ELEC-C5220
Lecture 1: Introduction

Machine learning in information technology

Aalto University
School of Electrical
Engineering

Lauri Juvela

11.1.2024

Language issue – Finnish or English?

- Materials are in English
- Finnish translations possible if time allows
- Lectures in whatever language the fewest participants don’t understand (poll)
- Exercise and Project materials are in English, but you can give solutions in Finnish, Swedish, or English
- How to program in Finnish? Python is basically plain English anyway

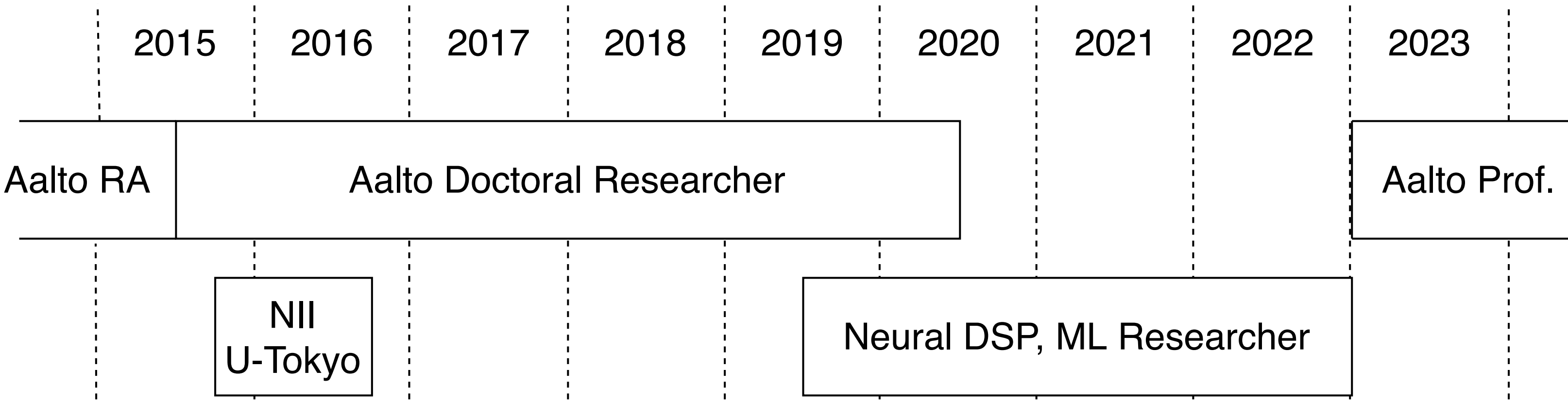
Introduction to the course

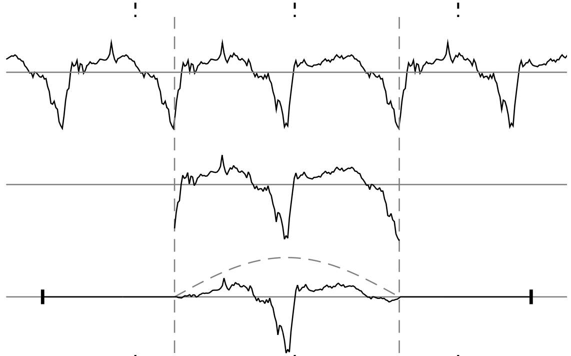
- Motivation
- Lecturer: personal introduction
- Intended learning outcomes
- Teaching and learning activities: Lectures, Exercises, Project
- Assessement
- Content
- Assessment
- Schedule

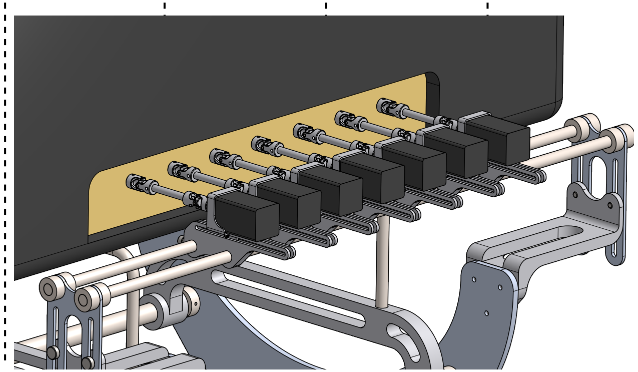
Motivation

- ELEC-C5520 is a new course intended to give bachelor’s students early exposure to practical deel learning
- Machine learning methods, especially Deep Learning, are essential in IT applications and research
- Familiriaise students to topics related to ELEC Dept. of Information and Communications Engineering (DICE)

Lauri Juvela – timeline







Lauri Juvela – current research

- **Assistant Professor in Speech and Language Technology**
- **Speech Synthesis research group**
- **Intererests**
    - Deep generative methods, Generative AI
    - Watermarking and deepfake detection
    - Efficiency, control, and interpretability in speech synthesis
    - Differentiable digital signal processing (DDSP)

Intended learning outcomes

After completing the course, the student can

- identify general principles and concepts of machine learning, especially neural net-works and deep learning, and their most essential methods.
- apply machine learning software to real-world applications of information technology, including speech technology, signal processing and communications.
- specify and implement machine learning problems and solutions in speech technology, signal processing and communications.

Teaching and learning activities:
1) Lectures

- **Lectures on Thursdays 14-16 at OK3, F175a**
- **Attendance is voluntary**
- **Lectures are designed to contain the information needed for solving the exercises**
- **Books are nice-to-know background:**
    - Deep Learning https://www.deeplearningbook.org
    - Speech Processing https://speechprocessingbook.aalto.fi

Teaching and learning activities

2) Exercises

- Exercise sessions on Mondays in Maari-A at 14, new exercise published for each session
- Exercises are Python programming with the PyTorch library
- Points for passing unit tests, automatic grading with nbgrader
- Hosted on Aalto JupyterHub, can be done remotely
- Exercise deadline on Mondays before the next session

Teaching and learning activities

3) Project

- **Build a small but practical deep learning system for speech denoising**
- **Groups of 1-3 people with random member assignment**
    - There will be a poll about your preferred group size
- **Two milestones with programming requirements (unit tests w)**
    - Data providers **(DL 7.3.)**
    - Model functionality **(DL 21.3.)**
- **Final report (DL 18.4.)**

Assessment

- No exam
- 60% from Exercise points
- 40% from Project points
- Grade 1-5
- 50% points required to pass

Workload

- Lectures 10x2h + independent study = 40h
- Exercises: 10x2h sessions + independent work = 50h
- Project work (3x 10h programming + report 5h): 35h
- Total workload 135 hours = 5 ECTS credits

Course content

Lecture 1: Introduction

- Practical information about the course
- Binary classification with simple fully connected neural networks

Course content

Lecture 2: Representations

- **Tensors in PyTorch, how to represent structure in data**
- **Images, audio, text and other discrete data**
- **Audio-as-image – spectrograms (short-time Fourier transforms)**

Course content

Lecture 3: Spoken digit recognition

- Convolution neural networks (CNNs)
- Multi-class classification
- Working with speech data

Course content

Lecture 4: Audio effect modeling

- **Recurrent neural networks (RNNs)**
- **Regression tasks**
- **Guitar amplifier modeling with neural networks**

Course content

Lecture 5: Losses, metrics and evaluation

- How to build expert knowledge into deep learning systems?
- Generic vs. specialised
- Perceptual metrics for speech and audio
- Why good metrics may not be good loss functions?

Course content

Lecture 6: Denoising and source separation

- Data augmentation
- U-Net architecture
- Speech denoising and enhancement
- Project Topic and Introduction

Course content

Lecture 7: Language modeling

- **Simple character-based language models**
- **Transformers (or maybe RNNs)**
- **Autoregressive text generation**

Course content

Lecture 8: Automatic speech recognition (ASR)

- **Recognise the components and sub-problems in ASR**
  - Acoustic model
  - Language model
  - Decoding
- **Inspect and experiment with a pre-trained system**

Course content

Lecture 9: Text-to-Speech (TTS) synthesis

- **Recognise the components and sub-problems in TTS**
  - Acoustic modeling
  - Waveform synthesis
- **Inspect and experiment with a pre-trained system**

Course content

Lecture 10: Profiling and energy use

- Deep learning models are often expensive
- Learn how to measure and estimate computational cost
- Sustainability

Schedule

- **Available on MyCourses**

Break

Questions?

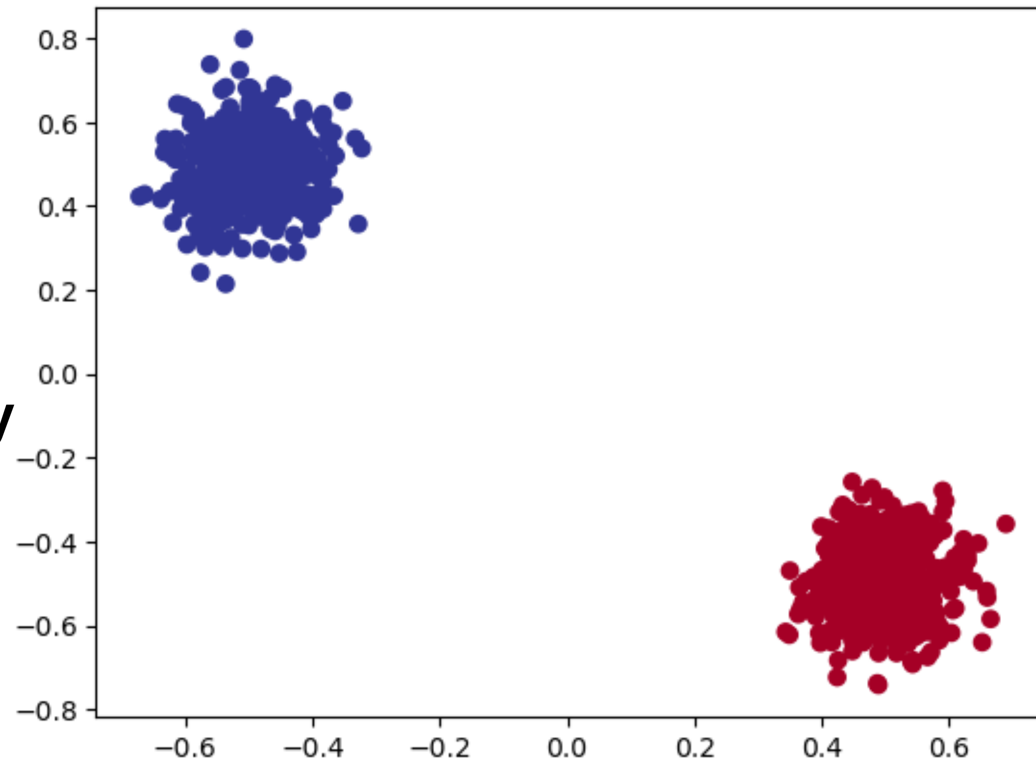
Introduction to Neural Networks

- Linear separability and linear models for classification
- Motivation for non-linear models
- Simple neural networks
- Backpropagation

Aalto University
School of Electrical
Engineering

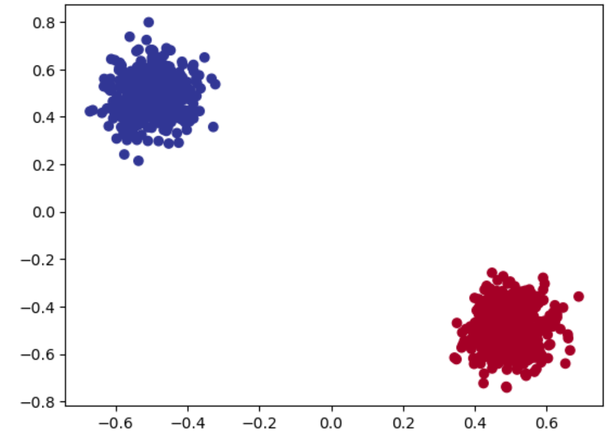
Linear classifier – Example 1

- Given x and y coordinates, what is the probability that data point is blue?
- Separate two data classes (blue and red) by a straight line



Linear classifier – Example 1

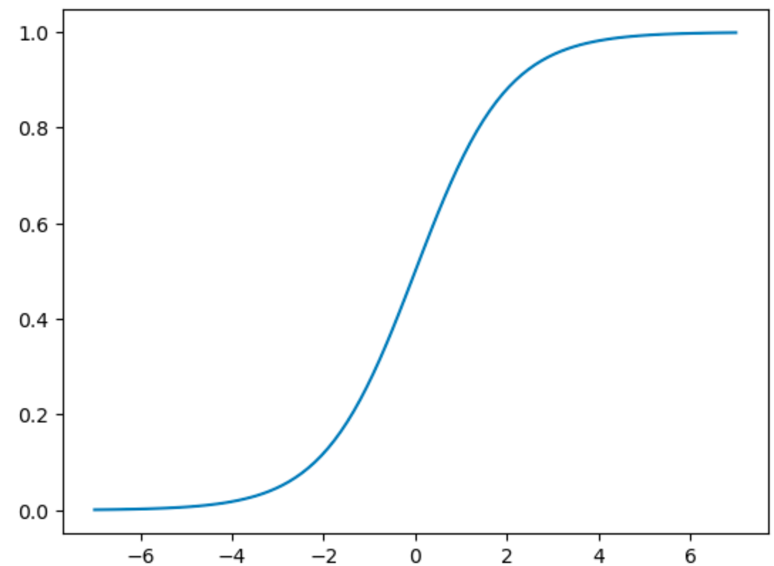
- **Label** $\mathbf{y} = \begin{cases} 1 & \text{if blue} \\ 0 & \text{if red} \end{cases}$ $\mathbf{y} \in \mathbb{R}^1$



- **Data point** $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$
- **Model** $\hat{\mathbf{y}} = \sigma(\mathbf{A}\mathbf{x} + \mathbf{b})$
- **Model parameters** $\mathbf{A} \in \mathbb{R}^{1\times 2}, \mathbf{b} \in \mathbb{R}^1$
- **Loss function** $L_{\text{BCE}} = \mathbb{E}[\mathbf{y} \log \hat{\mathbf{y}} + (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}})]$

Logistic sigmoid function

- Normalise model output to range (0,1) using the sigmoid function
- Use Logistic Regression to fit a linear model

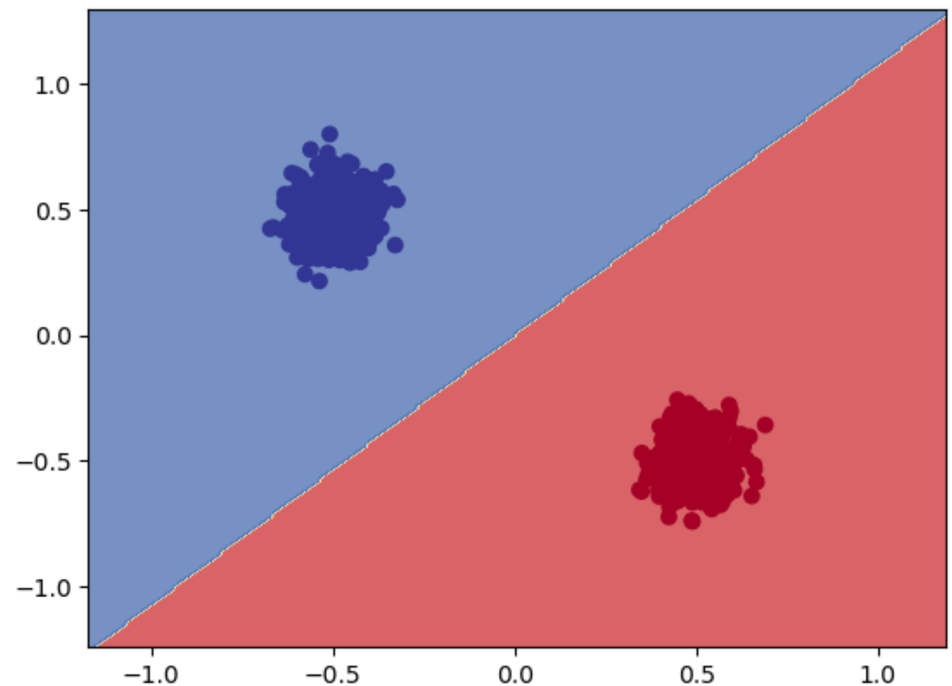


How to train a linear binary classifier?

- **Map input features to a scalar output prediction**
- **Normalise output probability to look like a probability**
- **Minimise loss function over data set, predictions should look like ground truth labels**
- **Linear models usually have closed-form optimal solutions, but we use gradient descent for everything on this course**

Linear classifier – Example 1

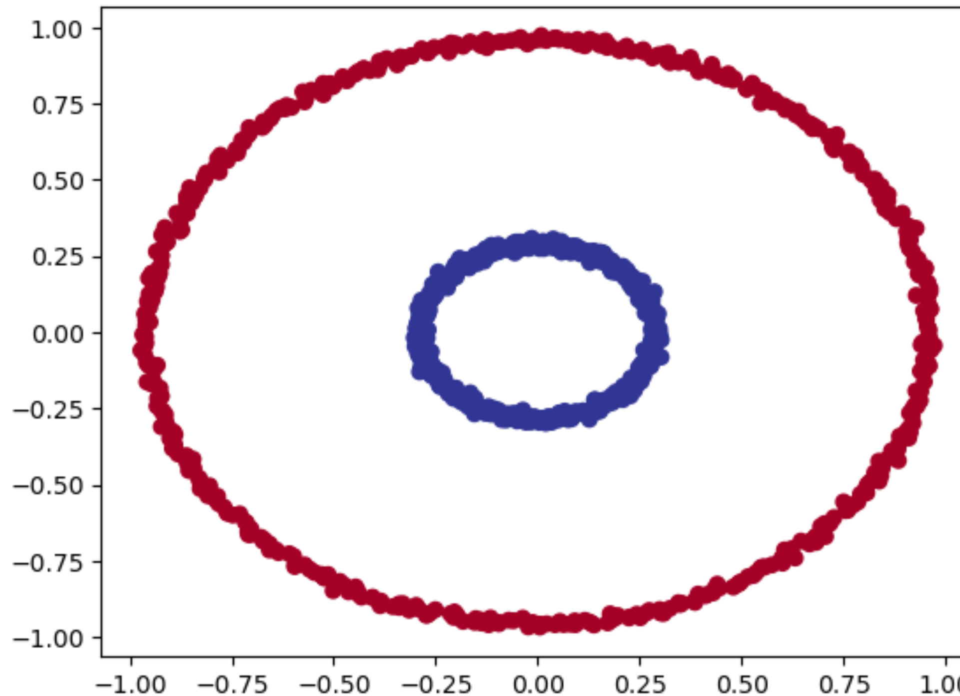
- **Problem is linearly separable**
- **Separating line is called the “decision boundary”**



Aalto University
School of Electrical
Engineering

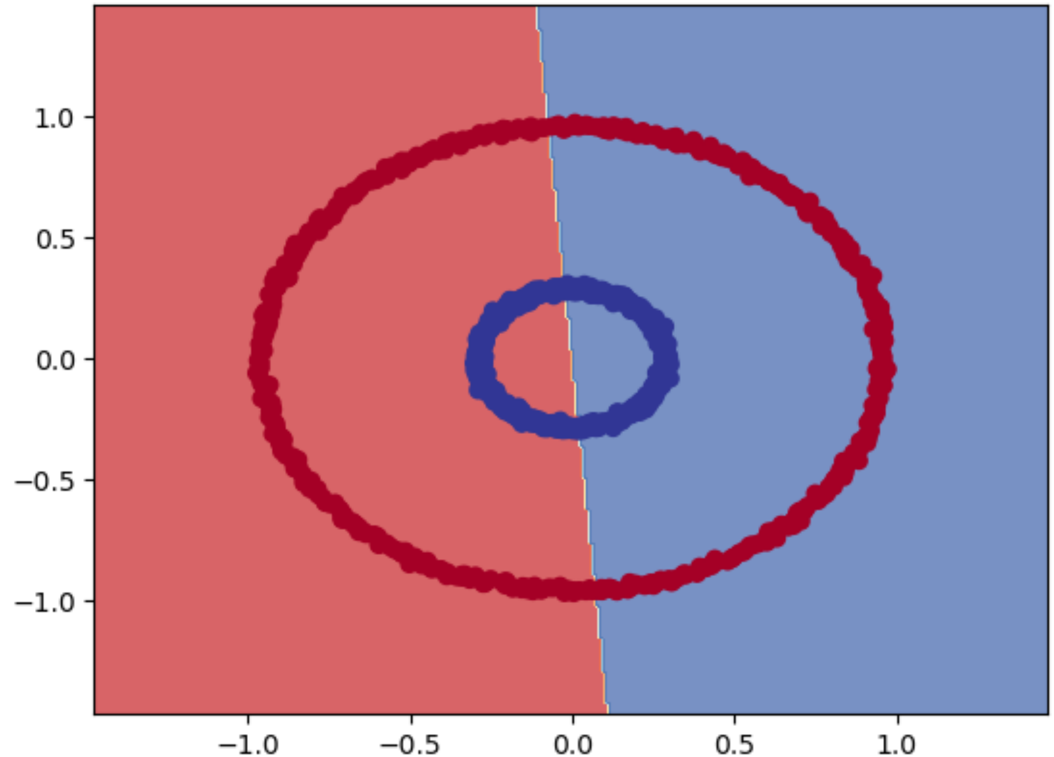
Linear classifier – Example 2

- Given x and y coordinates, what is the probability that data point is blue?
- Separate two data classes (blue and red) by a straight line



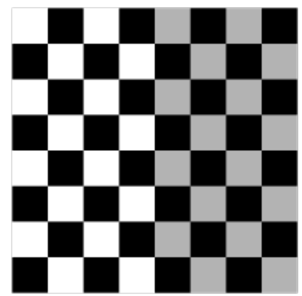
Linear classifier – Example 2

- **Best linear model is bad**
- **How can we do better?**

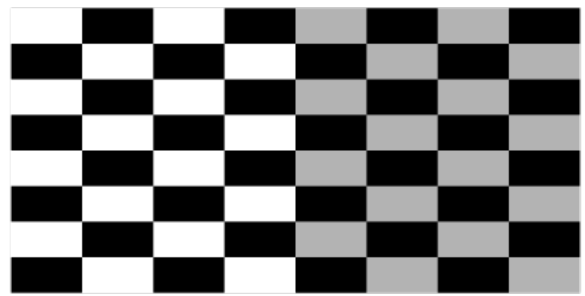


What can linear models do?

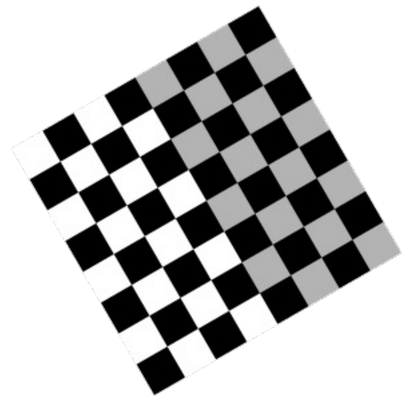
- Similar problem: separate white and grey squares on the checkerboard
- Linear map: y = Ax
- Affine map: y = Ax + b



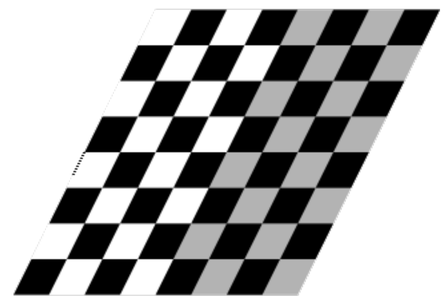
Scaling and stretching



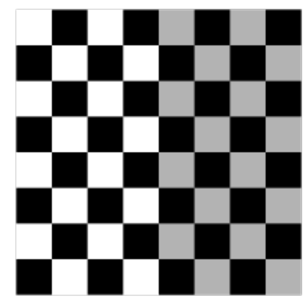
Rotation



Shearing

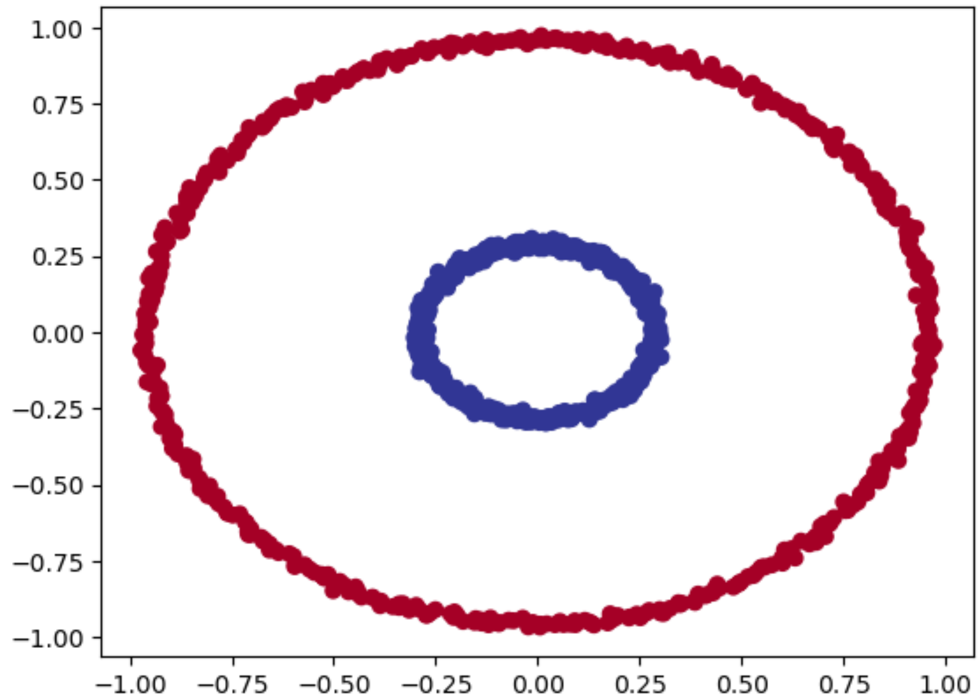


Translation



Linear classifier – Example 2

- **Can I fold the paper, please?**

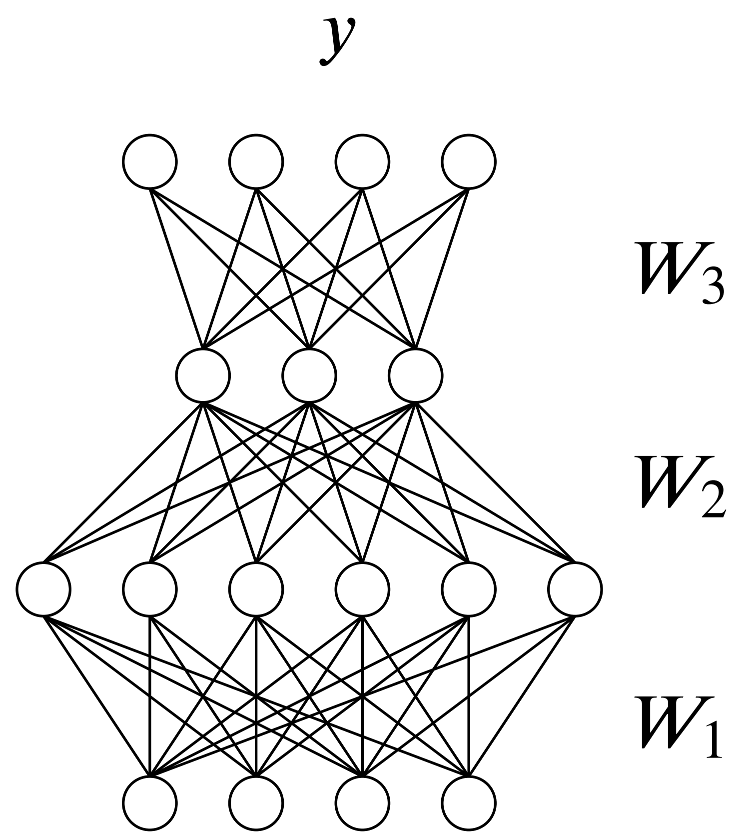


End of linear classification

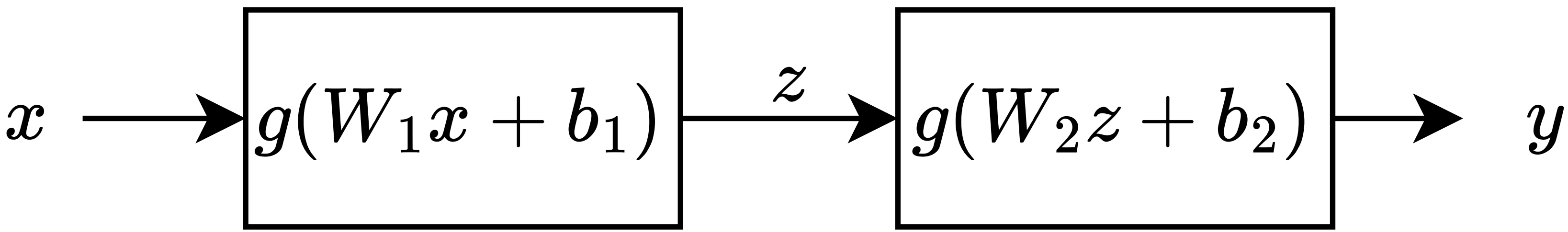
- Questions
- Next: a neural network

Neural networks

- **Common visualisation: draw every connection**
- **Scalar version is tedious to draw and work with**
- **Matrix version is more practical!**

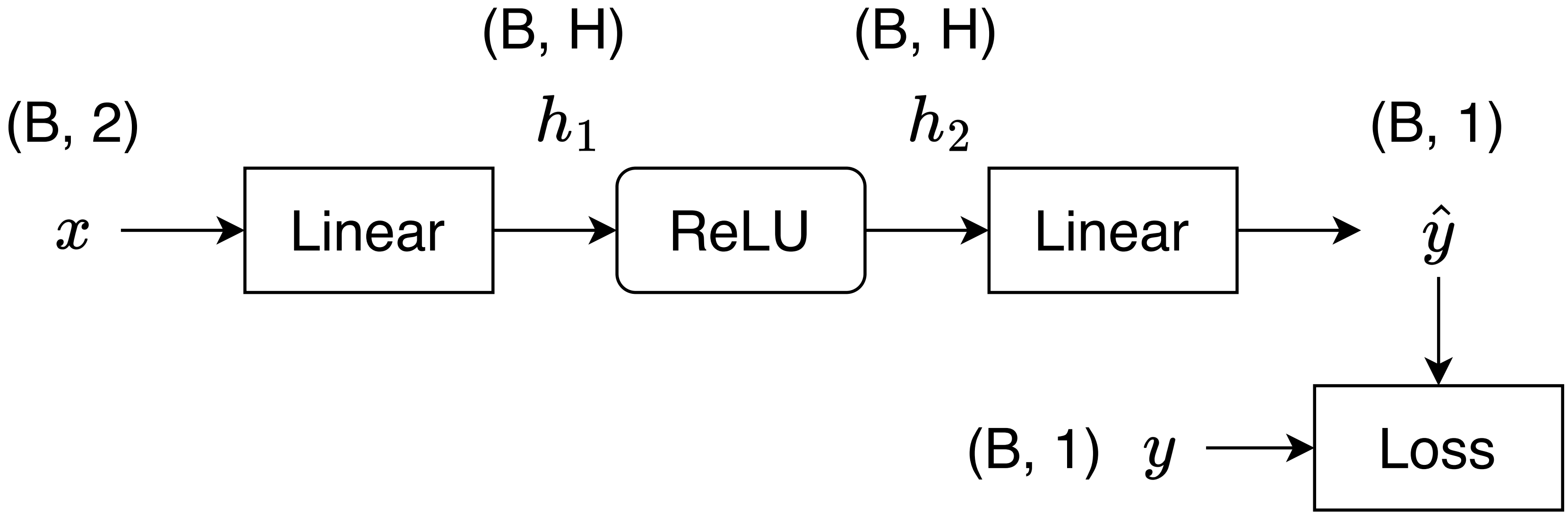


Neural network block diagrams are more friendly for ELEC students



- **Vector variables: x (input), z (hidden activation), y (predicted output)**
- **Affine layer: Weight W and bias b**
- **Non-linear activation function: g()**

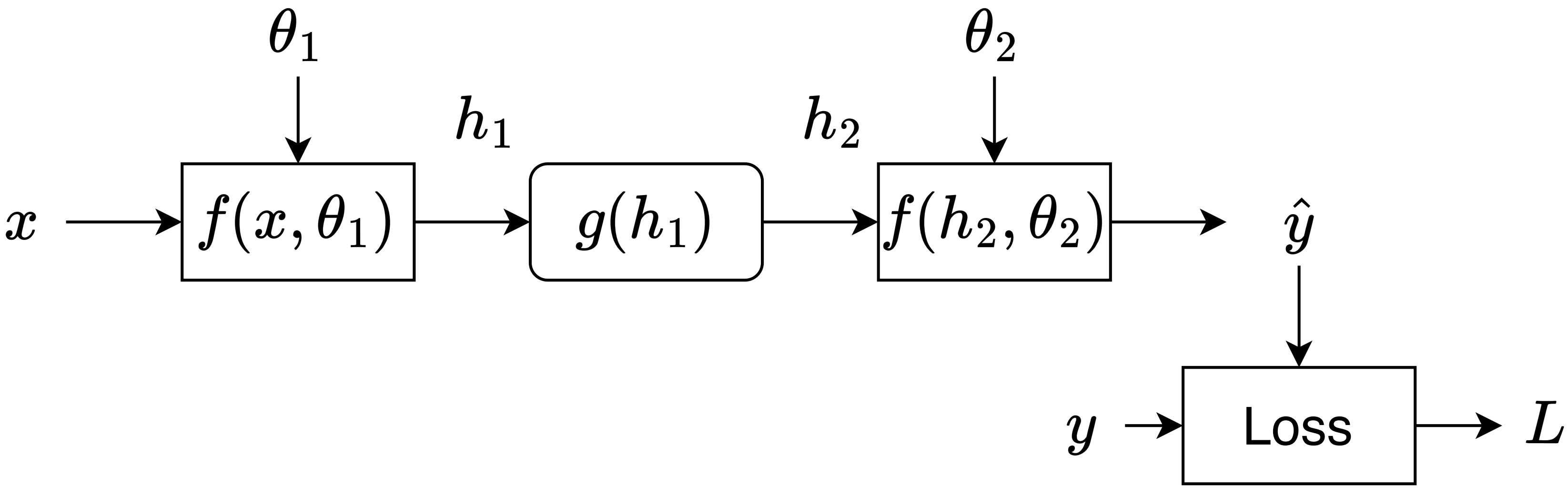
Neural net for Example 1



B = Batch size, number of datapoints in minibatch

H = Hidden size, network hyper parameter

Neural network forward pass



Gradient descent

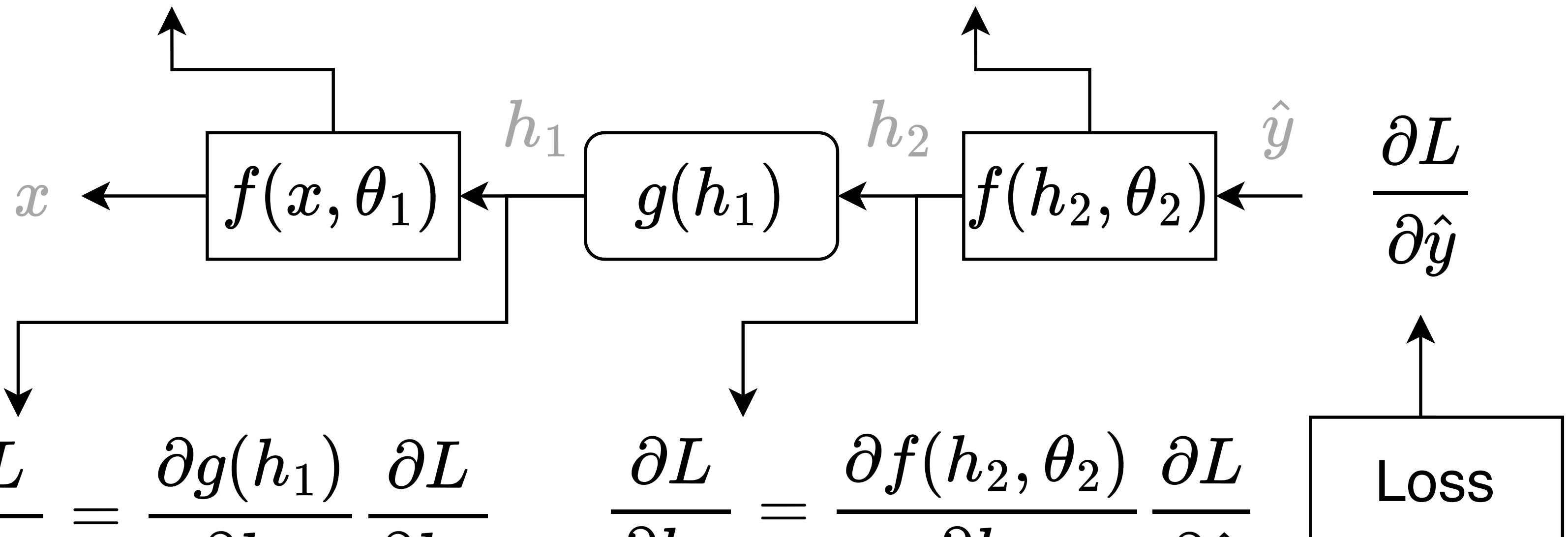
- **Update algorithm for parameters (theta)**

$$\theta \leftarrow \theta - \nu \frac{\partial L}{\partial \theta}$$

- **PyTorch has automatic gradient estimation, you only ever need to worry about the forward pass!**

Neural network backward pass

$$\frac{\partial L}{\partial \theta_1} = \frac{\partial f(h_1, \theta_1)}{\partial \theta_1} \frac{\partial L}{\partial h_1} \qquad \frac{\partial L}{\partial \theta_2} = \frac{\partial f(h_2, \theta_2)}{\partial \theta_2} \frac{\partial L}{\partial \hat{y}}$$



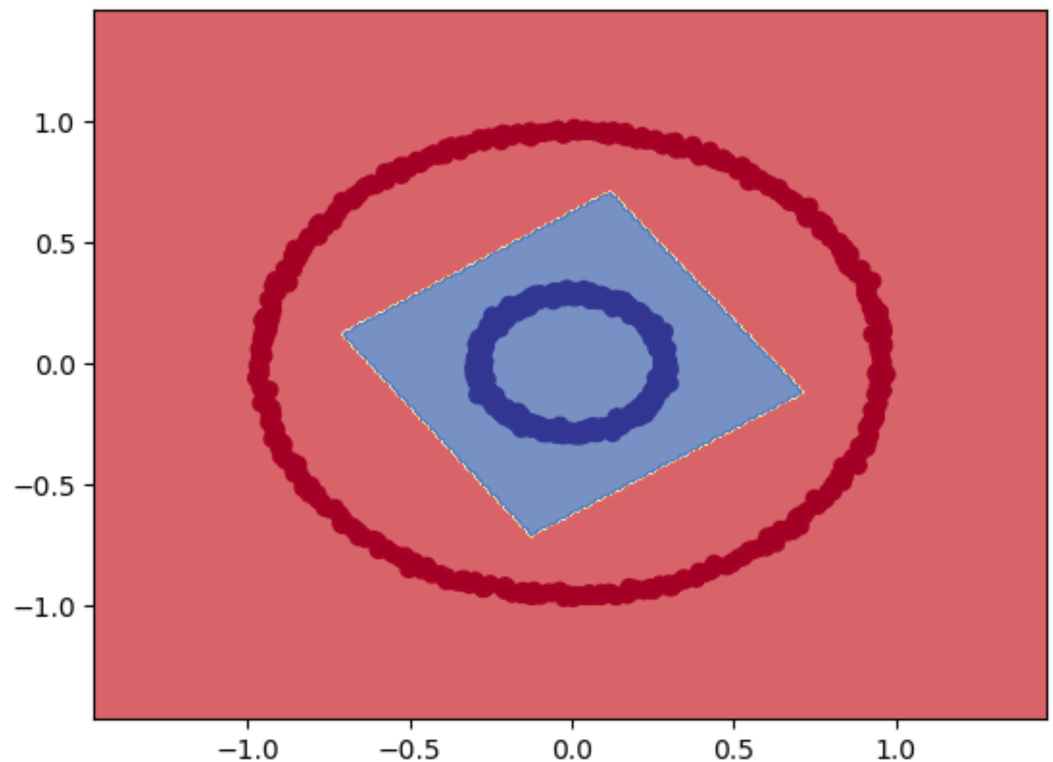
$$\frac{\partial L}{\partial h_1} = \frac{\partial g(h_1)}{\partial h_1} \frac{\partial L}{\partial h_2} \qquad \frac{\partial L}{\partial h_2} = \frac{\partial f(h_2, \theta_2)}{\partial h_2} \frac{\partial L}{\partial \hat{y}}$$

How to train a neural net binary classifier?

- Map input features to a scalar output prediction
- Normalise output probability to look like a probability
- Minimise loss function over data set, predictions should look like ground truth labels
- Use stochastic gradient descent and backpropagation

Neural network decision boundary

- **Classes are not linearly separable, but the decision boundary can be constructed from piece-wise linear segments**



End of Lecture 1

Questions?