Lecture 3: Artificial Neural Networks

Wei Qi Yan

Auckland University of Technology

March 15, 2023

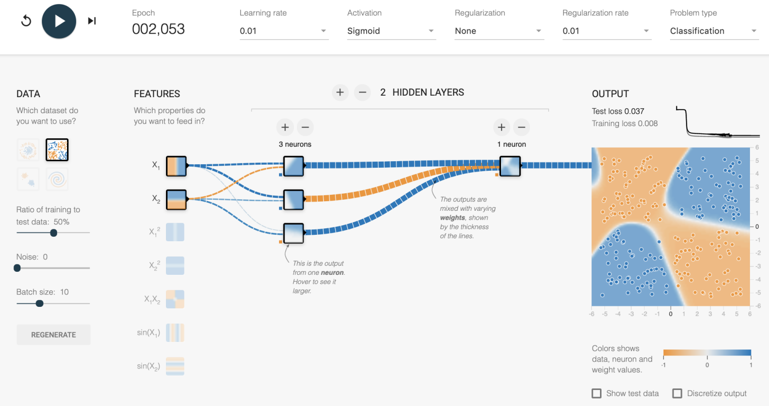
Table of Contents

Overview of ANNs

- ANNs are composed of simple elements operating in parallel.
- ANNs are adjusted, or trained, so that a particular input leads to a specific target output.
- ANNs have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems.
- ANNs can also be trained to solve problems that are difficult for computers or human beings.

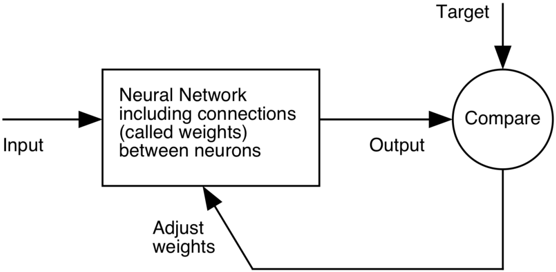
Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Demo: An ANN Playground



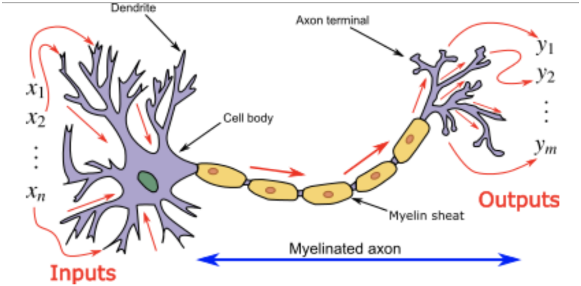
I. Goodfellow, Y. Bengio and A. Courville (2016) Deep Learning. The MIT Press.

Overview of ANNs



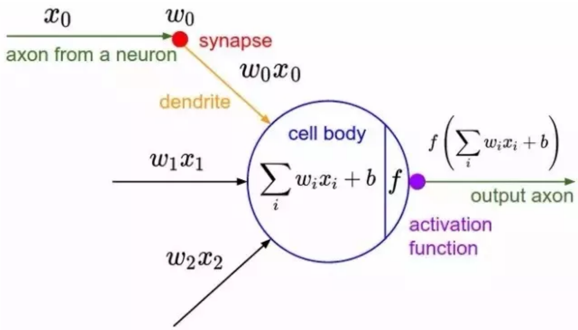
Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

A Neuron



Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

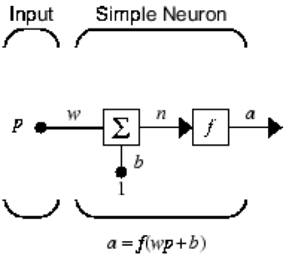
A Neuron of ANNs



N. Kasabov (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press.

Three Functional Operations

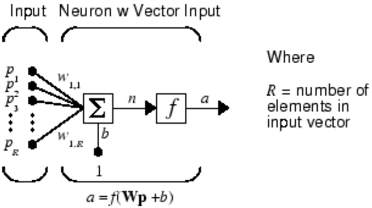
- Scalar weight function: Form the product wp
- Scalar input function: Form the net input $n = wp + b$
- Scalar transfer function: Produce the scalar output $a = f(wp + b)$



Note: The central idea of neural networks is that parameters w and b can be adjusted so that the network exhibits desired or interesting behaviors.

Three Functional Operations

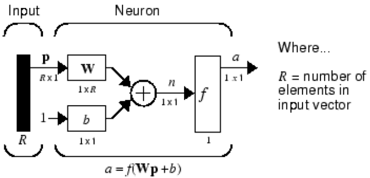
- Vector weight function: Form the product $\mathbf{W} \cdot \mathbf{p}$
- Vector input function: Form the net input $\mathbf{n} = \mathbf{W} \cdot \mathbf{p} + b$
- Vector transfer function: Produce the scalar output $a = f(\mathbf{W} \cdot \mathbf{p} + b)$



Note: The central idea of neural networks is that parameters $\mathbf{w}$ and b can be adjusted so that the network exhibits desired or interesting behaviors.

Three Functional Operations

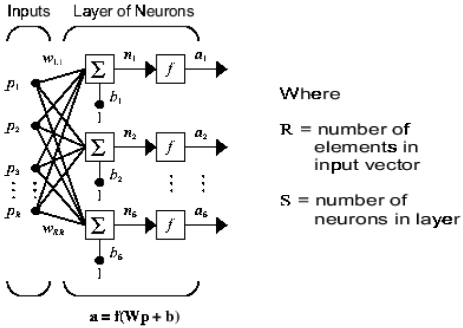
- General weight function: Form the product $\mathbf{W} \cdot \mathbf{p}$
- General input function: Form the net input $\mathbf{n} = \mathbf{W} \cdot \mathbf{p} + b$
- General transfer function: Produce the scalar output $a = f(\mathbf{W} \cdot \mathbf{p} + b)$



Note: The central idea of neural networks is that parameters $\mathbf{w}$ and b can be adjusted so that the network exhibits desired or interesting behaviors.

One Layer of Neurons

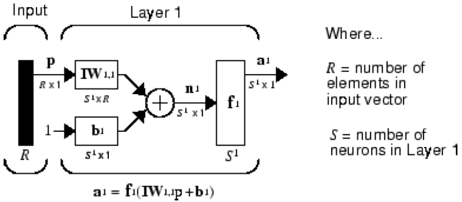
- Vector weight function: Form the product $\mathbf{W} \cdot \mathbf{p}$
- Vector input function: Form the net input $\mathbf{n} = \mathbf{W} \cdot \mathbf{p} + \mathbf{b}$
- Vector transfer function: Produce the vector output $\mathbf{a} = \mathbf{f}(\mathbf{W} \cdot \mathbf{p} + \mathbf{b})$



Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

One Layer of Neurons

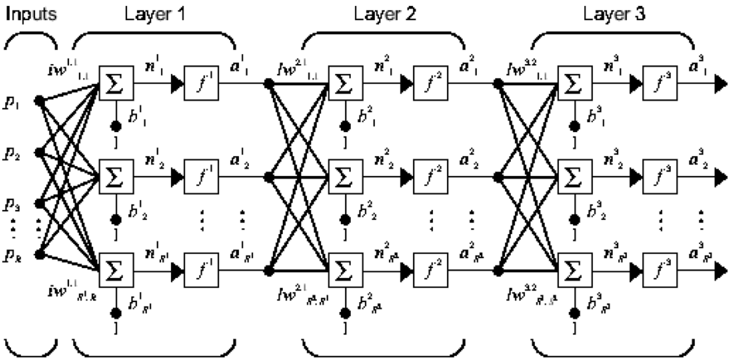
- General weight function: Form the product $\mathbf{IW}_{1,1} \cdot \mathbf{p}$
- General input function: Form the net input $\mathbf{n}_1 = \mathbf{IW}_{1,1} \cdot \mathbf{p} + \mathbf{b}_1$
- General transfer function: Produce the vector output $\mathbf{a}_1 = \mathbf{f}_1(\mathbf{IW}_{1,1}\mathbf{p} + \mathbf{b}_1)$



Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Artificial Neural Networks

Multiple Layers of Neurons

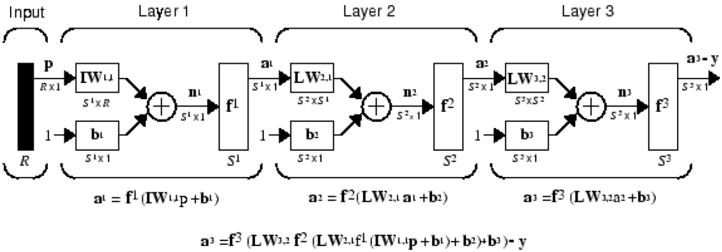


$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

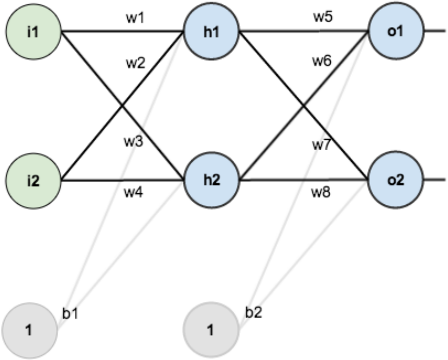
Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Multiple Layers of Neurons



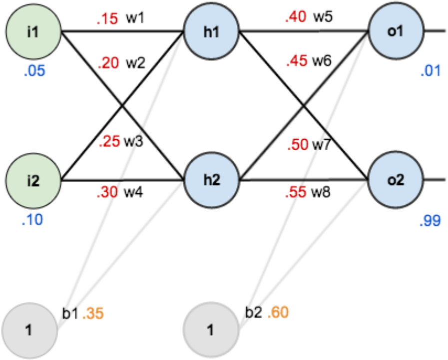
Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Multiple Layers of Neurons: An Example



Web: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Multiple Layers of Neurons: An Example



Web: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Questions?

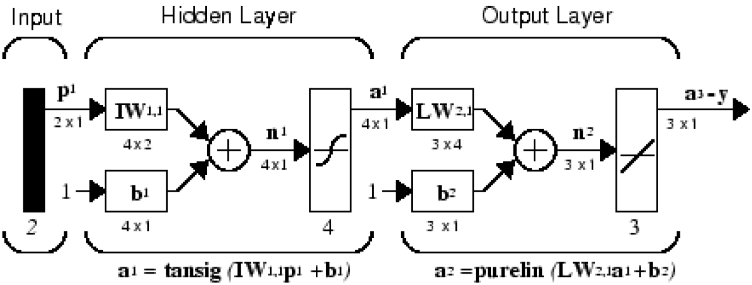
Multilayer Perceptron

Feedforward Neural Networks

- A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN).
- An MLP consists of at least three layers of nodes: An input layer, a hidden layer, and an output layer.
- Except for the input nodes, each node is a neuron that uses a nonlinear activation function.
- MLP utilizes a supervised learning technique called backpropagation for training.
- MLP can distinguish or discriminate data that is not linearly separable.

Web: https://en.wikipedia.org/wiki/Multilayer_perceptron

Feedforward Neural Networks



Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Backpropagation Algorithm

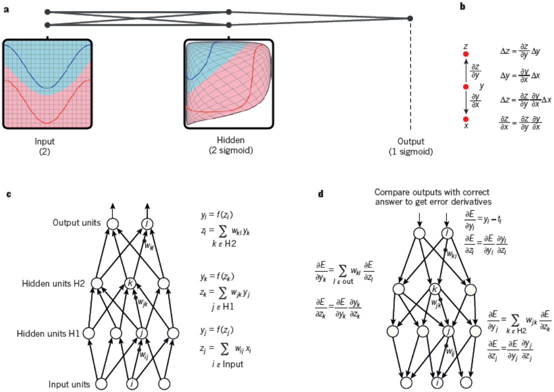
- The backpropagation algorithm involves performing computations backward through the network.
- Gradient descent updates the network weights and biases in the direction in which the performance function decreases most rapidly.
- "Backpropagation" refers specifically to the gradient descent algorithm, when applied to neural network training.
- In a backpropagation algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \cdot \mathbf{g}_k, k = 1, 2, \cdots$$

where $\mathbf{x}_k$ is a vector of current weights, $\mathbf{g}_k$ is the current gradient, and α is learning rate.

Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Backpropagation Algorithm



Y. LeCun, Y. Bengio and G. Hinton, Deep learning, Nature, volume 521, pages 436–444(2015)

Feedforward Neural Networks

- Multilayer networks are capable of performing any linear or nonlinear computations, they can approximate any reasonable function arbitrarily well.
- Networks are also sensitive to the number of neurons in their hidden layers.
- Too few neurons can lead to underfitting.
- Too many neurons can contribute to overfitting, in which all training points are well fitted, but the fitting curve oscillates wildly between these points.

Web: https://au.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf

Multilayer Perceptron

Horner Algorithm: $f(x) = \sum_{i=0}^{n} a_i \cdot x^i$ is written as:
$f(x) = (((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \cdot x) + \cdots + a_0,\ a_n \neq 0,$
$a_n, x \in \mathbb{R},\ i, n \in \mathbb{Z}^+$.

Kolmogorov Theorem

A MLP has the ability to represent any continuous function $g(\mathbf{x}), \mathbf{x} = (x_1, x_2, \cdots, x_d) \in [0,1]^d = \underbrace{[0,1] \times \cdots \times [0,1]}_{d}$, $d \geq 2$ for properly chosen functions $\xi_j(\cdot)$ and $\psi_{ij}(\cdot)$,

$$f(\mathbf{x}) = \sum_{j=1}^{2n+1} \xi_j\Big(\sum_{i=0}^{d} \psi_{ij}(x_i)\Big).$$

R. Duda, P. Hart, D. Stork (2000) Pattern Classification (2nd Edition), Wiley-Interscience.

Multilayer Perceptron

Stochastic Gradient Descent

- Multilayer architectures can be trained by simple stochastic gradient descent.
- Gradients can be computed by using the backpropagation procedure.
- The backpropagation procedure to compute the gradient of an objective function *w.r.t.* the weights of a multilayer stack of modules are as same as *chain rule* for derivatives.
- The backpropagation can be applied repeatedly to propagate gradients through all modules.

Y. LeCun, Y. Bengio, G. Hinton (2015) Deep learning, Nature, 521, pages 436 - 444.

Stochastic Gradient Descent

In practice, the procedure of Stochastic Gradient Descent (SGD) consists of:

- Input vector for a few of samples
- Outputs and errors
- Average gradient for those samples
- Adjustable weights

The process is repeated until the average of the objective function stops decreasing.

I. Goodfellow, Y. Bengio and A. Courville (2016) Deep Learning, MIT Press.

Loss Functions

- 0-1 loss function: $L(Y, f(X)) = \begin{cases} 1 & Y \neq f(X) \\ 0 & Y = f(X) \end{cases} \quad X, Y \in \mathbb{R}$
- Square loss function: $L(Y, f(X)) = (Y - f(X))^2, X, Y \in \mathbb{R},$
- Absolute loss function: $L(Y, f(X)) = |Y - f(X)|, X, Y \in \mathbb{R}$
- Logrithm loss function: $L(Y, p(Y|X)) = -\log p(Y|X), X, Y \in \mathbb{R}$
- Average loss function: $L = \frac{1}{m} \sum_{i=1}^{m} L(x_i, y_i)$, where the set $T = \{(x_i, y_i)\}(i = 1, 2, \cdots, m), x_i, y_i \in \mathbb{R}$ is the training set.
- $\cdots\cdots$

I. Goodfellow, Y. Bengio and A. Courville (2016) Deep Learning, MIT Press.

Activation/Transfer Functions

- ReLU: Rectified linear unit,

$$f(x) = \max(0, x), x \in \mathbb{R}.$$

- Tanh: Hyperbolic tangent function,

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, x \in \mathbb{R}.$$

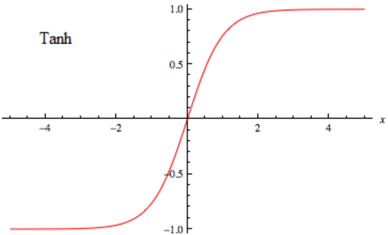
- Sigmoid: Logistic function,

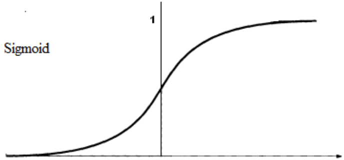
$$f(x) = \frac{1}{1 + e^{-x}}, x \in \mathbb{R}.$$

- ......

N. Kasabov (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press.

Activation Functions





N. Kasabov (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press.

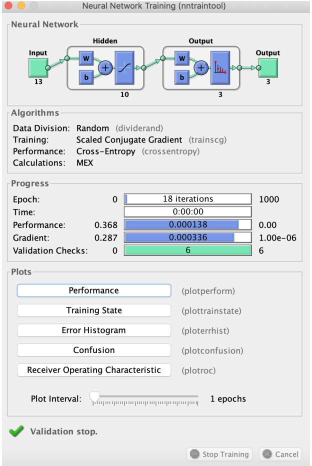
Questions?

Training, Test, and Validation Datasets

Three Datsets

- Training dataset: A set of examples are used to fit the parameters of the model. The dataset often consists of pairs of an input vector and the corresponding output vector.
- Test dataset: A set of samples are used only to assess the performance (i.e., generalization) of a fully specified classifier.
- Validation dataset: A dataset of samples are used to tune the hyperparameters (i.e., the architecture) of a classifier.
- Cross validation: A dataset can be repeatedly split into a training dataset and a validation dataset.

Wine Classification: Neural Network



- **TP**: True positive (hit)
- **TN**: True negative (correct rejection)
- **FP**: False positive (false alarm)
- **FN**: False negative (miss)
- **TPR**: True positive rate $TPR = \frac{TP}{TP+FN}$
- **FPR**: False positive rate $FPR = \frac{FP}{FP+TN}$
- **PPV**: Positive predictive $PPV = \frac{TP}{TP+FP}$
- **ACC**: Accuracy $ACC = \frac{TP+TN}{P+N}$
- **F1 Score**: $F1 = \frac{2TP}{2TP+FP+FN}$

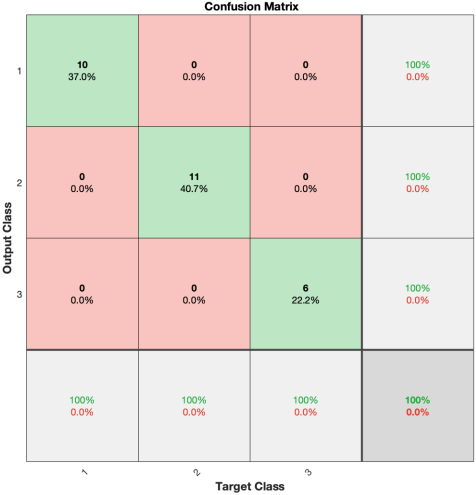
Web: https://au.mathworks.com/help/deeplearning/examples/wine-classification.html

Wine Classification: Performance

Web: https://au.mathworks.com/help/deeplearning/examples/wine-classification.html

Training, Test, and Validation Datasets

Wine Classification: Confusion Matrix



Training, Test, and Validation Datasets

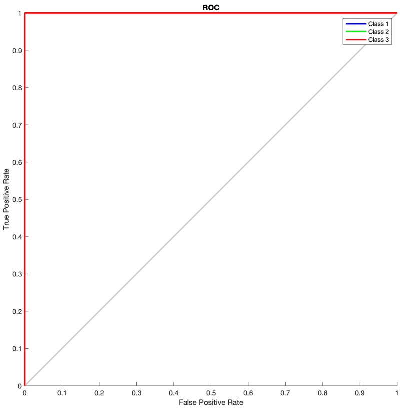
Confusion Matrix: An Example

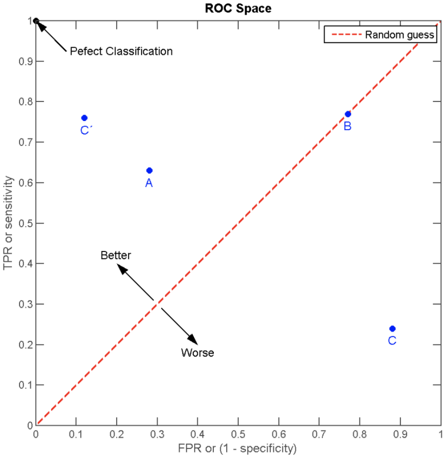
A			B			C			C′		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=76	FP=12	88
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=24	TN=88	112
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
PPV = 0.69			PPV = 0.50			PPV = 0.21			PPV = 0.86		
F1 = 0.66			F1 = 0.61			F1 = 0.23			F1 = 0.81		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

Web: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Training, Test, and Validation Datasets

Wine Classification: ROC Curve





Web: https://au.mathworks.com/help/deeplearning/examples/wine-classification.html

Questions?

Learning Objectives

- To critically compare and evaluate the major components of neural networks.