MTAT.03.311
Loomuliku keele töötlus *Python*is
(6 EAP)

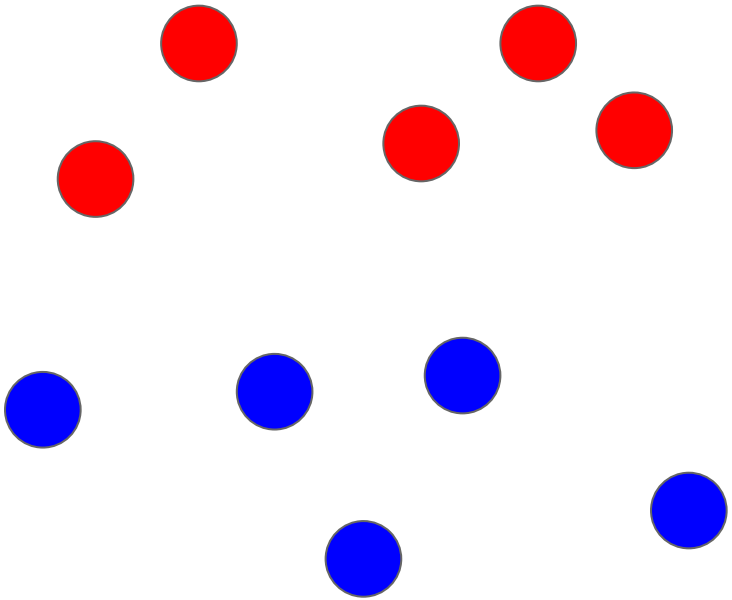
Karl-Oskar Masing

13.10.2015

Sissejuhatus masinõppesse

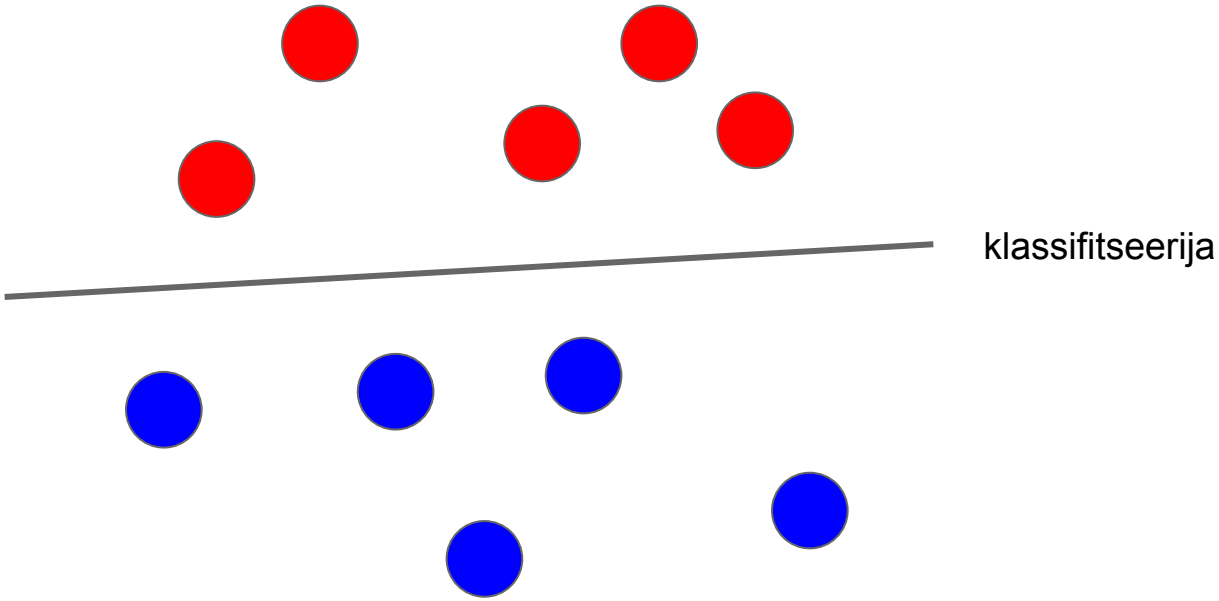
Pürg paremate mudelite poole

Klassifitseerimine



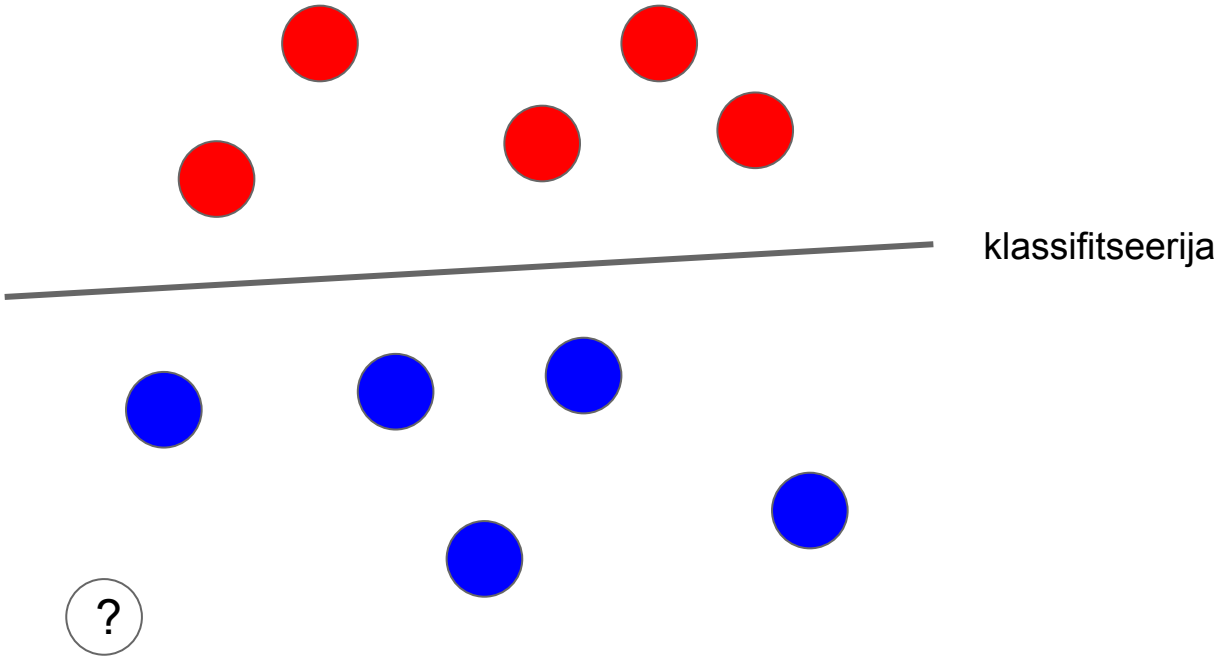
Klassidega märgendatud andmed.

Klassifitseerimine



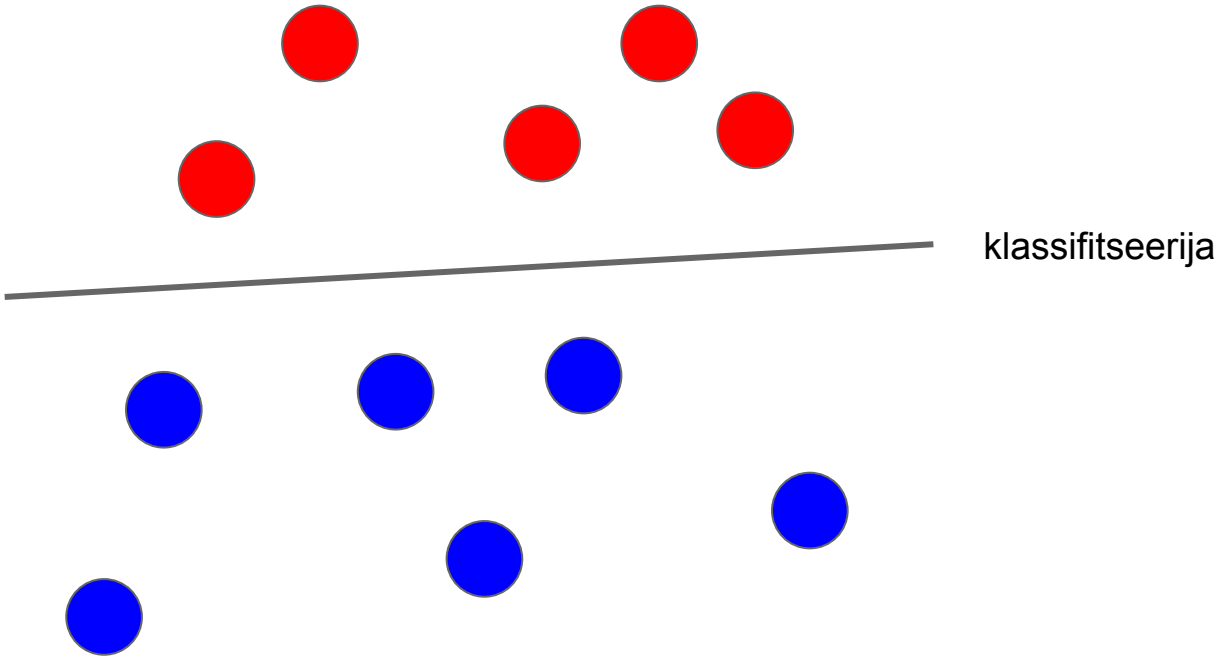
Leiame mudeli, mis suudaks võimalikult täpselt klasse eraldada.

Klassifitseerimine



Uute märgendamata andmete korral...

Klassifitseerimine



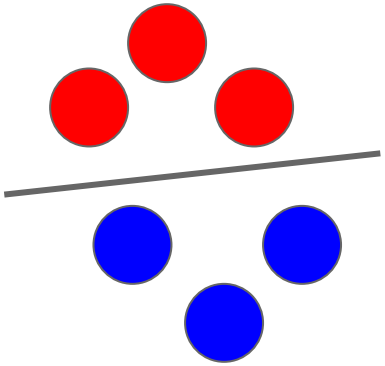
… palume klassifitseerijal öelda, millistesse klassidesse kuuluvad.

Tugivektormasin (SVM) ?

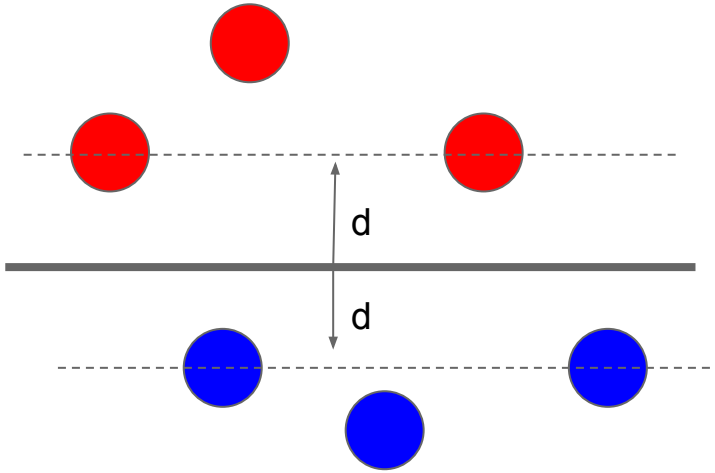
Tajur (*perceptron*)

- lihtne
- geomeetriline
- binaarne
- kiire
- online
- leiab hüpertasandi, mis eraldab kahte klassi
  - kui ei leia, põrub
- *sklearn.linear_model.Perceptron*

Tajur vs SVM

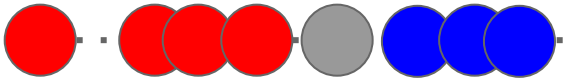


Tajur rahuldub mingi klasse ideaalselt kirjeldava klassifitseerijaga.

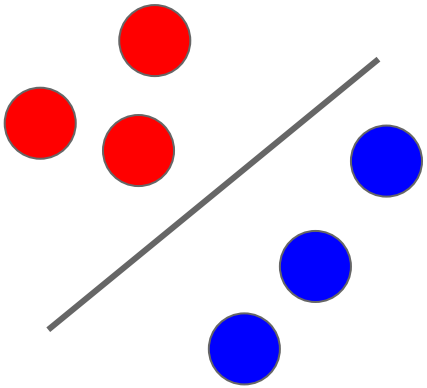


SVM eeldab klassifitseerijalt “optimaalset stabiilsust” ehk maksimaalset *d* väärtust.

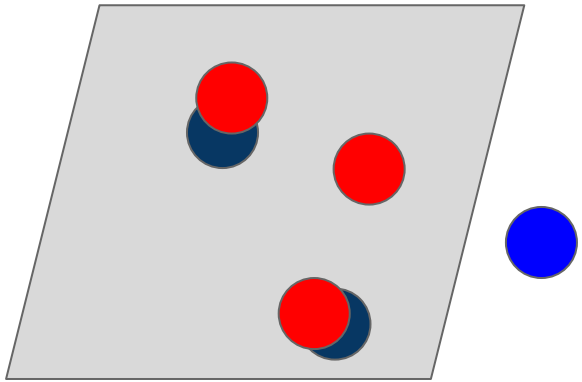
Geomeetriline binaarne klassifitseerija



1D - klassifitseerija on punkt



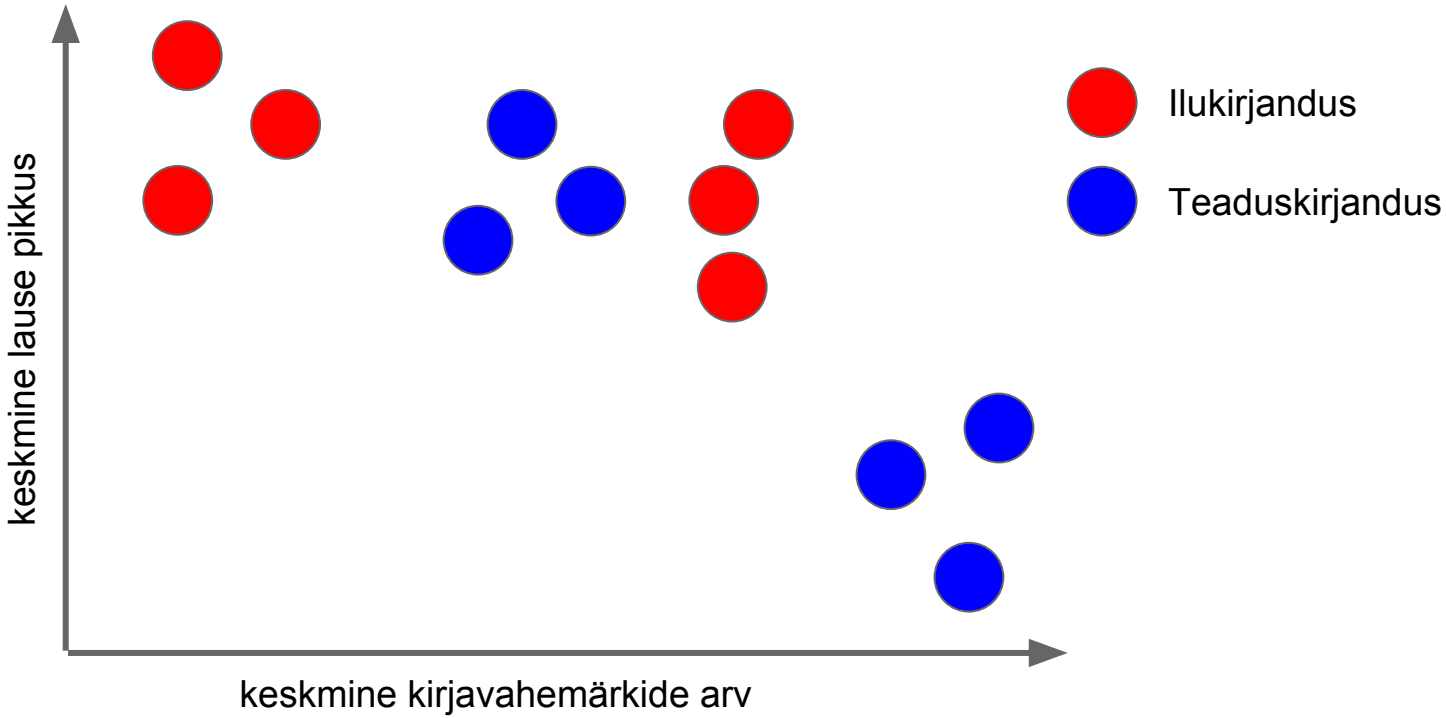
2D - klassifitseerija on “joon”



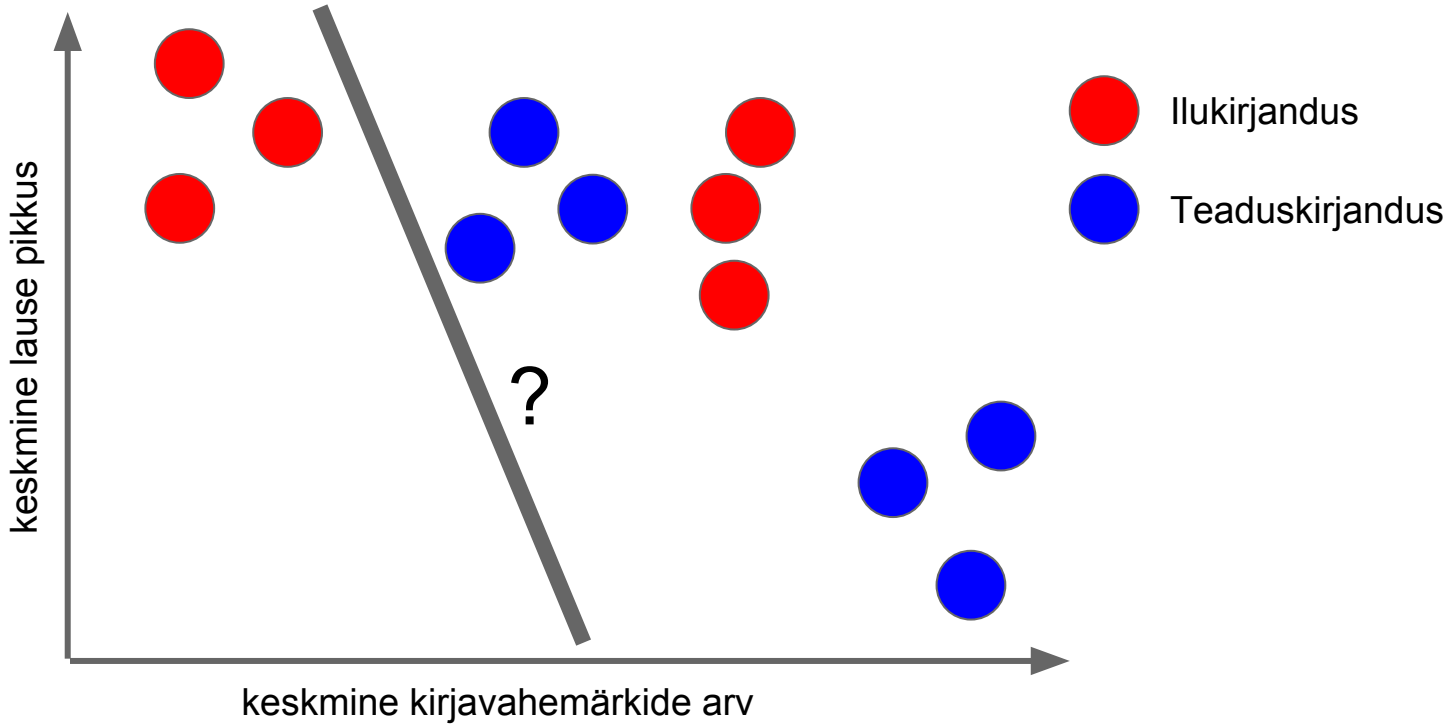
3D - klassifitseerija on tasand

Kas kõik probleemid on lahendatavad?

Mittelineaarne probleem



Mittelineaarne probleem

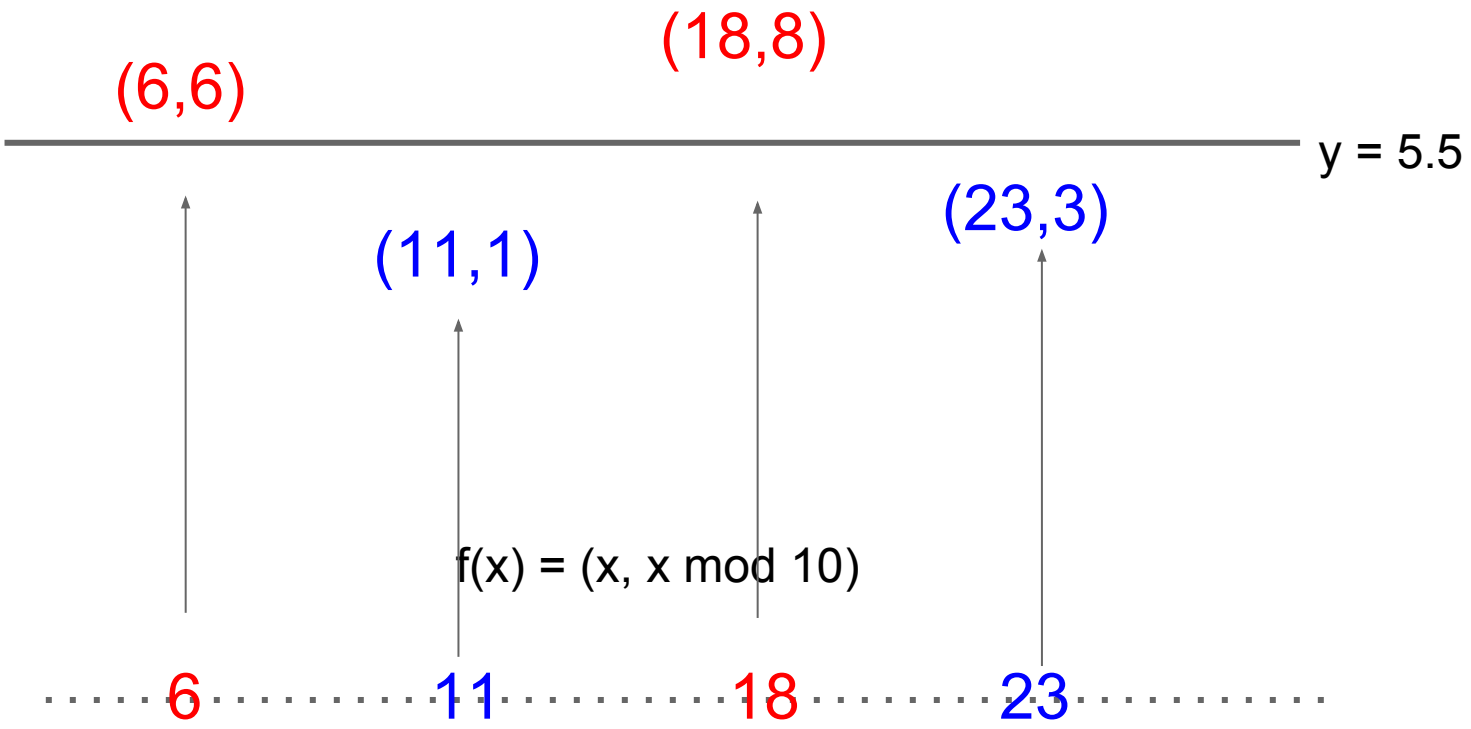


Lineaarsed mudelid ja mittelineaarsus

- teisendame sisendandmed kõrgemasse dimensiooni
  - loodame, et seal leidub eraldav hüpertasand

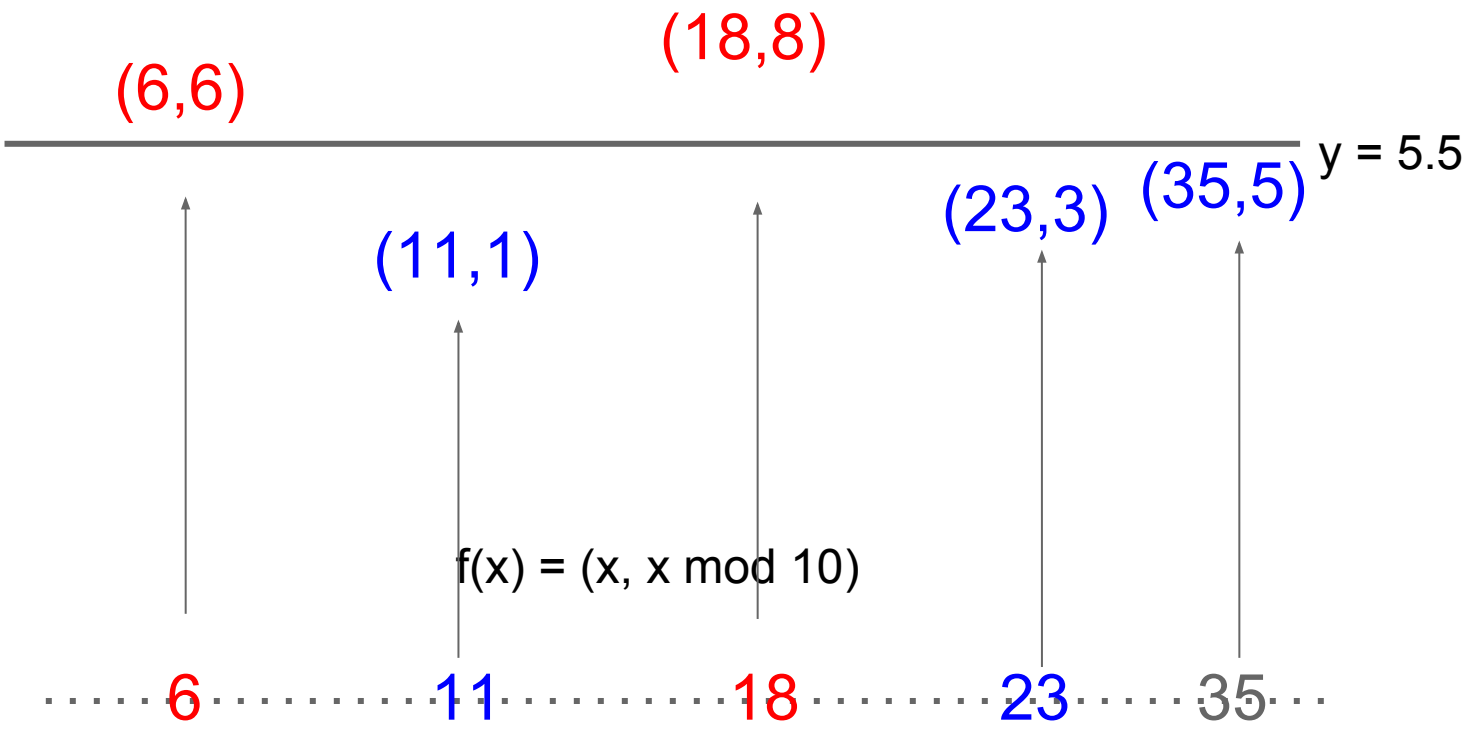
Probleem, millel 1D-ruumis ei leidu lineaarset klassifitseerijat.

Lineaarsed mudelid ja mittelineaarsus



Teisendame probleemi 2D-ruumi, nii et leiduks klassifitseerija.

Lineaarsed mudelid ja mittelineaarsus



Uue sisendi klassi ennustades teisendame sisendi vastavasse ruumi ja leiame klassi.

SVM ja mittelineaarsus

- kasutab *kernel trick*'i
  - teisendab sisendi kaudselt kõrgemasse dimensiooni
    - kasutab selleks sarnasusfunktsiooni *k* (*kernel*)
      - nt

$$\hat{y} = \operatorname{sgn} \sum_{i=1}^{n} w_i y_i k(\mathbf{x}_i, \mathbf{x}'),$$

https://en.wikipedia.org/wiki/Kernel_method

  - erinevad *kernel*'id annavad erinevatel andmetel erinevaid tulemusi

SVM

```
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
>>> y = np.array([1, 1, 2, 2])
>>> from sklearn.svm import SVC
>>> clf = SVC()
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3,
    gamma=0.0, kernel='rbf', max_iter=-1, probability=False,
    random_state=None, shrinking=True, tol=0.001, verbose=False)
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

- Võimalikud kernelid
  - ‘linear’, ‘poly’, ’sigmoid’, callable
  - vaikimisi ‘rbf’

Parameetrid

Parameters

Parameetrid

- masinõppe meetodid võtavad hulga “argumente”
  - defineerivad, kuidas õpe toimub
  - kasutatakse õppimisalgoritmis
  - sageli määravad kvaliteedi
- hea mudeli leidmiseks peame testima erinevaid parameetrite kombinatsioone

Meeldetuletus - treening- ja testandmestik

- eesmärk
  - hinnata mudeli kvaliteeti
- ei saa hinnata treenitaval andmestikul
  - iga mudeli saab treenida treeningandmestikul 100%-lise täpsusega
    - jätame meelde X-Y kujutuse
      - nt Lagrange'i polünoomid
- hoiame osa andmestikku treenimisest lahus
  - sama jaotusega

Arendamis/valideerimisandmestik

- eesmärk
  - hinnata õpinguparameetreid
- hoiame treening- ja testandmestikust lahus
- sarnane testandmestikule, aga mudeli hindamise asemel hindame mudeli parameetreid
- väikseim osa andmetest
- SAMA JAOTUSEGA

Uus andmete jagamise skeem

- 70-30 korral (näiteks)
  - 70% juhuslikest andmetest treeningandmed
  - 25% juhuslikest andmetest testandmed
  - 5% juhuslikest andmetest valideerimisandmed
- juhuslikkuse saame tagada andmete segamisega
  - random.shuffle(X)

Parameetrite valideerimisega masinõpe

- iga mudeli korral
  - iga huvipakkuva parameetrite kombinatsiooni korral
    - treenime parameetritegea mudeli treeningandmestikul
    - leiame mudeli kvaliteedi valideerimisandmestikul
    - talletame seni parima parameetrite kombinatsiooniga mudeli
  - hindame mudeli kvaliteeti parimate parameetritega testandmestikul
- valime parima mudeli vastavate parameetritega

Kas nüüd on kõik raskused teelt pühitud?

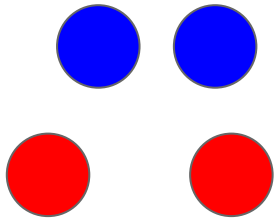
Klassifitseerimise kvaliteedi mõjutajad

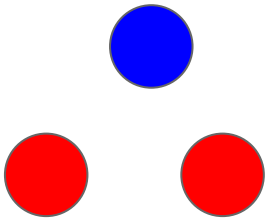
- Mudel
- Andmed

Andmete pädevus

- andmete hulk
  - lihtsam mudel ja rohkem andmeid >>> fääntsim mudel ja vähem andmeid
- andmete jaotus
  - treeningandmed kirjeldama reaalsust!
- mõõtmiste kvaliteet
  - vigaste andmetega saab vigase mudeli
- tunnuste relevantsus sõltuva muutuja suhtes
  - teose žanrit ei saa ennustada ainult autori abil

Andmete hulk ja jaotus





Väikese treeningandmestiku korral võime leida lihtsa kirjeldava mudeli.

Andmete hulk ja jaotus

Leidsime lihtsa kirjeldava mudeli ja oleme õnnelikud.

Andmete hulk ja jaotus

Uute andmete korral, mille sarnaseid treenides ei näinud, töötab mudel väga halvasti.

Alaõppimine

Underfitting

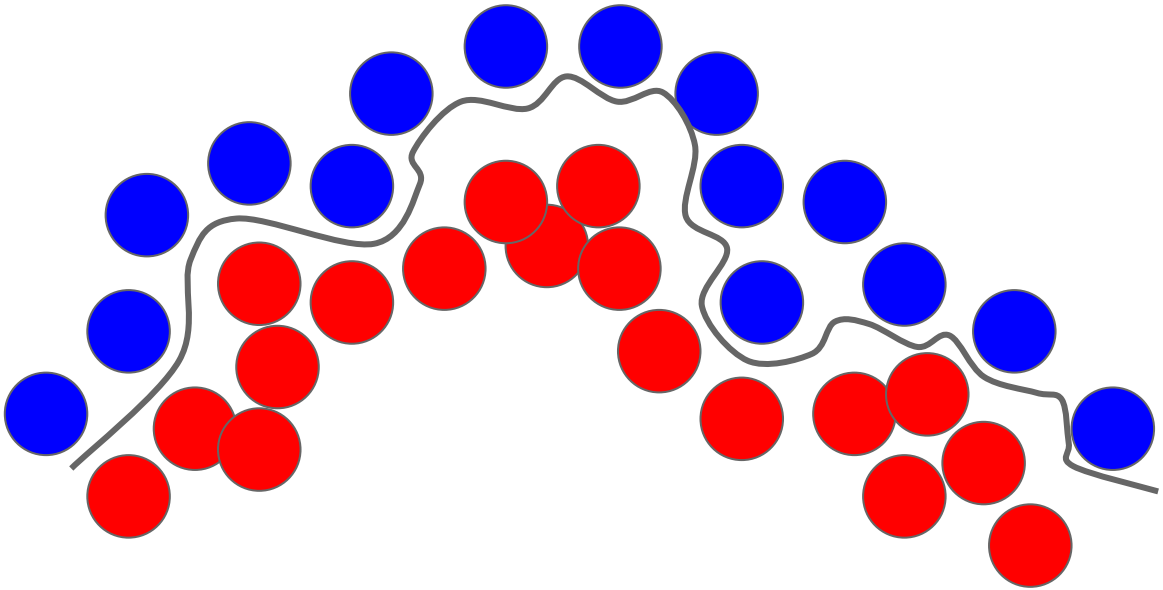
Alaõppimine

- kaotame mudeli kvaliteedis, kuna ei suuda süsteemi kirjeldada
- põhjused
  - liiga vähe andmeid
  - liiga lihtne mudel
- sümptom
  - vähe andmeid
  - andmed pole süsteemist juhuslikult mõõdetud
  - madal skoor testandmestikul

Alaõppimine

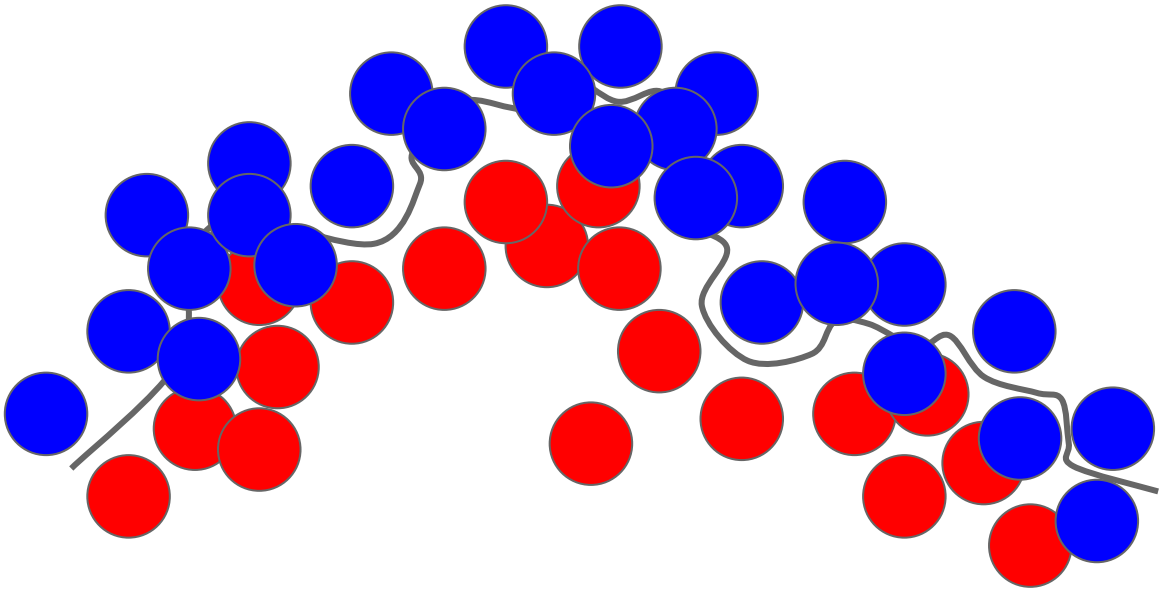
- lahendus
  - rohkem andmeid
  - keerulisem mudel
    - nt asendame lineaarse kerneli polünomiaalsega

Fääntsi mudel



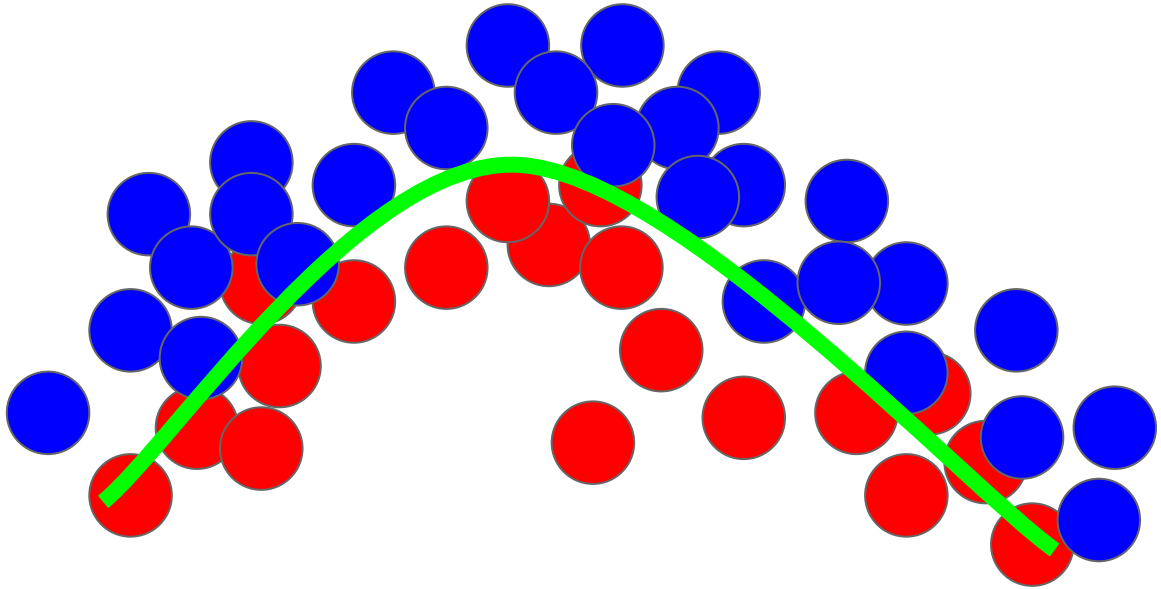
Meil on omajagu mõõtmisi ning selle baasil joonistame keerulise polünomiaalse mudeli, mis eraldab klassid perfektselt.

Fääntsi mudel



Uutele andmetele klasse ennustades läheb mudel katki.

Fääntsi mudel - reaalsus



Reaalselt on andmed lihtsamast jaotusest.

Andmed valetavad?

Müra

Noise

Müra

- seletamatu variatsioon andmetes
- põhjused
  - vigane mõõtmine
  - eksimus sõltuva muutuja märgendamisel
    - nt inimene määrab raamatule vale žanri
  - liiga primitiivsed tunnused
    - üritame sõltumatute tunnuste abil kirjeldada tunnust, mille
      - olemasolu me ei tea
      - ei saa kirjeldada teiste sõltumatute tunnuste alusel
      - mõjutab ennustatavat märgendust

Üleõppimine

Overfitting

Üleõppimine

- süsteemi modelleerides modelleerime liigselt müra
- põhjused
  - liiga keeruline mudel
  - vigane mõõtmistehnika andmeid kogudes
  - puuduvad tunnused
- sümptomid
  - kõrge skoor treeningandmestikul
  - madal skoor testandmestikul

Üleõppimine

- lahendus
  - lihtsam mudel
  - leida olulised puuduvad tunnused
    - uuesti mõõta

Üleõppimise vältimine otsustuspuudel

- puude pügamine
  - puud lõigatakse läbi teatud sügavusel
- üldistamine juhusliku metsa läbi
  - *random forest*
  - enamushääletus kümnete-sadade-tuhandete puude hulgas
  - igal puul juhuslikult valitud tunnuste alamhulk
    - puud erinevad
  - olulised tunnused saavad rohkem võimu
    - võitlus müra vastu

Tunnused

Features

Tunnused

- kasutada tuvastamisel domeenieksperdi abi
- pigem vähem ja olulisemaid
- peavad olema mõõdetavad ja objektiivsed

Tunnuste eraldamine

Feature extraction

Tunnuste eraldamine

```
>>> measurements = [
...     {'city': 'Dubai', 'temperature': 33.},
...     {'city': 'London', 'temperature': 12.},
...     {'city': 'San Fransisco', 'temperature': 18.},
... ]

>>> from sklearn.feature_extraction import DictVectorizer
>>> vec = DictVectorizer()

>>> vec.fit_transform(measurements).toarray()
array([[  1.,   0.,   0.,  33.],
       [  0.,   1.,   0.,  12.],
       [  0.,   0.,   1.,  18.]])

>>> vec.get_feature_names()
['city=Dubai', 'city=London', 'city=San Fransisco', 'temperature']
```

http://scikit-learn.org/stable/modules/feature_extraction.html

Tunnuste valimine

Feature selection

Tunnuste valimine

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.feature_selection import SelectKBest
>>> from sklearn.feature_selection import chi2
>>> iris = load_iris()
>>> X, y = iris.data, iris.target
>>> X.shape
(150, 4)
>>> X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
>>> X_new.shape
(150, 2)
```

http://scikit-learn.org/stable/modules/feature_selection.html

Küsimusi? :)