Deep Learning

Akka Ecole Technology

Overview

Three Day Overview

- First Day

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow
  - Apply Deep Learning to a Variety of Problems

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow
  - Apply Deep Learning to a Variety of Problems
- Second Day

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow
  - Apply Deep Learning to a Variety of Problems
- Second Day
  - Introduction to the Basic Concepts of ROS

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow
  - Apply Deep Learning to a Variety of Problems
- Second Day
  - Introduction to the Basic Concepts of ROS
  - Going through ROS tutorials

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow
  - Apply Deep Learning to a Variety of Problems
- Second Day
  - Introduction to the Basic Concepts of ROS
  - Going through ROS tutorials
  - Investigate the RC-Car

Three Day Overview

- First Day
  - Learn and Understand the Basics of Deep Learning
  - Introduction to TensorFlow
  - Apply Deep Learning to a Variety of Problems
- Second Day
  - Introduction to the Basic Concepts of ROS
  - Going through ROS tutorials
  - Investigate the RC-Car
- Third Day - Hackathon

Today's Goals

Today's Goals

- Overview of Deep Learning

Today's Goals

- Overview of Deep Learning
- Theoretical Background of Deep Learning

Today's Goals

- Overview of Deep Learning
- Theoretical Background of Deep Learning
- Knowledge about available Deep Learning Frameworks

Today's Goals

- Overview of Deep Learning
- Theoretical Background of Deep Learning
- Knowledge about available Deep Learning Frameworks
- Applying TensorFlow on a variety of problems

Today's Overview

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow
- 13:30 "Classical" Machine Learning with TensorFlow

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow
- 13:30 "Classical" Machine Learning with TensorFlow
- 14:00 Deep Learning with TensorFlow

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow
- 13:30 "Classical" Machine Learning with TensorFlow
- 14:00 Deep Learning with TensorFlow
- 15:15 Coffee Break

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow
- 13:30 "Classical" Machine Learning with TensorFlow
- 14:00 Deep Learning with TensorFlow
- 15:15 Coffee Break
- 15:30 Small Deep Learning Quiz: "Traffic Sign Classifier"

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow
- 13:30 "Classical" Machine Learning with TensorFlow
- 14:00 Deep Learning with TensorFlow
- 15:15 Coffee Break
- 15:30 Small Deep Learning Quiz: "Traffic Sign Classifier"
- 16:30 Semantic Segmentation

Today's Overview

- 10:00 Motivation and Theoretical Introduction to Deep Learning
- 11:00 Coffee Break
- 11:15 Motivation and Theoretical Introduction to Deep Learning - Continued
- 12:30 Lunch Break
- 13:00 Introduction to TensorFlow
- 13:30 "Classical" Machine Learning with TensorFlow
- 14:00 Deep Learning with TensorFlow
- 15:15 Coffee Break
- 15:30 Small Deep Learning Quiz: "Traffic Sign Classifier"
- 16:30 Semantic Segmentation
- 17:00 Checking who survived :P

Introduction

Why Deep Learning?

Why Deep Learning?

Why Deep Learning?

- It can save a lot of time of booooooring feature engineering and writing exceptions

Why Deep Learning?

- It can save a lot of time of booooooring feature engineering and writing exceptions
- It can write computational rules nobody of us would ever think of

Why Deep Learning?

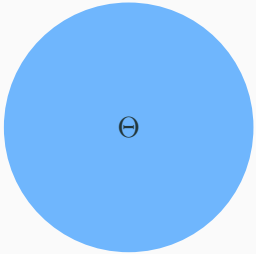
- It can save a lot of time of boooooring feature engineering and writing exceptions
- It can write computational rules nobody of us would ever think of
- It can often write better computational rules than we could

Why Deep Learning?

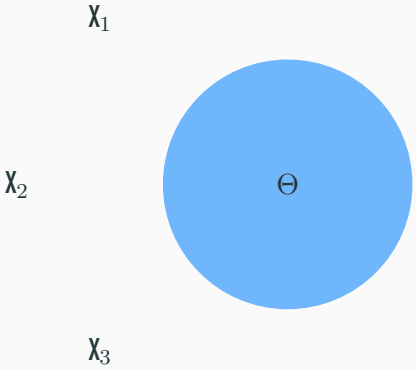
- It can save a lot of time of booooooring feature engineering and writing exceptions
- It can write computational rules nobody of us would ever think of
- It can often write better computational rules than we could
- It can check more cases than we could

Perceptron

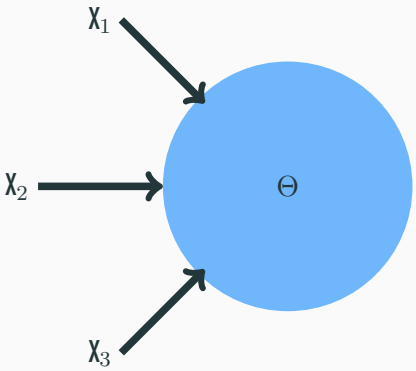
Perceptron



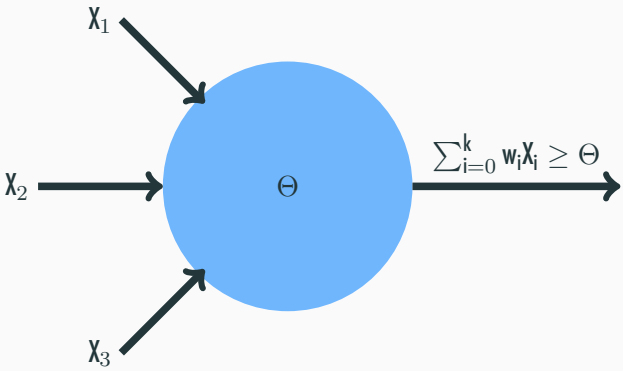
Perceptron



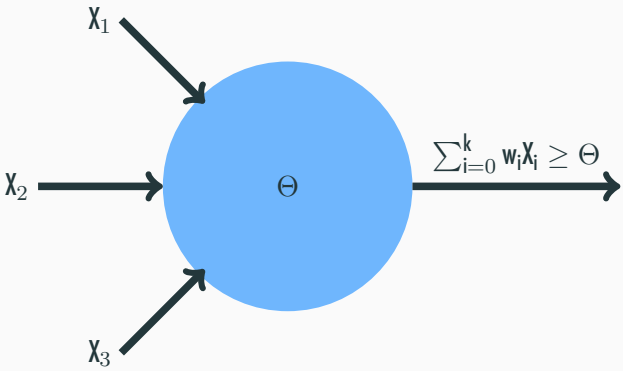
Perceptron



Perceptron



Perceptron



A perceptron can represent every arbitary! boolean function!

Perceptron Training

Perceptron Training

- We start with random weights and feed input for which we know the output

Perceptron Training

- We start with random weights and feed input for which we know the output
- We calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i x_i \geq 0)$

Perceptron Training

- We start with random weights and feed input for which we know the output
- We calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i x_i \geq 0)$
- We determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})x_i$

Perceptron Training

- We start with random weights and feed input for which we know the output
- We calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- We determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Then the new weights can be determined as: $w_i = w_i + \partial w_i$

Perceptron Training

- We start with random weights and feed input for which we know the output
- We calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- We determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Then the new weights can be determined as: $w_i = w_i + \partial w_i$
- Restart the process until the solution is found

Perceptron Training

- We start with random weights and feed input for which we know the output
- We calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- We determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Then the new weights can be determined as: $w_i = w_i + \partial w_i$
- Restart the process until the solution is found
- The solution can only be found in a finite number of iterations, if the problem is linearly seperable

Perceptron Training

- We start with random weights and feed input for which we know the output
- We calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- We determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Then the new weights can be determined as: $w_i = w_i + \partial w_i$
- Restart the process until the solution is found
- The solution can only be found in a finite number of iterations, if the problem is linearly seperable
- However, we can not be certain that the problem is linearly separable

Gradient Descent

Gradient Descent

- Let's forget about the threshold for a second and care about $a = \sum_i w_i X_i$

Gradient Descent

- Let's forget about the threshold for a second and care about $a = \sum_i w_i X_i$
- We can define a regression like error for it as: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$

Gradient Descent

- Let's forget about the threshold for a second and care about $a = \sum_i w_i X_i$
- We can define a regression like error for it as: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$
- As we want to decrease $E(w)$ by changing w we need to calculate the gradient

Gradient Descent

- Let's forget about the threshold for a second and care about $a = \sum_i w_i X_i$
- We can define a regression like error for it as: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$
- As we want to decrease $E(w)$ by changing w we need to calculate the gradient
- $\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2 = \sum_{(x,y) \in D} (y - a) \left(- \sum_k \frac{\partial}{\partial w_i} w_k X_k \right) = \sum_{(x,y) \in D} (y - a)(-X_i)$

Gradient Descent

- Let's forget about the threshold for a second and care about $a = \sum_i w_i X_i$
- We can define a regression like error for it as: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$
- As we want to decrease $E(w)$ by changing w we need to calculate the gradient
- $\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2 = \sum_{(x,y) \in D} (y - a) \left(- \sum_k \frac{\partial}{\partial w_i} w_k X_k \right) = \sum_{(x,y) \in D} (y - a)(-X_i)$
- From that we can then update the weights using: $w_i = w_i + \eta (y - a) X_i$

Comparison of the learning rules

Comparison of the learning rules

- Perceptron Rule: $w_i = w_i + \eta(y - \hat{y})x_i$
  - Guaranteed to converge, if problem is seperable
  - Finite number of steps

Comparison of the learning rules

- Perceptron Rule: $w_i = w_i + \eta(y - \hat{y})x_i$
  - Guaranteed to converge, if problem is seperable
  - Finite number of steps
- Gradient Descent: $w_i = w_i + \eta(y - a)x_i$
  - Robust due to the continuously differentiable functions
  - Only converges in the limit and only to local optima

Sigmoid $\rightarrow$ Differentiable Threshold

Sigmoid $\rightarrow$ Differentiable Threshold

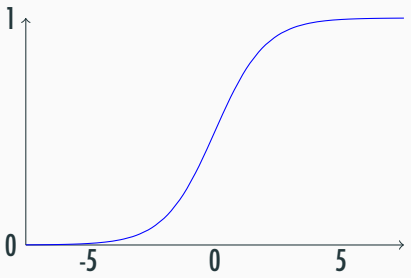
- Even though the perceptron is great for boolean functions it is not differentiable and has no continuous error function

Sigmoid $\rightarrow$ Differentiable Threshold

- Even though the perceptron is great for boolean functions it is not differentiable and has no continuous error function
- $\sigma(a) = \frac{1}{1+e^{-a}}$

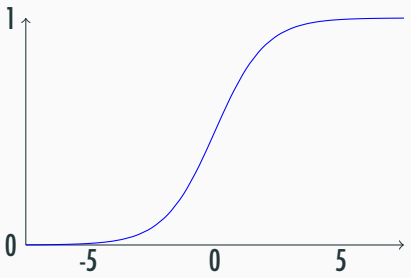
Sigmoid $\rightarrow$ Differentiable Threshold

- Even though the perceptron is great for boolean functions it is not differentiable and has no continuous error function
- $\sigma(a) = \frac{1}{1+e^{-a}}$



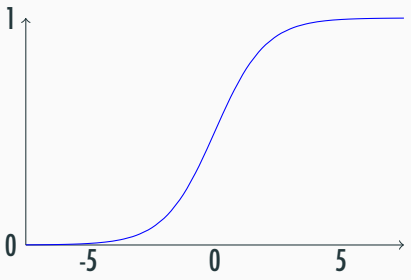
Sigmoid $\rightarrow$ Differentiable Threshold

- Even though the perceptron is great for boolean functions it is not differentiable and has no continuous error function
- $\sigma(a) = \frac{1}{1+e^{-a}}$
- The sigmoid allows us to convert the strict classification into a regression over the probability to classify the points



Sigmoid $\rightarrow$ Differentiable Threshold

- Even though the perceptron is great for boolean functions it is not differentiable and has no continuous error function
- $\sigma(a) = \frac{1}{1+e^{-a}}$
- The sigmoid allows us to convert the strict classification into a regression over the probability to classify the points
- $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$



Softmax

Softmax

- To generalise to n-classification problems we have to extend the function, that calculates the probability

Softmax

- To generalise to n-classification problems we have to extend the function, that calculates the probability
- Softmax: $p = \frac{e^{o_i}}{\sum_i e^{o_i}}$

Softmax

- To generalise to n-classification problems we have to extend the function, that calculates the probability
- Softmax: $p = \frac{e^{o_i}}{\sum_i e^{o_i}}$
- With this we can as well define the general loss

Softmax

- To generalise to n-classification problems we have to extend the function, that calculates the probability
- Softmax: $p = \frac{e^{o_i}}{\sum_i e^{o_i}}$
- With this we can as well define the general loss
- Cross- Entropy $E = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{m} y_{i,k} \ln p_{i,k}$

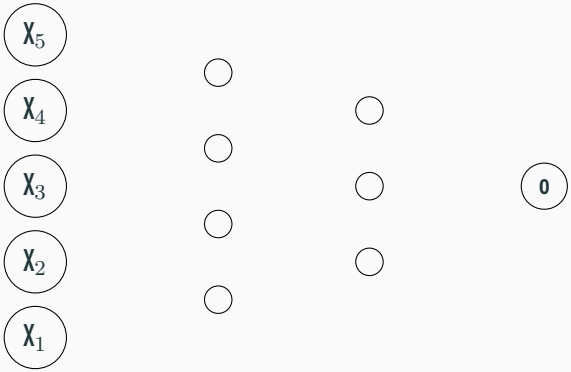
Neural Network

Neural Network

- Similar to the perceptron we have a lot of inputs and an output

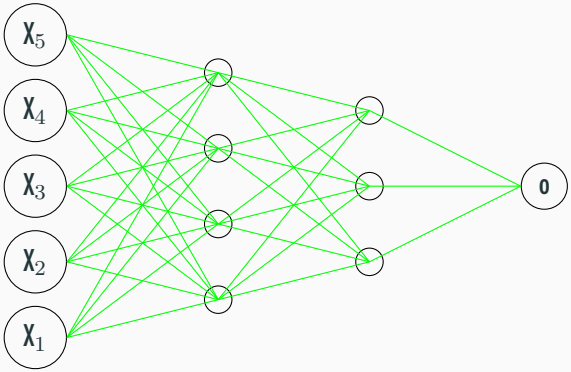
Neural Network

- Similar to the perceptron we have a lot of inputs and an output
- However, instead of going from start to end directly we now add hidden units



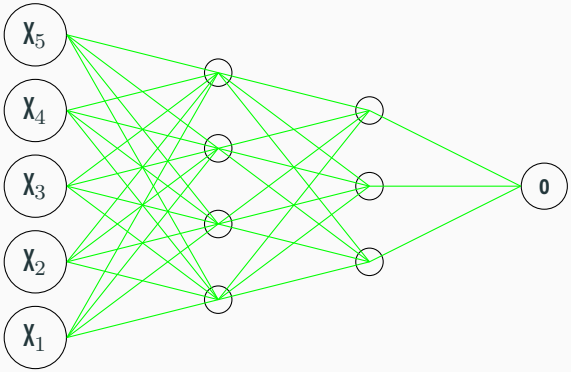
Neural Network

- Similar to the perceptron we have a lot of inputs and an output
- However, instead of going from start to end directly we now add hidden units
- For each of the units we now apply the sigmoid function



Neural Network

- Similar to the perceptron we have a lot of inputs and an output
- However, instead of going from start to end directly we now add hidden units
- For each of the units we now apply the sigmoid function
- Since every unit is differentiable, the whole network is differentiable $\rightarrow$ We can use backpropagation to minimise the error as we can calculate the gradients



Backpropagation

Backpropagation

- How do we do gradient descent now?

Backpropagation

- How do we do gradient descent now?
- Prediction: $\mathbf{o} = \sigma \circ \mathbf{W}^{(2)} \circ \sigma \circ \mathbf{W}^{(1)}(\mathbf{x})$

Backpropagation

- How do we do gradient descent now?
- Prediction: $o = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
- Error Function $E(W) = -\frac{1}{n} \sum_{i=1}^{n} y_i \ln p_i(x) + (1 - y_i) \ln (1 - p_i(x))$

Backpropagation

- How do we do gradient descent now?
- Prediction: $o = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
- Error Function $E(W) = -\frac{1}{n} \sum_{i=1}^{n} y_i \ln p_i(x) + (1 - y_i) \ln (1 - p_i(x))$
- Continuous increment and decrement of error with respect to the probability of the right label

Backpropagation

- How do we do gradient descent now?
- Prediction: $o = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
- Error Function $E(W) = -\frac{1}{n} \sum_{i=1}^{n} y_i \ln p_i(x) + (1 - y_i) \ln (1 - p_i(x))$
- Continuous increment and decrement of error with respect to the probability of the right label
- Gradient: $\nabla E(W) = \left\{ \ldots, \frac{\partial E}{\partial W_k^{(m)}}, \ldots \right\}$

Backpropagation

- How do we do gradient descent now?
- Prediction: $o = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
- Error Function $E(W) = -\frac{1}{n} \sum_{i=1}^{n} y_i \ln p_i(x) + (1 - y_i) \ln (1 - p_i(x))$
- Continuous increment and decrement of error with respect to the probability of the right label
- Gradient: $\nabla E(W) = \left\{ \ldots, \frac{\partial E}{\partial W_k^{(m)}}, \ldots \right\}$
- Let us write $h_1 = \sigma \circ W^{(1)}(x)$ and $h = \sigma \circ W^{(2)} \circ h_1$

Backpropagation

- How do we do gradient descent now?
- Prediction: $o = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
- Error Function $E(W) = -\frac{1}{n} \sum_{i=1}^{n} y_i \ln p_i(x) + (1 - y_i) \ln (1 - p_i(x))$
- Continuous increment and decrement of error with respect to the probability of the right label
- Gradient: $\nabla E(W) = \left\{ \ldots, \frac{\partial E}{\partial W_k^{(m)}}, \ldots \right\}$
- Let us write $h_1 = \sigma \circ W^{(1)}(x)$ and $h = \sigma \circ W^{(2)} \circ h_1$
- If we apply the chain rule: $\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial h} \frac{\partial h}{\partial h_1} \frac{\partial h_1}{\partial W^{(1)}}$ and $\frac{\partial E}{\partial W^{(2)}} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial h} \frac{\partial h}{\partial W^{(2)}}$

Backpropagation

- How do we do gradient descent now?
- Prediction: $o = \sigma \circ W^{(2)} \circ \sigma \circ W^{(1)}(x)$
- Error Function $E(W) = -\frac{1}{n} \sum_{i=1}^{n} y_i \ln p_i(x) + (1 - y_i) \ln (1 - p_i(x))$
- Continuous increment and decrement of error with respect to the probability of the right label
- Gradient: $\nabla E(W) = \left\{ \ldots, \frac{\partial E}{\partial W_k^{(m)}}, \ldots \right\}$
- Let us write $h_1 = \sigma \circ W^{(1)}(x)$ and $h = \sigma \circ W^{(2)} \circ h_1$
- If we apply the chain rule: $\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial h} \frac{\partial h}{\partial h_1} \frac{\partial h_1}{\partial W^{(1)}}$ and $\frac{\partial E}{\partial W^{(2)}} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial h} \frac{\partial h}{\partial W^{(2)}}$
- One can see, that we can start calculating the gradient at the end and continue to the input

Optimising Weights

Optimising Weights

- Gradient Descent

Optimising Weights

- Gradient Descent
- Advanced Methods

Optimising Weights

- Gradient Descent
- Advanced Methods
    - Momentum

Optimising Weights

- Gradient Descent
- Advanced Methods
  - Momentum
  - Higher Order Derivatives

Optimising Weights

- Gradient Descent
- Advanced Methods
  - Momentum
  - Higher Order Derivatives
  - Randomised Optimisation

Optimising Weights

- Gradient Descent
- Advanced Methods
  - Momentum
  - Higher Order Derivatives
  - Randomised Optimisation
  - Penalty for Complexity

Restriction Bias

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered
- Perceptron $\rightarrow$ half spaces

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered
- Perceptron $\rightarrow$ half spaces
- Sigmoids $\rightarrow$ non-linear function

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered
- Perceptron $\rightarrow$ half spaces
- Sigmoids $\rightarrow$ non-linear function
- What can we represent with a network?

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered
- Perceptron $\rightarrow$ half spaces
- Sigmoids $\rightarrow$ non-linear function
- What can we represent with a network?
    - Boolean Functions: Yes, because we have a network of threshold-like units

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered
- Perceptron $\rightarrow$ half spaces
- Sigmoids $\rightarrow$ non-linear function
- What can we represent with a network?
  - Boolean Functions: Yes, because we have a network of threshold-like units
  - Continous Functions: Yes, with one (infinitly wide) hidden layer

Restriction Bias

- Representational power of the algorithm due to the hypothesis considered
- Perceptron $\rightarrow$ half spaces
- Sigmoids $\rightarrow$ non-linear function
- What can we represent with a network?
  - Boolean Functions: Yes, because we have a network of threshold-like units
  - Continous Functions: Yes, with one (infinitly wide) hidden layer
  - Arbitrary Functions: Yes, with two (infinitly wide) hidden layers

Frameworks

Available Deep Learning Frameworks

Available Deep Learning Frameworks

- TensorFlow

Available Deep Learning Frameworks

- TensorFlow
- Caffe

Available Deep Learning Frameworks

- TensorFlow
- Caffe
- PyTorch

Available Deep Learning Frameworks

- TensorFlow
- Caffe
- PyTorch
- Theano (Deprecated)

Available Deep Learning Frameworks

- TensorFlow
- Caffe
- PyTorch
- Theano (Deprecated)
- Keras

Advantages, Disadvantages

Framework	Advantages	Disadvantages

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow		

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python
PyTorch		

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python
PyTorch	Perfect integration into Python Objects Graph can be dynamically changed without rebuilding it Will probably be integrated into TensorRT	

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python
PyTorch	Perfect integration into Python Objects Graph can be dynamically changed without rebuilding it Will probably be integrated into TensorRT	Still quite new

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python
PyTorch	Perfect integration into Python Objects Graph can be dynamically changed without rebuilding it Will probably be integrated into TensorRT	Still quite new
Caffe		

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python
PyTorch	Perfect integration into Python Objects Graph can be dynamically changed without rebuilding it Will probably be integrated into TensorRT	Still quite new
Caffe	Big Community Very well integrated into TensorRT A bit more	

Advantages, Disadvantages

Framework	Advantages	Disadvantages
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are relatively static No real integration into Python
PyTorch	Perfect integration into Python Objects Graph can be dynamically changed without rebuilding it Will probably be integrated into TensorRT	Still quite new
Caffe	Big Community Very well integrated into TensorRT A bit more	Graphs are relatively static No real integration into Python

Tensorflow Tutorials

Tensorflow Tutorials

- Let us have a look at some TensorFlow examples

Tensorflow Tutorials

- Let us have a look at some TensorFlow examples
- Use the terminal to navigate to TensorFlow-Examples/notebooks and run 'jupyter notebook'

Small Test: Traffic Sign Classifier

Small Test: Traffic Sign Classifier

- Let's see what you learned.

Small Test: Traffic Sign Classifier

- Let's see what you learned.
- Use the terminal to navigate to TrafficSignClassifier and run 'jupyter notebook'

Small Test: Traffic Sign Classifier

- Let's see what you learned.
- Use the terminal to navigate to TrafficSignClassifier and run 'jupyter notebook'
- Try solving the given problem with a network of your choice

SegNet

Motivation

Motivation

- Just assigning an image to class is not sufficent in lots of situations

Motivation

- Just assigning an image to class is not sufficent in lots of situations What label would you assign?

Motivation

- Just assigning an image to class is not sufficent in lots of situations What label would you assign?
  - Image from Car?

Motivation

- Just assigning an image to class is not sufficent in lots of situations What label would you assign?
  - Image from Car?
  - Street?
  - Traffic Lights?

Motivation

- Just assigning an image to class is not sufficent in lots of situations What label would you assign?
  - Image from Car?
  - Street?
  - Traffic Lights?
  - Akka Hamburg?

Motivation

- Just assigning an image to class is not sufficent in lots of situations What label would you assign?
  - Image from Car?
  - Street?
  - Traffic Lights?
  - Akka Hamburg?
- We clearly need another approach, if we want to have a car driving

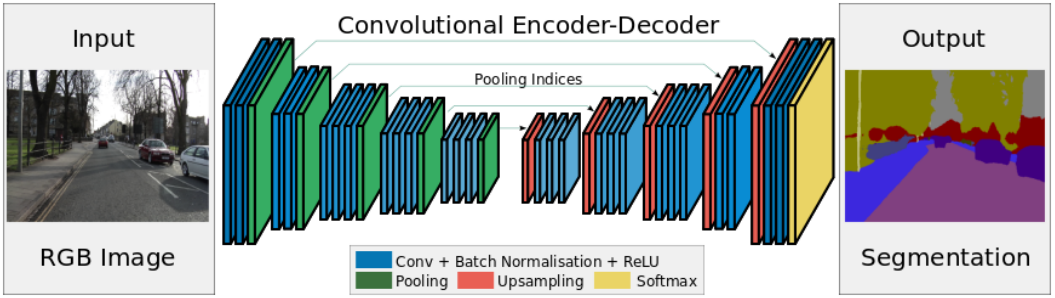
Bounding Boxes

Bounding Boxes

Semantic Segmentation

Semantic Segmentation

Network Structure



Implementation

- Let's investigate how to programm a SegNet

Implementation

- Let's investigate how to programm a SegNet
- Use the terminal to navigate to SegNet and run 'jupyter notebook'

Implementation

- Let's investigate how to programm a SegNet
- Use the terminal to navigate to SegNet and run 'jupyter notebook'
- We probably cannot run it as it takes 6+ GB of VRAM