Programmeren 8: Machine Learning (Samenvatting Lessen)

Knowledgebase: https://luukftf.github.io/knowledgebase
(code: https://github.com/LuukFTF/knowledgebase)
By: Lucas van der Vegt
2022-03-02

Leerdoelen

Deadlines

- Opdracht 1: Week 3
- Opdracht 2: Week 5
- Eindopdracht Ontwerp: Week 7
- Eindopdracht Uitwerking: Week 9

Index

Resources

https://github.com/HR-CMGT/Machine-Learning-Readinglist

https://github.com/tensorflow/models/tree/master/research/slim

https://storage.googleapis.com/tfjs-examples/webcam-transfer-learning/dist/index.html

Les 1 - Machine Learning Introduction

Tensorflow JS

voorbeelden:

- Tesla
- GPT3
- Google Search
- Youtube Recommendation

Resources Kaggle.com

Teachable Machine

Waarom is ML nu populair

- Rekenkracht
- Data
- Mindset

Disciplines:

- Leren van data
- beeld herkenning
- beeld generen
- spraak begrijpen
- tekst genereren
- tekst begrijpen

Artificial Intelligence
Dumb AI

Machine Learning
Supervised Learning

Deep Learning
Unsupervised Learning

Leren is patronen herkennen in data

Wat heb ik voor data?

Wat wil / kan ik leren van deze data

Welk algoritme of model past hierbij

Data > Algoritme > Model

Data
alles is data, maak er een reeks van getallen voor

Algoritme
De formule of library waarmee je een model maakt

Model
Gegenereerd door algoritme dmv data

Classification
verschillende keuzes

Regression
een waarde

White Box & Black Box
Is het achteraf te begrijpen (Ethiek)

We gaan aan de slag met:

Decision Tree KNN

Neural Network

Linear Regression

Pretrained Model
Al getrained model

- YOLO
- FaceAPI
- PoseNet
- DoodleNet
- ImageClassify

Libraries:

- TensorFlow
- ML5

Ethiek

- Privacy
- White vs Black box

Techstack:

- TensorflowJS (alleen local vanwege privacy)
- Frontend React
- Backend NodeJS

Les 2 - ML5 Introduction

ML5 Introduction

Datasets Kaggle

Training Algoritme Neural Network Linear Regression Decision Tree K-Means

Model Maken door trainen Pre-trained

Examples: Google Quickdraw Autodraw Drawthis (danmacnish)

Pixel Data 1 Color: Grijswaardes per pixel (12x16 px = 192 waardes)

RGB: Rood Groen & Blauw per pixel (12x16 px = 576 waardes)

Convolutional neural networks < *Een Convolutional Neural Network reduceert een afbeelding tot een heel klein stukje data, waar toch nog alle informatie in zit om te weten of een afbeelding een kat of een hond* >

https://www.youtube.com/watch?t=414&v=qPKsVAI_W6M&feature=youtu.be
https://www.youtube.com/watch?v=f0t-OCG79-U

Cat or Dog? (features data)

- hair color
- body length
- height
- weight
- ear length
- claws

Features < *De informatie die we gevonden hebben noemen we "features". Door op te slaan welke combinaties van features bij een "auto" horen kunnen we auto's herkennen* >

Pre-trained models: image classifier

Je kan een bestaand model gebruiken, dat getraind is op features van de meest voorkomende objecten in de wereld.

Hoe zwaarder het model dat je download, hoe beter de accuracy, en hoe meer objecten herkend worden.

ALL ML5 Libraries Image:

- ImageClassifier
- PoseNet
- BodyPix
- UNET
- Handpose
- Facemesh
- FaceApi
- StyleTransfer

- pix2pix
- CVAE
- DCGAN
- SketchRNN
- ObjectDetector

Sound:

- SoundClassification
- PitchDetection

Text:

- CharRNN
- Sentiment
- Word2Vec

Helpers

- NeuralNetwork
- FeatureExtractor
- KNNClassifier
- kmeans

General ML5 Code Load ML5 Library

```
<script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
```

ML5 Status

```
console.log('ml5 version:', ml5.version);
```

ML5 Image Classifier

MobileNet https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet/README.md https://learn.ml5js.org/#/reference/image-classifier

Example

```
const classifier = ml5.imageClassifier('MobileNet', modelLoaded)

function modelLoaded() {
    makePrediction()
}

function makePrediction() {
    classifier.classify(document.getElementById('image'), (err, results) => {
```

```
        console.log(results)
    })
}
```

```
<image id="image" src="./capibara.png"/>
```

ML5 Feature Extraction

https://learn.ml5js.org/#/reference/feature-extractor https://github.com/ml5js/ml5-library/tree/main/examples/javascript/FeatureExtractor/FeatureExtractor_Image_Classification https://www.youtube.com/watch?v=eeO-rWYFuG0 https://github.com/HR-CMGT/Machine-Learning-Readinglist/tree/master/extractfeatures

1. Load MobileNet image model
2. Re-train model with own data
3. Save new own Model
4. Load new own Model
5. Use new own Model

Opdracht praktijk (week 2) Bouw een photo hunting app voor mobile waarbij de speler op pad moet gaan om foto's te maken.

Maak eerst alles werkend met de imageClassifier

Als je eigen images wil kunnen herkennen, gebruik je de featureExtractor

https://github.com/HR-CMGT/PRG08-2021-2022/tree/main/week2

Les 3 - ML5 Pretrained Models

(opgenomen les: zie teams 2022-02-23)

handpose, bodypose, face, object recognition

- PoseNet
- Facemesh
- FaceApi
- Handpose API
- ObjectDetector
- FeatureExtractor

Inspiration: http://charliegerard.dev
https://unity.com/products/machine-learning-agents
https://www.youtube.com/watch?v=mPbtR4vorgY
https://code.visualstudio.com/docs/other/unity

PoseNet vs imageClassifier

PoseNet

https://learn.ml5js.org/#/reference/posenet

https://cmgt.hr.nl/project/danceflow https://glitch.com/~draw-circle https://github.com/ml5js/ml5-library/blob/main/examples/javascript/PoseNet/PoseNet_webcam/sketch.js https://www.youtube.com/watch?v=DMebdxAp0j0 https://glitch.com/edit/#!/pong-game-canvas

Maak de pose detector. Vul de "poses" variabele elke keer dat posenet een pose vindt.

```
// Maak een poseNet
constposeNet= ml5.poseNet(video,"single", modelLoaded)

// Berichtje dat het model is geladen
functionmodelLoaded(){
    console.log('Model Loaded!');
    // Luister naar 'pose' events. Zo lang het model niet geladen is zullen er
geen events zijn.
    poseNet.on('pose',(results)=>{
        pose = result;
        console.log(pose)
    });
}
```

Output

```
[
  {
```

```
    pose: {
      keypoints: [{ position: { x, y }, score, part }, ...],
      leftAngle: { x, y, confidence },
      leftEar: { x, y, confidence },
      leftElbow: { x, y, confidence },
      ...
    },
  },
];
```

Output Multipose

```
[
  {
    pose: {
      keypoints: [{ position: { x, y }, score, part }, ...],
      leftAngle: { x, y, confidence },
      leftEar: { x, y, confidence },
      leftElbow: { x, y, confidence },
      ...
    },
  },
  {
    pose: {
      keypoints: [{ position: { x, y }, score, part }, ...],
      leftAngle: { x, y, confidence },
      leftEar: { x, y, confidence },
      leftElbow: { x, y, confidence },
      ...
    },
  },
];
```

Pose Draw

```
function drawCameraIntoCanvas() {
    // teken het video element in een canvas element
    ctx.drawImage(video, 0, 0, 640, 480)

    // nu kunnen we de keypoints en bones ook in het canvas tekenen
    drawKeypoints()
    drawSkeleton()
    window.requestAnimationFrame(drawCameraIntoCanvas)
}

drawCameraIntoCanvas()
```

Maak een `<canvas>` element om het camera beeld en de poses in te kunnen tekenen.Maak een requestAnimationFrame functie die 60x per seconde wordt uitgevoerd. Hier teken je telkens het camerabeeld in, en daaroverheen de laatst gedetecteerde pose. Het `<video>` element kan je nu onzichtbaar maken.

FaceApi

https://learn.ml5js.org/#/reference/face-api

https://www.youtube.com/watch?t=765&v=Hd6PG9R3r6c&feature=youtu.be

Facemesh

https://learn.ml5js.org/#/reference/facemesh

Handpose API

https://learn.ml5js.org/#/reference/handpose

landmarks

ObjectDetector

https://learn.ml5js.org/#/reference/object-detector

FeatureExtractor

https://learn.ml5js.org/#/reference/feature-extractor

EmotionClassifier

https://brendansudol.com/writing/tfjs-emotions https://www.codeproject.com/Articles/5276827/AI-Age-Estimation-in-the-Browser-using-face-api-an https://www.codeproject.com/Articles/5276822/Pre-Trained-AI-Emotion-Detection-With-face-api-and

Opdracht Week 3 Deliverable

https://github.com/HR-CMGT/PRG08-2021-2022/tree/main/week3

1. Bedenk een concept voor het werken met gezichtsuitdrukking herkenning, lichaamspose herkenning, handpose herkenning, object detectie, of de image feature herkenning uit week 2. (Dat is de imageClassifier waar je je eigen images aan hebt toegevoegd)
2. data uit met javascript en geef feedback aan de gebruiker via de UI.
3. Bouw een eenvoudige UI voor dit concept met HTML en CSS. De gebruiker hoeft dus niet in de console te kijken.

Les 4 - KNN

https://github.com/HR-CMGT/PRG08-2021-2022/tree/main/week4 https://codepen.io/Qbrid/pen/OwpjLX https://github.com/NathanEpstein/KNear https://burakkanber.com/blog/machine-learning-in-js-k-nearest-neighbor-part-1/

Reduceer data zoveel mogelijk

dichtbijzijnste vinden

2 4 8

$$ 2-4=2 \ 4-8=4 $$

2 is het dichtstbij

wortel van de kwadraat = de min wegwerken

$$ \sqrt{(2-4)^2} = 2 \ \sqrt{(8-4)^2} = 4 $$

KNN

Hoeveel Dichtbijzijnste Getal

Algoritme, geen neural network

Drie dimensies

$$ \sqrt{(a-b)^2 + (a-b)^2 + (a-b)^2} $$

Vijf dimensies
$$ \sqrt{(a-b)^2 + (a-b)^2 + (a-b)^2 + (a-b)^2 + (a-b)^2} $$

Supervised Learning het algoritme wordt getrained met bestaande data die al labels heeft.

kNear KNNClassifier

PoseNet + KNN

1 lange array maken

PROs

- Veelzijdig, alle soorten data
- Geen last van slechte scheiding

CONs

- Trainingsdata heb je altijd nodig (data = het model)
- Alle opties moeten een keer voorbij gekomen zijn

Normaliseren Groot verschil tussen schalen in verschillend data verkleinen

Inladen Library

```
<script src="knear.js"></script>
```

https://github.com/HR-CMGT/PRG08-2021-2022/blob/main/week4/knear/js/knear.js

Aanmaken Algoritme

```
const k = 3
const machine = new kNear(k)
```

Learn Data

```
machine.learn([6.2, 20, 9], 'cat')
machine.learn([18,9.2,8.1,2],'cat')
machine.learn([20.1,17,15.5,5],'dog')
machine.learn([17,9.1,9,1.95],'cat')
machine.learn([23.5,20,20,6.2],'dog')
machine.learn([16,9.0,10,2.1],'cat')
machine.learn([21,16.7,16,3.3],'cat')
```

Classify

```
let prediction = machine.classify([7,18,7])
console.log(`I think this is a ${prediction}`)
```

https://github.com/HR-CMGT/PRG08-2021-2022/tree/main/week4/knear

---

Les 5 - Decision Tree

Inleveropdracht

- CSV data laden
- Decision Tree tekenen
- Voorspelling doen
- Accuracy uitrekenen

https://github.com/0HR-CMGT/PRG08-2021-2022/tree/main/week5 https://github.com/HR-CMGT/PRG08-2021-2022/blob/main/week5/inleveropdracht.md

Data > Algoritme > Model

ML5 Gebruik pre-trained modellen om poses, gezichten, etc te herkennen

KNN Vergelijk nieuwe data met de bestaande training dat

Decision Tree Tekent een beslisboom op basis van training data

Data: CSV files (Kaggle)

Decision Tree Algoritme

https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

Heeft voorkeur voor de meeste informatie

groter en kleiner dan een bepaald getal

grafiek, alle vragen bedenken om alles te kunnen classificeren

White Box Algoritme Na de training kan je goed zien waarom een bepaalde voorspelling gemaakt wordt.

pseudocode:

```
data = titanicdata.csv
tree = new DecisionTree()
model = tree.train(data)

prediction = tree.predict(Jack, Male, 22)

// output: DIED
```

libraries
https://github.com/HR-CMGT/PRG08-2021-2022/tree/main/week5/oefening/libraries

Papa Parse https://www.papaparse.com/

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.js">
</script>
```

DecisionTree
https://github.com/lagodiuk/decision-tree-js

```
import { DecisionTree } from "./libraries/decisiontree.js
```

VegaTree
https://vega.github.io/vega/examples/tree-layout/

```
import { VegaTree } from "./libraries/vegatree.js
```

Data analyseren Wat is het label?
Welke data is relevant?

https://www.kaggle.com/datasets

CSV Data

```
"Name","Toothed","Hair","Breathes","Legs","Species"
"Dog",1,1,1,1,"Mammal"
"Cat",1,1,1,1,"Mammal"
"Frog",1,0,1,0,"Reptile"
"Turtle",0,1,1,1,"Mammal"
"Bird",1,1,1,1,"Mammal"
"Human",1,1,1,1,"Mammal"
"Lizard",1,0,0,0,"Reptile"
"Crocodile",1,0,1,0,"Reptile"
"Lion",1,1,1,1,"Mammal"
"Snake",0,1,1,"Reptile"
```

Setup Constants

```
const csvFile = "./data/animals.csv"
const trainingLabel = "Species"
const ignoredColumns = ['Name']
```

Inladen csv data / log data

```

```

```
loadData() {
  Papa.parse("data/animals.csv", {
        download:true,
        header:true,
        dynamicTyping:true,
        complete: results => console.log(results.data)
    })
  }
}
```

Train model & Teken Decision Tree

```
function trainModel(data) {
    let decisionTree = new DecisionTree({
        ignoredAttributes: ignoredColumns,
        trainingSet: data,
        categoryAttr: trainingLabel
    })

    // Teken de boomstructuur - DOM element, breedte, hoogte, decision tree
    let treeImage = new VegaTree('#view', 800, 400, decisionTree.toJSON())
}
```

Training & Test Data

Dataset opdelen

- 80% Training
- 20% Testing

(Eerst data husselen)

```
data.sort(() => (Math.random() - 0.5)
```

```
let data      = Papa.parse("titanic.csv")
let trainData = data.slice(0, Math.floor(data.length * 0.8))
let testData  = data.slice(Math.floor(data.length * 0.8) + 1)

let tree = new DecisionTree()
let model = tree.train(trainData)
```

De testData heeft al een "Species" label

```

```

```
let testAnimal = testData[0]
console.log(testData[0])
```

Prediction

```
let animal = {Name:"Rat", Toothed: 1, Hair: 1, Breathes: 1, Legs: 1}
let prediction = decisionTree.predict(animal)
console.log(`${animal.Name} is of the ${prediction} species`)
```

Test op 1 uit testdata

```
let prediction = model.predict(testData[0])

if(prediction === testAnimal.species) {
  console.log("CORRECTE  VOORSPELLING!")
  }
```

Test hele testdata set

```
let amountCorrect = 0

for(let testAnimal of testData) {
  if(model.predict(testAnimal) === testAnimal.survived) {
    amountCorrect++
    }
  }
```

Accuracy berekenen

```
let accuracy = amountCorrect / testData.length
```

Bij testen moet je het juiste antwoord (species = Mammal of Reptile) niet meegeven!

```
let animalWithoutLabel= Object.assign({}, testData[0])
delete animalWithoutLabel.species

let prediction = model.predict(personWithoutLabel)
if(prediction === testAnimal.species) {
    console.log("Correct prediction!")
  }
```

Doe de prediction met een kopie van het testpersoon, zonder het label

Confusion Matrix

- False Negatives
- False Positives

Als een voorspelling fout is, maakt het dan nog uit wat er precies fout is? (verbeteren / bepalen welk van de 4 getallen belangrijk zijn)

Met een Confusion Matrix krijg je nog wat meer inzicht in je accuracy. Je gaat nu ook bijhouden waarom een voorspelling goed of fout was. Bijvoorbeeld bij de mushrooms:

```
if(prediction == "e" && label == "p") {
    console.log("🍄 predicted edible, but was actually poisonous!🤮 🏷️")
}
if(prediction == "p" && label == "e") {
    console.log("🍄 predicted poisonous, but was actually edible! 😬")
}
```

```
<div>
    <h4>Confusion Matrix</h4>
    <table id="confusion">
        <tr>
            <td></td>
            <td>Predicted true</td>
            <td>Predicted false</td>
        </tr>
        <tr>
            <td>Actually true</td>
            <td> - </td>
            <td> - </td>
        </tr>
        <tr>
            <td>Actually false</td>
            <td> - </td>
            <td> - </td>
        </tr>`
    </table>
</div>
```

Decision Tree Advanced

- tree dept
- overfitting (te specifiek trainen)

PROS White box: duidelijke visualisatie van beslissingen.

- Onbelangrijke features komen onderaan in de tree te staan.

- Een Excel sheet kan je zonder veel voorbereiding rechtstreeks in het algoritme gooien.
- Bij KNN moesten we de data altijd bewaren. Hier bewaar je alleen de tree (het model).
- Grote hoeveelheid data maakt het uiteindelijke model niet langzamer.

CONS

- Overfitting: de tree leert vooral de training data goed herkennen.- Bias: het algoritme heeft een voorkeur voor classes waar meer voorbeelden van zijn. (Meer katten dan honden in trainingdata).
- Het algoritme leert niet persé de overkoepelende relaties / doel van de classificatie. (greedy algorithm: https://en.wikipedia.org/wiki/Greedy_algorithm)

Opdracht Data:

Bij het inlezen van de data moet je controleren wat het label is waarop we willen trainen. Ook moet je even kijken of er kolommen zijn die niet relevant zijn bij het trainen.

- Bij 'mushrooms.csv' is het label "class", en de inhoud is "p" (poisonous) en "e" (edible.)
- Bij 'diabetes.csv' is het label "Label" en de inhoud is "1" (diabetes) en "0" (geen diabetes)
- Bij 'titanic.csv' is het label "Survived" en de inhoud is "1" (survived) en "0" (not survived). Ook heeft de titanic dataset veel kolommen die misschien niet relevant zijn: "Name", "Cabin", "PassengerId", "Ticket", "Fare". Je kan kijken of je algoritme beter wordt als de deze kolommen negeert.

Extra Uitdaging: https://github.com/HR-CMGT/PRG08-2021-2022/blob/main/week5/inleveropdracht.md#extra-uitdaging

---

Les 6

Neural Network

Hidden Layers

Regression Classification

Tensorflow

ChartJS Scatterplot

Map Data (data to axis)

Randomize data

Machine Learning Explained in 100 Seconds

https://www.youtube.com/watch?v=PeMlggyqz0Y

Machine Learning is the process of teaching a computer how perform a task with out explicitly programming it. The process feeds algorithms with large amounts of data to gradually improve predictive performance.

Goals

Classification Prediction (Regression)

Data

Aquire Data Better data > better results (Garbage in Garbage out)

Data scientist

Feature engineering

Dataset:

- Training set
- Test set

Algoritms

Linear Regression Decision Tree Convolutional Neural Network

Error Function (Loss function) Classification: Accuracy Regression: Mean Absolute Error

Languages: Python R Julia

Frameworks: NumPy scikit learn PyTorch Tensorflow Pandas

Model

Input > Prediction

Can be embedded or deployed to cloud.

TensorFlow.js Quick Start

https://www.youtube.com/watch?v=Y_XM3Bu-4yc

Webbased Machine Learning

https://playground.tensorflow.org https://www.kaggle.com/jeffd23 https://developers.google.com/machine-learning/crash-course/

Dataset MNIST https://ml4a.github.io/demos/confusion_mnist/