ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Decay Momentum for Improving Federated Learning

Miguel Fernandes, Catarina Silva, Joel P. Arrais,
Alberto Cardoso and Bernardete Ribeiro

University of Coimbra - Department of Informatics Engineering
Polo II Pinhal de Marrocos, Coimbra - Portugal

Abstract. We propose two novel Federated Learning (FL) algorithms based on decaying momentum (Demon): Federated Demon (FedDemon) and Federated Demon Adam (FedDemonAdam). In particular, we apply Demon to Momentum Stochastic Gradient Descent (SGD) and Adam in a Federated setting, which has shown to improve results in a centralized environment. We empirically show that FedDemon and FedDemonAdam have a faster convergence rate and performance improvements compared to state-of-the-art algorithms including FedAvg, FedAvgM and FedAdam.

1 Introduction

In recent years, data-intensive machine learning methods that can extract hidden information from data are becoming a powerful tool in several domains. However, latency and privacy problems arise when data needs to be transferred from multiple edge devices to centralized locations. As edge devices are becoming computationally more powerful, Federated Learning [1] was proposed with the objective of solving these issues by using edge devices to train machine learning models.

In FL, a Federated Server distributes the computing tasks to the clients where the private data is generated. Once at the client the shared model is updated to fit the available data. Afterwards, each updated model is sent back to the server, keeping clients' data private.

In the current paradigm, Federated Learning's deep neural networks require long periods of time to train. For example, studies have shown that recurrent neural networks can take up to five days to train in a Federated setting [2]. As a consequence, the communication costs of data transferring between servers and clients can be problematic, even in a Federated setting. Hence, advances in methods that can accelerate Federated model training are of utmost importance.

Federated Averaging (FedAvg) [1], the most commonly used FL algorithm, is based on a weighted average of client models sent to the server, giving higher importance to models trained with more data points. In addition, FL algorithms inspired by the Momentum (FedAvgM) and Adam (FedAdam) [2, 3] optimizers have also been explored and proven to have a better convergence rate than vanilla FedAvg.

In this work we apply the momentum decay rule (Demon) [4, 5] to FL. The Demon rule has the objective of decaying the total contribution of a gradient

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

to all future updates and was demonstrated to surpass the performance of both Adam and Momentum SGD optimizers.

We present a comparison between the proposed algorithms and current FL methodologies. The experiments are performed over two benchmark datasets [6, 7] which are used in FL studies. We make the following contributions:

- Proposal and implementation of Federated Demon (FedDemon)
- Proposal and implementation of Federated Demon Adam (FedDemonAdam)
- Comparison between the proposed approaches with current state-of-the-art FL algorithms including FedAvg [1], FedAvgM [3] and FedAdam [3]

We empirically demonstrate that the proposed algorithms present lower error values and converge faster compared to their family of algorithms.

2 Federated Decaying momentum

In this section, we present the proposed FL algorithms: FedDemon and FedDemonAdam. This adaptation of the Demon algorithm to the Federated setting is motivated by the improved performance of Demon in comparison to Momentum SGD and Adam in a centralized environment. The objective is to use momentum to speed up the early phases of training and then decay it throughout training in order to prevent weights from growing too quickly. In the Demon algorithm the momentum parameter β is given by:

$$\beta_t = \beta_0 \frac{(1 - \frac{t}{T})}{(1 - \beta_0) + \beta_0(1 - \frac{t}{T})} \tag{1}$$

where β_t is the momentum parameter at iteration t, β_0 is the initial momentum parameter and T is the total number of iterations.

In both algorithms, at the t communication round the server is connected to a random subset of clients S_t, $S_t < K$ and the current global model, θ_t, is broadcasted to each. Then the clients in S_t update the model using SGD locally for E epochs to optimize the local objective. Afterwards, the client model, θ_t^k, is sent to the server where the new model, θ_{t+1}, is generated. The local optimizer can be any gradient-based method. Algorithm 1 presents the clients' model update.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Algorithm 1 *client*$_k$ update

Receive θ_t from server
Initialize η_l, B, E
for *each local epoch i from 1 to E* **do**
  for *batch b in B* **do**
    $\theta^k_{t+1,i+1} = \theta^k_{t,i}$ - $\eta_l \nabla f_k(\theta^k_{t,i}$, b)
  end
end
$\theta^k_{t+1} = \theta^k_{t+1,i+1}$
return θ^k_{t+1} to the server

2.1 FedDemon

The FedDemon algorithm results from the use of the momentum value in the server model update with the decay rule of the Demon algorithm. Algorithm 2 presents the pseudo code for FedDemon.

Algorithm 2 FedDemon

Initialize θ_t, β_0, T
for *each round t = 1, 2, ..., T* **do**
  Select S_t = random subset of clients ($S_t < K$)
  for *each client* $k \in S_t$ **do**
    θ^k_{t+1} = Algorithm 1 (θ_t)
  end
  $\alpha = \sum_{k=1}^{S_t} \frac{n_k}{n}(\theta^k_{t+1} - \theta_t)$
  $\beta_t = \beta_0 \frac{(1-\frac{t}{T})}{(1-\beta_0)+\beta_0(1-\frac{t}{T})}$
  $v_{t+1} = \beta_t v_t + \alpha$
  $\theta_{t+1} = \theta_t + v_{t+1}$
end

Firstly, the FedDemon algorithm receives the updated clients models and calculates a weighted average of the local updates, α, giving higher importance to updates trained with more data points, where n is the total number of data points at the current S_t. Secondly, it calculates the current β_t value using Equation 1. Thirdly, it calculates the new momentum value, v_{t+1}, by summing a fraction (given by β_t) of the previous update and the averaged local updates. Afterwards, the global model is updated by summing the previous model with the Momentum update.

2.2 FedDemonAdam

The FedDemonAdam algorithm results from the use of the Adam optimizer in the server model update with the decay rule of the Demon algorithm. Algorithm 3 presents the pseudo code for FedDemonAdam.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Algorithm 3 FedDemonAdam

Initialize $\theta_t, \beta_0, \beta_2, \eta$, T
for *each round t = 1, 2, ..., T* **do**
  Select S_t = random subset of clients ($S_t < K$)
  for *each client* $k \in S_t$ **do**
    θ_{t+1}^k = Algorithm 1 (θ_t)
  end
  $\alpha = \sum_{k=1}^{K} \frac{n_k}{n}(\theta_{t+1}^k - \theta_t)$
  $\beta_t = \beta_0 \frac{(1-\frac{t}{T})}{(1-\beta_0)+\beta_0(1-\frac{t}{T})}$
  $m_{t+1} = \beta_t m_t + \alpha$
  $v_{t+1} = \beta_2 v_t + (1-\beta_2)\alpha^2$
  $\hat{v}_{t+1} = v_{t+1}/(1-\beta_2^t)$
  $\theta_{t+1} = \theta_t + \eta(\frac{m_{t+1}}{\sqrt{\hat{v}_{t+1}+\epsilon}})$
end

Firstly, FedDemonAdam receives the updated client models and similarly to FedDemon it calculates α and β_t. Secondly, the moving average of the squared α, v_t, and the moving average of α, m_t, using β_t are calculated. Lastly, the model is updated by summing the previous model with the scaled learning rate.

3 Experimentation

In this section, we empirically compare the performance of the two proposed algorithms (FedDemon and FedDemonAdam) to their state-of-the-art counterparts (FedAvgM, FedAdam) by training deep neural networks in a Federated setting. The simulation used is based on the LEAF project [8] which is a FL benchmark.

Table 1 shows the statistics of the FEMNIST [6] and CELEBA [7] datasets for image classification tasks. FEMNIST is built by partitioning the data on the Extend MNIST. CELEBA is a large-scale face attributes' dataset with celebrity images.

Dataset	Number of Clients	Samples per Client	
		Mean	Standard Deviation
FEMNIST	3500	226.26	89.12
CELEBA	9343	21.44	7.63

Table 1: Statistics of the datasets used in the experimentation.

The model's architecture used in the FEMNIST has an input layer of 28×28, followed by a convolutional layer which produces 32 features maps with a kernel of 5×5. This is then followed by a Max-pooling layer with a max-pool of 2×2 and a stride of 2. Afterwards, a second convolutional layer produces 64 feature maps with a kernel of 5×5 and a Max-pooling, similar to the previous. Finally,

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

the model has a fully connected layer with 2028 neurons followed by a softmax output layer. All the layers use ReLU as the activation function. The loss function used was the cross-entropy and the optimizing method was SGD.

The model's architecture used in the CELEBA dataset has an input layer of 84×84, followed by a convolutional layer with 32 filters and a kernel size of 3×3. This is then followed by a Max-pooling layer with a max-pool of 2×2 and a stride of 2. This sequence of layers is repeated three times. Finally, the model has a softmax output layer. All the layers use ReLU as the activation function. The loss function used was the cross-entropy and the optimizing method was SGD.

The number of client models used at each communication round was set to 5 and the local mini-batch size to 10.

For the FEMNIST dataset η_l was set to 0.001, β_0 was set to 0.9 in FedDemon and FedDemonAdam and β_2 and η were set to 0.999 and 0.01 in FedDemonAdam, respectively. For the CELEBA dataset η_l was set to 0.0001 in FedDemon and FedDemonAdam, β_0 was set to 0.9 and β_2 and η were set to 0.999 and 0.001 in FedDemonAdam, respectively.

Figures 1 and 2 present the performance comparison between FedAvg, FedAvgM, FedDemon, FedAdam and FedDemonAdam for the FEMNIST and CELEBA datasets. Looking at the results, it can be observed that FedDemonAdam was the best performing algorithm for both datasets since it was faster to converge and presented a smaller error value. FedAvg algorithm was outperformed by all the others. FedDemon converged at a similar rate to FedAvgM but was able to achieve a better testing accuracy and a lower training loss for the FEMNIST dataset. Table 2 presents the algorithms' performance.

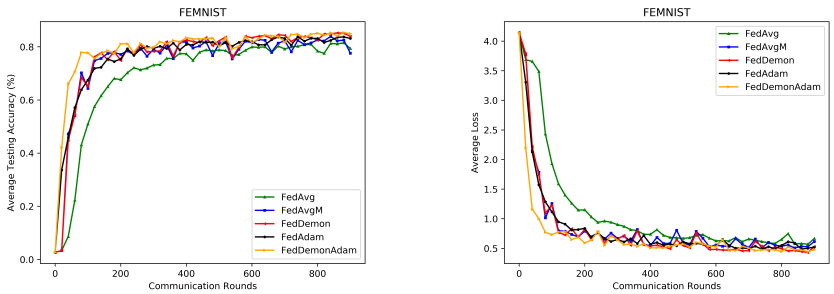

Fig. 1: Performance comparison between FedAvg, FedAvgM, FedDemon, FedAdam and FedDemonAdam for the FEMNIST dataset.

4 Conclusion

In this work, we proposed two novel Federated Learning algorithms: Federated Demon (FedDemon), Federated Demon Adam (FedDemonAdam). FedDemon utilizes Momentum to update the server model with a momentum decay rule.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

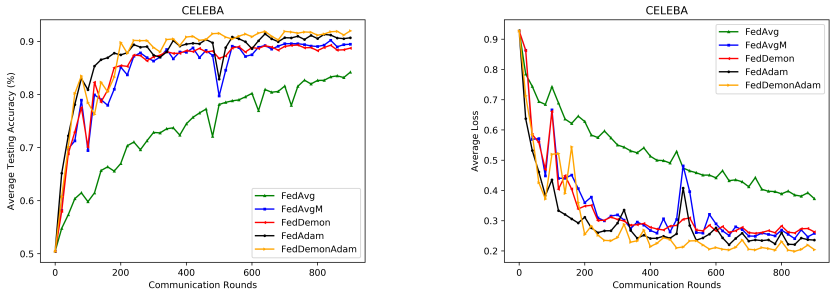

Fig. 2: Performance comparison between FedAvg, FedAvgM, FedDemon, FedAdam and FedDemonAdam for the CELEBA dataset.

Dataset	FedAvg	FedAvgM	FedDemon	FedAdam	FedDemonAdam
FEMNIST	79.3	77.6	83.9	83.1	**84.9**
CELEBA	84.2	89.4	88.7	90.6	**91.9**

Table 2: Test set accuracy for the FEMNIST and CELEBA after training.

FedDemonAdam is a Federated Learning reformulation of the Adam optimizer with a momentum decay rule for the moving average of the model update.

We compare the proposed models in a Federated Learning simulation on two benchmark datasets and the results show that the proposed algorithms had faster convergence rates and performance improvements compared to the state-of-the-art.

References

[1] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[2] Zhouyuan Huo, Qian Yang, Bin Gu, Lawrence Carin, and Heng Huang. Faster on-device training using new federated momentum algorithm. *CoRR*, abs/2002.02090, 2020.

[3] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020.

[4] John Chen and Anastasios Kyrillidis. Demon: Momentum decay for improved neural network training. *CoRR*, abs/1910.04952, 2019.

[5] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[6] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 1998.

[7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[8] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.