Deep Learning

QCon London, March 7th

Today's Overview

Today's Overview

- 9:00 Intro of Deep Learning

Today's Overview

- 9:00 Intro of Deep Learning
- 9:30 Build and train a Perceptron in numpy

Today's Overview

- 9:00 Intro of Deep Learning
- 9:30 Build and train a Perceptron in numpy
- 10:00 Move the code to the GPU using PyTorch

Today's Overview

- 9:00 Intro of Deep Learning
- 9:30 Build and train a Perceptron in numpy
- 10:00 Move the code to the GPU using PyTorch
- 11:00 Use a neural network for more complex time-series forecasting

Today's Overview

- 9:00 Intro of Deep Learning
- 9:30 Build and train a Perceptron in numpy
- 10:00 Move the code to the GPU using PyTorch
- 11:00 Use a neural network for more complex time-series forecasting
- 13:00 Introduction to NLP

Today's Overview

- 9:00 Intro of Deep Learning
- 9:30 Build and train a Perceptron in numpy
- 10:00 Move the code to the GPU using PyTorch
- 11:00 Use a neural network for more complex time-series forecasting
- 13:00 Introduction to NLP
- 15:00 TCN vs. RNN

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?

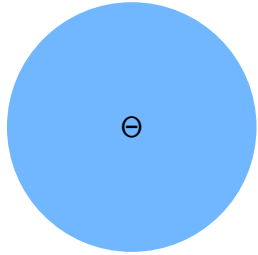
- It can save a lot of time of booooooring feature engineering and writing exceptions
- It can write computational rules nobody of us would ever think of
- It can often write better computational rules than we could

Why Deep Learning?

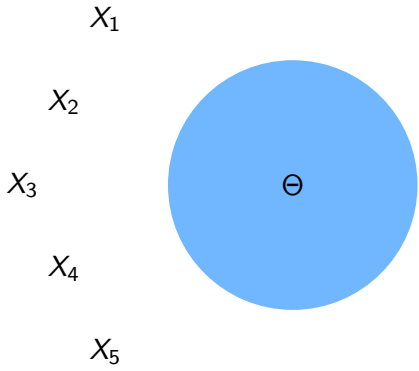
- It can save a lot of time of booooooring feature engineering and writing exceptions
- It can write computational rules nobody of us would ever think of
- It can often write better computational rules than we could
- It can check more cases than we could

Perceptron

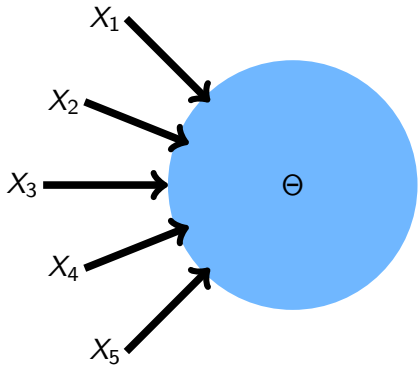
Perceptron



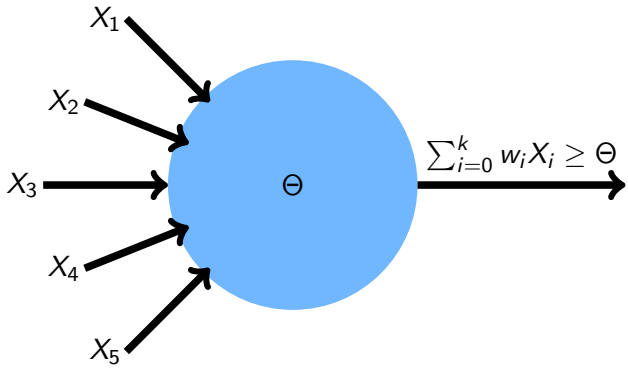
Perceptron



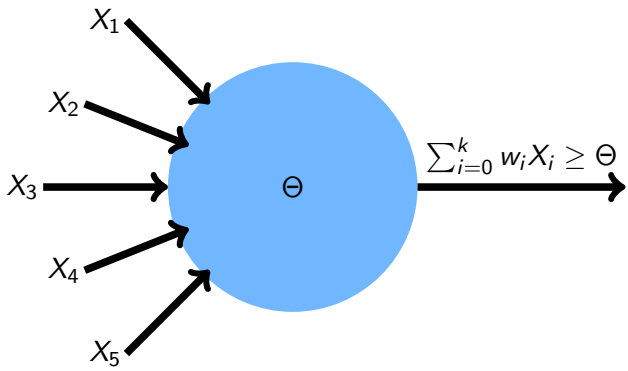
Perceptron



Perceptron



Perceptron



A perceptron can represent every arbitary! boolean function!

Perceptron Training

Perceptron Training

- Start with random weights

Perceptron Training

- Start with random weights
- Calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$

Perceptron Training

- Start with random weights
- Calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- Determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$

Perceptron Training

- Start with random weights
- Calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- Determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Calculate new weights as: $w_i = w_i + \partial w_i$

Perceptron Training

- Start with random weights
- Calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- Determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Calculate new weights as: $w_i = w_i + \partial w_i$
- Restart the process until the solution is found

Perceptron Training

- Start with random weights
- Calculate the perceptron's estimate of the solution with: $\hat{y} = (\sum_i w_i X_i \geq 0)$
- Determine the update for the weights using the error and the learning rate: $\partial w_i = \eta(y - \hat{y})X_i$
- Calculate new weights as: $w_i = w_i + \partial w_i$
- Restart the process until the solution is found
- Let us actually implement that in the first notebook (Perceptron)

Conclusions

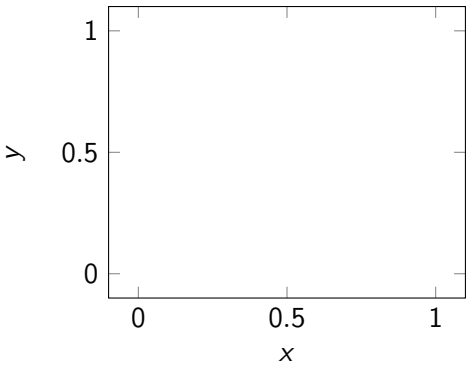
Conclusions

- Congratulations on training your first perceptron! You just taught a computer to learn!

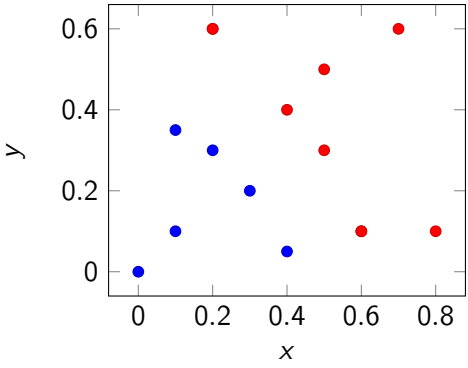
Conclusions

- Congratulations on training your first perceptron! You just taught a computer to learn!
- Why didn't we wait for the final weights?

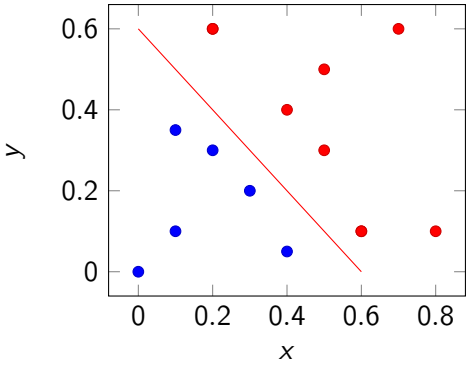
Linear Separability



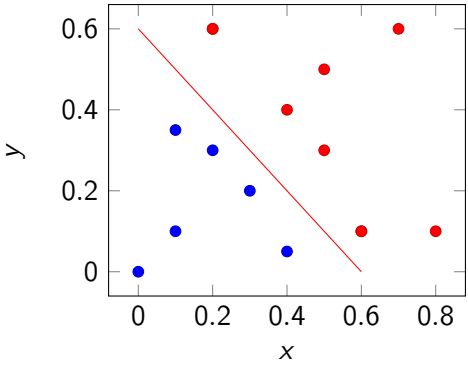
Linear Separability

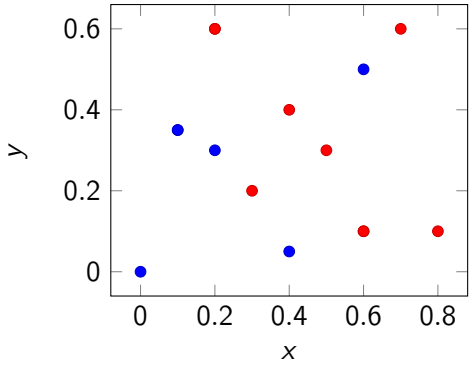


Linear Separability

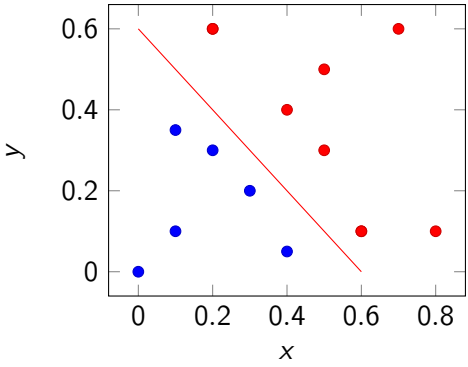


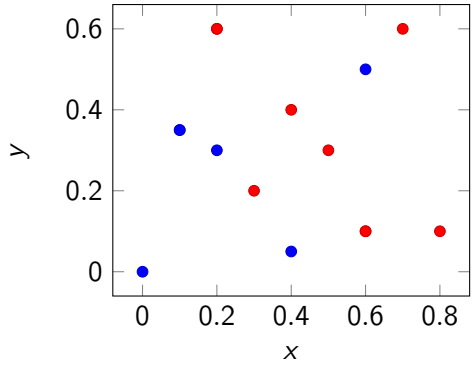
Linear Separability





Linear Separability





The algorithm only converges, if the problem is linearly separable.

Gradient Descent

Gradient Descent

- What we calculated is an activation $a = \sum_i w_i X_i$

Gradient Descent

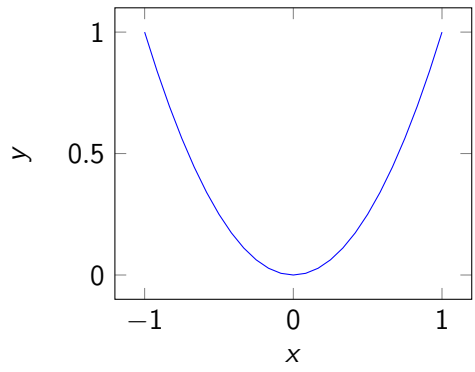
- What we calculated is an activation $a = \sum_i w_i X_i$
- Let us ignore the threshold and try to get the activation as close to the target value as possible.

Gradient Descent

- What we calculated is an activation $a = \sum_i w_i X_i$
- Let us ignore the threshold and try to get the activation as close to the target value as possible.
- This brings us back to regression with an error: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$

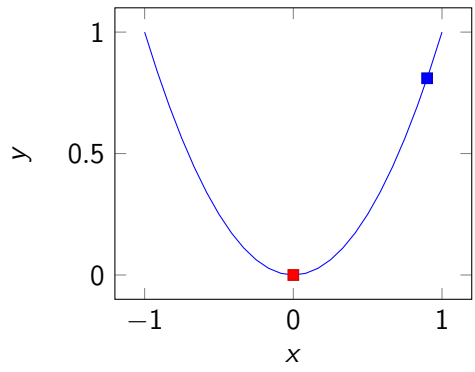
Gradient Descent

- What we calculated is an activation $a = \sum_i w_i X_i$
- Let us ignore the threshold and try to get the activation as close to the target value as possible.
- This brings us back to regression with an error: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$



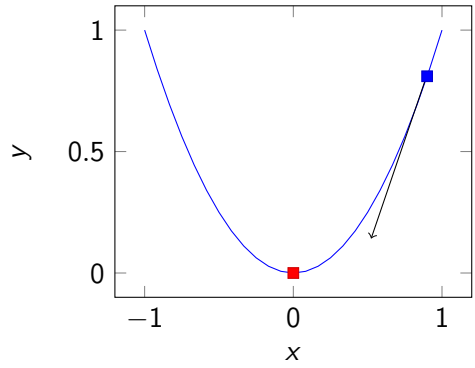
Gradient Descent

- What we calculated is an activation $a = \sum_i w_i X_i$
- Let us ignore the threshold and try to get the activation as close to the target value as possible.
- This brings us back to regression with an error: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$



Gradient Descent

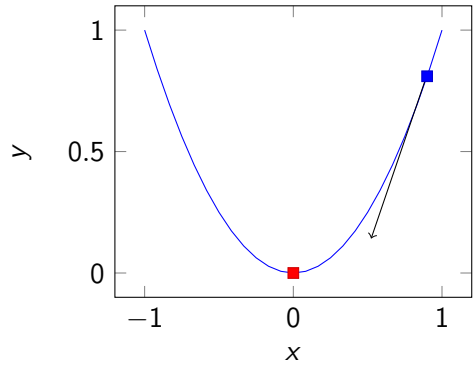
- What we calculated is an activation $a = \sum_i w_i X_i$
- Let us ignore the threshold and try to get the activation as close to the target value as possible.
- This brings us back to regression with an error: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$



- Every deep learning framework can calculate those weights using the chain rule and backpropagation.

Gradient Descent

- What we calculated is an activation $a = \sum_i w_i X_i$
- Let us ignore the threshold and try to get the activation as close to the target value as possible.
- This brings us back to regression with an error: $E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$
- If we want to decrease $E(w)$ by changing w we need to calculate the gradient of w
- Every deep learning framework can calculate those weights using the chain rule and backpropagation.



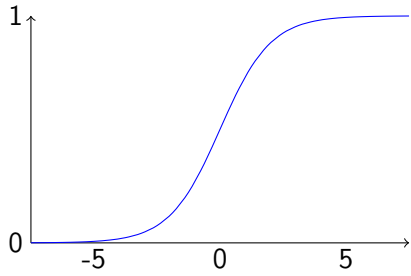
Sigmoid $\rightarrow$ Differentiable Threshold

Sigmoid $\rightarrow$ Differentiable Threshold

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

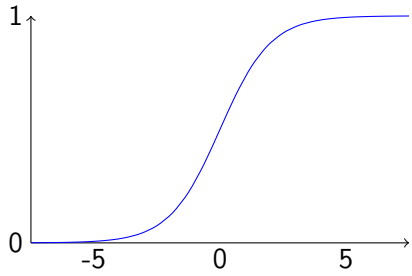
Sigmoid $\rightarrow$ Differentiable Threshold

- 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
- $\sigma(a) = \frac{1}{1+e^{-a}}$
- The sigmoid allows us to convert the strict classification into a regression over the probability to classify the points



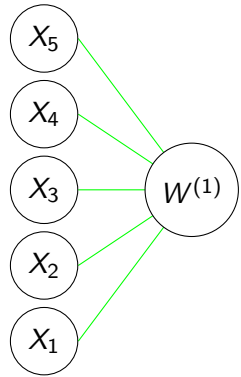
Backpropagation

Backpropagation

- How do we do gradient descent now?

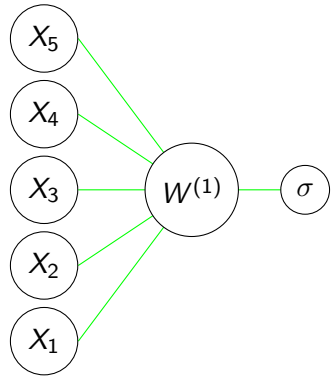
Backpropagation

- How do we do gradient descent now?
- Activation: $a = W^{(1)} \circ x$



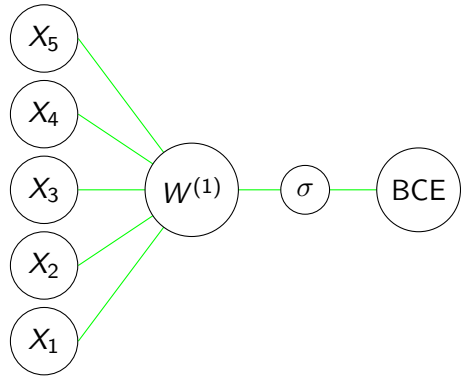
Backpropagation

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



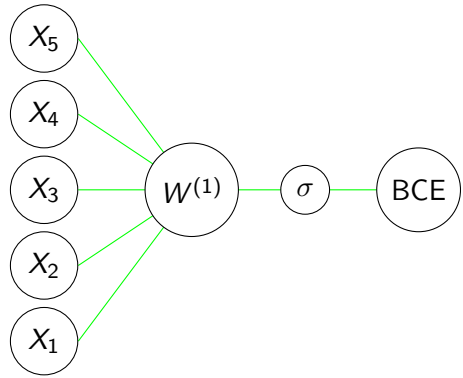
Backpropagation

- How do we do gradient descent now?
- Activation: $a = W^{(1)} \circ x$
- Prediction: $o = \sigma(a)$
- Error Function
  $E(o) = -t \log o - (1 - t) \log(1 - o)$



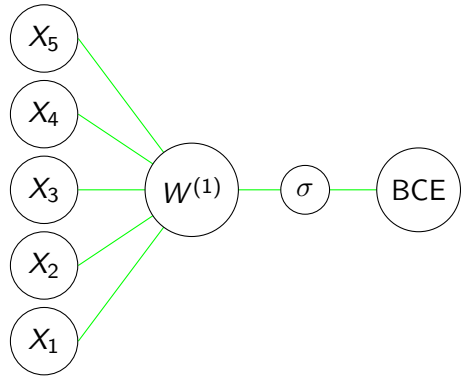
Backpropagation

- How do we do gradient descent now?
- Activation: $a = W^{(1)} \circ x$
- Prediction: $o = \sigma(a)$
- Error Function
  $E(o) = -t \log o - (1 - t) \log(1 - o)$
- Gradient: $\frac{\partial E}{\partial W} = E(o) \frac{\partial E}{\partial o} \frac{\partial o}{\partial a} \frac{\partial a}{\partial W^{(1)}}$



Backpropagation

- How do we do gradient descent now?
- Activation: $a = W^{(1)} \circ x$
- Prediction: $o = \sigma(a)$
- Error Function
  $E(o) = -t \log o - (1 - t) \log(1 - o)$
- Gradient: $\frac{\partial E}{\partial W} = E(o) \frac{\partial E}{\partial o} \frac{\partial o}{\partial a} \frac{\partial a}{\partial W^{(1)}}$
- This allows a framework to compute gradients for every node, given that the gradient for each step is known.



Numerical Instability

Numerical Instability

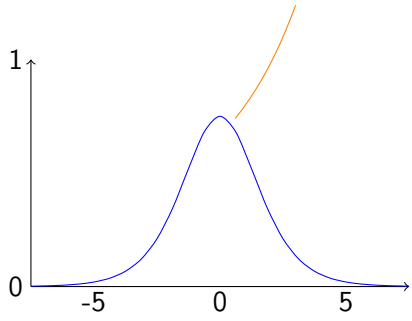
- The derivative of sigma is $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$

Numerical Instability

- The derivative of sigma is $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$
- Derivative for t=0: $-\frac{1}{\sigma(a)} * err$

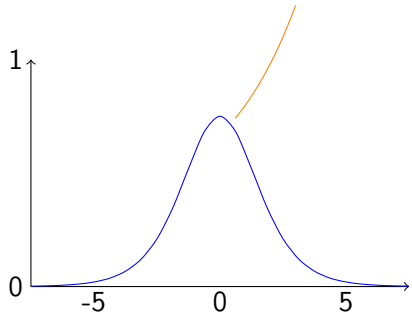
Numerical Instability

- The derivative of sigma is $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$
- Derivative for t=0: $-\frac{1}{\sigma(a)} * err$
- Derivative for t=1: $-\frac{1}{1-\sigma(a)} * err$



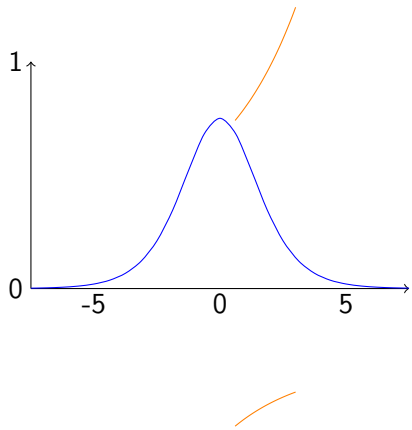
Numerical Instability

- The derivative of sigma is $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$
- Derivative for t=0: $-\frac{1}{\sigma(a)} * err$
- Derivative for t=1: $-\frac{1}{1-\sigma(a)} * err$
- t=0, $\sigma(a)$=0: $err = 0$, $\sigma(a) = 1$: $err = \infty$



Numerical Instability

- The derivative of sigma is $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$
- Derivative for t=0: $-\frac{1}{\sigma(a)} * err$
- Derivative for t=1: $-\frac{1}{1-\sigma(a)} * err$
- t=0, $\sigma(a)$=0: $err = 0$, $\sigma(a) = 1$: $err = \infty$
- t=1, $\sigma(a) = 0$: $err = \infty$, $\sigma(a) = 1$: $err = 0$



Softmax

Softmax

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

Softmax

- To generalize 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 generalize 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 generalize 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 $E = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{m} y_{i,k} \ln p_{i,k}$

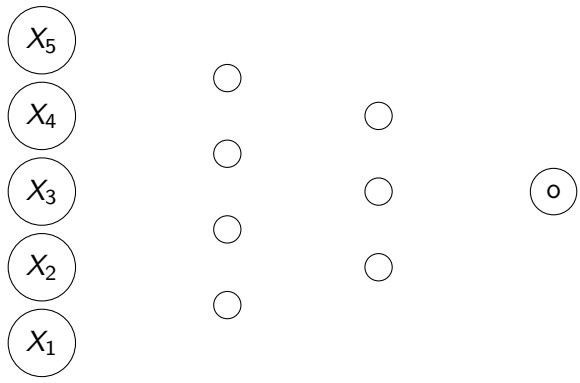
Neural Network

Neural Network

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

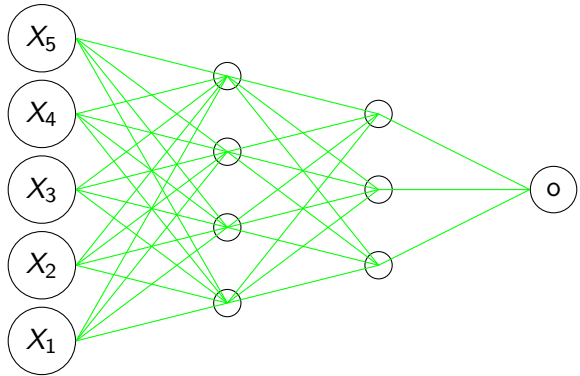
Neural Network

- 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



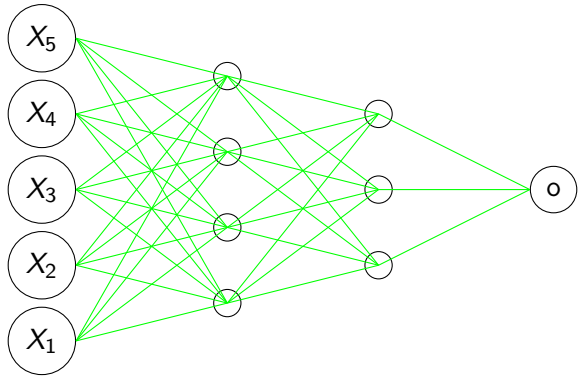
Neural Network

- 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



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

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	Disadva

Advantages, Disadvantages

Framework	Advantages	Disadva
TensorFlow		

Advantages, Disadvantages

Framework	Advantages	Disadva
TensorFlow	Huge Community Easy Conversion for mobile devices	

Advantages, Disadvantages

Framework	Advantages	Disadva
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are re No real integrat

Advantages, Disadvantages

Framework	Advantages	Disadva
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are re No real integrat
PyTorch		

Advantages, Disadvantages

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

Advantages, Disadvantages

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

Advantages, Disadvantages

Framework	Advantages	Disadva
TensorFlow	Huge Community Easy Conversion for mobile devices	Graphs are re No real integrat
PyTorch	Perfect integration into Python Objects Graph can be dynamically changed without rebuilding it Will probably be integrated into TensorRT	Still qui
Caffe		

Advantages, Disadvantages

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

Advantages, Disadvantages

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

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

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

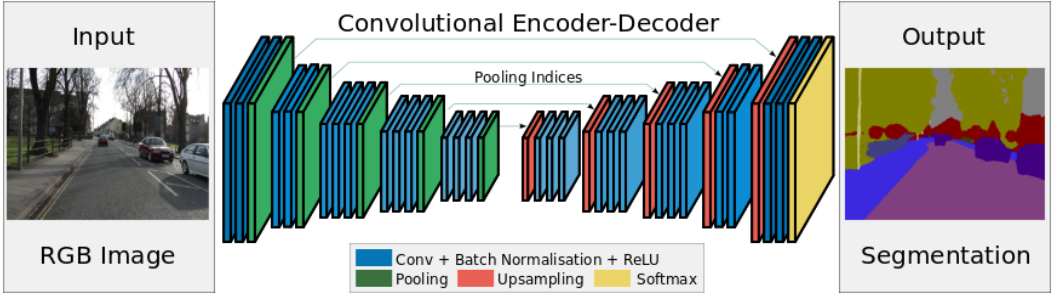
Bounding Boxes

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