Langevin Algorithms for Very Deep Neural Networks with Application to Image Classification

Pierre BRAS

Laboratoire de Probabilités, Statistique et Modélisation
Sorbonne Université, Paris, France

Presented at the International Neural Network Society, Deep Learning Innovations and Applications INNS DLIA workshop, part of the International Joint Conference on Neural Networks IJCNN 2023
# Table of contents

1 Introduction
   1 Langevin Gradient Descent
   2 Preconditioned Langevin Gradient Descent
   3 Training of very Deep Neural Networks
   4 Objectives

2 Side-by-side comparison of Langevin and non-Langevin optimizers
   1 Dense (Fully connected) networks
   2 Convolutional networks
   3 Highway networks

3 Layer-Langevin algorithm
   1 Definition and simple example
   2 Application to deep architectures for image classification
Consider a training problem with parameter $\theta$ and data $\mathcal{D}$ and learning rate $\gamma$:

### Gradient Descent versus Langevin Gradient Descent

<table>
<thead>
<tr>
<th>(Stochastic) Gradient:</th>
<th>$g_{n+1} = \nabla_\theta V(\theta_n; \mathcal{D}_{n+1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Stochastic) Gradient Descent:</td>
<td>$\theta_{n+1} = \theta_n - \gamma_{n+1} g_{n+1}$,</td>
</tr>
<tr>
<td>Langevin (Stochastic) Gradient Descent:</td>
<td>$\theta_{n+1} = \theta_n - \gamma_{n+1} g_{n+1} + \sigma \sqrt{\gamma_{n+1}} \mathcal{N}(0, I_d)$,</td>
</tr>
</tbody>
</table>

- Introduced in a Bayesian setting Welling and Teh (2011)
- The small white noise adds learning regularization
- Allows to escape from traps for the gradient descent: local minima, saddle points
- Adding noise is known to improve the learning in some cases Neelakantan et al. (2015); Anirudh Bhardwaj (2019); Gulcehre et al. (2016)
For some preconditioner rule $P_{n+1}$ depending on the previous updates of the gradient:

Preconditioned Gradient Descent: $\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1}$,

Preconditioned Langevin: $\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1} + \sigma \sqrt{\gamma_{n+1}} \mathcal{N}(0, P_{n+1})$

- Per-dimension adaptive step size
- Typical examples: Adam, RMSprop, Adadelta...
- Li et al. (2016); Ma et al. (2015); Patterson and Teh (2013); Simsekli et al. (2016) compares the benefits of noisy and/or preconditioned optimizers
Very deep neural networks are crucial, in particular in image classification He et al. (2016)

However much more difficult to train: much more "non-linear", local traps, vanishing gradients

Neelakantan et al. (2015): hints that noisy optimizers bring more improvements in this very deep setting

**Figure**: Architecture of the VGG-16 network for an input image of size 224 × 224.
Objectives

- Side-by-side comparison of preconditioned Langevin versus their respective non-Langevin counterparts: Adam vs L-Adam, RMSprop vs L-RMSprop etc
- We progressively increase the depth of the network
- Based on this heuristic, we introduce the Layer Langevin algorithm: Add noise only to some layers of the network
- Test Langevin and Layer Langevin algorithms on deep image analysis architectures
We compare Preconditioned Langevin optimizers with their non-Langevin counterparts while increasing the depth of the network on:

- Fully connected (Dense) neural networks
- Convolutional layers followed by dense layers,

on the MNIST, CIFAR-10 and CIFAR-100 datasets.

**Figure:** MNIST image dataset

**Figure:** CIFAR-10 image dataset
Results for dense (fully connected) networks

![Graphs showing test accuracy and train loss for different depths of neural networks on the MNIST dataset using Langevin algorithms compared with their non-langevin counterparts.](image)

**Figure:** Training of neural networks of various depths on the MNIST dataset using Langevin algorithms compared with their non-langevin counterparts. (a): 3 hidden layers, (b): 20 hidden layers, (c): 30 hidden layers, (d): 40 hidden layers.
Results for convolutional layers

Figure: Training of convolutional neural networks on the CIFAR-10 dataset. (a): 10 hidden dense layers, (b): 30 hidden layers.

⇒ The deeper the network is, the greater are the gain provided by Langevin optimizers.
To deal with very deep networks, highway networks Srivastava et al. (2015) introduce parametrized residual connection:

\[ y = T_{\theta_T}(x) \cdot D_{\theta_D}(x) + (1 - T_{\theta_T}(x)) \cdot x, \]

where \( T \) and \( D \) are dense or convolutional layers.

Figure: Training of a highway neural network with 80 highway hidden layers on the CIFAR-10 dataset.

\( \implies \) The previous conclusion is still true but only from a larger depth.
**Idea:** The deepest layers of the network bear the most non-linearities are more subject to Langevin optimization

\[
\theta_{n+1}^{(i)} = \theta_n^{(i)} - \gamma_{n+1} [P_{n+1} \cdot g_{n+1}]^{(i)} + 1_{i \in \mathcal{J}} \sigma \sqrt{\gamma_{n+1}} [\mathcal{N}(0, P_{n+1})]^{(i)},
\]

(1)

where $\mathcal{J}$: subset of weight indices; $P_n$: preconditioner.

We choose $\mathcal{J}$ to be the first $k$ layers.
Figure: Layer Langevin method comparison on a dense neural network with 30 hidden layers on the MNIST dataset.
Typical architecture in image recognition: Succession of convolutional layers with non-linearities (ReLU); the dimensions (width and height) of the image are progressively reduced while the number of channels is progressively augmented Simonyan and Zisserman (2015).

Depth is crucial.

Residual connections: each layer behaves in part like the identity layer to pass the information through the successive layers He et al. (2016); Huang et al. (2017).

Figure: ResNet elementary block
Layer Langevin for training of ResNet-20

**Test accuracy**

- Adam
- LL-Adam 30%
- RMSprop
- LL-RMSprop 30%
- Adadelta
- LL-Adadelta 30%

**Train loss**

- CIFAR-10
- CIFAR-100

Figure: Layer Langevin method comparison for the training of ResNet-20.
### Table: Final test accuracy values obtained for ResNet

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Adam</th>
<th>LL-Adam</th>
<th>RMSprop</th>
<th>LL-RMSprop</th>
<th>Adadelta</th>
<th>LL-Adadelta</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>76.95 %</td>
<td>77.39 %</td>
<td>84.29 %</td>
<td>85.14 %</td>
<td>75.23 %</td>
<td>75.74 %</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>45.33 %</td>
<td>45.41 %</td>
<td>55.15 %</td>
<td>55.68 %</td>
<td>42.28 %</td>
<td>43.84 %</td>
</tr>
</tbody>
</table>

### Table: Final test accuracy values on the CIFAR-10 dataset with DenseNet architecture.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Adam</th>
<th>LL-Adam</th>
<th>RMSprop</th>
<th>LL-RMSprop</th>
<th>Adadelta</th>
<th>LL-Adadelta</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>87.81 %</td>
<td>88.16 %</td>
<td>57.59 %</td>
<td>57.56 %</td>
<td>71.64 %</td>
<td>72.72 %</td>
</tr>
</tbody>
</table>
Thank you for your attention!


