The RPROP algorithm

Resilient propagation for NN

Contents

- Backpropagation learning
- The RPROP algorithm
- A comparison to other propagation algorithms through experiments

Backpropagation Learning

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

where, in regular gradient descent,

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}}(t) \quad .$$

With a momentum term:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}}(t) + \mu \Delta w_{ij}(t-1) \quad .$$

What makes RPROP special?

- Adaptation of the weight-step is not “blurred” by gradient behavior
- Instead, each weight has an individual evolving update-value
- The weight-step is only determined by its update-value and the *sign* of the gradient

RPROP: Weight-step Rule

$$\Delta w_{ij}(t) = \begin{cases} +\Delta_{ij}(t) & , \quad \text{if } \dfrac{\partial E}{\partial w_{ij}}(t) > 0 \\ -\Delta_{ij}(t) & , \quad \text{if } \dfrac{\partial E}{\partial w_{ij}}(t) < 0 \\ 0 & , \quad \text{otherwise} \end{cases}$$

exception :

$$\Delta w_{ij}(t) = -\Delta w_{ij}(t-1) \quad , \quad \text{if} \quad \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) < 0$$

To avoid double punishment, let $\dfrac{\partial E}{\partial w_{ij}}(t) = 0$

RPROP: Learning Rule

$$\Delta_{ij}(t) = \begin{cases} \eta^{+} \cdot \Delta_{ij}(t-1) & , \quad \text{if } s_{ij} > 0 \\ \eta^{-} \cdot \Delta_{ij}(t-1) & , \quad \text{if } s_{ij} < 0 \\ \Delta_{ij}(t-1) & , \quad \text{otherwise} \end{cases}$$

$$\text{where } s_{ij} = \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) \quad .$$

$$\eta^{+} = 1.2$$

$$\eta^{-} = 0.5$$

RPROP in program code

```
npos, nneg = 1.2, 0.5
dmax, dmin = 50.0, 0.000001

def update(weights):

    compute_gradients(weights)

    for (w, dw, d, prevE, E) in weights:

        switch (sign(prevE * E)):
            case +1:
                d = min(d * npos, dmax)
                dw = d * sign(E)
            case -1:
                d = max(d * nneg, dmin)
                E = 0
            case 0:
                dw = d * sign(E)

        w = w - dw
        prevE = E
```

Experiments

- Algorithms
  - Backpropagation (BP)
  - SuperSAB (SSAB)
  - Quickprop (QP)
  - Resilient propagation (RPROP) [our hero]
- Problems
  - The 10-5-10 encoder problem
  - The 12-2-12 encoder problem
  - Nine Men’s Morris

Experiment: 10-5-10

- A neural network with 10 input and output neurons, and 5 hidden neurons

Algorithm	**μ / v**	**Epochs**	**σ**
BP	0.0	121	30
SSAB	0.8	55	11
QP	1.75	21	3
RPROP	-	19	3

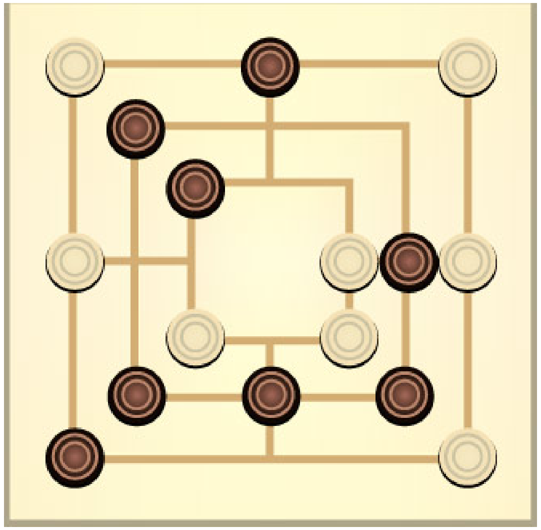
Experiment: 12-2-12

- A ‘tight’ neural network with 12 inputs and outputs, and just 2 hidden neurons

Algorithm	**μ / ν**	**Epochs**	**σ**
BP	div.	>15000	-
SSAB	0.95	534	90
QP	1.3	405	608
RPROP	-	322	112

Experiment: Nine Men's Morris

- A strategy board game for two players
- First, the players place their nine 'men' on the board, trying to get them in lines of three
- When all have been placed, the players take turns moving them



Experiment: Nine Men's Morris

- Two identical 60-30-10-1 networks linked together to play the endgame of Nine Men's Morris
- Two alternative moves are presented and the 'comparator neuron' is to decide which one is better

Algorithm	**μ / v**	**Epochs**	**σ**
BP	0.5	98	34
SSAB	0.9	34	4
QP	1.75	34	12
RPROP	-	23	3

Summary

- Requires no parameter tuning
- Learning and adaptation only affected by the *sign* of the partial derivative
  - Computer friendly
  - Learning is equally spread over the network
- RPROP is a fast learner