Artificial Neural Networks

Guide to AI and Data Courses · Part I: Core

Artificial neuron | Perceptron | Multilayer networks | Backpropagation | Training and generalization | Neighboring courses

Sercan KÜLCÜ · Giresun University · Computer Engineering

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides: https://sercankulcu.github.io/files/ai_courses/slides/Chapter_04_Neural_Networks.html

INTRODUCTION

What Question Does It Ask?

> **How do networks of simple units learn by adjusting their weights?**

Artificial neural networks are models built by connecting, in layers, simple units that take a weighted sum and pass it through an activation function. Learning means **adjusting the weights** so as to reduce a loss function; the main tools for this are backpropagation and gradient descent.

At a glance

APPROACH	INPUT	OUTPUT
network of simple weighted units	numeric feature vector	class or numeric prediction

SUCCESS MEASURE	MATH LOAD	DATA NEEDS
loss and accuracy	medium-high: derivatives, matrices	moderate

A Short History

1943
McCulloch and Pitts propose the first artificial neuron model with threshold logic units.

1949
Hebb suggests that connections between neurons that fire together grow stronger: the Hebbian rule.

1957
Rosenblatt introduces the perceptron; its weights are tuned from examples.

1969
Minsky and Papert show that a single-layer perceptron cannot solve problems such as XOR.

1982
Hopfield publishes a recurrent network that recalls memories by minimizing energy.

1986
Rumelhart, Hinton and Williams popularize backpropagation; multilayer networks become trainable.

THE COURSE

Core Concepts

Artificial neuron
Weighted sum of inputs, a bias and an output.

Activation
Sigmoid, tanh, ReLU: adds nonlinearity to the network.

Perceptron
Single-layer network that separates linearly separable classes.

Multilayer network
Hidden layers learn nonlinear boundaries.

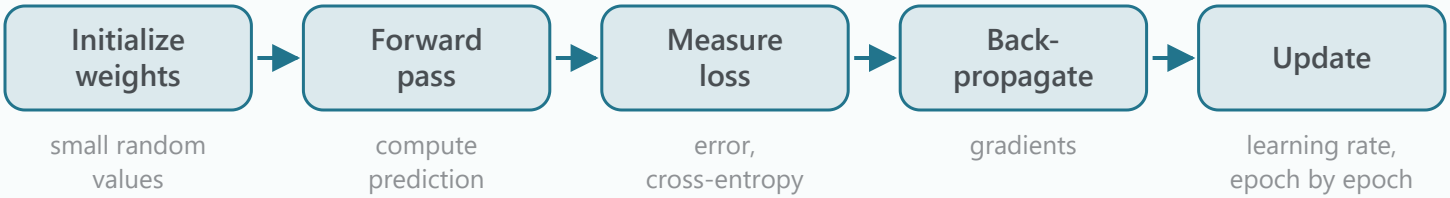
Forward pass
Computing the prediction layer by layer from input to output.

Backpropagation
Finding the gradient of every weight with the chain rule.

Gradient descent
Moves the weights in small steps where the loss decreases.

Regularization
Weight penalties and early stopping prevent overfitting.

A Typical Workflow



Example: A small network that reads handwritten digits takes a 28×28-pixel image as 784 inputs, passes it through a hidden layer and outputs probabilities for 10 classes. A wrong guess means a large loss; backpropagation computes each weight's share and gradient descent corrects the weights a little.

THE COURSE

What Does the Course Cover?

1. Biological and artificial neurons; McCulloch-Pitts
2. Hebbian learning and the perceptron algorithm
3. Adaline, least mean squares and the delta rule
4. Linear separability and the XOR problem
5. Multilayer networks and activation functions
6. Backpropagation: computational graphs, chain rule
7. Loss functions: squared error, cross-entropy
8. Gradient descent, learning rate and momentum
9. Weight initialization and vanishing gradients
10. Generalization: regularization, early stopping
11. Radial basis function networks
12. Hopfield networks and self-organizing maps (SOM)

Skills you will gain

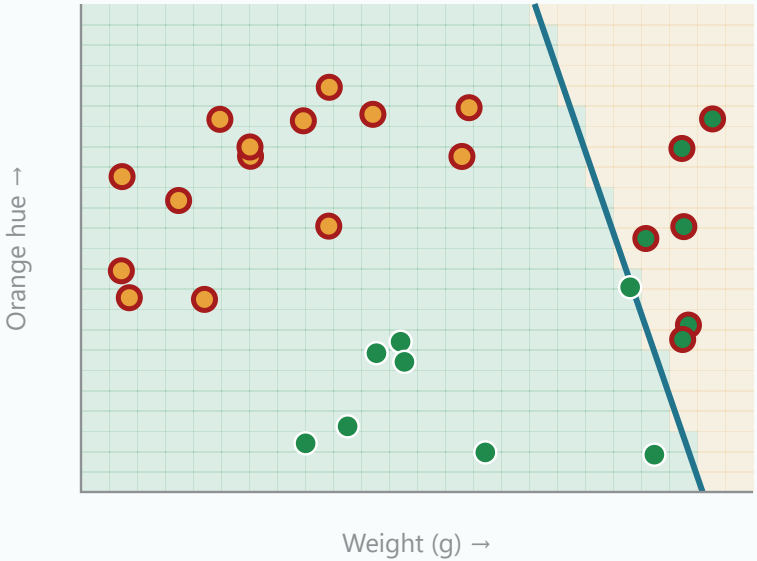
writing a network from scratch in NumPy | deriving backpropagation by hand | numerical gradient checking | training networks in PyTorch and Keras | reading learning curves

A typical semester; the order and coverage vary between universities.

TRY IT

Tangerine or Lemon?

A camera on the sorting belt measures the weight and color hue of each fruit. A single artificial neuron (a perceptron) learns to separate the two kinds: train it, or click a fruit to flip its label.



Train one step | One epoch | 10 epochs | Reset

output = sign(**+0.40**·weight **+0.10**·hue **−0.37**)

Epoch 0, next example 1/28.
Misclassified: **20** fruit

● tangerine ● lemon. Relabel a fruit deep among the others: no line can separate them, and the error **never reaches zero**.

Nobody wrote the rule: after each mistake the neuron nudged its weights toward the error. A single neuron can only draw a **linear** boundary; data that cannot be separated that way needs a multilayer network.

CONTEXT

Prerequisites

Courses worth knowing before this one, and why they are needed.

Course	Why it is needed
Machine Learning	parametric model, loss function, gradient descent
Linear Algebra	weight matrices, vectorized computation
Optimization	gradient descent, learning rate
Calculus	derivatives, partial derivatives and the chain rule: the math of backpropagation
Programming	arrays, loops and functions in Python

CONTEXT

Are You Ready for This Course?

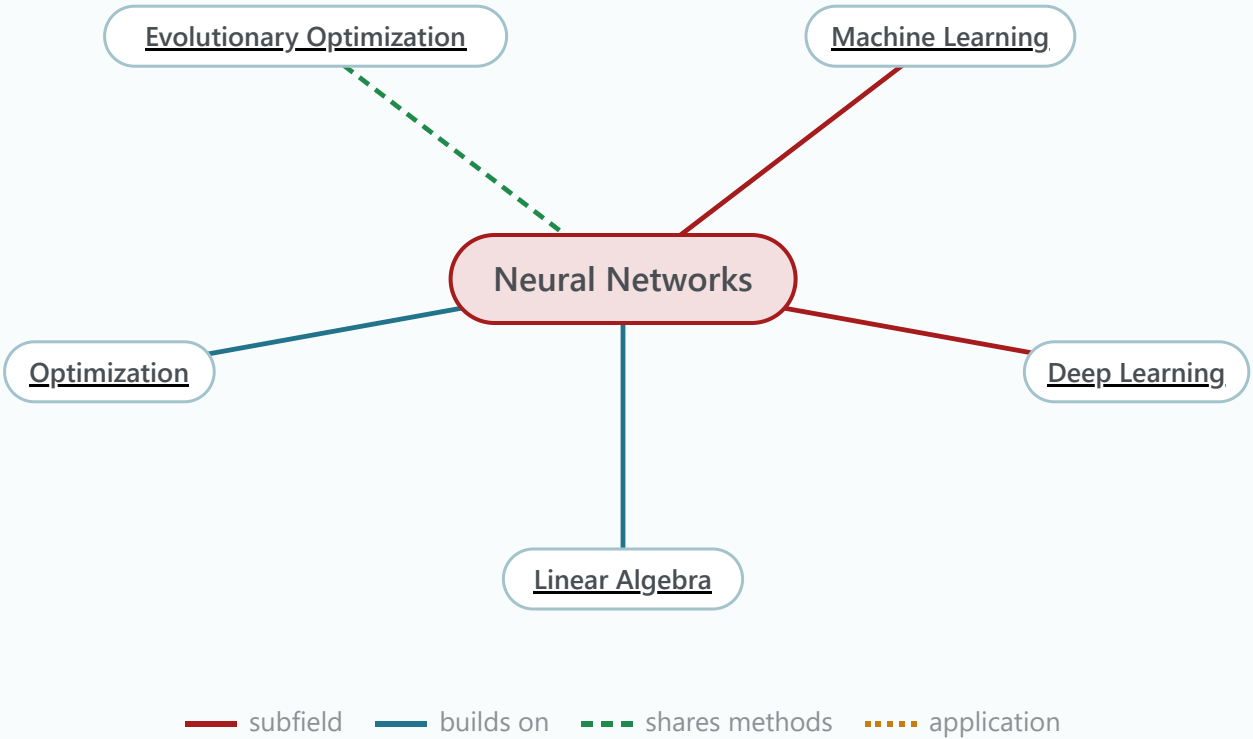
Ask yourself; for every "no", the right-hand column shows where to start.

Ask yourself	If not, start with
Can you differentiate a composite function with the chain rule?	Derivatives and the chain rule from calculus
Can you multiply a matrix by a vector and state the result's shape?	Linear Algebra
Can you do vectorized operations on NumPy arrays without loops?	An intro programming course, then a short NumPy tutorial
Can you explain the difference between training and test error?	Machine Learning

If you can say "yes" to most of these, you are ready; one or two gaps can be closed during the semester.

CONTEXT

Neighbouring Courses



Overlapping Courses

Relation	Course	Shared ground
subfield	This course is a subfield of Machine Learning	parametric model, loss function, gradient descent
subfield	Deep Learning is a subfield of this course	multilayer networks, backpropagation, activation functions
builds on	This course builds on Linear Algebra	weight matrices, vectorized computation
builds on	This course builds on Optimization	gradient descent, learning rate
shares methods	Shares methods with Evolutionary and Metaheuristic Optimization	neuroevolution, hyperparameter search

DIFFERENCES

Where It Differs (1/2)

Criterion	Neural Networks	Machine Learning	Deep Learning
Core question	How do networks of simple units learn by adjusting their weights?	How does a program generalize from examples?	How do many-layered networks learn useful representations from raw data?
Approach	network of simple weighted units	statistical learning	multilayer representation learning
Input	numeric feature vector	labeled or unlabeled examples	raw data: images, text, audio
Output	class or numeric prediction	a model that makes predictions	learned representation and prediction
Success measure	loss and accuracy	error on unseen data	accuracy, loss, scaling
Math load	medium-high: derivatives, matrices	high: linear algebra, probability, optimization	high: derivatives, matrices, probability
Data needs	moderate	moderate to large	large; needs GPUs

- **Machine Learning:** ML covers many model families from decision trees to SVMs; this course focuses only on networks built from neurons.
- **Deep Learning:** Deep learning moves on to convolutional, recurrent and transformer architectures; this course builds the basic network and training they rest on.

DIFFERENCES

Where It Differs (2/2)

Criterion	Neural Networks	Optimization	Evolutionary Optimization
Core question	How do networks of simple units learn by adjusting their weights?	How do we find the values that make an objective function best?	How do we find a good solution when there is no gradient or the search space is huge?
Approach	network of simple weighted units	mathematical model of objective and constraints	population-based heuristic search
Input	numeric feature vector	objective function and constraints	objective, solution encoding
Output	class or numeric prediction	optimal or near-optimal solution	good, not proven optimal, solution
Success measure	loss and accuracy	objective value, convergence rate	quality, evaluation count
Math load	medium-high: derivatives, matrices	the theory itself	low-medium
Data needs	moderate	none	none; evaluations suffice

- **Optimization:** Optimization studies gradient methods in general; this course uses them to learn weights.
- **Evolutionary Optimization:** Neuroevolution searches weights or architectures by evolution; this course learns weights by gradients and backpropagation.

PRACTICE

Applications

Digit recognition
Handwritten digits on postcodes and checks.

Tabular data
Classification tasks such as credit risk or churn.

Function approximation
Learning nonlinear relations from measurements.

Control
Simple robot and process controllers.

Signal classification
Detecting faults or motion from sensor data.

Associative memory
Completing a corrupted pattern with a Hopfield network.

PRACTICE

Risks and Ethics

Memorization and overfitting

A large network on a small dataset drives training error to zero and then fails on new examples; without a validation set you never see it.

Shortcut learning

A network may learn traces of data collection instead of the real feature: a ruler in the photo, the background or a scanner tag.

Unstable training

Poor initialization or a large learning rate makes the loss blow up; a result from a single random seed may not reproduce.

Decisions without reasons

Weights do not read as rules; that is a serious problem where a justification is required, as in credit or medical decisions.

In depth: Explainable AI and AI Ethics.

PRACTICE

Tools and Datasets

Tools and libraries

NumPy | PyTorch | Keras | scikit-learn (MLPClassifier) | TensorFlow Playground | MATLAB Deep Learning Toolbox

Well-known datasets

XOR problem | Iris | MNIST | Fashion-MNIST | Two spirals | CIFAR-10

In this course, writing a network **by hand in NumPy** first is the best way to see what backpropagation really does.

PRACTICE

Common Misconceptions

Common belief	In fact
Neural networks imitate the brain.	Biology is the inspiration, but the artificial neuron is a very crude model and its learning rule differs from the brain's.
More neurons are always better.	A large network can memorize the training data; size is chosen with a validation set.
Backpropagation is a separate learning method.	Backpropagation only computes the gradient; gradient descent updates the weights.
Training gives the same result every time.	Random initialization and shuffling can lead to different local minima.

NEXT STEPS

After This Course

Courses that build on or extend this one

Deep Learning

Related careers

Machine learning engineer | Deep learning engineer | Researcher | Embedded AI developer

This course is the gateway to deep learning: convolutional networks, recurrent networks and transformers are built on the ideas of neuron, loss and backpropagation introduced here. It also reinforces optimization and linear algebra.

NEXT STEPS

Further Reading

- Haykin, *Neural Networks and Learning Machines*, 3rd ed., Pearson, 2009.
- Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
- Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015 (online book).
- Rumelhart, Hinton and Williams, "Learning representations by back-propagating errors", *Nature* 323, 1986.

The End

Chapter 4: Artificial Neural Networks

← Previous chapter · Map of the field · Next chapter →