



UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Machine learning – Block 3

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2025



This week's lecture

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Feed-Forward Neural Networks
- Regularization of Neural Networks
- Neural Network Optimization



On this weeks assignment

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- It can be tricky to get Tensorflow and R to work. **Start early and use the zoom sessions.**
- Don't run Tensorflow in markdown.



UPPSALA
UNIVERSITET

- **Introduction**
- **Feed-Forward Neural Networks**
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- **Regularization**
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- **Optimization of Neural Networks**
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- **Neural Networks in Practice**

Section 1

Introduction



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Hype: Deep Learning

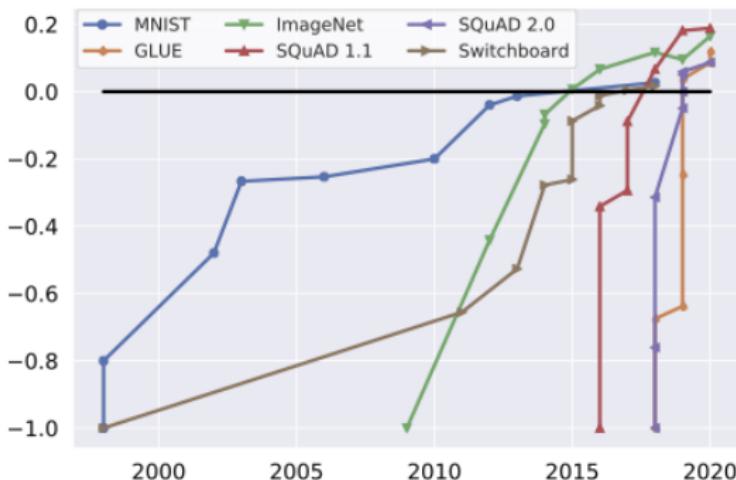


Figure: Benchmark saturation over time for popular benchmarks, normalized with initial performance at mi- nus one and human performance at zero. (Kiela et al., 2021)



The Hype: Large Language Models

- **GPT-3 (2020)** 175B parameters; breakthrough in general language capabilities.

- **Introduction**

- **Feed-Forward Neural Networks**

- Feed-Forward Neural Networks
- Output Units
- Hidden Units
- Architecture design

- **Regularization**

- Norm penalty
- Data augmentation
- Multi-Task Learning
- Early Stopping
- Drop-out

- **Optimization of Neural Networks**

- Difficulties in Neural Network Optimization
- Local minima
- Plateaus and Saddle Points
- Cliffs, exploding and vanishing gradients
- Parameter initialization

- **Neural Networks in Practice**



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Hype: Large Language Models

- **GPT-3 (2020)** 175B parameters; breakthrough in general language capabilities.
- **2021–2022: Scaling + Instruction Tuning : ChatGPT**
 - Models such as GPT-3.5, Gopher, Chinchilla, PaLM.
 - Emergence of instruction-following (ChatGPT)



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Hype: Large Language Models

- GPT-3 (2020) 175B parameters; breakthrough in general language capabilities.
- 2021–2022: Scaling + Instruction Tuning : ChatGPT
 - Models such as GPT-3.5, Gopher, Chinchilla, PaLM.
 - Emergence of instruction-following (ChatGPT)
- 2023: GPT-4 Era
 - Multimodal models (image+text)
 - Open models: LLaMA, Falcon
 - Focus on safety and alignment



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Hype: Large Language Models

- **GPT-3 (2020)** 175B parameters; breakthrough in general language capabilities.
- **2021–2022: Scaling + Instruction Tuning : ChatGPT**
 - Models such as GPT-3.5, Gopher, Chinchilla, PaLM.
 - Emergence of instruction-following (ChatGPT)
- **2023: GPT-4 Era**
 - Multimodal models (image+text)
 - Open models: LLaMA, Falcon
 - Focus on safety and alignment
- **2024: Open-Weights Explosion**
 - LLaMA-2/3, Mistral, Mixtral (MoE)
 - Efficient inference and training.
 - Training data reached internet-size.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Hype: Large Language Models

- **GPT-3 (2020)** 175B parameters; breakthrough in general language capabilities.
- **2021–2022: Scaling + Instruction Tuning : ChatGPT**
 - Models such as GPT-3.5, Gopher, Chinchilla, PaLM.
 - Emergence of instruction-following (ChatGPT)
- **2023: GPT-4 Era**
 - Multimodal models (image+text)
 - Open models: LLaMA, Falcon
 - Focus on safety and alignment
- **2024: Open-Weights Explosion**
 - LLaMA-2/3, Mistral, Mixtral (MoE)
 - Efficient inference and training.
 - Training data reached internet-size.
- **2025: Multimodal Models**
 - Native multimodal architectures (audio–vision–text).
 - Agents: reasoning + planning + tool execution.
 - GPT-5 released



The Hype

- **Introduction**
- **Feed-Forward Neural Networks**
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- **Regularization**
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- **Optimization of Neural Networks**
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- **Neural Networks in Practice**
 - Although - (Deep) Neural Networks are not a silver bullet



The Hype

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice
 - Although - (Deep) Neural Networks are not a silver bullet
 - Remember the Bayes error



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Although - (Deep) Neural Networks are not a silver bullet
 - Remember the **Bayes error**
 - Sometimes a linear regression (or Random Forest) is enough



Areas of Use

- **Introduction**

- **Feed-Forward Neural Networks**

- Feed-Forward Neural Networks
- Output Units
- Hidden Units
- Architecture design

- **Regularization**

- Norm penalty
- Data augmentation
- Multi-Task Learning
- Early Stopping
- Drop-out

- **Optimization of Neural Networks**

- Difficulties in Neural Network Optimization
- Local minima
- Plateaus and Saddle Points
- Cliffs, exploding and vanishing gradients
- Parameter initialization

- **Neural Networks in Practice**

- **Supervised learning**
- **Self-Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**



Why and when neural nets?

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Learning **feature representations**
- Needs a lot of data to learn **complex representations** (image, text, audio)
- Good for sensor data (high-dimensional)



Why and when neural nets?

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Learning **feature representations**
- Needs a lot of data to learn **complex representations** (image, text, audio)
- Good for sensor data (high-dimensional)
- When should we **not** use neural networks?



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Learning Representations

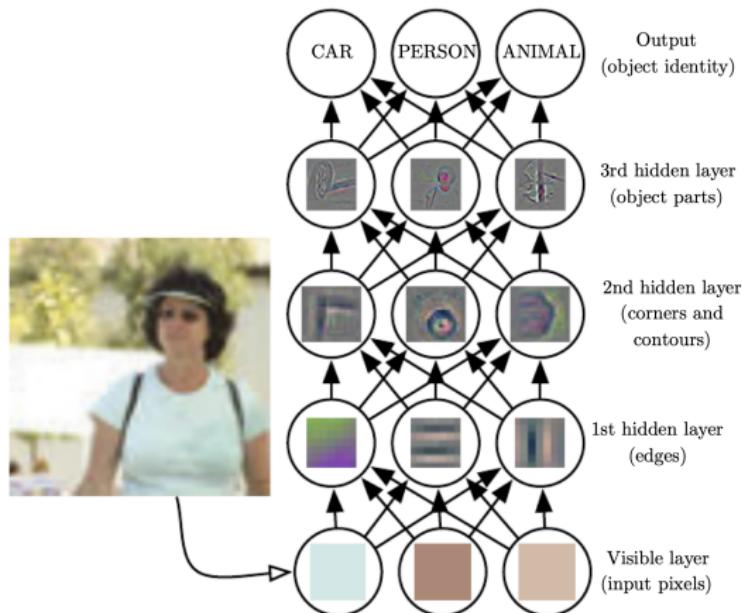


Figure: Learning representations can be crucial (Goodfellow et al, 2017, Fig. 1.2)



Different Network Architectures for Different Purposes

- Introduction

- Feed-Forward Neural Networks

- Feed-Forward Neural Networks
- Output Units
- Hidden Units
- Architecture design

- Regularization

- Norm penalty
- Data augmentation
- Multi-Task Learning
- Early Stopping
- Drop-out

- Optimization of Neural Networks

- Difficulties in Neural Network Optimization
- Local minima
- Plateaus and Saddle Points
- Cliffs, exploding and vanishing gradients
- Parameter initialization

- Neural Networks in Practice

- Different networks for different purposes

- **Feed-Forward Neural Network:** Basic building block



Different Network Architectures for Different Purposes

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Different networks for different purposes
 - **Feed-Forward Neural Network:** Basic building block
 - **Convolutional Neural Networks:** Computer Vision



Different Network Architectures for Different Purposes

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Different networks for different purposes
 - **Feed-Forward Neural Network:** Basic building block
 - **Convolutional Neural Networks:** Computer Vision
 - **Transformer Neural Networks:** Text/Audio/Vision
 - **Diffusion Neural Networks:** Image generation



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Section 2

Feed-Forward Neural Networks



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

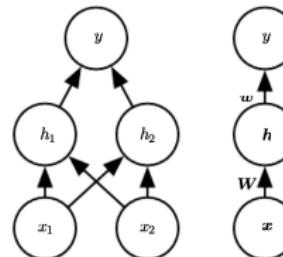


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

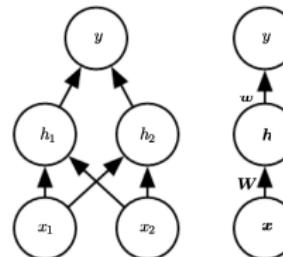


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers, neurons



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

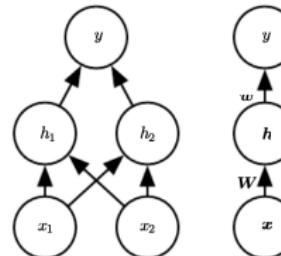


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers, neurons, input



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

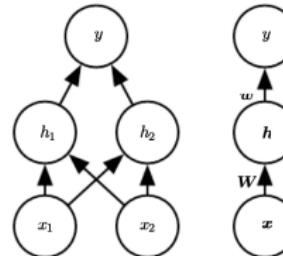


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers, neurons, input, output



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

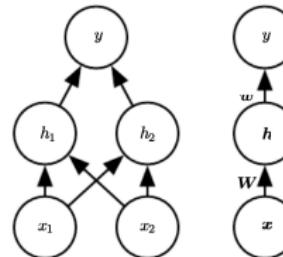


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers, neurons, input, output, weights



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

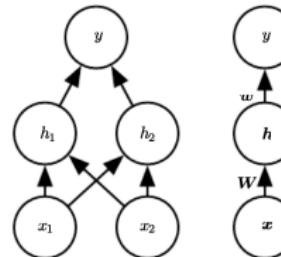


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers, neurons, input, output, weights, bias



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

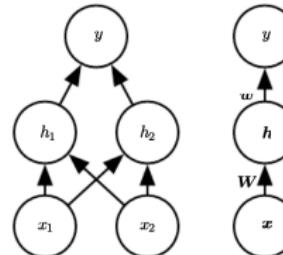


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

Important concepts:

Layers, neurons, input, output, weights, bias, architecture



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

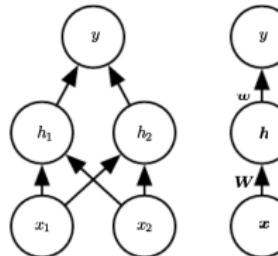


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit and unambiguous but for networks larger than this example it can consume too much space. (*Right*) In this style, we draw a node in the graph for each entire vector representing a layer's activations. This style is much more compact.

Figure: A simple feed-forward network (Goodfellow et al, 2017, Fig. 6.2)

In mathematical notation:

$$y_i = \mathbf{w}^T \underbrace{g(\mathbf{W}^T \mathbf{x}_i + \mathbf{b}_1)}_{\mathbf{h}} + \mathbf{b}_2$$



The Feed-Forward Network

$$y_i = \mathbf{w}^T g(\mathbf{W}^T \mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2$$

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice



The Feed-Forward Network

$$y_i = \mathbf{w}^T g(\mathbf{W}^T \mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2$$

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

$$\mathbf{W} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \mathbf{w} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \mathbf{b}_2 = (0), \mathbf{x}_i = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$



The Feed-Forward Network

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

$$y_i = \mathbf{w}^T g(\mathbf{W}^T \mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2$$

$$\mathbf{W} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \mathbf{w} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \mathbf{b}_2 = (0), \mathbf{x}_i = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

$$g(z) = \text{ReLU}(z) = \max(0, z)$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

$$y_i = \mathbf{w}^T g(\mathbf{W}^T \mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2$$

$$\mathbf{W} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \mathbf{w} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \mathbf{b}_2 = (0), \mathbf{x}_i = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

$$g(z) = \text{ReLU}(z) = \max(0, z)$$

$$\begin{aligned} y_i &= \begin{pmatrix} 1 \\ -2 \end{pmatrix}^T \text{ReLU} \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right) + (0) \\ &= \begin{pmatrix} 1 \\ -2 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} + (0) = 1 \end{aligned}$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The Feed-Forward Network

A feed-forward network for one observation (x_i).

$$\underbrace{\mathbf{h}_1}_{1 \times k_1} = g_1(\underbrace{\mathbf{x}^T}_{1 \times p} \underbrace{\mathbf{W}_1}_{p \times k_1} + \underbrace{\mathbf{b}_1}_{1 \times k_1})$$

⋮

$$\underbrace{\mathbf{h}_I}_{1 \times k_I} = g_I(\underbrace{\mathbf{h}_{I-1}^T}_{1 \times k_{I-1}} \underbrace{\mathbf{W}_I}_{k_{I-1} \times k_I} + \underbrace{\mathbf{b}_I}_{1 \times k_I})$$

⋮

$$\underbrace{\hat{\mathbf{y}}}_{1 \times m} = g_L(\underbrace{\mathbf{h}_{L-1}^T}_{1 \times k_{L-1}} \underbrace{\mathbf{W}_L}_{k_{L-1} \times m} + \underbrace{\mathbf{b}_L}_{1 \times m})$$

$$\hat{y} = f_L(f_{L-1}(\dots f_1(x)\dots))$$



Output units (g_L)

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Depend on the data y



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Output units (g_L)

- Depend on the data y
- Linear units for regression

$$\hat{y} = \mathbf{w}\mathbf{h} + \mathbf{b}$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Output units (g_L)

- Depend on the data y
- Linear units for regression

$$\hat{y} = \mathbf{w}\mathbf{h} + \mathbf{b}$$

- Bernoulli units for binary classification

$$\hat{y} = \sigma(\mathbf{w}\mathbf{h} + \mathbf{b}),$$

where

$$\sigma(z) = \frac{1}{1 + \exp(-z)},$$

i.e. the **logistic** or **sigmoid** function.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Output units (g_L)

- Multinoulli units for multi-class classification

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{w}\mathbf{h} + \mathbf{b}),$$

where

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Output units (g_L)

- Multinoulli units for multi-class classification

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{w}\mathbf{h} + \mathbf{b}),$$

where

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

- Other situations: Use the **negative log-likelihood**, e.g. Poisson, ordinal logit, survival models



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - **Hidden Units**
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Sometimes use notation σ as in $\sigma(Wh + b)$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - **Hidden Units**
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Activation functions (g_I)

- Sometimes use notation σ as in $\sigma(Wh + b)$
- Historically $g(z)$ has been the **sigmoid** or or hyperbolic tangent (\tanh)

$$g_{\text{sigmoid}}(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

$$g_{\tanh}(z) = \frac{\sinh z}{\cosh z} = \frac{e^{2z} - 1}{e^{2z} + 1}$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - **Hidden Units**
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Activation functions (g_I)

- Sometimes use notation σ as in $\sigma(Wh + b)$
- Historically $g(z)$ has been the **sigmoid** or or hyperbolic tangent (\tanh)

$$g_{\text{sigmoid}}(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

$$g_{\tanh}(z) = \frac{\sinh z}{\cosh z} = \frac{e^{2z} - 1}{e^{2z} + 1}$$

- Today, variants of Rectified linear unit (ReLU) is common
 - Easier to estimate with SGD
 - Easier for deep models



Activation functions (g_I): ReLU

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - **Hidden Units**
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

$$g_{\text{ReLU}}(z) = \max(0, z)$$

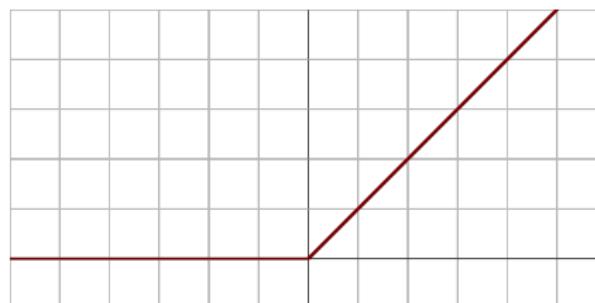


Figure: Rectified Linear Unit (Wikipedia)



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - **Hidden Units**
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Activation functions (g_I): GeLU

$$g_{\text{GeLU}}(z) = z\Phi(z)$$

where $\Phi(z)$ is a standard Gaussian.

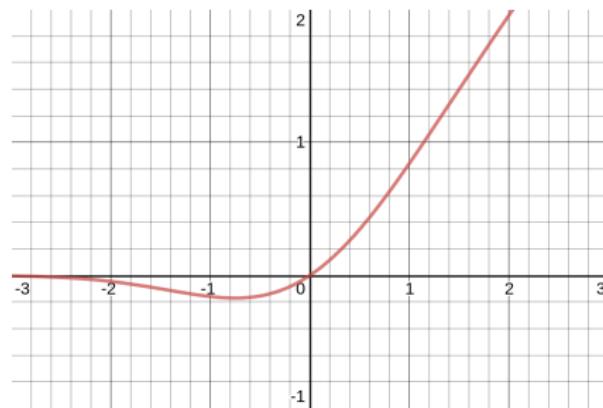


Figure: Gaussian Error Linear Unit (Wikipedia)



UPPSALA UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - **Hidden Units**
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Activation functions (g_I): ELU

$$g_{\text{ELU}}(z) = \begin{cases} \alpha(e^z - 1), & \text{if } z < 0, \\ z, & \text{if } z \geq 0. \end{cases}$$

where α is commonly set to 1.

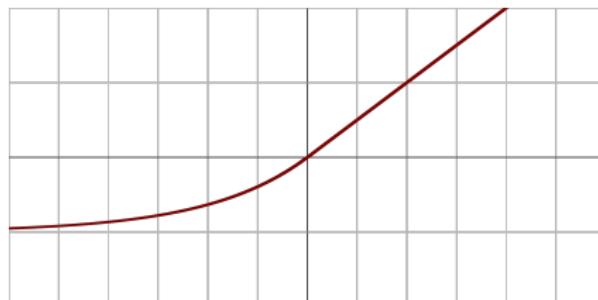


Figure: Exponential Linear Unit (Wikipedia)



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Architecture: the overall structure of the network
 - Choices:
 - How many layers?
 - How many hidden units in each layer?
 - Activation functions?



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Architecture: the overall structure of the network
 - Choices:
 - How many layers?
 - How many hidden units in each layer?
 - Activation functions?
 - *Note!* Output units is defined by both the **data** and by design



UPPSALA UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Universal Approximation Theorem

- A feed-forward neural network with **one hidden layer** and a **non-polynomial activation** (e.g., sigmoid, tanh, ReLU) can approximate **any continuous function** on a compact set $K \subset \mathbb{R}^n$.



UPPSALA UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Universal Approximation Theorem

- A feed-forward neural network with **one hidden layer** and a **non-polynomial activation** (e.g., sigmoid, tanh, ReLU) can approximate **any continuous function** on a compact set $K \subset \mathbb{R}^n$.
- Formally, for any continuous function $f : K \rightarrow \mathbb{R}^m$ and any $\varepsilon > 0$, there exists a one-hidden-layer network

$$g(x) = C \sigma(Ax + b)$$

such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon.$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Universal Approximation Theorem

- A feed-forward neural network with **one hidden layer** and a **non-polynomial activation** (e.g., sigmoid, tanh, ReLU) can approximate **any continuous function** on a compact set $K \subset \mathbb{R}^n$.
- Formally, for any continuous function $f : K \rightarrow \mathbb{R}^m$ and any $\varepsilon > 0$, there exists a one-hidden-layer network

$$g(x) = C \sigma(Ax + b)$$

such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon.$$

- Interpretation: The **worst-case error** between the network g and the target function f on the entire set K can be made **arbitrarily small** by using enough hidden units.



Universal Approximation Theorem, but...

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Neural networks are **universal function approximators**, but...



Universal Approximation Theorem, but...

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Neural networks are **universal function approximators**, but...
 - No guarantee we can **learn** the network



Universal Approximation Theorem, but...

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Neural networks are **universal function approximators**, but...
 - No guarantee we can **learn** the network
 - No guarantee that it will **generalize**



Universal Approximation Theorem, but...

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Neural networks are **universal function approximators**, but...
 - No guarantee we can **learn** the network
 - No guarantee that it will **generalize**
 - No indication of how large the network needs to be



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Depth matters

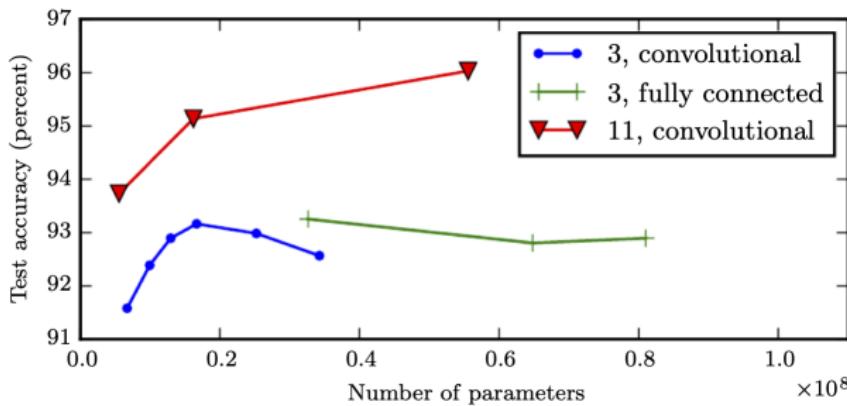


Figure: Effect of depth on accuracy (Goodfellow et al, 2017, Fig. 6.3)



UPPSALA UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Depth matters II

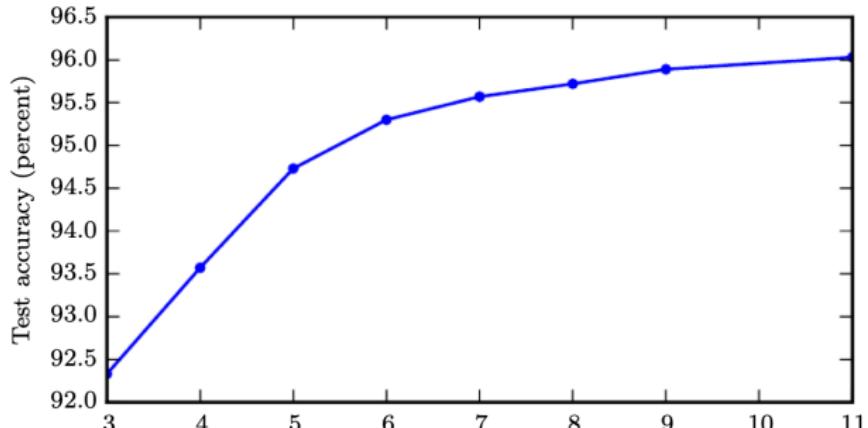


Figure: Depth vs. no of parameters (Goodfellow et al, 2017, Fig. 6.6)



How to choose architecture

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Trial and error on validation sets
- Art rather than science, however...



How to choose architecture

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Trial and error on validation sets
- Art rather than science, however...
 1. Start: get training error to zero



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

How to choose architecture

- Trial and error on validation sets
- Art rather than science, however...
 1. **Start**: get training error to zero
 2. **Regularize**: get validation error to minimum
- Specialized approaches (Bayesian optimization)



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Trial and error on validation sets
 - Art rather than science, however...
 1. **Start:** get training error to zero
 2. **Regularize:** get validation error to minimum
 - Specialized approaches (Bayesian optimization)
 - Grid search (combinatorial explosion)
 - Really bad if many hyperparameters has a little effect
 - If we have 5 irrelevant parameters we try 3 values for: 125 training per relevant run
 - Instead use...



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Trial and error on validation sets
 - Art rather than science, however...
 1. **Start:** get training error to zero
 2. **Regularize:** get validation error to minimum
 - Specialized approaches (Bayesian optimization)
 - Grid search (combinatorial explosion)
 - Really bad if many hyperparameters has a little effect
 - If we have 5 irrelevant parameters we try 3 values for: 125 training per relevant run
 - Instead use... **Random search**



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Grid search vs. Random Search

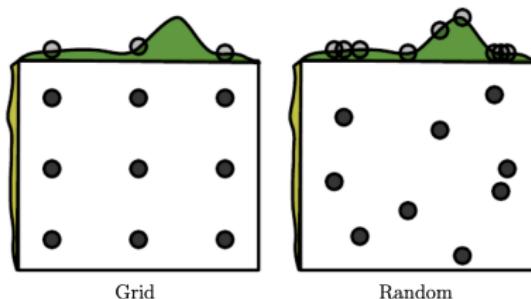


Figure: Grid search and random search (Goodfellow et al, 2017, Fig. 11.2)



UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Section 3

Regularization



Regularization of Neural Networks

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice
 - Reduce training error but **improve generalization error**



Regularization of Neural Networks

- Introduction

- Feed-Forward Neural Networks

- Feed-Forward Neural Networks
- Output Units
- Hidden Units
- Architecture design

- Regularization

- Norm penalty
- Data augmentation
- Multi-Task Learning
- Early Stopping
- Drop-out

- Optimization of Neural Networks

- Difficulties in Neural Network Optimization
- Local minima
- Plateaus and Saddle Points
- Cliffs, exploding and vanishing gradients
- Parameter initialization

- Neural Networks in Practice

- Reduce training error but **improve generalization error**
- Neural Networks are extremely flexible / high model capacity



Regularization of Neural Networks

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Reduce training error but **improve generalization error**
- Neural Networks are extremely flexible / high model capacity
- Regularization is important for good generalizability



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Norm penalty

- Let

$$\tilde{J}(\theta) = J(\theta) + \alpha\Omega(\theta),$$

where $J(\theta)$ is the cost function, θ all weights and biases, and $\alpha\Omega(\theta)$ is the penalty for the weight matrices and where α is the strength of the penalty.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Norm penalty

- Let

$$\Omega_1(\theta) = \sum_i \sum_j |\theta|_{i,j},$$

and

$$\Omega_2(\theta) = \sum_i \sum_j \theta_{i,j}^2,$$

be the L_1 and L_2 regularization respectively.

- We can then get the cost function

$$\tilde{J}(\theta) = J(\theta) + \sum_l \alpha_l \Omega_2(\theta_l),$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Weight decay

- Let's define the cost function for the weights w as

$$\begin{aligned}\tilde{J}(w) &= J(w) + \alpha\Omega_2(w) \\ &= J(w) + \alpha w^T w\end{aligned}$$

- Then the gradient update becomes

$$\nabla_w \tilde{J}(w) = \nabla_w J(w) + 2\alpha w$$

- To update our weights with gradient descent

$$w \leftarrow w - \epsilon(\nabla_w J(w) + 2\alpha w)$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Weight decay

- Let's define the cost function for the weights w as

$$\begin{aligned}\tilde{J}(w) &= J(w) + \alpha\Omega_2(w) \\ &= J(w) + \alpha w^T w\end{aligned}$$

- Then the gradient update becomes

$$\nabla_w \tilde{J}(w) = \nabla_w J(w) + 2\alpha w$$

- To update our weights with gradient descent

$$w \leftarrow w - \epsilon(\nabla_w J(w) + 2\alpha w)$$

$$w \leftarrow (1 - 2\alpha\epsilon)w - \epsilon\nabla_w J(w)$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Weight decay / Norm penalty

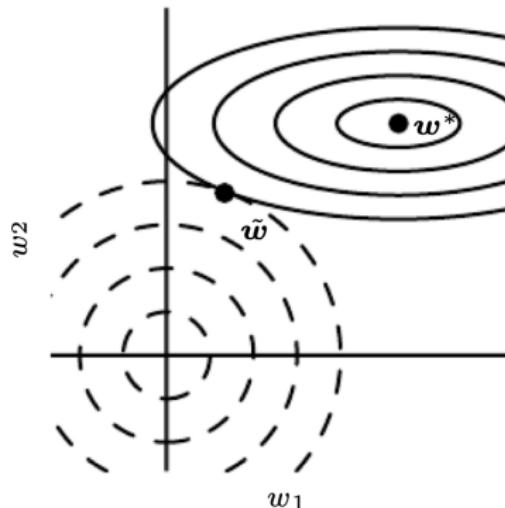


Figure: L_2 regularization (Goodfellow et al, 2017, Fig. 7.1)



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Dataset augmentation



Figure: Data Augmentation (Chollet and Allair, 2018, Fig 5.10)

- "Modify" data or inject noise
- Common in Convolutional neural networks



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Multi-Task Learning

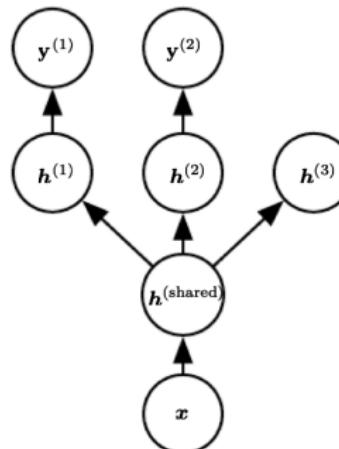


Figure: Multi-task learning (Goodfellow et al., 2017, Fig 7.2)

- Common in Pre-Trained Language Models (Transformers)



Early Stopping

- Stop optimization early based on validation error

- Introduction

- Feed-Forward Neural Networks

- Feed-Forward Neural Networks
- Output Units
- Hidden Units
- Architecture design

- Regularization

- Norm penalty
- Data augmentation
- Multi-Task Learning
- Early Stopping
- Drop-out

- Optimization of Neural Networks

- Difficulties in Neural Network Optimization
- Local minima
- Plateaus and Saddle Points
- Cliffs, exploding and vanishing gradients
- Parameter initialization

- Neural Networks in Practice



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Early Stopping

- Stop optimization early based on validation error
- Use the best iteration (hyperparameter)

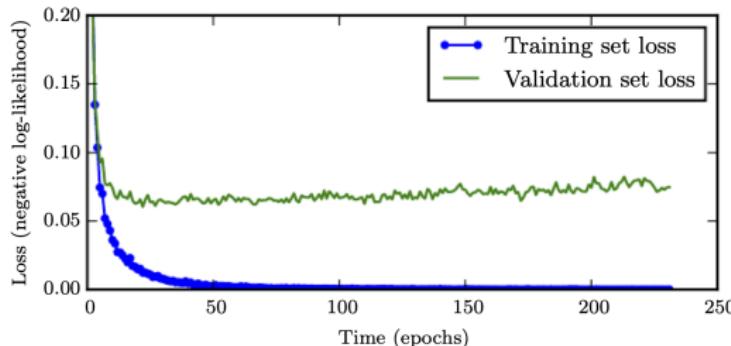


Figure: Early Stopping (Goodfellow et al, 2017, Fig. 7.3)

- Can be shown to be equivalent (under strict assumptions) to L_2 regularization (see Goodfellow et al, 2017)
- Efficient regularization



Early Stopping

UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

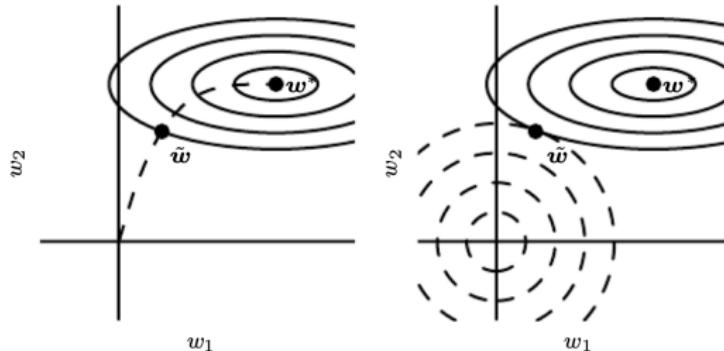


Figure: Early Stopping (Goodfellow et al, 2017, Fig. 7.4)



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- In each iteration:
 - Sample an indicator I_i for each node i
 - Set the value h_i to 0 with probability p
 - The dropout probability (dropout rate) is typically 0.2 for input nodes and 0.5 for hidden nodes
 - Forces the network to
 - not rely on individual nodes
 - spread out the weights over more nodes
 - Can be seen as an ensemble method



Dropout

UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - **Drop-out**
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

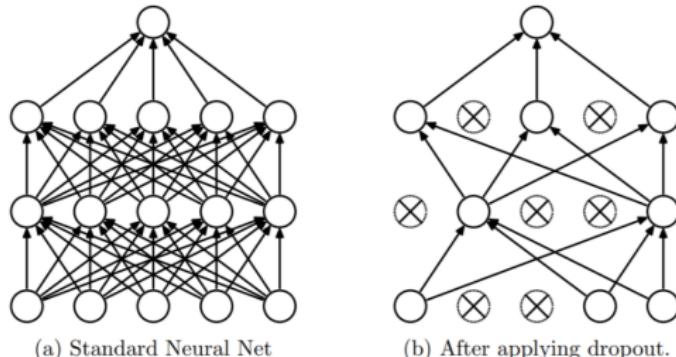


Figure 1: Dropout Neural Net Model. **Left:** A standard neural net with 2 hidden layers. **Right:** An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Figure: Dropout (Srivastava et al, 2014)



But...

UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The best regularizer is...



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

The best regularizer is... to get more data.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Section 4

Optimization of Neural Networks



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice
- Usually, a lot of data and many parameters ($\theta = (W, b)$)



Neural Network Learning

UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Usually, a lot of data and many parameters ($\theta = (W, b)$)
- We usually minimize our training cost function

$$J(\theta) = \sum_i^N L(\text{NN}(x_i|\theta), y_i) + \Omega(\theta),$$

where L is the observation level loss, $\text{NN}()$ is our neural network and Ω is the regularization term.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Neural Network Learning

- Usually, a lot of data and many parameters ($\theta = (W, b)$)
- We usually minimize our training cost function

$$J(\theta) = \sum_i^N L(\text{NN}(x_i|\theta), y_i) + \Omega(\theta),$$

where L is the observation level loss, $\text{NN}()$ is our neural network and Ω is the regularization term.

- **Learning Target:** Find $\hat{\theta}$ that minimize the **generalization error**



Optimization of Neural Networks

UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Stochastic Gradient Descent methods

$$\theta_t = \theta_{t-1} - \eta_t \hat{\nabla} J(\theta_{t-1}),$$

where η_t denotes the learning rate at iteration t and $\hat{\nabla} J(\theta_{t-1})$ is an estimate of the gradient of the objective function.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- Stochastic Gradient Descent methods

$$\theta_t = \theta_{t-1} - \eta_t \hat{\nabla} J(\theta_{t-1}),$$

where η_t denotes the learning rate at iteration t and $\hat{\nabla} J(\theta_{t-1})$ is an estimate of the gradient of the objective function.

- In practice, better optimizers are used:
 - Adam (the main optimizer today)
 - RMSprop



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Optimization of Neural Networks

- Stochastic Gradient Descent methods

$$\theta_t = \theta_{t-1} - \eta_t \hat{\nabla} J(\theta_{t-1}),$$

where η_t denotes the learning rate at iteration t and $\hat{\nabla} J(\theta_{t-1})$ is an estimate of the gradient of the objective function.

- In practice, better optimizers are used:
 - Adam (the main optimizer today)
 - RMSprop
- To compute $\hat{\nabla} J(\theta_{t-1})$:
 - **Backpropagation** algorithm (chain-rule for derivatives)



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Backpropagation

- Backpropagation provides an efficient procedure for computing all parameter gradients $\nabla J(\theta)$ in a feed-forward neural network.
- Uses the chain rule to propagate the error backward through layers:

$$\frac{\partial J}{\partial W_l} = h_{l-1}^\top \delta_l, \quad \frac{\partial J}{\partial b_l} = \delta_l,$$

where δ_l is the error signal at layer l .

- The algorithm consists of two passes:
 - **Forward pass:** compute

$$h_l = g_l(h_{l-1}^\top W_l + b_l)$$

for each layer l .

- **Backward pass:** compute error signals

$$\delta_l = (\delta_{l+1} W_{l+1}^\top) \odot g'_l(z_l), \quad z_l = h_{l-1}^\top W_l + b_l,$$

starting from the output layer.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Backpropagation

- Backpropagation provides an efficient procedure for computing all parameter gradients $\nabla J(\theta)$ in a feed-forward neural network.
- Uses the chain rule to propagate the error backward through layers:

$$\frac{\partial J}{\partial W_l} = h_{l-1}^\top \delta_l, \quad \frac{\partial J}{\partial b_l} = \delta_l,$$

where δ_l is the error signal at layer l .

- The algorithm consists of two passes:
 - **Forward pass:** compute

$$h_l = g_l(h_{l-1}^\top W_l + b_l)$$

for each layer l .

- **Backward pass:** compute error signals

$$\delta_l = (\delta_{l+1} W_{l+1}^\top) \odot g'_l(z_l), \quad z_l = h_{l-1}^\top W_l + b_l,$$

starting from the output layer.

- Enables efficient gradient-based optimization even for networks with many layers and millions of parameters.



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Forward Pass: Simple Example

- Consider a network with one hidden layer (2 units, we skip biases for simplicity) and ReLU activation:

$$h_1 = g_1(x^\top W_1), \quad \hat{y} = w^\top h_1.$$

- Let

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad W_1 = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}, \quad w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

- Compute hidden pre-activation:

$$z_1 = x^\top W_1 = \begin{bmatrix} -1 \\ 4 \end{bmatrix}.$$

- Apply ReLU:

$$h_1 = g_{\text{ReLU}}(z_1) = \max(0, z_1) = \begin{bmatrix} 0 \\ 4 \end{bmatrix}.$$

- Output:

$$\hat{y} = w^\top h_1 = -8.$$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Backward Pass: Simple Example

- Loss (squared error):

$$J = \frac{1}{2}(\hat{y} - y)^2, \quad y = 0, \quad \frac{\partial J}{\partial \hat{y}} = \hat{y} - y = -8.$$

- Output layer gradients:

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w} = h_1(\hat{y} - y) = \begin{bmatrix} 0 \\ -40 \end{bmatrix}.$$

- Backpropagated error signal:

$$\delta_1 := \frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_1} = (\hat{y} - y) w \odot g'_{\text{ReLU}}(z_1).$$

$$(\hat{y} - y) w = \begin{bmatrix} -8 \\ 16 \end{bmatrix}, \quad g'_{\text{ReLU}}(z_1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \delta_1 = \begin{bmatrix} 0 \\ 16 \end{bmatrix}.$$

- Gradients for first layer:

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial W_1} = x \delta_1^\top = \begin{bmatrix} 0 & 16 \\ 0 & 32 \end{bmatrix}.$$



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Neural Networks cost function $J(\theta)$ are (usually) **not a convex** function
 - We can have local minima
 - When will this be a problem?



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Neural Networks cost function $J(\theta)$ are (usually) **not a convex** function
 - We can have local minima
 - **When will this be a problem?**
 - A problem if local optima has a **high cost/bad performance**



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Local Minima

- Neural Networks cost function $J(\theta)$ are (usually) **not a convex** function
- We can have local minima
- **When will this be a problem?**
- A problem if local optima has a **high cost/bad performance**
- **Good thing:** This is probably not the situation!



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Neural Networks cost function $J(\theta)$ are (usually) **not a convex** function
 - We can have local minima
 - **When will this be a problem?**
 - A problem if local optima has a **high cost/bad performance**
 - **Good thing:** This is probably not the situation!
 - Many practitioners believe this is a problem



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Neural Networks cost function $J(\theta)$ are (usually) **not a convex** function
 - We can have local minima
 - **When will this be a problem?**
 - A problem if local optima has a **high cost/bad performance**
 - **Good thing:** This is probably not the situation!
 - Many practitioners believe this is a problem
 - **Diagnosis:** Plot the gradient norm, $\mathbf{g}^T \mathbf{g}$, where $\mathbf{g} = \hat{\nabla} J(\theta)$



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- Neural Networks cost function $J(\theta)$ are (usually) **not a convex** function
 - We can have local minima
 - **When will this be a problem?**
 - A problem if local optima has a **high cost/bad performance**
 - **Good thing:** This is probably not the situation!
 - Many practitioners believe this is a problem
 - **Diagnosis:** Plot the gradient norm, $\mathbf{g}^T \mathbf{g}$, where $\mathbf{g} = \hat{\nabla} J(\theta)$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Plateaus and Saddle Points

- Another problem is **saddle points**

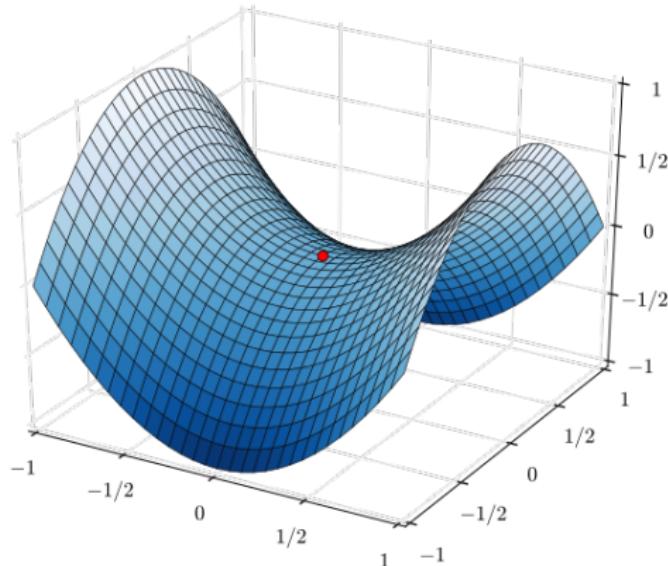


Figure: Saddle point for $z = x^2 - y^2$ (Wikipedia)

- What is the gradient in a saddle point?



Problems: Plateaus and Saddle Points

- In a saddle point the Hessian is **indefinite**, both positive and negative eigenvalues

- Introduction

- Feed-Forward Neural Networks

- Feed-Forward Neural Networks
- Output Units
- Hidden Units
- Architecture design

- Regularization

- Norm penalty
- Data augmentation
- Multi-Task Learning
- Early Stopping
- Drop-out

- Optimization of Neural Networks

- Difficulties in Neural Network Optimization
- Local minima
- **Plateaus and Saddle Points**
- Cliffs, exploding and vanishing gradients
- Parameter initialization

- Neural Networks in Practice



UPPSALA UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Plateaus and Saddle Points

- In a saddle point the Hessian is **indefinite**, both positive and negative eigenvalues
- A local minimum: Hessian is **positive definite**, only positive eigen values



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- In a saddle point the Hessian is **indefinite**, both positive and negative eigenvalues
 - A local minimum: Hessian is **positive definite**, only positive eigen values
 - In random functions of **high dimension**: **most points are saddle points**
 - **Intuition:** In random functions the sign of the eigen values of the Hessian is random



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
-
- In a saddle point the Hessian is **indefinite**, both positive and negative eigenvalues
 - A local minimum: Hessian is **positive definite**, only positive eigen values
 - In random functions of **high dimension**: **most points are saddle points**
 - **Intuition:** In random functions the sign of the eigen values of the Hessian is random
 - In random functions: eigen values become more positive in regions of lower cost. I.e. local minima more probable in low cost regions. Saddle points more common in high-cost regions



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Plateaus and Saddle Points

- In a saddle point the Hessian is **indefinite**, both positive and negative eigenvalues
- A local minimum: Hessian is **positive definite**, only positive eigen values
- In random functions of **high dimension**: **most points are saddle points**
- **Intuition:** In random functions the sign of the eigen values of the Hessian is random
- In random functions: eigen values become more positive in regions of lower cost. I.e. local minima more probable in low cost regions. Saddle points more common in high-cost regions
- The problem of saddle points:
 - Probably a reason why second order methods (using the Hessian) has not succeed
 - Empirically, (stochastic) gradient descent seem to escape saddle points



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - **Cliffs, exploding and vanishing gradients**
 - Parameter initialization
- Neural Networks in Practice

Problems: Cliffs

- Another problem is "**cliffs**" or large changes in gradients

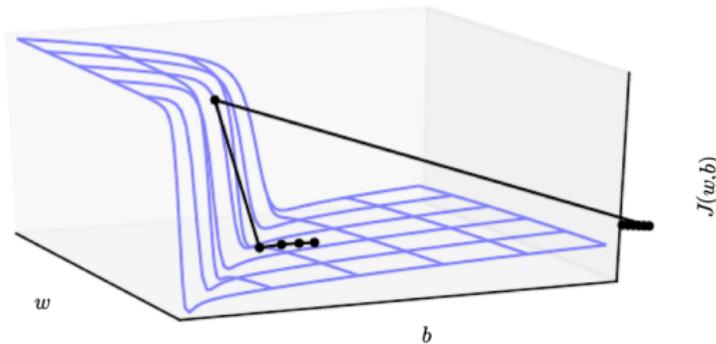


Figure: Cliff (Goodfellow et al., 2017, Fig. 8.3)

- Can undo many iterative steps
- Common in **Recurrent neural networks**
- **Mitigation:** Gradient clipping



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Exploding and vanishing gradients

- In deep neural networks, gradients can vanish or explode
- As an example: We want to compute the gradient for a situation where the weights are multiplied t times. Then using eigendecomposition

$$W^t = V \text{diag}(\lambda)^t V^T$$

- The gradient is scaled wrt $\text{diag}(\lambda)^t$



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Exploding and vanishing gradients

- In deep neural networks, gradients can **vanish or explode**
- As an example: We want to compute the gradient for a situation where the weights are multiplied t times. Then using eigendecomposition

$$W^t = V \text{diag}(\lambda)^t V^T$$

- The gradient is scaled wrt $\text{diag}(\lambda)^t$
- Cliffs is an example of gradient explosion
- **Mitigation for gradient explosion:** Gradient clipping



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Problems: Exploding and vanishing gradients

- In deep neural networks, gradients can **vanish or explode**
- As an example: We want to compute the gradient for a situation where the weights are multiplied t times. Then using eigendecomposition

$$W^t = V \text{diag}(\lambda)^t V^T$$

- The gradient is scaled wrt $\text{diag}(\lambda)^t$
- Cliffs is an example of gradient explosion
- **Mitigation for gradient explosion:** Gradient clipping
- Common problem in **Recurrent neural networks**



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- We need to have **starting values** for gradient descent



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

- We need to have **starting values** for gradient descent
- Initialization can be seen as a hyperparameter



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Initial values

- We need to have **starting values** for gradient descent
- Initialization can be seen as a hyperparameter
- Bad initial values might
 - Bad convergence (local optimum)
 - Numerical problems



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- We need to have **starting values** for gradient descent
 - Initialization can be seen as a hyperparameter
 - Bad initial values might
 - Bad convergence (local optimum)
 - Numerical problems
 - We want to **break symmetry** between layers
 - Otherwise the same units will be updated in the same way (deterministically)



- Introduction
 - Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
 - Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
 - Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
 - Neural Networks in Practice
- We need to have **starting values** for gradient descent
 - Initialization can be seen as a hyperparameter
 - Bad initial values might
 - Bad convergence (local optimum)
 - Numerical problems
 - We want to **break symmetry** between layers
 - Otherwise the same units will be updated in the same way (deterministically)
 - Good practice
 - Initialize values randomly close to zero (uniform or normal)



UPPSALA
UNIVERSITET

- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Section 5

Neural Networks in Practice



- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

TensorFlow

- Framework for large-scale machine learning and Neural Networks
- Developed by Google
- Can be used both from R and Python
- Used in both research and production
- What TensorFlow does:
 - Computing gradients (autodiff) for Neural Networks
 - Enable use of graphical processing units (GPU) and Tensor processing Units (TPU)
 - Enable training using common optimizers (such as Adam, RMSprop)
- TensorFlow Probability is a probabilistic programming framework using TF





- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

(Py)Torch

- Similar to TensorFlow
- Developed by Meta AI
- Can be used both from R and Python
- Used in both research and production
- **pyro** is a probabilistic programming framework using torch





- Introduction
- Feed-Forward Neural Networks
 - Feed-Forward Neural Networks
 - Output Units
 - Hidden Units
 - Architecture design
- Regularization
 - Norm penalty
 - Data augmentation
 - Multi-Task Learning
 - Early Stopping
 - Drop-out
- Optimization of Neural Networks
 - Difficulties in Neural Network Optimization
 - Local minima
 - Plateaus and Saddle Points
 - Cliffs, exploding and vanishing gradients
 - Parameter initialization
- Neural Networks in Practice

Keras

- Syntax for 'building' Neural Networks
- Available both in R and Python
- TensorFlow or Torch as backend

