Selected topics in Data Science

Grandjean, Martin (2014).
"La connaissance est un réseau".
Les Cahiers du Numérique

Wojciech Krzemień

20.11 2020, NCBJ

Last lecture(s) recap

- Elements of Statistical Learning Theory:
  - Optimal algorithm and optimal classifier
  - Bias-variance decomposition
  - Over-fitting vs model complexity
- Crash course in statistics II:
  - quantiles, expectation and variance
  - joint and marginal probabilities
  - conditional probability
  - probability density vs likelihood function
  - covariance
  - independence
  - conditional variance, conditional expectation

Todays plan

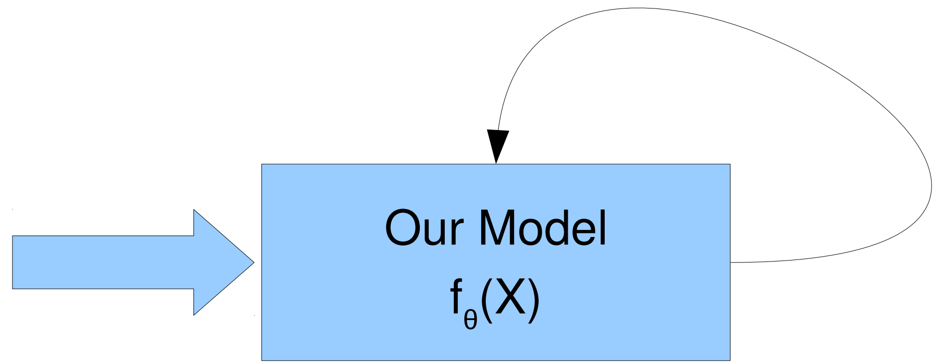
- What we did so far?
- Curse of dimensionality
- Linear regression
- Work in groups:
  - Statistics problems
  - Small programs to write
  - k-NN implementations

Supervised learning

1.Training

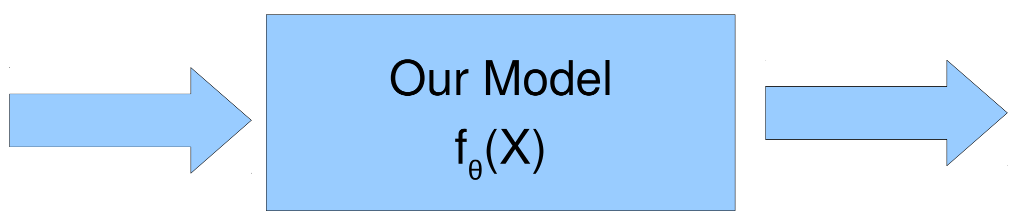
$\{X^1,Y^1\}, \{X^2,Y^2\}, \ldots$

training data



2.Prediction:

X



$f(X^k)$

X – input, feature vector
Y – output,
Training set T – pairs of $\{X^i, Y^i\}$ i=1,...N
$f_\theta(X)$ – our model (hypothesis)
θ – some parameters of the model

Supervised learning - spam filter

I am Mohammed Abacha,the son of the late Nigerian Head of State who died on the 8th of June 1998.If you are conversant with world news,you would understand better,while I got your contacts through my personal research.Please,I need your assistance to make this happen and please; do not undermine it because it will also be a source of upliftment to you also.You have absolutely nothing to loose in assisting us instead, you have so much to gain.

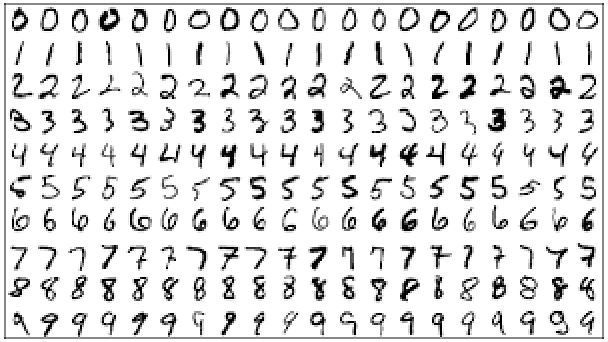
Please my dear,I repose great confidence in you and I hope you will not betray my confidence in you.I have secretly deposited the sum of **$30,000,000.00** with a security firm abroad whose name is withheld for now until we open communications.The money is contained in a metal box consignment with Security Deposit Number 009GM.

{X_1 , “non-spam”}, {X_2,”spam” } $\Rightarrow$

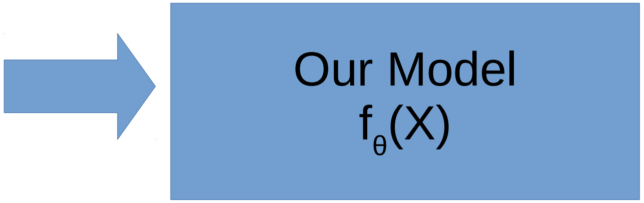
Our Model
$f_\theta(X)$

X – set of words indicative for spam
Y – „spam”/”non-spam”

Supervised learning
handwriting recognition



{  ,5}, {  ,2}, ...



X – image of a digit
Y – numerical value

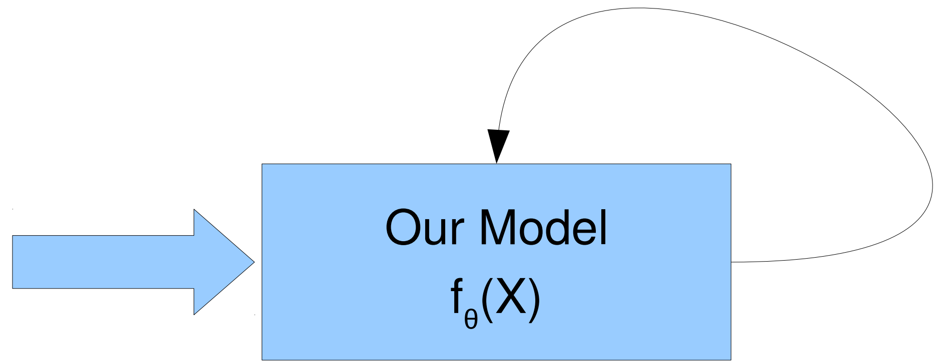
Yann LeCun – CNN pioneer work

Supervised learning

1.Training

$\{X^1,Y^1\}, \{X^2,Y^2\}, \ldots$

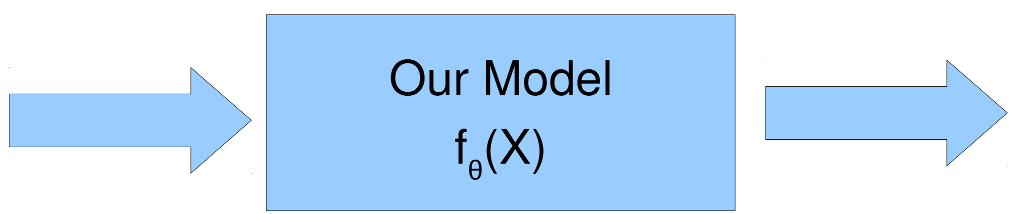
training data



Loss function to penalize an inaccuracy of prediction
$L(f_\theta(X),Y^{(i)})$

2.Prediction:

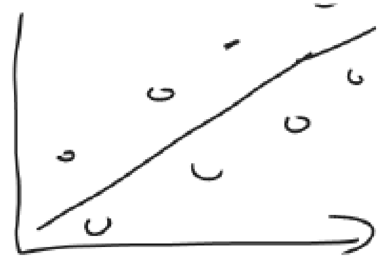
X



f(X)

X – input, feature vector
Y – output,
Training set T – pairs of $\{X^i, Y^i\}$ $i=1,\ldots N$
$f_\theta(X)$ – our model (hypothesis)
θ – some parameters of the model

Training expressed as **minimization** problem



$f_\Theta = \Theta_0 + \Theta_1 x_1$

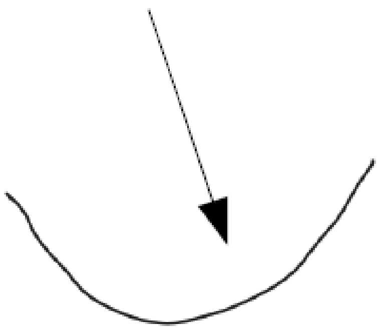
Find parameters θ^* for which E_T has minimum

$E_T(f(x), y)$

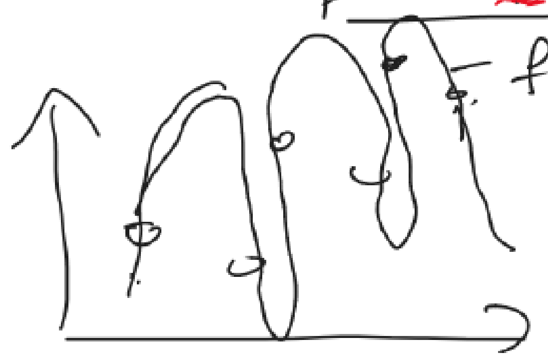
f_{Θ^*}

Test errors
calculated for points
not in training set

Convex function



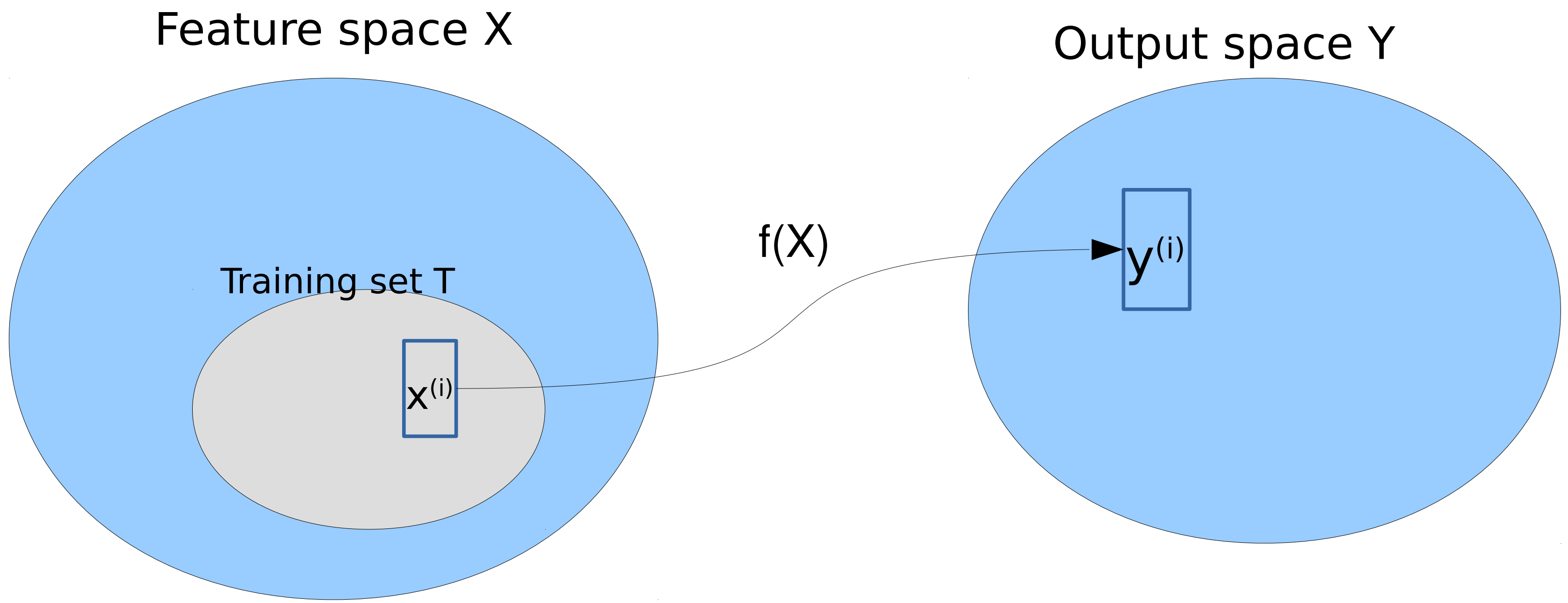
No more than one minimum



$E_T = 0$

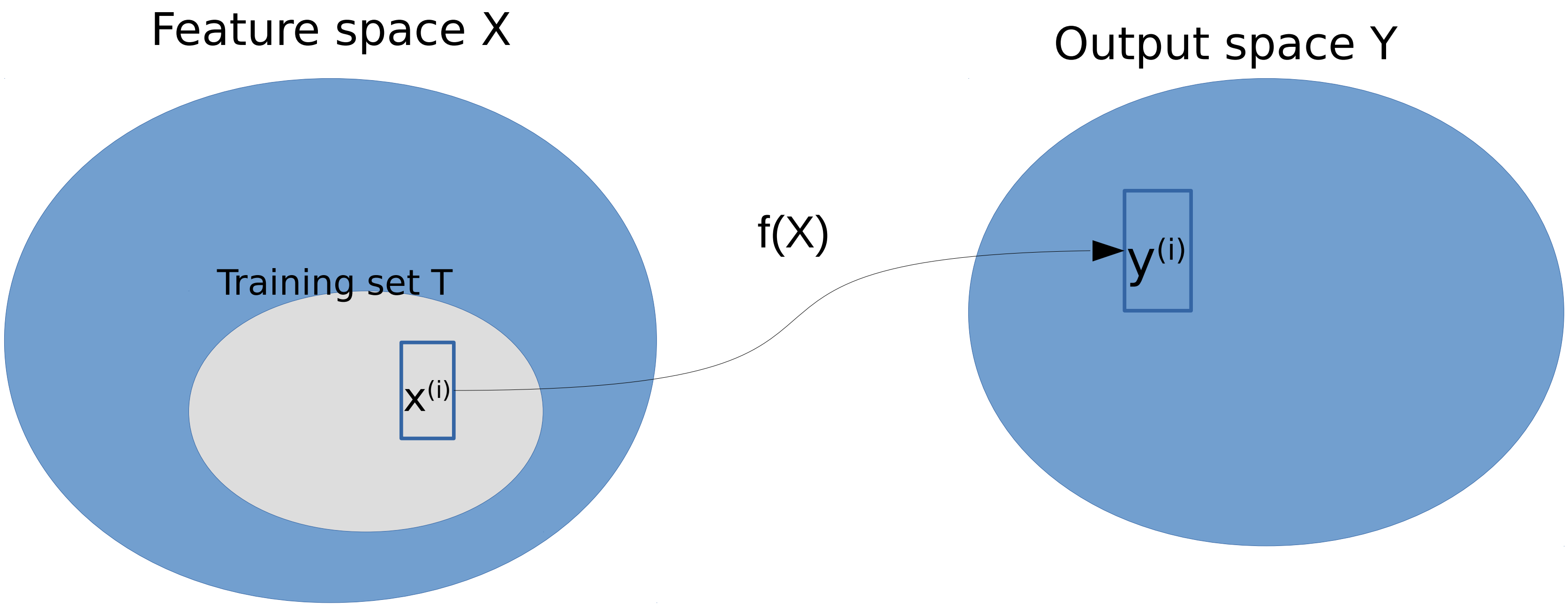
OVERFITTING

Generalization
Model should describe all the data



X,Y are random variables described by the (unknown) joint distribution $\mathbf{p_{XY}(x,y)}$

- Training set T consists of pairs of $\{x^{(i)}, y^{(i)}\}$ i=1,...N
- $\{x^{(i)}, y^{(i)}\}$ pairs are i.i.d.$^{(*)}$
- Loss function L (f(X),Y) to penalize inaccuracy of prediction



X,Y are random variables described by the (unknown) joint distribution $\mathbf{p_{XY}(x,y)}$

- Training set T consists of pairs of $\{x^{(i)}, y^{(i)}\}$ i=1,...N
- $\{x^{(i)}, y^{(i)}\}$ pairs are i.i.d.$^{(*)}$
- Loss function L (f(X),Y) to penalize inaccuracy of prediction

Goal:

Based on T estimate mapping $\bar{f}(X)$ between X→Y for "effective" prediction

$^{(*)}$ iid – independent (mutually) and identically distributed samples

Joint probability density function

$$E_{PE}[f] = \iint L(f(x), y)\, \rho_{\vec{x}, \vec{y}}\, dx\, dy$$

Minimizing $EPE[f]$

Indicator loss for classification

L_2 $L = (f(x) - y)^2$

L_1: $L = |f(x) - y|$

$L = [f(x) \neq y]$

$\boxed{f^* = E_{y|x}[y | x = x_0]}$

REGRESSION FUNCTION

$\boxed{f^* = \mathrm{Median}(y | x_0)}$

$\boxed{f^* = \arg\max_k P(k | x = x_0)}$

BAYES CLASSIFIER

Bias-variance decomposition

$$VAR(\varepsilon) + BIAS^2(\hat{f}) + VAR(\hat{f})$$

Irreducible error

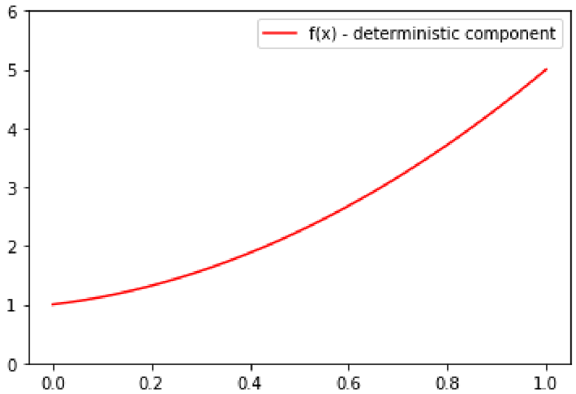
$$(\hat{f} - f^*)^2$$

How far we are from the optimal solution

How much or model would change for different training sets

Example

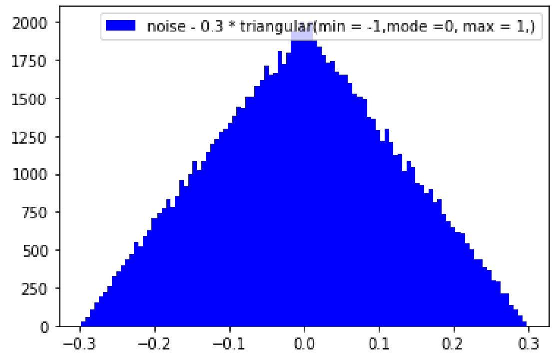
deterministic component



$$f^* = 1 + x + 3\,x^2$$

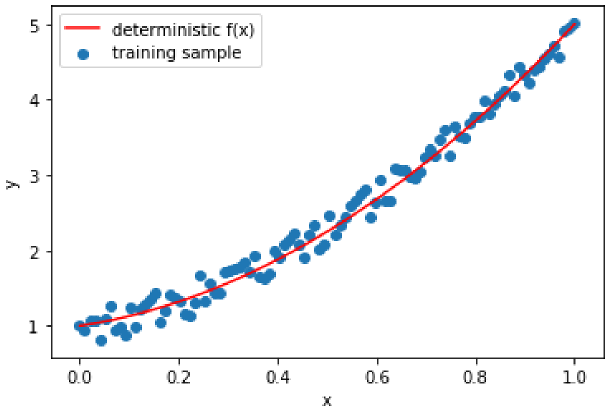
+

noise



triangular distribution

=



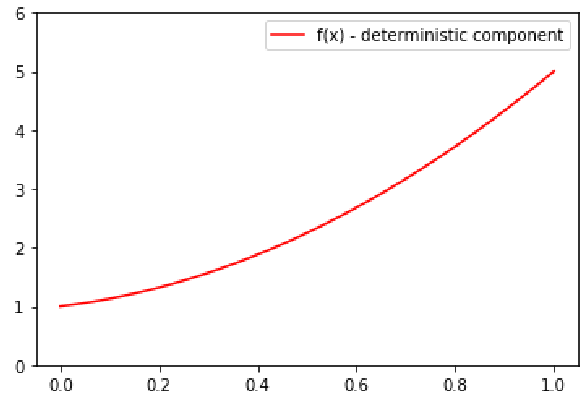
$$1 + x + 3\,x^2 + \mathbf{\color{red}{0.3\ e}}$$

Irreducible error

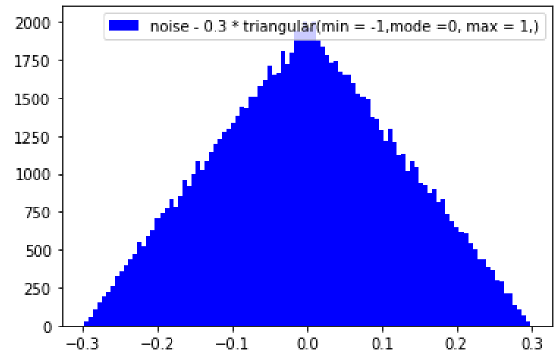
Example

deterministic component

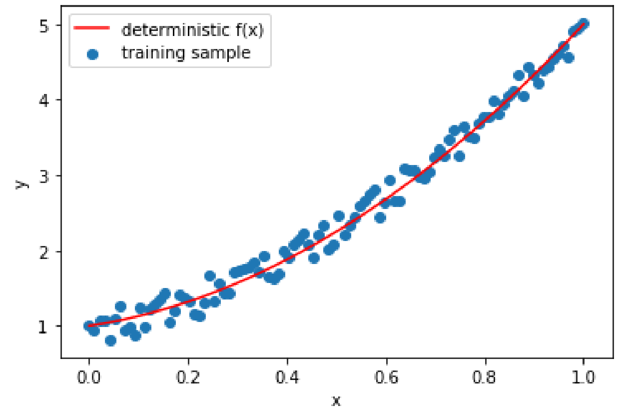
noise



+



=



$f^* = 1 + x + 3\,x^2$

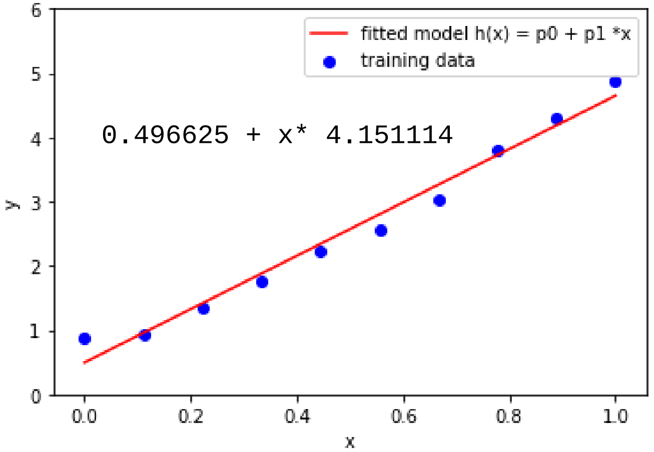
triangular distribution

$1 + x + 3\,x^2 + \mathbf{0.3\,e}$

Irreducible error

Our model

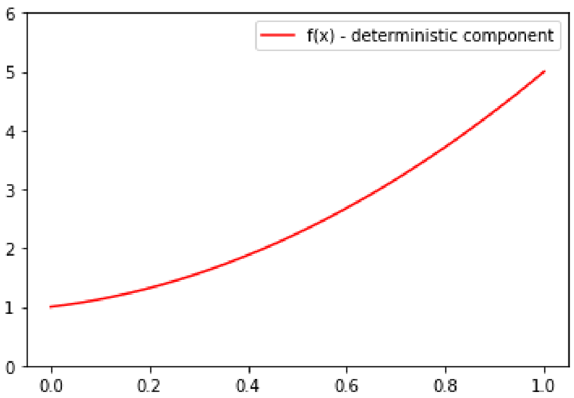
$f(x) = p_0 + p_1 * x$



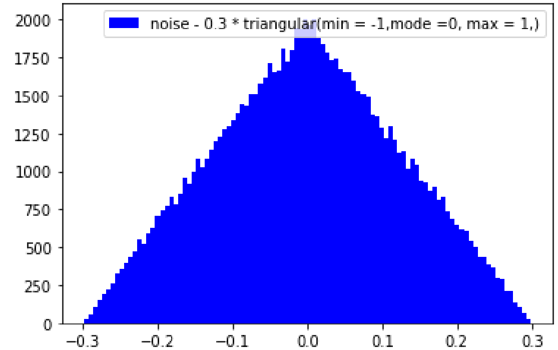
Example

deterministic component

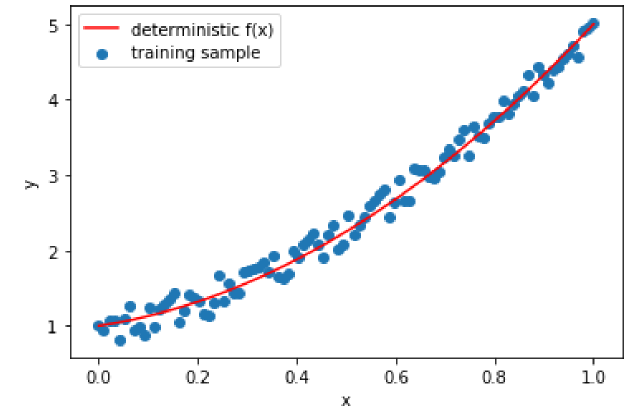
noise



+



=



$f^* = 1 + x + 3\,x^2$

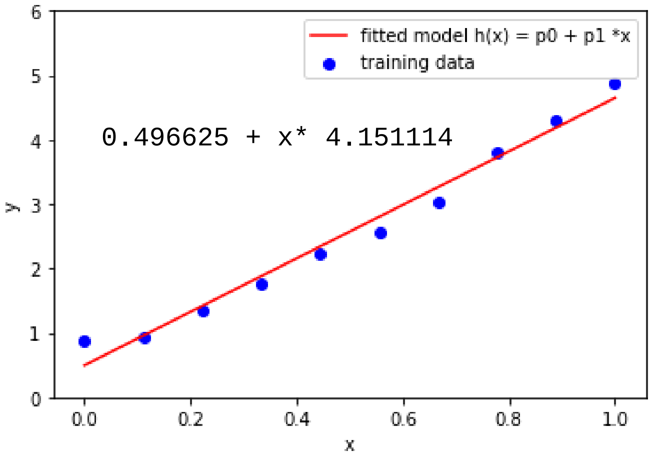
triangular distribution

$1 + x + 3\,x^2 + \mathbf{0.3\ e}$

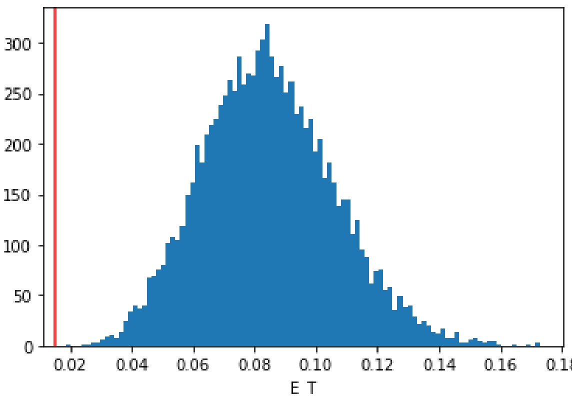
Irreducible error

Our model

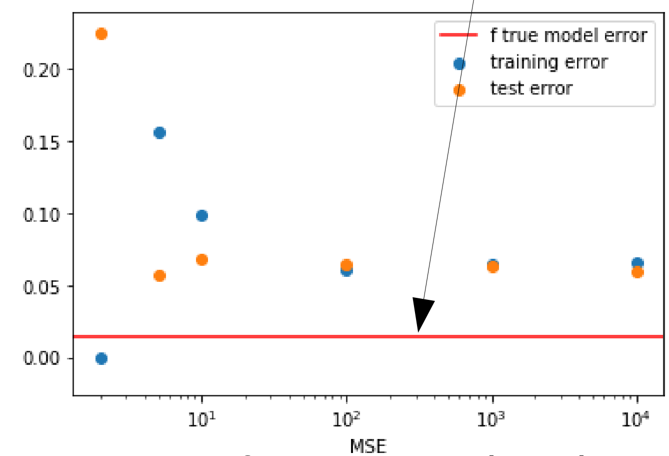
$f(x) = p_0 + p_1 * x$



Error of f
(for different training sets)

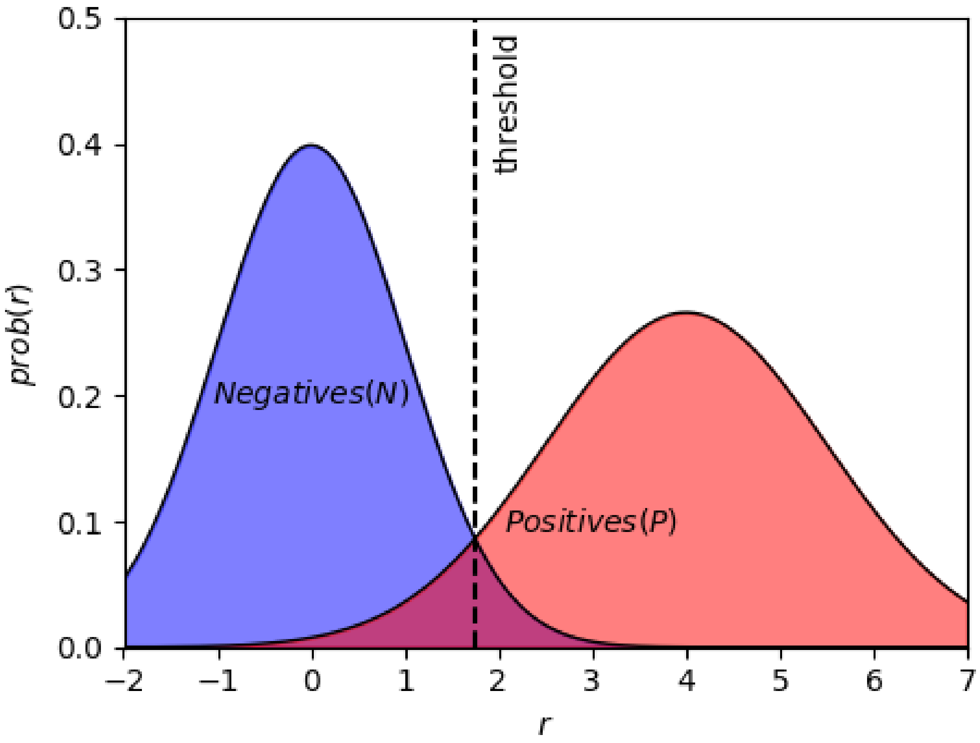


Training error vs sample size

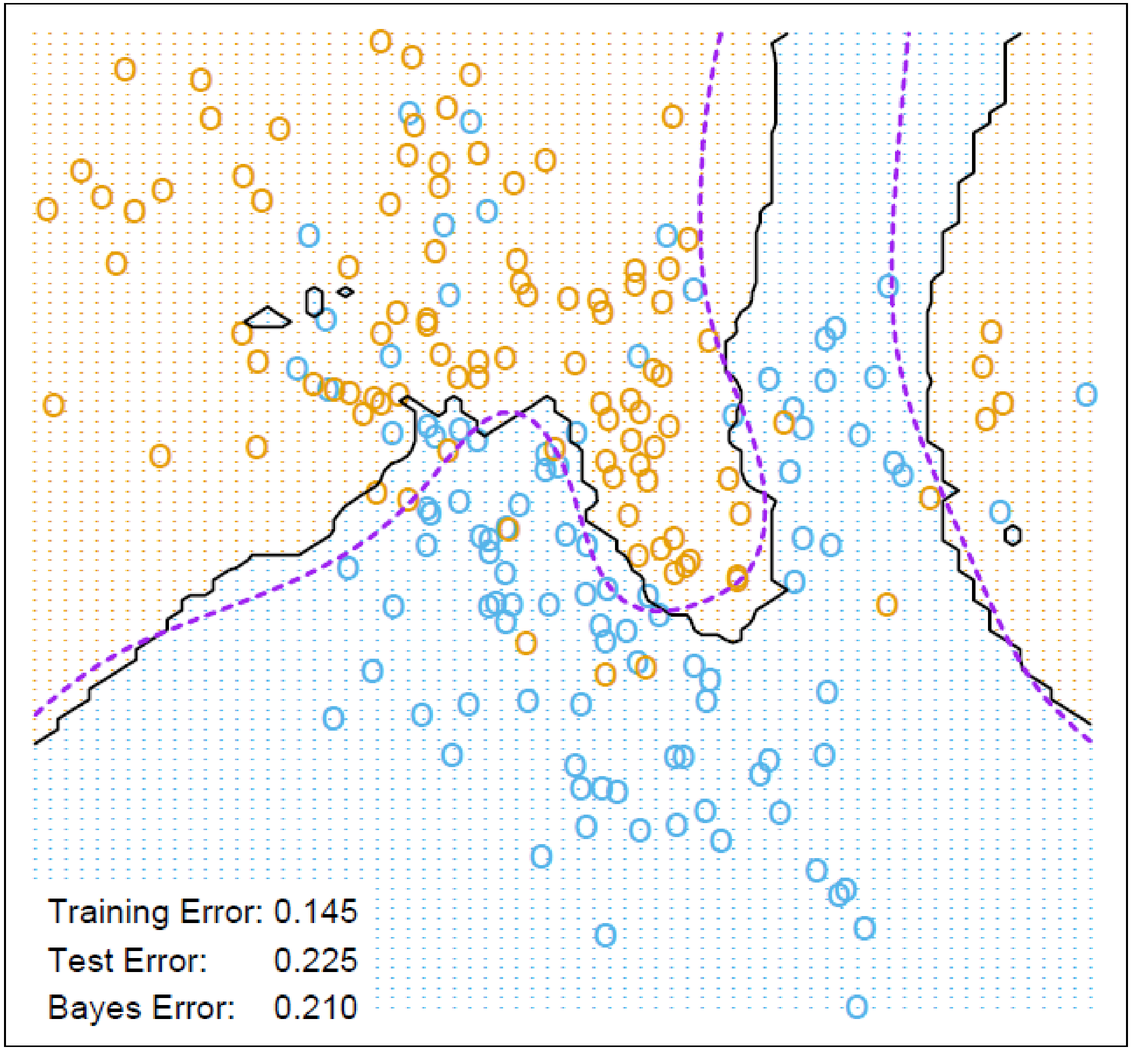


Test error for 100 samples always

Irreducible error and Bayes error



Irreducible error and Bayes error II



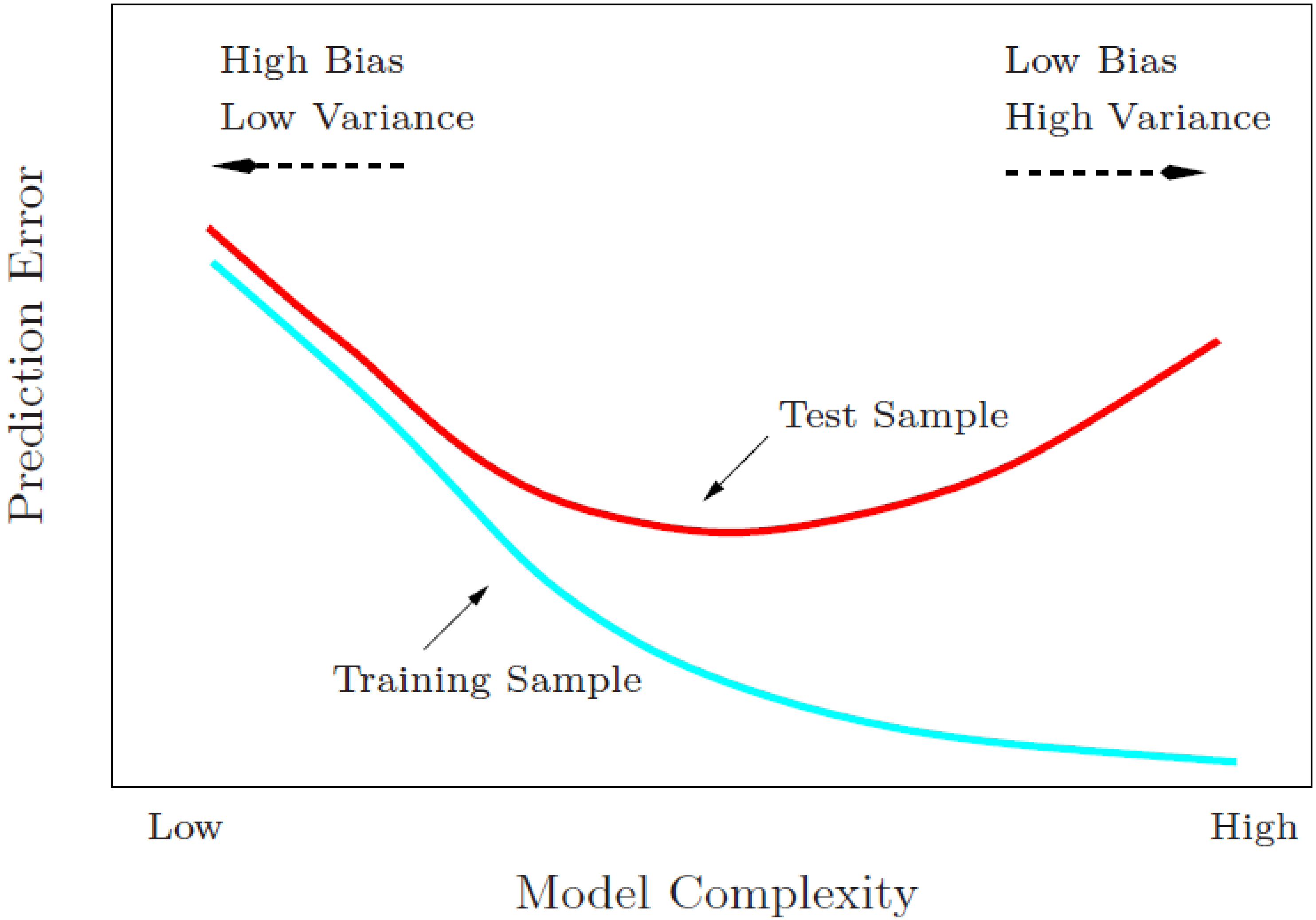
Adapted from Hastie et al. "Elements of Statistical Learning" Second Edition

$$\boxed{f^* = \arg\max_k P(k|x=x_0)}$$

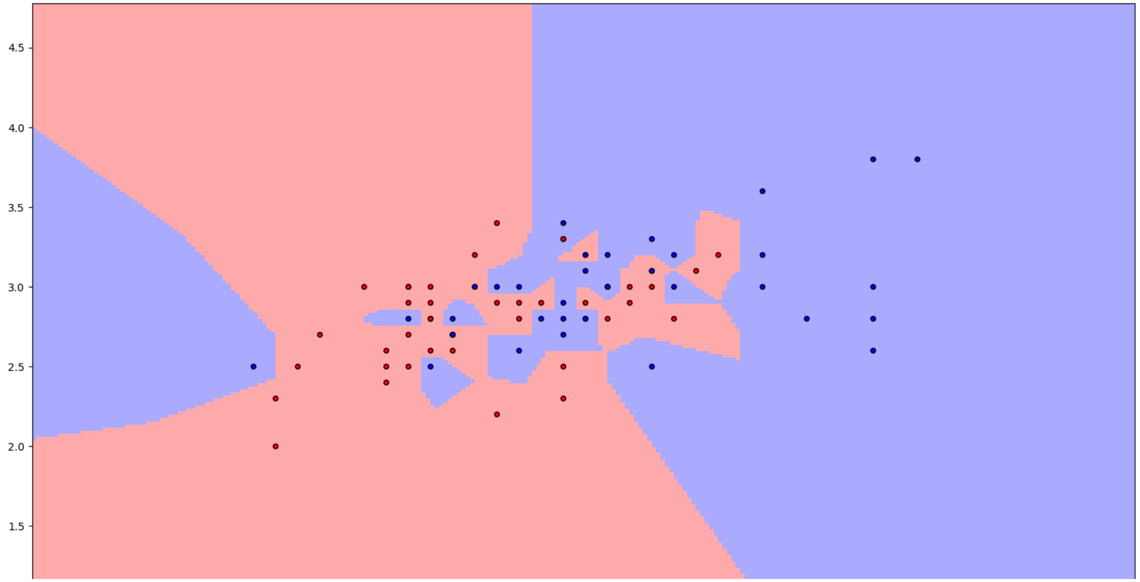
BAYES CLASSIFIER

kNN k=7

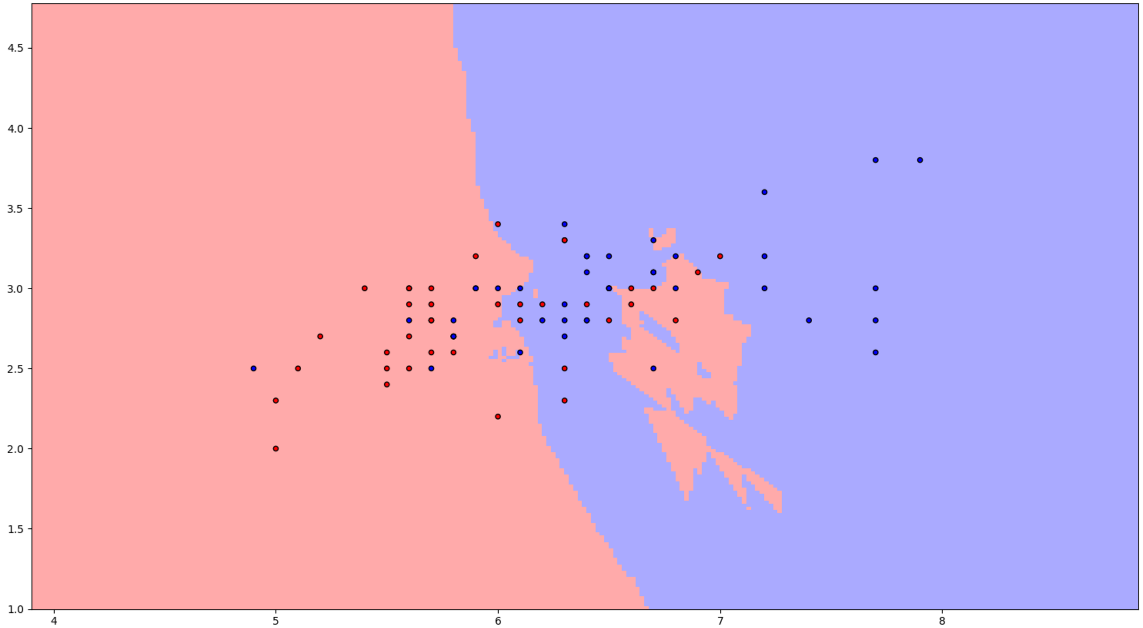
Bias-variance and complexity



Adapted from Hastie et al.
“Elements of Statistical Learning”
Second Edition

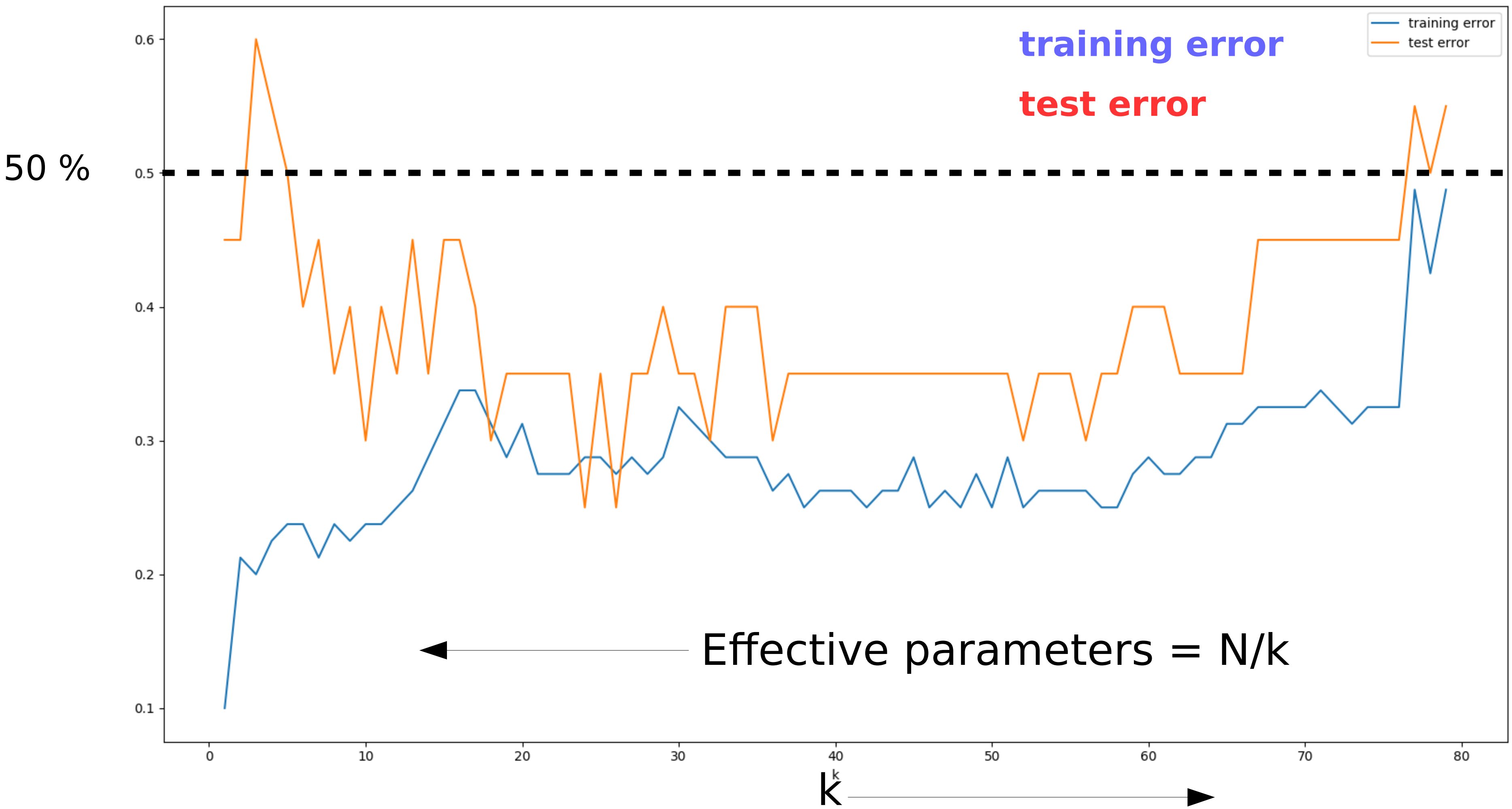


Small k



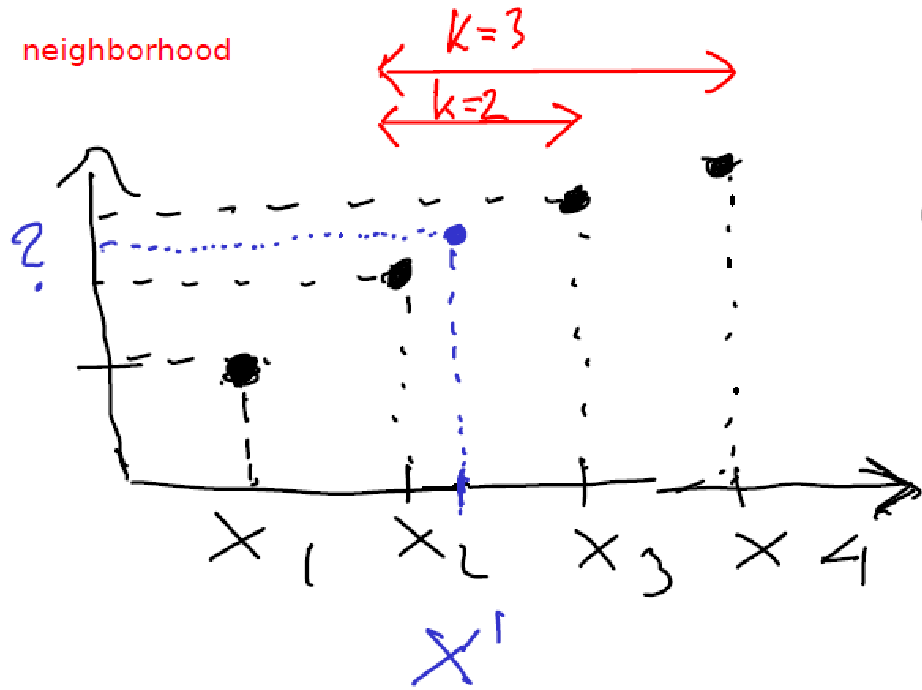
Large k

KNN training and test error



k-NN for Regression

neighborhood

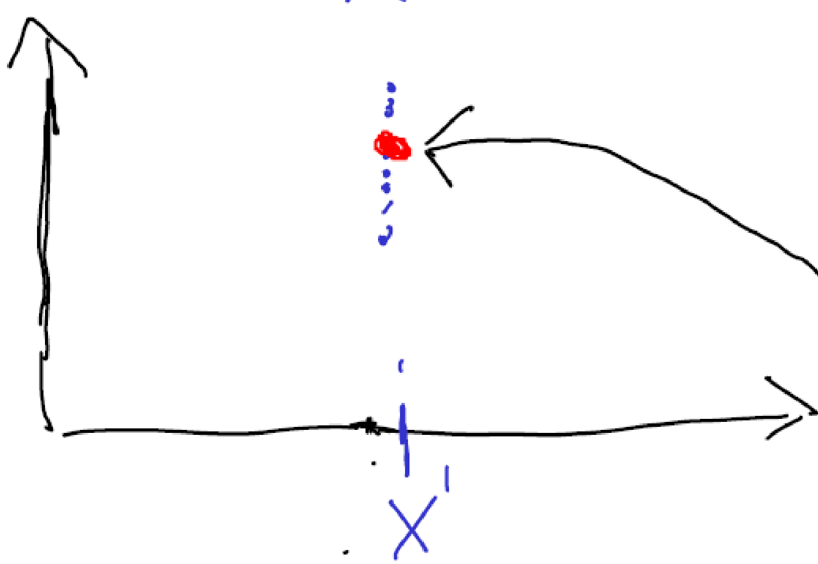


$$k=2$$

$$f=\frac{1}{2}(y_2+y_3)$$

We calculate the average value of y over k nearest neighbors

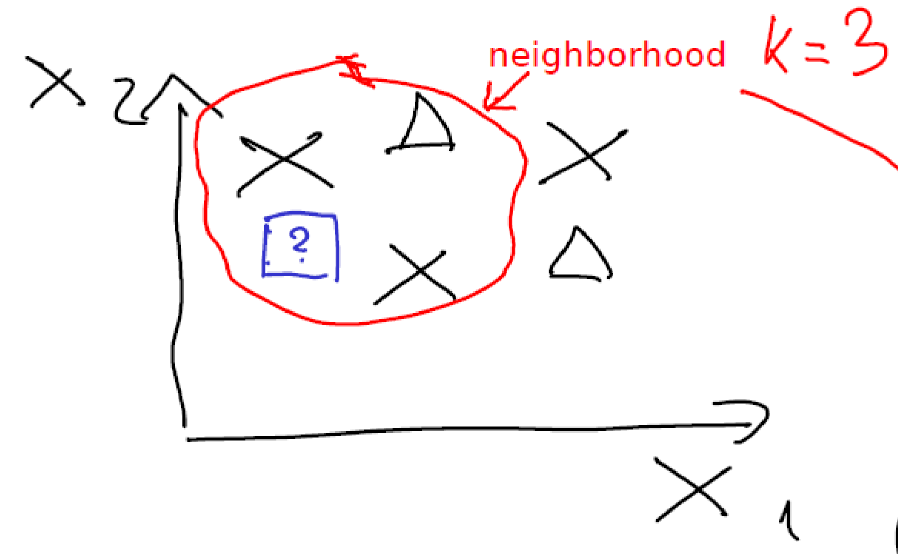
For the regression function (optimal model) we would calculate the expected value in x =x', but we don't have this data, so in k-NN we replace it by averaging over the neighborhood



$$L=(f(x)-y)^2$$

$$f^*=E[y|x=x_0]$$

k-NN for classification



× — 0

Δ — 1

$\{\times, \times, \Delta\}$

$$f = \frac{1}{3}(0 + 0 + 1) = \frac{1}{3}$$

$$\begin{cases} \text{if } f < 0{,}5 & \boxed{?} \rightarrow \times \\ \text{if } f > 0{,}5 & \boxed{?} \rightarrow \Delta \end{cases}$$

so $f = \frac{1}{3}$ $\boxed{?} \rightarrow \times$

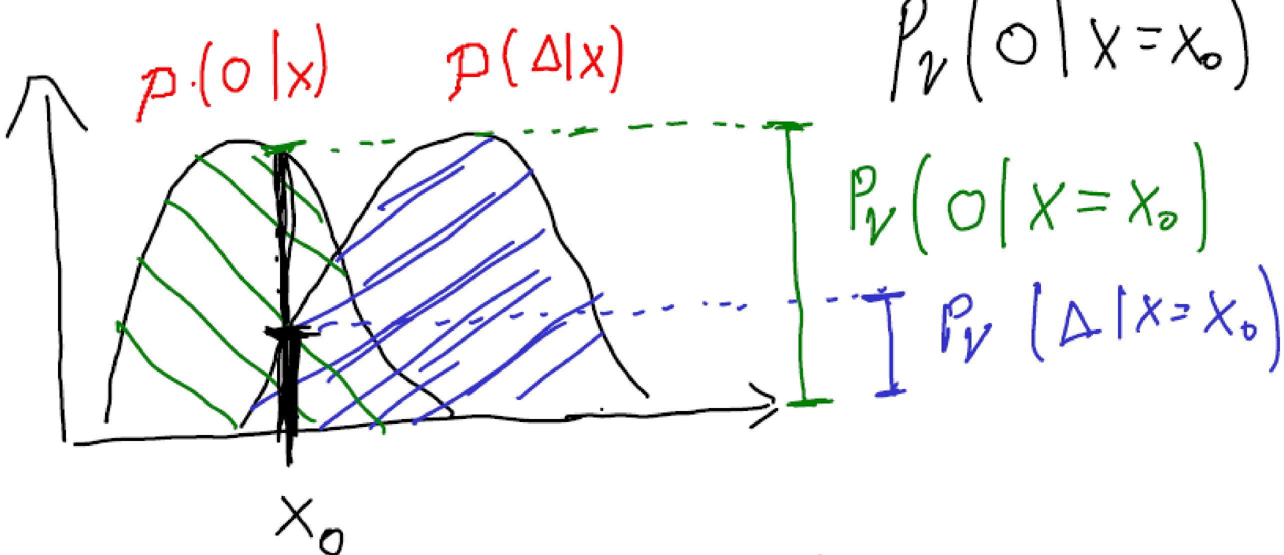
k-NN estimates the probabilities by counting the frequencies of occurence of objects of given class in the neighborhood, and chooses the higher one.

Bayes error number of misclassifications commited by the Bayes classifier (optimal classifier)

$$\boxed{\underset{k}{\operatorname{argmax}} \Pr(k \mid x = x_0)} = k^* - \text{class with the highest probability from all classes } k \text{ at the point } x = x_0$$

$$\text{Bayes error}(x = x_0) = 1 - \Pr(k^* \mid x = x_0)$$

Example with two classes (K=2)



$$\Pr(\circ \mid x = x_0) + \Pr(\Delta \mid x = x_0) = 1$$

$\circ$ - circle

Δ - triangle

Since at $x = x_0$ $\Pr(\circ) > \Pr(\Delta) \rightarrow k^* = \circ$ according to the Bayes classifier

$$\boxed{\text{Bayes error}(x = x_0) = 1 - \Pr(\circ \mid x = x_0)}$$

E.g. if Pr('circle'|x=x0) =80% and Pr('triangle'|x=x0)=20% in 20% of cases we would misclassify the object assuming it is a 'circle'

Error bound for the k-NN for k=2

see http://ssg.mit.edu/cal/abs/2000_spring/np_dens/classification/cover67.pdf

For $k=2$ one can show that in the limit of $N \to \infty$ (number of samples in the training set) the k-NN classifier error rate will be not larger than $2 \times$ Bayes error rate

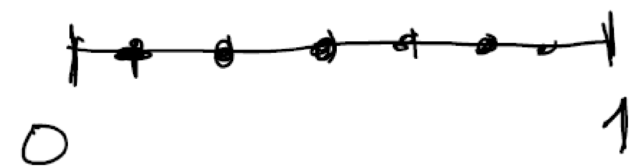
Why not to use kNN everywhere?

CURSE OF DIMENSIONALITY

1) dim=1

$X \in [0, 1]$

$N = 100$ $\quad dist \approx \frac{1}{100}$

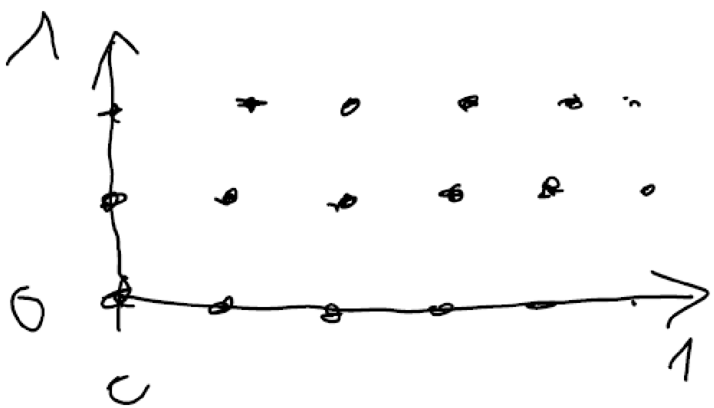


2) dim=2

$X \in [0, 1]^2$

To keep the same distance $\approx \frac{1}{100}$

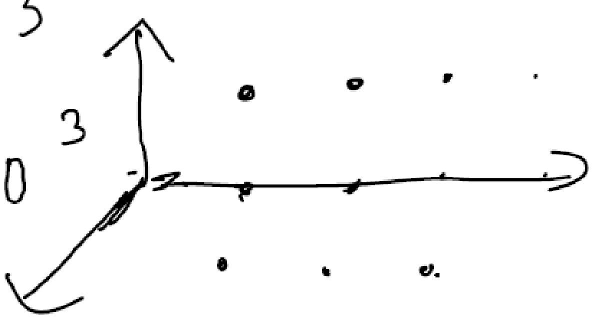
we need $N = 100^2$ points



for dim $= 10 \rightarrow N = 100^{10}$

3) dim=3

$N = 100^3$



We have limited number of points in our training sample so, for high-dim most of the feature space becomes empty

"Local" neighborhood is no local anymore

Problem

We want to use the KNN algorithm for the classification problem. We consider a training sample of $N=10^6$ points, which are distributed approximately uniformly on the available feature space . Calculate the mean distance between neighbors assuming:

- The feature space is 1-D $X=[X_1]$ X_1 in the range of [0,1]
- The feature space is 2-D $X=[X_1,X_2]$ X_i in the range of [0,1]
- The feature space is 3-D $X=[X_1,X_2, X_3]$ X_i in the range of [0,1]
- The feature space is 10-D $X=[X_1,X_2, X_3,\ldots, X_{10}]$ X_i in the range of [0,1]

How many points do we need for 10-D feature space to keep the same distance between the neighbors as in the first case ?

Curse of dimensionality

- For the given number of training samples N, if the feature space dimensions increase (e.g. we add new features) the distance between the neighbor points increases exponentially like $\sim 1/N^{1/d}$

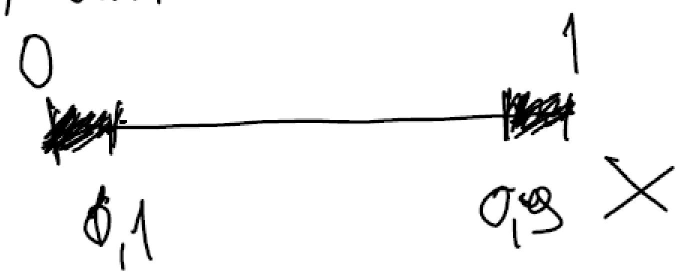
  For $N = 10^6$ samples:
  - one feature $r \sim 10^{-6}$
  - Two features $r \sim 10^{-3}$
  - Three features $r \sim 10^{-2}$
  - ∞ features $r \sim 1$
- For ten features we would need to generate $N = 10^{60}$ training samples to keep the distance between the neighbors at the same level as in the 1-D case.

In higher dimensions, the assumption about the local neighborhood cannot be fulfilled in practice.

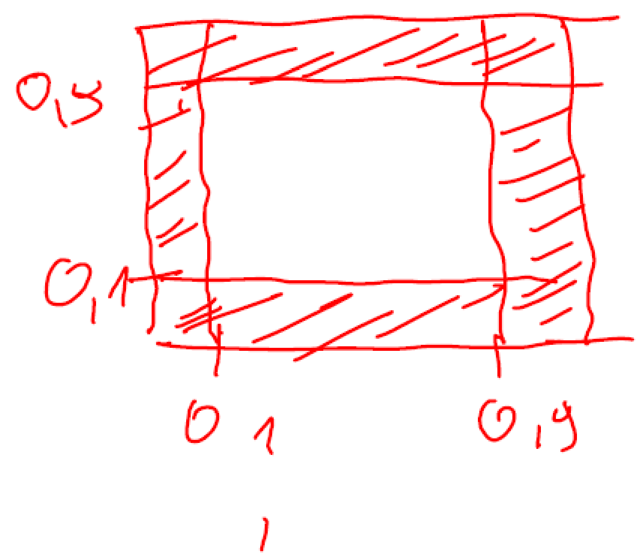
Number of outliers ($X < 0{,}1$ or $X > 0{,}9$)

We assume that the feature space is $\simeq$ uniformly populated

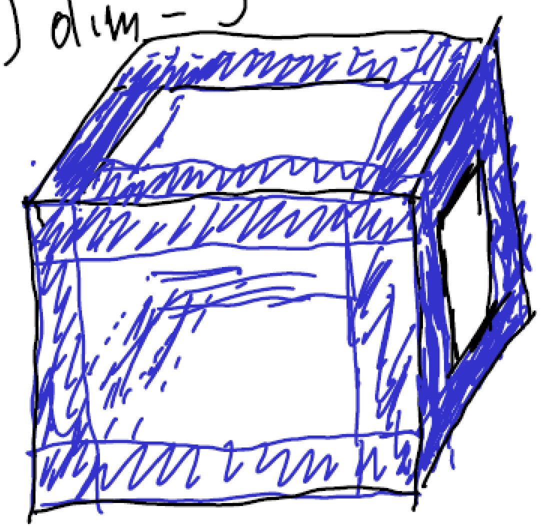
1) dim = 1



2) dim = 2

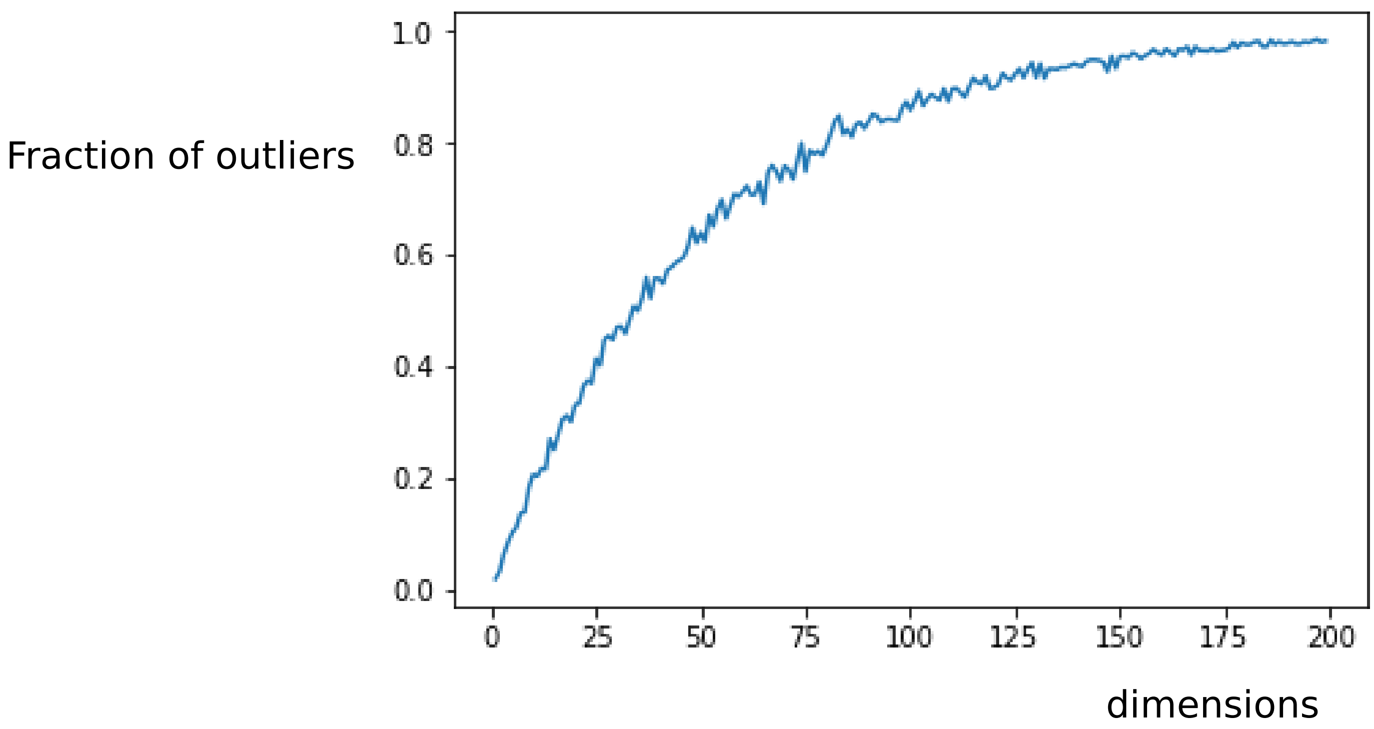


3) dim = 3



For high number of dimensions the number of outliers becomes dominant.
Most of the data lays near the boarders

Curse of dimensionality - outliers



https://github.com/wkrzemien/dataScienceAndML2020/blob/master/notebooks/curse_of_dimensionality/curse_of_dimensionality.ipynb

Linear models

- Are almost never correct

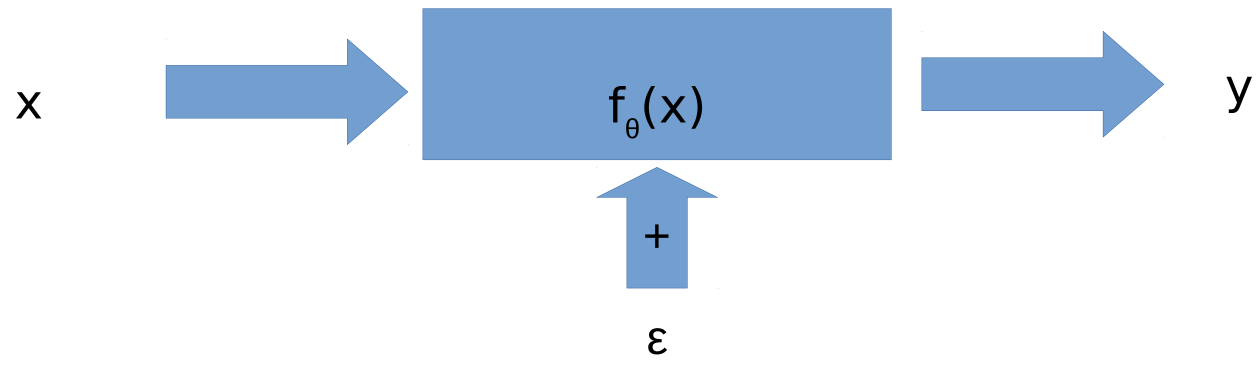
Linear models

- Are almost never correct

but:

- In many cases work reasonably well (~ Taylor expansion)
- relatively easy to interpret
- avoid curse of dimensionality

Additive error model

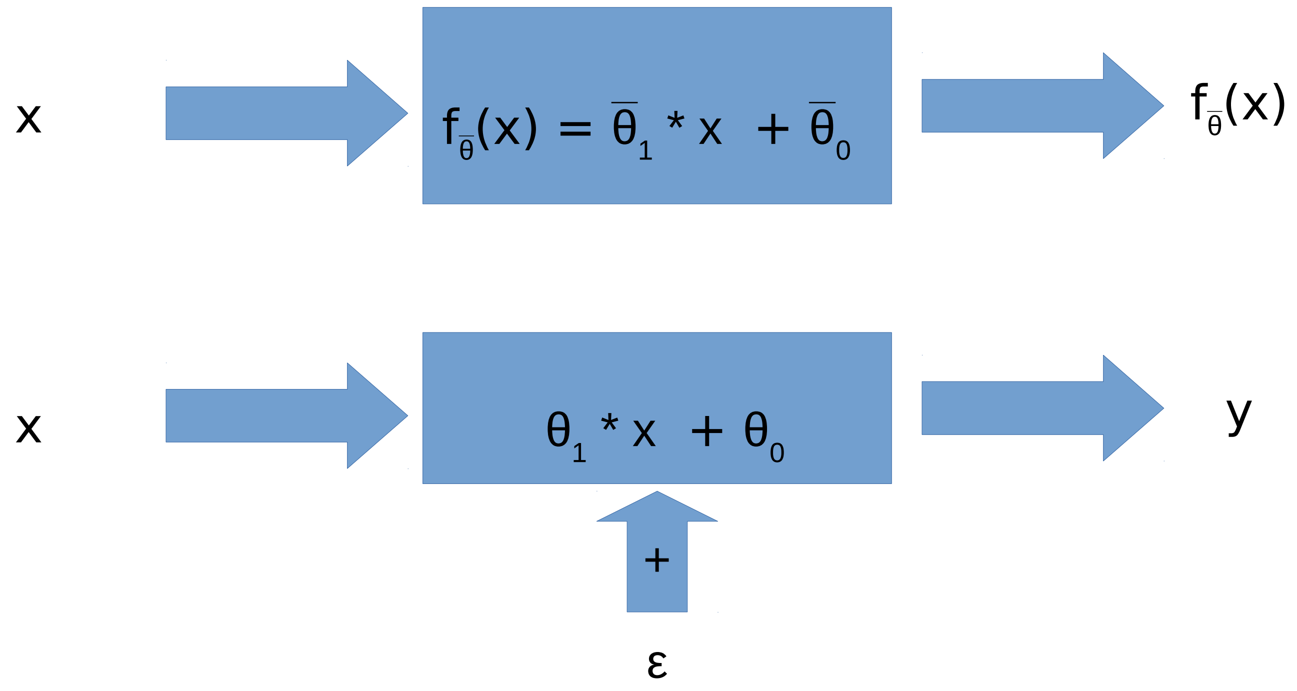


- $E[Y|X] = f_\theta(x)$
- ε – error/noise (typically assuming Gaussian)
- $E[\varepsilon] = 0$
- $Var[\varepsilon] = \sigma^2$

Additive model encapsulates all “indeterministic” behavior in the error term

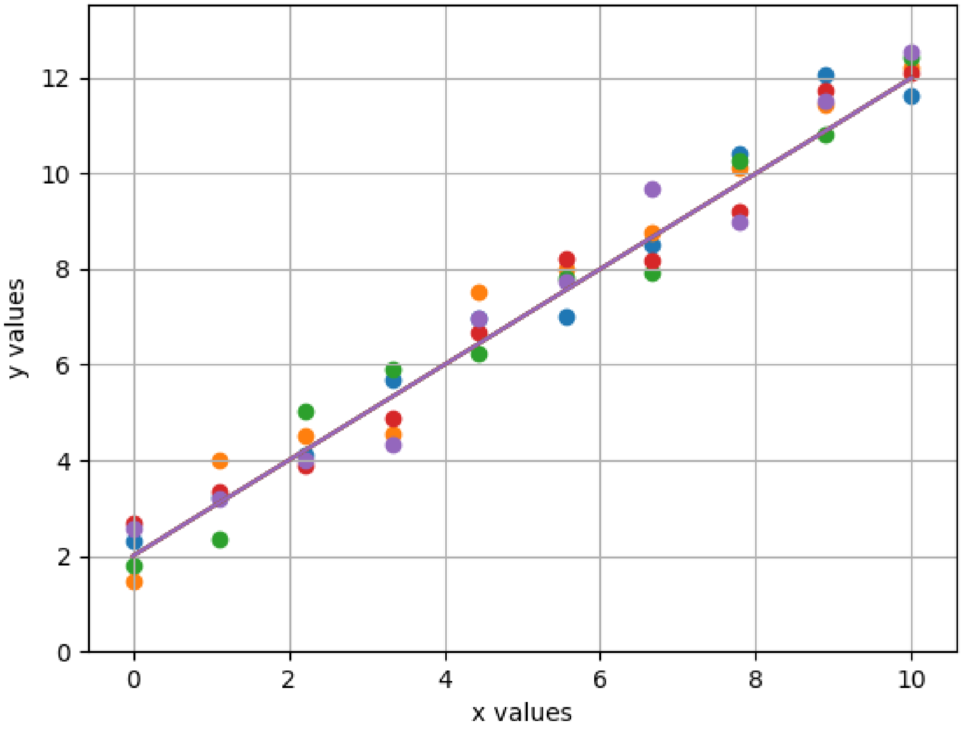
Simple linear regression - model

- We consider only one feature x
- $y = \theta_1 * x + \theta_0 + \varepsilon$
- θ_0 – intercept/bias
- ε – error/noise
- $f_{\bar{\theta}}(x) = \bar{\theta}_1 * x + \bar{\theta}_0$



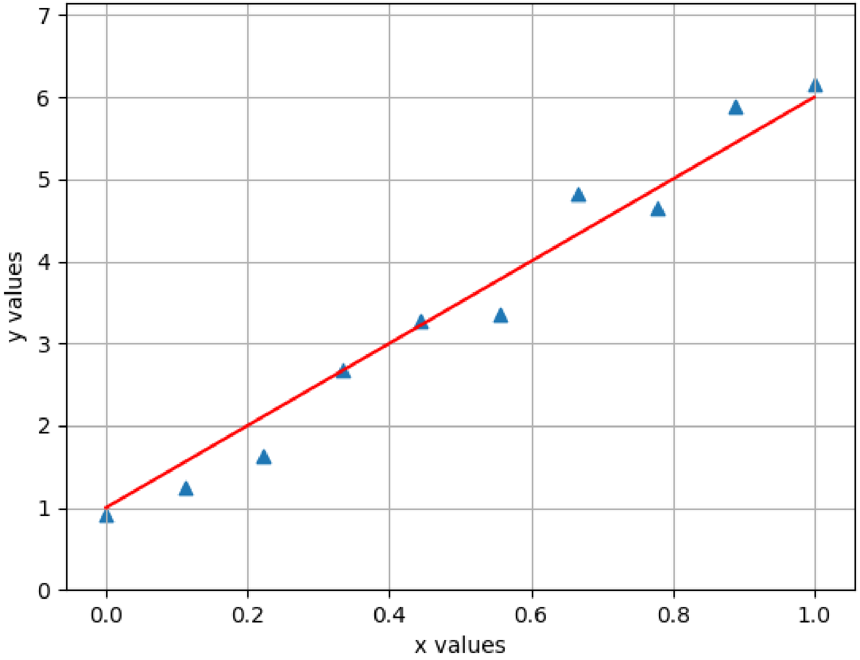
Simple linear regression - model

- We consider only one feature x
- $y = \theta_1 * x + \theta_0 + \varepsilon$
- θ_0 – intercept/bias
- ε – error/noise
- $f_{\bar{\theta}}(x) = \bar{\theta}_1 * x + \bar{\theta}_0$



Strong assumption about the model: function globally linear

Finding optimal parameters



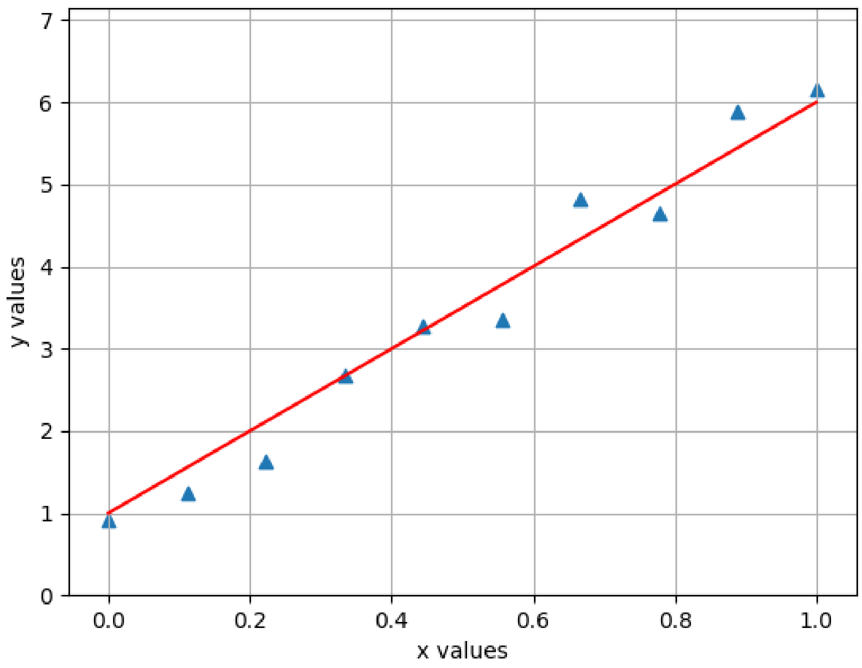
- $\mathbf{y = \theta_1 * x + \theta_0 + \varepsilon}$

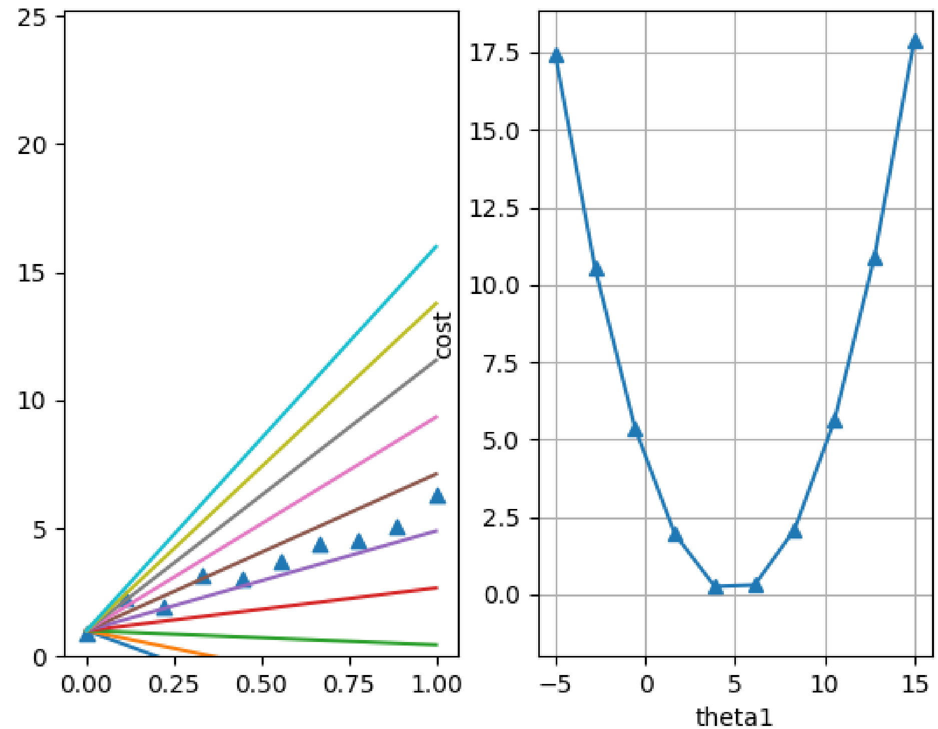
  $\theta_1 = 5$

  $\theta_0 = 1$

  $\varepsilon =$ gauss(0, 0.5)

Finding optimal parameters





- $\mathbf{y = \theta_1 * x + \theta_0 + \varepsilon}$

  $\theta_1 = 5$

  $\theta_0 = 1$

  $\varepsilon =$ gauss(0, 0.5)

$\mathbf{f_{\bar{\theta}}(x) = \bar{\theta}_1 * x + \bar{\theta}_0}$ $\bar{\theta}_1 = ?$ $\bar{\theta}_0 = 1$

$E_T[L(\bar{f}(X),Y)] = 1/N * \sum [(\bar{\theta}_1 * x^{(i)} + \bar{\theta}_0) - y^{(i)}]^2$

over i=1,…N

Ordinary least squares solution

- $f_{\bar{\theta}}(x) = \bar{\theta}_1 * x + \bar{\theta}_0$
- θ_0 – intercept/bias
- By minimization of :

  $err(\bar{\theta}) = E_T[L(f_{\bar{\theta}}(X),Y)] = 1/(2*N) * \sum [(\bar{\theta}_1 * x^{(i)} + \bar{\theta}_0) - y^{(i)}]^2$ over $i=1,\ldots N$ (*)

- $\bar{\theta}_1 = Cov_T[x,y]/Var_T[x]$
- $\bar{\theta}_0 = E_T[y] - \bar{\theta}_1 * E_T[x]$

The same solution can be derived also by maximum likelihood approach assuming Gaussian error distribution

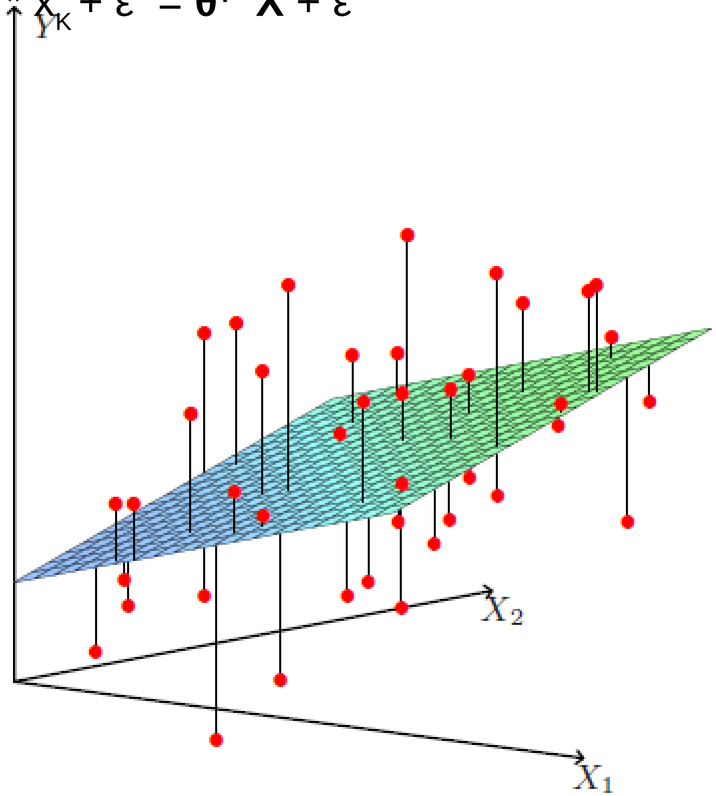
(*) - ½ factor is just for convention, to be conform with gradient descent formula.

Problem:

Derive OLS solution for simple linear regression model $f_{\overline{\theta}}(x) = \overline{\theta}_1 * x + \overline{\theta}_0$

(General) Linear regression

- We consider a vector of features $X = [X_1, X_2, \ldots X_K]^T$ K features
- Extend to $X_0=1$ $\mathbf{X} = [X_o, X_1, X_2, \ldots X_K]^T$ K+1 features
- $\boldsymbol{\theta} = [\theta_o, \theta_1, \theta_2, \ldots \theta_K]^T$
- $y = \theta_0 * x_0 + \theta_1 * x_1 + \theta_1 * x_1 + \theta_1 * x_1 + \ldots + \theta_K * x_K + \varepsilon = \boldsymbol{\theta}^T * \mathbf{X} + \varepsilon$
- ε – error/noise
- $f_{\bar{\theta}}(x) = \bar{\boldsymbol{\theta}}^T * \mathbf{X}$



Adapted from Hastie et al.

“Elements of Statistical Learning”

Second Edition

(General) Linear regression

- We consider a vector of features $X = [X_1, X_2, \dots X_K]^T$ K features
- Extend to $X_0 = 1$ $\mathbf{X} = [X_o, X_1, X_2, \dots X_K]^T$ K+1 features
- $\boldsymbol{\theta} = [\theta_o, \theta_1, \theta_2, \dots \theta_K]^T$
- $y = \theta_0 * x_0 + \theta_1 * x_1 + \theta_1 * x_1 + \theta_1 * x_1 + \dots + \theta_K * x_K + \varepsilon = \boldsymbol{\theta}^T * \mathbf{X} + \varepsilon$
- ε – error/noise
- $f_{\bar{\theta}}(x) = \bar{\boldsymbol{\theta}}^T * \mathbf{X}$

Ordinary Least Square solution:

$$\bar{\theta}^T = (X^T X)^{-1} X^T y$$

(General) Linear regression

- We consider a vector of features $X = [X_1, X_2, \ldots X_K]^T$ K features
- Extend to $X_0=1$ $\mathbf{X} = [X_o, X_1, X_2, \ldots X_K]^T$ K+1 features
- $\boldsymbol{\theta} = [\theta_o, \theta_1, \theta_2, \ldots \theta_K]^T$
- $y = \theta_0 * x_0 + \theta_1 * x_1 + \theta_1 * x_1 + \theta_1 * x_1 + \ldots + \theta_K * x_K + \varepsilon = \boldsymbol{\theta}^T * \mathbf{X} + \varepsilon$
- ε – error/noise
- $f_{\bar{\theta}}(x) = \bar{\boldsymbol{\theta}}^T * \mathbf{X}$

Ordinary Least Square solution:

$$\bar{\theta}^T = (X^T X)^{-1} X^T y$$

(1) For a large number of samples the analytical solution might be slow → iterative methods

(2) In cases of If $(X^T X)^{-1}$ non-invertible, one can use e.g. regularization techniques

Problem: Download the data file from:

http://koza.if.uj.edu.pl/~krzemien/machine_learning2021/materials/datasets/data1.csv

and write a program that:

- For every dataset separately calculate:
  - E[X], E[Y],
  - Var(X), Var(Y),
  - Cov(X,Y)
  - Pearson correlation coefficients
- Visualize the data (X vs Y)
- Visualize the means and variances for all datasets (e.g. E[X] vs dataset number) :-)

Notebook:
https://github.com/wkrzemien/dataScienceAndML2020/blob/master/notebooks/intro/simple_load_data.ipynb

Iris flower dataset

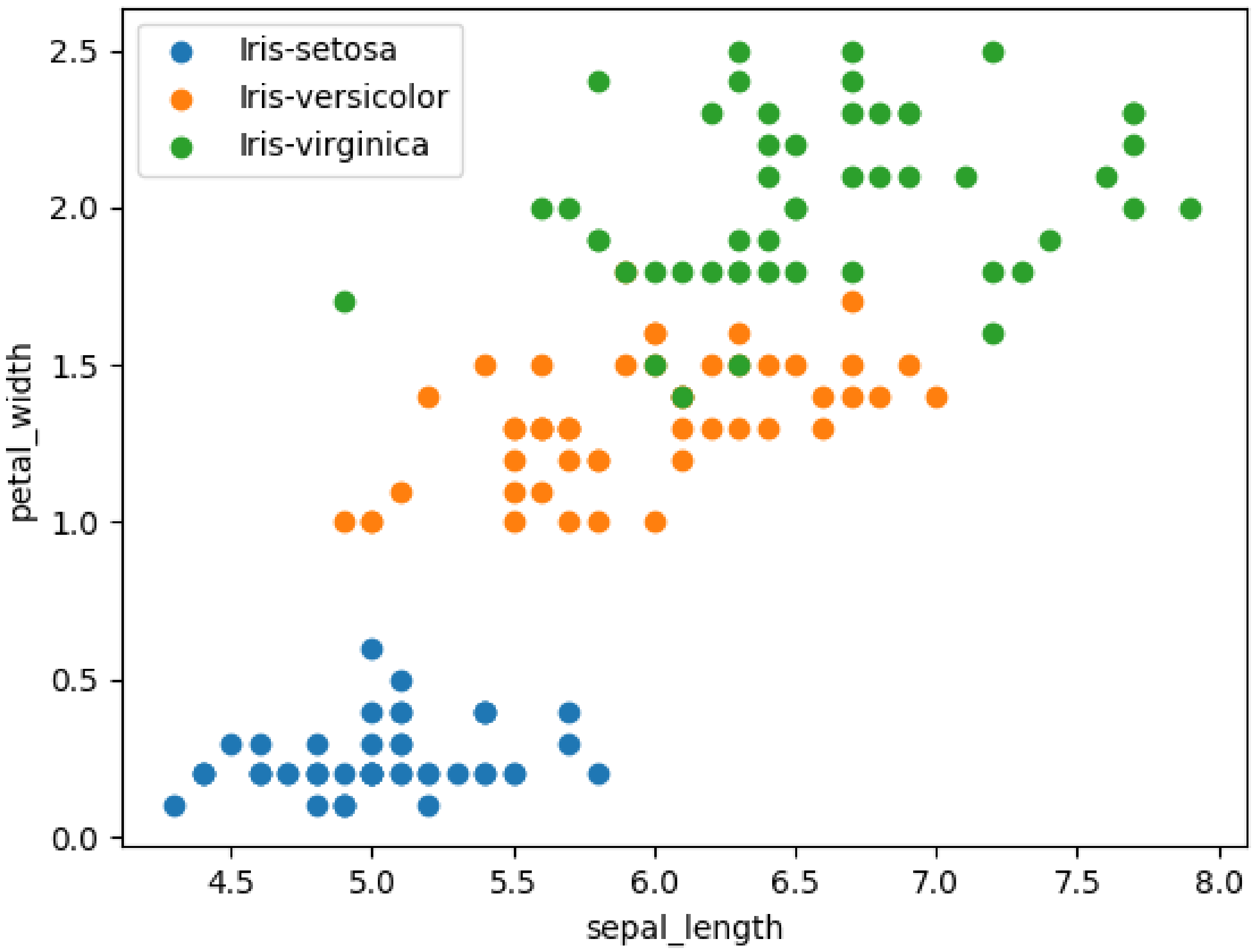
- Classical example of multivariate dataset from 1936
- Nowadays used as a “Hello world” set for ML classification
- Three output classes (Y): Iris setosa, Iris virginica, Iris versicolor
- Four features (X): length and width of petals and sepals
- 50 samples each

Iris setosa

Iris virginica

Iris versicolor

http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data


Iris-setosa
Iris-versicolor
Iris-virginica
2.5
2.0
1.5
1.0
0.5
0.0
petal_width
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal_length

kNN implementation

kNN –proposed implementation steps

Implement :

- distance function **dist(V,W)**
- function **getNeighbors(X, Xtraining,Ytraining)** that returns all the neighbours of Element sorted by distance.
- function **getKNNeighbors(X, Xtraining,Ytraining, k)** – that returns a list of k-nearest neighbors
- function **getMajorityVote(Neighbors)** that returns the result of the majority vote
- function **predict(X)** that returns the predicted class identifier
- function **predictList(Xlist)** that returns the list of predicted class identifier

  e.g. Xlist = [[1,2,0], [1,1,3]] list of two points for which we want to predict the class label

  predictList(Xlist) → [0,1], so first point belongs to class 0, the second to class 1

Problem

- Implement the k-NN algorithm
- Test your implementation on iris_data.csv:
  - Calculate the training error :-)
  - Plot the training error vs k
  - Plot the training error vs number of samples

You can make your own implementation of the kNN algorithm or use the scheme in the notebook below:

Notebook:

https://github.com/wkrzemien/dataScienceAndML2020/blob/master/notebooks/knn/knn_first.ipynb

Thank you