Introduction to Neural Networks

Fernando Martinez

January 30, 2020

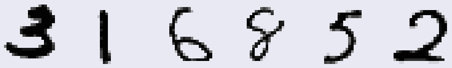
Outline

Introduction

Apparently easy problems that are difficult

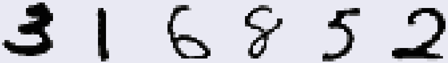
Apparently easy problems that are difficult

Handwritten numbers recognition



Apparently easy problems that are difficult

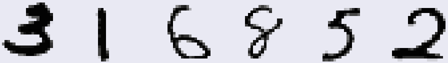
Handwritten numbers recognition



Cat or dog

Apparently easy problems that are difficult

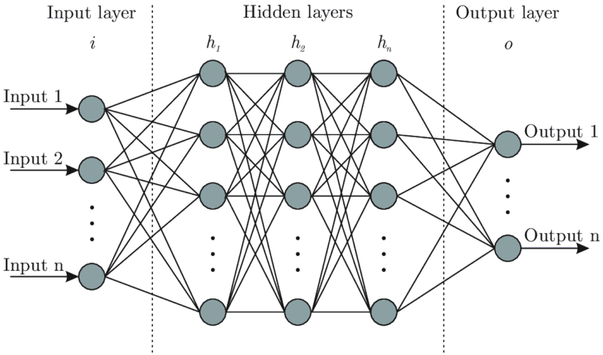
Handwritten numbers recognition



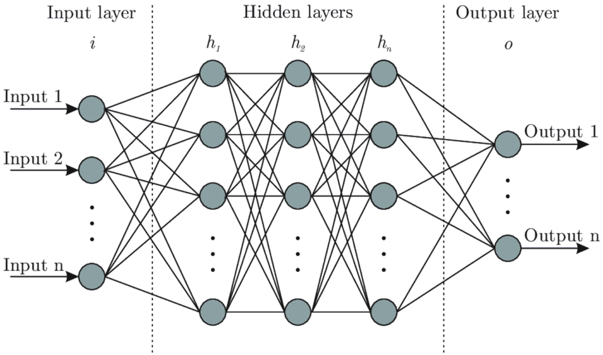
Cat or dog

How can we program a computer to perform these tasks?

What is a neural network?



What is a neural network?



We can think of different pairs of input and output:

- Pixels in an image → Number that the image represents.
- State of a chess game → What should be the next move.
- Text → Audio of a person reading that text.

Single neuron example

Processing the input

First, we need to understand how the neuron will process the input to produce the output:

Processing the input

First, we need to understand how the neuron will process the input to produce the output:

- We start with an input $\mathbf{x}$.

Processing the input

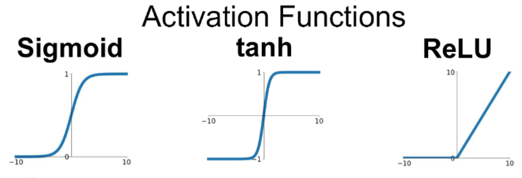
First, we need to understand how the neuron will process the input to produce the output:

- We start with an input $\mathbf{x}$.
- The input is moved to the neuron by multiplying by a weight $\mathbf{w}$ and adding a bias $\mathbf{b}$.

Processing the input

First, we need to understand how the neuron will process the input to produce the output:

- We start with an input **x**.
- The input is moved to the neuron by multiplying by a weight **w** and adding a bias **b**.
- The obtained value $\mathbf{x} \cdot \mathbf{w} + \mathbf{b}$ is evaluated by an activation function. This simulates the firing or not of the neuron.



$\sigma(x) = \frac{1}{1+e^{-x}}$ $\tanh(x)$ $\max(0, x)$

The obtained output will be Activation$(x \cdot w + b)$

Evaluating the output

We have managed to obtain an output, but since the weight and bias used are initialized randomly, this output will be random.

Evaluating the output

We have managed to obtain an output, but since the weight and bias used are initialized randomly, this output will be random.
We need a way of telling the neural network when it produces a wrong output and a way of correcting its parameters accordingly to improve its performance.

Evaluating the output

We have managed to obtain an output, but since the weight and bias used are initialized randomly, this output will be random.

We need a way of telling the neural network when it produces a wrong output and a way of correcting its parameters accordingly to improve its performance.

- Evaluation of the output's quality $\rightarrow$ Loss criterion

Evaluating the output

We have managed to obtain an output, but since the weight and bias used are initialized randomly, this output will be random.
We need a way of telling the neural network when it produces a wrong output and a way of correcting its parameters accordingly to improve its performance.

- Evaluation of the output's quality $\rightarrow$ Loss criterion
- Correction of the parameters $\rightarrow$ Backpropagation

Loss criterion

- We have obtained an **output**.

Loss criterion

- We have obtained an **output**.
- We know that the correct value of that the output should be **target**.

Loss criterion

- We have obtained an **output**.
- We know that the correct value of that the output should be **target**.
- We can define a function as our criterion of how far is the **output** from the true **target** (loss function).

$$\mathrm{MSELoss}(\mathrm{output}, \mathrm{target}) = (\mathrm{output} - \mathrm{target})^2 \tag{1}$$

Backpropagation

- Now we know how bad/good was our output.

Backpropagation

- Now we know how bad/good was our output.
- We need a method for using the loss (L) obtained through the network to change the parameters in such a way that reduces the loss.

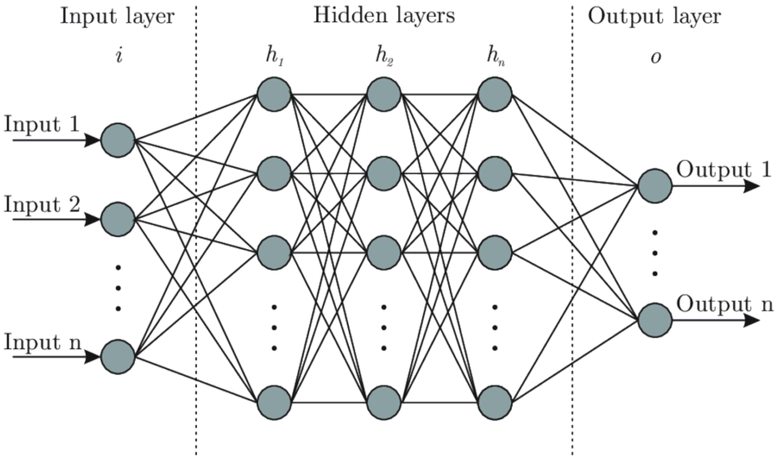
Backpropagation

- Now we know how bad/good was our output.
- We need a method for using the loss (L) obtained through the network to change the parameters in such a way that reduces the loss.
- $\Delta L \approx \frac{\partial L}{\partial w}\Delta w + \frac{\partial L}{\partial b}\Delta b \rightarrow \Delta w = -\eta \frac{\partial L}{\partial w}, \Delta b = -\eta \frac{\partial L}{\partial b}$
  $\eta \equiv$ learning rate

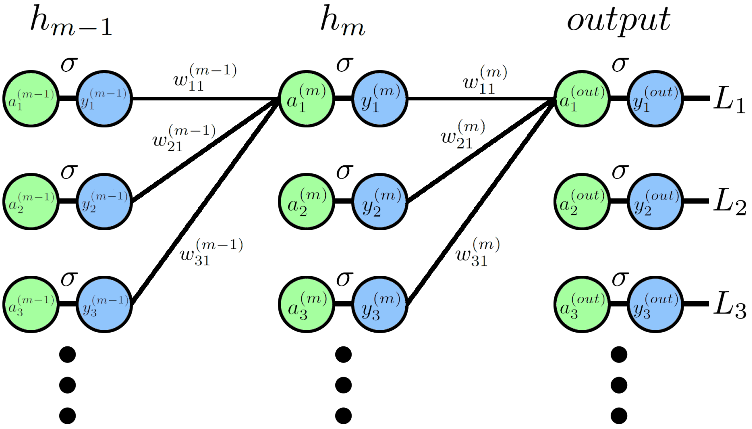
Neuron with PyTorch

Hidden layers

Structure of a neural network



Correcting the parameters of the neural network



Correcting $w_{11}^{(m)}$:

- $$\frac{\partial L_1}{\partial w_{11}^{(m)}} = \frac{\partial L_1}{\partial y_1^{(out)}} \frac{\partial y_1^{(out)}}{\partial a_1^{(out)}} \frac{\partial a_1^{(out)}}{\partial w_{11}^{(m)}}$$

Backpropagation example

For every layer k:

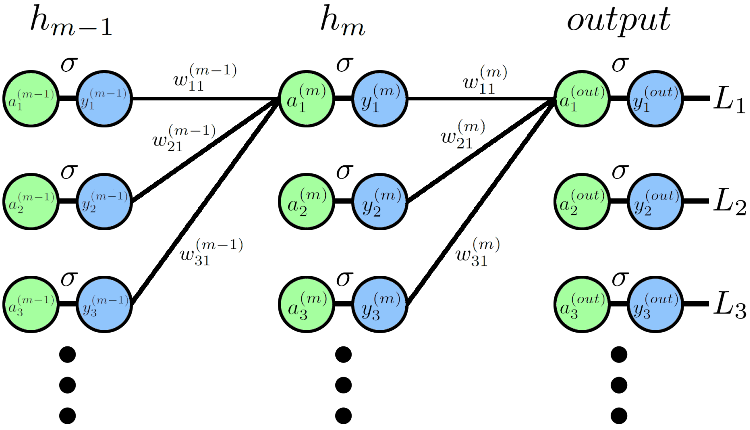
- $a_j^{(k)} = \sum_i w_{ij}^{(k)} y_i^{(k-1)} + b_j^{(k)}$, $y_j^{(k)} = \sigma(a_j^{(k)})$.
- $\sigma(x) \equiv$activation function.

Thus, considering MSE as our loss function ($L = \sum_i (y_i^{(out)} - target_i)^2$):

- $\frac{\partial L_1}{\partial y_1^{(out)}} = 2(y_i^{(out)} - target_i)$
- $\frac{\partial y_1^{(out)}}{\partial a_1^{(out)}} = \sigma'(a_1^{(out)}) = \sigma(a_1^{(out)})(1 - \sigma(a_1^{(out)}))$ (For the sigmoid activation function)
- $\frac{\partial a_1^{(out)}}{\partial w_{11}^{(m)}} = y_1^{(m-1)}$

We can obtain the corrections $\Delta w_{ij}^{(m)} = -\eta \frac{\partial L_j}{\partial w_{ij}^{(m)}}$ for all the weights after the last hidden layer.

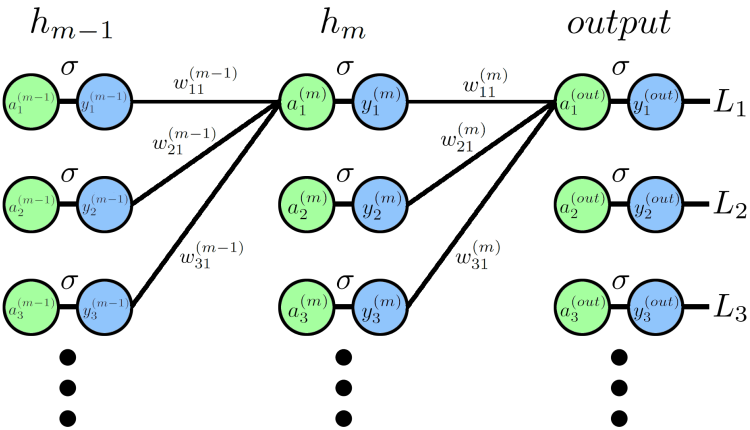
Correcting the parameters of the neural network



Correcting $w_{11}^{(m-1)}$:

- $$\frac{\partial L_1}{\partial w_{11}^{(m-1)}} = \frac{\partial L_1}{\partial y_1^{(out)}} \frac{\partial y_1^{(out)}}{\partial a_1^{(out)}} \frac{\partial a_1^{(out)}}{\partial y_1^{(m)}} \frac{\partial y_1^{(m)}}{\partial a_1^{(m)}} \frac{\partial a_1^{(m)}}{\partial w_{11}^{(m-1)}}$$

Correcting the parameters of the neural network

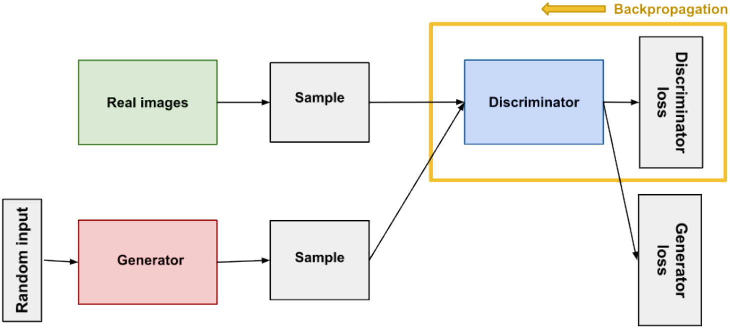


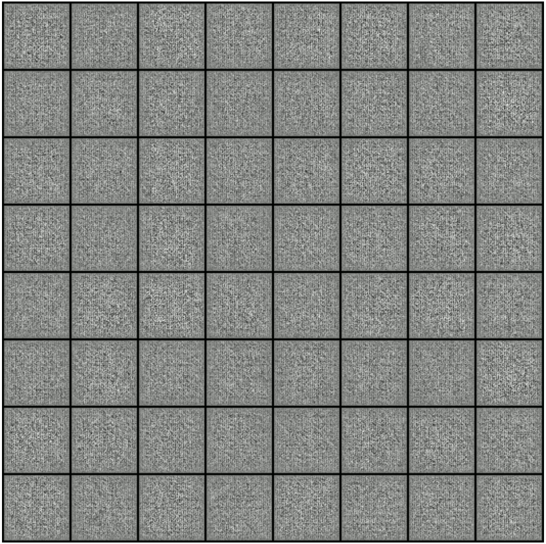
Obtaining the corrections for the parameters of the neural network requires simple calculations.

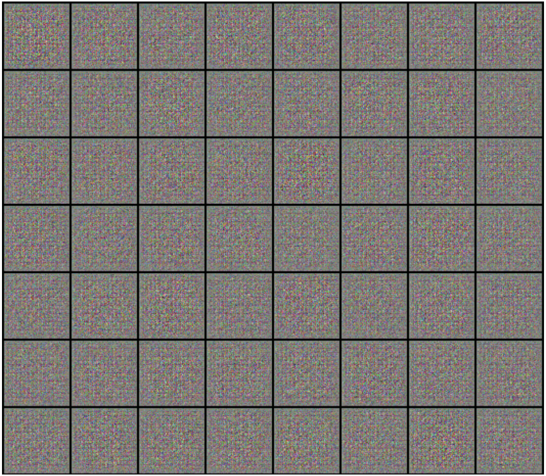
MNIST image recognition

Generative Adversarial Networks (GANs)

GANs structure







Summary

Go

- We have learned the very basics of neural networks.

Go

- We have learned the very basics of neural networks.
- We have seen the basics of how to use PyTorch to define perform the training of a neural network.

Go

Introduction to Neural Networks
Fernando Martinez
Introduction
Single neuron example
Neuron with PyTorch
Hidden layers
MNIST image recognition
Generative Adversarial Networks (GANs)
Summary
Other examples

- We have learned the very basics of neural networks.
- We have seen the basics of how to use PyTorch to define perform the training of a neural network.
- Neural networks can have different forms that allows them to perform different tasks.

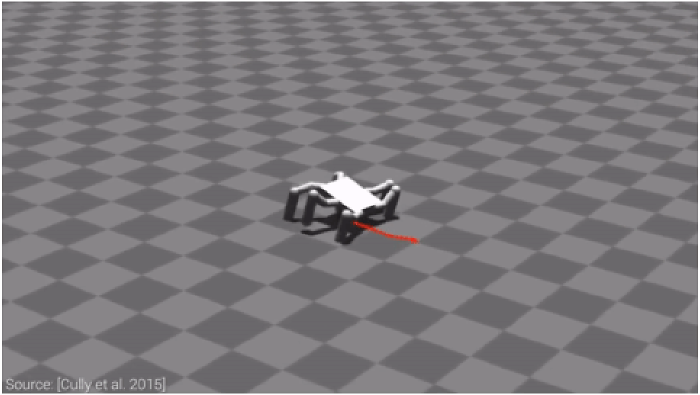
Go

Configuration and strength[62]

Versions	Hardware	Elo rating	Date	Results
AlphaGo Fan	176 GPUs,[53] distributed	3,144[52]	Oct 2015	5:0 against Fan Hui
AlphaGo Lee	48 TPUs,[53] distributed	3,739[52]	Mar 2016	4:1 against Lee Sedol
AlphaGo Master	4 TPUs,[53] single machine	4,858[52]	May 2017	60:0 against professional players; Future of Go Summit
AlphaGo Zero	4 TPUs,[53] single machine	5,185[52]	Oct 2017	100:0 against AlphaGo Lee 89:11 against AlphaGo Master
AlphaZero	4 TPUs, single machine	N/A	Dec 2017	60:40 against AlphaGo Zero

Silver, D. et al. "Mastering the game of go without human knowledge." Nature, 550(7676), 354-359.

Teaching robots how to walk



Teaching robots how to walk

Thanks for your attention!