O. V. Kuzik

NEURAL NETWORKS AND ARTIFICIAL INTELLIGENCE

National Aviation University

software engineering department

Lecturer: Glazunov M.M. (doctor of physical and mathematical sciences, professor)

Keywords: *Neural Networks, Artificial Neural Networks, Artificial Intelligence, Neuroscience, Cybernetics, Robotics, Expert Systems, Perceptron.*

Problem statement

Uncover the foundation of neuroscience and getting basic understanding about Neural Networks, Artificial Neural Networks, Artificial Intelligence and their application in different parts of science.

Related work

This paper describes the constructs one may use in creating or modifying modern neural network applications like music pattern recognition, web camera face and fingerprints recognition, implementation of expert systems and design of object-oriented software using the biological neural network principles.

Introduction

Artificial Neural Network (ANN) is a mathematical or computational model which is based on biological neural networks.

It consists of an interconnected group of artificial neurons and performs computations in a connective manner. Usually NN is an adaptive system that changes it`s structure from external disturbances which occur during the learning phase.

More practically, NNs are non-linear statistical datum modeling tools. We can use them for modeling complex relationships between inputs and outputs to recognize patterns in data.

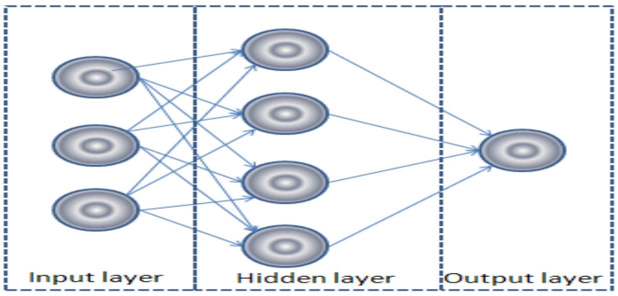

Figure 1. A simple representation of a Neural Network

On **Figure 1.** we may see a group of interconnected nodes which are similar to the neurons in our brain.

There in no completely agreed definition of what a Neural Network is but it is a group of simple processing elements called **neurons** which exhibit complex global behavior depending on the connections between elements and their parameters. Originally the NN was discovered under research of human Central Nervous System and neurons (in human organism they have **synapses**, **dendrites** and **axons**) which are the most significant information processing elements.

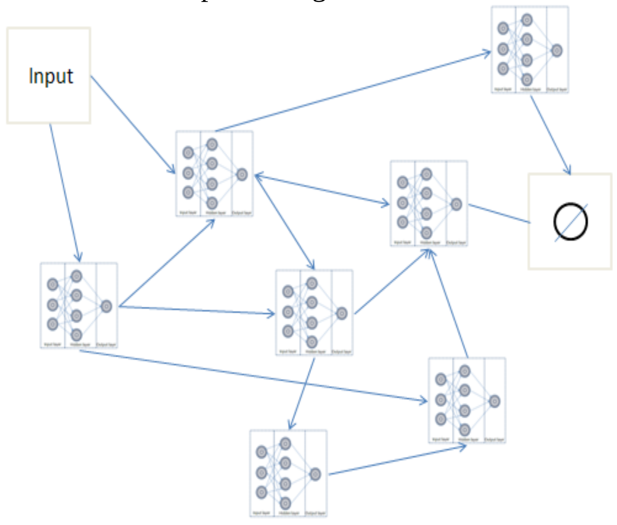

Figure 2. Component-based representation of a Neural Network

We use this kind of representation (**Figure 2.**) in some software products for processing Neural Networks. In this model we see that the simple elements form a larger, more complex network structure. If we don`t need an adaptive Neural Network, these structures` practical use is application of algorithms for altering the weights or **strength of connections** which leads to the desired output signal flow.

Every element may represent a small neural network of a different type:

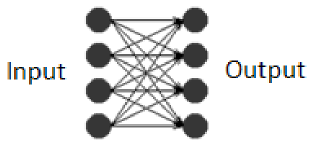

Figure 3. Single Layer Feedforward

The Single-Layer-FeedForward Neural network **(Figure 3.)** doesn`t have a hidden layer. And the connections between units never form a directed cycle.

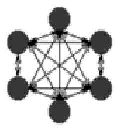
Figure 4. Fully Recurrent network

The connections in a Fully Recurrent Neural Network **(Figure 4.)** form a directed cycle. This allows the network to have an internal global state which allows it to have a dynamic behavior.

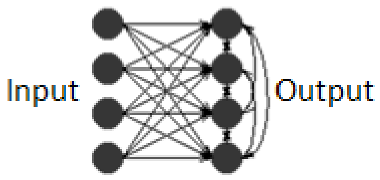

Figure 5. Competitive Network

The Competitive Feedforward Neural network **(Figure 5.)** doesn`t (in some cases does) have a hidden layer. The connections between units never form a directed cycle, and the outputs interact with each other.

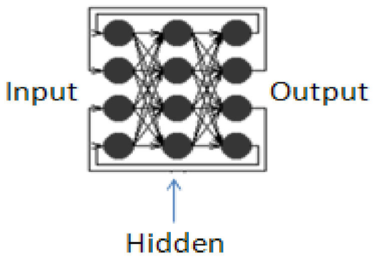

Figure 6. Jordan`s Network

Jordan`s Neural Network **(Figure 6.)** is the Recurrent Neural Network that has directed cycles which realize complete feedback on the signal`s way to output.

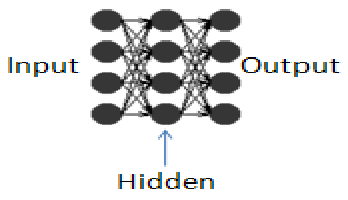

Figure 7. Multi-Layer Feedforward

In the Multi-Layer-Feedforward Neural network **(Figure 7.)** the connections between units never form a directed cycle. It has a hidden layer also referred to as **Perceptron.**

As we see, these networks are also alike the biological Neural Network, thus the functions are performed in parallel / in series / collectively. They`re not just a set of subtasks with assigned units (like OOP programming concept).

Nowadays we refer to Artificial Neural Network mostly as models used in **statistics**, **artificial intelligence** and such study as **cognitive psychology**. The study of computational neuroscience is based on modeling the human brain and central nervous system using the artificial neurons.

Supervised learning

When we implement the supervised learning, as an initial condition we`re given a set of pairs: $(x,y), x \in X, y \in Y$ and or aim is to find a the function F in the allowed class of functions that matches the pairs. So we want to infer the mapping implied by the datum. Our cost function is related to the mismatch between our datum and mapping, it implicitly contains knowledge about the problem domain.

In mathematics we commonly use a cost where the mean-square error shows the mismatch between the output of our network F(x) and the target value y over our pairs. We get a well-known back-propagation algorithm when we try to minimize the cost using gradient descent in the process of training our neural network.

Such tasks as pattern recognition (classification) and regression (approximation of functions) fall under this concept. For speech, gestures and bitmap recognition we also use the **supervised learning paradigm**.

This process may be described as learning by a teacher in the form of a function with endless feedback based on the quality of obtained solutions.

Unsupervised learning

In unsupervised learning we have some datum X and a cost function to be optimized may be any function of input X and F as the output of our network.

This cost function is partially dependent on the task we are modeling and our basic assumptions about the properties, variables and parameters of our model.

The trivial example from math is the model **F(X) = A** where A is a constant and our cost equals **C = E[(X – F(x)^2)]**. By optimizing this cost we get a value which is the mean of our input data. This function may be may more complicated! And it`s form and representation depends on the field of science where we apply our Neural Network. As an example in a compression or cipher algorithm it will describe the relation between mutual input X and output Y which is a description of the compression / cipher algorithm.

Tasks that fall under the **unsupervised learning paradigm** are usually **estimation problems**, **clustering applications**, some **statistical distributions**, **compression**, **filtering**, **encryption** and **coding**.

Leaning algorithms

Training the created Neural Network usually all goes down to **selection of a model from the set of allowed ones** (determination of distribution among the sets of allowed models) that minimizes the cost criteria.

We may find plenty of training algorithms for various neural networks. Most of them are **straightforward (Figures 3, 5, 6, 7)** but we as well may use the **recurrent algorithms** for networks of form like on **Figure 4.**

The most common algorithms use the feedback method implying the **gradient descent**. We do this by **derivating our cost function** with respect to the neural network`s parameters and then changing them in a gradient-related fashion.

Applications of Artificial Intelligence Today

Space modeling and heavy industry

Expert systems applied to robotics and cybernetics have created a leap in their development. The manufacturing processes are completely automated and controlled by the computers. This is used in all fields of manufacturing: car industry, tools design, silicon chip production, food and vegetables planting e t c. neural network-controlled devices can handle hazardous materials for people as well as perform some dangerous works. The control of aircraft is also to some extent controlled by such machinery which helps it perform unique maneuvers.

Weather Forecasting:

In weather forecasting laboratories and stations we use neural networks to model the behavior of atmospheric activity and build weather patterns for different regions.

Finance application:

We all know about trading markets where our profit depends on the ability to predict market prices. Neural networks help us by creating expert systems which can do prediction better then human`s can do.

Computer Science:

The neural network concept has led software engineers towards revolutionary development techniques like object-oriented programming, intelligent storage management systems, social networks management. Nowadays the computer is the main tool for creating and designing artificial intelligence.

Swarm Intelligence

Swarm intelligence is a very interesting field where programmers observe the behavior of some animal or an insect specimen and design the neural model of it`s behavior and properties. This model is further used in creating stunning 3D graphics or computer games.

Expert Systems

The very interesting application of Neural Networks are expert systems. For example an engineer must consult an expert before starting the performance of task. An expert system may hold the references to nearly all types of data concerning nearly any field of it`s application. Of course it can`t be done without a recurrent Neural Network **(Figure 4.)** which will form a recurrent encyclopedia.

Conclusions:

Knowledge of the basic principles of the neural networks is vital for moving the science forward in any field. The basic constructs and principles allow human to create devices, robotics and programs which simplify our daily life and realize new possibilities for us. Interesting note: all artificial intelligences are just the result of the activity of a neural network, the one which is human brain. When we have a team of people working for the same goal, this is the example of the component representation of the Figure 2. A biological Neural Network creating an Artificial one. Some day the hybrid neural network will be developed which will be an enormous leap for all humanity. We should work hard to push neuroscience forward as this is the most effective branch of our development.

References:

Rosenblatt F. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms", Spartan Books, 2004.

David E. Rumelhart; Geoffrey E. Hinton; Ronald J. Williams. Learning Internal Representations by Error Propagation.

A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Back-propagation: Theory, Architectures and Applications. Hillsdale, NJ: Erlbaum, 1994.

J. Schmidhuber. A local learning algorithm for dynamic feedforward and recurrent networks. Connection Science, 1(4):403–412, 1989.

Neural and Adaptive Systems: Fundamentals through Simulation. J.C. Principe, N.R. Euliano, W.C. Lefebvre

J. Schmidhuber. A fixed size storage O(n3) time complexity learning algorithm for fully recurrent continually running networks. Neural Computation, 4(2):243–248, 1992.

R. J. Williams. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science, 1989.

B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. Neural Computation, 1(2):263–269, 1989.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich, 1991.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735– 1780, 1997.

Neural Networks as Cybernetic Systems 2nd and revised edition, Holk Cruse

H. Jaeger. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science, 304:78–80, 2004.

W. Maass, T. Natschläger, and H. Markram. A fresh look at real-time computation in generic recurrent neural circuits. Technical report, Institute for Theoretical Computer Science, TU Graz, 2002