



UPPSALA  
UNIVERSITET

# Machine learning – Block 3

Måns Magnusson  
Department of Statistics, Uppsala University

Autumn 2022

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice



# Evaluation assignment 2

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

● t



# This week's lecture

---

- **Practicalities**
- **Introduction**
- **Feed-Forward Neural Networks**
  - Feed-Forward Neural Networks
  - Hyper-parameters
- **Regularization**
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- **Optimization of Neural Networks**
  - 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**



UPPSALA  
UNIVERSITET

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Section 2

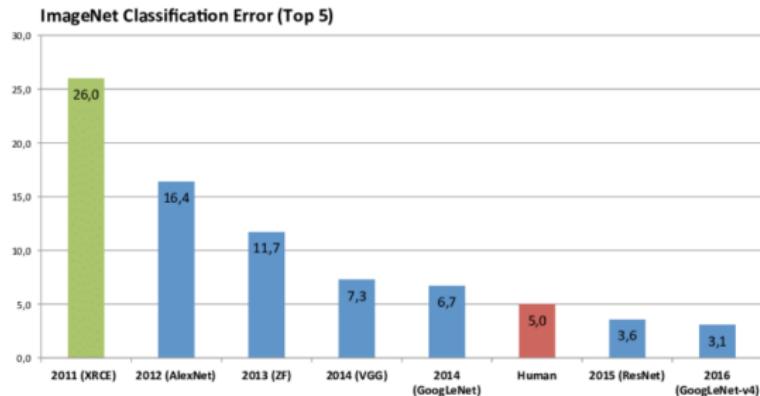
### Introduction



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# The Hype: Computer Vision

Figure: ImageNet performance (Roessler, 2019)





- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# The Hype: Speech Recognition

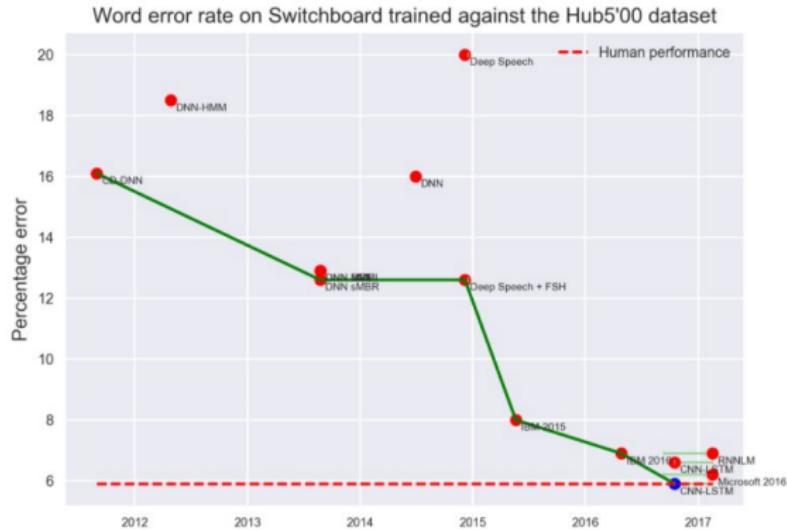


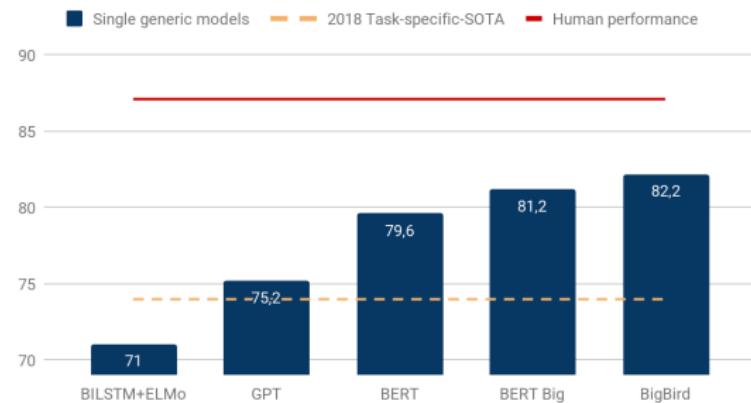
Figure: Speech recognition performance (source:  
<https://eff.org/ai/metrics>)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# The Hype: Natural Language Processing

GLUE scores evolution over 2018-2019



**Figure:** General Language Understanding (source:  
<https://www.programmersought.com/article/4251948498/>)

Work is very much ongoing:

<https://gluebenchmark.com/leaderboard>



# The Hype

---

- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - Local minima
    - Plateaus and Saddle Points
    - Cliffs, exploding and vanishing gradients
    - Parameter initialization
  - Neural Networks in Practice
- Although - Neural Networks is not a silver bullet



# The Hype

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Although - Neural Networks is not a silver bullet
- Remember the Bayes error



# The Hype

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Although - Neural Networks is not a silver bullet
- Remember the **Bayes error**
- Some times a linear regression (or Random Forest) is enough



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Section 3

### Feed-Forward Neural Networks



# UPPSALA UNIVERSITET

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# The Feed-Forward Network

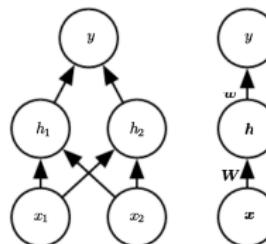


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*) In this style, we draw every unit as a node in the graph. This style is very explicit 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)

## Important concepts:

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



# Different Architectures for Different Purposes

---

- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - Local minima
    - Plateaus and Saddle Points
    - Cliffs, exploding and vanishing gradients
    - Parameter initialization
  - Neural Networks in Practice
- Different networks for different purposes
    - Convolutional Neural Networks: Computer Vision



# Different Architectures for Different Purposes

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice
  - Different networks for different purposes
    - Convolutional Neural Networks: Computer Vision
    - Recurrent Neural Networks: Speech Audio (?)



# Different Architectures for Different Purposes

---

- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - Local minima
    - Plateaus and Saddle Points
    - Cliffs, exploding and vanishing gradients
    - Parameter initialization
  - Neural Networks in Practice
- 
- Different networks for different purposes
    - Convolutional Neural Networks: Computer Vision
    - Recurrent Neural Networks: Speech Audio (?)
    - Transformers/Attention: Textual data
  - The Neural Network Zoo: <https://www.asimovinstitute.org/neural-network-zoo/>



# Areas of Use: All fields

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Supervised learning
- Unsupervised learning
- Reinforcement learning



# Why and when neural nets?

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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
- Good for sensor data (high-dimensional)



# Why and when neural nets?

---

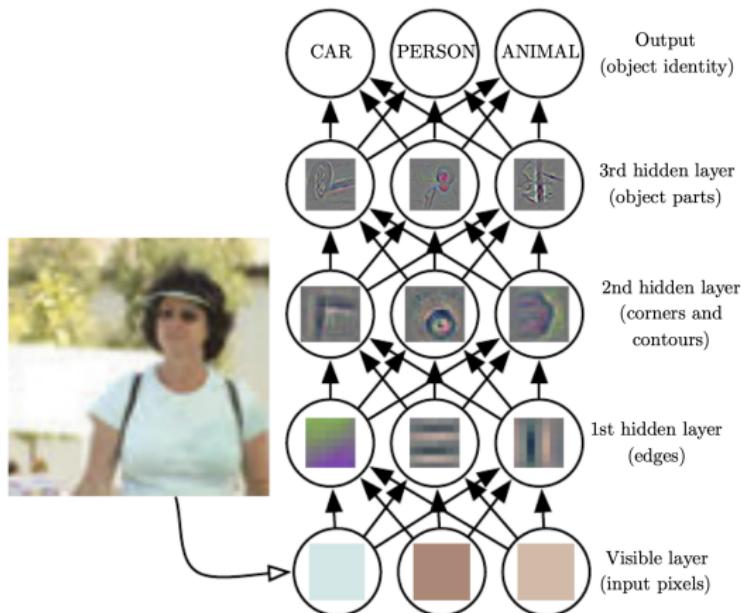
- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# The Feed-Forward Network

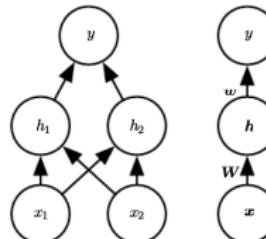


Figure 6.2: An example of a feedforward network, drawn in two different styles. Specifically, this is the feedforward network we use to solve the XOR example. It has a single hidden layer containing two units. (*Left*)In this style, we draw every unit as a node in the graph. This style is very explicit 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:

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



# The Feed-Forward Network

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

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

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

$$g(z) = \text{ReLU}(z) = \max(0, z), \mathbf{x}_i = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

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

$$y_i = \begin{pmatrix} 1 \\ -2 \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} + (0) = 1$$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 ( $x_i$ ).

$$\begin{aligned}\underbrace{\mathbf{h}_1}_{1 \times k_1} &= g_1(\underbrace{\mathbf{x}^T}_{1 \times p} \underbrace{\mathbf{W}_1}_{p \times k_1} + \underbrace{\mathbf{b}_1}_{1 \times k_1}) \\ &\vdots \\ \underbrace{\mathbf{h}_I}_{1 \times k_I} &= g_I(\underbrace{\mathbf{h}_{I-1}^T}_{1 \times k_{I-1}} \underbrace{\mathbf{W}_I}_{k_{I-1} \times k_I} + \underbrace{\mathbf{b}_I}_{1 \times k_I}) \\ &\vdots \\ \underbrace{\hat{\mathbf{y}}}_{1 \times m} &= g_L(\underbrace{\mathbf{h}_{L-1}^T}_{1 \times k_{L-1}} \underbrace{\mathbf{W}_L}_{k_{L-1} \times m} + \underbrace{\mathbf{b}_L}_{1 \times m})\end{aligned}$$
$$\hat{y} = f_L(f_{L-1}(\dots f_1(x)\dots))$$



# Activation functions ( $g_I$ )

---

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

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Activation functions ( $g_I$ )

---

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

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

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Activation functions ( $g_I$ )

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

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

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

- Now, usually variants of Rectified linear unit (ReLU)

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

- Easier to estimate with SGD
- Easier for deep models
- Last activation is the output function  $g_L$ , usually a softmax (if classification)

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^J e^{z_j}}$$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Activation functions ( $g_I$ )

---

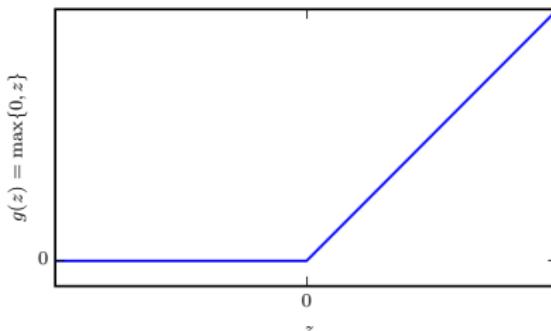


Figure: Rectified Linear Unit (Goodfellow et al, 2017, Fig. 6.3)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 a linear output layer and at least one hidden layer with any 'squashing' activation function can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units." (Goodfellow et al. 2017, p. 198)

- Also holds for ReLU
- No guarantee we can learn the network
- No guarantee that it will generalize
- No indication of how large the network need to be



# Hyper-parameters in feed-forward networks

---

- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - Local minima
    - Plateaus and Saddle Points
    - Cliffs, exploding and vanishing gradients
    - Parameter initialization
  - Neural Networks in Practice
- The number of layers
  - The number of neurons
  - Activation functions
  - The type of layers (CNN, MaxPooling, Multi-head attention)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# How to choose parameters

---

- Trial and error on validation sets
- Art rather than science
- Specialized approaches (Bayesian Optimization)



- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - 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
  - Specialized approaches (Bayesian Optimization)
  - Grid search (combinatorial explosion)
    - Really bad with many parameters with less effects
    - If we have 5 irrelevant parameters we try 3 values for:  $125$  training per relevant run
    - Instead use...



- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - 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
  - Specialized approaches (Bayesian Optimization)
  - Grid search (combinatorial explosion)
    - Really bad with many parameters with less effects
    - If we have 5 irrelevant parameters we try 3 values for:  $125$  training per relevant run
    - Instead use...
  - Random search



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Grid search vs. Random Search

---

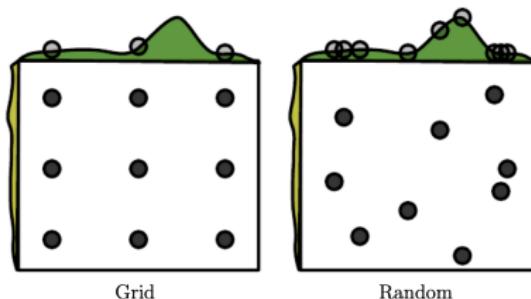


Figure: Grid search and random search (Goodfellow et al, 2017, Fig. 11.2)



UPPSALA  
UNIVERSITET

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Section 4

### Regularization



# Regularization of Neural Networks

---

- Practicalities
  - Introduction
  - Feed-Forward Neural Networks
    - Feed-Forward Neural Networks
    - Hyper-parameters
  - Regularization
    - Norm penalty
    - Data augmentation
    - Multi-Task Learning
  - Optimization of Neural Networks
    - Local minima
    - Plateaus and Saddle Points
    - Cliffs, exploding and vanishing gradients
    - Parameter initialization
  - Neural Networks in Practice
- Reduce training error but improve test/validation error



# Regularization of Neural Networks

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Reduce training error but improve test/validation error
- Neural Networks are extremely flexible / high model capacity



# Regularization of Neural Networks

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Reduce training error but improve test/validation error
- Neural Networks are extremely flexible / high model capacity
- Regularization is crucial for good generalizability



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Weight decay / Norm penalty

---

- Let

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

where  $J(W, b)$  is the cost function and  $\alpha\Omega(W)$  is the penalty for the weight matrices.

- $\alpha$  is the strength of the penalty.



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Weight decay / Norm penalty

---

- Let

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

and

$$\Omega_2(W) = \sum_i \sum_j w_{i,j}^2,$$

be the  $L_1$  and  $L_2$  regularization respectively.

- We can then get the cost function

$$\tilde{J}(W, b) = J(W, b) + \sum_I \alpha_I \Omega_2(W_I),$$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Weight decay / Norm penalty

---

- Lets define the cost function 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

$$\begin{aligned}w &\leftarrow w - \epsilon(\nabla_w J(w) + 2\alpha w) \\ w &\leftarrow (1 - 2\alpha\epsilon)w - \epsilon\nabla_w J(w)\end{aligned}$$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Weight decay / Norm penalty

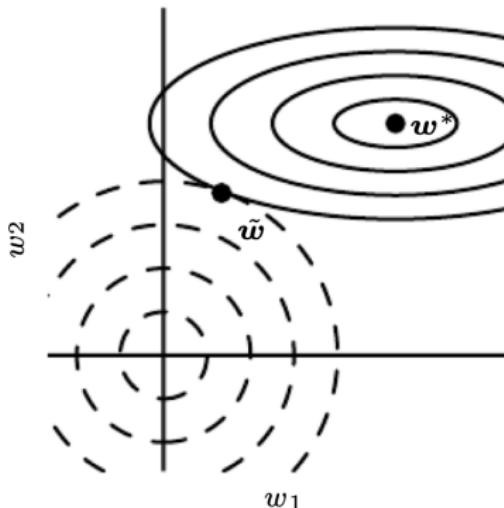


Figure:  $L_2$  regularization (Goodfellow et al, 2017, Fig. 7.1)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Multi-Task Learning

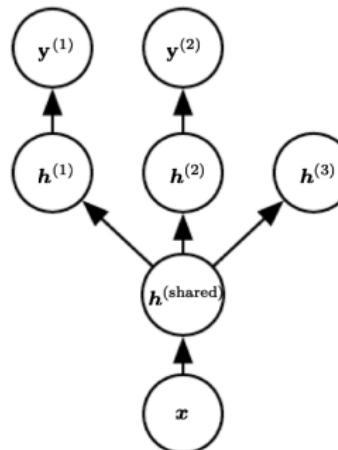


Figure: Multi-task learning (Goodfellow et al., 2017, Fig 7.2)

- Common in Large Language Models (Transformers)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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
- Rerun to that number of epochs (hyperparameter)
- Can be shown to be equivalent (under strict assumptions) to  $L_2$  regularization

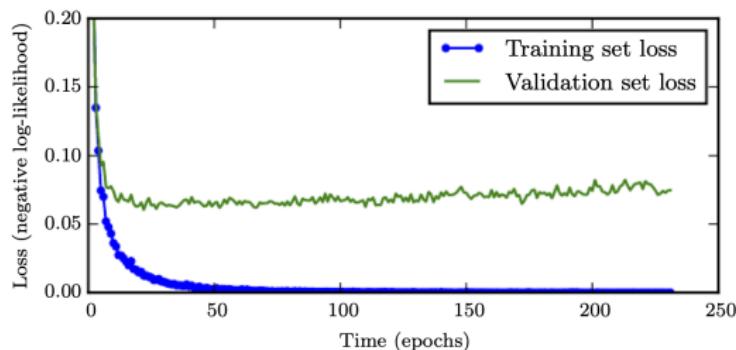


Figure: Early Stopping (Goodfellow et al, 2017, Fig. 7.3)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Early Stopping

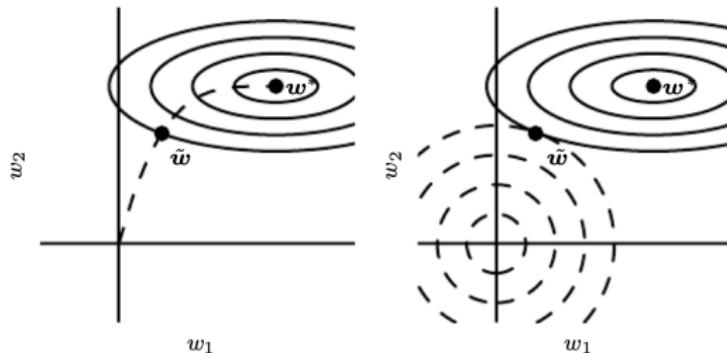


Figure: Early Stopping (Goodfellow et al, 2017, Fig. 7.4)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Dropout

---

- 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 is typically 0.8 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

# Dropout

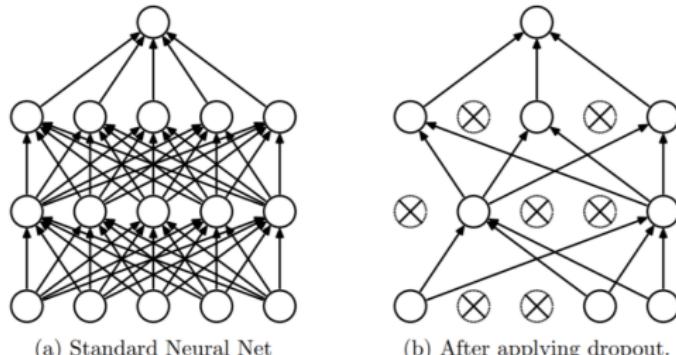


Figure 1: Dropout Neural Net Model. **Left:** A standard neural net with 2 hidden layers. **Right:** An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

**Figure: Dropout (Srivastava et al, 2014)**



# Other regularization techniques

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- In CNN: Dataset augmentation
- Get more data...



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Section 5

# Optimization of Neural Networks



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 ( $\theta = (W, b)$ )



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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

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

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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

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

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

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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

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

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

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Stochastic Gradient Descent methods

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

- In practice, better optimizers are used:
  - Adam
  - RMSprop



- Practicabilities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

- Stochastic Gradient Descent methods

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

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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?



# Problems: Local Minima

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**
- **Good thing:** This is probably not the situation!



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**
- **Good thing:** This is probably not the situation!
- Many practitioners believe this is a problem



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**
- **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  $\mathbf{g} = \hat{\nabla} J(\theta)$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**
- **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  $\mathbf{g} = \hat{\nabla} J(\theta)$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**

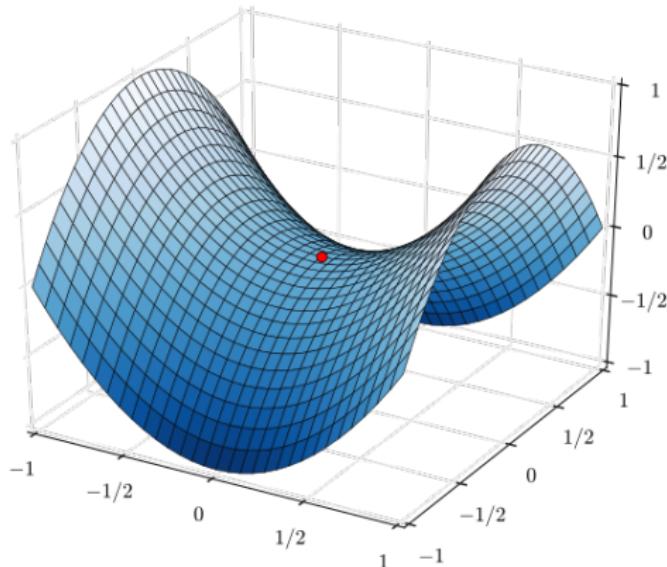


Figure: Saddle point for  $z = x^2 - y^2$  (Wikipedia)

- What is the gradient in a saddle point?



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**

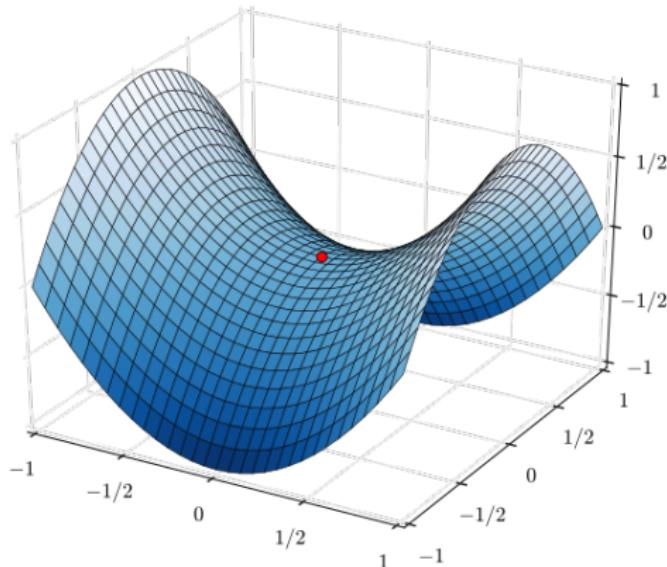


Figure: Saddle point for  $z = x^2 - y^2$  (Wikipedia)

- 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 eigen values

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - **Plateaus and Saddle Points**
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 eigen values
- A local minimum: Hessian is **positive definite**, only positive eigen values



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 eigen values
- 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 eigen values
- 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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 eigen values
- 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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

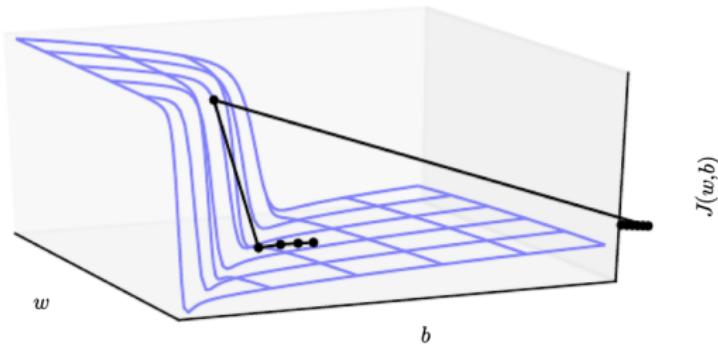


Figure: Cliff (Goodfellow et al., 2017, Fig. 8.3)

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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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  $\text{diag}(\lambda)^t$



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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**



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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



# Initial values

---

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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
- We want to **break symmetry** between layers
  - Otherwise the same units will be updated in the same way (deterministically)



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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
- 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

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - Local minima
  - Plateaus and Saddle Points
  - Cliffs, exploding and vanishing gradients
  - Parameter initialization
- Neural Networks in Practice

## Section 6

### Neural Networks in Practice



- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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





# UPPSALA UNIVERSITET

- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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





- Practicalities
- Introduction
- Feed-Forward Neural Networks
  - Feed-Forward Neural Networks
  - Hyper-parameters
- Regularization
  - Norm penalty
  - Data augmentation
  - Multi-Task Learning
- Optimization of Neural Networks
  - 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

