Sensitivity Analysis

Sensitivity Analysis

Problem of Multi-Layer Perceptrons:

- The knowledge that is learned by a neural network
  is encoded in matrices/vectors of real-valued numbers and
  is therefore often difficult to understand or to extract.
- Geometric interpretation or other forms of intuitive understanding are possible
  only for very simple networks, but fail for complex practical problems.
- Thus a neural network is often effectively a *black box*,
  that computes its output, though in a mathematically precise way,
  in a way that is very difficult to understand and to interpret.

Idea of Sensitivity Analysis:

- Try to find out to which inputs the output(s) react(s) most sensitively.
- This may also give hints which inputs are not needed and may be discarded.

Sensitivity Analysis

Question: How important are different inputs to the network?

Idea: Determine change of output relative to change of input.

$$\forall u \in U_{\text{in}}: \qquad s(u) = \frac{1}{|L_{\text{fixed}}|} \sum_{l \in L_{\text{fixed}}} \sum_{v \in U_{\text{out}}} \frac{\partial \operatorname{out}_v^{(l)}}{\partial \operatorname{ext}_u^{(l)}}.$$

Formal derivation: Apply chain rule.

$$\frac{\partial \operatorname{out}_v}{\partial \operatorname{ext}_u} = \frac{\partial \operatorname{out}_v}{\partial \operatorname{out}_u} \frac{\partial \operatorname{out}_u}{\partial \operatorname{ext}_u} = \frac{\partial \operatorname{out}_v}{\partial \operatorname{net}_v} \frac{\partial \operatorname{net}_v}{\partial \operatorname{out}_u} \frac{\partial \operatorname{out}_u}{\partial \operatorname{ext}_u}.$$

Simplification: Assume that the output function is the identity.

$$\frac{\partial \operatorname{out}_u}{\partial \operatorname{ext}_u} = 1.$$

For the second factor we get the general result:

$$\frac{\partial \operatorname{net}_v}{\partial \operatorname{out}_u} = \frac{\partial}{\partial \operatorname{out}_u} \sum_{p \in \operatorname{pred}(v)} w_{vp} \operatorname{out}_p = \sum_{p \in \operatorname{pred}(v)} w_{vp} \frac{\partial \operatorname{out}_p}{\partial \operatorname{out}_u}.$$

This leads to the recursion formula

$$\frac{\partial \operatorname{out}_v}{\partial \operatorname{out}_u} = \frac{\partial \operatorname{out}_v}{\partial \operatorname{net}_v} \frac{\partial \operatorname{net}_v}{\partial \operatorname{out}_u} = \frac{\partial \operatorname{out}_v}{\partial \operatorname{net}_v} \sum_{p \in \operatorname{pred}(v)} w_{vp} \frac{\partial \operatorname{out}_p}{\partial \operatorname{out}_u}.$$

However, for the first hidden layer we get

$$\frac{\partial \operatorname{net}_v}{\partial \operatorname{out}_u} = w_{vu}, \qquad \text{therefore} \qquad \frac{\partial \operatorname{out}_v}{\partial \operatorname{out}_u} = \frac{\partial \operatorname{out}_v}{\partial \operatorname{net}_v} w_{vu}.$$

This formula marks the start of the recursion.

Sensitivity Analysis

Consider as usual the special case with

- output function is the identity,
- activation function is logistic.

The recursion formula is in this case

$$\frac{\partial \operatorname{out}_v}{\partial \operatorname{out}_u} = \operatorname{out}_v(1 - \operatorname{out}_v) \sum_{p \in \operatorname{pred}(v)} w_{vp} \frac{\partial \operatorname{out}_p}{\partial \operatorname{out}_u}$$

and the anchor of the recursion is

$$\frac{\partial \operatorname{out}_v}{\partial \operatorname{out}_u} = \operatorname{out}_v(1 - \operatorname{out}_v) w_{vu}.$$

Sensitivity Analysis

Attention: Use weight decay to stabilize the training results!

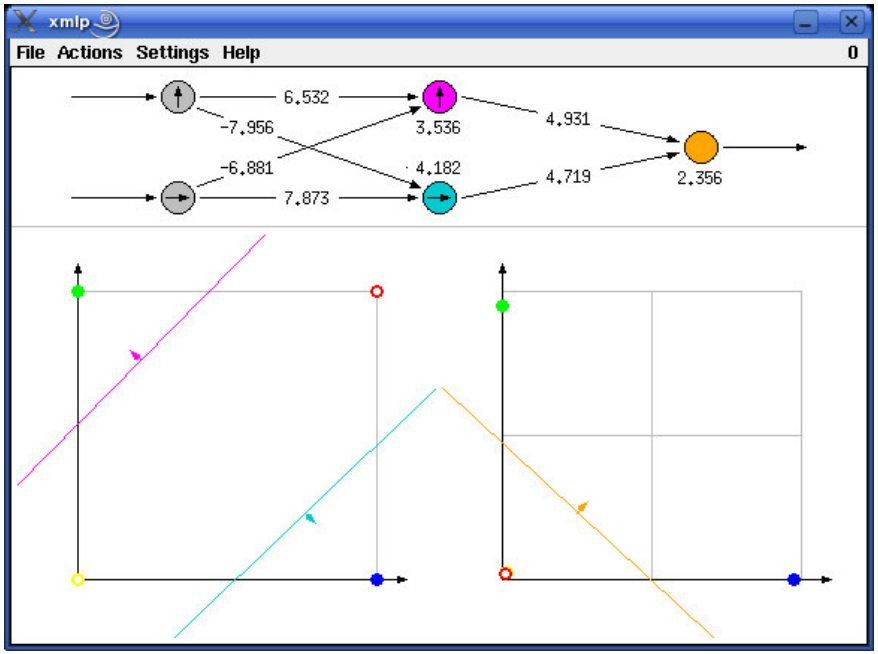
$$\Delta w(t) = -\frac{\eta}{2}\nabla_w e(t) - \xi\, w(t),$$

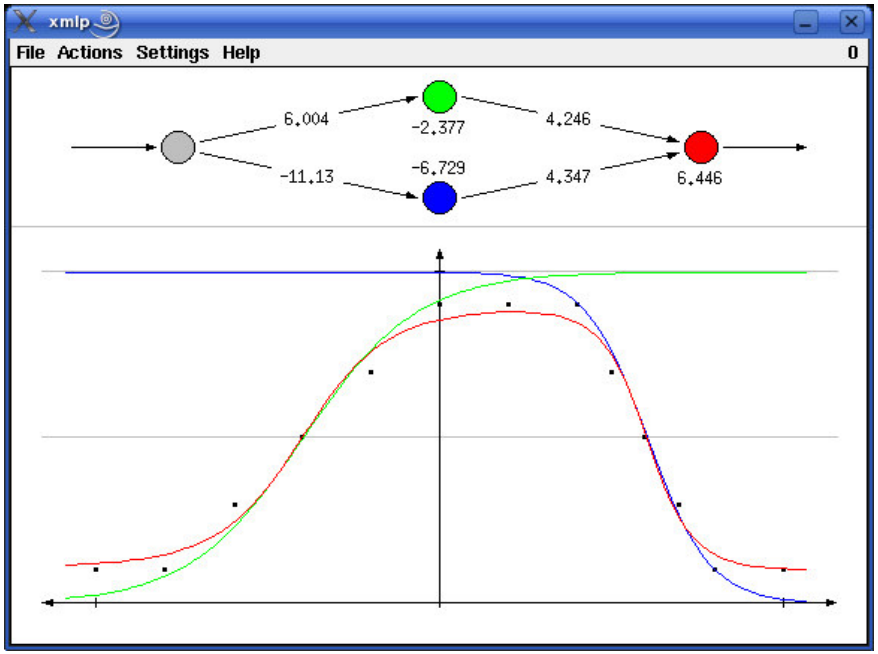
- Without weight decay the results of a sensitivity analysis can depend (strongly) on the (random) initialization of the network.

Example: Iris data, 1 hidden layer, 3 hidden neurons, 1000 epochs

attribute	$\xi = 0$				$\xi = 0.0001$			
sepal length	0.0216	0.0399	0.0232	0.0515	0.0367	0.0325	0.0351	0.0395
sepal width	0.0423	0.0341	0.0460	0.0447	0.0385	0.0376	0.0421	0.0425
petal length	0.1789	0.2569	0.1974	0.2805	0.2048	0.1928	0.1838	0.1861
petal width	0.2017	0.1356	0.2198	0.1325	0.2020	0.1962	0.1750	0.1743

Demonstration Software: xmlp/wmlp

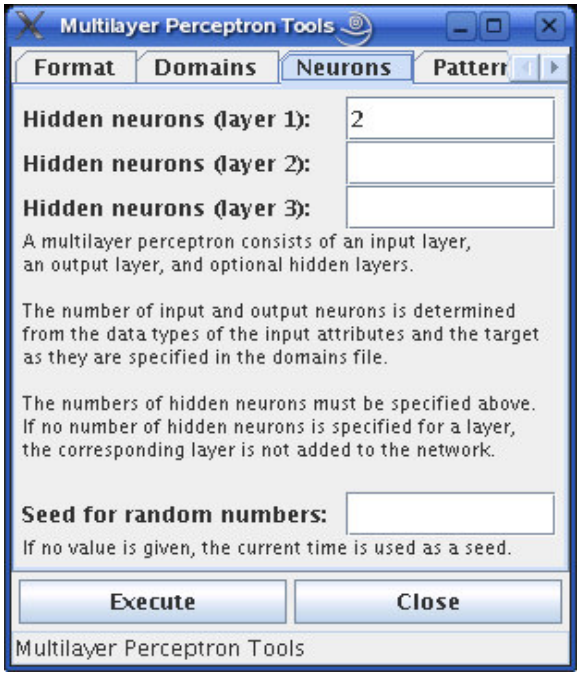


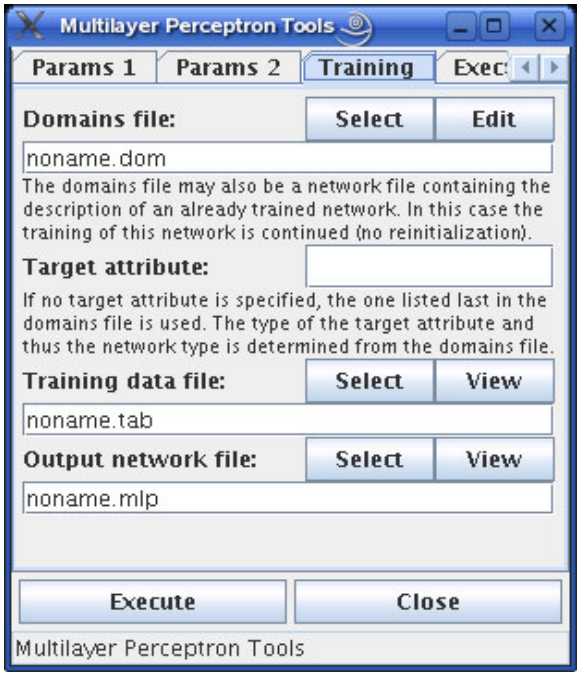


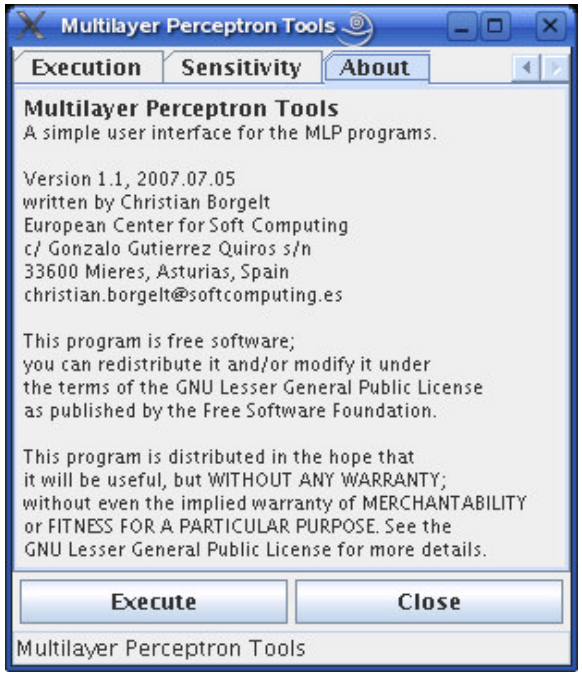
Demonstration of multi-layer perceptron training:

- Visualization of the training process
- Biimplication and Exclusive Or, two continuous functions
- `http://www.borgelt.net/mlpd.html`

Multi-Layer Perceptron Software: mlp/mlpgui







Software for training general multi-layer perceptrons:

- Command line version written in C, fast training
- Graphical user interface in Java, easy to use
- `http://www.borgelt.net/mlp.html`
  `http://www.borgelt.net/mlpgui.html`