Chapter 2
Principles of Sensitivity Analysis

Sensitivity refers to how a neural network output is influenced by its input and/or weight perturbations. Sensitivity analysis dates back to the 1960s, when Widrow investigated the probability of misclassification caused by weight perturbations, which are caused by machine imprecision and noisy input (Widrow and Hoff, 1960). In network hardware realization, such perturbations must be analyzed prior to its design, since they significantly affect network training and generalization. The initial idea of sensitivity analysis has been extended to the optimization of neural networks, such as through sample reduction, feature selection, and critical vector learning.

2.1 Perturbations in Neural Networks

Perturbations of neural networks are caused by machine imprecision and/or input noise. For the purpose of analysis, these perturbations can be simulated by embedding disturbance to the original inputs or connection weights. Perturbation analysis allows the study of the characteristics of a function under small perturbations of the function’s parameter (Holtzman, 1992; Zurada et al., 1997). Perturbation analysis is also important also when assessing network robustness against input noise to measure the uncertainty or fluctuations. In perturbation analysis we are interested in evaluating the disturbance in the function’s response to small perturbations in its parameters. Fig. 2.1 shows how to investigate the effects of perturbations to neural networks.

Assuming that the performance function is differentiable, the relationship between the perturbed response of this function and parameter perturbations is expressed by a Taylor expansion of the function. For example, for a one-dimensional cost function \( g \)

\[
g (\Theta + \Delta \Theta) = g (\Theta) + \frac{\Delta \Theta}{1!} g' (\Theta) + \frac{\Delta \Theta^2}{2!} g'' (\Theta) + \cdots \tag{2.1}
\]

where \( \Theta \) is a parameter of the function and typically includes weight \( W \) and input \( X \); \( \Delta \Theta \) is a small perturbation of \( \Theta \). The performance cost function \( g \) is usually taken...
by the noise-to-signal ratio (NSR) or expectation of decision errors. The Taylor expansion shows that the derivatives of the function with respect to the perturbed parameter encapsulate the characteristics of that function under the perturbations. Ideally, when $\Delta \theta \to 0$, $\frac{\Delta \theta}{\Delta x} g'(\theta) + \frac{\Delta \theta^2}{2!} g''(\theta) + \cdots \to 0$.

Eq. (2.1) shows that the derivatives play a very important role in determining the influence of parameter perturbations on the output of the performance function. Sensitivity analysis is applied to investigate how the derivatives can be used to quantify the response of the system to parameter perturbations, and how these derivatives can be calculated.

Sensitivity analysis techniques differ mainly in the cost function used, the order of the derivatives that are considered, whether the analysis is in continuous time or for discrete time intervals, and the way in which the derivatives are calculated. Due to computational considerations, sensitivity analysis is based on approximations of Eq. (2.1), usually first-order or second-order approximations. The higher the order of the approximation, the more accurate but more complex and time consuming the process. Usually, sensitivity analysis is done at discrete time intervals for that time interval only. Sensitivity analysis can also be performed for continuous time models, referred to as stochastic analysis (Koda, 1995; 1997).

### 2.2 Neural Network Sensitivity Analysis

Without loss of generality, let us consider a neural network performing a nonlinear, differentiable mapping $\Gamma: \mathbb{R}^I \to \mathbb{R}^K$, from input $x = (x_1, x_2, \ldots, x_I)$ to output $o = (o_1, o_2, \ldots, o_K)$. Suppose $x^{(n)} \in \Omega$, where $\Omega$ is an open set. Since $o$ is differentiable at $x^{(n)}$ we have

$$o(x + \Delta x) = o(x^{(n)}) + J(x^{(n)})\Delta x + g(\Delta x)$$

(2.2)

where
is the Jacobian matrix and

\[
\lim_{\Delta x \to 0} \frac{g(\Delta x)}{\|\Delta x\|} = 0 \tag{2.4}
\]

Fig. 2.2 illustrates geometrical interpretation of Eq. (2.2) in space \( \mathbb{R}^K \). Point \( o(x^{(n)}) \) represents the nominal response of the neural network for the \( n \)th element of the training set \( x^{(n)} \). The disturbance \( \Delta x \) of the input vector causes the perturbed response at \( o(x^{(n)} + \Delta x) \). This response can be expressed as a combination of three vectors as indicated in Eq. (2.2).

The sensitivity analysis can be used for different purposes in neural networks (Engelbrecht, 1999):

**Optimization.** The calculation of the gradient of a function forms an important part of optimization. One of the first uses of sensitivity analysis is therefore in optimization problems (Cao, 1985; Holtzman, 1992). In neural networks, derivatives of the objective function with respect to the weights are computed to locate minima by driving these derivatives to 0. Second order derivatives have also been used to develop more sophisticated optimization techniques to improve convergence and accuracy. Koda (1995, 1997) employed stochastic sensitivity analysis to compute the gradient for time-dependent networks such as recurrent neural networks.

**Robustness.** Neural network robustness and stability analysis is the study of the conditions under which the outcome of the neural network changes. This study is important for hardware implementation of neural networks to ensure stable

---

**Fig. 2.2** Illustration of output disturbance caused by input disturbance
networks that are not adversely affected by weight, external input and activation function perturbations (Alippi, Piuri and Sami, 1995; Oh and Lee, 1995). Instead of using derivatives to compute the gradient of the objective function with respect to the weights, Jabri and Flower (1991) use differences to approximate the gradient, thereby significantly reducing hardware complexity.

**Generalization.** Fu and Chen (1993) state good generalization must imply insensitivity to small perturbations in inputs. They derive equations to compute the sensitivity of the neural network output vector to changes in input values, and show under what conditions global neural network sensitivity can be reduced. For example, using small slopes for the sigmoid activation function, using as small as possible weights, reducing the number of units, and ensuring activation levels close to 0 or 1 will reduce network sensitivity. Choi and Choi (1992) derive a neural network sensitivity norm which expresses the sensitivity of the neural network output with respect to input perturbations. This neural network norm is then used to select from sets of optimal weights the weight set with lowest neural network sensitivity, which results in the best generalization.

**Measure of nonlinearity.** Lamers, Kok and Lebret (1998) use the variance of the sensitivity of the neural network output to input parameter perturbations as a measure of the nonlinearity of the data set. This measure of nonlinearity is then used to show that the higher the variance of noise injected to output values, the more linearized the problem.

**Causal inference.** Sensitivity analysis has been used to assess the significance of model inputs. Engelbrecht, Cloete and Zurada (1995) use exact derivative calculations to compute the significance values which have a high influence on the neural network output. Goh (1993) derived a similar method using differences to approximate the gradient of the neural network output function with respect to inputs.

**Selective learning.** Hunt and Deller (1995) use weight perturbation analysis to determine the inference each pattern has on weight changes during training. Only patterns that exhibit a high influence on weight changes are used for training. Engelbrecht (1998) presents new active learning models based on sensitivity analysis, which use a measure of pattern informativeness to dynamically select patterns during training.

**Decision boundary visualization.** Goh (1993) uses an approximation to the derivatives of the neural network output function with respect to inputs to graphically visualize decision boundaries. Engelbrecht (1998) shows how exact derivative calculations can be used to locate and visualize decision boundaries. Victor (1998) uses the decision boundary algorithm to improve the accuracy of rules extracted from trained neural networks in a cooperative learning environment.

**Pruning.** Sensitivity analysis has been applied extensively to neural network pruning. One technique is to compute the sensitivity of the objective function with respect to neural network parameters (Le Cun, 1990; Moody et al., 1995). Another method of sensitivity analysis pruning is to compute the sensitivity of the neural network output function to parameter perturbations (Zurada, 1994, 1997).
Learning derivatives. Basson and Engelbrecht (1999) developed a new learning algorithm for feedforward neural networks that also learns the first-order derivatives of the neural network output with respect to each input unit while learning the underlying function. The neural network consists of two parts, one representing the learned function, and the other representing the derivatives of the learned function. Concepts from sensitivity theory are used to create a training set for the training of the derivative part of the neural network using gradient descent.

2.3 Fundamental Methods of Sensitivity Analysis

In system sensitivity theory, some sensitivity functions are introduced, such as the output sensitivity, the trajectory sensitivity and the performance-index sensitivity functions (Frank, 1978). All the methods of neural network sensitivity analysis can be divided into two categories, namely, geometrical approach and statistical approach. In 1962, Hoff used an \( n \)-dimensional hypersphere to model Adaline for sensitivity analysis (Hoff, 1962), which was further simplified in Glanz (1965). After two decades, Winter (1989) was the first one to derive an analytical expression for the probability of error in Madaline caused by weight perturbations. Stevenson continued Winter’s work and established the sensitivity of Madaline to weight error (Stevenson, 1990; Stevenson, Winter and Widrow, 1990). A milestone work was done by Piché, who used a statistical approach to relate the output error to the change of weights for an ensemble of Madalines, with several activation functions such as linear, sigmoid, and threshold (Piché, 1992; Piché, 1995).

Figure 1.6 can be treated as a general neural network model. A neural network can have \( L \) layers, and each layer \( l \) (\( 0 \leq l \leq L \)) has \( n^l (n^l \geq 1) \) neurons. \( n^0 \) stands for the input layer and \( n^L \) for the output layer. Since the number of neurons in layer \( l-1 \) is equal to the output dimension of that layer, which is also equal to the input dimension of layer \( l \), the input dimension of layer \( l \) is \( n^{l-1} \). For a neuron \( i \) (\( 1 \leq i \leq n^l \)) in layer \( l \), its input vector, weight vector and output are \( X^l = \left( x^l_1 \cdots x^l_{n^{l-1}} \right)^T \), \( W^l_i = \left( w^l_{i1} \cdots w^l_{in^{l-1}} \right)^T \) and \( y^l_i = f \left( X^l \cdot W^l \right) \) respectively, where \( f(\cdot) \) is an activation function. For each layer \( l \), its input vector is \( X^l \), its weight set is \( W^l = \{ W^l_1 \cdots W^l_{n^l} \} \), and its output vector is \( Y^l = \left( y^l_1 \cdots y^l_{n^l} \right)^T \). For the network, its input is the vector \( X^1 \) or \( y^0 \), its weight is \( W \), and its output is \( Y^L \). Let \( \Delta X^l = \left( \Delta x^l_1 \cdots \Delta x^l_{n^{l-1}} \right)^T \) and \( \Delta Y^l = \left( \Delta y^l_1 \cdots \Delta y^l_{n^l} \right)^T \) be the corresponding deviations for input and output, respectively.

2.3.1 Geometrical Approach

The use of \( n \)-dimensional geometry has proven to be a valuable tool for understanding and analyzing the Adaline. The geometrical interpretation of the equation
dictating the Adaline’s input-output map is the bias for most of the analysis presented in Stevenson (1990).

The input vector \( \mathbf{X} \) in a neural network can be treated as a vector from the origin to the point \((x_1, x_2, \ldots, x_n)\) in \(n\)-space. The point \((x_1, x_2, \ldots, x_n)\) will be referred to “the tip of \(X\)” in the following discussion. In \(n\)-space, points at a distance \(r\) from the point \(c\) form a hypersphere of radius \(r\) centered at \(c\). The surface area of such a hypersphere \(A_n(r)\) is

\[
A_n(r) = 2 \sqrt{\pi^n} \Gamma \left( \frac{n}{2} \right) \cdot r^{n-1}
\]

where \(\Gamma(\bullet)\) is the Gamma function. As shown in Fig. 2.3, the connection weights, which are a vector at angle \(\theta\) to the input vector \(\mathbf{X} = (x_1, x_2, \ldots, x_n)\) in \(n\)-space, satisfy

\[
\mathbf{X} \cdot \mathbf{W} = \sum_{i=1}^{n} x_i w_i = c
\]

which is called a hyperplane for some scalar \(c\). This hyperplane is perpendicular to the vector \(\mathbf{W}\) and is at a distance \(c/|\mathbf{W}|\) from the origin. Particularly, \(HP_w\) is one of the hyperplanes passing through the origin with \(c = 0\). Similarly to \(HP_w\), \(HP_X\) denotes a hyperplane passing through the origin and perpendicular to the vector \(X\). Hence, both \(HP_w\) and \(HP_X\) divide the hypersphere in two hemi hyperspheres, i.e., \(H_w^+\) versus \(H_w^-\), and \(H_X^+\) versus \(H_X^-\), respectively. There are four lunes intersected by these four hemi hyperspheres, \(H_X^+ \cap H_W^+\), \(H_X^+ \cap H_W^-\), \(H_X^- \cap H_W^+\) and \(H_X^- \cap H_W^-\), as shown in Fig. 2.3.

As the angle between \(X\) and \(W\) is \(\theta\), both the intersections, \(H_X^+ \cap H_W^-\) and \(H_X^- \cap H_W^+\), describe lunes of angle \(\theta\) whereas the intersections \(H_X^+ \cap H_W^+\) and \(H_X^- \cap H_W^-\) both describe lunes of angle \((\pi - \theta)\). The ratio of the surface content of a lune of angle \(\theta\) to the surface content of the entire hypersphere is \(\theta/2\pi\).

Assuming binary-valued inputs, there are \(2^n\) possible input patterns for an Adaline with \(n\) variable inputs. Each input pattern corresponds to a point in \(n\)-space which lies on a hypersphere of radius \(\sqrt{n}\) centered at the origin. The Hoff

Fig. 2.3 Hypersphere approximation for sensitivity analysis
hypersphere-area approximation states that as \( n \) gets large, the points corresponding to the \( n \)-dimensional input patterns are approximately uniformly distributed over the surface of a hypersphere in \( n \)-space. Consequently, the percentage of input patterns which correspond to points on a selected region of the hypersphere can be approximated as the ratio of the surface content of the selected region to the surface content of the entire hypersphere (Hoff, 1962).

### 2.3.2 Statistical Approach

Let us consider the behavior of a multi-input/single-output mapping first. Let us assume that the connection weight vector changes from \( W^* \) to \( W = W^* + \Delta W \), where \( \Delta W \) indicates weight perturbations. Under the assumption of statistical weight perturbations, the statistical sensitivity can be defined as follows (Choi and Choi, 1992):

\[
S_p(W^*) = \lim_{\sigma \to 0} \frac{\sqrt{\text{var}[\Delta X_p(L)]}}{\sigma} \tag{2.7}
\]

where \( \Delta X_p(L) \) is the output error, \( \sigma \) is the standard deviation of each component of deviation of each component of \( \Delta W \), and \( \text{var}[...] \) is the variance. The output error vector \( \Delta X_p(L) \) in the \( l^{th} \) layer arising from weight perturbations \( \Delta W \) is given by

\[
\Delta X_p(l) = X_p(l) - X^*_p(l) \cong \sum_{k=1}^{l} C_{kl} \left(W^*, X^*_p\right) \Delta W(k)X^*_p(k-1) \tag{2.8}
\]

where the \( N_l \times N_k \) matrix \( C_{kl} \left(W^*, X^*_p\right) \) is defined as

\[
C_{kl} \left(W^*, X^*_p\right) = \prod_{n=k+1}^{l} \nabla T_n W^*_n \nabla T_k = \nabla T_l W^*_l \nabla T_{l-1} W^*_1 \cdots \nabla T_{k+1} W^*_k \tag{2.9}
\]

Here, \( T_k \) refers to the nonlinear transformation associated with the \( k^{th} \) layer, and \( \nabla T_k \) refers to its derivation with respect to the output of the \( k^{th} \) layer.

As in the case of the weight perturbation discussed above, the sensitivity to input perturbation can be also calculated by Eq. (2.7) with \( \sigma \) being the standard deviation of each component of the input perturbation \( \Delta X_p(0) \). In this respect, the output error \( \Delta X_p(L) \) in output layer \( L \) caused by input perturbation is given by:

\[
\Delta X_p(L) = C_{L,0} \left(W^*, X^*_p\right) \Delta X_p(0) \tag{2.10}
\]

To measure sensitivity for a multi-output neural network, typically, the sensitivity of each output neuron is calculated first, and then the final sensitivity is set as the maximum or the average value among all output neurons.
Piché (1995) introduced a stochastic model for the statistical analysis of sensitivity. He assumed that (1) over the ensemble of networks, the weights of a particular layer all have the same variance, (2) the weights in the networks are statistically independent and (3) the mean value of each weight of the network over the ensemble is 0. Under these assumptions, it is possible to model the ensemble of weights associated with each node of the network as a vector of random variables. He discussed the selection of weight accuracies for Madaline using a statistical approach to sensitivity analysis. According to his stochastic model, all neurons have the same activation function. All network inputs, weights, input perturbations, and weight perturbations are random variables. The sensitivity of Madaline in this model is defined as the NSR of the output layer; the output NSR of a layer of threshold Adaline is:

\[ NSR = \frac{\sigma_{\Delta y}^2}{\sigma_y^2} = \frac{4}{\pi} \sqrt{\frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2}} \]  

(2.11)

where \( \sigma_y^2, \sigma_x^2, \sigma_w^2, \sigma_{\Delta y}^2, \sigma_{\Delta x}^2 \) and \( \sigma_{\Delta w}^2 \) refer to the variances of output \( y \), input \( x \), weight \( w \), output error \( \Delta y \), input perturbation \( \Delta x \) and weight perturbation \( \Delta w \), respectively.

### 2.4 Summary

Sensitivity is initially investigated for the realization of a neural network to calculate its output perturbation caused by machine imprecision and noisy input. It is particularly useful and valuable to apply sensitivity analysis to the software design of networks, in which artificial perturbation is embedded in the training. In this paper, the principle of sensitivity analysis is discussed in detail, followed by systematic introduction to the advanced research in this area over the last two decades. All the existing techniques can be categorized into geometrical and statistical approaches. The former use hypersphere or hyper-rectangle model in \( n \)-dimensional space to analyze the deviation of the output vector caused by the perturbation of the input and weights. The latter calculate the sensitivity by noise-to-signal ratio or expectation of the output deviation.
Sensitivity Analysis for Neural Networks
Yeung, D.S.; Cloete, I.; Shi, D.; Ng, W.W.Y.
2010, VIII, 86 p. 24 illus., Hardcover
ISBN: 978-3-642-02531-0