Optimizing Neural-network Learning Rate by Using a Genetic Algorithm with Per-epoch Mutations

Yasusi Kanada
Hitachi, Ltd.
Contents

► Introduction

► Proposed learning method (LOG-BP)

► Two application results
  ■ Pedestrian recognition (Caltech benchmark)
  ■ Hand-written digit recognition (MNIST benchmark)

► Conclusion and future work
Introduction

 ► **Two difficult problems concerning BP** (back propagation)
   
   ■ **Decision (or scheduling) of learning rate**
   
     • Constant or prescheduled learning rates — not adaptive
     • Adaptive scheduling methods — have sensitive hyper parameters difficult to be tuned.

   ■ **To control locality of search properly**
   
     • The gradient descent algorithm, including SGD, does not search the space globally.
     • To find a better solution efficiently, multiple trials are required.
Proposal: LOG-BP (the learning-rate optimizing genetic back-propagation) learning method

- A method of new combination of BP and GA (genetic algorithm)
- Multiple neural networks run in parallel.
- Per-epoch genetic operations are used.
Outline of LOG-BP

► Multiple “individuals” (neural networks) learn and search for a best network in parallel in LOG-BP.

► Each individual contains a chromosome $c$.

- $c = (\eta; w_{11}, w_{12}, \ldots, w_{1n_1}, b_1; w_{21}, w_{22}, \ldots, w_{2n_2}, b_2; \ldots; w_{N1}, w_{N2}, \ldots, w_{Nn_N}, b_N)$

  $\eta$: learning rate,
  $w_{ij} (1 \leq j \leq n_i)$: weights of $i$-th layer of the network,
  $b_i$: bias of $i$-th layer.

► A mutation-only GA is applied to these chromosomes.
Learning algorithm of LOG-BP

**Initialization**

Randomize the weights and learning rates

| c1 | \( W_{10} \) | \( \eta_1 \) | c2 | \( W_{20} \) | \( \eta_2 \) | ... | \( c_n \) | \( W_{n0} \) | \( \eta_n \) |

**Epoch 1**

1.1 Learning by back-propagation (using stochastic gradient-descent with mini-batch)

| c1 | \( W_{11} \) | \( \eta_1 \) | c2 | \( W_{21} \) | \( \eta_2 \) | ... | \( c_n \) | \( W_{n1} \) | \( \eta_n \) |

Weights are muted.

1.2 Evaluation (calculating validation losses) (least-square errors)

| c1 | \( e_{11} \) ... best | c2 | \( e_{21} \) ... worst | \( c_n \) | \( e_{n1} \) |

1.3 Selection and mutation (no crossover)

Duplicate and mute \( c_1 \).

Kill \( c_2 \).

| c1 | \( W_{11} \) | \( \eta_1 \) | \( c_1' \) | \( W_{11} \) | \( \eta_1' \) | ... | \( c_n \) | \( W_{n1} \) | \( \eta_n \) |

\( \eta_1' = f \eta_1 \) (probability of 0.5)

\( \eta_1' = \eta_1 / f \) (probability of 0.5) \((f > 1)\)
Application to Pedestrian Recognition

► Caltech Pedestrian Dataset
  ■ A famous pedestrian detection benchmark that contains videos with more than 190,000 “small” pedestrians.
  ■ Sets of training data and test data, both of which are 24×48- and 32×64-pixel images were generated.

► Network: CNN2 and CNN3
  ■ Convolutional neural networks (CNNs) with two/three convolution layers

► Environment for computation
  ■ Deep learning environment: Theano
  ■ GPU: NVIDIA GeForce GTX TITAN X
Pedestrian: Change of Learning Rate

Example: CNN2

(a) A trial with 12 individuals (filters = [16, 26])

Learning rate usually decreases.

(b) A trial with 12 individuals (filters = [16, 32])

The learning rate may increase if the initial value is too small.

Learning rate may become mostly stationary at some epoch.
Example: CNN3

(a) A trial with 12 individuals (mutation rate = 8.3% (1/12))

(b) A trial with 24 individuals (mutation rate = 4.2% (1/24))

The decrease rate may be exponential.

Learning rate continuously decrease by more than three orders of magnitude.

Learning rate may become mostly stationary at some epoch.
Application to MNIST (Character Recognition)

► MNIST benchmark
  ■ A set of hand-written digit images (28×28) containing a training set with 60,000 samples and a test set with 10,000 samples.

► Adaptive learning rate is not required!
  ■ The learning rate during the whole learning process of LOG-BP was around 0.1.

► Summary: LOG-BP may still have benefits in terms of parallel-search performance.
MNIST: Performance (CNN3, Examples)

- **Mutation rate 4%**
  - Quicker convergence & better final error rate

- **Mutation rate 2%**

- **Mutation rate 0**

<table>
<thead>
<tr>
<th>Error rate (%)</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
<th>1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (S)</td>
<td>1000</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
<td>5000</td>
</tr>
</tbody>
</table>
# MNIST: Performance (CNN3, Statistics)

## Convergence time and final error rate

<table>
<thead>
<tr>
<th>Mutation rate</th>
<th>Average convergence time (std. dev., s)</th>
<th>Final error rate (std. dev., %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4%</td>
<td>$2.6 \times 10^3 (0.5 \times 10^3)$</td>
<td>0.82 (0.01)</td>
</tr>
<tr>
<td>2%</td>
<td>$4.9 \times 10^3 (2.6 \times 10^3)$</td>
<td>0.84 (0.07)</td>
</tr>
<tr>
<td>0%</td>
<td>$5.6 \times 10^3 (0.9 \times 10^3)$</td>
<td>0.86 (0.03)</td>
</tr>
</tbody>
</table>
Conclusion

► LOG-BP that combines BP and a GA by a new manner is proposed.

► LOG-BP solves two problems concerning BP.
  ■ Scheduling of learning rate
  ■ Controlling locality of search

► Two benchmarks show high performance of LOG-BP.
  ■ The MNIST benchmarking suggests advantages of LOG-BP over conventional SGD algorithms.
  ■ LOG-BP will make machine learning less dependent to properties of various applications.