1}{\delta}M_{l+1}) + Z_l + \frac{1}{\tau_{X_l}}X_l^k))$
end
```

Numerical Experiments

For numerical experiments, we consider the auto-encoder architecture with hidden dimension h set at 64. We experiment on the MNIST dataset for the image denoising task. The performance of our proposed algorithm is compared with SGD and Adam optimizer, both implemented with PyTorch. All three algorithms are set to run for 500 iterations on a fixed 1,000 training images.



The auto-encoder is trained to reconstruct clear images from noisy inputs. The figure below shows a comparison of sample outputs from trained auto-encoder using different methods. We can see that our proposed algorithm, in addition to faster convergence performance, also produces better quality denoised images.



References

[1] J. Geiping, M. Moeller, Parametric Majorization for Data-Driven Energy Minimization Methods.
[2] H. Wang, A. Banerjee, Bregman Alternating Direction Method of Multipliers.