1

Deep Learning
- introductie -

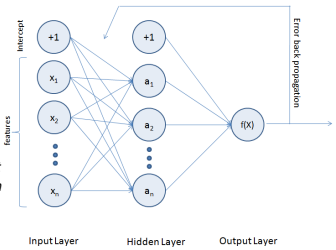
- neurons, layers, perceptron,
- Artificial Neural Network (ANN)
- neural network playground
- tensorflow 2 en keras

"Do not worry about your difficulties in Mathematics. I can assure you mine are still greater."
Albert Einstein

2

Deep Learning
- introductie -

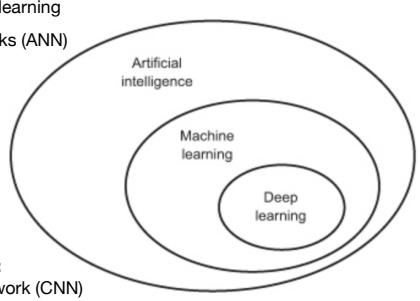
- *"Deep Learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with raw input) into representation at a higher, slightly more abstract level."*
- *"The key aspect of deep learning is that these layers are not designed by human engineers: they are learned from data using a general-purpose learning procedure."*
- **[LeCun, Benhgio, and Hinton , Nature 2015]**



Deep Learning

- introductie -

- Deep Learning is een subset van methoden voor machine learning
  - Artificial Neural Networks (ANN)
  - Neurons, Perceptron,
  - multilayer perceptron



- **Niet behandeld worden:**
  - convolution neural network (CNN)
  - recurrent neural network (RNN)

3

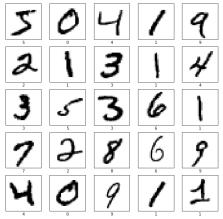
Ik ga je net genoeg materiaal geven om gevaarlijk te zijn, en er zullen enkele oefeningen zijn, zodat je daadwerkelijk wat zelfvertrouwen kunt krijgen en deze technieken daadwerkelijk kunt toepassen.

Oefening

Deep Learning

- hello deep learning -

1. open Jupiter notebook **Hello Deep Learning OEFENING**
   1. installeer tensorflow (versie 2)
      - via de Anaconda Prompt
      - conda install tensorflow
   2. ANN voor het classificeren van handgeschreven cijfers (*mnist*)
2. **type** in de code van het scherm



mnist dataset: http://yann.lecun.com/exdb/mnist/

4

interactieve klassikale oefening:

1. download/clone College 6 introduction Deep learning.
2. open Jupiter notebook **`hello deep learning OEFENING`**.
3. installatie tensorflow (versie 2):
   - versie 2 want daar is dan *keras* bij inbegrepen!
   - Anaconda prompt: **`conda install tensorflow`**
4. volg/type in hetgeen op het scherm van de docent te zien is.

Wanneer uitgevoerd: gefeliciteerd met het uitvoeren van je eerste *artificial neural network* (ANN).

Oefening

Deep Learning

- installatie van *tensorflow* en *keras* -

- **voorwaarde**: Anaconda is geïnstalleerd (Python 3.7+)
- open een **Anaconda command prompt** (*) en installeer tensorflow:

```
conda install tensorflow
```

  - installeert *tensorflow versie 2 (oktober 2019), welke* is inclusief *keras*

- test in Jupiter notebook:

```
import tensorflow as tf
```

```
[2]: import tensorflow as tf
     tf.__version__

[2]: '2.0.0'
```

(*) **Windows**: via *start* menu, **MacOS** of **Linux**: *Terminal* prompt.

5

NB. Gebruik '**tensorflow2**' en niet 'tensorflow1'.

keras is een high-level neural-networks library voor Deep Learning, dat draait op basis van T**ensorFlow** of T**heano**.

De aanbevolen manier is TensorFlow2, omdat Keras is geïntegreerd en niet meer apart geïnstalleerd hoeft te worden.

Deep Learning

- enkele begrippen vooraf -

gradient descient

- een algoritme om fouten over meerdere stappen te minimaliseren

autodiff

- calculus truc voor het vinden van de hellingen in gradient desent

softmax

- algorithm voor het kiezen van de meest waarschijnlijke classificatie gegeven verschillende invoerwaarden

6

softmax:
Ook hier hoeven we niet echt in te gaan op de hardcore wiskunde. *Softmax* wordt door *tensorflow* in de berekeningen gebruikt, en je ziet de term vaak voorbijkomen. Je moet even weten wat het is en waarom het belangrijk is.

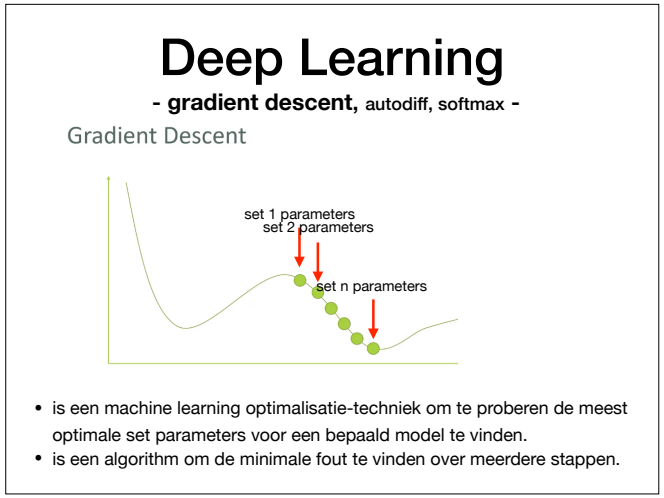
Als je het eindresultaat van een neuraal netwerk hebt, krijg je een verzameling 'gewichten' (*weights*) die aan het eind uit het neurale netwerk komen. Wat doe je ermee?

Hier komt *softmax* in beeld: het zet elk van de gewichten die uit het neurale netwerk komt, om in een waarschijnlijkheid. Bijvoorbeeld wat de kansen zijn dat een afbeelding is een afbeelding van een gezicht, hond, kat of verkeersbord.

Softmax wordt dan aan het eind van het neurale network gebruikt om die laatste uitgangen van de neuronen om te zetten in kansen voor elke klasse. Dan kun je de klasse kiezen die de hoogste kans heeft.

Het is een manier om de uiteindelijke output van jouw neurale netwerk om te zetten in een

concreet antwoord op een classificatieprobleem.



7

Gradient Descent is een machine learning optimalisatie-techniek om te proberen de meest optimale set parameters voor een bepaald probleem te vinden *(College 1)*.

Grafisch: soort kostenfunctie, en we zoeken het zgn. lokale minimum. We kiezen een willekeurig punt met een bepaalde set parameters waarvan we de fout berekenen en we blijven doorgaan, met verschillende parameters in een bepaalde richting te duwen totdat de fout zichzelf minimaliseert.

Deep Learning

- gradient descent, **autodiff,** softmax -

- **autodiff**
  - Dit is een (wiskundige) techniek/calculus-truc om de berekeningen te versnellen van de *gradient descent* op basis van meerdere (model) parameters en error/loss-functies.
  - Tensorflow gebruikt *autodiff* in de berekeningen

8

Autodiff:
En we hoeven niet echt in te gaan op de hardcore wiskunde van hoe *autodiff* werkt. *Autodiff* wordt door *tensorflow* in de berekeningen gebruikt, en je ziet de term vaak voorbijkomen. Je moet even weten wat het is en waarom het belangrijk is.

Gradiënt descent: We moeten weten wat de helling is die we nemen op basis van onze kostenfunctie? Onze foutmeting is misschien de gemiddelde standaardfout (MSE) voor zover we weten.

Als je de helling van een curve probeert te vinden en je hebt te maken met meerdere parameters dan hebben het over gedeeltelijke afgeleiden, en dat blijkt zeer wiskundig intensief en inefficiënt te zijn voor computers om dit te doen.

Autodiff is een (wiskundige)techniek om dat te versnellen. Omdat je in een neuraal netwerk vaak neuronen hebt die heel veel ingangen hebben, maar waarschijnlijk slechts één uitgang of heel weinig uitgangen werkt *autodiff* goed.

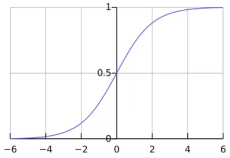
Deep Learning

- gradient descent, autodiff, **softmax** -

softmax
- wordt gebruikt voor de classificatie
  - gegeven een score voor een klasse
  - het produceert een waarschijnlijkheid voor elke klasse
  - de klasse met de grootste waarschijnlijkheid is het antwoord dat je krijgt
- Het is een manier om de uiteindelijke output van jouw neurale netwerk om te zetten in een concreet antwoord op een classificatieprobleem.

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)},$$

x is a vector of input values
theta is a vector of weights



9

softmax:
Ook hier hoeven we niet echt in te gaan op de hardcore wiskunde. *Softmax* wordt door *tensorflow* in de berekeningen gebruikt, en je ziet de term vaak voorbijkomen. Je moet even weten wat het is en waarom het belangrijk is.

Als je het eindresultaat van een neuraal netwerk hebt, krijg je een verzameling ‘gewichten’ (*weights*) die aan het eind uit het neurale netwerk komen. Wat doe je ermee?

Hier komt *softmax* in beeld: het zet elk van de gewichten die uit het neurale netwerk komt, om in een waarschijnlijkheid. Bijvoorbeeld wat de kansen zijn dat een afbeelding is een afbeelding van een gezicht, hond, kat of verkeersbord.

Softmax wordt dan aan het eind van het neurale network gebruikt om die laatste uitgangen van de neuronen om te zetten in kansen voor elke klasse. Dan kun je de klasse kiezen die de hoogste kans heeft.

Het is een manier om de uiteindelijke output van jouw neurale netwerk om te zetten in een

concreet antwoord op een classificatieprobleem.

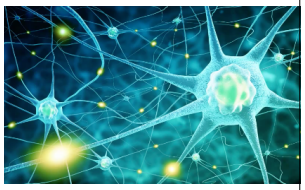
Deep Learning

- Artificial Neural Network (ANN) - concepts

10

biologische neuronen

- neuronen in de hersenschors zijn verbonden via axonen en dendrieten.
- een neuron 'vuurt' naar de neuronen waarmee het is verbonden, wanneer er voldoende van zijn ingangs-signalen zijn geactiveerd.
- heel eenvoudig op individueel neuron niveau, maar lagen van neuronen die op de manier verbonden zijn, kunnen leergedrag opleveren (*emergent behavior*).
- miljarden neuronen, elk met duizenden verbindingen, levert een 'menselijke bewustzijn' op.



Perceptron

Geïnspireerd door de biologische neuronen, introduceerden McCulloch en Pitts in 1943 het concept van perceptron als een kunstmatig neuron, de basisbouwsteen van een ANN. Ze zijn niet alleen vernoemd naar hun biologische tegenhangers, maar zijn ook gemodelleerd naar het gedrag van de neuronen in onze hersenen.

Een **biologisch neuro**n heeft dendrieten om signalen te ontvangen, een cellichaam om ze te verwerken en een axon / axon-terminal om signalen over te dragen naar andere neuronen.

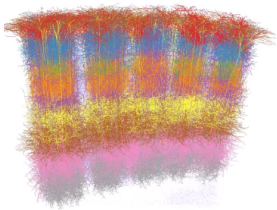
Schaal menselijk brein: miljarden neuronen met elk duizenden verbindingen. Voorlopig voor de AI om van te dromen.

Deep Learning

- Artificial Neural Network (ANN) - concepts

cortical kolommen

- de *cortex*-neuronen lijken te zijn gerangschikt in stapels of *corticale* kolommen die informatie parallel verwerken.
- 'mini'kolommen van plm. 100 neurons zijn georganiseerd in grotere 'hyper'-kolommen. Er zijn ± 100 miljoen mini-kolommen in je cortex.
- dit is toevallig hetzelfde als hoe de videokaart (GPU) werkt…



(credit: Marcel Oberlaender et al.)

11

De rol van GPU's is niet toevallig:
Het heeft een aantal verschillende eenvoudige, zeer kleine verwerkingseenheden die verantwoordelijk zijn voor de berekeningen van de pixels op jouw scherm. En dat blijkt toevallig een zeer nuttige architectuur te zijn om na te bootsen hoe de hersenen werken

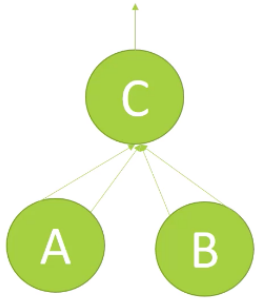
---

Deep Learning

- Artificial Neural Network (ANN) - concepts

de eerste artificial neuron

- 1943
- een *articial* neuron 'vuurt' als meer dan N input connecties actief zijn
- logische constructies OR, AND en NOT te maken, afhankelijk van het aantal verbindingen tussen de neurons.



C = A OR B
als de drempel is dat 2 of meer inputs actief zijn

12

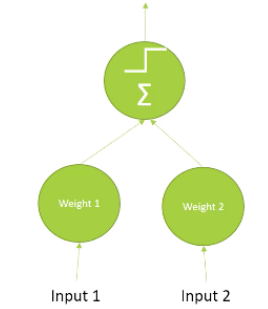
13

Deep Learning

- Artificial Neural Network (ANN) - concepts

linear threshold unit (LTU)

- 1957
- voegt gewichten toe aan de ingangen
- resultaat is gegeven door een stap-functie



tel op de producten van de ingangen en hun gewichten. Resultaat als de som ≥ 0.

In 1957 is voortgebouwd op het idee en werd iets gemaakt dat genoemd is de **Linear Threshold Unit**, of kortweg LTU.

Dus in plaats van alleen een eenvoudige aan en uit-schakelaar, hebben we nu de mogelijkheid om op elk van die ingangen ‘gewichten’ te hangen, zodat je steeds opnieuw kunt afstemmen. Dit werkt meer in de richting van ons begrip van de biologie.

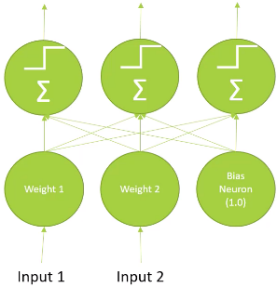
14

Deep Learning

- Artificial Neural Network (ANN) - concepts

Perceptron

- een laag van LTU’s
- een perceptron kan leren door gewichten te versterken die leiden tot correct gedrag tijdens de training
- ook dit heeft een biologische basis, waar een ‘gezegde rondgaat van “cells that fire together, wire together”.



Nu beginnen we in te gaan op dingen die echt kunnen leren: een perceptron. Met een Perceptron hebben we al een systeem dat daadwerkelijk kan leren. Je kunt echt proberen de gewichten te optimaliseren.

Misschien is dit van toepassing op classificaties, waarin een perceptron een afbeelding probeert te classificeren in een of van de drie dingen.

Een ander ding dat we hier hebben geïntroduceerd, is iets dat het **bias-neutron** daar rechts wordt genoemd. En dat is iets om bij de wiskunde-resultaten soms een vaste, constante waarde toe te voegen.

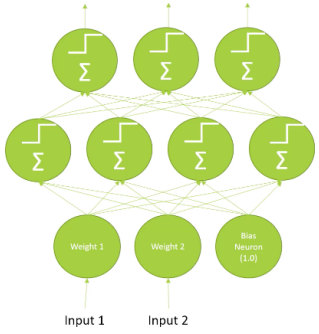
Geïnspireerd door de biologische neuronen, introduceerden McCulloch en Pitts in 1943 het concept van perceptron als een kunstmatig neuron, de basisbouwsteen van een ANN. Ze zijn niet alleen vernoemd naar hun biologische tegenhangers, maar zijn ook gemodelleerd naar het gedrag van de neuronen in onze hersenen.

Deep Learning

- Artificial Neural Network (ANN) - concepts

Multi-layer *Perceptrons*

- toevoegen van 'hidden' layers'
- de hoeveelheid verbindingen geeft de mogelijkheid om de gewichten tussen elke verbinding te optimaliseren
- dit is wat genoemd wordt 'Deep Neural Network'.
- training ervan is lastiger, omdat het systeem complexer is geworden.



15

Meerdere perceptions, elk bestaande uit een laag van LTU's, in weer nieuwe lagen inbrengen -> deep learning network.

input layer -> 1 of meerdere *hidden* layers -> output layer

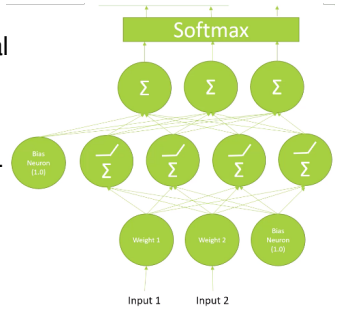
Dus hoewel we hier slechts een handvol kunstmatige neuronen hebben, kunt u zien dat er **veel verbindingen tussen hen** zijn. Dit geeft de mogelijkheid om de gewichten tussen elke verbinding te optimaliseren. Het maakt de training wel lastiger wegens de complexiteit.

Deep Learning

- Artificial Neural Network (ANN) - concepts

Een modern Deep Neural Network

- vervanging van de stap-functie alternatieve activerings-functies ('relu').
- toepassen van *softmax* op de uitvoer.
- training met gebruik van *gradient descent*.

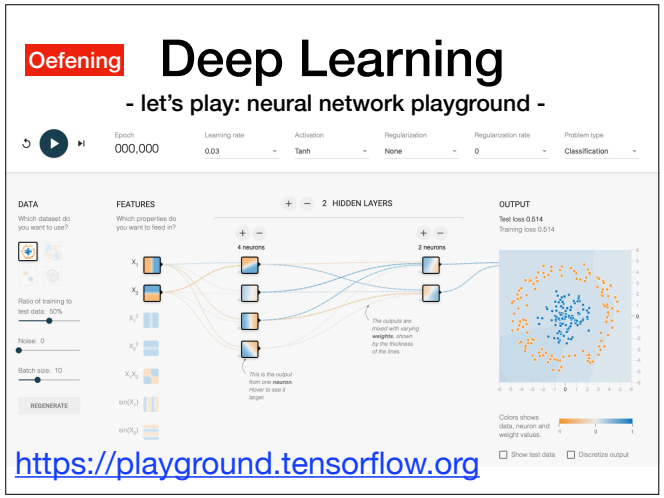


16

Hierbij is vrijwel de hele geschiedenis in een notendop gegeven, wat geleid heeft tot de huidige moderne deep learning networks.

Je hebt heel eenvoudige bouwstenen. Maar wanneer je deze bouwstenen op interessante manieren samenbrengt, gebeuren zeer complexe en soms mysterieus dingen.

Dat is een van de redenen wat de deep learning technologie zo interessant maakt maar ook ongrijpbaar ("zelf de experts weten niet wat er gebeurd").



Houd in gedachten dat dit een **binair classificatieprobleem** is: het is blauw of oranje, dus uiteindelijk houden we gewoon een enkel signaal over, en dat is wat in de uitvoer komt (rechts in de figuur).

RECHTS in figuur:
We hebben hier een aantal punten en die in het midden zijn blauw en die aan de buitenkant zijn oranje. Ons doel is om een neuraal netwerk te creëren dat, zonder enige voorkennis, er daadwerkelijk achter kan komen of een bepaald punt blauw of oranje moet zijn en met succes kan voorspellen welke classificatie het moet zijn. Denk er wel aan: dit zijn onze trainingsgegevens.

MIDDEN in figuur.
Hier is een diagram van het neurale netwerk zelf en we kunnen hiermee spelen. We kunnen het manipuleren. We kunnen lagen toevoegen, lagen verwijderen, etc.

LINKS in figuur:
De gegevensset waarmee we willen spelen. De initiële keuze is de 'Circle' dataset, zoals in figuur is aangegeven.
De input waarden zijn de X en Y coördinaten van de datapunten.
Dit neurale netwerk krijgt een datapunt om te classificeren. Gegeven de twee waarden: de horizontale positie en de verticale positie (x, y). Initieel zijn alle gewichten even zwaar voor zowel de horizontale als verticale positie.

PLAY
Druk op 'play'knop en bekijk wat er gebeurd is aan de rechterkant.
Rond epoche 208: je kunt zien dat er een oplossing is gevonden. Punten in het midden: allemaal blauw, die erbuiten allemaal oranje.
'Hover' over de lijnen: verschillende gewichtswaarden zijn gemaakt. Vanuit input naar 4-neuronen laag redelijk zwak.
Hover over de bovenstae neuron in het midden: deze neuron zegt dat hij de dingen een beetje zwaarder wil wegen in deze hoek, oké? En dingen die zijn zoals in de linker benedenhoek, niet zo veel.

Houd in gedachten dat dit een **binair classificatieprobleem** is: het is blauw of oranje, dus uiteindelijk houden we gewoon een enkel signaal over, en dat is wat in de uitvoer komt (rechts in de figuur).

PLAY
Druk op 'play'knop en bekijk wat er gebeurd is aan de rechterkant.

Voorbeeld play-session:
Wegens enige willekeur factoren kan het er van *run* tot *run* anders uitzien.

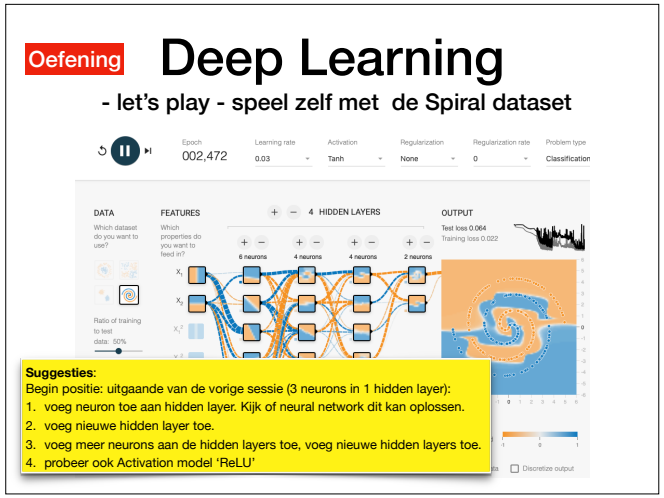
1. **Rond epoche 209**: je kunt zien dat er een oplossing is gevonden: Punten in het midden: allemaal blauw, die erbuiten allemaal oranje.
2. **'Hover' over de verbindingen**: verschillende gewichtswaarden zijn gemaakt. Vanuit input naar 4-neuronen laag redelijk zwak.
3. **'Hover' over de bovenste neuron in het midden**: deze neuron zegt dat hij de dingen een beetje zwaarder wil wegen in rechter-bovenhoek en minder zwaar in de linker-bovenhoek. Bekijk ook de keuze gewichten van de andere neurons.
4. **'Hover' over de rechter hidden-layer neurons**: je krijgt de 'blobby things' in de uitvoerlaag, waar een soort boost gegeven is aan dingen aan de rechterkant of linkerkant.
5. **Resultaat aan rechterkant**: combinatie van de hidden-layer neurons resulteert in de OUTPUT afbeelding aan de rechterkant van de figuur.

Hebben we eigenlijk wel een deep neural netwerk nodig om deze classificatie op te lossen?
Een optimalisatie is om lagen te verwijderen en te kijken of je ermee wegkomt. Misschien hebben we niet eens deep neural netwerk nodig.
Dit is een eenvoudig probleem: dingen in het midden zijn blauw, die aan de buitenkant zijn oranje.

Mogelijkheden om te doen:
- Verwijder een van de neuronen uit de rechter hidden layer (*klik op '-'*): Hit 'play' -> ja, het kan het probleem nog steeds oplossen (epoche ±200).
- Verwijder een van de layers (*klik op '-' naast hidden layer*): Hit 'play', en ja, het kan het probleem nog steeds oplossen (epoche ±200). Dus dit is niet eens een multi-layer

perceptron, maar een enkele perceptron.

- Kan het met minder neurons in de hidden-layer? Dat hangt ervan af: een pass-through neuron (geen voorkeur in gewichten) kan je weghalen. Anderen? Speel ermee, zolang het probleem van classificatie maar opgelost wordt.
  - Het blijkt dat je met 3 neurons dit probleem kan oplossen. Dat is alles wat nodig is om dit probleem te doen. Ik bedoel, vergelijk dat met de miljarden neuronen die in je hoofd bestaan.
  - Nu kunnen we waarschijnlijk niet wegkomen met minder dan dat. Haal nog een neuron weg (2 neuronen) en kijk wat er gebeurt. Yip, dat gaat gewoon niet gebeuren.



19

Speel zelf met de **Spiral dataset**: deze is veel lastiger om op te lossen.
Ik moedig je echt aan om er gewoon mee te spelen en te kijken wat voor soort resultaten je kunt krijgen. Het spiraalpatroon is met name een interessant probleem.

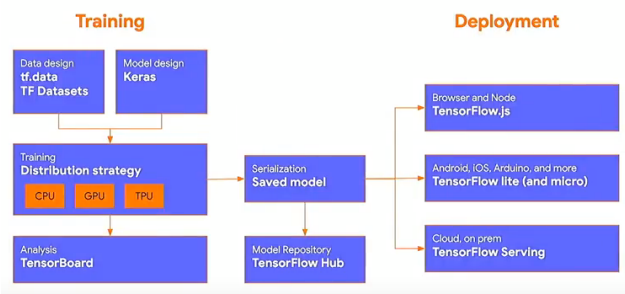
Suggesties:
Begin positie: uitgaande van de vorige sessie (3 neurons in 1 hidden layer):

1. voeg neuron toe aan hidden layer. Kijk of neural network dit kan oplossen.
2. voeg een nieuwe hidden layer toe.
3. voeg meer neurons aan de hidden layers toe.
4. Speel met andere parameters, bijv. voor de Activation: 'ReLU' is populair.

Op gegeven moment wordt het heel moeilijk om intuïtief te begrijpen wat er speelt in het neurale netwerk. Dit wordt een beetje griezelig.

Deep Learning

- tensorflow2 en keras -



versie 2 sinds oktober 2019

- *keras* is onderdeel van *tensorflow*.
- het getrainde model kan je bewaren, waardoor het opgepakt kan worden in browser, microcontrollers,…

20

Adafruit: TensorFlow Lite op hun microcontrollers (met o.a. CircuitPython) op basis van een getraind deep learning model. Zie verder https://learn.adafruit.com/search?q=tensorflow

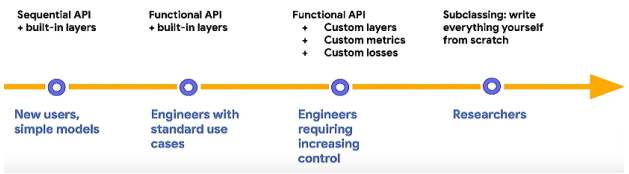
Training model: op krachtige computers, wel/niet met GPU's.
Deployment op resource-beperkte devices, zoals in de browser, op microcontrollers en microcomputers en in de Cloud.

Deep Learning

- tensorflow2 en keras -

Model building: from simple to arbitrarily flexible

Progressive disclosure of complexity



- Josh Gordon - **Introduction TensorFlow 2.0**: https://youtu.be/9pHCch_d9hE, november 2019
- **TensorFlow tutorials**: https://www.tensorflow.org/tutorials

21

Twee bronnen voor een goede introductie van en verder leren over TensorFlow2.

NB. Een **tensor** is een wiskundige object dat beschouwd kan worden als een generalisatie van getallen (scalars), vectoren en matrices. Zie verder https://nl.wikipedia.org/wiki/Tensor

Deep Learning

- bronnen -

1. A neural network playground: https://playground.tensorflow.org, gezien 2019-1215
2. Tensorflow, https://www.tensorflow.org/tutorials/, gezien 2019-1215
3. Frank Kane, Machine Learning, Data Science and Deep Learning with Python, video-tutorial distributed by Manning Publications, 2019 (*commercieel*).
4. Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd edition, 2019, O'Reilly. (*commercieel*). aanbevolen ML-boek
5. Swamynathan, Manohar, Mastering Machine Learning with Python in Six Steps, *A Practical Implementation Guide to Predictive Data Analytics Using Python*, Apress, 2017 (*commercieel*).

Opmerking: [Aurélien Géron] *Neem 2de editie want daar staat in tensorflow 2. Neem dus niet de 1ste druk (tensorflow 1, die werkt heel anders met keras).*

OPDRACHT

Deep Learning

- Titanic deep learning -

Open Jupiter notebook **`Titanic Deep Learning OEFENING`**

- zet de gegeven code op de juiste volgorde d.w.z. in opeenvolgende code-cellen.
- gebruik 1ste oefening als referentie.
- bestudeer materiaal op tensorflow (tutorials)

Machine Learning

- logistiek en toetsing -

- **8 januari 2020 géén college.**
  - Gelegenheid puntjes-op-de-i te zetten voor de eindtoets casus oplevering (8 januari)
  - ELO: 'inleveropdracht' - **voor ieder individueel !**
- **Presentatie eindcasus resultaten op vrijdag 10 januari 2020**
  - details: zie studiehandleiding
  - github: zie folder 'assignment casus'
  - ELO: folder 'Eindtoets - casus'

24

25

Succes en fijne feestdagen en jaarwisseling
Stefan, Peter