Johdatus tekoälyyn

Luento 6.10.2011: Koneoppiminen

Patrik Hoyer

[Kysykää ja kommentoikaa luennon aikana!]

Koneoppiminen?

- Määritelmä:

  kone = tietokone, tietokoneohjelma

  oppiminen = ongelmanratkaisukyvyn parantuminen kokemuksen avulla

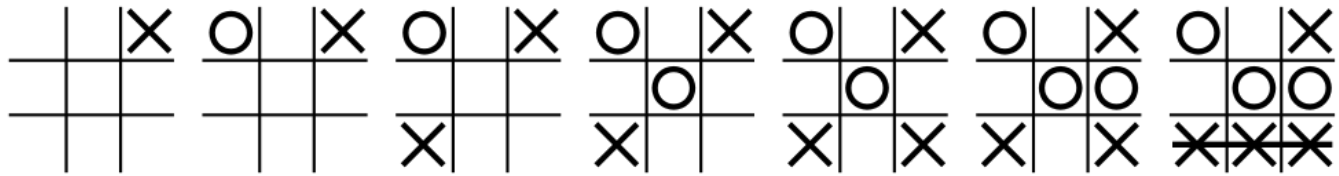
- Toisin sanoen...

  - sen sijaan että ohjelmoija kirjoittaa tarkat säännöt jonkun ongelman ratkaisuun, ohjelmoija ohjeistaa tietokonetta <u>oppimaan esimerkeistä</u>

  - monissa tapauksissa tietokoneohjelma voi tällöin oppia ratkaisemaan jotain tehtävää paremmin kuin ohjelmoija itse osaa!

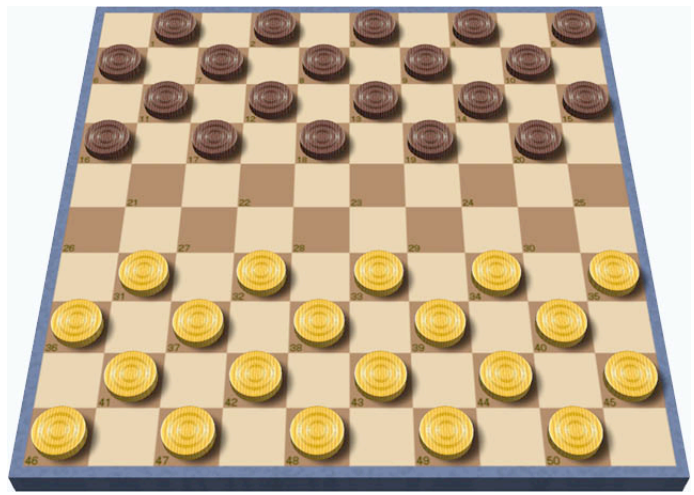
Esimerkki 1

- Miten ohjelmoida tietokonetta pelaamaan esim ristinollaa?



- Vaihtoehto 1: Katso koko pelipuu läpi, valitse optimaalisesti (vain hyvin yksinkertaiset pelit!)
- Vaihtoehto 2: Ohjelmoija kirjoittaa säännöt, tyyliin 'jos vastustajalla on kahden suora ja kolmas on vapaa, estä suora laittamalla oma merkki siihen', jne (työlästä, vaikeaa!)
- Vaihtoehto 3: Tietokoneohjelma kokeilee erilaisia sääntöjä, ja pelaa itseään (tai muita) vastaan kokeillen millä säännöillä voittaa ja millä säännöillä hävitään, oppien näin voittamaan ('koneoppiminen')

- Arthur Samuel (50-60 luvuilla):
  - Tietokoneohjelma oppii pelaamaan tammea
  - Ohjelma pelaa itseään vastaan tuhansia kertoja, oppii mitkä positiot ovat hyviä, mitkä huonoja (sen perusteella, kuinka usein ne johtavat voittoon/häviöön)
  - Ohjelma tulee lopulta paremmaksi kuin sitä ‘opettava’ ohjelmoija



Ongelman ymmärtäminen

Eräs koneoppimisen määritelmä on että tietokone parantaa suorituskykyään suorittaessaan jotain tiettyä tehtävää sitä mukaan kun se näkee esimerkkejä.

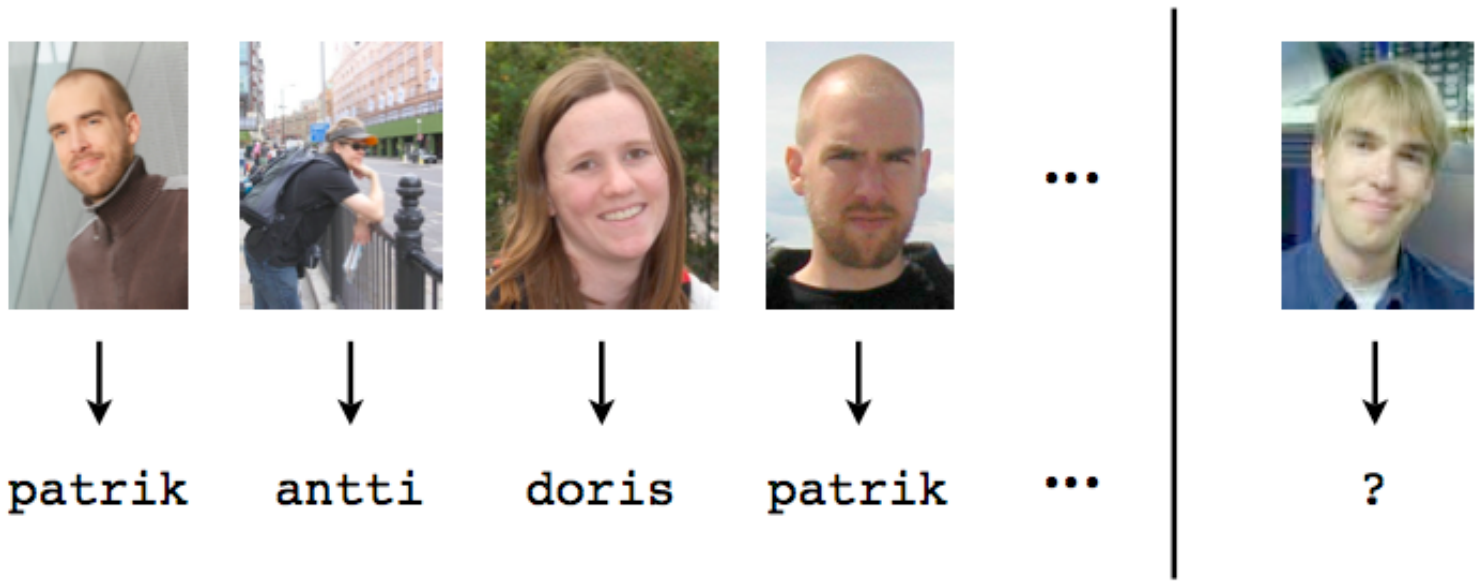
Siispä täytyy erottaa:

- Tehtävä: Mitä kone/ohjelma yrittää tehdä? Minkä ongelman se ratkaisee?
- Hyvyysmitta: Miten mitataan kuinka hyvin kone/ohjelma ratkaisee tehtävää?
- Esimerkit/data: Mikä on se kokemuspohja/esimerkkidata jonka perusteella kone/ohjelma oppii?

Esimerkki 2

Automaattinen kasvojentunnistus (facebook, apple, ...):

- ~~Ohjelmoija kirjoittaa sääntöjä: 'Jos tumma lyhyt tukka, joskus silmälasit, iso nenä, niin se on Mikko'~~ (ei toimi! miten tunnistaa 'tukan'? miten arvioida 'nenän' koko? jne...)
- Tietokoneelle näytetään (kuva, nimi)-esimerkkipareja, tietokone oppii itse mitkä 'piirteet' ovat relevantteja (vaikea ongelma, mutta mahdollista jos riittävästi esimerkkejä!)



Esimerkki 3

Roskapostin suodatus:

- ~~Ohjelmoija kirjoittaa sääntöjä: 'Jos sisältää 'viagra', niin se on roskaa'~~ (vaikeaa, eikä käyttäjälle räätälöity)
- Käyttäjä ohjeistaa tietokonetta, mitkä postit ovat roskaa, tietokone oppii itse mitä sanoja pitäisi katsoa!

From: medshop@spam.com Subject: viagra cheap meds...	spam
From: my.professor@helsinki.fi Subject: important information here's how to ace the test...	non-spam
⋮	⋮
From: mike@example.org Subject: you need to see this how to win $1,000,000...	?

Esimerkki 4

Hakulausekkeiden ennustaminen:

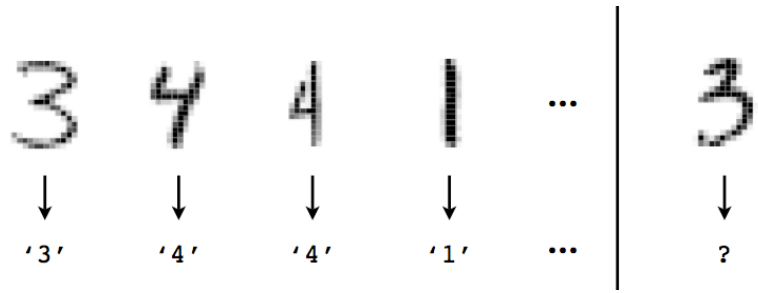
- ~~Ohjelmoija antaa valmiin sanakirjan.~~ (sanat muuttuu!)
- Edelliset hakulausekkeet käytetään esimerkkeinä!

Hakutulosten järjestäminen:

- ~~Ohjelmoija kirjoittaa tarkan kaavan jolla pisteytetään sanan esiintymismäärä, linkkien määrä, y.m. ja sitten järjestetään pisteiden mukaan.~~ (mistä ohjelmoija tietää miten eri sivun ominaisuudet pitäisi painottaa?)
- Tallennetaan mitkä linkit valitaan minkäkin hakulausekkeen jälkeen, ja laitetaan suosituimmat sivut kärkeen (yritetään ennustaa mitä käyttäjä haluaa!)

Esimerkki 5

Käsinkirjoitettujen merkkien tunnistaminen:



(postin lajittelu, vanhojen kirjojen digitointi, ...)

Esimerkki 6

Autot jotka eivät tarvitse kuljettajaa:

http://www.youtube.com/watch?v=Atmk07Otu9U

Esimerkki 7

Suositusjärjestelmät:

- amazon.com: “jos ostit tämän kirjan, saatat myös olla kiinnostunut tästä toisesta kirjasta”
- netflix.com: käyttäjät arvioivat elokuvia, järjestelmä ehdottaa sen pohjalta käyttäjille uusia elokuvia (‘netflix prize’: $1.000.000 palkinto, vuodet 2006-2009)

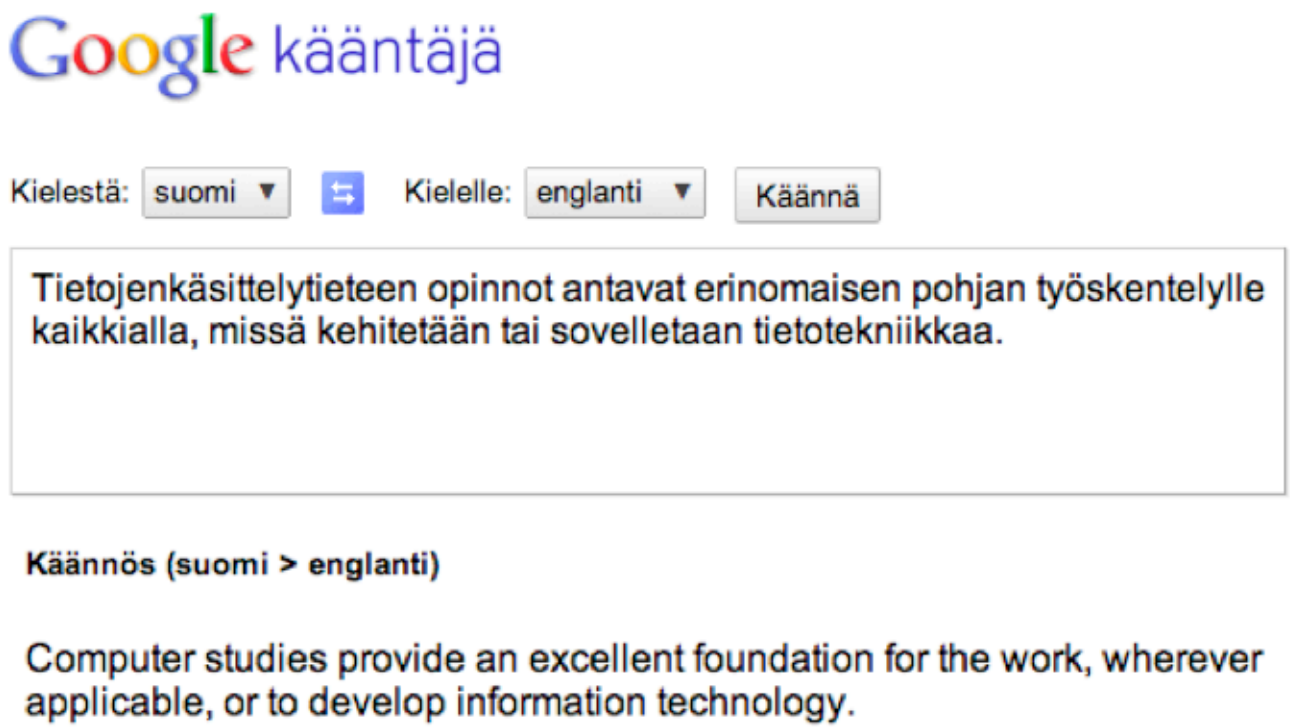
	Seven	Fargo	Aliens	Leon	Avatar
Linda	4	5	5	1	2
		3		4	3		
Jack	1		4		1	5	1
Bill			?	4	1		?
Lucy		2	1	1		5	
John	1			1	4		5
	4				5		5
	2		3				3	

http://www.youtube.com/watch?feature=player_detailpage&v=ImpV70uLxyw

Esimerkki 8

Konekääntäminen:

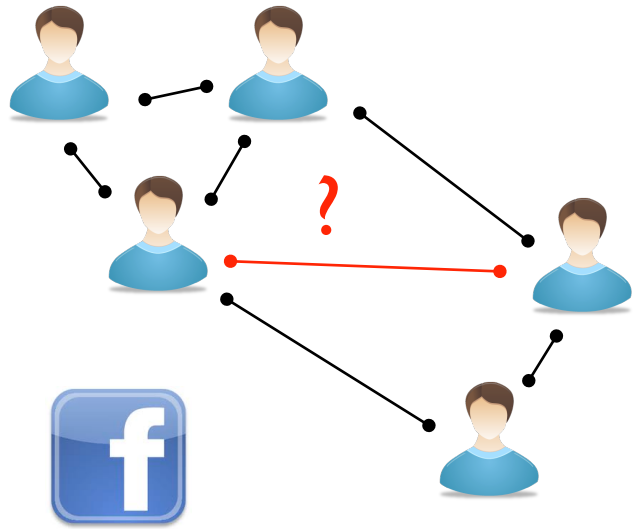
- Perinteinen tapa: Sanakirja ja kielioppi
- Nykyään (enemmän ja enemmän): Tilastollinen konekääntäminen, joka perustuu esimerkkeihin/dataan



Esimerkki 9

‘Kavereiden’ ehdottaminen:

- Voiko facebook kaverigraafin perusteella arvata tuntevatko kaksi henkilöä toisensa vaiko ei?
- Osaako twitter arvata ketä olisit kiinnostunut ‘seuraamaan’?



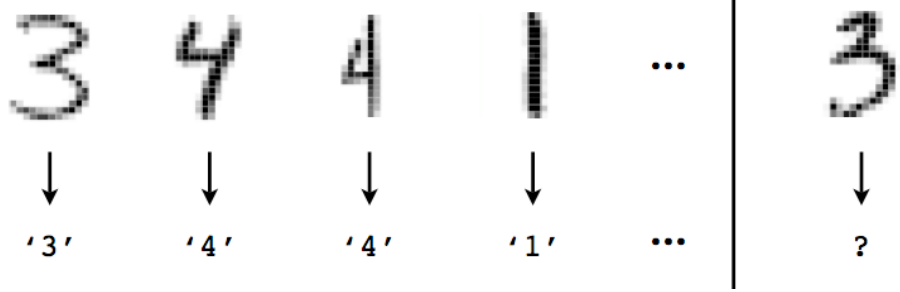
Miksi koneoppimista?

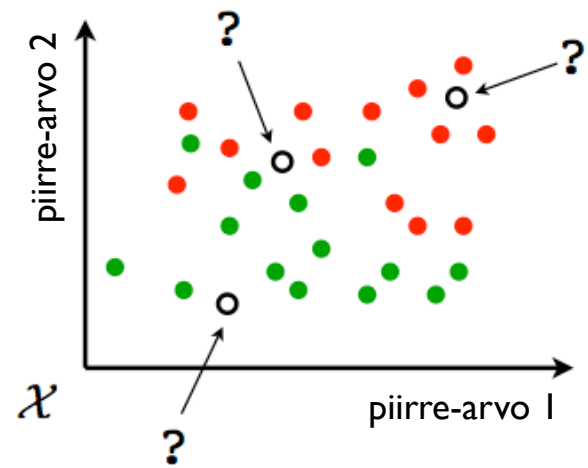
- Nykypäivänä on helppoa ja halpaa:
  - Kerätä dataa (halvat sensorit, paljon jo digitaalista)
  - Tallentaa dataa (kovalevytila halpaa)
  - Lähettää dataa (lähes ilmaista netissä)
- Joten: Kaikki keräävät suuria määriä dataa!
  - Yritykset: kaupat (ostotapahtumat), hakukoneet (hakulausekkeet, valinnat), finanssisektori (osakkeet, valuuttakurssit), tehtaat (erilaiset sensorit), sosiaalinen media (facebook, twitter, ...), kaikki palvelimet
  - Tiede: geenisekvenssit, geenieksпressio, hiukkaskokeet fysiikassa, tähtitiede, ...
- Mutta: Miten hyödyntää sitä?

Koneoppimisen lajit

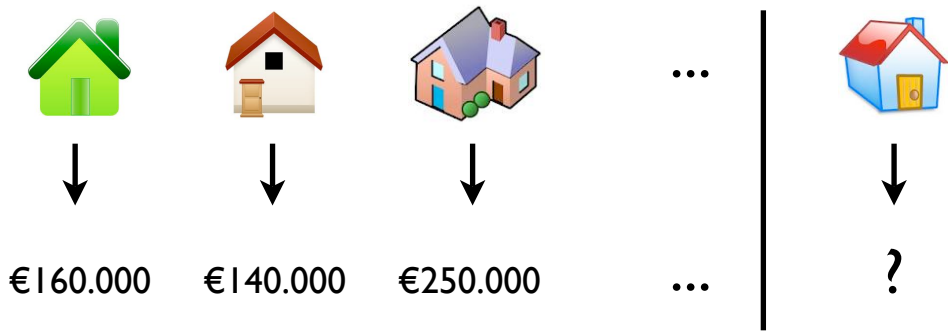
- Ohjattu oppiminen:
  - Esimerkit ovat pareja (x, y), tavoitteena on oppia ennustamaan y annettuna x.

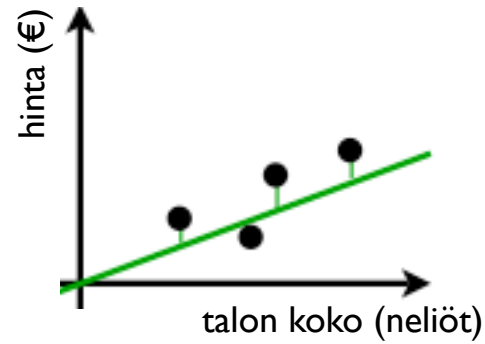
Luokittelu (diskreetti ‘y’):





Regressio (jatkuva ‘y’):

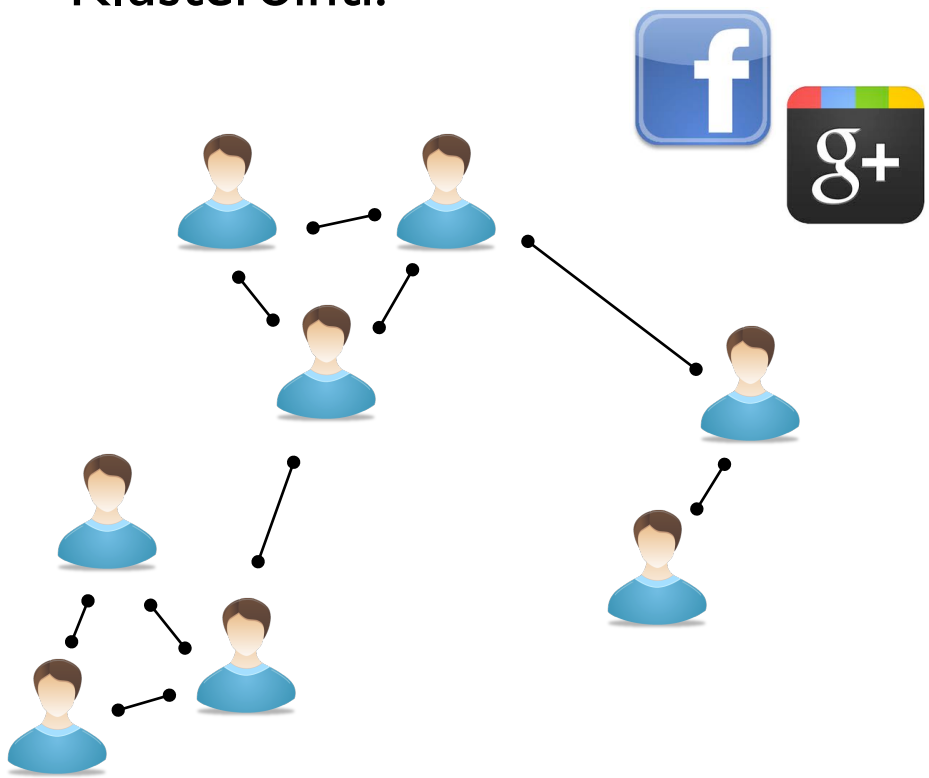


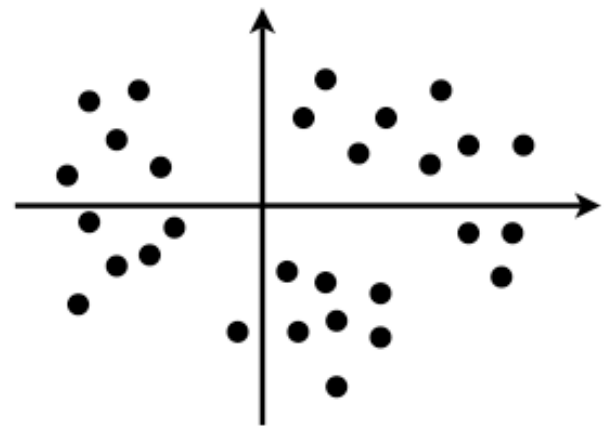


Koneoppimisen lajit

- Ohjaamaton oppiminen:
  - Esimerkit eivät sisällä ‘oikeaa vastausta’ y. Sen sijaan tavoitteena on vain ymmärtää annettu datajoukko.

Klusterointi:

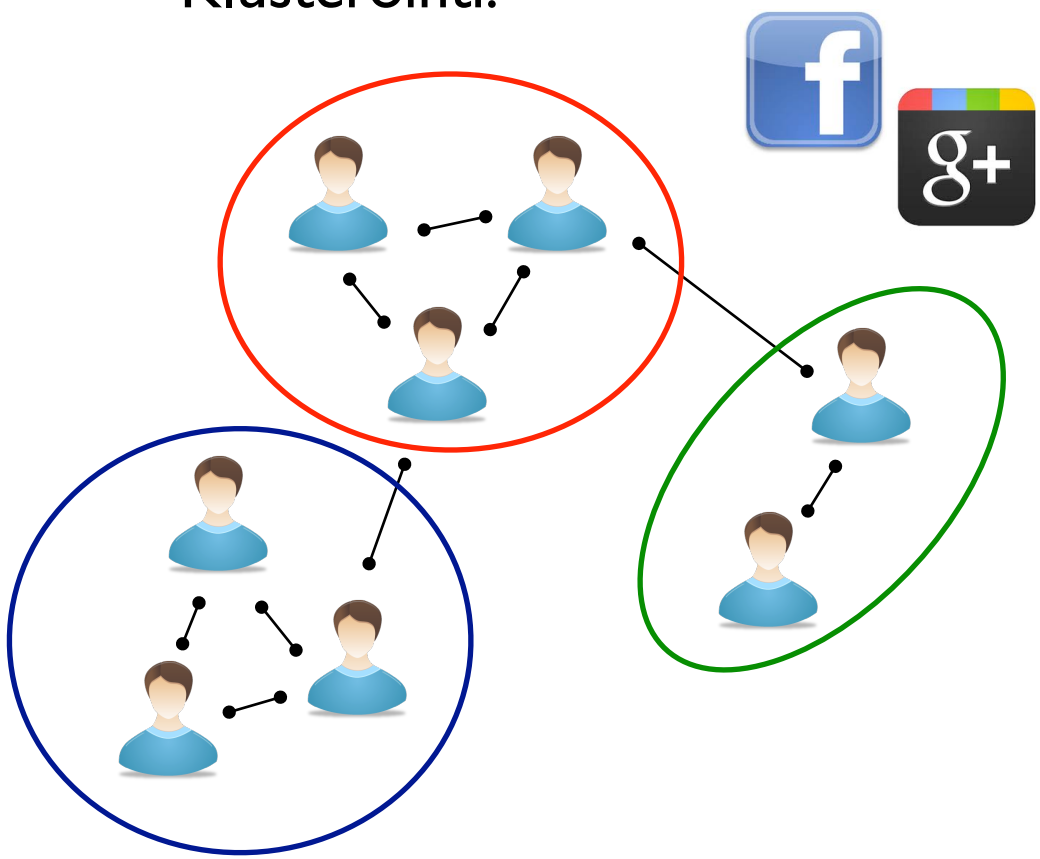


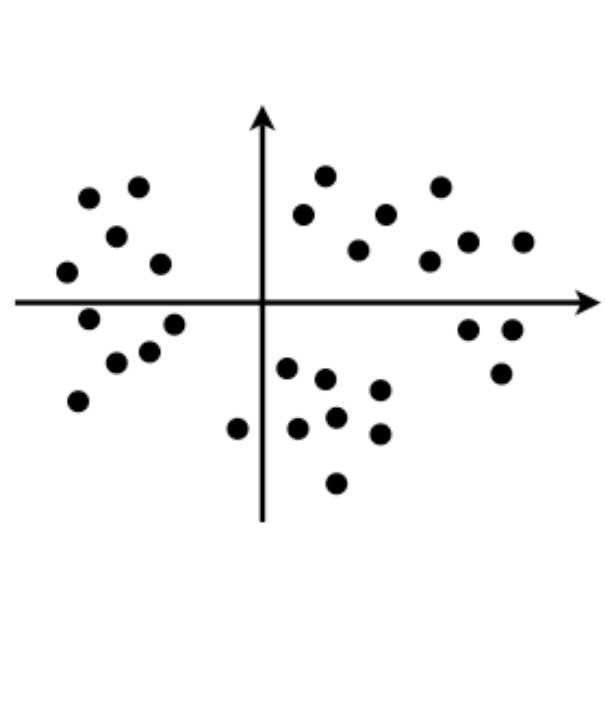


Koneoppimisen lajit

- Ohjaamaton oppiminen:
  - Esimerkit eivät sisällä ‘oikeaa vastausta’ y. Sen sijaan tavoitteena on vain ymmärtää annettu datajoukko.

Klusterointi:





Koneoppimisen lajit

- Vahvistusoppiminen:
  - Annettuna syöte ‘x’, ohjelma tulostaa ‘y’, mutta sen sijaan että meillä olisi tarkka ‘oikea’ y, osataankin vain antaa karkea arvio siitä, kuinka ‘hyvä’ ohjelman tulostama y oli (eli: ‘kannustetaan, muttei anneta mitään yhtä oikeaa vastausta’)
- Kaikki koneoppimistehtävät eivät ainakaan ihan suoraan mahdu näiden yksinkertaisten otsikoiden alle! (esim: ‘semi-supervised learning’, ‘yhteisöllistä suodattamista’, ‘learning to rank’, ...)

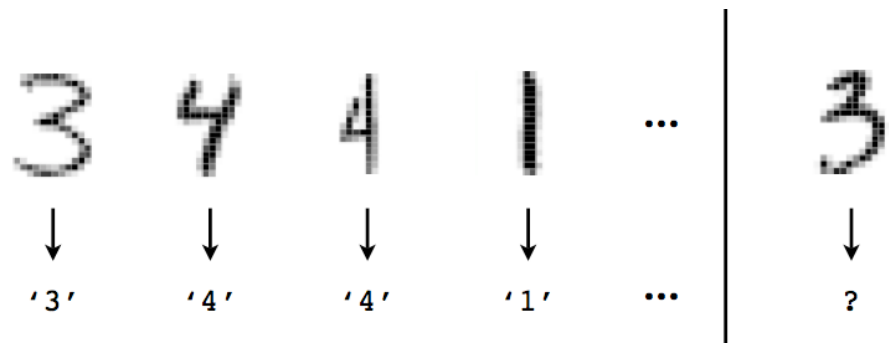
Käsinkirjoitetut merkit

Koneoppimisongelman määrittely (esim.):

- Tehtävä: Annettuna uusi 28 x 28 pikselikuva, luokittele se johonkin luokkaan 0 – 9. (Funktion syöte on siis 28*28 = 784-pituinen binäärivektori, ja funktion pitää palauttaa jokin kokonaisluku 0 – 9.)

- Hyvyysmitta: Kuinka monta prosenttia uusista kuvista luokitellaan oikein (eli siihen luokkaan mitä meidän esimerkeissä on annettuna)

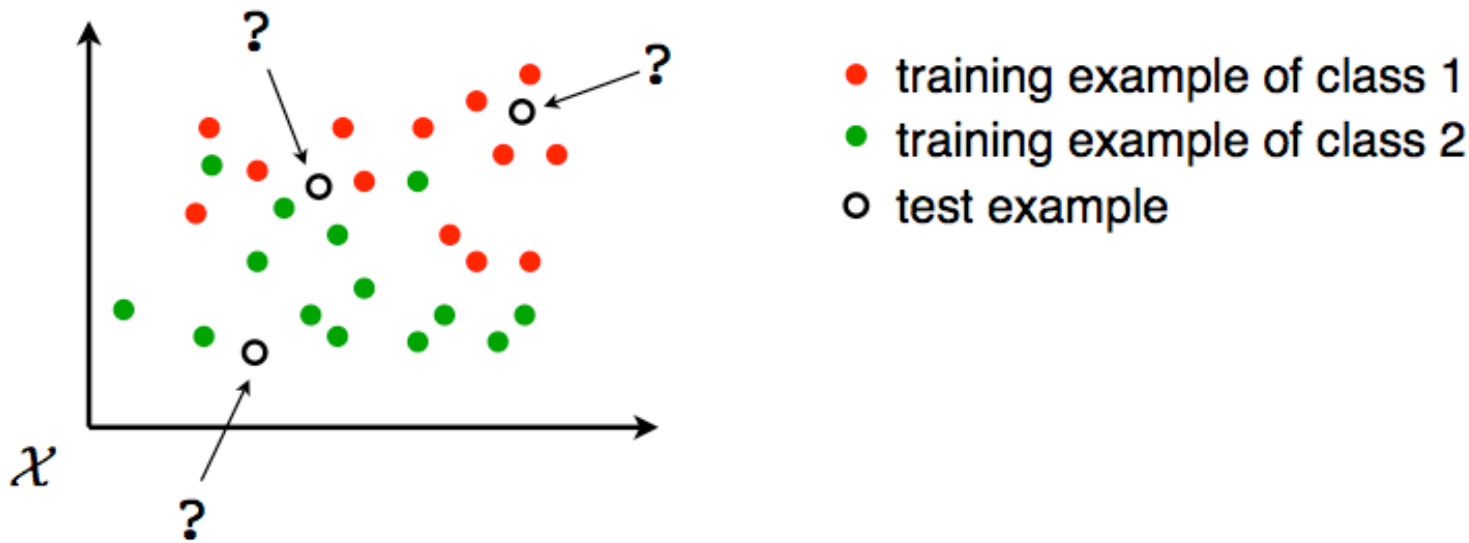
- Data: MNIST-käsinkirjoitetut numerot, jossa jokaiselle kuvalle on annettu luokka 0 – 9. Ensimmäiset 5000 kpl käytetään opetusdatana, seuraavat 1000 kpl testidatana.



Lähimmän naapurin luokittelija

'Yksinkertaisin mahdollinen' luokittelija.

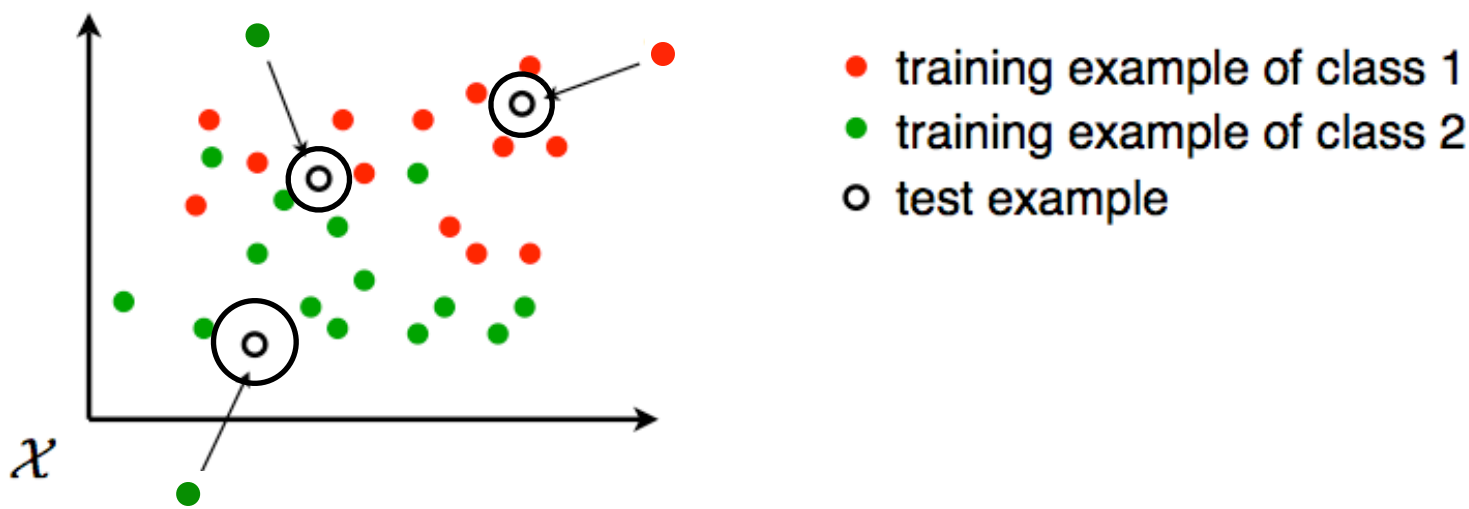
- Toiminta:
  1. Tallenna koko opetusdata
  2. Annettuna uusi (testi-) syötevektori x, löydä sitä lähimpänä oleva opetusdatan vektori x^{train} ja palauta sitä vastaava luokka y^{train}.
- Havainnollisesti siis näin:



Lähimmän naapurin luokittelija

'Yksinkertaisin mahdollinen' luokittelija.

- Toiminta:
  1. Tallenna koko opetusdata
  2. Annettuna uusi (testi-) syötevektori x, löydä sitä lähimpänä oleva opetusdatan vektori x^{train} ja palauta sitä vastaava luokka y^{train}.
- Havainnollisesti siis näin:



Lähimmän naapurin luokittelija

Merkkien väliset etäisyydet?

- Käytännössä kahden binäärivektorin etäisyyttä voidaan mitata esim laskemalla kuinka monessa kohtaa vektorit ovat erisuuret:

1	0	0	0	1	1	0	0	0	1	0	1	0
1	1	0	0	0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	0	1	1	0	0	0

$\Rightarrow$ 4 eroa

...tämä vastaa sitä että laitetaan 'merkit päällekkäin', ja katsotaan kuinka monessa kohtaa ne eroavat.

- Onko tämä hyvä etäisyysmitta? (esim kaksi melkein identtistä merkkiä jotka eroavat vain niin että toisessa kuvassa merkki on siirtynyt parin pikselin verran saattavat tämän etäisyysmitan suhteen olla hyvin kaukana toisistaan)

Lähimmän naapurin luokittelija

- Toinen esitys samasta asiasta:

Opetusdata (luokat annettu):

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

Uudet kuvat:

1

Lähimmän naapurin luokittelija

- Toinen esitys samasta asiasta:

Opetusdata (luokat annettu):

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

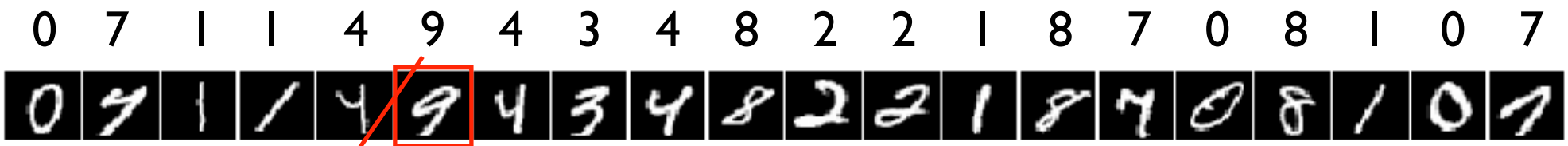
Uudet kuvat:

1 1

Lähimmän naapurin luokittelija

- Toinen esitys samasta asiasta:

Opetusdata (luokat annettu):



Uudet kuvat:

1 1 9

Lähimmän naapurin luokittelija

- Toinen esitys samasta asiasta:

Opetusdata (luokat annettu):

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

Uudet kuvat:

1 1 9 9

Lähimmän naapurin luokittelija

- 1-NN (1-nearest neighbor) luokitin käsinkirjoitetuille merkeille:

Opetusdata (luokat annettu):

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

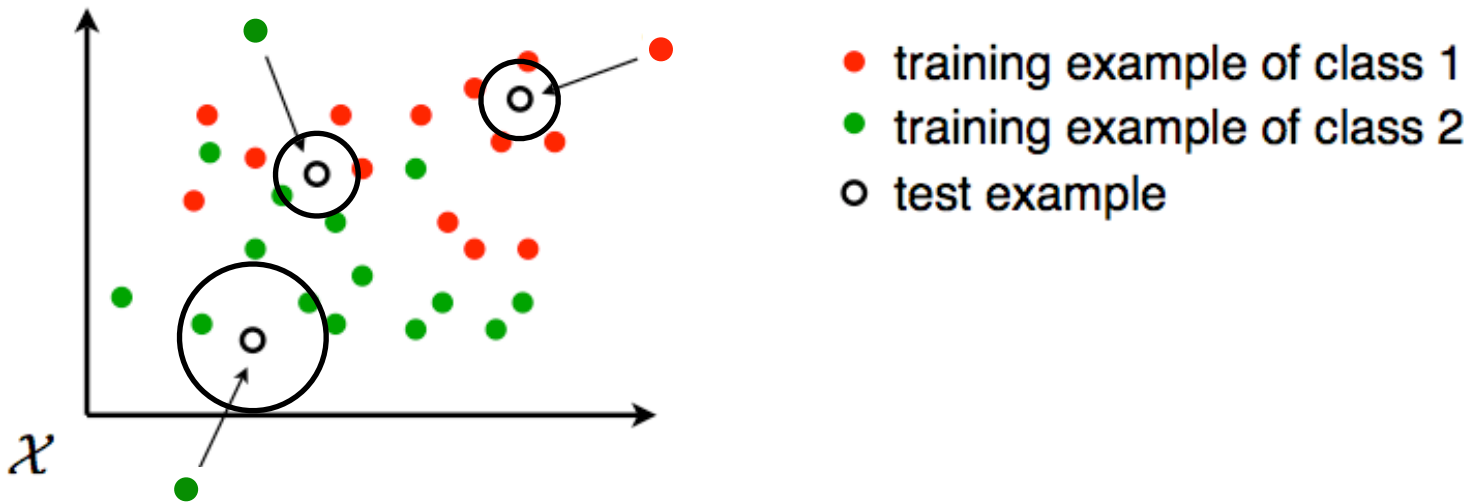
Uudet kuvat:

1 1 9 9 ...

Virheitä: 7.6%
(ensimmäiset 5000 merkkiä opetusdataa, seuraavat 1000 merkkiä testidataa)

k-lähimmän naapurin luokittelija

- Toiminta: Samoin kuin lähimmän naapurin luokittelija, mutta löydä k lähintä opetusdatan esimerkkiä, ja arvaa luokka tämän perusteella (k lähintä 'äänestävät' mitä luokkaa ehdotetaan)
- Esim k=3:



- Harjoitustehtävänä kokeilla, miten tällainen 'k-NN' luokitin toimii käsinkirjoitettujen numeroiden tunnistamisessa

‘Naive Bayes’ -luokittelija

Käsitelty jo aiemmin kurssilla roskapostisuodatuksen yhteydessä

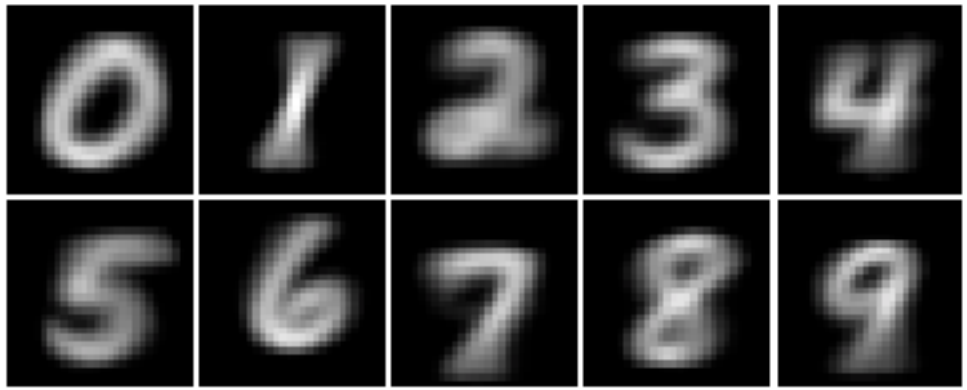
- Toiminta:

```
Oppiminen: (annettuna iso esimerkkijoukko kuvia+luokkia)
Jokaiselle luokalle i (tässä: 0-9)
  Jokaiselle piirteelle j (tässä: 784 pikseliä)
    Estimoi todennäköisyysjakauma tälle luokka-piirre
    kombinaatiolle (tässä: todennäköisyys että kyseinen
    pikseli, tämän luokan kuville, on 1)

Luokittelu: (annettuna uusi kuva)
Jokaiselle luokalle i (tässä: 0-9)
  Aseta log-todennäköisyys logP(i)=0
  Jokaiselle piirteelle j (tässä: 784 pikseliä)
    logP(i) = logP(i) + log P(i,j) (eli summaa
    pikselien log-todennäköisyydet, annettuna luokka)
Palauta luokka i jolla suurin log-todennäköisyys.
```

‘Naive Bayes’ -luokittelija

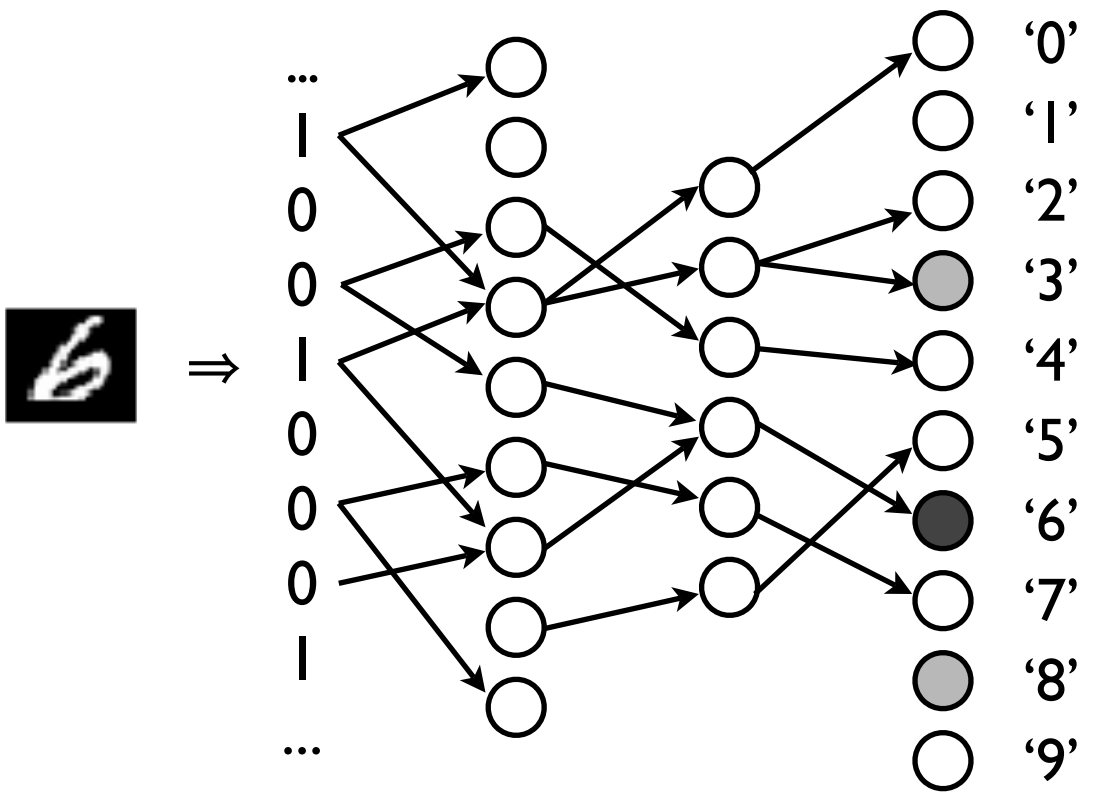
- Jokaisen luokan, jokaisen pikselin todennäköisyyet voidaan esitellä yksinkertaisesti näin (harmaasävy = todennäköisyys että pikseli on ‘päällä’ kyseisen luokan kuvilla)



Eli tässä saadaan eräänlaiset ‘prototyypit’ luokille. Luokka arvioidaan sen perusteella, kuinka hyvin uusi kuva sopii näihin prototyyppeihin.

Neuroverkko

- Matkitaan aivojen toimintaa. Monta ‘hermosolua’, kukin pystyy vain hyvin yksinkertaiseen operaatioon, mutta yhdessä ne tekevät jotain mielenkiintoista:



Jokainen ‘solu’ laskee painotetun summan tuloistaan, ja lähettää eteenpäin epälineaarisen funktion siitä summasta

Painot ovat adaptiivisia, ja säädetään niin että verkko ‘oppii’ datasta

Yhteenveto

- Koneoppimista kannattaa käyttää kun
  - Tehtävä on vaikea ratkaista ‘manuaalisella’ ohjelmoinnilla: Ohjelmoija ei osaa itse ratkaista tehtävää, tai ehkä osaa muttei kuitenkaan pysty kertomaan ‘miten’ se tehdään.
  - Esimerkkejä (dataa) on paljon, jonka perusteella kone voi oppia itse suorittamaan jotain tehtävää
  - Tarvitaan adaptiivinen menetelmä, joka mukautuu käyttäjän tottumuksiin ja tarpeisiin
- Aloita määrittelemällä tehtävä, hyvyysmitta, ja data. Tämän jälkeen valitse tehtävään sopiva menetelmä.

Mistä lisää tietoa?

- Koneoppiminen, ongelman määrittely, ohjattu/ohjaamaton oppiminen:

  Esim: Stanfordin kurssin intro videot (40 min)
  http://www.ml-class.org/course/video/preview_list

- Lähimmän naapurin luokittelija, k-NN. Esim:
  http://cseweb.ucsd.edu/~elkan/250B/nearestn.pdf

- Naive Bayes -luokitin. Esim:
  http://en.wikipedia.org/wiki/Naive_Bayes_classifier

Kurssit

- ‘Introduction to Machine Learning’ (engl.)
  - Periodi II (1.11 – 9.12.2011)
  - ‘Algoritmit ja koneoppiminen’-linjan pakollinen kurssi
- Jatkokurssit: ‘Supervised machine learning’, ‘Probabilistic models’, ‘Unsupervised machine learning’, ‘Data mining’, ...
- Myös Stanfordin yliopiston kurssi: www.ml-class.org
  - Videoita, tehtäviä, tentti, ohjelmointitehtäviä
  - Kurssi alkaa 10.10 (siis pian!)
  - Opintopisteitä tarjolla, katso: http://www.cs.helsinki.fi/en/uutiset/65779