Coen's AI Notes and Links

diverse

2025-04-10

Table of contents

Chapter 1

Introduction

Welcome! This is a Work-in-Progress, a collection of notes on AI I am collecting and which I use in my workshops about AI and GenAI. The newest version of this pdf can be downloaded from
here

It is not complete nor self-describing, but when you attended one of my workshops you will probably find familiar stuff in one or more chapters.

Our world is changing rapidly through AI and GenAI. One can ignore it or decide to not use it, but that does not stop it… One can also decide to dive in and help 'invent' the future, or at least learn about all the new stuff.

These notes started out as visualizations of Perceptrons and Neural Networks in the Glamorous Toolkit, which helped me give students insights in Neural Networks.

I advise to get hands-on with the tools around.

When using online AI tools, please keep the privacy in mind when using personal data! One way to make sure private data will stay private is using local AI's.

Please also keep the Societal impact in mind! We can use AI to help us all, but there is of course also a dark side:
People getting fired, it's easier to create fake news, a few people getting rich at the expense of others,
some nice activities (I like programming for example) will never be the same.
Please use it wisely…

Figure 1.1: art and laundry

Chapter 2

AI Overview

2.1 AI, Machine Learning, Deep Learning, Generative AI

The video "AI, Machine Learning, Deep Learning and Generative AI Explained" provides an excellent 10-minute overview:

AI, Machine Learning, Deep Learning and Generative AI Explained

Part I

GenAI

Chapter 3

GenAI

Recent:

- Amy Webb SxSW 2025 - Emerging Tech Trend

3.1 What is GenAI?

- Why not ask perplexity.ai ?

3.2 GPT - Generative Pre-Trained Transformer

- Generative AI & the Transformer (Financial Times, interactive site)
- History of ChatGPT (30 min)
- But what is a GPT? (3Blue1Brown, 30 min)

3.3 Prompting

- Prompting basics
- Look for course with 'Prompting' in name: https://www.deeplearning.ai/short-courses/

3.4 Hallucinating

- 'Which day do I have to put the garbage can out on the street?'
- 'Can you help me find my lost keys?'
- 'Can you create an image of a watch that says it is 3 o' clock?'

3.5 RAG - Retrieval Augmented Generation

- IBM, Marina Danilevsky (7 min)
- https://www.deeplearning.ai/short-courses: Great resource for courses!

3.6 Active Inference

- Andy Clark about Active Interference: How the Brains shapes reality (60 min)

3.7 Running LLM's locally

On your laptop/desktop or on a company server:

- ollama
- LM-studio
- Open Web AI

3.8 Coding with GenAI

- vs code with co-pilot (free plan)
- Cursor.com (20 euro p/m)
- AIDER.chat (free)
- Open Devin: Create any Application with Open Source AI Engineer
- Avante (AI in neovim, free)

3.9 GenAI & education

- How AI is changing education: https://www.youtube.com/watch?v=OaEk-ZYzh80

3.10 GenAI

- awesome GenAI guide
- huggingface

3.11 Some more sites, nice to play around with

- https://skyreels.ai/
- https://civitai.com/

Chapter 4

Resources and References GenAI

4.1 Blogs and articles

Perplexity is often a great start for finding things (with references): perplexity.ai

- Jessy: Het belang van duidelijke AI-prompts
- Journalists on Hugging Face
- How polite should we be when prompting LLMs?
- Information literacy and chatbots as search

To understand about Transformers this is a very nice start: https://ig.ft.com/generative-ai/ 'Our own' page about (Gen)AI: https://stasemsoft.github.io/FontysICT-sem1/docs/artificial-intelligence/ai.html To dive further into how Transformers works: https://www.deeplearning.ai/short-courses/how-transformer-llms-work/ and also to other short courses on deeplearning.ai The development I showed was https://www.cursor.com/ you have like only 500 requests for free… after that you could choose to pay 20 euro a Month (yes, that can be a lot for students, I know), or look for alternatives, 2 of which I tried a bit (you can use local LLM's with them, which basically makes them free): AIDER: https://aider.chat/.
Avante: https://github.com/yetone/avante.nvim (but then you need to learn about 'vi': https://neovim.io/ which is a hurdle).

4.2 Online Platforms

4.2.1 (Short) Courses

- Short courses at Deeplearning.ai

  - Implementations of papers
  - Benchmarks
  - State-of-the-art tracking

4.2.2 Code Repositories

- Papers With Code
  - Implementations of papers
  - Benchmarks
  - State-of-the-art tracking

4.3 Community Resources

- Distill.pub
  - Interactive explanations
  - Visual learning
  - Deep insights

4.4 Academic Papers

4.4.1 Modern Breakthroughs

- "Deep Residual Learning for Image Recognition" (He et al., 2015)
- "Attention Is All You Need" (Vaswani et al., 2017)
- "Language Models are Few-Shot Learners" (Brown et al., 2020)

Part II

AI Act Europe

Chapter 5

AI Act Resources and References

- SURF startdocument AI Act

Part III

Train, Fine Tune, RAG

Chapter 6

Train, Fine Tune, RAG

Several ways to 'teach' the AI about the knowledge it needs to perform the task you need it for. The most easy of these is building a RAG system: Retrieval Augmented Generation.

Chapter 7

RAG: Retrieval Augmented Generation

Chapter 8

Finetune

Chapter 9

Training

Training a model from scratch is a complex and resource-intensive process. It involves collecting a large dataset, preprocessing the data, and training the model using powerful hardware. This is typically done by large organizations with significant resources.

short course: fine tuning

Part IV

Data

Chapter 10

Finding and Preparing Data

10.1 The Importance of Data

Data is the foundation of most machine learning projects. The quality and quantity of your data often matter more than the sophistication of your model.

10.2 Popular Data Sources

10.2.1 1. Public Datasets

- Kaggle
  - Competitions and datasets
  - Active community
  - Detailed documentation
- Eindhoven open data
  - lots of data about Eindhoven

Part V

Agents

Chapter 11

Agents

11.1 a source

- https://google.github.io/adk-docs/https://google.github.io/adk-docs/

Part VI

Neuron & Network

Chapter 12

Understanding the Perceptron

12.1 The Biological Inspiration: From Brain Neurons to Artificial Intelligence

The Perceptron represents one of the most fundamental concepts in artificial intelligence, drawing its inspiration directly from the human brain's neural structure. This groundbreaking idea was first introduced in 1943 by Warren S. McCulloch and Walter Pitts in their seminal paper 'A Logical Calculus of the Ideas Immanent in Nervous Activity', where they proposed a mathematical model of biological neurons.

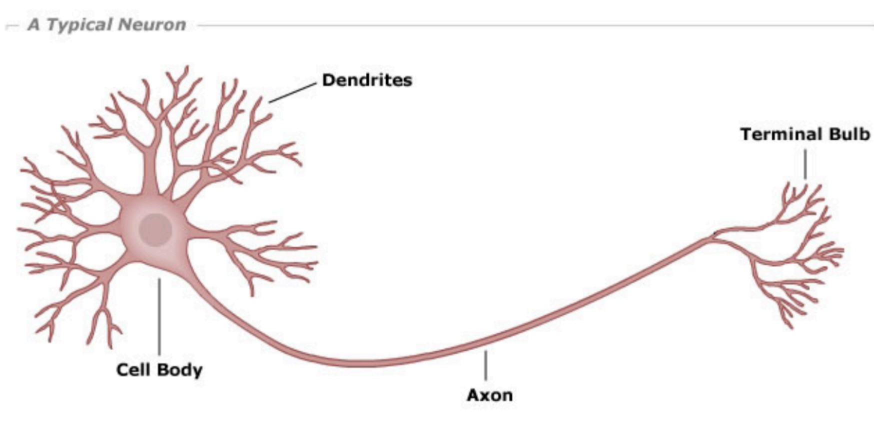

Figure 12.1: A typical biological neuron structure

12.2 From Biology to Machine: Implementing a Perceptron

A Perceptron's architecture mirrors its biological counterpart through three key components: `inputs`, `weights`, and a `bias`. Each input connection has an associated weight that determines its relative importance, while the bias helps adjust the Perceptron's overall sensitivity to activation.

Let's look at a simple yet useful perceptron with 2 inputs.

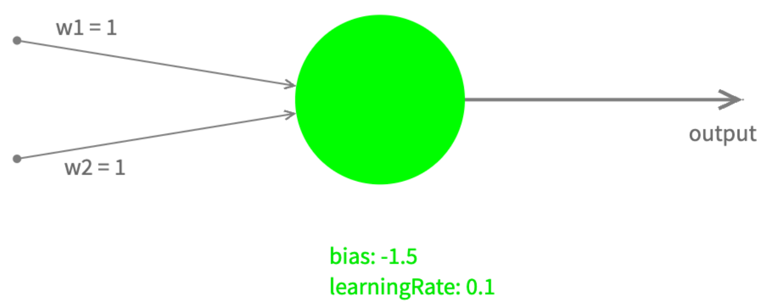

Figure 12.2: Perceptron's architectural diagram

We'll call our input values `x1`, `x2` with their corresponding weights `w1`, `w2`. The Perceptron processes these inputs in two steps:

1. First, it calculates a **`weighted sum`** and adds the bias: `z := w1*x1 + w2*x2 + bias`

2. Then, it applies what we call an **activation function** to produce the final output: let's use a very simpel activation function, called a Step function:

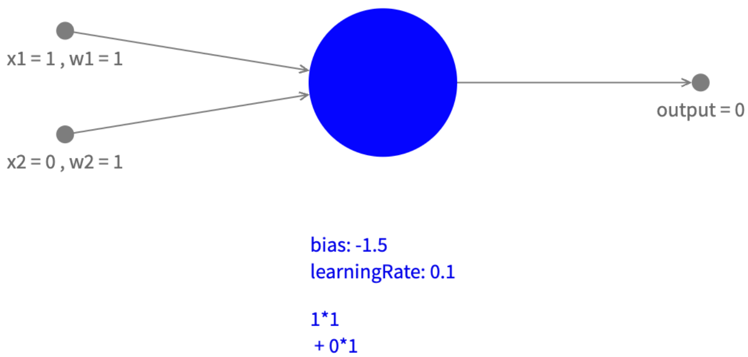

Figure 12.3: Perceptron's architectural diagram

$$\begin{cases} \text{Output is 1 if } z > 0 \\ \text{Output is 0 if } z \leq 0 \end{cases}$$

which determines the final output.

Let's restrict ourselves for now to possible input values 0 and 1: If we look at all possibilities combinations of input and the corresponding output we can create a table:

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

A close look will tell us that the output is only 1 when inputs are 1, and 0 in all other cases, which you could recognize as
a logical AND. So with these weights and bias this Perceptron can be used to act as a logical AND.

For different values it will behave like a logical OR (and more). Can you come up with those values?

12.3 Network

By combining several Perceptrons (sending the output of a perceptron to the input of another one) you can probably imagine that it is possible to create Networks of Perceptrons. By changing the values of weights and biases of the connected Perceptrons it is possible to build complex electronic circuits.

When we generalize this concept to other values, not only 0 and 1, and different activation functions, the Perceptron becomes an incredibly versatile tool. This generalization opens up possibilities for pattern recognition, classification tasks, regression problems, and complex decision-making systems. This is where the true power of neural networks begins to emerge, as they can learn to handle continuous data and make sophisticated decisions based on multiple inputs.

Up until now we didn't look at how a perceptron can learn and become smarter. That will be subject of next chapter chapters. The concept of a Perceptron was generalized to what we now call an (artificial) Neuron.

Search terms: Perceptron, Artificial Neuron, `Multi Layered Perceptron` (MLP), `(Artificial) Neural Network` (ANN).

12.4 First Implementation of Perceptron algorithm

According to Wikipedia:

> The artificial neuron network was invented in 1943 by Warren McCulloch and Walter Pitts in 'A logical calculus of the ideas immanent in nervous activity'. the Perceptron Machine was first implemented in hardware in the Mark I, which was demonstrated in 1960.
>
> It was connected to a camera with 20×20 cadmium sulfide photocells to make a 400-pixel image. The main visible feature is the sensory-to-association plugboard, which sets different combinations of input features. To the right are arrays of potentiometers that implemented the adaptive weights.

12.5 Reference

- wikipedia: perceptron

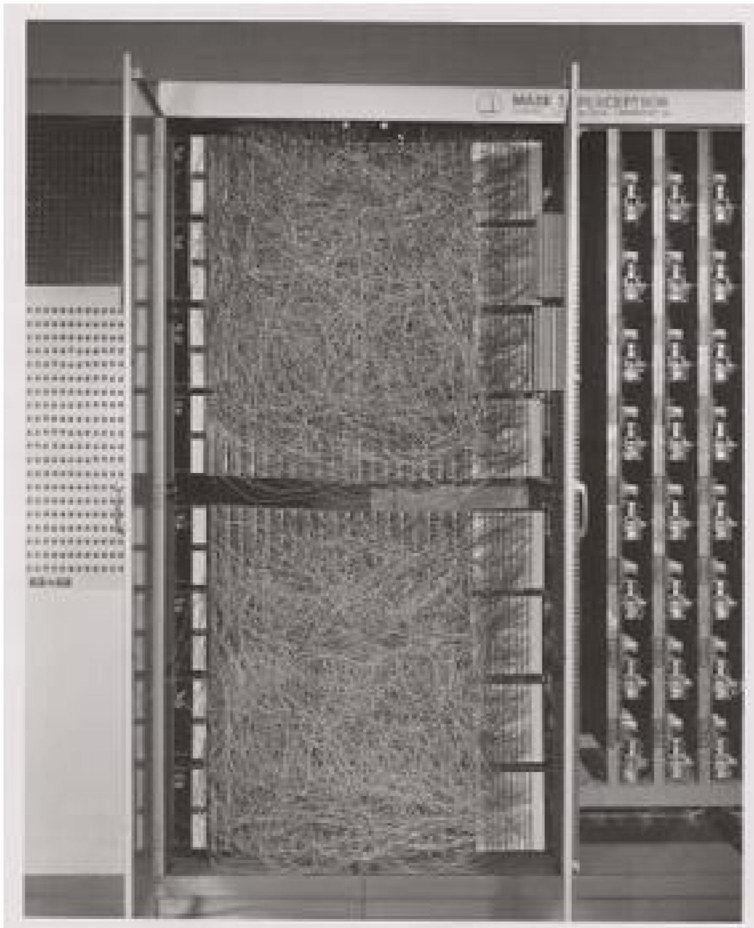

Figure 12.4: The Mark I Perceptron machine, the first implementation of the perceptron algorithm (source: wikipedia)

Chapter 13

The Learning Perceptron

One of the most fascinating aspects of Perceptrons is their ability to learn from examples. Instead of manually setting weights and bias, we can train a Perceptron to discover the optimal parameters through a process called supervised learning.

13.1 The Learning Algorithm

The learning process follows these key steps:

1. Start with random weights and bias
2. Present a training example
3. Compare the Perceptron's output with the desired output
4. Adjust the weights and bias based on the error
5. Repeat with more examples until performance is satisfactory

13.1.1 Mathematical Foundation

The weight update rule is elegantly simple:

```
new_weight := current_weight + learning_rate * error * input
```

Where:

- `learning_rate` is a small number (like 0.1) that controls how big each adjustment is
- `error` is the difference between desired and actual output (1 or -1)
- `input` is the input value for that weight

13.1.2 Training Process

To train the Perceptron, we have to have `labeled data` (ie. input data combined with the desired output for those values)

So for training AND gate behavior we have to list all combinations of 2 bits that are possible as input, and also the desired output value:

```
| Input  | Desired Output |
|--------|----------------|
| (0, 0) | 0              |
| (0, 1) | 0              |
| (1, 0) | 0              |
| (1, 1) | 1              |
```

and training (1 epoc) means calling the train function with each of these examples:

```
foreach dataItem in trainingData do:
    inputs := dataItem[0]
    desiredOutput := dataItem[1]
    learningPerceptron train(inputs, desiredOutput)
```

13.2 Visualizing the Learning Process

As the Perceptron learns, its decision boundary gradually moves to the correct position. You can monitor this progress by:

1. Tracking the error rate over time
2. Visualizing the decision boundary's movement
3. Testing the Perceptron with new examples

13.3 Practical Considerations

For successful learning: - Ensure your training data is representative - Consider using multiple training epochs (complete passes through the data) - Monitor for convergence (when the weights stabilize) - Be aware that not all problems are linearly separable

In the next chapter, we'll explore the limitations of what a single Perceptron can learn, which will lead us naturally to the need for more complex neural networks.

Chapter 14

Understanding Perceptron Limitations

14.1 The XOR Problem: A Classic Challenge

While Perceptrons are powerful tools for many classification tasks, they face a fundamental limitation: they can only solve linearly separable problems. The classic example of this limitation is the XOR (exclusive OR) function.

14.1.1 What is XOR?

The XOR function outputs 1 only when exactly one of its inputs is 1: - Input (0,0) $\rightarrow$ Output: 0 - Input (0,1) $\rightarrow$ Output: 1 - Input (1,0) $\rightarrow$ Output: 1 - Input (1,1) $\rightarrow$ Output: 0

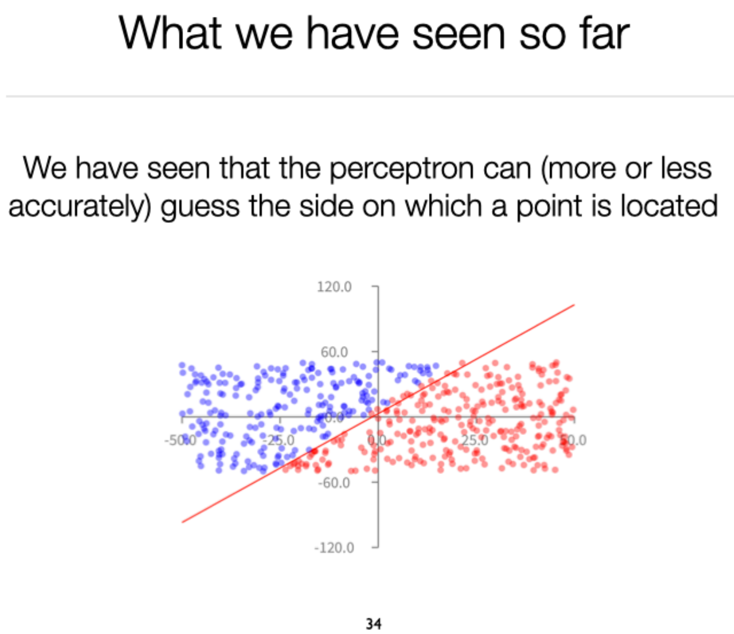

Figure 14.1: Visual representation of XOR problem

14.1.2 Why Can't a Single Perceptron Solve XOR?

A Perceptron creates a single straight line (or hyperplane in higher dimensions) to separate its outputs. However, the XOR problem requires two separate lines to correctly classify all points.

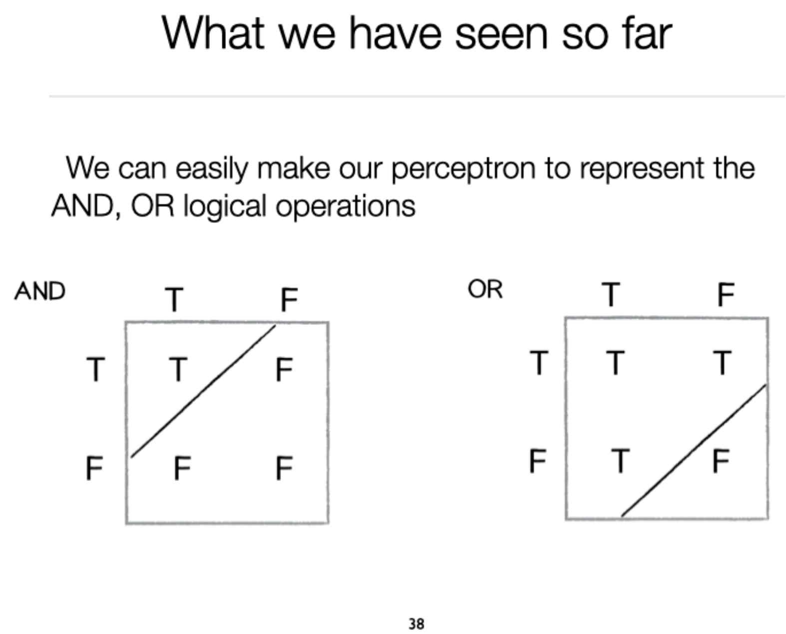

Figure 14.2: Attempted linear separation of XOR

As you can see, no single straight line can separate the points where output should be 1 (blue) from points where output should be 0 (red).

14.2 The Solution: Multiple Layers

To solve the XOR problem, we need to combine multiple Perceptrons in layers. This is our first glimpse at why we need neural networks!

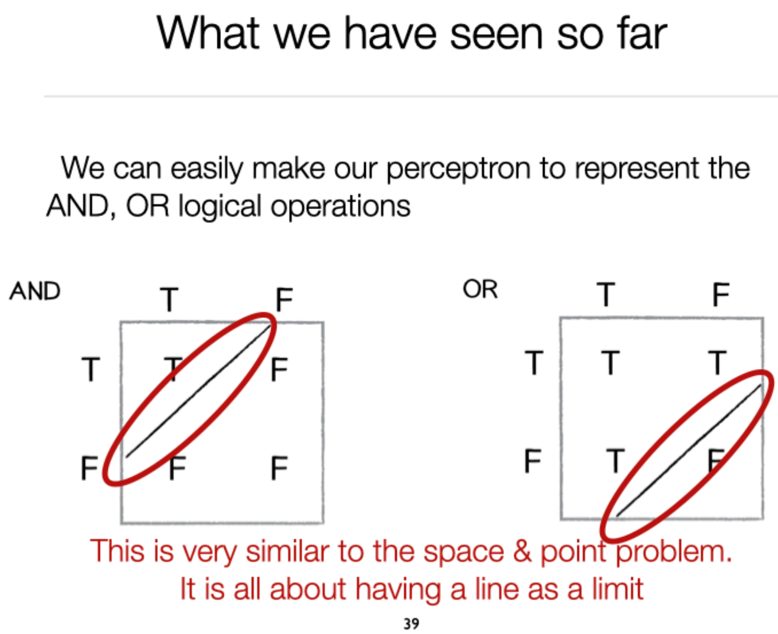

Figure 14.3: Multi-layer solution

By using multiple Perceptrons, we can: 1. First create separate regions with individual Perceptrons 2. Then combine these regions to form more complex decision boundaries

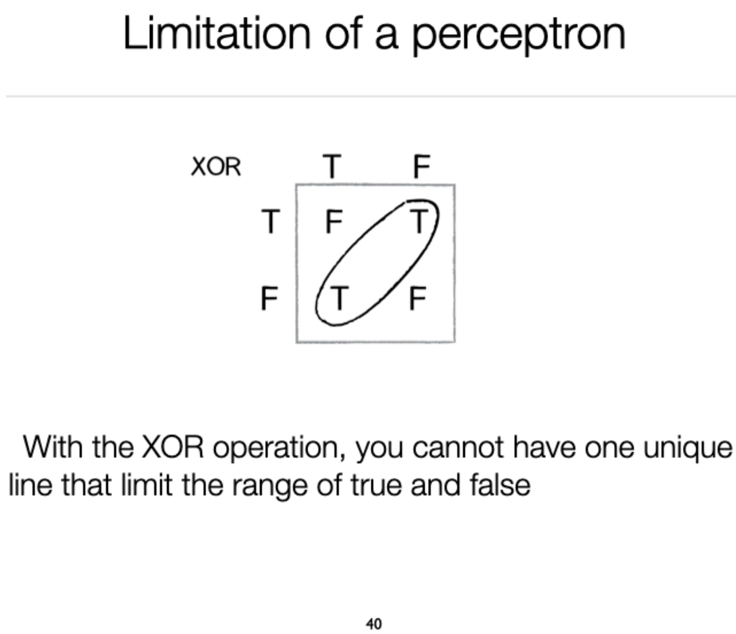

Figure 14.4: Complete neural network solution

14.3 Key Takeaways

1. Single Perceptrons can only solve linearly separable problems
2. Many real-world problems (like XOR) are not linearly separable
3. Combining Perceptrons into networks overcomes this limitation
4. This limitation led to the development of multi-layer neural networks

In the next section, we'll explore how to build and train these more powerful multi-layer networks.

Chapter 15

Introduction to Neural Networks

15.1 Beyond Single Perceptrons: Building Neural Networks

Having seen the limitations of single Perceptrons, we now venture into the fascinating world of neural networks. These powerful structures combine multiple Perceptrons in layers to solve complex problems that single Perceptrons cannot handle.

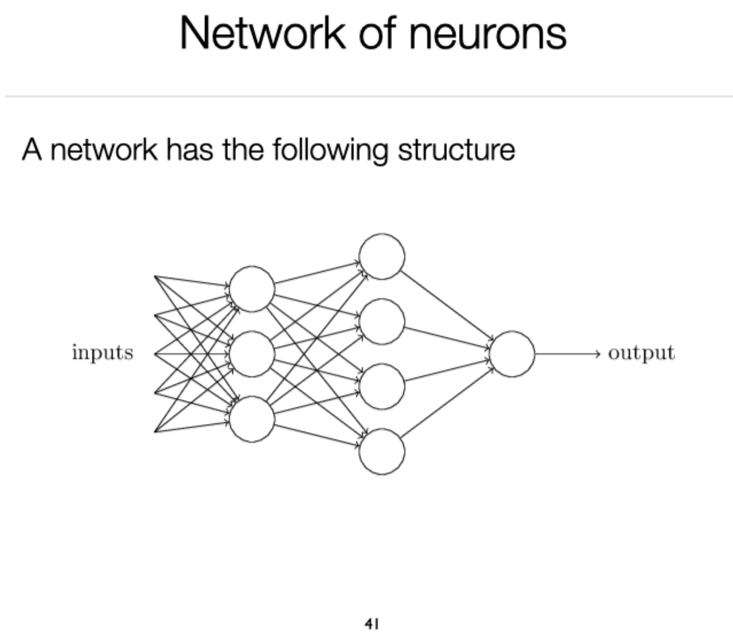

Figure 15.1: Basic neural network architecture

15.2 Understanding Network Architecture

A typical neural network consists of three main components:

1. **Input Layer**: Receives the raw data
2. **Hidden Layer(s)**: Processes the information through multiple Perceptrons
3. **Output Layer**: Produces the final result

15.2.1 Key Components

Each connection in the network has: - A weight that determines its strength - A direction of information flow (forward only) - An associated neuron that processes the incoming signals

15.3 How Information Flows

The network processes information in these steps:

1. Input values are presented to the input layer
2. Each neuron in subsequent layers:

  - Receives weighted inputs from the previous layer
  - Applies its activation function
  - Passes the result to the next layer
3. The output layer produces the final result

15.4 Creating a Simple Network

You probably have seen a picture of a neural network before.

Neural Networks can

1. Can solve problems thar are more difficult.
2. Handle complex pattern recognition
3. Learn hierarchical features automatically
4. Scale well to large problems

In the next sections, we'll explore practical applications and see how to train networks on real-world data.

Chapter 16

Practical Example: Classifying Iris Flowers

16.1 A Real-World Machine Learning Challenge

The Iris flower classification problem is a classic example in machine learning. It involves predicting the species of an Iris flower based on measurements of its physical characteristics. This problem perfectly illustrates how neural networks can solve real-world classification tasks.

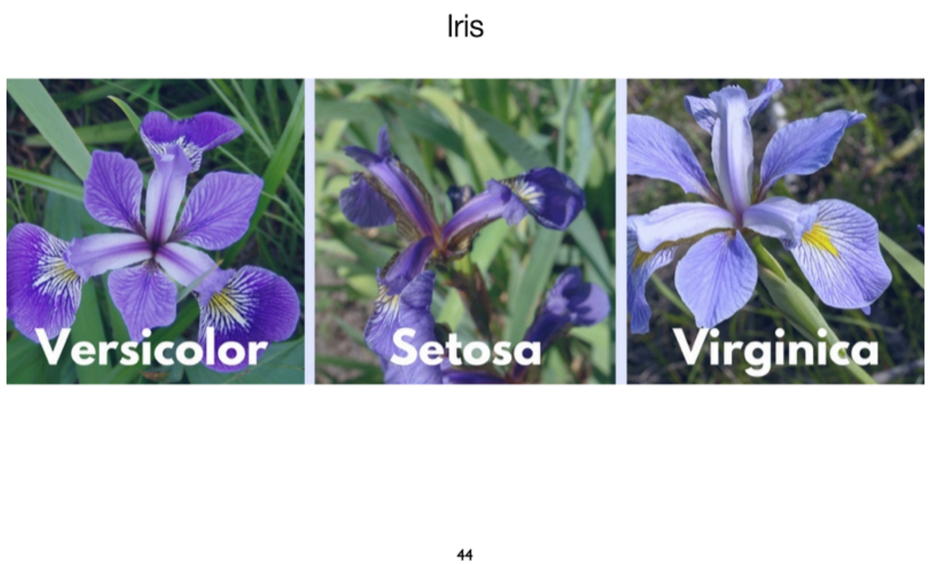

Figure 16.1: Different types of Iris flowers

16.2 The Dataset

The Iris dataset includes measurements of three different Iris species: - Iris Setosa - Iris Versicolor - Iris Virginica

For each flower, we have four measurements: 1. Sepal length 2. Sepal width 3. Petal length 4. Petal width

Building a network that can do this is really outside of the scope of these notes, but a lot of info can be found on the internet on `Iris Classification`.

16.3 Key Learning Points

1. Neural networks can handle multi-class classification
2. Real-world data often needs preprocessing
3. We can measure success with accuracy metrics
4. The same principles apply to many similar problems

This practical example demonstrates how neural networks can solve real classification problems. In the next section, we'll explore the mathematics behind how these networks learn.

Chapter 17

The Mathematics Behind Neural Networks

17.1 Understanding the Magic

While neural networks might seem magical, they're built on solid mathematical foundations. Let's demystify (a bit of) how they actually work under the hood.

17.2 The Building Blocks

17.2.1 1. Neurons and Weights

To be formally correct we should say `artificial neuron` to distinguish them from `biological neurons` like we have in our brain. A neuron normally has inputs: 1, or 2, or $\cdots$

Each neuron performs two key operations: 1. Weighted sum of inputs. 2. Activation function: $a = f(z)$

17.2.2 2. Activation Functions

Common activation functions include:

1. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
    - Outputs between 0 and 1
    - Useful for probability predictions
2. ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$
    - Simple and efficient
    - Helps prevent vanishing gradients
3. Tanh: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

- Outputs between -1 and 1
- Often better than sigmoid for hidden layers

17.3 The Learning Process

17.3.1 1. Forward Propagation

Information flows through the network.

17.3.2 2. Loss Calculation

Measure the network's error and Backpropagation

- what is the output?
- What would be my desired output?

The smaller the difference between the output I got and the output I desired, the better the output of my model is. This difference is calculated with a so-called Loss function. Backpropagation is an algorithm that helps make that difference small. When backpropagation is performed we call that Training the AI model.

Chapter 18

Exploring Neural Network Architectures

18.1 The Rich Landscape of Neural Networks

Neural networks come in many shapes and sizes, each designed to excel at specific types of tasks. Let's explore some of the most important architectures and their applications.

18.2 Feedforward Neural Networks (FNN)

The classic architecture: Information flows in one direction:

- Input layer → Hidden layer(s) → Output layer
- Perfect for classification and regression tasks
- Examples: Our Iris classifier, handwriting recognition

18.3 Convolutional Neural Networks (CNN)

Inspired by the human visual cortex:

- Specialized for processing grid-like data (images, video)
- Uses convolution operations to detect patterns
- Excellent at feature extraction
- Applications: Image recognition, computer vision, medical imaging

18.4 Recurrent Neural Networks (RNN)

Networks with memory:

- Can process sequences of data
- Information cycles through the network
- Great for time-series data and natural language
- Applications: Language translation, speech recognition, stock prediction

18.5 Long Short-Term Memory (LSTM)

A sophisticated type of RNN:

- Better at remembering long-term dependencies
- Controls information flow with gates
- Solves the vanishing gradient problem
- Applications: Text generation, music composition

18.6 Autoencoders

Self-learning networks:

- Learn to compress and reconstruct data
- Useful for dimensionality reduction
- Can detect anomalies
- Applications: Data compression, noise reduction, feature learning

18.7 Generative Adversarial Networks (GAN)

Two networks competing with each other:

- Generator creates fake data
- Discriminator tries to spot fakes
- Through competition, both improve
- Applications: Creating realistic images, style transfer, data augmentation

18.8 Choosing the Right Architecture

The choice of architecture depends on:

1. Type of data (images, text, time-series)
2. Task requirements (classification, generation, prediction)
3. Available computational resources
4. Need for real-time processing

18.9 Future Directions

Neural network architectures continue to evolve:

- Hybrid architectures combining multiple types
- More efficient training methods
- Better handling of uncertainty
- Integration with other AI techniques

In the next section, we'll dive deeper into training these networks effectively.

Chapter 19

Resources and References AI

19.1 Books

1. **Neural Networks and Deep Learning**
    - Author: Michael Nielsen
    - Free Online Book
    - Perfect for beginners and intermediate learners
    - Clear explanations with interactive examples
2. **Deep Learning**
    - Authors: Ian Goodfellow, Yoshua Bengio, Aaron Courville
    - Available Online
    - Comprehensive coverage of deep learning
    - Industry standard reference
3. **Agile AI in Pharo**
    - Author: Alexandre Bergel
    - Practical implementation in Pharo
    - Hands-on examples and exercises
    - Book Link

19.2 Video Courses and Tutorials

19.2.1 1. Foundational Series

- 3Blue1Brown Neural Networks

  - Visual explanations
  - Mathematical intuition
  - Clear animations

https://www.youtube.com/watch?v=O5xeyoRL95U

19.2.2 2. Programming Tutorials

- Fast.ai Deep Learning Course
  - Practical approach
  - Top-down learning
  - Real-world applications

19.2.3 3. Advanced Topics

- Stanford CS231n
  - Computer Vision
  - Deep Learning
  - State-of-the-art techniques

19.3 Online Platforms

19.3.1 1. Interactive Learning

- Kaggle Learn
  - Hands-on exercises
  - Real datasets
  - Community support

19.3.2 2. Research Papers

- arXiv Machine Learning
  - Latest research
  - Open access
  - Preprint server

19.3.3 3. Code Repositories

- Papers With Code
  - Implementations of papers
  - Benchmarks
  - State-of-the-art tracking

19.4 Community Resources

19.4.1 1. Forums and Discussion

- r/MachineLearning
- Cross Validated
- AI Stack Exchange

19.4.2 2. Blogs and Newsletters

- Distill.pub
  - Interactive explanations
  - Visual learning
  - Deep insights

19.4.3 3. Tools and Libraries

- TensorFlow
- PyTorch
- Scikit-learn

19.5 Academic Papers

19.5.1 Foundational Papers

- "A Logical Calculus of Ideas Immanent in Nervous Activity" (McCulloch & Pitts, 1943)
- "Learning Internal Representations by Error Propagation" (Rumelhart et al., 1986)
- "Gradient-Based Learning Applied to Document Recognition" (LeCun et al., 1998)

Part VII

Experiments

Chapter 20

Experiments

Experiments, maybe incomplete... never finished, the whole reutemeteut!