Task Sheet 2, Solution

Cristiano Alessandro (alessandro@ifi.uzh.ch), Mathias Weyland (mathias.weyland@uzh.ch)

Solutions

1. Both the Octave/Matlab as well asathe pen and paper solution is provided below:

Octave and Matlab

Nobody noted the scalar factor of 2 with $\beta = 1$, so we give the resuts for $\beta = \frac{1}{2}$ instead to match your solution.

a)

```
ativ = @(W,x)(1 + exp(-W*x)).^-1 ;
```

b)

```
outPut=@(Wa,Wb,x)ativ(Wb,[ativ(Wa,x);1]);
```

c)

```
changeW2=-delta*err.*deriv.*(Vj');
```

d)

```
changeW1= ...
  -delta*[sumation(1)*ones(1,3);sumation(2)*ones(1,3)].*[ini';ini'].*deriv2;
```

e) The smaller the learning rate, the slower the convergence.

Pen and paper

Note that we are assuming $\beta = 2$.

a)

$$h_1(i_1, i_2) = g(i_1 w_{11} + i_2 w_{12} + w_{13}) = \frac{1}{1 + \exp(-2(i_1 w_{11} + i_2 w_{12} + w_{13}))}$$

$$h_2(i_1, i_2) = g(i_1 w_{21} + i_2 w_{22} + w_{23}) = \frac{1}{1 + \exp(-2(i_1 w_{21} + i_2 w_{22} + w_{23}))}$$

b)

$$\begin{aligned} o(i_1, i_2) = &\quad g(h_1(i_1, i_2)w_{o1} + g_2(i_1, i_2)w_{o2} + w_o 3) \\ = &\quad \frac{1}{1+\exp(-2(h_1(i_1,i_2)w_{o1}+g_2(i_1,i_2)w_{o2}+w_o 3))} \end{aligned}$$

c) Assuming $o = o(i_1, i_2)$ and $h_j = h_j(i_1, i_2)$:

$$\Delta w_{oj} = 2\eta o(1-o)(\zeta - o)h_j$$

d)

$$\Delta w_{jk} = 4\eta h_j(1-h_j)w_{oj} \cdot o(1-o)(\zeta - o)i_k$$

e) Table for $\beta = 2$:

w_{11}	0.02	0.0200
w_{12}	-0.05	-0.0506
w_{21}	-0.03	-0.0300
w_{22}	0.03	0.0306
w_{13}	-0.05	-0.0506
w_{23}	-0.003	-0.0024
w_{o1}	-0.01	0.0479
w_{o2}	0.01	0.0760
w_{o3}	-0.03	0.0986
i_1	0	1
i_2	1	0
h_1	0.4502	
h_2	0.5135	
o	0.4853	
Δw_{11}	0	
Δw_{12}	-0.6364×10^{-3}	
Δw_{21}	0	
Δw_{22}	0.6423×10^{-3}	
Δw_{13}	-0.6364×10^{-3}	
Δw_{23}	0.6423×10^{-3}	
Δw_{o1}	0.0579	
Δw_{o2}	0.0660	
Δw_{o3}	0.1286	

2. Learning:

a) The trained neural network fits features of the training data that are not related to the model to be lerned (e.g. noise). As a result, the trained model will generalize badly.

b) The trainig data should not be used to assess the quality of the learned model. A new dataset should be used to check if the learning procedure has captured the model underlying the data (as opposed to the specific features of the training samples only).

c) Please see p. 42 of the script.

d) N-fold cross validation can be used to estimate the generalization error. The generalization-vs-#hidden-unit plot can be computed by sequentially adding a hidden unit and evaluating the generalization error. The optimal number of unit correspond to the minimum of such a plot.

3. The momentum can help to speed up learning in flat areas of the error surface. Furthermore it can help to find the way out of (flat) local minima. By the same mechanisms, though, it can also cause the learning algorithm to overshoot out of the global minimum.

4. It becomes 0 in both cases. As a consequence, the deltas also become zero and no error will be propagated (although the error itself could be very large). To avoid this add a small constant to the derivative. This way the deltas are only zero if the error is zero.

5. Cascade correlation:

   a) Please refer to p. 43–45 of the script.

   b) It builds the network in stage: hidden nodes are added only when it is needed based on the data to be approximated. Therefore it ameliorates the problem of overfitting by autmatically "growing" the optimal netwok architecture.

6. More cascade correlation:

   a) With a higher patience parameter, we get a lower final error, higher number of epochs used, and less hidden nodes are recruited. Lower patience parameter, we get a higher final error, lower number of epochs used, and more hidden nodes are recruited.

   b) In general, cascade correlation algorithm is faster than back-propagation in reducing the error.