Optimization

EE18ACMTECH11006 Bhavani Machnoori

March 4, 2019

An Improved Adagrad Gradient Descent Optimization Algorithm

Gradient descent optimization algorithm is veryimportant in deep learning. In order to obtain a more stable convergence process and reduce overfitting in multiple epochs, we propose an improved Adagrad gradient descent optimization algorithm in this paper.

The gradient descent optimization algorithms can be classified into three categories: Batch gradient descent,Stochastic gradient descent (SGD) and Mini-batch gradient descent .
Although SGD is one of the most widely used gradient descent optimization algorithms in recent years, it still has many disadvantages.
convergence process of SGD is not stable enough.
The overfitting exists in SGD .
Thus our approach is proposed to handle these issues.

Adagrad gradient descent optimization algorithm is proposed by Duchi.

Learning rate can be adaptive during the whole convergence process.

Thus this approach works well in sparse datasets.

weakness

learning rate slows down during training.

To solve this problem,we are going for improved version.

To make the convergence process become more stable and reduce overfitting in multiple epochs.
We proposed an improved Adagrad gradient descent optimization algorithm. Our approach replaces the square of the gradient with the length of the gradient compared with the original Adagrad algorithm.

A. Gradient descent optimization algorithm

The formula of the gradient descent algorithm is as follows

$\theta = \theta_0 - \eta \nabla_\theta J(\theta_0)$

where,

θ_0= parameter to be update

J(θ_0)=loss function

$\nabla_\theta J(\theta_0)$=gradient of the parameterθ_0

η=learning rate.

The unchangeable learning rate is the weakness of many gradient descent optimization algorithms.

Because the unchangeable learning rate is not suitable for the sparse data in deep learning.

B. Adagrad algorithm

To make the learning rate become adaptable in sparse data, Adagrad algorithm was born. Adagrad algorithm divides the learning rate by a gradient-related quantity to achieve this purpose.

s$\leftarrow$ s $+\nabla_\theta$ J(θ_0) ∇_θ J(θ_0)

$\theta = \theta_0 - \eta \div \sqrt{s + \epsilon}\nabla_\theta$ J(θ_0)

s=sum of the square of gradients,the initial value=0

ϵ=smoothing term that avoids division by zero (usually is 1e-8).

Adagrad algorithm divides the learning rate by the sumof the square of gradients to achieve the purpose thatchanging the learning rate during training.
This approach increases the adaptability of the learning rate in sparse datasets and performs well in deep learning tasks.
AlthoughAdagrad algorithm has many advantages, it still has the weakness of the diminishing learning rate.
This weakness makes the convergence process and the validation loss become unsatisfactory in multiple epochs.

C. An improved Adagrad algorithm To improve the performance of Adagrad, both improve the stability of the convergence process and reduce overfitting.

We propose an improved Adagrad algorithm. The specific algorithm is as follows.

s$\leftarrow$ s+$|\ \nabla_\theta\ J(\theta_0)|$

$\theta = \theta_0 - \eta \div$ s+$\epsilon \nabla_\theta\ J(\theta_0)$

where$|\ J(\theta_0)|$=length of the gradient

The improved Adagrad algorithm replaces the square of the gradient with the length of the gradient and also removes the square root operation . Our approach is tested on different datasets and it performs well. The results show that our approach has a more stable convergence process and can reduce overfitting in multiple epochs.

PSUEDO CODE

Input:

$nb\_epochs \leftarrow$ number of epochs

$lr \leftarrow$ learning rate

BEGAIN:

$smooth\_t \leftarrow$ smooth term

$s \leftarrow 0$

$epochs \leftarrow 0$

while $epochs < nb\_epochs$

do $g \leftarrow$ get gradients of θ

$loss \leftarrow J(\theta)$

$s \leftarrow s +$ length of g

$\theta \leftarrow \theta - lr * g / (s + smooth\_t)$

$epochs \leftarrow epochs + 1$

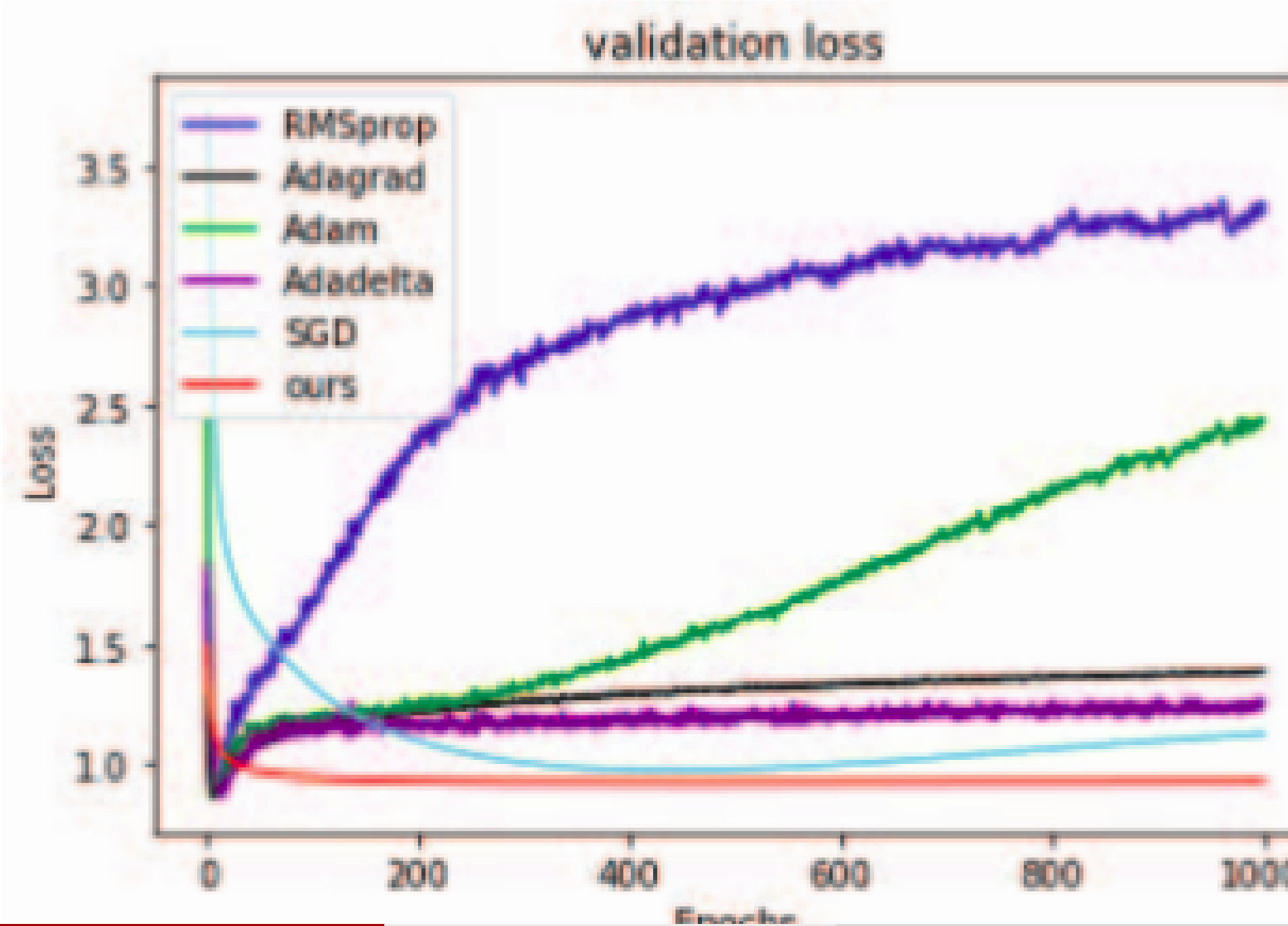
return $loss$

END

Experment results:

Our approach is tested on the Reuters dataset with other different algorithms, Adam, RMSprop, Adagrad, SGD and Adadelta.
The activation function of the front layers is relu.
The activation function of the last layer is softmax.
The encoding method is one-hot encoding The loss function is the categorical crossentropy .
The initial learning rate is the same for all algorithms. The number of the epochs is 1000 and the batch size is 512.
Because the overfitting is mainly observed from the validation set, therefore the experiment result figures of the validation set are given.

The validation loss of different algorithms are as follows Reuters dataset



IDMB data set

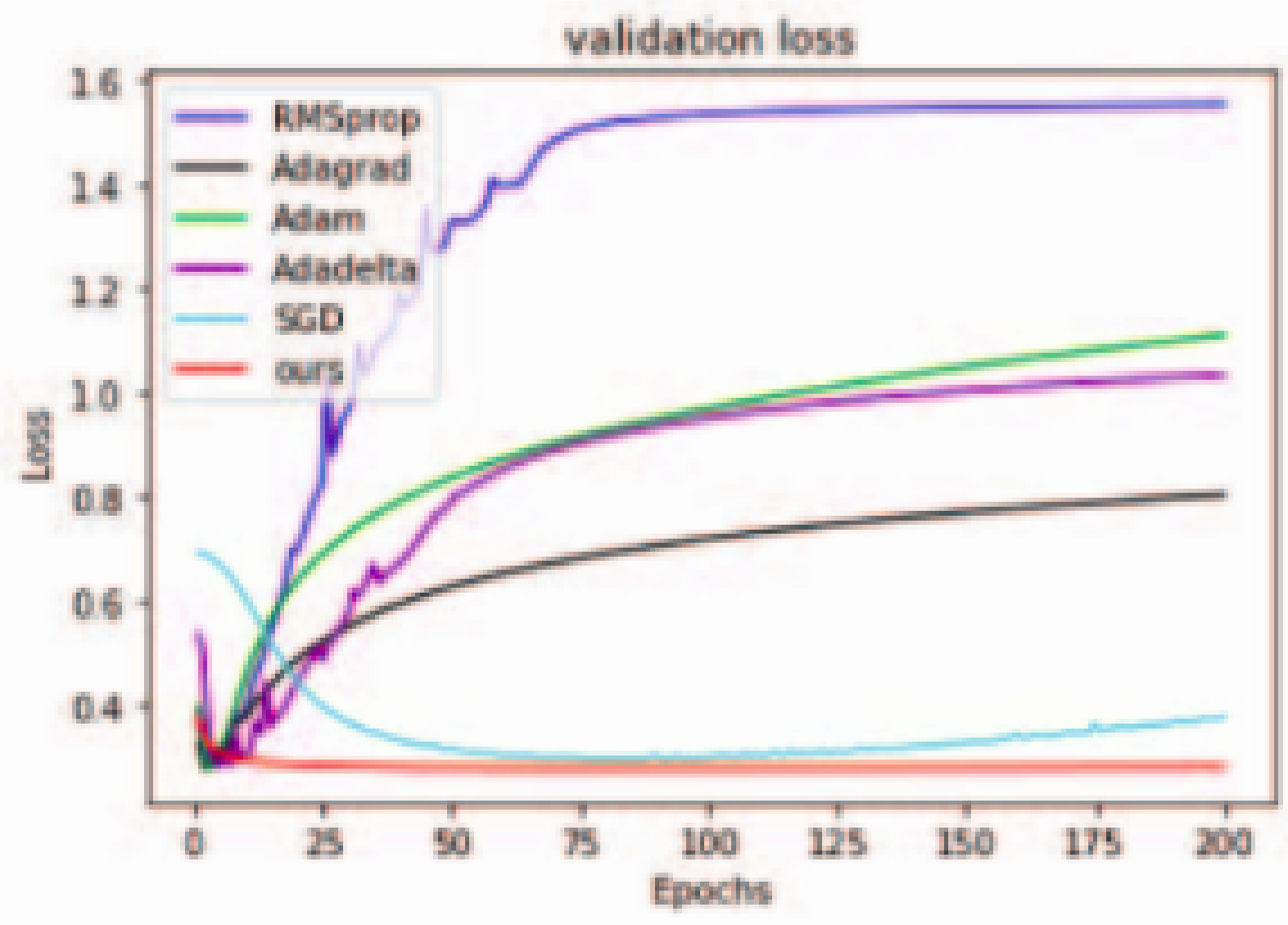


Fig. 3. The validation loss of different algorithms on the IMDB dataset