ML for natural and physical scientists 2023 5

K-NN - CART

dr.federica bianco | fbb.space | fedhere | fedhere

this slide deck:

https://slides.com/federicabianco/mlpns23_5

1 Extraction of features

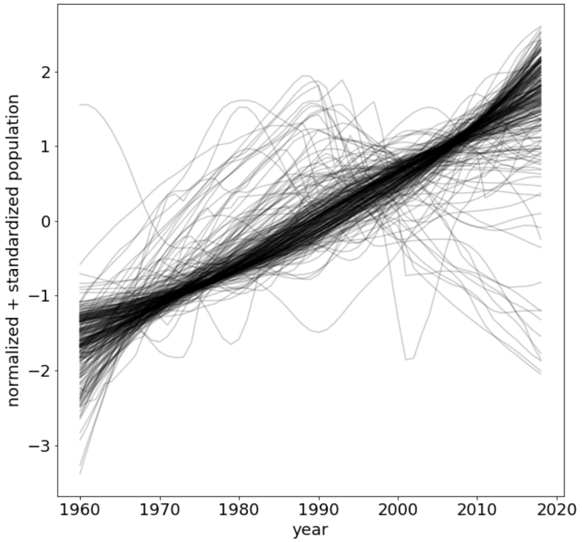
midterm

https://github.com/fedhere/MLPNS2021/blob/main/midterm/MLPNS2021midterm.ipynb

Consider a classification task:

if I want to use machine learning methods
I need to choose:

use raw representation:



e.g. clustering:

1) take each time series and standardize it
(mean 0 standard 1).

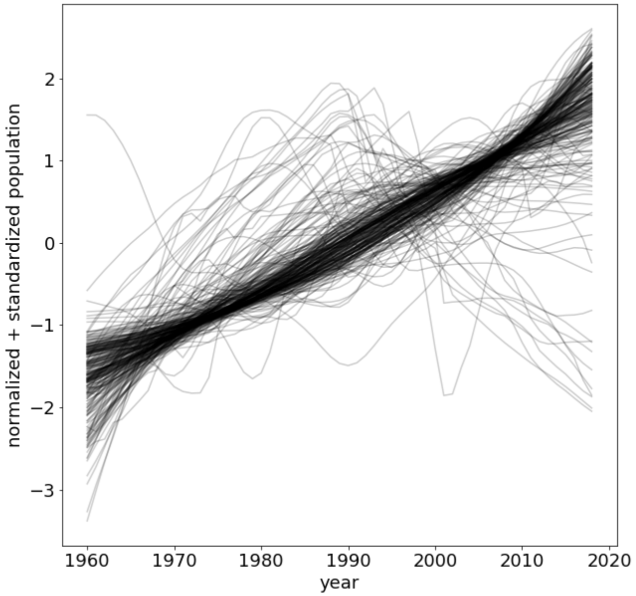
2) for each time stamps compare them to the
expected value (mean and stdev)

essentially each datapoint is treated as a feature

Consider a classification task:

if I want to use machine learning methods (e.g. clustering) I need to choose:

use raw representation



essentially each datapoint is treated as a feature

1) take each time series and standardize it (μ=0 ; σ=1).

2) for each time stamps compare them to the expected value (μ & σ)

problems:

1. *scalability*: for *N* time series of lenght *d* the dataset has dimension *Nd*
2. time series may be asynchronous
3. time series may be warped

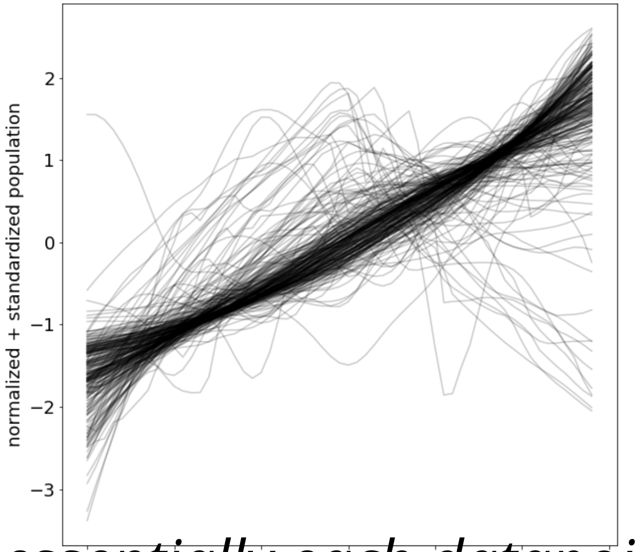
(in small dataset you can optimize over warping and shifting but in large dataset this solution is computationally limited)

Consider a classification task:

if I want to use machine learning methods (e.g. clustering) I need to choose:

choose a low dimensional representation

Extract features that describe the time series:



essentially each datapoint is treated as a feature

simple descriptive statistics (look at the distribution of points, regardless of the time evolution:

- mean
- standard deviation
- other moments (skewness, kurtosis)

parametric features (based on fitting model to data):

- slope of a line fit
- intercept of a line fit
- linear regression R2

Consider a classification task:

the learned representations should:

- preserve the pairwise similarities and serve as feature vectors for machine learning methods;
- lower bound the comparison function to accelerate similarity search;
- allow using prefixes of the representations (by ranking their coordinates in descending order of importance) for scaling methods under limited resources;
- support efficient and memory-tractable computation for new data to enable operations in online settings; and
- support efficient and memory-tractable eigendecomposition of the datato-data similarity matrix to exploit highly effective methods that rely on such cornerstone operation.

http://www.vldb.org/pvldb/vol12/p1762-paparrizos.pdf

2

Supervise learning

what is machine learning?

supervised learning

classification
prediction
feature selection

unsupervised learning

understanding structure
organizing/compressing data
anomaly detection
dimensionality reduction

supervised learning methods
(nearly all other methods you heard of)
learns by example

used to:

classify, predict (regression)

- Similarity can be used in conjunction to parametric or non-parametric methods

- Need labels, in some cases a lot of labels
- Dependent on the definition of similarity

supervised learning methods
(nearly all other methods you heard of)
learns by example

used to:

classify, predict (regression)

- Similarity can be used in conjunction to parametric or non-parametric methods

- Need labels, in some cases a lot of labels
- Dependent on the definition of similarity

clustering vs classifying

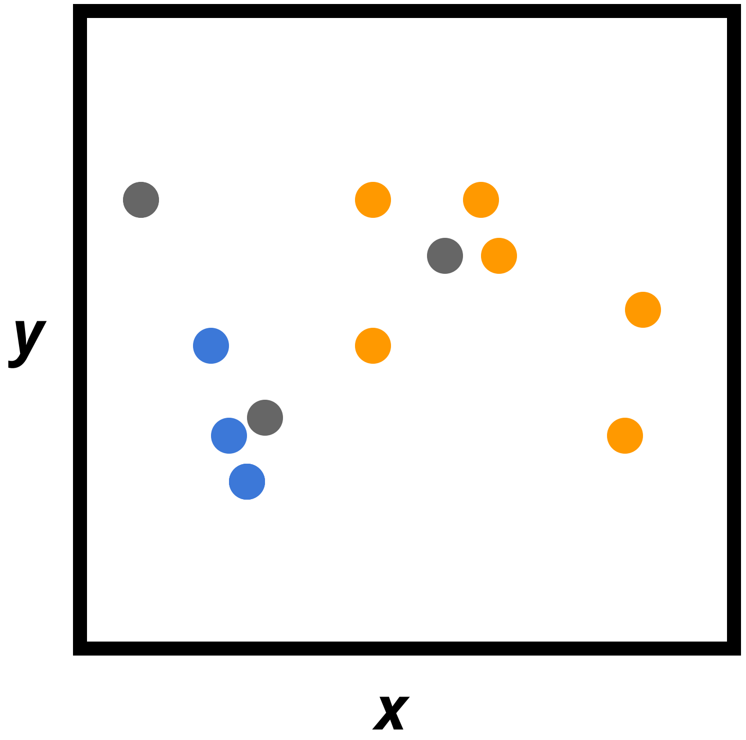
unsupervised *supervised*

goal is to partition the space so that the **unobserved** variables are

separated in groups
consistently with
an observed subset

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

models typically return a partition of the space

supervised ML: classification

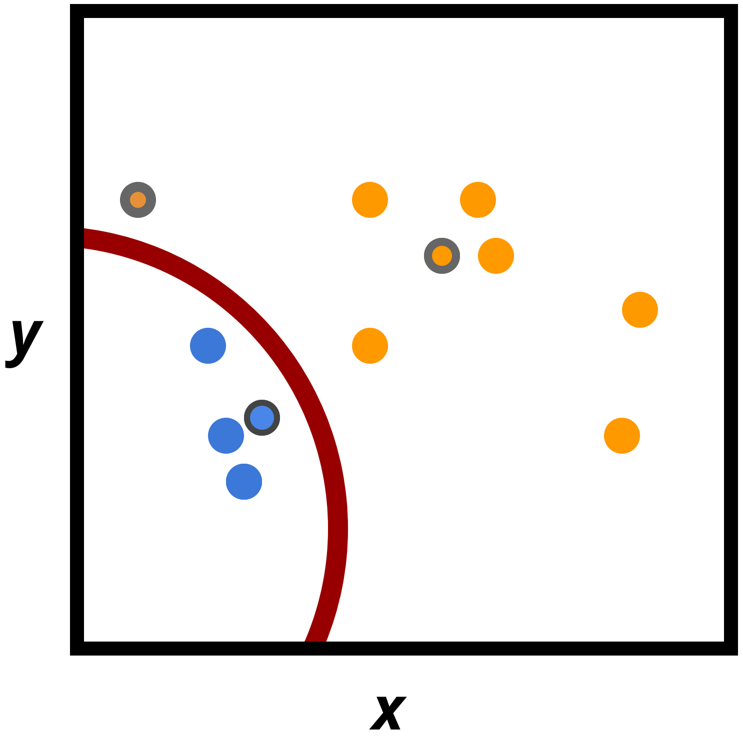
A subset of variables has class labels.
Guess the label for the other variables

SVM

finds a hyperplane that optimally separates observations

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

```
if x**2 + y**2 <= (x-a)**2 + (y-b)**2 :
        return blue
else:
        return orange
```

supervised ML: classification

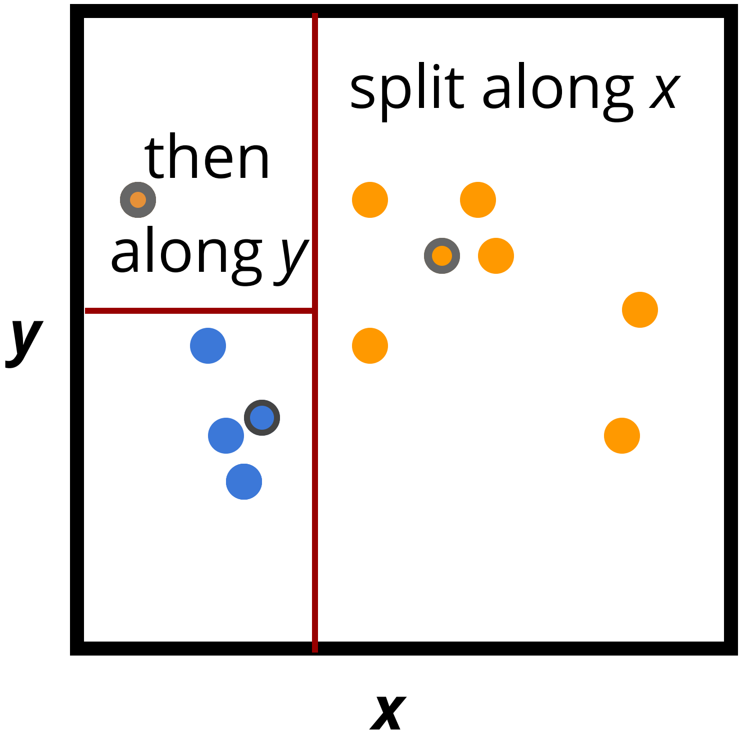
A subset of variables has class labels.
Guess the label for the other variables

Tree Methods

split spaces along each axis separately

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

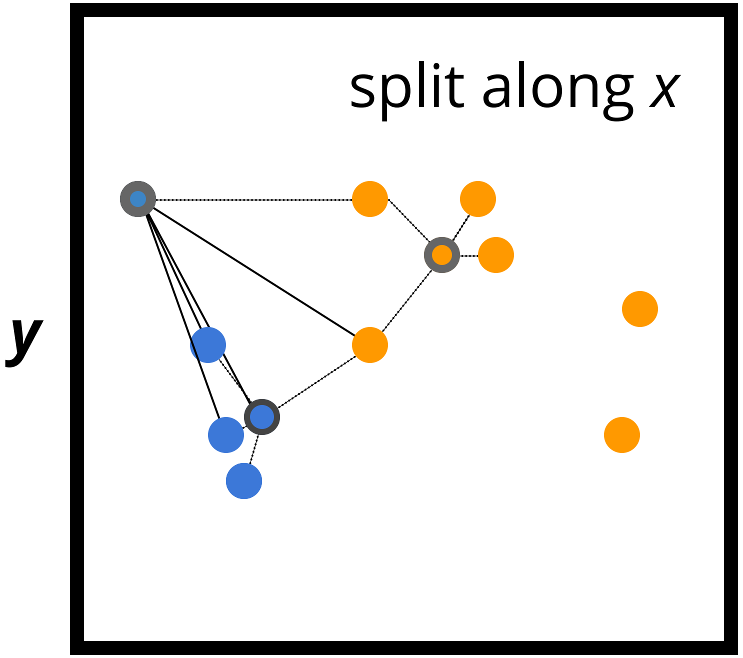
```
if x <= a :
        if y <= b:
                return blue
return orange
```

supervised ML: classification

A subset of variables has class labels.
Guess the label for the other variables

KNearest Neighbors

Assigns the class of closest neighbors



observed features:

$(\vec{x}, \vec{y})$

target features:

$(\overrightarrow{color})$

```
k = 4
if (label[argsort(distance((x,y), trainingset))][:k] == "blue").sum() > (labels[argsort(distance((x,y), trainingset))][:k] == "orange").sum():
        return blue
return orange
```

3

k-Nearest Neighbor

"lazy learner"

k-Nearest Neighbors

Calculate the distance *d* to all known objects

Select the *k closest objects*

Assign the most common among the *k* classes:

```
# k = 1
d = distance(x, trainingset)
C(x) = C(trainingset[argmin(d)])
```

$$C^{kNN}(x) = Y_{(1)}$$

k-Nearest Neighbors

Calculate the distance *d* to all known objects

Select the *k closest objects*

Classification:

Assign the most common among the *k* classes

Regression:

Predict the average (median) of the *k* target values

k-Nearest Neighbors

Good

non parametric

very good with large training sets

Cover and Hart 1967: As n→∞, the 1-NN error is no more than twice the error of the Bayes Optimal classifier.

$$\arg\max_{y} \sum_{h_i \in H} P(y|h_i)P(h_i|D)$$

k-Nearest Neighbors

Good

non parametric

very good with large training sets

Cover and Hart 1967: As n→∞, the 1-NN error is no more than twice the error of the Bayes Optimal classifier.

$$\arg\max_{y} \sum_{h_i \in H} P(y|h_i)P(h_i|D)$$

Let *x*NN be the nearest neighbor of *x*.

For n→∞, *x*NN→*x(t)* => dist(*x*NN,*x(t)*)→0

Theorem: *e*[C(*x(t)*) = C(*x*NN)]< *e*_BayesOpt

*e*_BayesOpt = argmax_y P(y|***x***)

Proof: *assume* P(y|*xt*) = P(y|*x*NN)

(always assumed in ML)

*e*NN = P(*y*|*x(t)*) (1−P(*y*|*x*NN)) +
P(*y*|*x*NN) (1−P(y|*x(t)*)) ≤

(1−P(*y*|*x*NN)) + (1−P(*y*|*x(t)*)) =

2 (1−P(*y*|*x(t)*) = 2ϵBayesOpt,

k-Nearest Neighbors

Good	**Not so good**
non parametric	*it is only as good as the distance metric*
very good with large training sets	**If the similarity in feature space reflect similarity in label then it is perfect!**
	poor if training sample is sparse
	poor with outliers

k-Nearest Neighbors

using Kaggle data programmatically https://www.kaggle.com/docs/api

Lazy Learning

Evaluation on demand, no global optimization - doesn’t learn a discriminative function from the training data but “memorizes” the training dataset instead.

PROS:

Because the model does not need to provide a global optimization the classification is "on-demand".

This is ideal for recommendation systems: think of Netflix and how it provides recommendations based on programs you have watched in the past.

CONS:

Need to store the entire training dataset (cannot model data to reduce dimensionality).

Training==evaluation => there is no possibility to frontload computational costs