Introduction to Neural Network
Lecturer outline

- Neural network definition
- Biological analogy
- Potential application
- Why neural networks
- Limitations of neural networks
- Network architectures
- Neuron/node/perceptron
- Learning/training Types & Rules
- Developing Neural Networks
- Practical aspect of neural computing
- Neural networks performance
- Performance factors
- References
Neural network definition

- NN is a computing system that consists of a number of simple, highly interconnected nodes (processing elements) that process information.
- A concept that tries to mimic how the human brain functions.
Biological Analogy

Dendrites

Nodes

Synapses (weights)

Axon

Impulse
Example

input – touching hot surface

Output – withdrawing your hand

The processing in your brain that led from one to the other remains hidden
Potential Applications

• Classification-given symptoms, determine the most likely disease, speech analysis, fault diagnosis

• Prediction-given wind velocity and humidity, predict evaporation rate

• Pattern association-retrieve an image from corrupted one
Potential Applications

- Data conceptualization - cluster data with many attributes
- Data filtering - smooth input signal
- Optimization - optimize the settings of process controllers, thus feeding the correct amount of additives
Clustering example

Input layer

Output layer

Input patterns

Sorted patterns

“No disease”

“Meningitis”

“Pneumonia”

“Flu”
**classification example**

**Input units**

- Cough
- Headache

**Weights**

- No disease
- Pneumonia
- Flu
- Meningitis

**Output units**

**rule**

*change weights to decrease the error*

what we got

what we wanted

error
Why neural networks?

• Compare to empirical model (curve fitting), NN model performs better for noisy or incomplete data. Better generalization
Why neural networks?

• Compare to theoretical/mechanistic model, NN model is easier to develop especially for complex, nonlinear and uncertain syst.

• Potential for online use because a trained network may take less than a second to calculate results
Limitations of Neural Networks

• Large amount of data – broad-based data set or experimental design is essential
• Long training times – however as computers become more powerful, time requirements are less
Network architectures

- Three different classes of network architectures
  - single-layer feed-forward neurons are organized
  - multi-layer feed-forward in acyclic layers
  - recurrent

- The architecture of a neural network is linked with the learning algorithm used to train
Single Layer Feed-forward

(one processing layer – the output layer)

*Input layer of source nodes*

*Output layer of neurons*
Multi layer feed-forward

(several processing layers – hidden and output layers

3-4-2 Network

Input layer

Hidden Layer

Output layer
Recurrent network

Recurrent Network with *hidden neuron*: unit delay operator $z^{-1}$ is used to model a dynamic system.
The Unit of a Neural Network

- The unit of a neural network is modeled on the biological neuron.
- The unit combines its inputs into a single value, which it then transforms to produce the output; together these are called the activation function.
Neuron/node: perceptron

\[
\sum \left( \sum_{i=1}^{m} x_i w_i + b \right) \rightarrow \varphi(v)
\]

- Input values: \( x_1, x_2, \ldots, x_m \)
- Weights: \( w_1, w_2, \ldots, w_m \)
- Bias: \( b \)
- Summing function
- Local Field: \( v \)
- Activation function: \( \varphi(\cdot) \)
- Output: \( y \)
Neuron

The neuron is the basic information processing unit of a NN. It consists of:

A set of **links**, describing the neuron inputs, with **weights** $W_1, W_2, \ldots, W_m$

An **adder** function (linear combiner) for computing the weighted sum of the inputs (real numbers):

$$v = \sum_{j=1}^{m} W_j x_j$$

**Activation function** (squashing function) $\varphi$ for limiting the amplitude of the neuron output.

$$y = \varphi(v + b)$$
Bias as extra input

- The bias is an external parameter of the neuron. It can be modeled by adding an extra input.

\[
\sum_{j=0}^{m} w_jx_j = v
\]

\[
w_0 = b
\]
Neuron Models

• The choice of \( \varphi \) determines the neuron model. Examples:

  - step function:
    \[
    \varphi(v) = \begin{cases} 
      a & \text{if } v < c \\
      b & \text{if } v > c 
    \end{cases}
    \]

  - ramp function:
    \[
    \varphi(v) = \begin{cases} 
      a & \text{if } v < c \\
      b & \text{if } v > d \\
      a + ((v - c)(b - a)/(d - c)) & \text{otherwise}
    \end{cases}
    \]

  - sigmoid function:
    with \( z, x, y \) parameters
    \[
    \varphi(v) = z + \frac{1}{1 + \exp(-xv + y)}
    \]

  - Gaussian function:
    \[
    \varphi(v) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{1}{2} \left( \frac{v - \mu}{\sigma} \right)^2 \right)
    \]
Logistic function

Inputs

- Age: 34
- Gender: 1
- Stage: 4

Coefficients

- 0.5
- 0.4
- 0.8

Output

- "Probability of being alive": 0.6

Prediction

"Probability of being alive"
Neural Network Model

**Inputs**

- **Age**: 34
- **Gender**: 2
- **Stage**: 4

**Weights**

- From Age: 0.6
- From Gender: 0.1
- From Stage: 0.3

**Hidden Layer**

- Summation: 0.4
- Summation: 0.2

**Weights**

- From Hidden Layer: 0.5
- From Hidden Layer: 0.8

**Output**

- "Probability of beingAlive": 0.6

**Independent variables**

- Age
- Gender
- Stage

**Dependent variable**

- Prediction
• How to calculate the output $y$
• How to update or adjust weights $w$? Depend on type of training algorithm use

$$v = \sum_{j=1}^{m} w_j x_j$$

$$y = \varphi(v + b)$$
Learning/training Types

- Supervised learning – require input & desired output data
- Unsupervised learning – normally for pattern analysis – discover patterns or features, learning from observation, blur or corrupted images
- Reinforcement learning – the output can be correct or incorrect, do not know what the correct answer is. Example: chess game – reward right moves and punish wrong ones
Learning Rules/Algorithms

• Gradient or steepest Descent - Backprop
• Widrow-Hoff (Least Mean Square)
• Generalized Delta
• Error-Correction
• Scaled Conjugate gradient
• Levenberg-Marquardt
• Kohonen
• Hebbian
• Etc.
Developing Neural Networks

- Requires 3 phases:
  - **Training or learning phase** – update/adjust the weights using learning/training algorithm, time consuming and the critical phase
  - **Recall phase** – access outputs deviation from target response using the calculated weight factors, biases and training data
  - **Generalization phase** – subject the network to new input patterns and known output data, where the system hopefully performs properly
NN Training – supervised learning

• **Training** - process of setting/determining the **best weights** on the edges connecting all the units in the network

• The goal is to use the training set to calculate weights where the output of the network is as close to the desired output as possible for as many of the examples in the training set as possible
Neural Network Training

• Back propagation has been used since the 1980s to adjust the weights (other methods are now available):

Calculates the error by taking the difference between the calculated result and the actual result

The error is fed back through the network and the weights are adjusted to minimize the error
Practical aspect of neural computing

• Selecting number of hidden layers
• Normalizing input and output data sets
• Initializing the weights
• Setting learning rate and momentum coefficient – depend on training algorithm use
• Selecting proper transfer function
• Generating and using a network learning curve
Neural networks performance

• Accuracy – MSE, SSE, EI, overfitting
• Complexity of a model
• Convergence
Minimizing the Error

- Error surface
- Initial error
- Negative derivative
- Final error
- Local minimum
- \( w \) initial
- \( w \) trained
- Positive change
Gradient descent

Error

Global minimum

Local minimum
Overfitting

Real Distribution

Overfitted Model
Overfitting in Neural Nets

Overfitted model  "Real" model

Overfitted model

error

holdout

training

cycles
Performance factors

- Data Preparation
- Weight Initialization
- Learning rate and momentum
- Optimization method
- Architecture selection
- Activation functions
- Active Learning
Some References

Introductory Textbooks