

## TODO

---

- 295 KNN-luokitin
- 299 Viikon 9 sisältö
- 302 Hierarkinen klusterointi
- 303 klusterointin tehtävät
- 305 Viikon 10 tavoitteet
- 307 neuroverkkojen tehtävät
- 309 Viikon 11 tavoitteet
- 311 CNN-harjoitukset
- 313 Viikon 12 tavoitteet
- 315 RNN-harjoitukset
- 317 Keksi aihe harjoitustyölle
- 319 Kertausharjoituksia
- 323 Harjoituksia liittyen TÄ:n tulevaisuuteen

# 4 CO22KETV1400

## Tekoälyn ohjelmointi

Janne Koponen

7. helmikuuta 2022

1	Yleistä kurssista	8
2	Kurssiympäristö – Viikko 2	11
3	Python– Viikko 2	16
4	Mitä on tekooaly? – Viikko 2	25
5	Kirjastoja – Viikko 2–3	33
•	NumPy	34
•	pyplot	53
•	Keras	58
•	scikit-learn	63
•	SciPy	66
•	Pandas	67
•	Orange	71
•	Harjoituksia	72

6	Tekoälyn menetelmiä – Viikko 3	74
•	Optimointi	76
•	Koneoppiminen	88
•	Regressio	94
•	Luokittelu	98
•	Klusterointi	106
•	Neuroverkot	109
•	Harjoituksia	127
7	pyplot – Viikko 4	128
•	Harjoituksia	148
8	Lineaarialgebraa – Viikko 5	149
•	Vektorit	151
•	Tason ja $\mathbb{R}^2$ :n samaistus	153
•	Muita avaruuksia, $\mathbb{R}^n$	155
•	Matriisit	185
•	Harjoituksia	212

9	Aineiston esikäsittely – Viikko 6	214
	• Aineiston lukeminen	218
	• Virheellisten mittaustulosten käsittely	221
	• Kategoristen tietojen käsittely	223
	• Poikkeavien havaintojen poisto	224
	• Jako opetus-, testi- (ja validointiaineistoon)	227
	• Piirteiden skalaus	230
	• Dimensioiden vähentäminen	231
	• Sovitus malliin ja testaus	232
	• Harjoituksia	237
10	Regressio – Viikko 7	239
	• Harjoituksia	265
11	Pääkomponenttianalyysi (PCA)– Viikko 8	266
	• Harjoituksia	283

12	Luokittelu	284
	• Tukivektorikone (SVM) – Viikko 9	286
	• K-lähimmän naapurin menetelmä (KNN)	295
	• Harjoituksia	296
13	Klusterointi – Viikko 11	298
	• K-means klusterointi	301
	• Hierarkinen klusterointi	302
	• Harjoituksia	303
14	Neuroverkot – Viikko 12	304
	• Harjoituksia	307
15	Konvolutioalaiset neuroverkot – Viikko 13	308
	• Harjoituksia	311

16	Rekurrentit neuroverkot – Viikko 14	312
	• Harjoituksia	315
17	Kertaus – Viikko 15	316
	• Harjoituksia	319
18	Tekoälyn tulevaisuus – Viikko 16	320
	• Harjoituksia	323

## 1. Yleistä kurssista

### Kurssin tiedot (1/2)

- Koodi: 4 CO22KETV1400
- Nimi: Tekoälyn ohjelointi
- Laajuuus: 5 op
- Sisältö:
  - Tekoälysovelluksen toteutus Python-kielellä
  - Datan esikäsittely
  - Datan visualisointi
  - Regressio
  - Pääkomponenttianalyysi
  - Klusterointi ja luokittelu
  - Neuroverkot ja syväoppiminen
  - Kirjastoja: NumPy, pyplot, scikit-learn, Tensorflow/Keras jne.

### Kurssin tiedot (2/2)

- Lähtötaso: Python-ohjelmoinnin perusteet
- Tavoitteet:
  - Osaat hyödyntää Python-kirjastojen valmiita tekoälyalgoritmeja
  - Ymmärrät joidenkin tärkeimpien tekoälyalgoritmien toiminnan pääperiaatteet
  - Osaat esikäsitellä datan koneoppimiseen sopivaksi
  - Hallitset datan visualisoinnin perusteet
- Toteutus: Videoidut luentotunnukset, itsenäinen opiskelu, harjoitustöitä
- Oppimateriaali: Luentokalvot ja kirjastojen dokumentaatio
- Arvioinnin perusteet: Palautettavat harjoitustehtävät

50% → 1

60% → 2

70% → 3

80% → 4

90% → 5

## 2. Kurssiympäristö – Viikko 2

### Viikon sisältö

- ① Kurssilla käytettyjen ympäristöjen asentaminen
- ② Python-kielen kertaaminen
- ③ Tarvittavien kirjastojen asennus ja testaus
- ④ Tutustuminen tärkeimpiin kirjastoihin
- ⑤ Tekoäly-termi
- ⑥ Joitakin tekoälykirjastoja (jatkuu viikolla 3)

### Python-ympäristö

- Anaconda
- Python 3.7 tai uudempi
- Spyder<sup>1</sup>
- Pandas
- NumPy
- Matplotlib
- SciPy
- scikit-learn
- TensorFlow
- Keras
- Näin asennat Anacondan ja tarvittavat kirjastot

---

<sup>1</sup>Voit toki käyttää muutakin ympäristöä

Näiden ohjelmien avulla voit testata ympäristösi toiminnan:

- NumPy
- Tensorflow/Keras
- pyplot
- Pandas
- scikit-learn
- Video - Kirjastojen testaus

## 2. Kurssiympäristö – Viikko 2

### Tehtävien tarkastusympäristö

- Tarkastusympäristö vaatii Python-version 3.7 tai uudemman.
- Testit käyttävät `unittest`-modulia, asenna se tarvittaessa.
- Ensimmäisten tehtävien paketti löytyy [Moodlesta](#).
- Toinen tehtäväpaketti tulee myös [Moodleen](#) helmikuun aikana.
- Paketit sisältävät hakemistorakenteen, pohjat ja testit.
- Testien avulla voit testata että ratkaisusi on tehtävänannon mukainen.
- Kurssin lopuksi kasaa kummankin paketin ratkaisusi omiksi paketeikseen.
- Palauta paketit Moodleen
- Arviointi tapahtuu automaattisesti toukokuun aikana.
- Tarkastusympäristön asennus ja käyttö

### 3. Python– Viikko 2

- Kehitystyö aloitettiin 80-luvun lopulla
- Ensimmäinen julkaistu versio 1990
- Pohjana toimi tulkattava ABC-kieli
- Guido van Rossum
- Suunniteltu helposti laajennettavaksi
- Moduleja voi toteuttaa sekä Pythonilla, että myös käännettävillä kielillä
- Toteutettu alusta alkaen tukemaan useita alustoja

- Versio 2.0 julkaistiin vuonna 2000
- Versio 3.0 julkaistiin 2008
- Versiot eivät ole yhteensovivia. ⇒ Käytä aina versiota 3.x!
- Viimeisin versio 3.9.xx
- Python Software Foundation
- Python-tulkki on avoimen lähdekoodin alainen (Python Software Foundation License), mutta kielellä tehdyt ohjelmistot voivat olla suljettuja.

- Tulkattava (Nuitka-kääntäjä)
- Dynaaminen tyyppitys (muuttujan tyyppiä ei tarvitse esitellä)
- Sisennykset merkitsevät (ei muita lohkon alku- ja loppumerkkejä)
- Ei osoittimia, automaattinen roskienkeruu
- Suuri standardikirjasto.
- PyPI:ssä yli 200 000 kirjastoa.

## IPython

- Interaktiivinen shell
- Jupyter kernel
- Interaktiiviset demot
- Voidaan upottaa osaksi omaa ohjelmaa
- <https://ipython.org>

#### Esimerkki IPython-sessiosta

```
In [3]: for i in range(10):
....:     print(i, end=" ")
....: print("loppu")
....:
0 1 2 3 4 5 6 7 8 9 loppu
```

```
In [4]: 10+3
Out[4]: 13
```

```
In [5]: i=2
```

```
In [6]: j=4
```

```
In [7]: i**j
Out[7]: 16
```

- Python-ohjelmia voi kirjoittaa periaatteessa millä tahansa tekstitoideilla.
- Virheenetsintä voi olla haasteellista.
- Toimii kuitenkin pienissä projekteissa.
- Muista käyttää python3:a ohjelman suorittamiseksi!

- Interaktiivinen ympäristö
- Useita kieliä: Python, C#, R, Perl, PHP, Octave, Matlab, Bash, PowerShell, Lua, Gnuplot...
- Jupyter Notebook - WWW-pohjainen ympäristö
- Jupyter Hub - Monen käyttäjän palvelin
- JupyterLab, 2018

## Spyder

- Käytetään tällä kurssilla
- Integroitu kehitysympäristö (IDE)
- Editori
- Debuggeri
- IPython-konsoli
- Online help
- History
- Plugineja:
  - Spyder-Notebook: Jupyter notebookien käyttö
  - Spyder-Unitest: Yksikkötestaus
  - Spyder-Terminal: Terminaali (komentorivi)
  - ...

## 4. Mitä on tekooäly? – Viikko 2

#### Definition

Tekoälyllä tarkoitetaan koneen kykyä käyttää perinteisesti ihmisen älyyn liitettyjä taitoja, kuten päätelyä, oppimista, suunnittelemistä tai luomista. (Lähde: Euroopan parlamentti - *Mitä tekoäly on ja mihin sitä käytetään?*)

#### Remark

Määritelmiä on paljon muitakin, esimerkiksi

- *Kaplan et al.* - "System's ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation"
- *McCarthy* - "It is the science and engineering of making intelligent machines..."
- *Turing* - "Systems that act like humans."

## 4. Mitä on tekoäly? – Viikko 2

### Yleinen tekoäly (General AI)

#### Definition

*Yleinen tekoäly tarkoittaa koneen kykyä ratkaista ihmisen kaltaisella päätelyllä mitä tahansa ongelmaa. (Lähde: Advian - *Pieni Suuri Tekoälysanasto*)*

- Hypoteettinen teknologia, ei vielä käytössä
- Lukuisia määritelmiä
- Kyky ratkaista ennalta tuntemattomia tehtäviä
- Uhka?

## 4. Mitä on tekoäly? – Viikko 2

### Suppea tekoäly (Weak AI)

#### Definition

*Suppea tekoäly on tekoäly, joka on luotu ratkaisemaan yhtä ongelmaa.*

- Esimerkiksi shakkia pelaava kone
- Usein tulokset ovat varsin hyviä hyvin rajatun ongelman ratkaisussa.
- Yhdistelemällä saadaan luotua älykkään tuntuisia sovelluksia, esimerkiksi puhetta ymmärtäviä robotteja.

- Automaatiolla tarkoitetaan itsetoimivaa laitetta tai järjestelmää.
- Tekoäly on automaation osajoukko.
- Säätimet
- Perinteiset PLC- ja tietokoneohjelmat.
- Teollisuusautomaatio käsittää esimerkiksi koneiden ja prosessien ohjaukseen.

## 4. Mitä on tekoäly? – Viikko 2

### Apuäly (Augmented intelligence)

- Termiä käytetään korostamaan tekoälyn roolia päätöksenteon tukena.
- Viittaa ihmisen älykkyyden laajentamiseen tekoälyn avulla.
- Esimerkiksi asiantuntijajärjestelmä, joka suosittaa joitain ratkaisua, mutta ihminen tekee lopullisen päätöksen.
- Autonavigaattorin reitin laskenta.

## 4. Mitä on tekoäly? – Viikko 2

### Missä tekoälyä käytetään?

# Tekoäly

## Käyttö ja mahdollinen käyttö arjessa

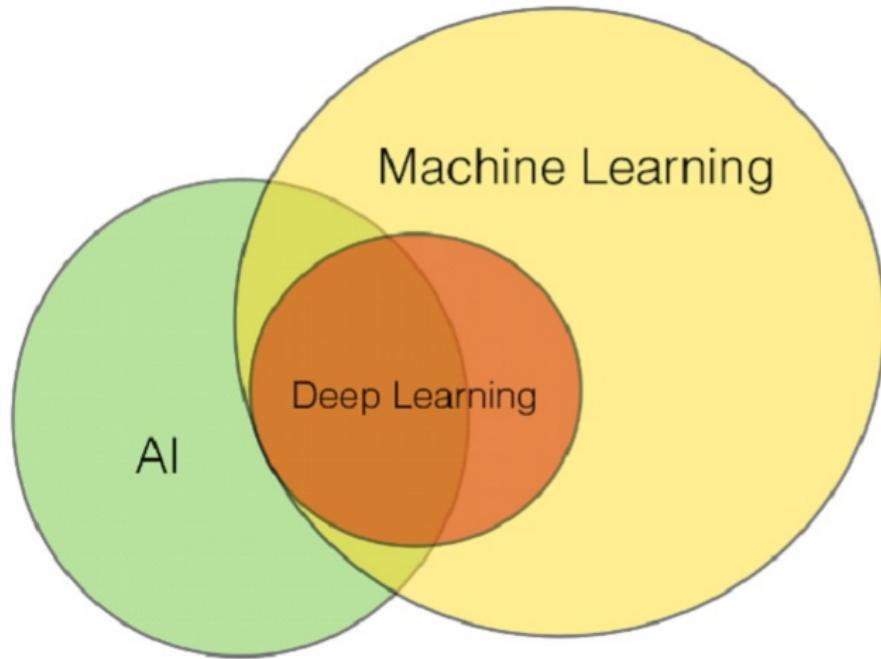
### Esimerkkejä tekoälyn nykyisestä käytöstä ja mahdollisista käyttökohteista

The diagram illustrates various applications of AI across different sectors:

- Ålypuhelinten ja tietokoneiden virtualiset avustajat (Smartphone and computer virtual assistants)
- Älykäs ilmanvaihto (Smart ventilation)
- Itseohjautuvat autot (Self-driving cars)
- Asioiden internet: verkkoon yhdistetyt imurit, jääkaapit, kelloit...
- Nettiostokset ja mainonta (Online stores and advertising)
- Älykäs maanviljely: kastelu, eläinten ruokkiminen, robottien hyödyntäminen haittakasvien poistossa
- Verkon hakukoneet (Search engines)
- Käännöskeet (Translation tools)
- Kyberturvallisuus (Cybersecurity)
- Disinformaation vastainen taistelu (Counter-disinformation efforts)
- Tuotteiden ja logistiikan optimointi (Product and logistics optimization)
- Tehtaissa käytetyt robotit (Robots in factories)

europarl.eu

### Koneoppiminen



- Kone oppii sille annetusta aineistosta.

## 5. Kirjastoja – Viikko 2–3

- N-ulotteiset taulukot, vektorit ja matriisit
- Tärkein tietotyyppi `array`
- `arrayn` kaikkien alkioiden on oltava samaa tyyppiä.
- Kattava kokoelma matemattisia funktioita.
  - Funktioille voi antaa parametrina `arrayn`
    - ⇒ `for`-looppien määrä vähenee
    - ⇒ Nopeampi ohjelman suoritus
- Tiedostonkäsitys
  - Luku/kirjoitus yhdellä komennolla

## Yksinkertainen esimerkki

## Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 N_points=50
5
6 #Generate N_points in [0, 2*pi[
7 x=np.linspace(0, 2*np.pi, N_points, endpoint=False)
8 y_sin=np.sin(x)
9 y_cos=np.cos(x)
10
11 plt.plot(x, y_sin)
12 plt.plot(x, y_cos)
13 plt.show()
```

Source file: **numpy01.py**

## 5.1. Kirjastoja – Viikko 2–3 – NumPy

### Taulukon luonti (1/2)

arrayn voi luoda useilla tavoilla, esimerkiksi:

- ① Listasta
- ② zeros, ones
- ③ eye
- ④ arange, linspace
- ⑤ random.random, random.normal, random.randint

## Taulukon luonti (2/2)

## Code

```
1 import numpy as np
2
3 a1=np.array([1, 2, 3])
4
5 a2=np.ones((2, ))
6 a3=np.zeros((3, 2, 2))
7
8 a4=np.eye(5, 5, dtype=np.float64)
9
10 a5=np.arange(4, 32, step=2, dtype=np.int32)
11 a6=np.linspace(-1.5, 1.5, 50, dtype=np.float32)
12
13 a7=np.random.random((2,3))
14 a8=np.random.normal(size=(2,3))
15 a9=np.random.randint(-3, 2 , size=(4, 2))
```

---

Source file: **numpy02.py**

### Tiedostonkäsittely

- `load()` – Lataa taulukoita tiedostosta
- `save()` – Tallettaa taulukon tiedostoon
- `savez()` – Tallettaa useita taulukoita samaan tiedostoon
- `savetxt()` – Tallettaa taulukon tekstimuotoiseen tiedostoon
- `savez_compressed()` - Tallettaa taulukoita pakattuun tiedostoon

#### Code

```
1 import numpy as np
2
3 filename='np_example_03.npy'
4
5 a1=np.array([1, 2, 3])
6
7 np.save(filename, a1)
8
9 del a1
10
11 a1=np.load(filename)
12 print(a1)
```

---

Source file: **numpy03.py**

## Code

```
1 import numpy as np
2
3 filename='np_example_04.npz'
4
5 a1=np.array([0.1, 0.4, 0.4], dtype=np.float64)
6 a2=np.array(['a', 'b', 'c', 'd'])
7 z=np.array([0, 3, 8])
8 np.savez(filename, my_named_parameter=a1, a2=a2,
          z=z)
9
10 del a1, a2, z
11
12 npz_file=np.load(filename)
13 a1=npz_file['my_named_parameter']
14 a2=npz_file['a2']
15 z=npz_file['z']
```

Source file: `numpy04.py`



- Indekointi alkaa 0:sta ja toimii pääpiirteissään kuten listankin tapauksessa.
- Vasen yläkulma on indeksi 0, 0.
- Useampiulotteisen taulukon alkioihin viitataan erottamalla indeksit pilkulla.
- Kaksiulotteisessa tapauksessa ensimmäinen indeksi viittaa riviin ja toinen sarakkeeseen.

### Esimerkki indeksoinnista

#### Code

```
1 import numpy as np
2
3 a1=np.array([[0.1,    0.4,    0.4],
4              [0.0,    1.0,   -1.0]], dtype=np.
5              float64)
6
7 print('Vasen ylakulma on', a1[0, 0]);
8 print('Oikea alakulma on', a1[1, 2]);
9 print('Ensimmainen sarake', a1[:, 0])
10 print('Toinen rivi', a1[1, :])
```

---

Source file: **numpy05.py**

## arrayn ominaisuudet

- `shape` – Taulukon dimensiot
- `size` – Alkioiden lukumäärä
- `dtype` – Alkion tyyppi

## Code

```
1 import numpy as np
2
3 a1=np.array([[0.1, 0.4, 0.4],
4             [0.0, 1.0, -1.0]], dtype=np.
5               float64)
6
7 print('Taulukko on', len(a1.shape), '
8       ulotteinen')
9 print('Taulukon koko on', a1.shape)
10 print('Taulukossa on', a1.size, 'alkiota')
11 print('Taulukon alkio on tyyppia', a1.dtype)
```

Source file: `numpy06.py`

### Taulukoilla laskeminen

- Yleisimmät aritmeettiset operaatiot toimivat samankokoisten taulukoiden välillä.(Tämä ei ole koko totuus, mutta tähän saatetaan palata myöhemmin.)
- Operaatiot tapahtuvat taulukoiden vastaavien alkioiden välillä (element wise).
- Skalaarin ja taulukon väliset operaatiot on myös määritelty.

## Esimerkki

## Code

```
1 import numpy as np
2
3 a=np.array([[0.1,    0.4,    0.4], [0.0,    1.0,
4     -1.0]])
5
6 b=np.array([[1.1,    0.8,   -0.4], [0.1,    1.1,
7     -0.3]])
8
8 print('a+b =', a+b)
9 print('a*b =', a*b)
10 print('3.0*a =', 3.0*a)
```

---

Source file: **numpy07.py**

- Vektorilaskennassa voidaan hyödynää 1-ulotteisia taulukoita.
- Yhteen- ja vähenneyslasku toimivat kuten aiemmin esitettiin.
- Pistetulo voidaan laskea `array``.dot()`-metodilla tai `@`-operaattorilla.
- Vektorin normi voidaan laskea `linalg.norm()`-funktiolla.

## Code

```
1 import numpy as np
2
3 x=np.array([1, 2, 1])
4 y=np.array([-2, 1, 0])
5
6 print('x.y =', x@y)
7 print('||x|| =', np.linalg.norm(x))
8 print('||y|| =', np.linalg.norm(y))
9 print('(x, y) =', np.arccos(
10     x.dot(y)/(np.linalg.norm(x)*np.linalg.norm(
11         y))))
```

---

Source file: **numpy08.py**

- Mariiseina voidaan käyttää 2-ulotteisia taulukoita.<sup>1</sup>
- Yhteen- ja vähennyslaskut toimvat kuten matematiikassa yleensä määritellään.
- \* tekee alkioittaisen kertolaskun, eli ei toimi kuten matriisikertolasku.
- Matriisien välinen tulo voidaan laskea `array``.dot()`-metodilla tai @-operaattorilla.
- Transpoosi lasketaan `array``.transpose()`-metodilla.
- Käänteismatriisi lasketaan `linalg.inv()`-metodilla.

---

<sup>1</sup>`numpy.matrix` on poistuva ominaisuus, älä käytä!

### Esimerkki

Code

```
1 import numpy as np
2
3 A=np.array([[1, 2],[3, 4],[5, 6]])
4 B=np.array([[0, 1],[1, 0]])
5 x=np.array([1, 2])
6
7 print(A@B@x)
```

---

Source file: **numpy09.py**

## Esimerkki (1/2)

Lasketaan  $\begin{pmatrix} 1 \\ 2 \end{pmatrix} (-1 \quad 2)$ .

## Code

```
1 import numpy as np
2
3 x=np.array([1, 2])
4 y=np.array([-1, 2])
5
6 print('x shape =', np.shape(x))
7 print('y shape =', np.shape(y))
8
9 #Ei toimi, pitäisi tulla 2x2 matriisi!
10 print('x^T y =', x.transpose()@y)
```

---

Source file: **numpy10.py**

## Esimerkki (2/2)

## Code

```
1 import numpy as np
2
3 #Luodaan vektorit matriiseina (2d-taulukoina
4 x=np.array([1, 2], ndmin=2)
5 y=np.array([-1, 2], ndmin=2)
6
7 print('x shape =', np.shape(x))
8 print('y shape =', np.shape(y))
9
10 print('x^T y =\n', x.transpose()@y)
```

---

Source file: **numpy11.py**

### reshape()

- Joskus voi olla tarpeen muuntaa taulukon muotoa, esimerkiksi muuttaa matriisi vektoriksi.
- Tämä tapahtuu `numpy.reshape()`-metodilla.

#### Code

```
1 import numpy as np
2
3 A=np.array([[0, 1, 2], [3, 4, 5]])
4 print(A)
5
6 x=A.reshape((6,))
7 print(x)
8
9 B=A.reshape((3,2))
10 print(B)
```

---

Source file: **numpy12.py**

## 5.2. Kirjastoja – Viikko 2–3 – pyplot

### Matplotlib ja pyplot

- Kirjasto kuvien piirtoon
- Oliopohjainen API
- Esimerkkejä
- Tilapohjainen API, `pyplot`
- Tutoriaali

### plot (1/2)

#### Code

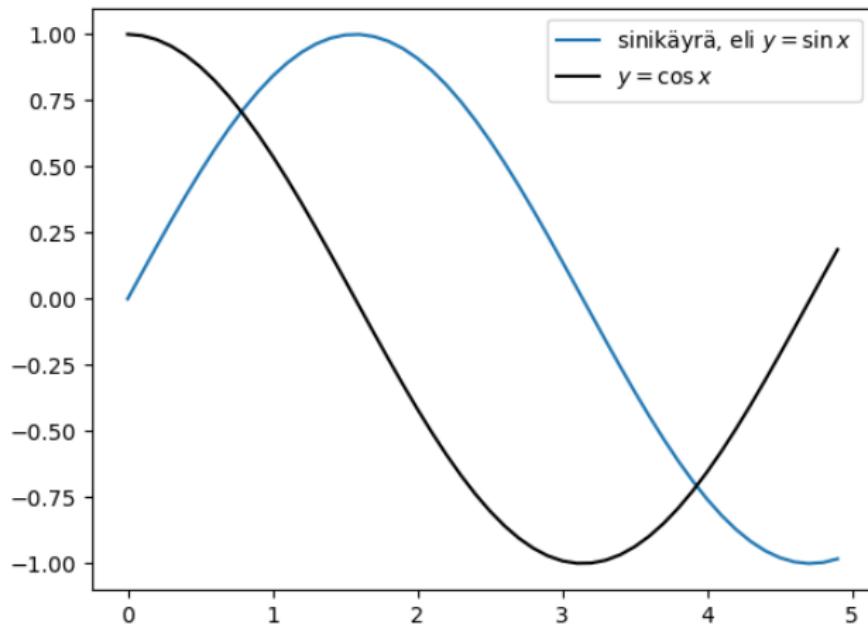
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.arange(0, 5, 0.1)
5 y_sin = np.sin(x)
6 y_cos = np.cos(x)
7
8 plt.plot(x, y_sin, label=r'sinikayra , eli $y = \
      sin\, , x$')
9 plt.plot(x, y_cos, 'k', label=r'$y = \cos\, , x$')
10
11 plt.legend()
12 plt.show()
13 #plt.savefig('pyplot01.png')
```

---

Source file: **pyplot01.py**

## 5.2. Kirjastoja – Viikko 2–3 – pyplot

plot (2/2)



### imshow (1/2)

#### Code

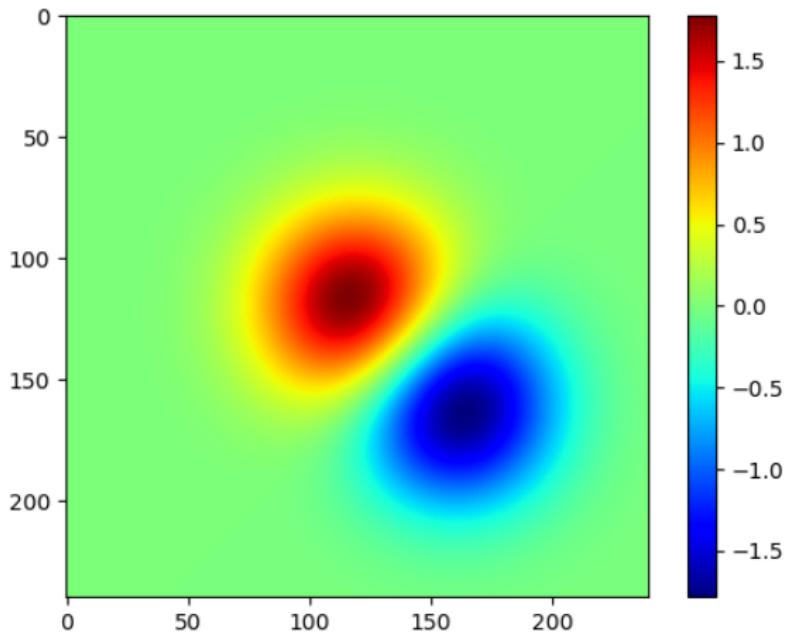
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 delta = 0.025
5 x = y = np.arange(-3.0, 3.0, delta)
6 X, Y = np.meshgrid(x, y)
7 Z1 = np.exp(-X**2 - Y**2)
8 Z2 = np.exp(-(X - 1)**2 - (Y - 1)**2)
9 Z = (Z1 - Z2) * 2
10
11 plt.imshow(Z, cmap='jet')
12 plt.colorbar()
13 plt.show()
14 #plt.savefig('pyplot02.png')
```

---

Source file: `pyplot02.py`

## 5.2. Kirjastoja – Viikko 2–3 – pyplot

imshow (2/2)



## 5.3. Kirjastoja – Viikko 2–3 – Keras

### MNIST luokittelija (1/5)

#### Code

```
1 #import keras
2 from keras.datasets import mnist
3 from keras.utils import np_utils
4 from keras.models import Sequential
5 from keras.layers import Dense
6
7 batch_size = 128
8 num_classes = 10
9 epochs = 8
```

---

Source file: **keras01.py**

## 5.3. Kirjastoja – Viikko 2–3 – Keras

### MNIST luokittelija (2/5)

#### Code

```
11 # the data, split between train and test sets
12 (x_train, y_train), (x_test, y_test) = mnist.
13     load_data()
14
15 x_train = x_train.reshape(60000, 784)
16 x_test = x_test.reshape(10000, 784)
17 x_train = x_train.astype('float32')
18 x_test = x_test.astype('float32')
19 x_train /= 255
20 x_test /= 255
21 print(x_train.shape[0], 'train samples')
22 print(x_test.shape[0], 'test samples')
```

---

Source file: **keras01.py**

## 5.3. Kirjastoja – Viikko 2–3 – Keras

### MNIST luokittelija (3/5)

#### Code

```
23 # convert class vectors to binary class matrices
24 y_train = np_utils.to_categorical(y_train,
25     num_classes)
25 y_test = np_utils.to_categorical(y_test,
26     num_classes)
```

---

Source file: **keras01.py**

## 5.3. Kirjastoja – Viikko 2–3 – Keras

### MNIST luokittelija (4/5)

#### Code

```
27 model = Sequential()
28 model.add(Dense(512, activation='relu',
29                 input_shape=(784,)))
30 model.add(Dense(512, activation='relu'))
31 model.add(Dense(num_classes, activation='softmax',
32                 ))
33
34 model.summary()
35
36 model.compile(loss='categorical_crossentropy',
37                 metrics=['accuracy'], optimizer='adam')
```

---

Source file: **keras01.py**

## 5.3. Kirjastoja – Viikko 2–3 – Keras

### MNIST luokittelija (5/5)

#### Code

```
36 history = model.fit(  
37     x_train, y_train,  
38     batch_size=batch_size,  
39     epochs=epochs,  
40     verbose=1,  
41     validation_data=(x_test, y_test))  
42  
43 score = model.evaluate(x_test, y_test, verbose=0)  
44 print('Test loss:', score[0])  
45 print('Test accuracy:', score[1])
```

---

Source file: **keras01.py**

## 5.4. Kirjastoja – Viikko 2–3 – scikit-learn

### Numeroiden luokittelija (1/3)

Code

```
1 # Author: Gael Varoquaux <gael dot varoquaux at  
# normalesup dot org>  
2 # License: BSD 3 clause  
3 from sklearn import datasets, svm, metrics  
4 from sklearn.model_selection import  
    train_test_split  
5  
6 digits = datasets.load_digits()  
7  
8 n_samples = len(digits.images)  
9 data = digits.images.reshape((n_samples, -1))
```

---

Source file: **sk01.py**

## 5.4. Kirjastoja – Viikko 2–3 – scikit-learn

### Numeroiden luokittelija (2/3)

#### Code

```
11 clf = svm.SVC(gamma=0.001)
12
13 X_train, X_test, y_train, y_test =
14     train_test_split(
15         data, digits.target, test_size=0.5, shuffle=
16             False
17     )
18
19 clf.fit(X_train, y_train)
20
21 predicted = clf.predict(X_test)
```

---

Source file: **sk01.py**

## 5.4. Kirjastoja – Viikko 2–3 – scikit-learn

### Numeroiden luokittelija (3/3)

Code

```
21 print(
22     f"Classification report for classifier {clf
23         }:\n"
24     f"{metrics.classification_report(y_test,
25         predicted)}\n"
26 )
```

---

Source file: **sk01.py**

- SciPy
- Wikipedia
- Tieteelliseen ja tekniseen laskentaan

## Esimerkki (1/4)

## Code

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from sklearn.linear_model import LinearRegression
4
5 #algebra.csv is available on shared materials/
6     data
7 data=pd.read_csv('algebra.csv', skiprows=1)
8
9 print('\n--- Lahtodata -----')
10 print(data)
11 print('\n--- Lahtodatan statistiikka ---')
12 print(data.describe())
```

---

Source file: **pandas01.py**

### Esimerkki (2/4)

Code

```
14 print('n--- Kiinnostavat sarakkeet ----\n')
15
16 tehtava_tentti=data[['Tehtavapisteet', '
17     Tenttipisteet']]
17 print(tehtava_tentti)
18
19 print('n--- Poistetaan NaN-arvot -----n')
20 tehtava_tentti=tehtava_tentti.dropna()
21
22 print(tehtava_tentti)
```

---

Source file: **pandas01.py**

## Esimerkki (3/4)

Code

```
24 print('n--- Sovitetaan suora pisteisiin n')
25
26 X=tehtava_tentti[ 'Tehtavapisteet' ].values.reshape
27     (-1, 1)
27 Y=tehtava_tentti[ 'Tenttipisteet' ].values.reshape
28     (-1, 1)
29
29 regr = LinearRegression()
30 regr.fit(X, Y)
31 Y_pred = regr.predict(X)
```

---

Source file: **pandas01.py**

## Esimerkki (4/4)

Code

```
33 print('n--- Lineaarinen malli -----n')
34
35 print('Tenttipisteet = '+str(regr.coef_[0][0])+
      * '+ 'Tehtavapisteet + '+str(regr.intercept_-
      [0]))
36
37 plt.plot(X, Y, '*')
38 plt.plot(X, Y_pred, '--')
39 plt.show()
```

Source file: **pandas01.py**

- Orange Tutorial
- Introduction to Data Mining
- Orange
- Orange Data Mining (YouTube)
- Graafinen työkalu tiedonlouhintaan

## 5.8. Kirjastoja – Viikko 2–3 – Harjoituksia

### Harjoituksia – NumPy

- M1 Luo taulukko **a**, jossa on 3 riviä ja 4 saraketta ja kaikkien alkioiden arvo on 0. Talleta taulukko tiedostoon **a.npy**.
- M2 Lue taulukko tiedostosta **a.npy** ja aseta siinä indeksin 0,0 arvo 1 ja oikeaan alakulmaan arvo -1. Talleta muokattu taulukko tiedostoon **b.npy**.
- M3 Lue taulukot tiedostoista **a.npy** ja **b.npy**. Talleta ne molemmat taulukot tiedostoon **ab.npz**. Tallenna ensimmäinen taulukko nimellä **a** ja jälkimmäinen nimellä **b**.
- M4 Lue tiedosto **ab.npz** ja talleta taulukko jonka nimi on **b** tiedostoon **b2.npy**.

## 5.8. Kirjastoja – Viikko 2–3 – Harjoituksia

### Harjoituksia – NumPy

- (1) Muodosta matriisitulo  $\begin{pmatrix} 2 & 4 \\ 1 & 1 \\ 4 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$ .
- (2) Totea edellisen tehtävän matriiseilla, että kaava  $\mathbf{AB} = (\mathbf{B}^T \mathbf{A}^T)^T$  pätee.
- (3) Laske matriisin  $\mathbf{A} = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$  käänneismatriisi.

## 6. Tekoälyn menetelmiä – Viikko 3

### Viikon sisältö

- 1 Esitellään joitain tärkeimpiä tekoälyn menetelmiä

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Esimerkki fysiikasta

#### Example 1

*Halutaan maksimoida nuolen kantama ammuttaessa jousella. Jätetään ilmanvastus huomioimatta, ja oletetaan, että nuoli putoaa samalle korkeudelle, mistä se ammuttiin. Merkitään*

*$\alpha$  = nuolen ampumiskulma,*

*$v_0$  = nuolen lähtönopeus ja*

*$g$  = gravitaatiokiihtyvyys.*

*Tällöin nuolen lentämä matka*

$$s(\alpha) = \frac{v_0^2 \sin 2\alpha}{g} .$$

*Huomataan, että millä tahansa fysikaalisesti järkevillä arvoilla  $\alpha^* = 45^\circ$  kulmassa ammuttu nuoli lentää pisimmälle.*

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Optimointimenetelmistä

- Esimerkissä optimoitava funktio oli niin yksinkertainen, että tuntemalla trigonometristen funktioiden perusteet optimikulma  $\alpha^*$  saatiin päättelyä.
- Usein optimoitava *kohdefunktio* on niin hankala, että vastaava päätely ei ole mahdollista. Näin käy usein varsinkin useampiulotteisten kohdefunktioiden tapauksessa.
- Tällöin on käytettävä numeerisia optimointimenetelmiä.
- Idea näissä menetelmissä on yleensä se, että kullakin askelleella siirretään tarkastelupistettä ”ylämäkeen”, siis kohden suurempaa kohdefunktion arvoa. (maksimoinnissa)
- Minimoitaessa siirrytään ”alamäkeen”.
- On olemassa myös muunlaisia optimointimenetelmiä, esimerkiksi menetelmiä, joissa liikutan joskus myös väärään suuntaan.

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Optimointitehtävä

#### Definition

*Tehtävää, jossa on tarkoitus etsiä  $x_1, x_2, \dots, x_n \in \mathbb{R}$  siten, että valituilla  $x_i, i = 1 \dots n$  arvoilla  $f(x_1, x_2, \dots, x_n)$  saa pienimmän arvon, kutsutaan minimointitehtäväksi.*

#### Remark

*Vastaavasti tehtävää, jossa etsitään suurimman arvon tuottavia  $x_i, i = 1 \dots n$  kutsutaan maksimointitehtäväksi.*

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Kauppamatkustajan ongelma

#### Example 2

*Jos kauppamatkustaja tietää kaupunkien  $1, 2, \dots, n$  keskinäiset etäisyydet, miten hän voi laskea itselleen lyhimmän kulkureitin, jossa palataan lopussa lähtökaupunkiin ja kuljetaan kaikkien muiden kaupunkien kautta tasamittaisesti?*

- Jos kaupunkeja on  $n$  kappaletta, niin tällöin on  $(n - 1)!$  erilaista ehdot täytyvästä reittiä.
- Jos kaupunkeja on 10, on mahdollisia reittejä 362 880.
- Jos kaupunkeja on 20, on mahdollisia reittejä 121 645 100 408 832 000.
- Ratkaisua voi yrittää hakea jollain sopivalla heuristisella algoritmillä. (Palataan tähän myöhemmin)
- Usein riittää, että löydetään ”riittävän hyvä” ratkaisu.

### Harjoitus

(4) Kauppamatkustaja lähtee Kuopiosta. Hänen on vieraitava

- Joensuussa,
- Oulussa,
- Lappeenrannassa,
- Porissa,
- Helsingissä ja
- Tampereella.

Lopuksi kauppamatkustaja palaa takaisin Kuopioon. Etsi kauppamatkustajalla mahdollisimman lyhyt reitti. Käytä hyväksesi **välimatkataulukkoa**. Kuinka pitkä on lyhin löytämäsi reitti?

(5) Käytä lisäksi Suomen karttaa reitin suunnittelussa. Kuinka pitkä on löytämäsi reitti nyt?

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Erilaisia numeerisia menetelmiä

- Gradienttimenetelmät:
  - ”Pallo pyörii alamäkeen”
  - Visualisointi
  - Suuri määrä erilaisia muunnoksia, esimerkiksi Backpropagation.
- Geneettinen algoritmi:
  - Luodaan joukko keinotekoisia olioita, joilla on satunnaisesti tuotetut geenit.
  - Simuloidaan evoluutiota: Paremmin soveltuvat yksilöt (geenit) tuottavat todennäköisemmin jälkeläisiä.
  - AI Learns to be a Car using a Genetic Algorithm.
- Simuloitu jäähdytys (Simulated annealing):
  - Kauppamatkustajan ongelman ratkaisu simuloituna
- Parviäly:
  - Ant Colony Optimization

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Esimerkki fysiikasta

#### Example 3

Halutaan maksimoida nuolen kantama ammuttaessa jousella. Jätetään ilmanvastus huomioimatta, ja oletetaan, että nuoli putoaa samalle korkeudelle, mistä se ammuttiin. Merkitään

$\alpha$  = nuolen ampumiskulma,

$v_0$  = nuolen lähtönopeus ja

$g$  = gravitaatiokiihtyvyys.

Tällöin nuolen lentämä matka

$$s(\alpha) = \frac{v_0^2 \sin 2\alpha}{g} .$$

Huomataan, että millä tahansa fysikaalisesti järkevillä arvoilla  $\alpha^* = 45^\circ$  kulmassa ammuttu nuoli lentää pisimmälle.

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Optimointimenetelmistä

- Esimerkissä optimoitava funktio oli niin yksinkertainen, että tuntemalla trigonometristen funktioiden perusteet optimikulma  $\alpha^*$  saatiin päättelyä.
- Usein optimoitava *kohdefunktio* on niin hankala, että vastaava päätely ei ole mahdollista. Näin käy usein varsinkin useampiulotteisten kohdefunktioiden tapauksessa.
- Tällöin on käytettävä numeerisia optimointimenetelmiä.
- Idea näissä menetelmissä on yleensä se, että kullakin askelleella siirretään tarkastelupistettä ”ylämäkeen”, siis kohden suurempaa kohdefunktion arvoa. (maksimoinnissa)
- Minimoitaessa siirrytään ”alamäkeen”.
- On olemassa myös muunlaisia optimointimenetelmiä, esimerkiksi menetelmiä, joissa liikutan joskus myös väärään suuntaan.

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Optimointitehtävä

#### Definition

*Tehtävää, jossa on tarkoitus etsiä  $x_1, x_2, \dots, x_n \in \mathbb{R}$  siten, että valituilla  $x_i, i = 1 \dots n$  arvoilla  $f(x_1, x_2, \dots, x_n)$  saa pienimmän arvon, kutsutaan minimointitehtäväksi.*

#### Remark

*Vastaavasti tehtävää, jossa etsitään suurimman arvon tuottavia  $x_i, i = 1 \dots n$  kutsutaan maksimointitehtäväksi.*

#### Example 4

Mitataan suuretta  $Y$ , joka riippuu lineaarisesti toisesta suureesta  $X^a$ . Olkoon mittaustulokset  $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$ . Tehtäväänä on etsiä vakiot  $a$  ja  $b$  siten, että suora  $y = ax + b$  sovittuu mahdollisimman hyvin mitattuun aineistoon.

---

<sup>a</sup>Esimerkiksi jousen venymä ja venytävä voima

- Lineaarinen regressio on tärkeä työkalu esimerkiksi fysiikassa.
- Optimointitehtävä
- Ongelmalle tunnetaan yksinkertainen ratkaisukaava.
- Kun  $a$  ja  $b$  on laskettu mittausarvoista, voidaan mallia käyttää ennustamaan toisen suureen arvoa, kun toinen tunnetaan.  
(Koneoppiminen)
- Näin, vaikka taustalla olevaa mekanismia, esimerkiksi fysikaalista teoriaa, ei tunnettaisikaan.

## Harjoitus

- (6) Olkoon terästangon mitattu pituus eri lämpötiloissa alla olevan taulukon mukainen. Määritä graafisesti teräksen lämpölaajenemiskerroin. Vinkki: Vähennä taulukon arvoista jokin sopiva arvo, jolloin piirtäminen helpottuu. Tämä vähennys ei vaikuta sovitettun suoran kulmakertoimeen, eli lämpölaajenemiskertoimeen.

T	l
0° C	1.0028 m
5° C	1.0029 m
15° C	1.0029 m
25° C	1.0039 m
30° C	1.0031 m
100° C	1.0040 m
110° C	1.0040 m

Todellinen arvo on noin  $0.000012 \frac{1}{\text{°C}}$

## 6.1. Tekoälyn menetelmiä – Viikko 3 – Optimointi

### Erilaisia numeerisia menetelmiä

- Gradienttimenetelmät:
  - ”Pallo pyörii alamäkeen”
  - Visualisointi
  - Suuri määrä erilaisia muunnoksia, esimerkiksi Backpropagation.
- Geneettinen algoritmi:
  - Luodaan joukko keinotekoisia olioita, joilla on satunnaisesti tuotetut geenit.
  - Simuloidaan evoluutiota: Paremmin soveltuvat yksilöt (geenit) tuottavat todennäköisemmin jälkeläisiä.
  - AI Learns to be a Car using a Genetic Algorithm.
- Simuloitu jäähdytys (Simulated annealing):
  - Kauppamatkustajan ongelman ratkaisu simuloituna
- Parviäly:
  - Ant Colony Optimization

## 6.2. Tekoälyn menetelmiä – Viikko 3 – Koneoppiminen

### Määritelmä

#### Definition

*Koneoppiminen on tekoälyn osa-alue, jossa ohjelma oppii mallin parametrit itsenäisesti sille syötetystä datasta (opetusaineistosta).*

#### Remark

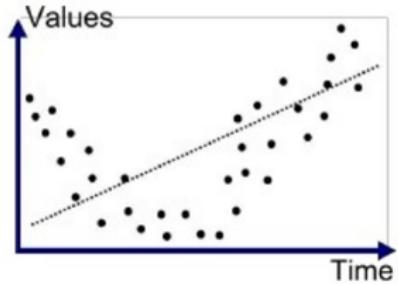
*Edellisessä määritelmässä sanotaan, että ohjelma oppii itsenäisesti. Tämä ei kuitenkaan välttämättä tarkoita sitä, että ihmistä ei tarvittaisi esimerkiksi*

- ❶ *keräämään ja muokkaamaan opetusaineistoa,*
- ❷ *kertomaan opetusalgoritmille millainen ratkaisu on haluttu (optimoinnin kohdefunktio) ja*
- ❸ *kehittämään sopiva malli.*

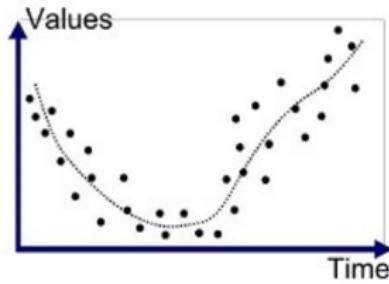
## 6.2. Tekoälyn menetelmiä – Viikko 3 – Koneoppiminen

### Ali- ja ylisovittuminen

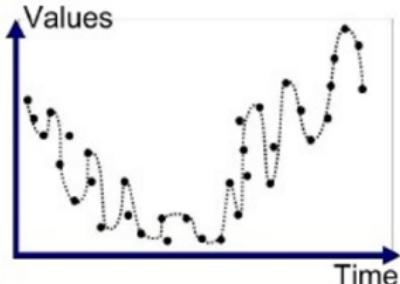
- Jos malli on liian ”jäykkä”, ei paraskaan opetusalgoritmi voi opettaa mallia kovin tarkasti. (Alisovittuminen)
- Jos malli on liian ”joustava”, malli oppii myös opetusdatan sisältämän kohinan. (Ylisovittuminen)



Underfitted



Good Fit/Robust

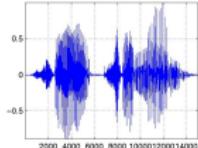


Overfitted

## 6.2. Tekoälyn menetelmiä – Viikko 3 – Koneoppiminen

### Ohjattu oppiminen

- Opetusjoukko koostuu pareista ( $x, y$ ), jotka edustavat järjestelmän syötettä ja odotettua ulostuloa.
- Kuvantunnistuksessa opetusjoukko koostuu yleensä valokuvista ja niitä vastaavista ulostuloista. Esimerkiksi (, auto)
- Äänentunnistuksessa opetusjoukko voi koostua ääninäytteistä ja

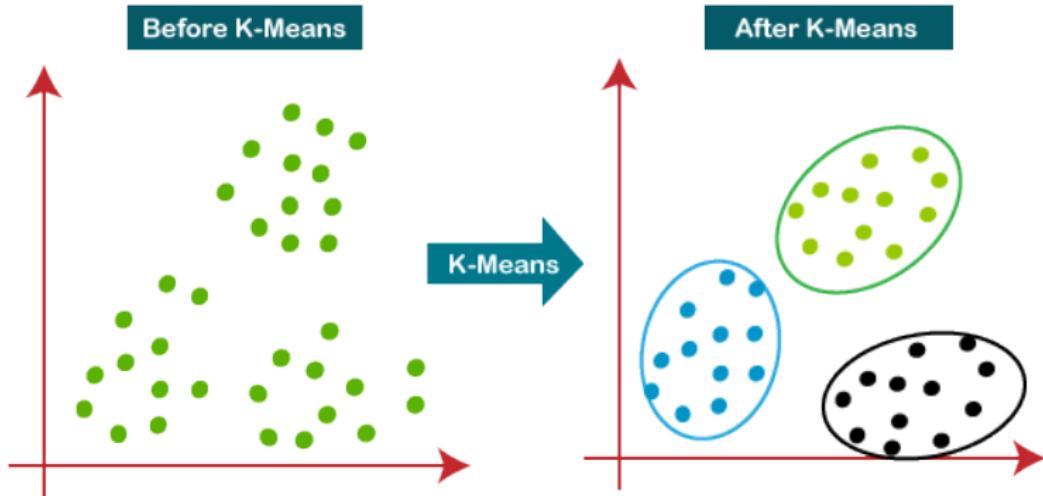


- niitä vastaavista teksteistä. Esimerkiksi ( , "7")
- Ongelmana saattaa olla sopivien näytteiden generointi, sillä vaadittu opetusjoukko voi olla kohtalaisen iso.

## 6.2. Tekoälyn menetelmiä – Viikko 3 – Koneoppiminen

### Ohjaamaton oppiminen

- Opetusjoukko koostuu vain lähtöjoukosta  $X$ , josta malli oppii erottelemaan tiettyjä piirteitä.

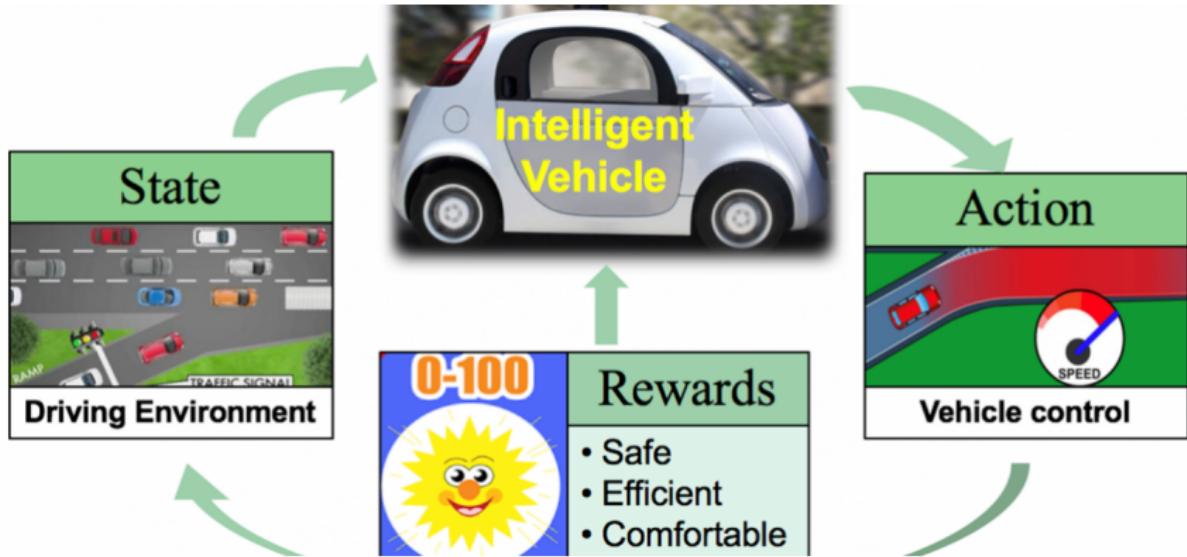


## **6.2. Tekoälyn menetelmiä – Viikko 3 – Koneoppiminen**

### **Vahvistusoppiminen (1/2)**

- Malli havaitsee ympäristön tilan, jonka perusteella se tekee sarjan toimintoja. Palaute onnistumisesta/epäonnistumisesta annetaan vasta jälkikäteen.
- Esimerkiksi autonomiset kulkuneuvot. Jos törmätään johonkin, annetaan negatiivinen palaute. Jos ajo sujuu hyvin, annetaan positiivista palautetta.
- Mahdollistaa opettamisen reaalimaailmassa tai simulointien avulla.
- Tekoäly voidaan opettaa pelaamaan erilaisia pelejä.

## 6.2. Tekoälyn menetelmiä – Viikko 3 – Koneoppiminen Vahvistusoppiminen (2/2)



### Empty

- Tehtävässä 6 määritettiin teräksen lämpölaajenemiskerrointa mittausdatan avulla.
- Koska lämpölaajeneminen on mittausalueella likimain lineaarista, käytettiin mallia

$$l(T) = l_0 + l_0 \alpha (T - T_0). \quad (1)$$

- Yleisemmin lineaarisessa regressiossa käytetään mallia

$$y(x) = ax + b. \quad (2)$$

- Tehtävänä on löytää kerron  $a$  ja vakiotermi  $b$  siten, että saatu suora sovittuu mittausaineistoon mahdollisimman hyvin.

- Lineaarinen malli

$$y(x) = ax + b$$

on 1. asteen polynomi.

- Vastaavasti mallina voidaan käyttää korkeamman asteen polynomia, esimerkiksi 2. asteen polynomia

$$y(x) = ax^2 + bx + c. \quad (3)$$

- Polynomin kerrointen laskenta on samankaltaista kuin lineaarisen mallin tapauksessa.
- Vastaavaa regressioanalyysiä voidaan tehdä myös muille kuin polynomifunktioille.

### Esimerkki

#### Example 5

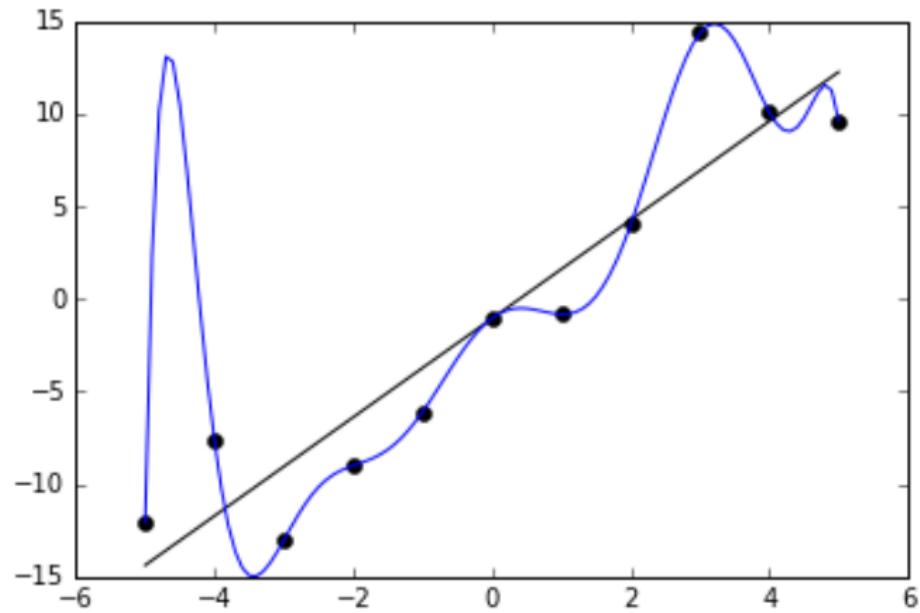
- Vapaassa putoamisessa kappaleen korkeus

$$y(t) = y_0 + v_0 t + \frac{1}{2} g t^2.$$

- Yhtälö on 2. asteen polynomi, eli samaa muotoa kuin yhtälö (3).
  - Tämä nähdään, kun valitaan
    - $x = t$ ,
    - $a = \frac{1}{2}g$ ,
    - $b = v_0$  ja
    - $c = y_0$ .
- ⇒  $g$  voidaan määritää pudottamalla kappale ja mittaamalla sen putoama matka tunnettuina ajanhetkinä.

## 6.3. Tekoälyn menetelmiä – Viikko 3 – Regressio

### Esimerkki polynomisesta ylisovittumisesta

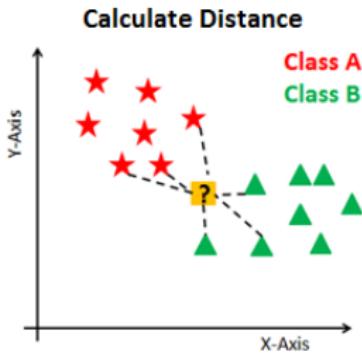
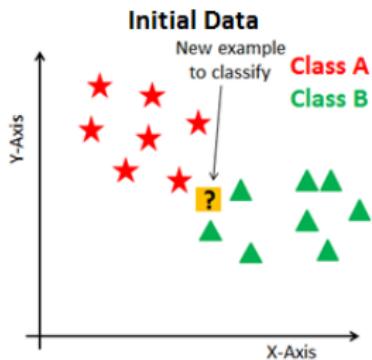


## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu

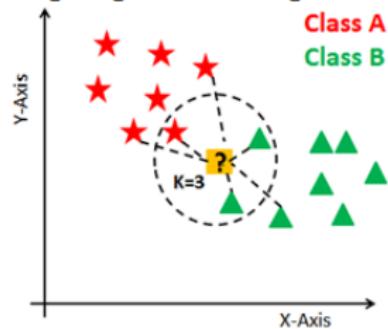
### KNN-luokitin

- ① Opetusaineisto koostuu  $x_i \in \mathbb{R}^n$  pisteistä, joihin jokaiseen on kiinnitetty luokka  $y_i$ .
- ② Algoritmi ei vaadi varsinaista opetusvaihetta, vaan luokittelut tehdään suoraan opetusaineiston perusteella.
- ③ Luokiteltavan pisteen  $x$  luokka  $y$  valitaan  $k$   $x$ :ää lähimmän naapurin luokkien (K Nearest Neighbours) perusteella.

### Esimerkki



### Finding Neighbors & Voting for Labels

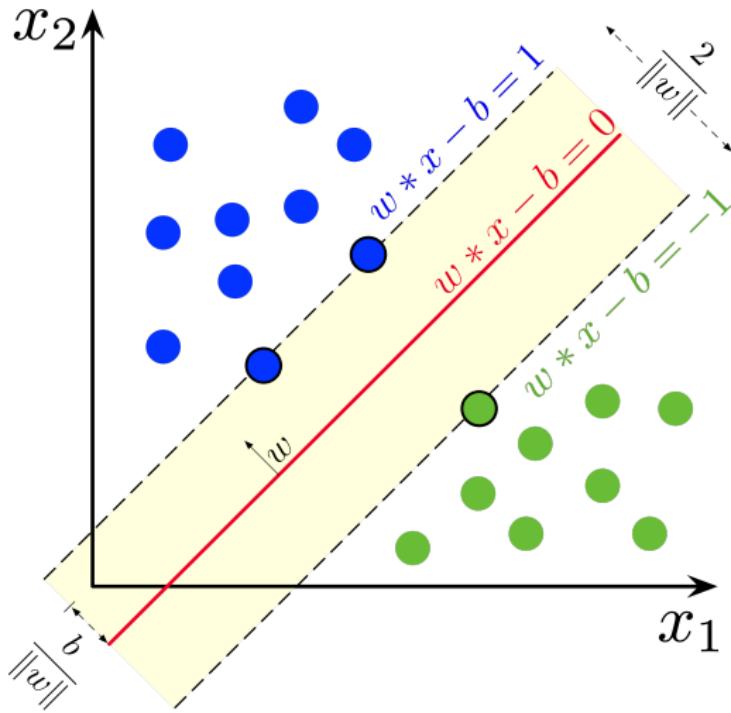


## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu Tukivektorikone (SVM)

- Opetusaineisto koostuu  $x_i \in \mathbb{R}^n$  pisteistä, joihin jokaiseen on kiinnitetty luokka  $y_i \in \{0, 1\}$ .
- Opetuksessa etsitään hypertaso, esimerkiksi suora, joka jakaa opetusaineiston kahteen luokkaan mahdollisimman selkeästi.
- SVM voidaan laajentaa toimimaan myös monimutkaisempiin tapauksiin:
  - Luokkia voi olla enemmän kuin 2
  - Avaruuden jako voidaan tehdä myös muunlaisilla objekteilla kuin hypertasolla.
- Luokittelu nopeaa.

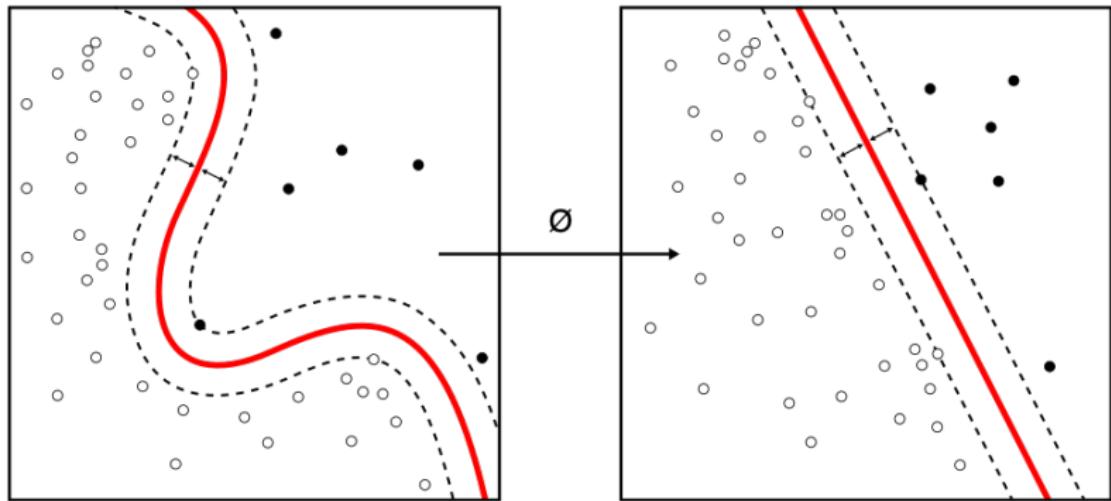
## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu

### Esimerkki



## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu

### Laajennus



## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu

### MNIST-aineisto

- Paljon käytetty testi- tai esimerkkiaineisto
- Käsin kirjoitettuja numeroita
- Harmaasävykuvia
- Resoluutio  $28 \times 28$
- Opetusjoukko 60000 merkkiä
- Testijoukko 10000 merkkiä

## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu

### Esimerkkikuvia

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4  
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5  
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6  
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7  
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

## 6.4. Tekoälyn menetelmiä – Viikko 3 – Luokittelu

### Merkkien tunnistaminen

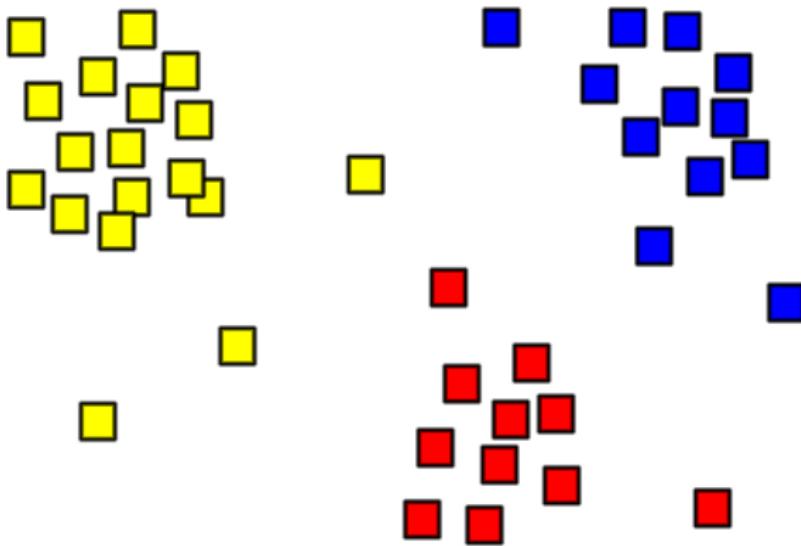
- SVM-luokittelulla 93.2% tarkkuus (parhaat muunnokset 99.5%)
- KNN-luokittelulla 97.4% tarkkuus (parhaat muunnokset 99.5%)
- Parhaat neuroverkkopohjaiset ratkaisut 99.8%

## 6.5. Tekoälyn menetelmiä – Viikko 3 – Klusterointi

### Klusteroinnin periaate

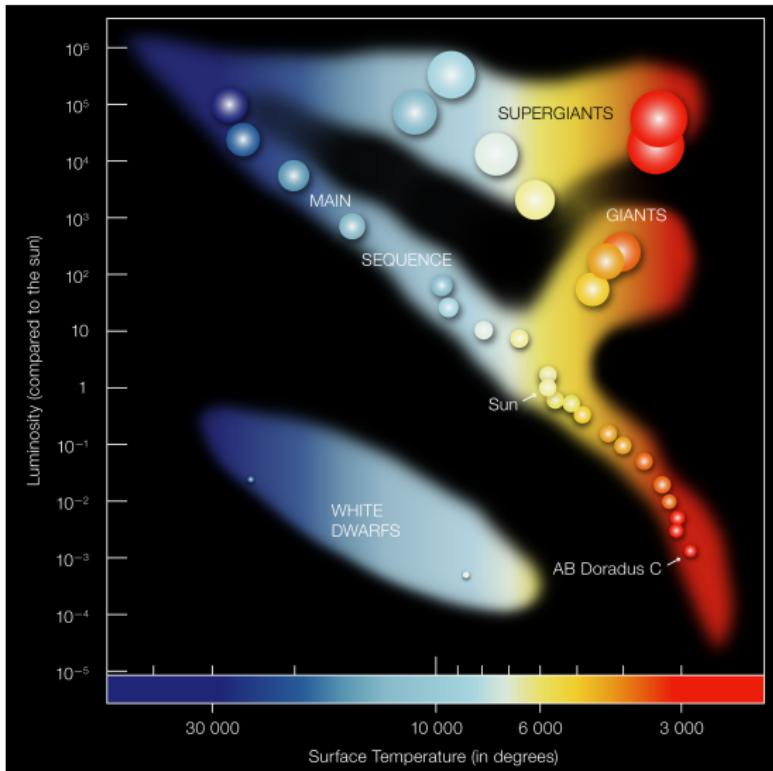
- Luokittelussa jokaisen opetusjoukon alkion luokka oletettiin tunnetuksi.
- Tällaisen luokitellun aineiston luominen voi olla erittäin työlästä.
- Onnistuisiko opetus ilman etukäteen luokiteltua aineistoa?
- Vastaus: Ainakin joissain tapauksissa. Tällöin puhutaan klusteroinnista.

### Klusterointi



## 6.5. Tekoälyn menetelmiä – Viikko 3 – Klusterointi

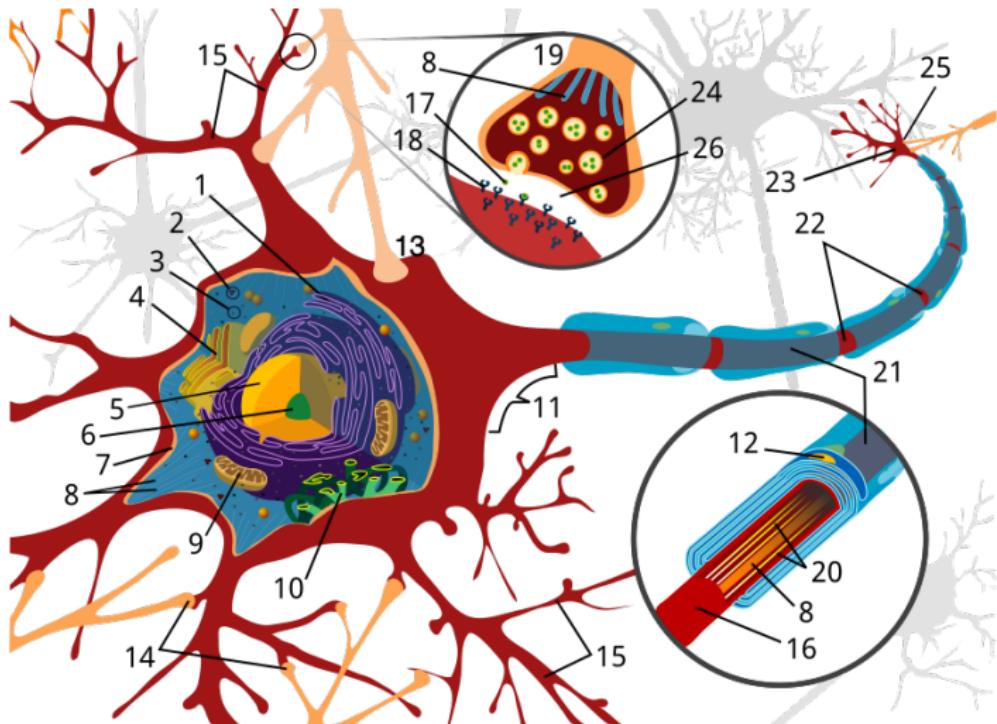
### Esimerkki



The Hertzsprung-Russell Diagram

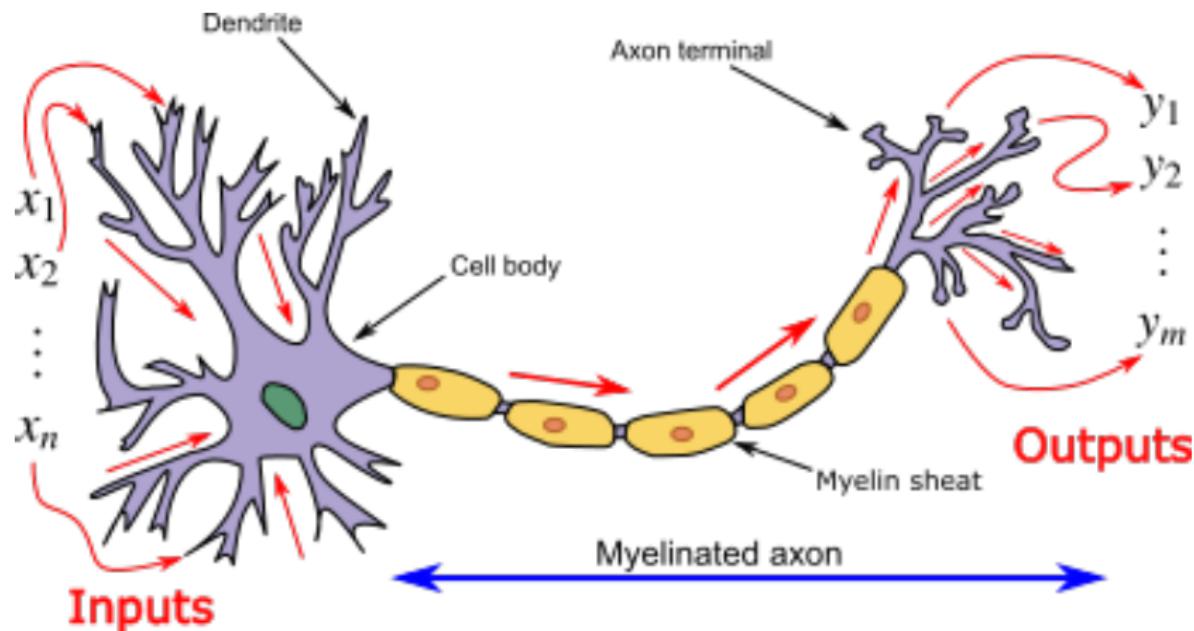


### Neuroni



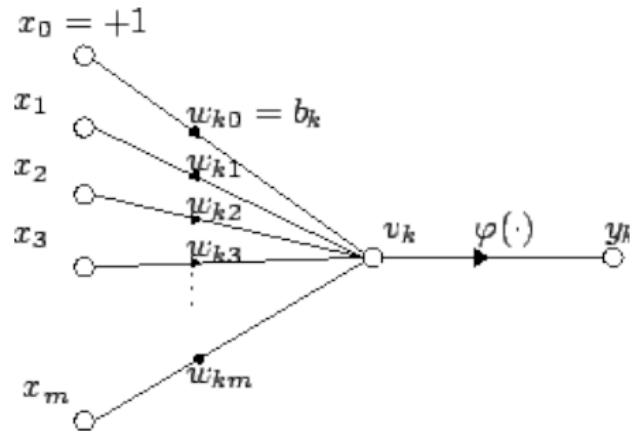
## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

### Keinotekoinen neuroni (1/2)



## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

### Keinotekoinen neuroni (2/2)



$$y_k = \varphi(b_k + w_{k1}x_1 + w_{k2}x_2 + \cdots + w_{km}x_m)$$

### Esimerkki

Example 6

Neuronin

$$b_k = 6 ,$$

$$w_{k1} = -1 , \quad w_{k2} = 1 \quad \text{ja} \quad w_{k3} = 2 .$$

Olkoon aktivaatiofunktio  $\varphi(z) = 2z$ .

Lisäksi syöte

$$x_1 = 3 , \quad x_2 = 2 \quad \text{ja} \quad x_3 = 1 .$$

Tällöin neuronin ulostulo

$$\begin{aligned}y_k &= \varphi(b_k + w_{k1}x_1 + w_{k2}x_2 + w_{k3}x_3) \\&= 2 \cdot (6 + (-1) \cdot 3 + 1 \cdot 2 + 2 \cdot 1) \\&= 14 .\end{aligned}$$

## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

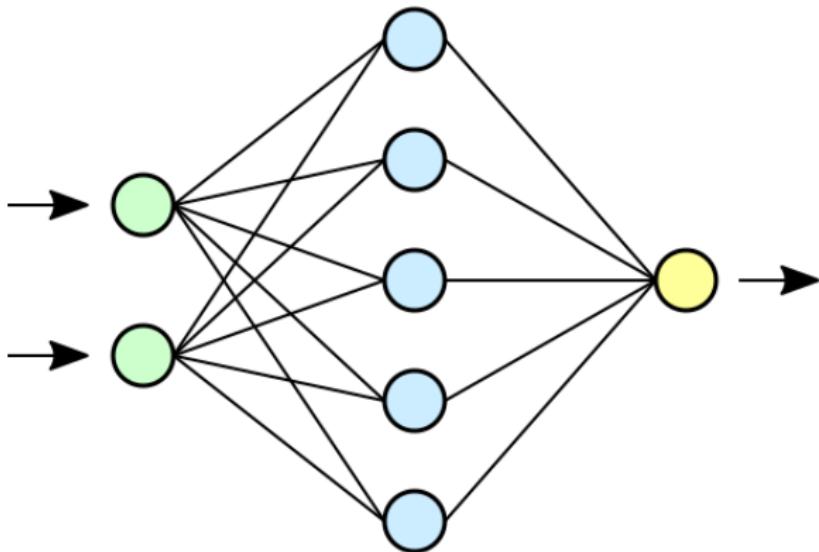
### Yleisimpiä aktivointifunktioita

- Lineaarinen:  $\varphi(z) = z$
- ReLU:  $\varphi(z) = \max(0, z)$
- Logistinen:  $\varphi(z) = \frac{1}{1+\exp(-z)}$

---

Wikipedia – Activation function

### Esimerkki



- Vasemmalla kaksipaikkainen syötevektori
- Keskellä viisi neuronia käsittävä kerros (layer)
- Oikealla yhden neuronin käsittävä ulostulokerros

## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

### Perusperiaatteet

- ① Neuronit on järjestetty kerroksiksi
- ② Tieto kulkee verkossa vain yhteen suuntaan, kerrokselta toiselle
- ③ Tieto ei palaa takaisin aiemmalle kerrokselle
- ④ Jokaisella neuronilla on muista neuroneista riippumattomat parametrit
- ⑤ Kullakin kerroksella voi olla erilainen aktivointifunktio

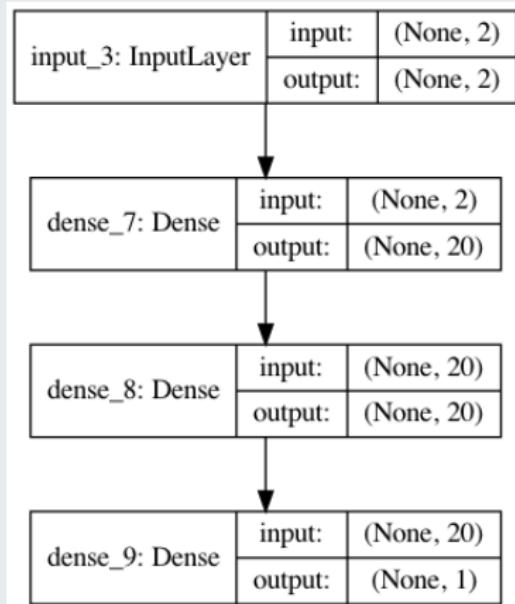
---

Nämä eivät ole mitenkään kiveenhakattuja sääntöjä. On paljon erilaisia neuroverkkoja, jotka toimivat näiden sääntöjen vastaisesti.

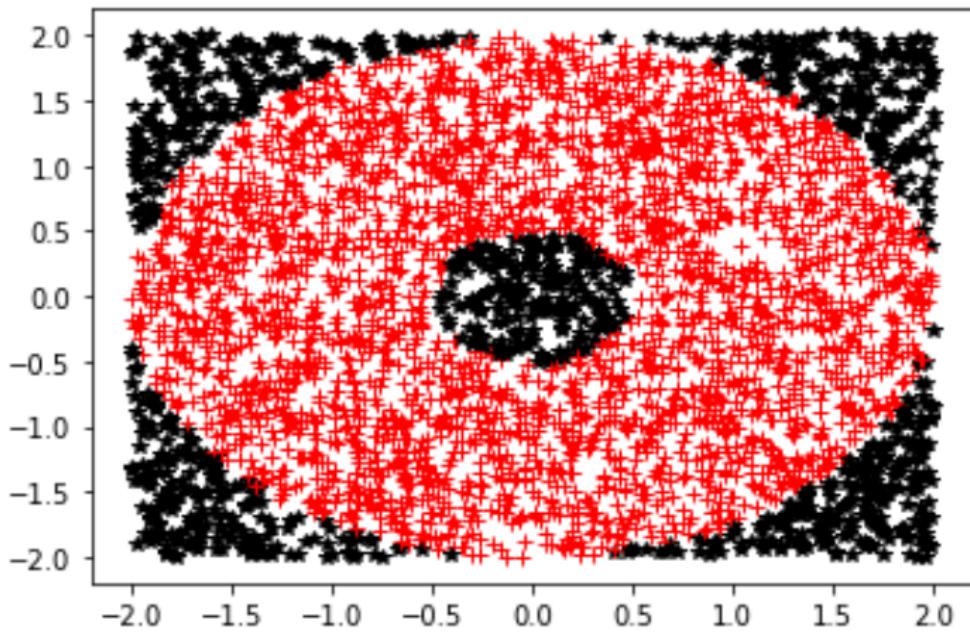


## Esimerkki (1/2)

Example 7



Example 7



### Taustaa

- Neuronissa  $k$  on suuri joukko vapaita parametreja,  $b_k, w_{k1}, w_{k2} \dots$
- Neuroverkossa on useita neuroneita, joten se sisältää vielä suuremman joukon vapaita parametreja.
- Vapaiden parametrien kiinnittäminen ”käsin” on hyvin hankala tehtävä jo pienellekin neuroverkolle.
- Parametrien valintaa kutsutaan neuroverkon kouluttamiseksi (training/teaching).
- Yksinkertaisimillaan koulutus tehdään luomalla riittävän suuri opetusjoukko, joka sisältää kattavasti esimerkkejä ratkaistavasta ongelmasta.
- Lisäksi on valittava ongelmaan sopiva verkon arkkitehtuuri ja aktivaatiofunktiot. Tätä ei käsitellä tässä yhteydessä tämän enempää.

## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

### Optimointitehtävä

- Opetusprosessi voidaan nähdä optimointitehtävänä.
- Tehtävänä on hakea mahdollisimman hyvät (optimaaliset) arvot parametreille **b** ja **w**.
- Mittarina voi toimia esimerkiksi opetusjoukon tapausten keskimääräinen virhe.

- ① Luo riittävän suuri opetusjoukko, jossa lähtöarvot ovat taulukossa **X** ja halutut vasteet **Y**.
- ② Alusta neuroverkon parametrit **b** ja **w**
- ③ Etsi parametrit s.e keskimääräinen virhe on mahdollisimman pieni.
- ④ Yleisin opetusmenetelmä on vastavirta-algoritmi (backpropagation)

#### Example 8

- Esimerkissä 7 opetettiin neuroverkko tunnistamaan kuuluuko annettu piste (koordinaatti) origokeskeiseen renkaaseen, jonka sisäsäde on  $\frac{1}{2}$  ja ulkosäde 2.
- Opetusjoukkona käytettiin 16384 satunnaisesti valittua pistettä ja tieto siitä, kuuluuko piste renkaaseen vai ei.
- Haluttu vaste oli 1 mikäli piste oli renkassa ja muuten 0.
- Parametreja optimoitiin  $512 \times 5000$  kertaa.

- Ohjattu oppiminen
- Ohjaamaton oppiminen
- Vahvistusoppiminen

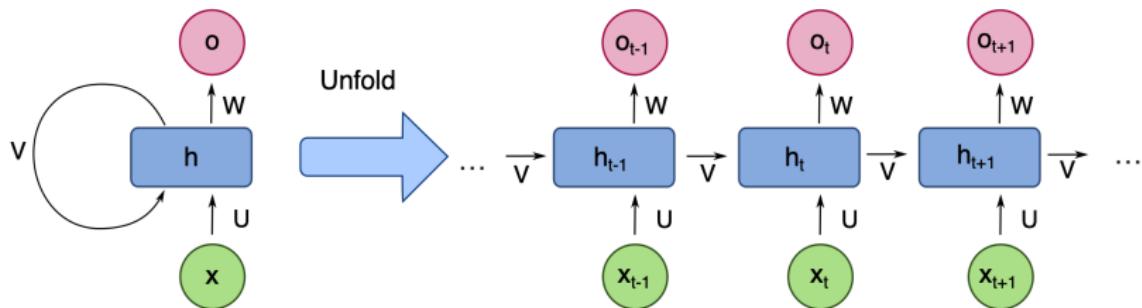
### Rajoitteita

- Edellä esiteltiin arkkitehtuuri, jossa
    - kerroksittain järjestetylle neuroverkolle syötettiin
    - kiinteän kokoinen syöte joka loi
    - kiinteän kokoinen ulostulon.
  - Entä jos tekoäly analysoisikin tekstiä ja tekisi siitä yhteenvedon?
    - Syöte ei ole kiinteän pituinen.
    - Ulostulo ei ole kiinteän pituinen.
- ⇒ Aiemmin esitetty verkon malli ei ole sovellettavissa tällaiseen tapaukseen.

## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

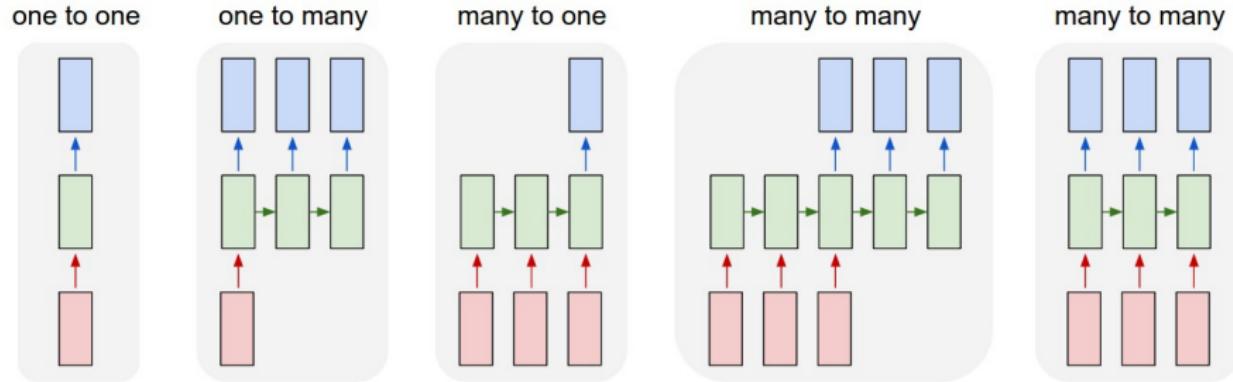
### Takaisinkytetty verkko (1/2)

- Verkon ulostulo kytketään sisäänmenoona.
- Näin verkko pystyy muistamaan jo näkemiään asioita.
- Esimerkiksi tekstin analysointi.



## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

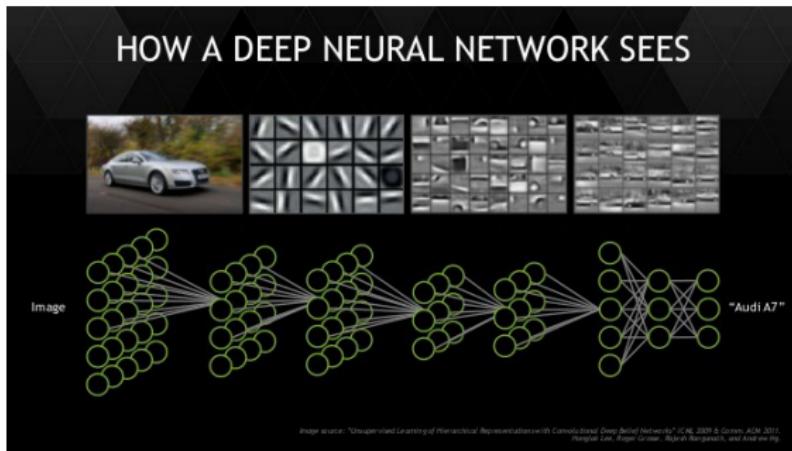
### Takaisinkyttykettä verkko (2/2)



## 6.6. Tekoälyn menetelmiä – Viikko 3 – Neuroverkot

### Konvolutioalaiset verkot

- Monipuolisempi rakenne, jossa esimerkiksi valokuvista voidaan etsiä erisuuntaisia reunuja, gradientteja jne.
- Tehokkaita esimerkiksi kuvantunnistuksessa.
- Käytetään usein yhdessä tavallisten verkkojen tai takaisinkytettyjen verkkojen kanssa.



## 6.7. Tekoälyn menetelmiä – Viikko 3 – Harjoituksia

### Harjoituksia

- (7) Onko olemassa tekoon menetelmiä, jotka eivät käytä koneoppimista? Tutki asiaa Internetin avulla.
- (8) Mitä eroa on luokittelulla ja klusteroinnilla?

## 7. pyplot – Viikko 4

### Viikon sisältö

- ➊ pyplot-kirjaston käytön perusteet

### Taustaa

- pyplot on vaihtoehtoinen rajapinta matplotlib:n oliopohjaiselle rajapinnalle.
- Joissain tapauksissa hieman rajoittuneempi
- Rajapinnan funktion kutsuminen muokkaa käsittelyssä olevaa kuvaaa:
  - Luo kuvan
  - Luo tulostusalueen kuvaan
  - Lisää graafin kuvaan
  - Lisää otsikoita
  - Vaihtaa akseleiden ominaisuuksia
  - jne.
- Tila säilyy funktiokutsujen yli.

### pyplot-esimerkki (1/9)

#### Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 N=100
5
6 t=np.arange(0, 2*np.pi, 2*np.pi/N)
7 x=np.sin(2*t)
8 y=np.cos(t)
```

---

Source file: **pyplot03.py**

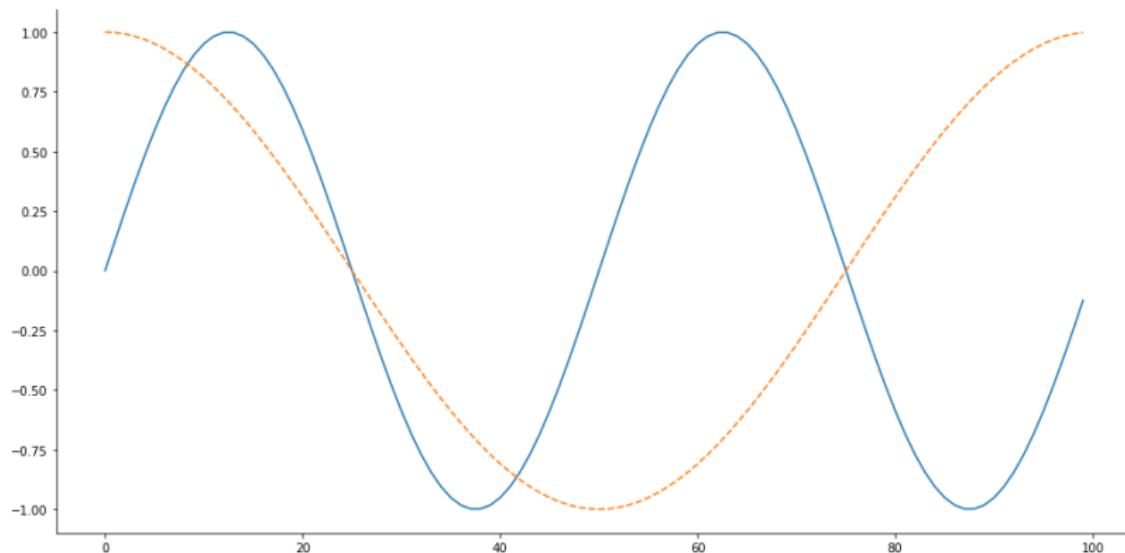
## pyplot-esimerkki (2/9)

Code

```
11 fig=plt.figure(figsize=(16, 8))
12 plt.plot(x)
13 plt.plot(y, '--')
14 plt.savefig("03_01.png", bbox_inches='tight',
15             pad_inches = 0)
16 plt.show()
```

---

Source file: **pyplot03.py**



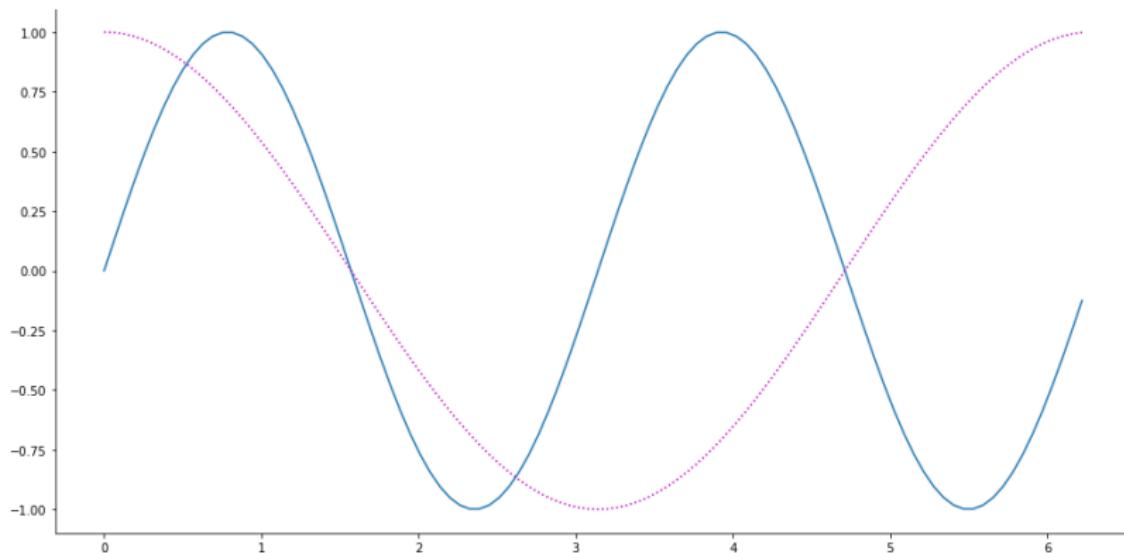
## pyplot-esimerkki (3/9)

## Code

```
17 fig=plt.figure(figsize=(16, 8))
18 plt.plot(t, x)
19 plt.plot(t, y, 'm:')
20 plt.savefig("03_02.png", bbox_inches='tight',
21             pad_inches = 0)
22 plt.show()
```

---

Source file: **pyplot03.py**



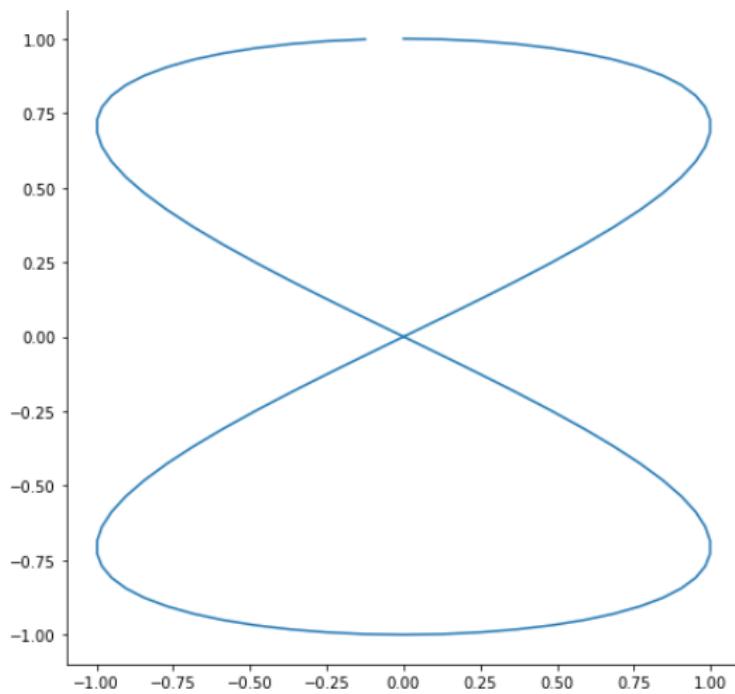
### pyplot-esimerkki (4/9)

#### Code

```
23 fig=plt.figure(figsize=(8, 8))  
24 plt.plot(x, y)  
25 plt.savefig("03_03.png", bbox_inches='tight',  
             pad_inches = 0)  
26 plt.show()
```

---

Source file: **pyplot03.py**



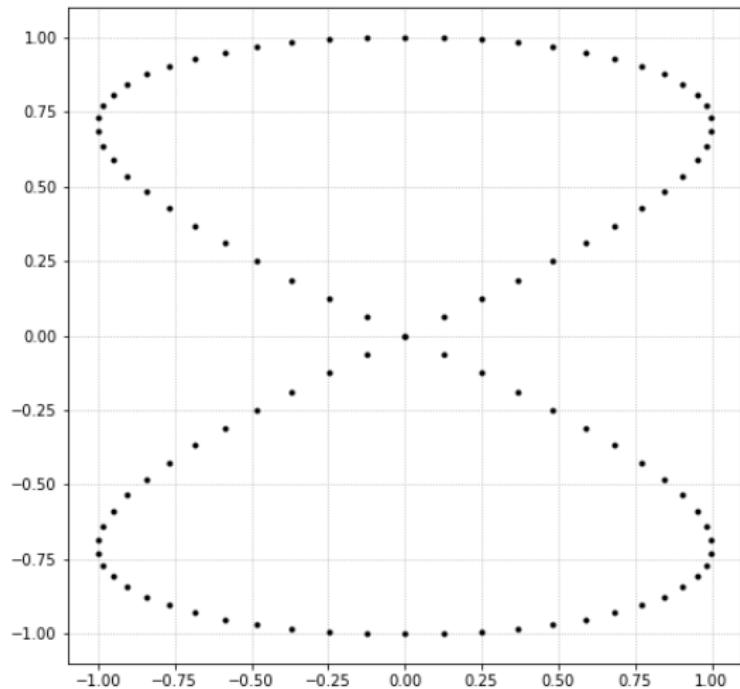
### pyplot-esimerkki (5/9)

#### Code

```
28 fig=plt.figure(figsize=(8, 8))
29 plt.plot(x, y, 'k.')
30 plt.grid(b=True, linestyle=':')
31 plt.savefig("03_04.png")
32 plt.show()
```

---

Source file: **pyplot03.py**

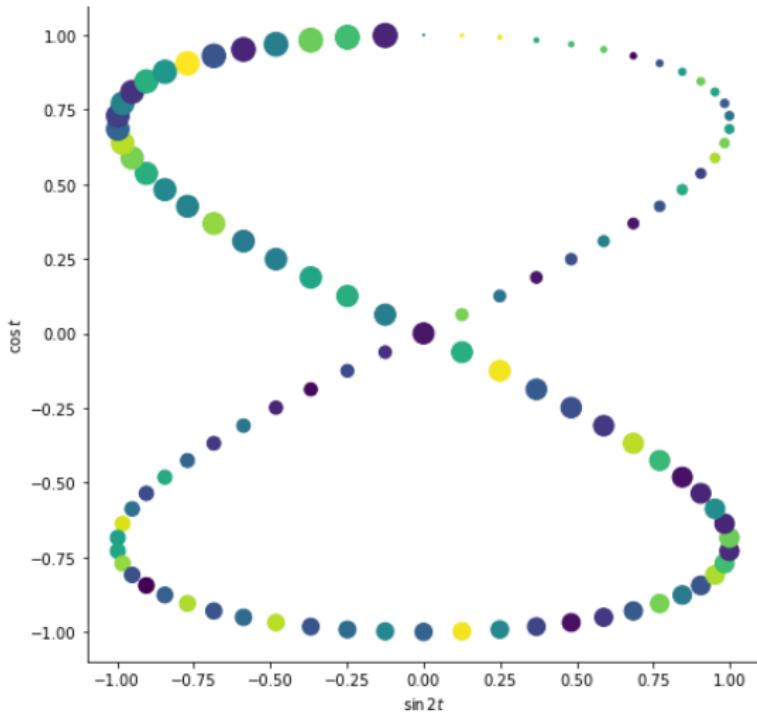


## pyplot-esimerkki (6/9)

Code

```
34 data={ 'a': x, 'b':y, 's':40*t+1, 'c': np.random.  
       randint(0, 50, size=x.shape)}  
35 plt.figure(figsize=(8,8))  
36 plt.scatter('a', 'b', s='s', c='c', data=data)  
37 plt.xlabel(r'$\sin\sqrt{2t}$')  
38 plt.ylabel(r'$\cos\sqrt{t}$')  
39 plt.savefig("03_05.png", bbox_inches='tight',  
             pad_inches = 0)  
40 plt.show()
```

Source file: **pyplot03.py**



## pyplot-esimerkki (7/9)

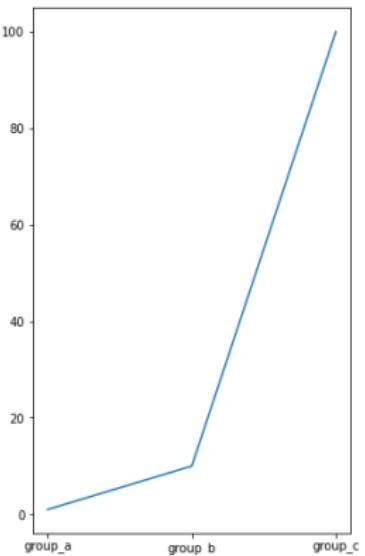
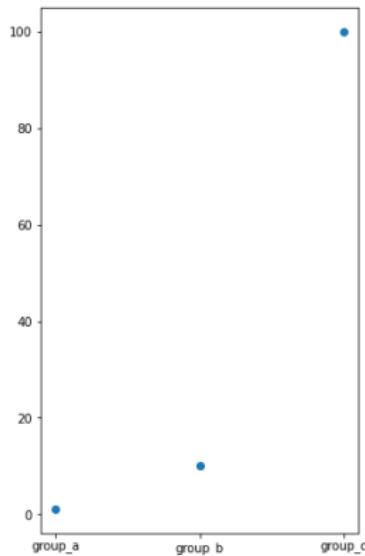
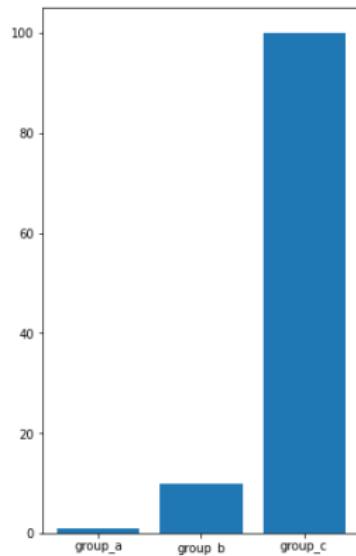
Code

```
43 names = [ 'group_a' , 'group_b' , 'group_c' ]
44 values = [1 , 10 , 100]
45 plt.figure(figsize=(16 , 8))
46 plt.subplot(131)
47 plt.bar(names , values)
48 plt.subplot(132)
49 plt.scatter(names , values)
50 plt.subplot(133)
51 plt.plot(names , values)
52 plt.suptitle('Categorical Plotting')
53 plt.savefig("03_06.png" , bbox_inches='tight' ,
54 pad_inches = 0)
55 plt.show()
```

---

Source file: **pyplot03.py**

### Categorical Plotting



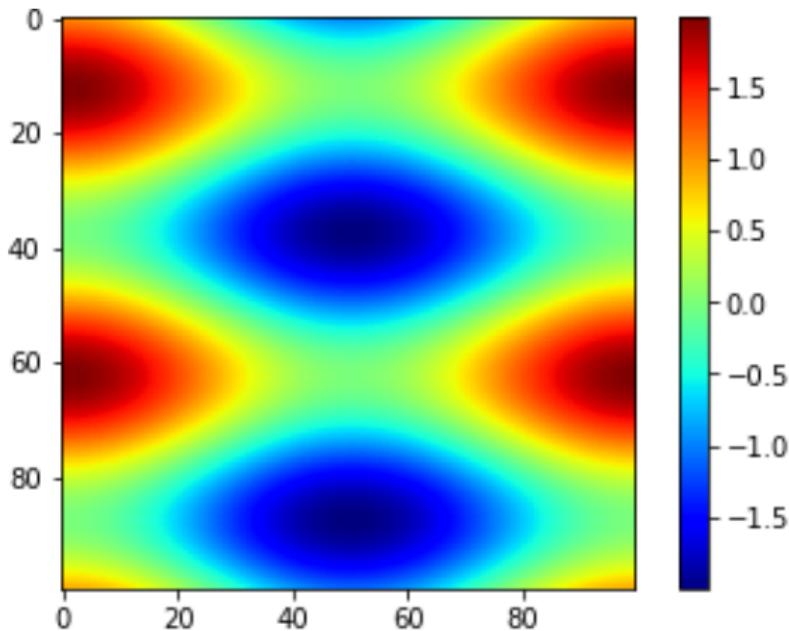
## pyplot-esimerkki (8/9)

Code

```
57 YT=np.reshape(y, (1, N))
58 X=np.reshape(x, (N, 1))
59 A=X+YT
60 del X, YT
61 plt.imshow(A, cmap='jet')
62 plt.colorbar()
63 plt.savefig("03_07.png", bbox_inches='tight',
64 pad_inches = 0)
65 plt.show()
```

---

Source file: **pyplot03.py**



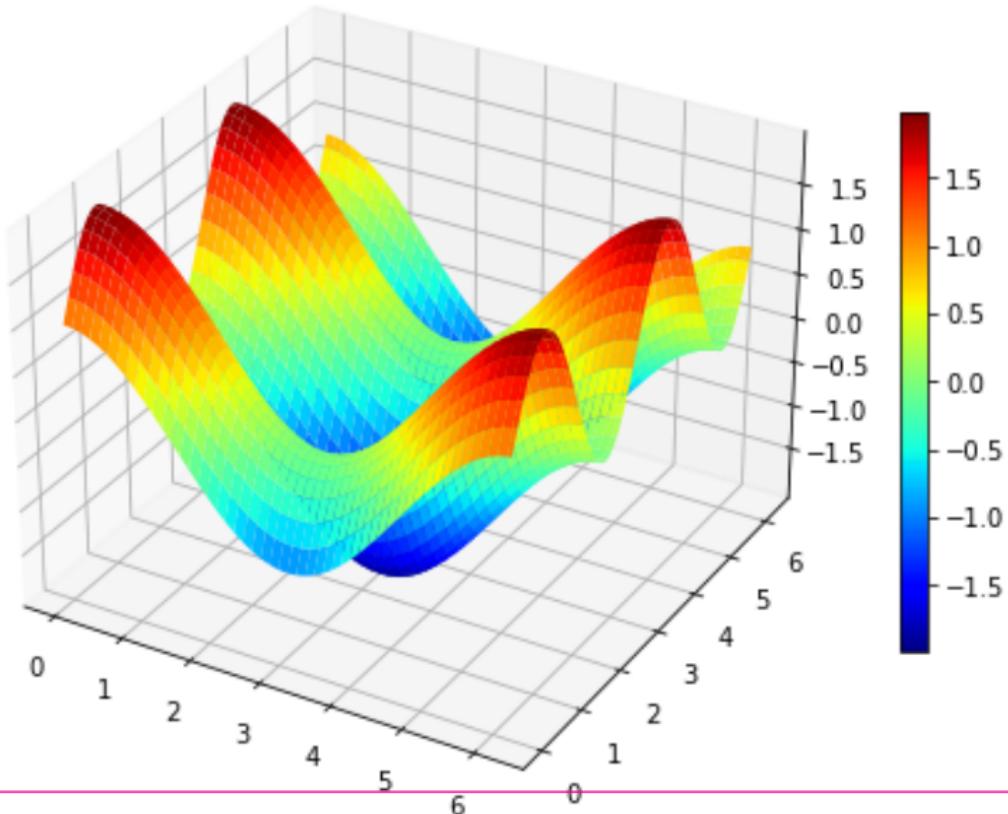
## pyplot-esimerkki (9/9)

## Code

```
67 fig=plt.figure(figsize=(8, 8))
68 ax = plt.axes(projection='3d')
69 X, Y = np.meshgrid(t, t)
70 surf=ax.plot_surface(X, Y, A, cmap='jet')
71 fig.colorbar(surf, shrink=0.5)
72 plt.savefig("03_08.png", bbox_inches='tight',
73             pad_inches = 0)
74 plt.show()
```

---

Source file: **pyplot03.py**



### Harjoituksia

- (9) Olkoon  $f(x) = x^2 - 2x - 1$ . Piirrä funktion kuvaaja välillä  $x \in [-2, 2]$ . Talleta kuvaja tiedostoon `fx.png`.
- (10) Olkoon  $f(x, y) = x^2 - 2xy - 1$ . Piirrä funktion kuvaaja välillä  $(x, y) \in [-2, 2] \times [-2, 2]$ . Valitse kuvaajatyyppi, joka mielestäsi soveltuu parhaiten tähän tapaukseen. Talleta kuvaja tiedostoon `fxy.png`.
- (11) Olkoon kahden tuotteen kuukausittainen myynti (tammi – joulukuu)  
`myynti1=[26, 43, 22, 33, 9, 1, 9, 8, 33, 11, 32, 7]` ja  
`myynti2=[16, 23, 21, 13, 19, 11, 39, 28, 23, 21, 23, 17]`.  
Piirrä molempien tuotteiden myynti samaan kuvaan. Talleta kuva nimellä `myynti.png`.

## 8. Lineaarialgebraa – Viikko 5

### Viikon sisältö

- ① Vektori
- ② Matriisi
- ③ Edellisten väliset laskutoimitukset

HUOM: Joissain kohden matemaattisesta tarkkuudesta on saatettu tinkiä, jotta asiat olisivat helpommin omaksuttavissa.

### **n-jono**

#### Definition

*n-jono on n kappaletta peräkkäin lueteltuja lukuja  $(a_1, a_2, \dots, a_n)$ .*

*Mikäli luvut on valittu reaalilukujen joukosta merkitään  
 $(a_1, a_2, \dots, a_n) \in \mathbb{R}^n$ .*

#### Example 9

- $(0, 0, 0)$  on 3-jono.
- $(1, 0)$  on 2-jono.
- $(1.2, 3.4, 1.2, 1)$  on 4-jono.

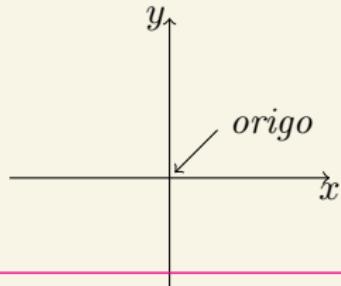
## 2-ulotteinen koordinaatisto

### Definition

- Tason koordinaatisto muodostetaan valitsemalla yksi tason piste origoksi ja kiinnittämällä siihen kaksi toisiaan vastaan kohtisuoraa koordinaattiakselia.
- Koordinaattiakseleille täytyy kiinnittää positiiviset suunnat.

### Remark

Yleensä käytetään koordinaatistoa, jossa positiiviselta  $x$ -akselilta kierryttääessä vastapäivään  $90^\circ$  tullaan positiiviselle  $y$ -akselille.



## 8.2. Lineaarialgebraa – Viikko 5 – Tason ja $\mathbb{R}^2$ :n samaistus Piste koordinaatistossa

### Remark

Jokaisen tason pisteen paikka voidaan ilmaista lukuparina  $(x, y)$ .  
Tässä

- ensimmäinen luku ( $x$ ) ilmaisee pisteen  $x$ -koordinaatin ja
- jälkimmäinen luku ( $y$ ) ilmaisee pisteen  $y$ -koordinaatin.

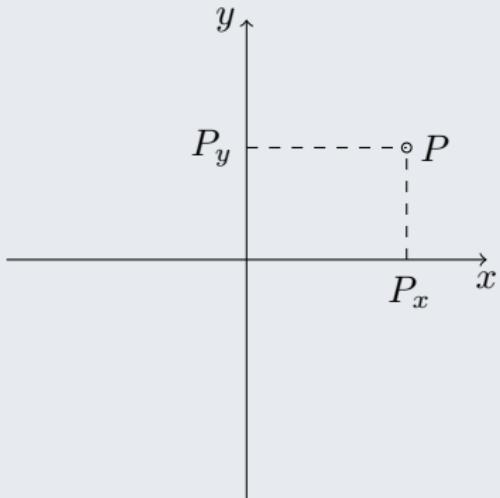
### Definition

Samaistetaan taso  $\mathbb{R}^2$ :n kanssa:

- Jokaista tason pistettä vastaa lukupari  $(x, y)$  ja
- jokaista lukuparia  $(x, y)$  vastaa yksi tason piste.
- Merkitään pistettä vastaavaa vektoria jatkossa merkinnällä  $(x \quad y)$ .

## 8.2. Lineaarialgebraa – Viikko 5 – Tason ja $\mathbb{R}^2$ :n samaistus Tason ja $\mathbb{R}^2$ :n samaistus

Example 10



Piste  $P$  samaistetaan  $\mathbb{R}^2$ :n vektorin  $(P_x \quad P_y)$  kanssa.

Remark

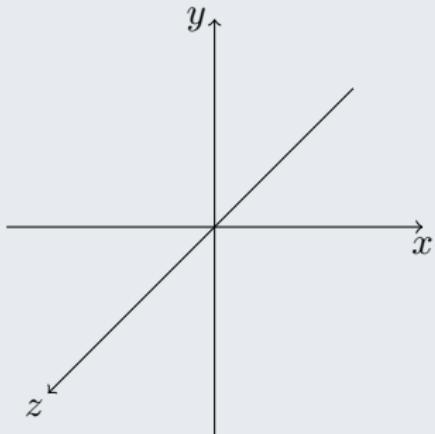
Vastaavasti 1d-avaruus voidaan samaistaa  $\mathbb{R}$ :n (lukusuora) kanssa.

**Definition**

- *3-ulotteinen avaruus ja  $\mathbb{R}^3$  voidaan samaistaa vastaavalla tavalla kuin 2d-tapauksessa.*
- *Samaistetaan piste  $P$  ja  $\mathbb{R}^3:n$  vektori  $(P_x \quad P_y \quad P_z)$ , jossa
  - $P_x$  on pisteen  $P$  x-koordinaatti,
  - $P_y$  on pisteen  $P$  y-koordinaatti ja
  - $P_z$  on pisteen  $P$  z-koordinaatti.*

## 3-ulotteinen koordinaatisto

## Example 11



- Kuvassa on oikeakäytinen koordinaatisto, jota täällä kurssilla jatkossa käytetään.
- Vasenkätisessä koordinaatistossa z-akselin suunta on päinvastainen.

### Remark

- Vastaavalla tavalla voidaan samaistaa  $n$ -ulotteinen avaruus ja  $\mathbb{R}^n$ .
- Kuvan piirtäminen on hankalaa.

### Notation

Muutetaan aiemmin käytettyä merkintää hieman:

- Numeroidaan akselit, eikä kutsuta niitä enää  $x$ -,  $y$ - ja  $z$ -akseleiksi.
- 
- Merkitään pistettä  $x$  vastaavaa  $\mathbb{R}^n$ :n vektoria  $\mathbf{x} = (x_1 \quad x_2 \quad \dots \quad x_n)$ .
- Käytetään jatkossa tässä määriteltyä esitystapaa riippumatta siitä, mikä avaruuden dimensio on.

## Yhteenlasku

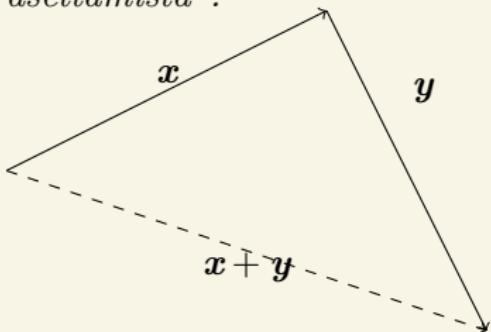
### Definition

Olkoon  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . Tällöin

$$\begin{aligned}\mathbf{x} + \mathbf{y} &= (x_1 + y_1 \quad x_2 + y_2 \quad \dots \quad x_n + y_n) \text{ ja} \\ \mathbf{x} - \mathbf{y} &= (x_1 - y_1 \quad x_2 - y_2 \quad \dots \quad x_n - y_n) .\end{aligned}$$

### Remark

Geometrisesti yhteenlasku tarkoittaa "kahden vektorin peräkkäin asettamista".



Example 12

$$\begin{aligned}(1 &\quad -1) + (2 &\quad 3) &= (3 &\quad 2) \\(0 &\quad 0 &\quad 1) + (0 &\quad -4 &\quad -1) &= (0 &\quad -4 &\quad 0) \\(\pi &\quad \ln 2) + (\sqrt{2} &\quad \ln 1) &= (\pi + \sqrt{2} &\quad \ln 2) \\(1 &\quad 2) + (3 &\quad 1 &\quad 1) && Ei määritelty! \\(1 &\quad -1) - (2 &\quad 3) &= (-1 &\quad -4)\end{aligned}$$

## Skalaarilla kertominen

### Remark

Skalaarilla tarkoitetaan  $\mathbb{R}$ :n lukua.

### Definition

Skalaarin  $a \in \mathbb{R}$  ja vektorin  $\mathbf{x} \in \mathbb{R}^n$  tulo

$$a\mathbf{x} = (ax_1 \quad ax_2 \quad \dots \quad ax_n) .$$

### Remark

Geometrisesti tämä tarkoittaa vektorin pituuden kertomista sen suunnan säilyttäen.

### Definition

Vektorin  $\mathbf{x} \in \mathbb{R}^n$  vastavektori  $-\mathbf{x} = -1\mathbf{x}$ .

#### Remark

- Kaikki jatkossa esitetyt operaatiot määritellään yleisesti  $\mathbb{R}^n$  vektoreille.

#### Definition

Vektorin  $\mathbf{x}$  pituus, eli normi

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} .$$

Remark

*Yleisesti ei päde:*

- ①  $\|\mathbf{x}\| = \|\mathbf{y}\| \Rightarrow \mathbf{x} = \mathbf{y}$
- ②  $\|\mathbf{x}\| + \|\mathbf{y}\| = \|\mathbf{x} + \mathbf{y}\|$

Theorem 1

*Olkoon  $\mathbf{x}$  vektori ja  $a \in \mathbb{R}$ . Tällöin*

$$\|a\mathbf{x}\| = |a|\|\mathbf{x}\| .$$

## Esimerkkejä

$$\begin{aligned}\|(1 \quad 2)\| &= \sqrt{1^2 + 2^2} = \sqrt{5} \\ \left\| \begin{pmatrix} -1 & -2 \end{pmatrix} \right\| &= \sqrt{(-1)^2 + (-2)^2} = \sqrt{5} \\ \left\| \begin{pmatrix} 2 & -1 & 3 \end{pmatrix} \right\| &= \sqrt{2^2 + (-1)^2 + 3^2} = \sqrt{14} \\ \|3(1 \quad 2)\| &= |3|\|(1 \quad 2)\| = 3\sqrt{5} \\ \|(1 \quad 2) + (1 \quad 1)\| &= \|(2 \quad 3)\| = \underbrace{\sqrt{13}}_{\approx 3.61} \\ &\leq \|(1 \quad 2)\| + \|(1 \quad 1)\| = \underbrace{\sqrt{5} + \sqrt{2}}_{\approx 3.65}\end{aligned}$$

## Example 13

Avaruuden  $\mathbb{R}^n$  kahden pisteen  $\mathbf{x}$  ja  $\mathbf{y}$  etäisyys voidaan laskea normin avulla.

Olkoon  $\mathbf{x} = \begin{pmatrix} 1 & 3 & -1 & 1 \end{pmatrix}$  ja  $\mathbf{y} = \begin{pmatrix} -1 & 3 & 2 & 3 \end{pmatrix}$ . Tällöin pisteiden välinen etäisyys

$$\begin{aligned}\|\mathbf{x} - \mathbf{y}\| &= \| (2 \quad 0 \quad -3 \quad -2) \| \\ &= \sqrt{2^2 + 0^2 + (-3)^2 + (-2)^2} \\ &= \sqrt{4 + 0 + 9 + 4} \\ &= \sqrt{17} \\ &= \|\mathbf{y} - \mathbf{x}\| .\end{aligned}$$

## Remark

Joskus pisteiden etäisyyttä merkitään  $d(\mathbf{x}, \mathbf{y})$ :llä.

### Yksikkövektori ja samansuuntaisuus

#### Definition

*Yksikkövektori on vektori jonka pituus on 1.*

#### Definition

- ① Olkoon  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . Tällöin  $\mathbf{x}$  on samansuuntainen kuin  $\mathbf{y}$  jos on olemassa  $a \in \mathbb{R}$  s.e  $\mathbf{x} = a\mathbf{y}$  ja  $a > 0$ .
- ② Jos  $-\mathbf{x}$  on samansuuntainen kuin  $\mathbf{y}$ , niin tällöin vektorit ovat vastakaissuuntaiset.
- ③ Jos vektorit ovat samansuuntaisia tai vastakaissuuntaisia, niin tällöin vektorit ovat yhdensuuntaiset.

Theorem 2

Olkoon  $\mathbf{x} \in \mathbb{R}^n$ ,  $\|\mathbf{x}\| \neq 0$ . Tällöin  $\mathbf{x}$ :n kanssa samansuuntainen yksikkövektori  $\hat{\mathbf{x}} = \frac{1}{\|\mathbf{x}\|} \mathbf{x}$ .

Remark

- ① Usein käytetään termiä  $\mathbf{x}$ :n suuntainen yksikkövektori.
- ② Jos  $\mathbf{x} \in \mathbb{R}^n$ , niin jatkossa  $\hat{\mathbf{x}}$  on  $\mathbf{x}$ :n suuntainen yksikkövektori.

## Example 14

Olkoon  $\mathbf{x} = \begin{pmatrix} 9 & 2 & -7 \end{pmatrix}$ . Tällöin

$$\begin{aligned}\|\mathbf{x}\| &= \sqrt{16465} \\ \hat{\mathbf{x}} &= \frac{1}{\sqrt{16465}} \mathbf{x}.\end{aligned}$$

## Example 15

Tiedetään, että  $\hat{\mathbf{x}} = \begin{pmatrix} \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \end{pmatrix}$  ja  $\|\mathbf{x}\| = \sqrt{8}$ , lasketaan  $\mathbf{x}$ .

- ① Tiedetään, että  $\hat{\mathbf{x}} = \frac{1}{\|\mathbf{x}\|} \mathbf{x}$ .
- ② Siten

$$\begin{aligned}\mathbf{x} &= \|\mathbf{x}\| \hat{\mathbf{x}} \\ &= \sqrt{8} \begin{pmatrix} \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \end{pmatrix} \\ &= \left( \sqrt{8} \sqrt{\frac{1}{2}}, -\sqrt{8} \sqrt{\frac{1}{2}} \right) \\ &= (\sqrt{4}, -\sqrt{4}) \\ &= (2, -2)\end{aligned}$$

### 8.3. Lineaarialgebraa – Viikko 5 – Muita avaruuksia, $\mathbb{R}^n$ Suuntavektori avaruuden $\mathbb{R}^n$ pisteestä toiseen

#### Example 16

Monesti tarvitaan tieto kahta avaruuden pistettä yhdistävän janan suunnasta. Tämä tieto voidaan laskea käyttämällä yksikkövektoria.

Olkoon  $\mathbf{x} = \begin{pmatrix} -1 & 2 \end{pmatrix}$  ja  $\mathbf{y} = \begin{pmatrix} 5 & 1 \end{pmatrix}$ . Tällöin suunta  $\mathbf{x}$ :stä  $\mathbf{y}$ :hyn

$$\begin{aligned}\hat{\mathbf{z}} &= \frac{1}{\|\mathbf{y} - \mathbf{x}\|} (\mathbf{y} - \mathbf{x}) \\ &= \frac{1}{\|(6 \quad -1)\|} (6 \quad -1) \\ &= \frac{1}{\sqrt{37}} (6 \quad -1) .\end{aligned}$$

Voit laskea tämän osion harjoituksia esimerikiksi numpy:llä.

- (12) Olkoon  $\mathbf{x} = \begin{pmatrix} 1 & -1 & 0 \end{pmatrix}$ ,  $\mathbf{y} = \begin{pmatrix} 3 & -2 & 0 \end{pmatrix}$ ,  $\mathbf{z} = \begin{pmatrix} 1 & 2 \end{pmatrix}$  ja  $\mathbf{w} = \begin{pmatrix} -2 & -1 \end{pmatrix}$ . Mikäli laskutoimitukset on määritelty, niin laske
- ①  $\mathbf{x} + \mathbf{y}$ ,  $\mathbf{x} - \mathbf{y}$ ,  $\mathbf{x} + \mathbf{w}$ ,  $\mathbf{z} + \mathbf{w}$  ja  $\mathbf{y} - \mathbf{y}$ .
  - ②  $\|\mathbf{x} + \mathbf{y}\|$ ,  $\|\mathbf{x} - \mathbf{y}\|$ ,  $\|\mathbf{x}\| + \|\mathbf{w}\|$ ,  $\|\mathbf{z}\| + \mathbf{w}$ ,  $\|\mathbf{z}\| + \|\mathbf{w}\|$  ja  $\|\mathbf{y}\| \mathbf{y}$ .
- (13) Olkoon vektorit kuten edellä. Laske kunkin vektorin kanssa samansuuntainen yksikkövektori.
- (14) Jos  $\hat{\mathbf{x}} = \hat{\mathbf{y}}$  ja  $\|\mathbf{x}\| = \|\mathbf{y}\|$ , niin onko  $\mathbf{x} = \mathbf{y}$ ?

## Definition

Olkoon  $a_i \in \mathbb{R}$ ,  $i = 1 \dots n$ . Määritellään

$$\sum_{i=1}^n a_i = a_1 + a_2 + \dots + a_n .$$

## Example 17

$$\sum_{i=1}^n i = 1 + 2 + 3 + \dots + n ,$$

$$\sum_{i=1}^n i^2 = 1 + 4 + 9 + \dots + n^2 \text{ ja}$$

$$\sum_{i=1}^n \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} .$$

## 8.3. Lineaarialgebraa – Viikko 5 – Muita avaruuksia, $\mathbb{R}^n$ $\sum$ -merkinnän ominaisuuksia

### Theorem 3

Olkoon  $c \in \mathbb{R}$ . Tällöin

$$c \sum_{i=1}^n a_i = \sum_{i=1}^n ca_i ,$$

$$\sum_{i=1}^n b_i + \sum_{i=1}^n a_i = \sum_{i=1}^n (b_i + a_i) \text{ ja}$$

### Remark

Huomaa erityisesti, että yleensä

$$(\sum_{j=1}^n b_j)(\sum_{i=1}^n a_i) \neq \sum_{i=1}^n (b_i a_i) .$$

Example 18

Olkoon  $a_i = i$  ja  $b_i = \frac{1}{i}$ , kun  $i = 1 \dots 3$ . Tällöin

$$\sum_{i=1}^3 a_i = 1 + 2 + 3 = 6$$

$$\sum_{i=1}^3 3a_i = 3 \sum_{i=1}^3 a_i = 3 \cdot 6 = 18$$

$$\sum_{i=1}^3 b_i = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} = \frac{11}{6}$$

- (15) Laske  $\sum_{i=1}^4 i$ .
- (16) Laske  $\sum_{i=1}^4 \frac{1}{i}$ .
- (17) Laske  $\sum_{i=1}^4 (i + \frac{1}{i})$ .
- (18) Olkoon annettu kaksi  $\mathbb{R}^3$ -vektoria  $\mathbf{x} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$  ja  
 $\mathbf{y} = \begin{pmatrix} 3 & 1 & 0 \end{pmatrix}$ . Laske  $\sum_{i=1}^3 (x_i y_i)$ .

**Pistetulo****Definition**

Olkoon  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . Tällöin pistetulo (sisätulo)

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^n x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

**Theorem 4**

Olkoon  $\mathbf{x}, \mathbf{y}$  ja  $\mathbf{z} \in \mathbb{R}^n$  sekä  $a \in \mathbb{R}$ . Tällöin

$$\begin{aligned}\mathbf{x} \cdot \mathbf{y} &= \mathbf{y} \cdot \mathbf{x}, \\ \mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) &= \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z}, \\ (a\mathbf{x}) \cdot \mathbf{y} &= a(\mathbf{x} \cdot \mathbf{y}) \text{ ja} \\ \mathbf{x} \cdot \mathbf{y} &= \|\mathbf{x}\| \|\mathbf{y}\| \cos \gamma,\end{aligned}$$

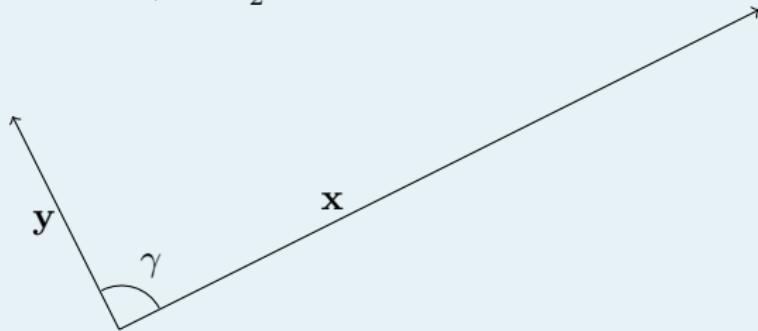
jossa  $\gamma$  on vektoreiden  $\mathbf{x}$  ja  $\mathbf{y}$  välinen kulma.

Theorem 5

Olkoon  $\mathbf{x}$  ja  $\mathbf{y} \in \mathbb{R}^n$  ja  $\mathbf{x} \neq \mathbf{0} \neq \mathbf{y}$ . Jos  $\mathbf{x} \cdot \mathbf{y} = 0$ , niin tällöin  $\cos \gamma = 0$ .

Corollary 6

Tällöin  $\gamma = \pm \frac{1}{2}\pi$ , eli vektorit ovat toisiaan vastaan kohtisuorassa.



Theorem 7

Olkoon  $\mathbf{x}$  ja  $\mathbf{y} \in \mathbb{R}^n$  ja  $\mathbf{x} \neq \mathbf{0} \neq \mathbf{y}$ . Tällöin

$$\cos \gamma = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} .$$

Remark

Edellisestä saadaan vektoreiden välinen kulma ottamalla puolittain  $\arccos$ .

Corollary 8

Yksikkövektoreille  $\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \mathbb{R}^n$

$$\cos \gamma = \hat{\mathbf{x}} \cdot \hat{\mathbf{y}} .$$

## Example 19

Olkoon  $\mathbf{x} = \begin{pmatrix} 1 & 2 & -1 & 3 \end{pmatrix}$  ja  $\mathbf{y} = \begin{pmatrix} 2 & -2 & -1 & 1 \end{pmatrix}$ . Tällöin

$$\mathbf{x} \cdot \mathbf{y} = 1 \cdot 2 - 2 \cdot 2 + 1 \cdot 1 + 3 \cdot 1 = 2$$

$$\|\mathbf{x}\| = \sqrt{1^2 + 2^2 + 1^2 + 3^2} = \sqrt{15}$$

$$\|\mathbf{y}\| = \sqrt{2^2 + 2^2 + 1^2 + 1^2} = \sqrt{10}$$

$$\begin{aligned}\cos \gamma &= \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \\ &= \frac{2}{\sqrt{150}}\end{aligned}$$

$$\Rightarrow \gamma = \arccos \frac{2}{\sqrt{150}} \approx 1.407.$$

- (19) Laske vektorien  $\mathbf{x} = (1, 1, 0)$  ja  $\mathbf{y} = (0, 0, 1)$  pistetulo  $\mathbf{x} \cdot \mathbf{y}$  ja sen avulla vektoreiden välinen kulma.
- (20) Laske vektorien  $\mathbf{x} = (2, 1, 3)$  ja  $\mathbf{y} = (1, 1, 1)$  pistetulo  $\mathbf{x} \cdot \mathbf{y}$  ja sen avulla vektoreiden välinen kulma.
- (21) Osoita, että  $\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}}$ .

## 8.3. Lineaarialgebraa – Viikko 5 – Muita avaruuksia, $\mathbb{R}^n$ Pistetulon sovellus – projektio

### Theorem 9

Olkoon  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . Tällöin  $\mathbf{x}$  voidaan jakaa  $\mathbf{y}$ :n kanssa samansuuntaiseen komponenttiin  $P_{\mathbf{y}}\mathbf{x}$  ja tätä vastaan kohtisuoraan komponenttiin  $P_{\perp\mathbf{y}}\mathbf{x}$ .

- ① Jako on yksikäsitteinen,
- ②  $P_{\mathbf{y}}\mathbf{x} = (\mathbf{x} \cdot \hat{\mathbf{y}})\hat{\mathbf{y}} = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}$  ja
- ③  $P_{\perp\mathbf{y}}\mathbf{x} = \mathbf{x} - P_{\mathbf{y}}\mathbf{x}$ .

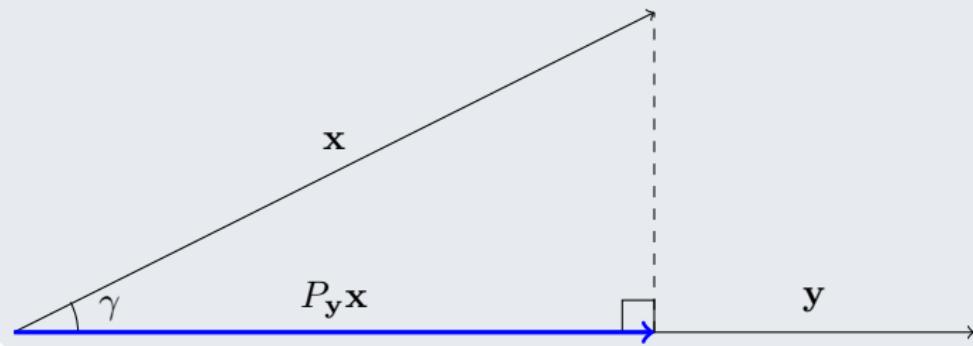
Vektoria  $P_{\mathbf{y}}\mathbf{x}$  kutsutaan  $\mathbf{x}$ :n projektioksi  $\mathbf{y}$ :lle.

### Remark

Edellä siis pätee

- ①  $\mathbf{y} \cdot P_{\perp\mathbf{y}}\mathbf{x} = 0$ ,
- ②  $P_{\mathbf{y}}\mathbf{x} \neq \mathbf{0}$  on yhdensuuntainen vektorin  $\mathbf{y}$  kanssa.
- ③  $P_{\mathbf{y}}\mathbf{x} + P_{\perp\mathbf{y}}\mathbf{x} = \mathbf{x}$ .

Example 20



**Remark**

*Projektio voidaan laske ilman neliöjuurta käyttääen identiteettiä*

$$\|\mathbf{y}\|^2 = \mathbf{y} \cdot \mathbf{y} ,$$

*jolloin*

$$P_{\mathbf{y}}\mathbf{x} = \frac{\mathbf{x} \cdot \mathbf{y}}{\mathbf{y} \cdot \mathbf{y}} \mathbf{y} .$$

## Esimerkki

Example 21

Olkoon  $\mathbf{x} = \begin{pmatrix} 1 & -\frac{1}{2} \end{pmatrix}$  ja  $\mathbf{y} = \begin{pmatrix} \frac{2}{3} & \frac{3}{2} \end{pmatrix}$ . Tällöin

$$\begin{aligned} P_{\mathbf{y}} \mathbf{x} &= \frac{\mathbf{x} \cdot \mathbf{y}}{\mathbf{y} \cdot \mathbf{y}} \mathbf{y} \\ &= \frac{\left(1 \quad -\frac{1}{2}\right) \cdot \left(\frac{2}{3} \quad \frac{3}{2}\right)}{\left(\frac{2}{3} \quad \frac{3}{2}\right) \cdot \left(\frac{2}{3} \quad \frac{3}{2}\right)} \begin{pmatrix} 1 & -\frac{1}{2} \end{pmatrix} \\ &= \frac{\frac{2}{3} - \frac{3}{4}}{\frac{4}{9} + \frac{9}{4}} \begin{pmatrix} 1 & -\frac{1}{2} \end{pmatrix} \\ &= -\frac{9}{291} \begin{pmatrix} 1 & -\frac{1}{2} \end{pmatrix} \end{aligned}$$

- (22) Olkoon  $\mathbf{x} = \begin{pmatrix} -1 & 1 \end{pmatrix}$  ja  $\mathbf{y} = \begin{pmatrix} 2 & -1 \end{pmatrix}$ . Laske  $P_{\mathbf{y}}\mathbf{x}$  ja  $P_{\mathbf{x}}\mathbf{y}$ .
- (23) Olkoon  $\mathbf{x} = \begin{pmatrix} -1 & 1 & 0 \end{pmatrix}$  ja  $\mathbf{y} = \begin{pmatrix} 2 & -1 & 2 \end{pmatrix}$ . Laske  $P_{\mathbf{y}}\mathbf{x}$  ja  $P_{\mathbf{x}}\mathbf{y}$ .

Notation

*Edellä vektorit kirjoitettiin aina vaakavektoreina ladontateknisistä syistä johtuen. Jatkossa vektori on kuitenkin aina pystyvektori jos ei toisin mainita. Vektorien laskusäännöt pysyvät kuitenkin samoina.*

Example 22

*Aikaisemmin  $\mathbf{x} = (x_1 \quad x_2 \quad \dots \quad x_n)$ , jatkossa  $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ .*

### Definition

Matriisi on  $m \times n$ -matriisi lukutaulukko, jossa on  $m$  riviä ja  $n$  saraketta. Tällä kurssilla matriisien alkiot ovat reaalilukuja.

### Notation

Merkitään jatkossa matriiseja isoilla lihavoiduilla kirjaimilla. Olkoon  $\mathbf{A}$   $m \times n$ -matriisi. Tällöin merkitään  $\mathbf{A} \in \mathbb{R}^{m \times n}$ .

### Example 23

Olkoon

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 1 & 4 \\ 2 & 4 \end{pmatrix} .$$

Tällöin  $\mathbf{A} \in \mathbb{R}^{3 \times 2}$ .

## Notation

Merkitään matriisin  $\mathbf{A}$  yksittäistä alkiota  $a_{ij}$ :llä (tai  $a_{i,j}$ :llä jos pilkku on tarpeen).  $a_{ij}$  on matriisin  $i$ :nnen rivin  $j$ :s alkio.

## Example 24

Olkoon  $A \in \mathbb{R}^{3 \times 2}$ . Tällöin

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix} .$$

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Yhteen- ja vähennyslasku

#### Definition

Matriisien väliset yhteen- ja vähennyslaskut suoritetaan alkioittain, kuten vektoreidenkin tapauksessa.

#### Example 25

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} -3 & 1 \\ 3 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -2 & 3 \\ 6 & 4\frac{1}{2} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} - \begin{pmatrix} -3 & 1 \\ 3 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ 0 & 3\frac{1}{2} \end{pmatrix}$$

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Matriisin transpoosi

#### Definition

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Tällöin matriisin  $\mathbf{A}$  transpoosi

$$\mathbf{A}^T = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}^T = \begin{pmatrix} a_{1,1} & a_{2,1} & \dots & a_{m,1} \\ a_{1,2} & a_{2,2} & \dots & a_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{m,n} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

muodostetaan “vaihtamalla matriisin rivit ja sarakkeet keskenään”.

#### Remark

Pystyvektori  $\mathbf{x}$  voidaan kirjoittaa muodossa  $\mathbf{x} = (x_1 \quad x_2 \quad \dots \quad x_n)^T$ .

## Esimerkkejä

Example 26

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

$$(8 \quad 2 \quad 0 \quad 1)^T = \begin{pmatrix} 8 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 8 \\ 2 \\ 0 \\ 1 \end{pmatrix}^T = (8 \quad 2 \quad 0 \quad 1)$$

$$\begin{pmatrix} 8 & 2 & 0 & 1 \\ -2 & 4 & 1 & 7 \end{pmatrix}^T = \begin{pmatrix} 8 & -2 \\ 2 & 4 \\ 0 & 1 \\ 1 & 7 \end{pmatrix}$$

#### Definition

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$ .

- Jos  $m = n$ , niin  $\mathbf{A}$ :n sanotaan olevan neliömatriisi.
- Jos  $n = 1$  sanotaan matriisia pystyvektoriksi.
  - Samaistetaan jatkossa  $\mathbf{R}^m$ :n vektorit ja  $\mathbb{R}^{m \times 1}$  matriisit.
- Jos  $m = 1$  sanotaan matriisia vaakavektoriksi.
- Jos  $n = m = 1$ , niin samaistetaan  $\mathbf{A}$  skalaarin  $a_{1,1} \in \mathbb{R}$  kanssa.

## Example 27

- Neliömatriisi:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

- Vaakavektori:

$$\begin{pmatrix} 8 & 2 & 0 & 1 \end{pmatrix}$$

- Pystyvektori:

$$\begin{pmatrix} 8 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

- Skalaari:

$$(4.3) = 4.3$$

#### Remark

*Neliömatriisi:*

- *Yhtä monta saraketta ja riviä.*
- *Merkitään usein  $A \in \mathbb{R}^{n \times n}$ .*
- *Samaistetaan lineaarikuvaukseen  $\mathbb{R}^n \rightarrow \mathbb{R}^n$ .*

### Definition

Matriisin  $\mathbf{A} \in \mathbb{R}^{m \times n}$  alkioita  $a_{ij}$ , joissa  $i = j$  kutsutaan  $\mathbf{A}$ :n diagonaalialkioiksi. Mikäli matriisin kaikki alkiot lukuunottamatta diagonaalialkioita ovat 0, sanotaan matriisia diagonaalimatriisiksi.

### Example 28

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ ,  $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$  ja  $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}^T$  ovat diagonaalimatriiseja.

## Notation

Diagonaalimatriisia, joka on myös neliömatriisi, voidaan merkitä lyhyemmin

$$\begin{pmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{n,n} \end{pmatrix} = \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_n \end{pmatrix} = \text{diag}(a_1, a_2, \dots, a_n).$$

## Nollamatriisi

### Definition

*Nollamatriisi*

$$\mathbf{0} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}.$$

### Remark

- $\mathbf{0}$  on matriisiyhteenlaskun nolla-alkio, eli pätee  $\mathbf{0} + \mathbf{A} = \mathbf{A}$ .
- Merkintää  $\mathbf{0}$  käytetään sekä matriisille, että vektorille. Mikäli sekaannuksen vaaraa on, käytetään nollavektorille jatkossa merkintää  $\overline{\mathbf{0}}$ .
- Nollamatriisin ei tarvitse olla neliömatriisi.

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Matriisin kertominen skalaarilla

#### Definition

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$  ja  $c \in \mathbb{R}$ . Tällöin

$$c\mathbf{A} = \begin{pmatrix} ca_{1,1} & ca_{1,2} & \dots & ca_{1,n} \\ ca_{2,1} & ca_{2,2} & \dots & ca_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ ca_{m,1} & ca_{m,2} & \dots & ca_{m,n} \end{pmatrix}.$$

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Matriisin ja vektorin välinen kertolasku (1/2)

#### Definition

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$  ja  $\mathbf{x} \in \mathbb{R}^n$ . Määritellään tällöin tulo

$$\mathbf{Ax} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n a_{1,i}x_i \\ \sum_{i=1}^n a_{2,i}x_i \\ \vdots \\ \sum_{i=1}^n a_{m,i}x_i \end{pmatrix} \in \mathbb{R}^m.$$

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Matriisin ja vektorin välinen kertolasku (2/2)

Remark

Tulosvektorin alkio paikassa  $i$  on  $(\mathbf{Ax})_i = \begin{pmatrix} a_{i,1} \\ a_{i,2} \\ \vdots \\ a_{i,n} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ .

Remark

Huomaa, että tässä määriteltiin  $\mathbf{Ax}$ , mutta ei  $\mathbf{xA}$ !

## Esimerkki

Example 29

Olkoon  $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ -1 & -2 \end{pmatrix}$  ja  $\mathbf{x} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$ . Tällöin

$$\begin{aligned}\mathbf{Ax} &= \begin{pmatrix} 2 \cdot (-1) + 1 \cdot 3 \\ (-1) \cdot (-1) + (-2) \cdot 3 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ -5 \end{pmatrix}.\end{aligned}$$

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Matriisien kertolasku (1/2)

Definition

Olkoon  $\mathbf{A} \in \mathbb{R}^{m \times n}$  ja  $\mathbf{B} \in \mathbb{R}^{n \times p}$ . Määritellään tulo

$$\begin{aligned}\mathbf{AB} &= \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,p} \\ b_{2,1} & b_{2,2} & \dots & b_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \dots & b_{n,p} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^n a_{1,i}b_{i,1} & \sum_{i=1}^n a_{1,i}b_{i,2} & \dots & \sum_{i=1}^n a_{1,i}b_{i,p} \\ \sum_{i=1}^n a_{2,i}b_{i,1} & \sum_{i=1}^n a_{2,i}b_{i,2} & \dots & \sum_{i=1}^n a_{2,i}b_{i,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^n a_{m,i}b_{i,1} & \sum_{i=1}^n a_{m,i}b_{i,2} & \dots & \sum_{i=1}^n a_{m,i}b_{i,p} \end{pmatrix} \in \mathbb{R}^{m \times p}.\end{aligned}$$

## Matriisien kertolasku (2/2)

## Remark

Määritelmästä seuraa suoraan, että tulomatriisiin rivien lukumäärän määräää  $\mathbf{A}$ :n rivien lukumäärä ja sarakkeiden lukumäärän määräää  $\mathbf{B}$ :n sarakkeiden lukumäärä. Lisäksi on huomioitava, että  $\mathbf{A}$ :ssa on oltava yhtä monta saraketta kuin  $\mathbf{B}$ :ssä on rivejä.

## Remark

Tulomatriisin riville  $i$ , sarakkeeseen  $j$  tulee kertolaskussa arvo

$$(\mathbf{AB})_{ij} = \begin{pmatrix} a_{i,1} \\ a_{i,2} \\ \vdots \\ a_{i,n} \end{pmatrix} \cdot \begin{pmatrix} b_{1,j} \\ b_{2,j} \\ \vdots \\ b_{n,j} \end{pmatrix} = \sum_{k=1}^n a_{i,k} b_{k,j} .$$

## Remark

Huomaa, että yleensä ei päde  $\mathbf{AB} = \mathbf{BA}$ .

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Matriisin ja vektorin välinen kertolasku ja pistetulo

#### Remark

*Matriisin ja vektorin kertolasku saadaan näin erikoistapauksena matriisien kertolaskusta. Lisäksi pistetulo  $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T \mathbf{y}$ .*

**Esimerkki**

Example 30

Olkoon  $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$  ja  $\mathbf{B} = \begin{pmatrix} -1 & 0 & 3 \\ 1 & 2 & 0 \end{pmatrix}$ . Tällöin

$$\begin{aligned}\mathbf{AB} &= \begin{pmatrix} 1 \cdot (-1) + 0 \cdot 1 & 1 \cdot 0 + 0 \cdot 2 & 1 \cdot 3 + 0 \cdot 0 \\ 2 \cdot (-1) + 1 \cdot 1 & 2 \cdot 0 + 1 \cdot 2 & 2 \cdot 3 + 1 \cdot 0 \end{pmatrix} \\ &= \begin{pmatrix} -1 & 0 & 3 \\ -1 & 2 & 6 \end{pmatrix}.\end{aligned}$$

## Definition

*Identiteetti- eli yksikkömatriisi*

$$\mathbf{I} = \text{diag}(1, 1, \dots, 1) = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{n \times n} .$$

## Remark

*Identiteettimatriisi on matriisikertolaskun ykkösalkio, eli sille pätee  $\mathbf{IA} = \mathbf{A}$ . Vastaavasti pätee myös  $\mathbf{BI} = \mathbf{B}$ .*

Theorem 10

Olkoon  $\mathbf{A}$ ,  $\mathbf{B}$  ja  $\mathbf{C}$  matriiseja ja  $c$  skalaari. Mikäli laskutoimitukset ovat määriteltyjä, niin

$$\begin{aligned}\mathbf{A} + \mathbf{0} &= \mathbf{A}, \\ \mathbf{A} + \mathbf{B} &= \mathbf{B} + \mathbf{A}, \\ \mathbf{A} + (\mathbf{B} + \mathbf{C}) &= (\mathbf{A} + \mathbf{B}) + \mathbf{C}, \\ c(\mathbf{A} + \mathbf{B}) &= c\mathbf{A} + c\mathbf{B}, \\ (\mathbf{cA})\mathbf{B} &= \mathbf{c}(AB) = \mathbf{A}(cB), \\ \mathbf{AO} &= \mathbf{O} \text{ ja } \mathbf{OA} = \mathbf{O}, \\ \mathbf{AI} &= \mathbf{A} \text{ ja } \mathbf{IA} = \mathbf{A}, \\ \mathbf{A}(BC) &= (AB)C, \\ \mathbf{A}(B+C) &= (AB) + (AC), \\ \mathbf{A} &= (\mathbf{A}^T)^T \text{ ja} \\ (\mathbf{AB})^T &= \mathbf{B}^T \mathbf{A}^T.\end{aligned}$$

(24) Muodosta matriisitulo  $\begin{pmatrix} 2 & 4 \\ 1 & 1 \\ 4 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \end{pmatrix}$ .

(25) Totea että  $\mathbf{AI} = \mathbf{A}$  pätee kun  $\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 5 \\ 4 & 2 & 3 \end{pmatrix}$ .

## Käänteismatriisi

### Definition

Olkoon  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , siis neliömatriisi. Mikäli on olemassa matriisi  $\mathbf{A}^{-1} \in \mathbb{R}^{n \times n}$  siten, että

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I},$$

niin  $\mathbf{A}^{-1}$  on tällöin  $\mathbf{A}$ :n käänteismatriisi ja matriisin  $\mathbf{A}$  sanotaan olevan kääntyvä (tai säännöllinen).

### Theorem 11

Mikäli käänteismatriisi on olemassa, niin

- $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I} = \mathbf{A}^{-1}\mathbf{A}$ ,
- $\mathbf{A}^{-1}$  on yksikäsittinen ja
- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$ .

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Diagonaalimatriisin käänneismatriisi

Example 31

Olkoon  $\mathbf{A} = \text{diag}(a_1, a_2, \dots, a_n)$ ,  $a_i \neq 0$  kaikilla  $i$ . Etsitään, jos on olemassa,  $\mathbf{B} \in \mathbb{R}^{n \times n}$  s.e  $\mathbf{AB} = \mathbf{I}$ .

Lasketaan

$$\mathbf{I} = \mathbf{AB} = \begin{pmatrix} a_1 b_{1,1} & a_1 b_{1,2} & \dots & a_1 b_{1,n} \\ a_2 b_{2,1} & a_2 b_{2,2} & \dots & a_2 b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_n b_{n,1} & a_n b_{n,2} & \dots & a_n b_{n,n} \end{pmatrix},$$

joten on oltava  $b_{i,j} = 0$  kun  $i \neq j$  ja  $b_{i,i} = \frac{1}{a_i}$ .

Siis  $\mathbf{A}^{-1} = \text{diag}(\frac{1}{a_1}, \frac{1}{a_2}, \dots, \frac{1}{a_n})$ .

## 8.4. Lineaarialgebraa – Viikko 5 – Matriisit

### Kiertomatriisiin käänteismatriisi

Example 32

Matriisi  $\mathbf{R}(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}$  kiertää tason pisteitä kulman  $\alpha$  verran. Etsitään, jos on olemassa,  $\mathbf{B} \in \mathbb{R}^{2 \times 2}$  s.e  $\mathbf{R}(\alpha)\mathbf{B} = \mathbf{I}$ .

Lasketaan

$$\mathbf{I} = \mathbf{R}(\alpha)\mathbf{B} = \begin{pmatrix} cb_{1,1} - sb_{2,1} & cb_{1,2} - sb_{2,2} \\ sb_{1,1} + cb_{2,1} & sb_{1,2} + cb_{2,2} \end{pmatrix},$$

joka toteutuu kun  $\mathbf{B} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \mathbf{R}(-\alpha)$ .

Siis  $\mathbf{R}(\alpha)^{-1} = \mathbf{R}(-\alpha)$ .

**Definition**

Matriisin  $\mathbf{A} \in \mathbf{R}^{n \times n}$  sanotaan olevan ortogonaalinen jos

$$\mathbf{A}\mathbf{A}^T = \mathbf{I} = \mathbf{A}^T\mathbf{A} .$$

**Corollary 12**

Mikäli matriisi  $\mathbf{A}$  on ortogonaalinen, niin sen käänneismatriisi

$$\mathbf{A}^{-1} = \mathbf{A}^T .$$

(26) Matriisi  $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$  on ortogonaalinen. Laske  $\mathbf{A}^{-1}$ .

(27) Olkoon  $\mathbf{x} = \begin{pmatrix} 3.2 \\ -2.1 \end{pmatrix}$  ja  $\mathbf{A} = \begin{pmatrix} 1 & 3 \\ -1.1 & 0.2 \end{pmatrix}$ . Laske  $\mathbf{Ax}$ .

- M5** Kirjoita funktio, joka tarkastaa onko kaksi vektoria toisiaan vastaan kohtisuorassa. Funktio palauttaa `True` tai `False`.
- M6** Kirjoita funktio, joka palauttaa kahden vektorin välisen kulman radiaaneina.
- M7** Kirjoita funktio, joka palauttaa annetun vektorin suuntaisen yksikkövektorin.
- M8** Kirjoita funktio, joka palauttaa  $\frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{y}\|}$ , kun  $\mathbf{x}$  ja  $\mathbf{y}$  annetaan funktion parametreina. Tätä kutsutaan vektorin  $\mathbf{x}$  projektioksi  $\mathbf{y}$ :lle.

## 9. Aineiston esikäsittely – Viikko 6

### Viikon sisältö

- ① Aineiston esikäsittely
- ② Aineiston lukeminen
- ③ Virheellisten mittausarvojen käsittely
- ④ Kategoristen tietojen käsittely
- ⑤ Aineiston jako opetus-, testi- ja validointiaineistoksi
- ⑥ Piirteiden skaalaus
- ⑦ Aineiston sovitus malliin ja mallin testaaminen



## The Google's 7 steps of Machine Learning in practice

- 1. Gathering data**
- 2. Preparing data**
- 3. Choosing a model**
- 4. Training**
- 5. Evaluation**
- 6. Hyperparameter tuning**
- 7. Prediction**

- ① Tietoaineiston lukeminen
- ② Puuttuvien pisteiden käsittely
- ③ Kategoristen tietojen käsittely
- ④ Jako opetus-, testi- (ja validointiaineistoon)
- ⑤ Piirteiden skaalaus
- Data Preprocessing in Python

## 9.1. Aineiston esikäsittely – Viikko 6 – Aineiston lukeminen

### Pandas

- Pandas
- Tiedon muokkaukseen ja analysointiin
- Erityisesti taulukkoille ja aikasarjoille
- **DataFrame** – Taulukkomuotoinen
- Tietojen luku ja kirjoitus tiedostoon (ja muistiin)
- **Tiedon yhdistely**
- Puuttuvan tiedon käsittely
- Osajoukkoihin jako
- jne...

## 9.1. Aineiston esikäsittely – Viikko 6 – Aineiston lukeminen Datan lukeminen (1/2)

### Code

```
1 import pandas
2 from scipy import stats
3 import numpy as np
4 from sklearn import datasets, linear_model
5 import matplotlib.pyplot as plt
6
7 #Original data: https://www.kaggle.com/quantbruce
    /real-estate-price-prediction/version/1
8 #Data modified by Janne Koponen
9
10 filename = 'RealEstate.csv'
```

---

Source file: **preprocessing01.py**

## 9.1. Aineiston esikäsittely – Viikko 6 – Aineiston lukeminen Datan lukeminen (2/2)

### Code

```
12 #Load data
13 data=pandas.read_csv(filename)
14 print("Read data shape = "+str(data.shape))
15 print()
16
17 #Remove index column
18 print("Remove index column")
19 data.drop(data.columns[0], axis=1, inplace=True)
20 print("data shape = "+str(data.shape))
21 print()
```

---

Source file: [preprocessing01.py](#)

## 9.2. Aineiston esikäsittely – Viikko 6 – Virheellisten mittaustulosten käsittely

### Puuttuvien pisteiden käsittely

#### Code

```
23 #Show lines with missing values
24 print("Lines with missing valaues:")
25 print(data[data.isnull().any(axis=1)])
26 print()
27
28 #Remove missing values
29 print("Remove missing values")
30 data.dropna(inplace=True)
31 print("data shape = "+str(data.shape))
32 print()
```

---

Source file: **preprocessing01.py**

## 9.2. Aineiston esikäsittely – Viikko 6 – Virheellisten mittaustulosten käsittely

### Kopioiden poisto

#### Code

```
34 #Remove duplicates
35 print("Remove duplicates")
36 data.drop_duplicates(inplace=True)
37 print("data shape = "+str(data.shape))
38 print()
```

---

Source file: **preprocessing01.py**

## 9.3. Aineiston esikäsittely – Viikko 6 – Kategoristen tietojen käsittely Menetelmiä

- Itsenäinen opiskelu
- One-Hot encoding
- Gray-koodi on vaihtoehtoinen tapa esittää binääriluvut, mutta se ei ole oleellinen asia tekstin ymmärtämisen kannalta.

## 9.4. Aineiston esikäsittely – Viikko 6 – Poikkeavien havaintojen poisto Z-score

---

- Z-score-esimerkki
- stats.zscore()

## 9.4. Aineiston esikäsittely – Viikko 6 – Poikkeavien havaintojen poisto Poikkeavien havaintojen poisto (1/2)

Code

```
40 #Find outliers by z-test
41 print("Find outliers by z-test")
42 z_threshold=6.0 #3 is recommended for most cases
43
44 z = np.abs(stats.zscore(data))
45 outlier=(z>=z_threshold)
46 print(data[outlier])
47
48 filtered_entries = (z < z_threshold).all(axis=1)
```

---

Source file: **preprocessing01.py**

## 9.4. Aineiston esikäsittely – Viikko 6 – Poikkeavien havaintojen poisto Poikkeavien havaintojen poisto (2/2)

### Code

```
50 #Remove outliers
51 print("Remove outliers")
52 data = data[filtered_entries]
53 print(data)
54 print()
```

---

Source file: **preprocessing01.py**

## **9.5. Aineiston esikäsittely – Viikko 6 – Jako opetus-, testi- (ja validointiaineistoon)**

### **Aineiston jakaminen**

- Todellisessa tilanteessa käytettäväää aineistoa ei ole opetustilanteessa käytössä.
- Tätä simuloidaan jakamalla aineisto opetus- ja testijoukkoon.
- Testijoukkoa ei käytetä opetuksessa, vaan sen avulla tutkitaan sitä kuinka hyvin opetus on onnistunut.
- Jos on tapahtunut ylioppimista, niin opetusjoukko sovittuu malliin huomattavasti paremmin kuin testijoukko.
- Joskus käytetään lisäksi myös validointijoukkoa.

## 9.5. Aineiston esikäsittely – Viikko 6 – Jako opetus-, testi- (ja validointiaineistoon)

### Jakaminen

#### Code

```
56 #Split training set and test set
57 train_fraction=0.8
58
59 print("Create training data set")
60 traindata=data.sample(frac=train_fraction ,
61     random_state=200) #random state is a seed
62     value
63 print("data shape = "+str(traindata.shape))
64 print()
65
66 print("Create test data set")
67 testdata=data.drop(traindata.index)
68 print("data shape = "+str(testdata.shape))
69 print()
```

Source file: preprocessing01.py

## 9.5. Aineiston esikäsittely – Viikko 6 – Jako opetus-, testi- (ja validointiaineistoon)

### Muunnetaan data NumPy-muotoon

#### Code

```
69 # Convert dataframe to numpy array
70 print("Convert df to array")
71 traindata=traindata.to_numpy()
72 print(traindata.shape)
73 print()
```

---

Source file: **preprocessing01.py**

## **9.6. Aineiston esikäsittely – Viikko 6 – Piirteiden skalaus**

### **Piirteiden skaalaus**

---

- Esimerkiksi ihmisen ikä (v) ja pituus (m) ovat suuruusluokaltaan erilaisia lukemia.
- Tällaiset suuruusluokkaerot aiheuttavat ongelmia monissa koneoppimisalgoritmeissa.
- Tämän vuoksi arvot muokataan usein ”standardijakauman” mukaiseksi. (keskiarvo=0, hajonta=1)

## **9.7. Aineiston esikäsittely – Viikko 6 – Dimensioiden vähentäminen**

### **Dimensioiden vähentäminen**

- Joskus lähdeainasto sisältää redundanttia tietoa.
- Tällöin voi olla tarpeen muokata aineistosta pienempi ulottuvuuksinen.
- Idea: Yhdistetään informaatiota korreloivista piirteistä.
- Dimension vähentämistä käsitellään myöhemmin esimerkiksi pääkomponenttianalyysin yhteydessä.

## **9.8. Aineiston esikäsittely – Viikko 6 – Sovitus malliin ja testaus**

### **Ei varsinaisesti esikäsittelyä**

- Käytettävä malli voi määräätä sen, millaista esikäsittelyä vaaditaan.
- Mallin opetus ja testaus eivät ole osa esikäsittelyä.
- Jatkossa on esimerkki siitä, miten lineaarista regressiota käytetään.

## 9.8. Aineiston esikäsittely – Viikko 6 – Sovitus malliin ja testaus

### $X_{\text{train}}$ , $Y_{\text{train}}$ ja malli

#### Code

```
85 #Linear regression
86 print("Create training X and Y")
87
88 def splitXY(data):
89     return data[:, :-1], data[:, -1:]
90
91 trainX, trainY=splitXY(traindata)
92 print(trainX.shape, trainY.shape)
```

Source file: **preprocessing01.py**

## 9.8. Aineiston esikäsittely – Viikko 6 – Sovitus malliin ja testaus Malli ja mallin opettaminen

### Code

```
95 print("Create linear regression model")
96 regr = linear_model.LinearRegression()
97
98 # Train the model using the training sets
99 print("Train the model")
100 regr.fit(trainX, trainY)
```

---

Source file: **preprocessing01.py**

### Code

```
102 #Create test x and test y
103 print("Create test X and Y")
104 testdata=testdata.to_numpy()
105 print(testdata.shape)
106 testX, testY=splitXY(testdata)
107 print(testX.shape, testY.shape)
```

---

Source file: **preprocessing01.py**

## 9.8. Aineiston esikäsittely – Viikko 6 – Sovitus malliin ja testaus

### Mallin testaus

#### Code

```
109 #Predict values
110 predY = regr.predict(testX)
111
112 #Plot prediction
113 plt.plot(testY, predY, '.')
114 plt.plot([10, 70], [10, 70])
115 plt.xlabel("real value")
116 plt.ylabel("predicted value")
117 plt.show()
```

---

Source file: **preprocessing01.py**

## 9.9. Aineiston esikäsittely – Viikko 6 – Harjoituksia

### Harjoituksia (1/2)

- M9 Lue tiedosto `weather_data.csv` ja poista siitä päiväys (`utc_timestamp`-sarake). Poista tämän jälkeen rivit, joilla on puuttuvia tietoja. Tallenna tulos tiedostoon `preprocessed.csv`. (Data from [Data Platform](#))

## 9.9. Aineiston esikäsittely – Viikko 6 – Harjoituksia

### Harjoituksia (2/2)

- M10
- 1 Lue tiedosto `time_series.csv` (Data from Data Platform)
  - 2 Hae dataasta Saksan tuulivoiman todellista tuotantoa koskevat tiedot. Näissä tiedoissa `region=DE`, `variable=wind` ja `attribute=generation_actual`.
  - 3 Poista edellä käytetyt sarakkeet, koska nyt niiden arvo on kaikilla riveillä sama.
  - 4 Poista rivit, joilla on puuttuvia tietoja tai outliereita. Käytä `z-testin` kynnysarvona 4:ää. HUOM: Anna `stats.zscore`-funktiolle parametriksi vain `data`-sarake. Ei kaikkia sarakkeita, kuten luentokalvojen esimerkissä.
  - 5 Normalisoi `data`-sarake siten, että sarakkeen pienin arvo on 0 ja suurin 1. Voit käyttää tässä esimerkiksi `sklearn.preprocessing` kirjastoa.
  - 6 Jaa data opetus- ja testijoukkoon siten, että opetusjoukkoon menee 70% käsitellystä aineistosta ja loput testijoukkoon. Talleta joukot nimillä `train.csv` ja `test.csv`.

## 10. Regressio – Viikko 7

- ① Lineaарinen regressio
- ② Polynominen regressio
- ③  $R^2$

### Lineaarinen regressio (1/2)

- Lineaarisessa regressioanalyysissä etsitään suoran parametrit siten, että suora kuvaa mittaustulokset mahdollisimman hyvin.
- Suoran yhtälö voidaan esittää muodossa

$$y = \alpha + \beta x ,$$

jossa vakiot  $\alpha$  ja  $\beta$  määritetään suoran yksikäsitteisesti.

- Olkoon mittaustulokset  $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$ .
- Näille voidaan kirjoittaa

$$y_i = \alpha + \beta x_i + \varepsilon_i ,$$

kun  $i = 1 \dots n$  ja jossa  $\varepsilon_i$  on virhetermi.

- Tutoriaali

## Lineaarinen regressio (2/2)

- Tämä voidaan kirjoittaa vektoreiden avulla

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{=\mathbf{y}} = \underbrace{\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}}_{=\mathbf{X}} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}}_{=\boldsymbol{\varepsilon}} .$$

- Tehtäväänä on löytää  $\alpha$  ja  $\beta$  siten että  $\|\boldsymbol{\varepsilon}\|$  on mahdollisimman pieni.
- Ratkaisuksi saadaan

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} .$$

## Esimerkki (1/3)

## Lineaarinen regressio matriisien avulla

## Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 N_meas=11
5
6 #real values
7 b_real=0.7
8 a_real=11.2
9
10 #generate measurement data
11 data_x=np.linspace(-5, 5, N_meas)
12 data_y=a_real+b_real*data_x+np.random.randn(
    N_meas)
```

---

Source file: **regression01.py**

## Esimerkki (2/3)

## Code

```
14 #create matrices
15 X=np.ones((N_meas, 2))
16 X[:, 1]=data_x
17 Y=np.reshape(data_y, (N_meas, 1))
18
19 #compute a and b
20 ab=np.linalg.inv(X.T @ X) @ X.T @ Y
21 a_estimate=ab[0, 0]
22 b_estimate=ab[1, 0]
23
24 #compute estimated values
25 y_estimate=a_estimate+b_estimate*data_x
```

---

Source file: **regression01.py**

### Esimerkki (3/3)

#### Code

```
27 #plot figure
28 fig=plt.figure(figsize=(16, 8))
29 plt.plot(data_x, data_y, '*')
30 plt.plot(data_x, y_estimate)
31 #plt.savefig("regression01.png")
32 plt.show()
```

---

Source file: **regression01.py**

## 10. Regressio – Viikko 7

`sklearn.linear_model`

- Edellä esitetty analyysi yleistyy myös tilanteisiin, joissa lähtö- ja maaliavaruuden dimensio on korkeampi.
- Laskentaan voidaan käyttää esimerkiksi `sklearn.linear_model`-kirjastoa.

## Esimerkki (1/3)

## Lineaarinen regressio sklearn-kirjaston avulla

Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn import linear_model
4
5 N_meas=11
6
7 #real values
8 b_real=0.7
9 a_real=11.2
10
11 #generate measurement data
12 data_x=np.linspace(-5, 5, N_meas)
13 data_y=a_real+b_real*data_x+np.random.randn(
    N_meas)
```

Source file: regression02.py

## Esimerkki (2/3)

## Code

```
15 #convert to matrix
16 data_x = data_x.reshape((N_meas, 1))
17 data_y = data_y.reshape((N_meas, 1))
18
19 #create linear regression model
20 regr = linear_model.LinearRegression()
21
22 #train the model using the training set
23 regr.fit(data_x, data_y)
```

---

Source file: **regression02.py**

## Esimerkki (3/3)

## Code

```
25 #predict values
26 y_estimate = regr.predict(data_x)
27
28 print('b_estimate =', regr.coef_[0,0])
29 print('a_estimate =', regr.intercept_[0])
30
31 #plot figure
32 fig=plt.figure(figsize=(16, 8))
33 plt.plot(data_x, data_y, '*')
34 plt.plot(data_x, y_estimate)
35 #plt.savefig("regression02.png")
36 plt.show()
```

---

Source file: **regression02.py**

## Code

```
1 import numpy as np
2 from sklearn import linear_model
3 import pickle
4
5 N_meas=200
6 #Input R^3, output R^2
7
8 #real values
9 b_real=np.array([[1.8, -0.5, -1], [-0.4, 8.0,
10      2]])
11 a_real=np.array([11.2, 0.2])
12 n_dim_in=np.shape(b_real)[1]
13 n_dim_out=np.shape(b_real)[0]
```

---

Source file: **regression03.py**

## Code

```
15 #generate measurement data
16 data_x=10*(np.random.rand(N_meas, n_dim_in)-0.5)
17 data_y=a_real+data_x@b_real.T+np.random.randn(
    N_meas, n_dim_out)
18
19 #create linear regression model
20 regr = linear_model.LinearRegression()
21
22 #train the model using the training set
23 regr.fit(data_x, data_y)
```

---

Source file: **regression03.py**

## Code

```
25 print("y=a+x@b.T")
26 print("input shape    =", np.shape(data_x))
27 print("output shape   =", np.shape(data_y))
28 print("b shape        =", np.shape(regr.coef_))
29 print("a shape        =", np.shape(regr.intercept_))
30 print()
31
32 print('b_estimate =\n', regr.coef_, '\nb_real =\n',
      , b_real)
33 print()
34 print('a_estimate =\n', regr.intercept_, '\n
      na_real =\n', a_real)
35 print()
```

---

Source file: **regression03.py**

## Code

```
37 # save the model to disk with pickle
38 filename = 'regression03_model.sav'
39 pickle.dump(regr, open(filename, 'wb'))
40
41 del regr, data_x
42
43 # load the model from disk
44 regr_loaded = pickle.load(open(filename, 'rb'))
```

---

Source file: **regression03.py**

## Code

```
46 #predict values
47 data_x=10*(np.random.rand(N_meas, n_dim_in)-0.5)
48 y_estimate = regr_loaded.predict(data_x)
49 print('b_loaded =\n', regr_loaded.coef_, '\
    nb_real =\n', b_real)
50 print()
51 print('a_loaded =\n', regr_loaded.intercept_, '\
    na_real =\n', a_real)
52 print()
```

---

Source file: **regression03.py**

## Polynominen regressio (1/2)

- $n$ -asteen polynomi:

$$\begin{aligned}y &= \beta_0 + \beta_1 x + \cdots + \beta_n x^n \\&= (1 \quad x \quad \dots \quad x^n) \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}\end{aligned}$$

- Vektoreiden avulla kirjoitettu yhtälöryhmä, kun on  $m$  mittausta,

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}}_{=\mathbf{y}} = \underbrace{\begin{pmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & \dots & x_2^n \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^n \end{pmatrix}}_{=\mathbf{X}} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}}_{=\boldsymbol{\beta}} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}}_{=\boldsymbol{\varepsilon}}.$$

### Polynominen regressio (2/2)

- Tehtävänä on löytää  $\beta$  siten että  $\|\varepsilon\|$  on mahdollisimman pieni.
- Ratkaisuksi saadaan

$$\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} .$$

- Polynominen tehtävä saadaan siis muunnettua samaan muotoon kuin lineaarinen tehtävä.
- Muunnos tehdään  
`sklearn.preprocessing.PolynomialFeatures`-funktion avulla.
- Älä käytä liian korkea-asteista polynomia! (ylisovittuminen)
- **Tutoriaali**

- Mitta, jonka avulla voi mitata kuinka hyvin malli sopii havaintoihin.
- 0 – Malli ei selitä havaintoja ollenkaan.
- 1 – Malli selittää havainnot täysin.
- Ylisovittumisen yhteydessä voi antaa epärealistisen hyviä arvoja.
- On olemassa myös muita vastaavia mittareita.
- **Investopedia:R-squared**
- **sklearn.metrics.r2\_score**

## Esimerkki (1/7)

Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn import linear_model
4 from sklearn.metrics import r2_score
5 from sklearn.preprocessing import
    PolynomialFeatures
6
7 N_meas = 31
8
9 #real values
10 c_real = -0.3
11 b_real = 0.7
12 a_real = 11.2
13
14 modeldegree = 2
```

---

Source file: **regression04.py**

## Esimerkki (2/7)

Code

```
16 #generate measurement data
17 data_x=np.linspace(-10, 10, N_meas)
18 data_y=a_real+b_real*data_x+c_real*(data_x**2)+3*
    np.random.randn(N_meas)
19
20 #convert to matrix
21 data_x=data_x.reshape((N_meas, 1))
22 data_y=data_y.reshape((N_meas, 1))
```

---

Source file: **regression04.py**

### Esimerkki (3/7)

Code

```
24 #####  
25 #create linear regression model  
26 linregr = linear_model.LinearRegression()  
27  
28 linregr.fit(data_x, data_y)  
29  
30 y_estimate = linregr.predict(data_x)
```

---

Source file: **regression04.py**

## Esimerkki (4/7)

Code

```
32 print(32*'-')
33 print('Linear model')
34 print()
35 print('linear b_estimate =', linregr.coef_[0,0])
36 print('linear a_estimate =', linregr.intercept_
      [0])
37 print()
38 print('r^2                  =', r2_score(data_y,
      y_estimate))
39 print()
```

---

Source file: **regression04.py**

## Esimerkki (5/7)

## Code

```
41 #####  
42 #create polynomial regression model  
43  
44 #create X matrix (convert to linear form)  
45 poly_reg=PolynomialFeatures(degree=modeldegree)  
46 X_poly=poly_reg.fit_transform(data_x)  
47 #print('X_poly =', X_poly[1:10, :])  
48  
49 #fit data to linear model  
50 linreg2=linear_model.LinearRegression()  
51 linreg2.fit(X_poly,data_y)  
52  
53 y_estimate2=linreg2.predict(poly_reg.  
    fit_transform(data_x))
```

---

Source file: **regression04.py**

## Esimerkki (6/7)

Code

```
55 print(32*'-')
56 print('Polynomial model, degree =', modeldegree)
57 print()
58 print('polyn c_estimate =', linreg2.coef_[0,2])
59 print('polyn b_estimate =', linreg2.coef_[0,1])
60 print('polyn a_estimate =', linreg2.intercept_
       [0])
61 print()
62 print('r^2 =', r2_score(data_y,
                           y_estimate2))
63 print()
64 print(32*'-')
65 print()
66 print('c_real =', c_real)
67 print('b_real =', b_real)
68 print('a_real =', a_real)
```

Source file: regression04.py

### Esimerkki (7/7)

#### Code

```
71 fig=plt.figure(figsize=(16, 8))
72 plt.plot(data_x, data_y, '*')
73 plt.plot(data_x, y_estimate, ':')
74 plt.plot(data_x, y_estimate2,color='blue')
75 plt.savefig("regression04.png")
76 plt.show()
```

---

Source file: **regression04.py**

### Harjoituksia

- M11 Fysiikan laboratoriotaan mittauspöytäkirja on talletettu tiedostoon<sup>1</sup> `measurements.csv`. Kappaleen massa on 0.75 kg. Tehtävänäsi on tarvittaessa esikäsitellä data ja määrittää mitatun aineen **ominaislämpökapasiteetti**. Ohjelma tulostaa ominaislämpökapasiteetin lukuarvon<sup>2</sup> yksikössä  $\frac{\text{kJ}}{\text{kg K}}$ . (noin 0.45)
- M12 Konenäkölaitteisto mittaa kuulantyönnön harjoituksissa kuulan  $x$ - ja  $y$ -koordinaattia 50 ms välein.  $y$ -koordinaatin 0-taso on kentän pinnan tasolla. Yhden työnnön esikäsitellyt mittaustulokset on talletettu tiedostoon `mittaus.csv`. Tehtävänäsi on ennustaa kuulan laskeutumispaikan  $x$ -koordinaatti, eli työnnön pituus. Ilmanvastus jätetään huomioimatta. Ohjelma tulostaa kuulan laskeutumispaikan  $x$ -koordinaatin (metreissä, ilman yksikköä). (noin 14 m)

---

<sup>1</sup> $T$ =kappaleen lämpötila ( $^{\circ}\text{C}$ ),  $E$ =kappaleen luovuttama energia (kJ)

<sup>2</sup>Ohjelma ei saa tulostaa mitään muuta, ei siis yksikkökäään.

## 11. Pääkomponenttianalyysi (PCA) – Viikko 8

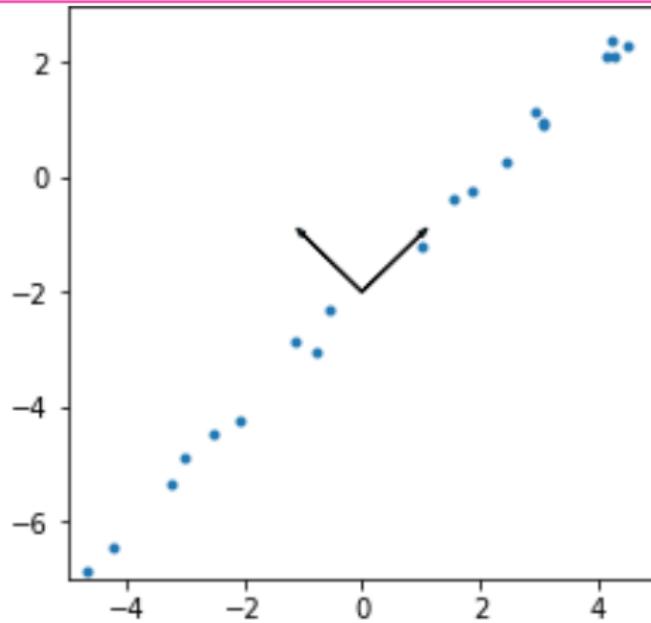
## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Viikon sisältö

- ① Miksi datan suuri dimensio voi olla ongelma
- ② PCA:n käyttö datan dimension pienentämiseen

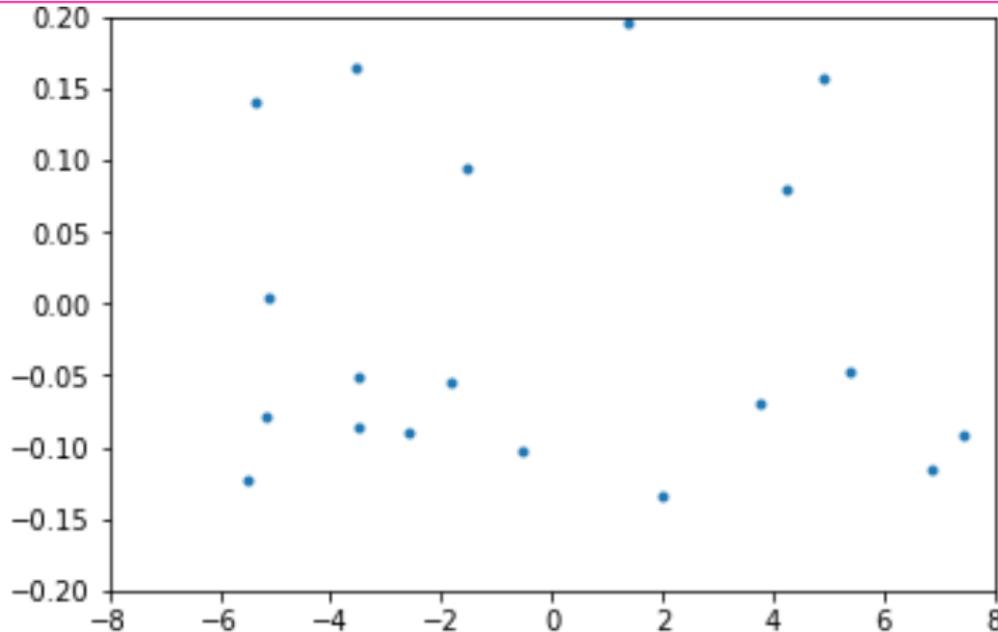
## 11. Pääkomponenttianalyysi (PCA) – Viikko 8

### Johdanto (1/4)



- Kuvassa pisteiden  $x$ - ja  $y$ -koordinaattien välillä on huomattava riippuvuuus.

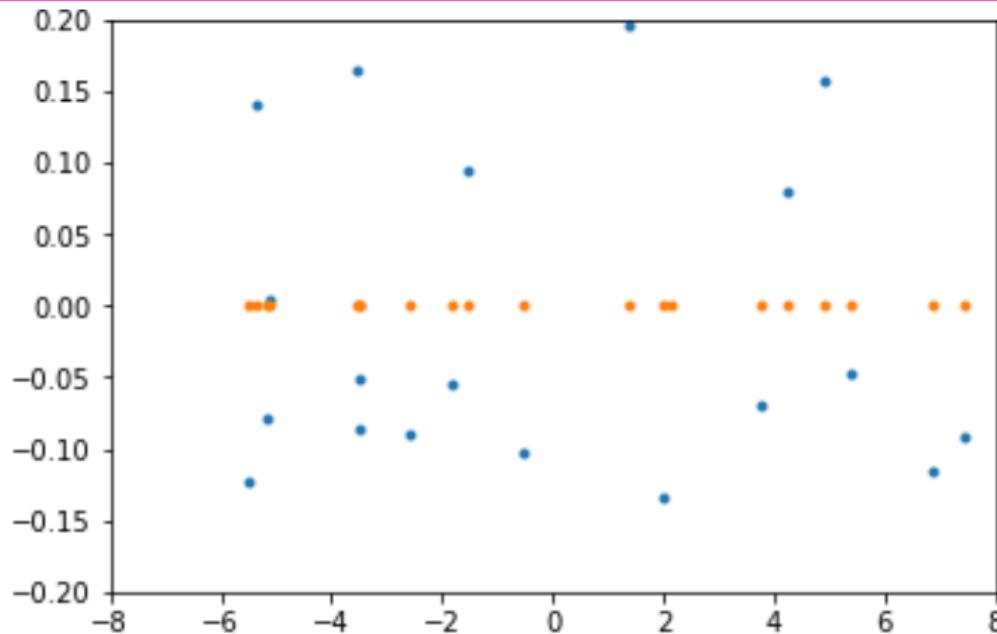
## Johdanto (2/4)



- Pisteet on siirretty uuteen koordinaatistoon, jossa  $x$ -koordinaatti selittää mahdollisimman paljon pisteiden informaatiosta.

## 11. Pääkomponenttianalyysi (PCA) – Viikko 8

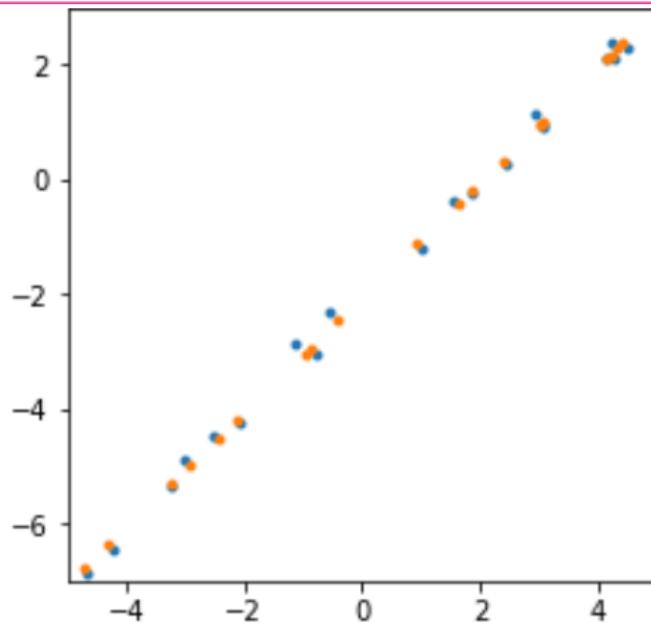
### Johdanto (3/4)



- Asetetaan  $y$ -koordinaatti arvoon 0.

## 11. Pääkomponenttianalyysi (PCA) – Viikko 8

### Johdanto (4/4)



- Edelliset pisteet on siirretty takaisin alkuperäiseen koordinaatistoon.

- PCA using Python
- PCA on menetelmä datan dimensioiden lukumäärän pienentämiseen.
- Usein tietojoukon muuttujat korreloivat keskenään.  
⇒ Yhden (tai muutaman) muuttujan arvosta voidaan ennustaa suhtellisen tarkasti jonkin toisen muuttujan arvo samassa mittauksessa.
- Esimerkiksi henkilön pituus, paino ja vyötärön ympärys korreloivat keskenään.
- Korrelaatio ei tarkoita sitä, että edellisessä esimerkissä esimerkiksi pituuden ja painon perusteella voidaan ennustaa vyötärön ympärys suhteellisen tarkasti.
- Eri syistä halutaan vähentää muuttujien määrää, jolloin tieto voidaan ”pakata” pienempään muuttujamäärään PCA:n avulla.
- PCA tarjoaa myös keinon arvioida ”pakkaamisen” aiheuttamaa informaation menetystä.

## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Korkeiden ulottuvuuksien kirous

- Wikipedia: Curse of Dimensionality
- On esitetty, että jokaista lähtödatan ulottuvuutta kohden vaaditaan 5 opetusjoukon alkiota.
- Suuridimensioinen lähtödata vaatii suuren opetusjoukon.
- On siis edullista, jos lähtödatan dimensio on mahdollisimman pieni.

## Esimerkki (1/9)

## Code

```
1 import pandas
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.decomposition import PCA
5 from sklearn import linear_model
6
7
8 #http://mreed.umtri.umich.edu/mreed/downloads.
9      html
9 #Male dataset!
10 #We have preprocessed data set which does not
11      contain missing values nor duplicates
11 filename="../../share/data/ANSUR2_2012/ANSUR
12      II MALE Public.csv"
12 train_fraction = 0.8
13 Y_column='Weightlbs'
```

Source file: **pca02.py**

## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Esimerkki (2/9)

Code

```
15 #Load data
16 data=pandas.read_csv(filename, encoding = "ISO
17 -8859-1")
18 print("Read data shape = "+str(data.shape))
19 print()
20
21 #Remove index column, all text columns, and
22 # weightkg
23 for col in ["weightkg", "Gender", "Date",
24     "Installation", "Component", "Branch", "PrimaryMOS",
25     "SubjectsBirthLocation", "SubjectNumericRace",
26     "Ethnicity", "DODRace", "WritingPreference",
27     "subjectid"]:
28     print("Remove "+col)
29     data.drop(col, axis=1, inplace=True)
```

---

Source file: **pca02.py**

## Code

```
25 print("data shape = "+str(data.shape))
26 print()
27
28 def splitXY(d):
29     return d.drop(Y_column, axis=1), d[Y_column]
30
31 print("Create training data set")
32 traindata=data.sample(frac=train_fraction,
33                     random_state=200)
34 trainX, trainY=splitXY(traindata)
35
36 print("Create test data set")
37 testdata=data.drop(traindata.index)
38 testX, testY=splitXY(testdata)
```

---

Source file: **pca02.py**

### Esimerkki (4/9)

Code

```
39 #Remove unused datasets
40 del data,testdata,traindata
41
42 print("trainX shape = "+str(trainX.shape))
43 print("trainY shape = "+str(trainY.shape))
44 print("testX shape = "+str(testX.shape))
45 print("testY shape = "+str(testY.shape))
46 print()
```

---

Source file: **pca02.py**

## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Esimerkki (5/9)

Code

```
49 # Compute full PCA
50 print("Compute full PCA")
51 pca = PCA()
52 pca.fit(trainX)
53
54 plt.plot(pca.explained_variance_)
55 plt.show()
56
57 plt.plot(pca.explained_variance_)
58 plt.semilogy()
59 plt.show()
```

---

Source file: **pca02.py**

## 11. Pääkomponenttianalyysi (PCA) – Viikko 8

### Esimerkki (6/9)

#### Code

```
61 #PCA with reduced dimension, try values 1, 2, 5,  
62     10  
63 for packed_dimension in [1, 2, 5, 50]:  
64     print(30*"-")  
65     print("Compute reduced dimension PCA, n =",  
66           packed_dimension)  
67     del pca  
68     pca = PCA(packed_dimension)  
69     pca.fit(trainX)
```

---

Source file: **pca02.py**

## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Esimerkki (7/9)

#### Code

```
70     pca_trainX=pca.transform(trainX)
71     pca_testX=pca.transform(testX)
72     print("pca_trainX shape = "+str(pca_trainX.
73         shape))
73     print("pca_testX shape = "+str(pca_testX.
74         shape))
74     print()
75
76
77     print("Create linear regression model")
78     regr = linear_model.LinearRegression()
79
80     # Train the model using the training sets
81     print("Train the model")
82     regr.fit(pca_trainX, trainY)
```

---

Source file: **pca02.py**

## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Esimerkki (8/9)

#### Code

```
84     #Predict values
85     print("Predict values")
86     predY = regr.predict(pca_testX)
87
88     plt.plot(testY, predY, '.', label=str(
89         packed_dimension))
```

---

Source file: **pca02.py**

## 11. Pääkomponenttianalyysi (PCA)– Viikko 8

### Esimerkki (9/9)

Code

```
90 plt.plot([90, 340],[90, 340], 'k:')
```

```
91 plt.xlabel("Weight (lbs)")
```

```
92 plt.ylabel("Predicted weight (lbs)")
```

```
93 plt.legend()
```

```
94 plt.show()
```

---

Source file: **pca02.py**

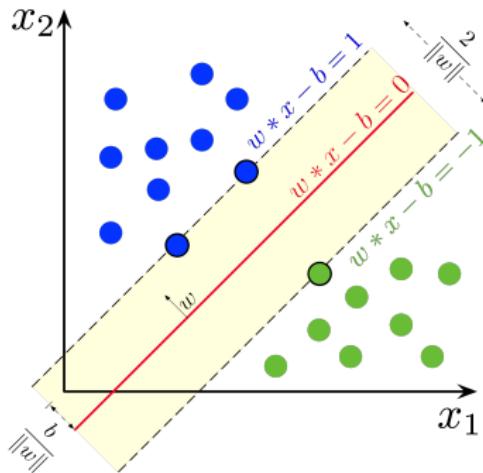
- M13 Toteuta ohjelma, joka lukee mittaustulokset `in.npy`-tiedostosta ja pakaa ne PCA:n avulla pienempään dimensioon. Määrää pienempi dimensio  $n$  siten, että `explained_variance_`:n sisältämistä (ominais)arvoista pienin mukaan otettava on  $\frac{1}{10}$  suurimmasta arvosta. Talleta pakattu data tiedostoon `out.npy`.
- M14 Muokkaa ohjelmaa, joka opettaa KNN-luokittimen tunnistamaan käsinkirjoitettuja numeroita (MNIST-aineistosta). Tehtävänäsi on pienentää aineiston dimensio 32:een. Tavoitteena on saada vähintään 95% pakatun testiaineiston merkeistä tunnistettua. Älä muokkaa ohjelmaa muualta kuin rivien 15...25 välistä.

## 12. Luokittelu

### Viikon sisältö

- ① Tukivektorikone
- ② KNN-luokitin

## Tukivektorikone



- Luokittelumenetelmä
- Opetusvaiheessa haetaan hypertaso, joka jakaa luokat mahdollisimman suurella marginaalilla.
- On olemassa myös muunnoksia, jossa jakava pinta voi olla monimutkaisempikin ja mahdollisia luokkia enemmän kuin 2.

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (1/8)

#### Code

```
1 import pandas
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.utils import shuffle
5 from sklearn.model_selection import
    train_test_split
6 from sklearn import svm
```

---

Source file: **svm01.py**

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (2/8)

#### Code

```
8 #http://mreed.umtri.umich.edu/mreed/downloads.  
9     html  
9 #Male dataset!  
10 #We have preprocessed data set which does not  
11     contain missing values nor duplicates  
11 filename1="../../share/data/ANSUR2_2012/ANSUR  
12     II MALE Public.csv"  
12 filename2="../../share/data/ANSUR2_2012/ANSUR  
13     II FEMALE Public.csv"  
13 train_fraction = 0.8  
14 Y_column='Gender'
```

---

Source file: **svm01.py**

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (3/8)

#### Code

```
16 #Load data
17 data1=pandas.read_csv(filename1)
18 data2=pandas.read_csv(filename2)
19 data=pandas.concat([data1, data2])
20 del data1, data2
21 print("Read data shape = "+str(data.shape))
22 print()
```

---

Source file: **svm01.py**

## Code

```
24 #Remove index column and all text columns
25 for col in ["SubjectId", "Date", "Installation", "
26     Component", "Branch", "PrimaryMOS", "
27     SubjectsBirthLocation", "SubjectNumericRace", "
28     Ethnicity", "DODRace", "WritingPreference", "
29     subjectid"]:
30     print("Remove "+col)
31     data.drop(col, axis=1, inplace=True)
32 print()
33
34 #Replace Male->0, Female->1
35 print("Convert gender to number")
36 data['Gender'].replace(['Female', 'Male'], [0, 1],
37     inplace=True)
38 print("data shape = "+str(data.shape))
39 print()
```

Source file: **svm01.py**

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (5/8)

Code

```
36 #Shuffle data set
37 print("Shuffle data")
38 data=shuffle(data)
39 print("data shape = "+str(data.shape))
40 print()
```

---

Source file: **svm01.py**

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (6/8)

#### Code

```
42 def splitXY(d):
43     return d.drop(Y_column, axis=1), d[Y_column]
44
45 print("Split training/test data set")
46 traindata,testdata = train_test_split(data,
47     test_size=1-train_fraction)
47 trainX, trainY=splitXY(traindata)
48 testX, testY=splitXY(testdata)
49 print()
```

---

Source file: **svm01.py**

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (7/8)

#### Code

```
51 print("Standardize data")
52 stdev=trainX.std(axis=0)
53 mean=trainX.mean(axis=0)
54 trainX=(trainX-mean)/stdev
55 testX=(testX-mean)/stdev
56 print()
57
58 #Remove unused datasets
59 del data,testdata,traindata
60
61 print("trainX shape = "+str(trainX.shape))
62 print("trainY shape = "+str(trainY.shape))
63 print("testX shape = "+str(testX.shape))
64 print("testY shape = "+str(testY.shape))
65 print()
```

---

Source file: **svm01.py**

## 12.1. Luokittelu – Tukivektorikone (SVM) – Viikko 9

### Esimerkki (8/8)

#### Code

```
67 classifier = svm.SVC()
68 classifier.fit(trainX, trainY)
69
70 predY=classifier.predict(testX)
71
72 print("Faulty predictions:", (np.abs(testY)-np.
    abs(predY)).sum())
```

---

Source file: **svm01.py**

## 12.2. Luokittelu – K-lähimmän naapurin menetelmä (KNN) KNN-luokitin

KNN-luokitin

### Harjoituksia (1/2)

- M15 Kirjoita ohjelma, joka lukee opetusdataan tiedostoista `teach_data.npy` ja `teach_class.npy`. Jälkimmäinen tiedosto sisältää kunkin datapisteen luokan. Jaa data opetus- ja testijoukkoon ja opeta silä valitsemasi luokittelija. Luokittele tiedoston `data_in.npy` aineisto. Talleta luokittelun tulos tiedostoon `data_classified.npy`. Tämä tiedosto on samaa muotoa kuin `teach_class.npy`. Vaatimuksena on vähintään 92% tarkkuus luokittelussa.

### Harjoituksia (1/2)

M16 Tiedosto `grading.csv` sisältää tiedot opiskelijoiden 3 eri harjoituksesta saamista pisteistä ja tiedon tentin läpäisemisestä.

- Kustakin harjoituksesta voi saada 0...6 pistettä.
- Mikäli jonkin sarakkeen arvo on tyhjä, opiskelija ei ole osallistunut harjoitukseen tai tenttiin. Tällaista opiskelijaa ei tule huomioida aineistossa.

Esikäsittele aineisto ja opeta sen perusteella SVM jakamaan opiskelijat kahteen luokkaan tentin läpäisen perusteella. Jaa aineisto opetus- ja testiaineistoon haluamallasi tavalla.

Tiedostossa `assignments.csv` on opiskelijoiden harjoituspisteet.

Tehtävänäsi on ennustaa vähintään 80% opiskelijoista oikea tenttitulos. Tallenna ennusteet tiedostoon `prediction.csv`, joka sisältää opiskelijan nimen 1. sarakkeessa ja ennusteen 2. sarakkeessa.

## 13. Klusterointi – Viikko 11

### Viikon sisältö

- ① K-means
- ② Hierarkinen klusterointi

Viikon 9 sisältö

### Yleistä klusteroinnista

- Luokittelussa tiedettiin etukäteen luokkien lukumäärä ja opetusjoukon alkioiden luokat.
- Entä jos aineistoa ei ole annotoitu ja
- ei välttämättä tiedetä edes luokkien lukumäärää?
- Aineistoa voidaan koettaa klusteroida erilaisilla algoritmeilla.

### K-means

- Idea:
  - ① Arvo  $N$  kappaletta klustereiden keskipisteitä
  - ② Kiinnitä kukaan aineiston piste lähimpään keskipisteeseen. Nämä muodostuu  $N$  klusteria.
  - ③ Siirrä kukaan klusterin keskipiste muodostuneen klusterin pisteiden keskipisteeseen.
  - ④ Jos askeleessa 2 jokin piste siirtyi klustereiden välillä, hyppää kohtaan 1.
  - ⑤ Valmis
- Algoritmi ei toimi ”hankalan muotoisilla” klustereilla.
- Orange: `clustering01.ows`
- Lisätietoja:
  - Teemu Holopaisen gradu
  - Wikipedia
  - Orange, YouTube
  - Orange, YouTube

## 13.2. Klusterointi – Viikko 11 – Hierarkinen klusterointi

### Hierarkinen klusterointi

- Orange, YouTube
- Wikipedia
- Hierarkinen klusterointi

M17

klusteroinnin tehtävät

## 14. Neuroverkot – Viikko 12

### Viikon sisältö

- ① foo

Viikon 10 tavoitteet

- foo

neuroverkkojen tehtävät

M18

## 15. Konvolutioonalaiset neuroverkot – Viikko 13

## 15. Konvolutioonaiset neuroverkot – Viikko 13

### Viikon sisältö

- ① foo

Viikon 11 tavoitteet

- foo

## 15.1. Konvolutioonaiset neuroverkot – Viikko 13 – Harjoituksia Harjoituksia

M19

CNN-harjoitukset

## 16. Rekurrentit neuroverkot – Viikko 14

### Viikon tavoitteet

- ① foo

Viikon 12 tavoitteet

- foo

## 16.1. Rekurrentit neuroverkot – Viikko 14 – Harjoituksia

### Harjoituksia

M20

RNN-harjoituukset

## 17. Kertaus – Viikko 15

### Viikon tavoitteet

- ① Kerrataan kurssin sisältöjä
- ② Aloitetaan laajahko harjoitustyö, joka palautetaan seuraavalla viikolla.

Keksi aihe harjoitustyölle

- foo

Kertausharjoituksia

M21

## 18. Tekoälyn tulevaisuus – Viikko 16

### Viikon sisältö

- ① Tulevaisuden pohtiminen
- ② AWS- ja Azure-palikat

- foo

## 18.1. Tekoälyn tulevaisuus – Viikko 16 – Harjoituksia

### Harjoituksia

M22

Harjoituksia liittyen TÄ:n tulevaisuuteen