LESSON 17

Bayes Optimal Classification,
Course of dimensionality,
Intelligent systems classical models:
Nearest Neighbor Classifiers (kNN)



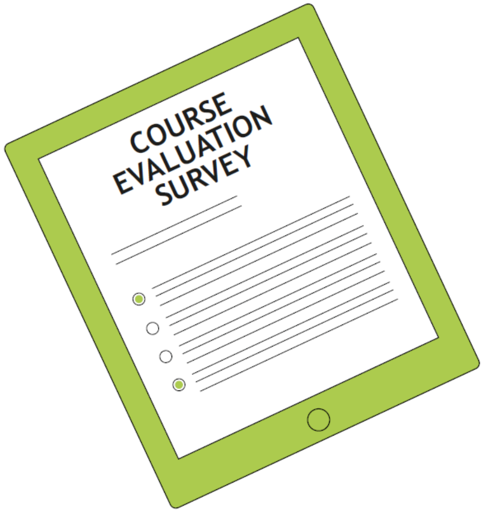

Outline

- Bayes Optimal Classification
- Relevance of Classical (non-neural) models
- Eager and Lazy Learning Methods
- Course of dimensionality
- Nearest Neighbor Classifiers (kNN)
  - Relevance of the kNN in pattern recognition
  - Definition
  - Problems
    - Speed
    - Course of dimensionality
- Main points




Faculty communication

Course Evaluation Survey

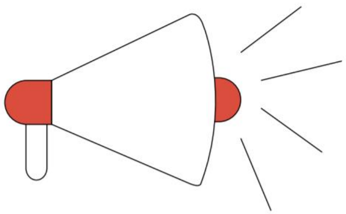


Course Evaluation Survey

A tool for the constant improvement of study courses

PREVIOUS RESULTS (ITA):
https://www.unimi.it/it/ateneo/assicurazione-della-qualita/assicurazione-della-qualita-nei-corsi-di-studio/rilevazione-delle-opinioni-degli-studenti

PREVIOUS RESULTS (ENG):
https://www.unimi.it/en/university/quality-assurance/quality-teaching/survey-opinions-students



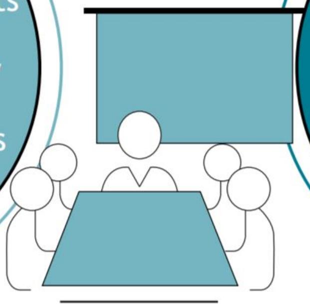
Your opinion really matters!

By filling in the survey:

- you contribute **to the improvement** of your courses
- **your opinion** directly reaches the academic bodies in charge of **teaching quality**

Professors who are motivated to improve the quality of their courses

The Joint Committee of students and professors which analyzes survey results, identifies problems and suggests solutions.



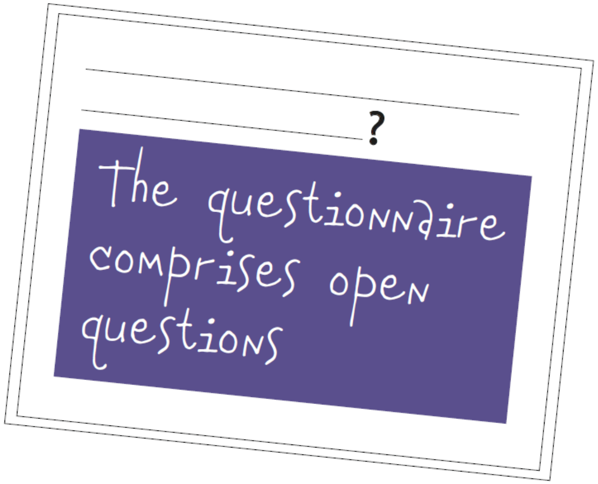
the Education Board where students and professors evaluate and publish the results, propose and put into pratice concrete measures to improve the quality of courses.

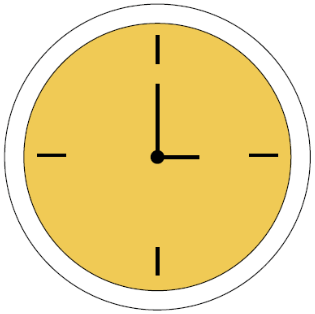
How to fill in the survey

Answer the questions carefully and clearly, be sure to point out to your professors and students' representatives problems and suggestions.

Remember that questionnaires are:

- **completely anonymous**
- **must be filled in online for each of your courses**

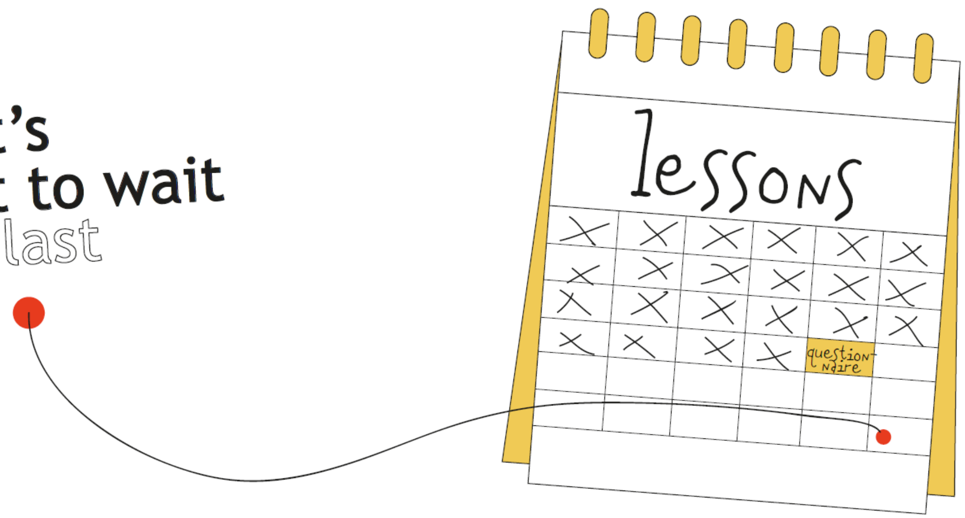




When to fill in the questionnaire

It’s better to fill in the questionnaire after you have attended more than half of the classes in your course.
It’s mandatory to fill it in before registering for the final exam in your course. **You can’t register for the exam until you fill it in.**

Anyway, it’s better not to wait until the last minute!






Bayes Optimal Classification

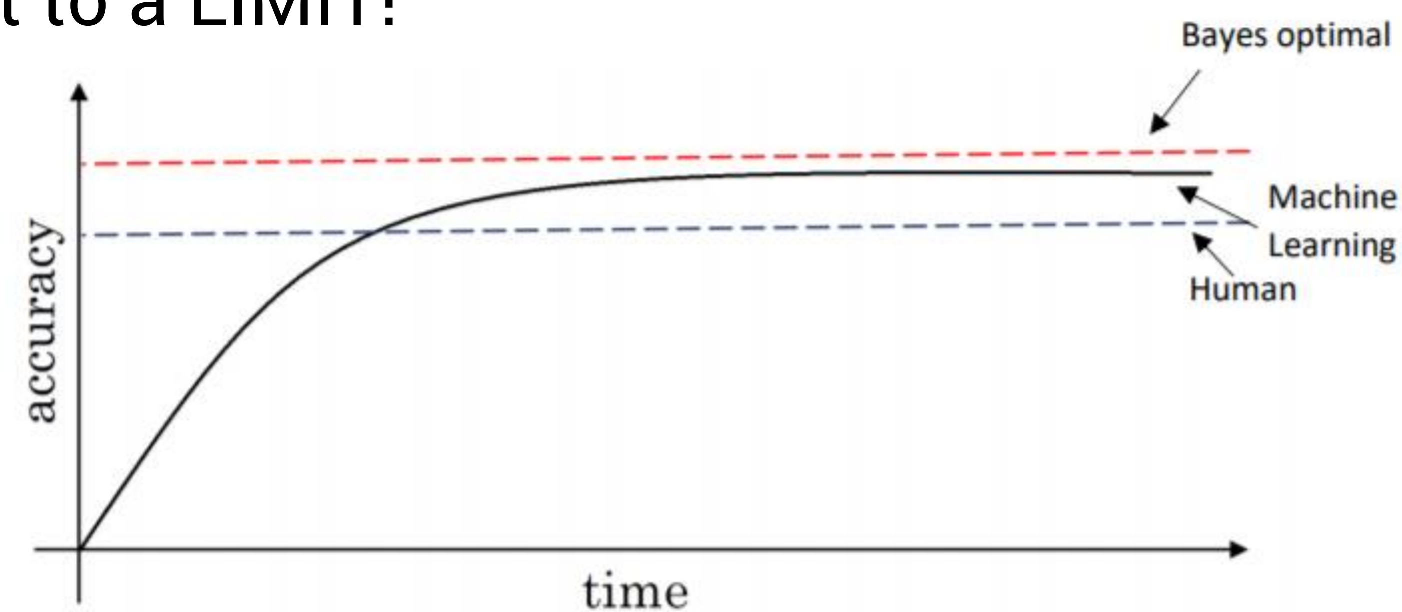
We are chasing the minimal error,

but what is the limit?!

Bayes optimal limit

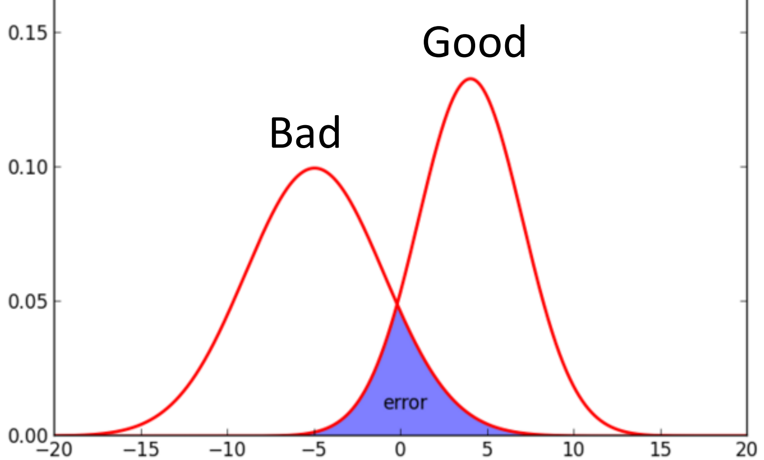
Always chasing the
0% mistakes
can be a mistake!

Given a specific problem/dataset, over time, making more attempts and using new techniques (e.g., Deep Learning) the ML algorithms can outperform humans…
but to a LIMIT!



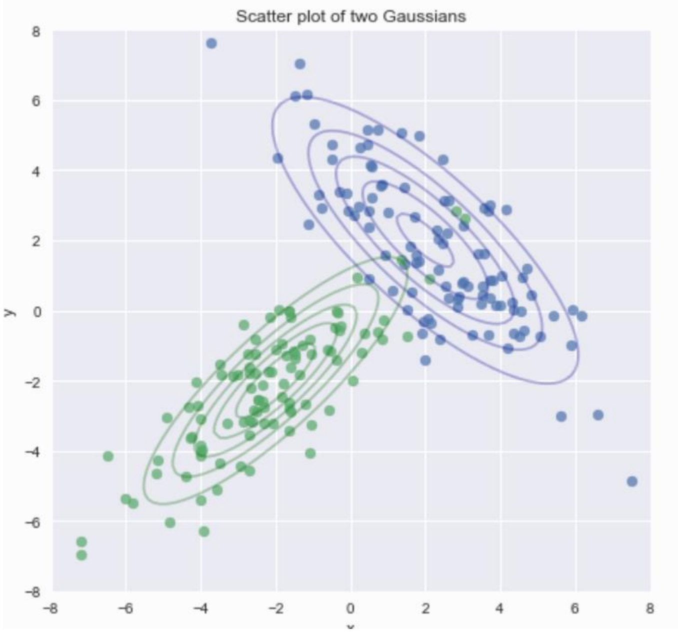
Bayes optimal error

P(F1)

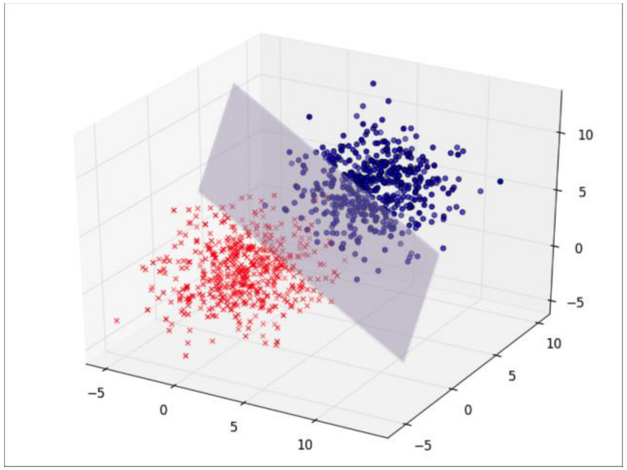


F1

2D



3D

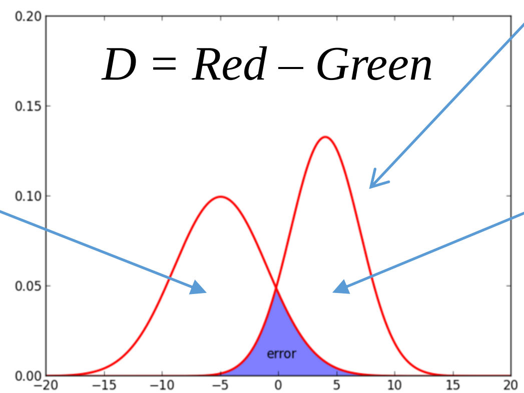


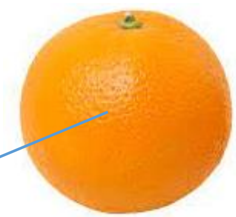
Every time the distributions of the classes are not separated
even the best classifier is producing an error realated to the overlaps

Bayes Optimal error

Estimation of the prob. density

$v_1 = avocados$

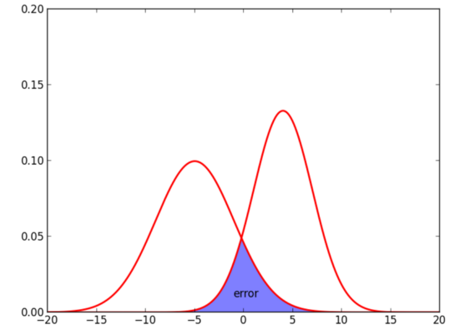




$v_2 = oranges$

(Formula not in the exam)

Bayes Optimal Classification



- Defined as the label produced by the most probable classifier

$$\arg\max_{v_j \in V} P(v_j|D) = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

(Formula not in the exam)

- Computing this can be very very inefficient
- Must known the distributions. Independent features.

- Theoretical concept: No other classification method can outperform this method on average (using the same hypothesis space and prior knowledge).

Minimum possible error can be > 0!

Naive Bayes Classification

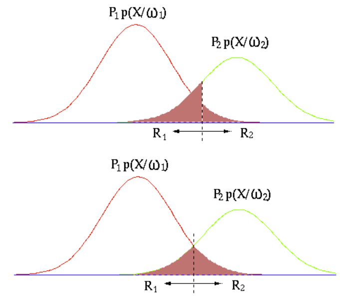
$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

Likelihood

Class Prior Probability

Posterior Probability

Predictor Prior Probability



- From the theory of the Bayes optimal classifier one can build a classifier using the data in the train dataset to build the estimation of the distributions and hence to find a proper separation plane
- Hypothesis: strong independence of the features (rarely satisfied → hence "naïve")
- Requiring a number of parameters linear in the number of variables in a learning problem.
- Suited with high dimensionality of the inputs.
- In general, the accuracy is good w.r.t. other classical classifier!
- A simple block (net=fitcnb(X,Y) … out=sim(…))




Classical models

Pro and cons of standard and non-neural models

Classical (non neural) methods are important!

- (Some of them) are simple → Occam's razor
- The learning methods are very well known, and they tend to be present in all ML tools/environments
- They give you a very solid reference about accuracy
- Explainability is higher
  - E.g., in deep learning models explainability is very hard

I want to do it differently

Examples of classical methods (non neural)

- kNN
- Decision tree
- Linear, quadratic, Logistics classifiers
- Kernel method and SVM

“Eager” and “Lazy” Learning methods

An important difference to be noted

Eager and Lazy Learning Methods

- **Eager** Learning
  - Explicit description of target function on the whole training set
  - The system tries to construct a general, input-independent target function during training of the system
- **Instance-based** Learning (Lazy)
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - The generalization beyond the training data is delayed until a query is made to the system.
  - Referred to as “**Lazy**” learning

Eager Learning

A general, input-independent target function is created during the training

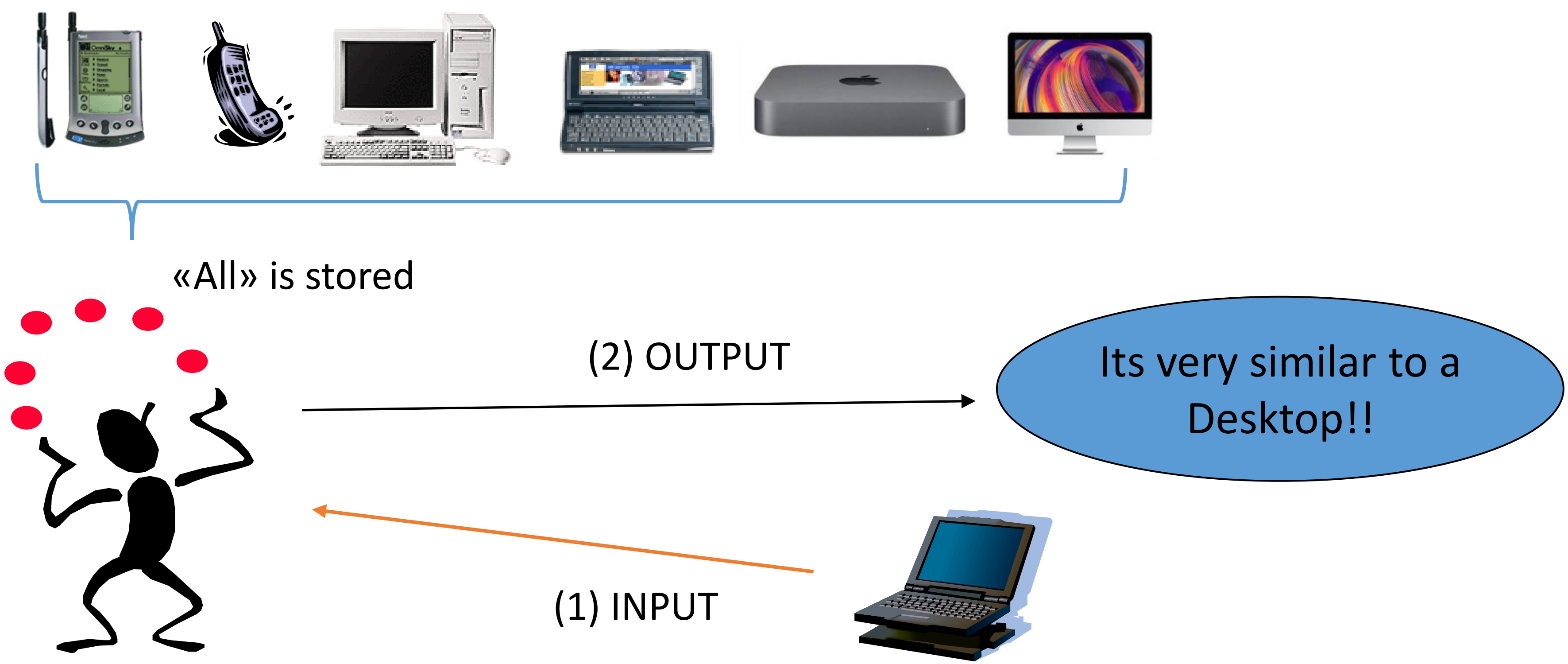
Any random movement
=>It’s a mouse

It’s a mouse!

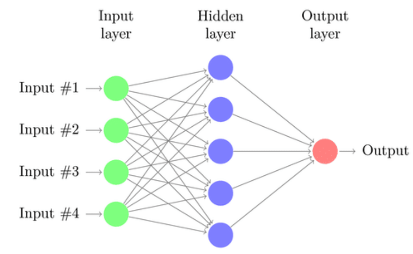
Generalizations and concepts are produced during learning

Instance-based Learning (Lazy)

The generalization beyond the training data is delayed until a query is made to the system.

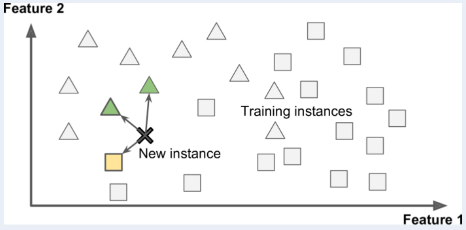


Tradeoff/features

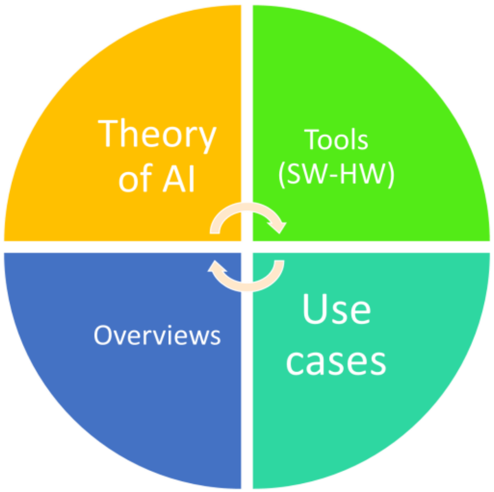


	Training	Recall (deployment)
Eager learners	• Long • #DoF limited (not for deep nets)	• Fast • Low Memory
Instance-based	• Fast • #DoF ←→ data	• Long • Large Memory req.

Examples

	Non-neural (classic) models	Neural methods
Eager learners	• Decision Trees • Induction and rule-based systems	• Feed-forward • Convolutional Neural Networks (CNN)
Instance-based 	• k-Nearest Neighbors algorithm • Support Vector Machines • Weighted Regression • Case-based reasoning	• Kernel Machines • Radial Basis Function Neural Networks (RBFNN)

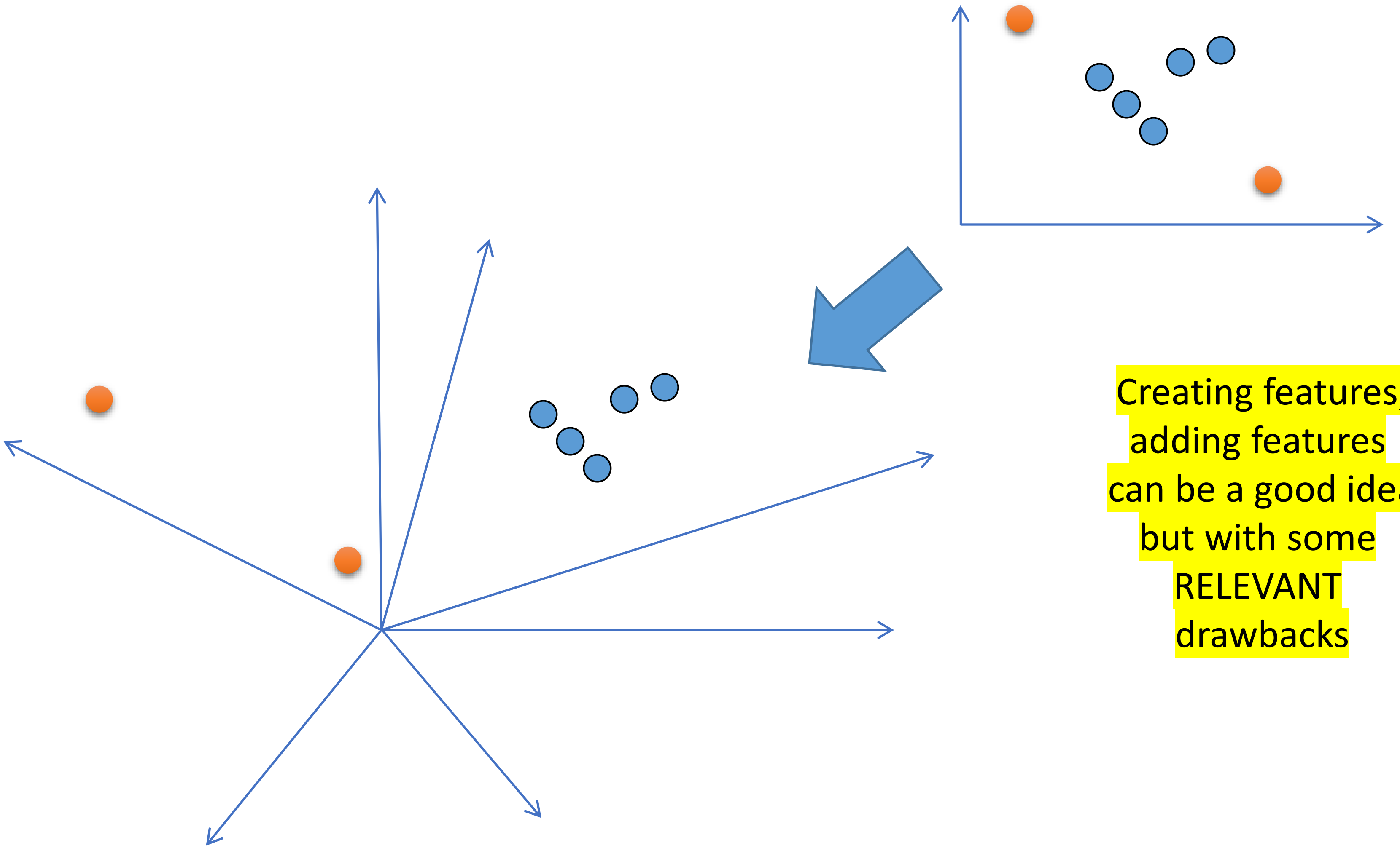
Theory: Curse of Dimensionality



When dimensions matter!

Every time you what to add a new feature think…

Curse of Dimensionality



Creating features, adding features can be a good idea but with some RELEVANT drawbacks

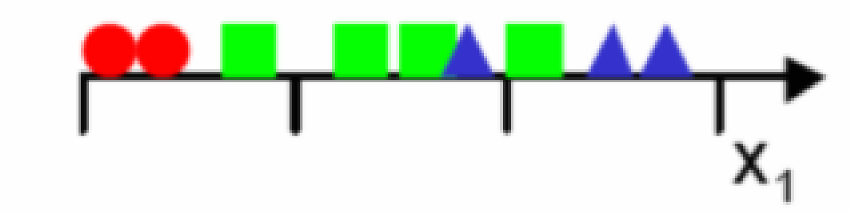
Curse of Dimensionality

- In data science and machine learning adding more attributes is always helping the learner?
- More information can hurt?
  → Sometimes it does!
- Curse of dimensionality.
  - It refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces (often with hundreds or thousands of dimensions)…
    - A Full HD Image 1920 × 1080 to be processed by a NN is composed by 2.073.600 pixel.

The "bin" algorithm example

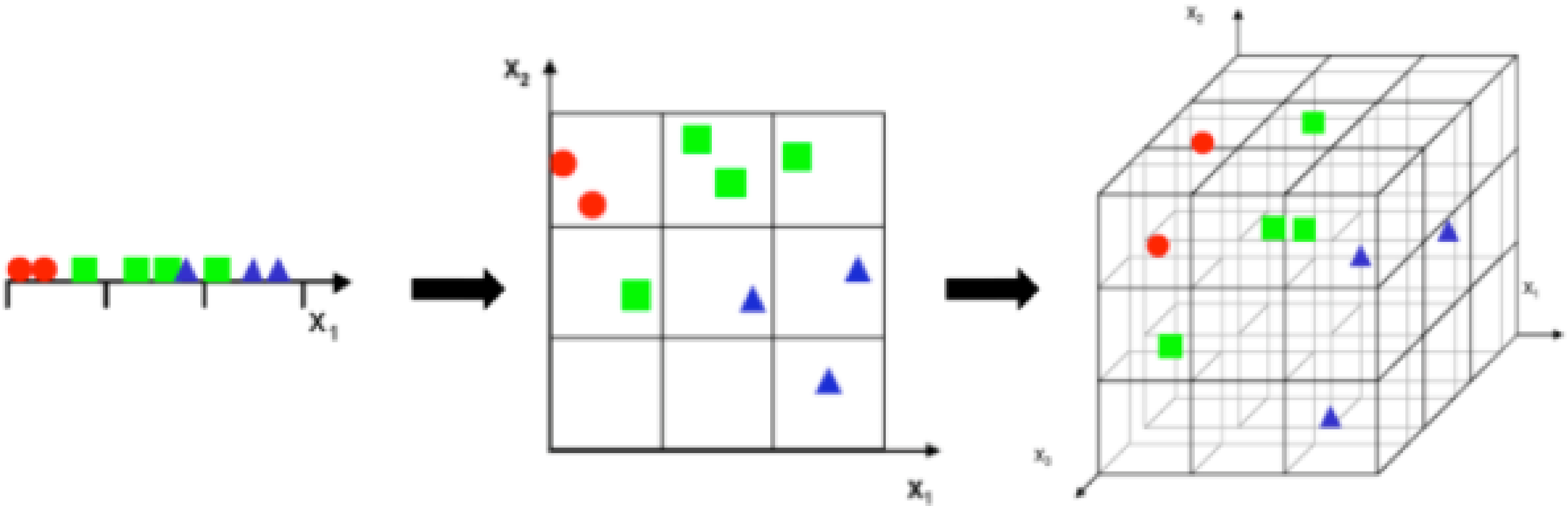
- The algorithm:
  1. divides the feature space uniformly into bins and
  2. for each new example that we want to classify, we just need to figure out the bin the example falls into and find the predominant class in that bin as the label.

- **One feature where the input space is divided into 3 bins:**



- **Noticing the overlap, we add one more feature…**

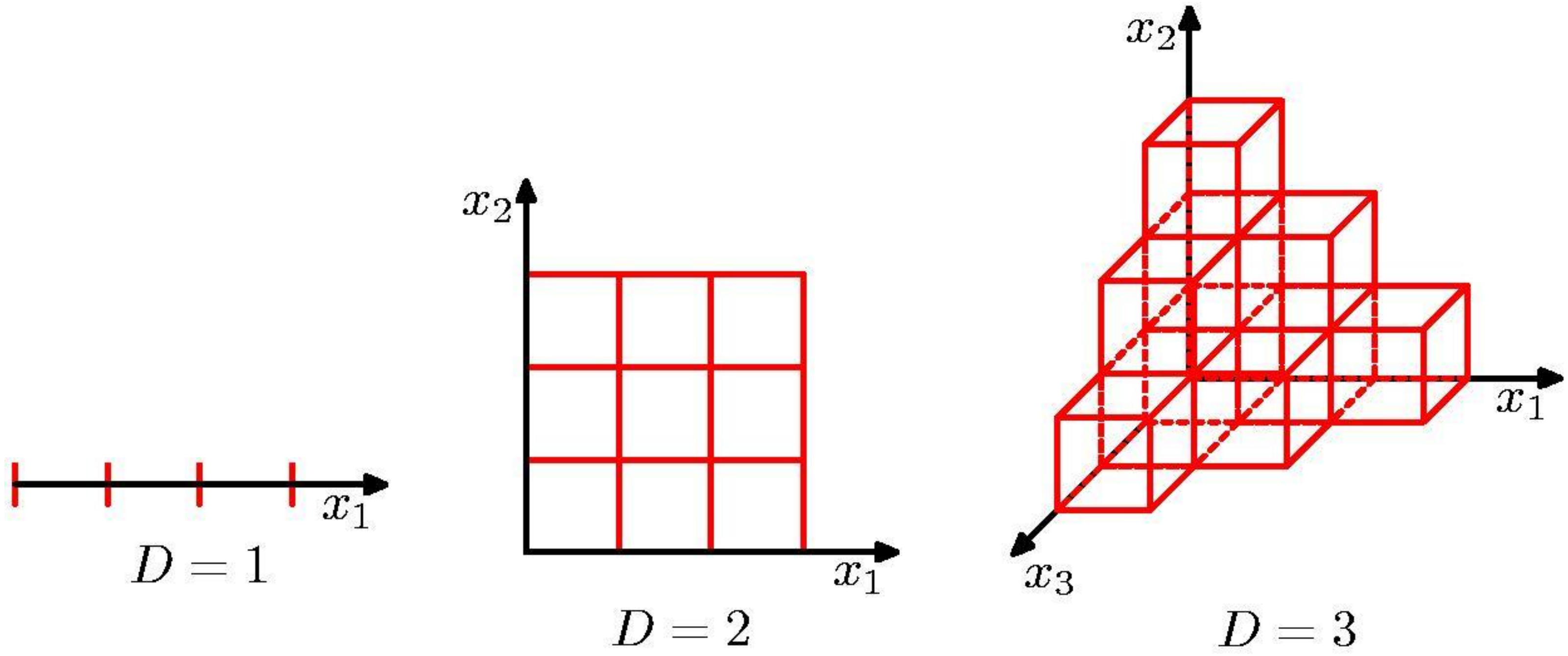
1D→2D→3D→ … here's the problem..



- With increasing dimensionality, the number of bins required to cover the feature space increases exponentially and there won't be enough data to populate each bin.
- Finding the predominant class in each bin or finding the class condional probabilities (p(x given C) is very difficult in high dimensional spaces.

It's hard and harder to have data to process the distributions!

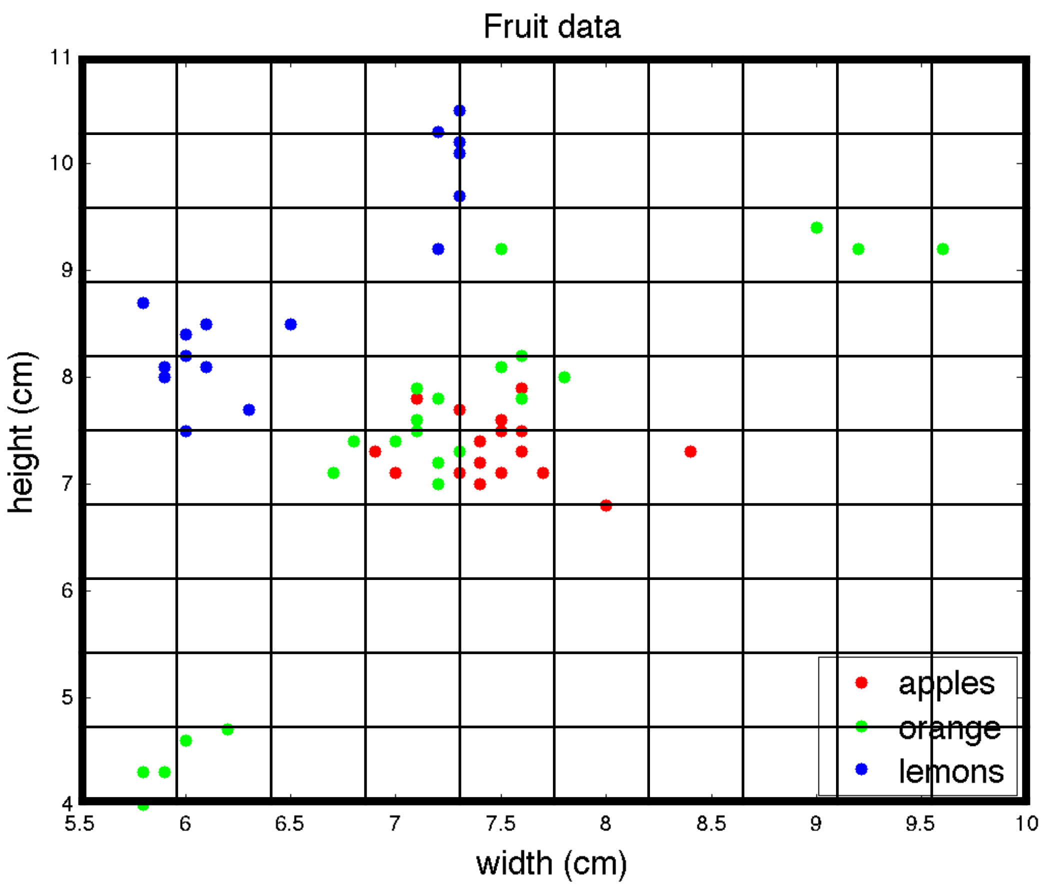
Curse of Dimensionality



As dimensionality D increases,
the amount of data needed increases **exponentially** with D.

Numbers about the Curse of Dimensionality

How many neighborhoods are there?



d = 2

#bins = 10x10

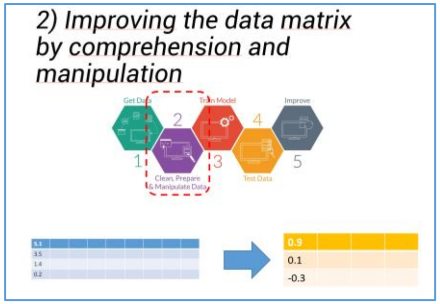
d = 1000

#bins = 10^d

Estimated # of atoms in the universe: **10^{78} - 10^{82}**

How to face the Course of D.?

- We can still find effective techniques applicable to high-dimensional spaces
  - Real data will often be confined to a region of the space having lower effective dimensionality
  - Real data will typically exhibit smoothness properties
- In case you can
  - Feature selection
  - Dimensionality reduction
  - Controlling model complexity



- Many classifiers may be significantly affected by the curse of dimensionality or not.

Know your classifier!

Classical models: kNN

Nearest Neighbor Classifiers

Nearest Neighbor Classifier: A “must-have”. Why?

- It’s a classical classifier not based on neural techniques
- It’s deterministic
  - No random initialization (like NN., EVO. algo, …)
  - Perfect repeatability
- A minimum number of parameters is needed

Nearest Neighbor Classifier: A “must-have”. Why? (2)

- Learning is very simple
- Perfect explainability
  - In term of how is really functioning
  - In term of «why this sample has been classified as bad»
- Present in all libraries

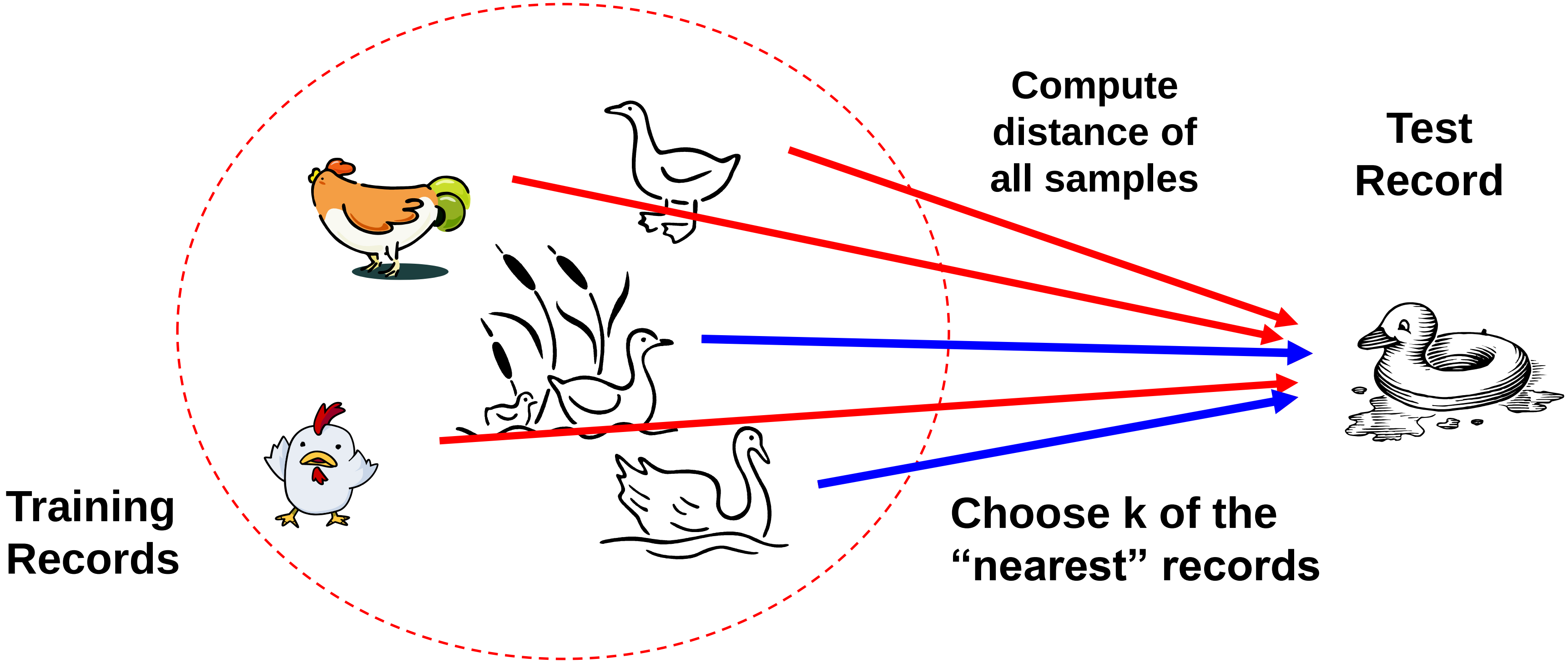
Why study the kNN?

- Because it is very powerful classifier to understand your data
  - if the dataset is not too big… (out-of-memory…)
- You can consider the kNN like a debug tool con control the data and the accuracy of the other classifiers (even neural classifiers)
- Last but not least: it tends to the Bayes optimal classifier….
  - the “god” classifier, the best classifier you can build even knowing exactly the probability distribution of your classes

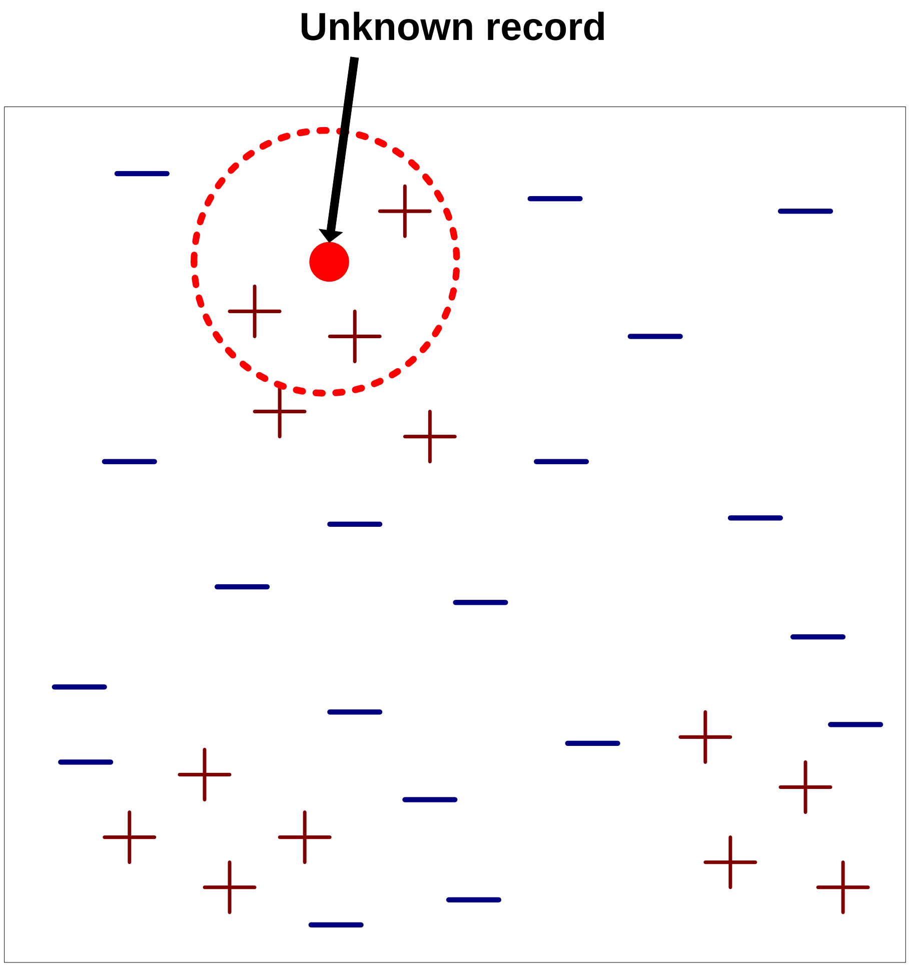
Nearest Neighbor Classifiers

Basic idea:

If it walks like a duck, quacks like a duck, then it's probably a duck



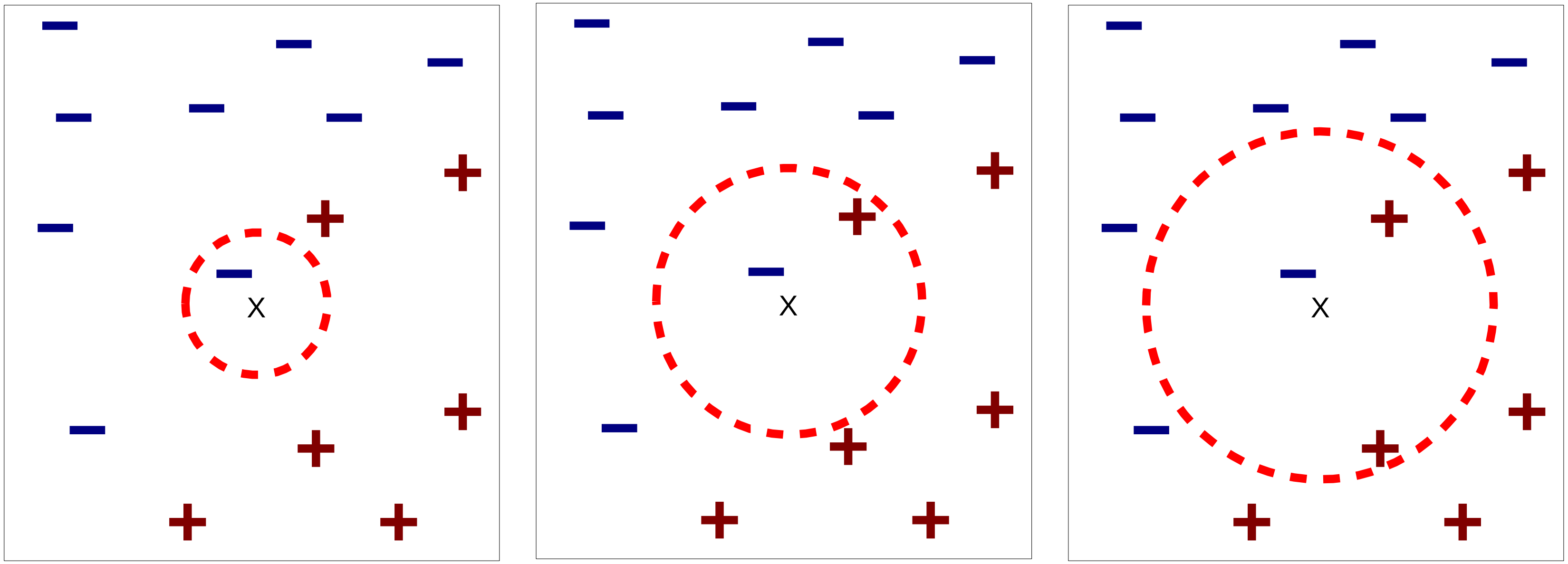
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

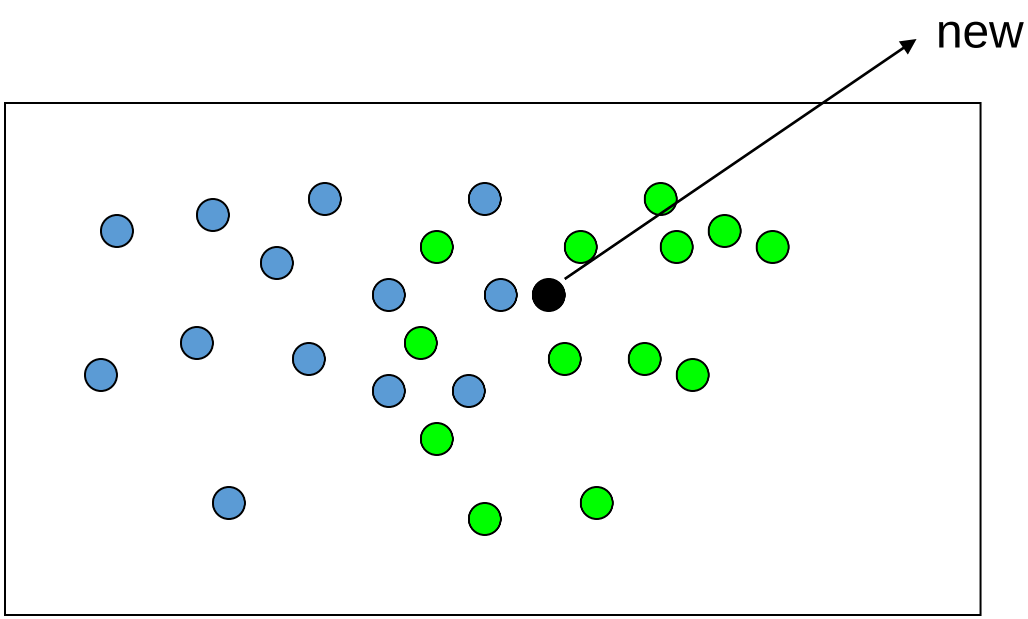


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest neighbor method

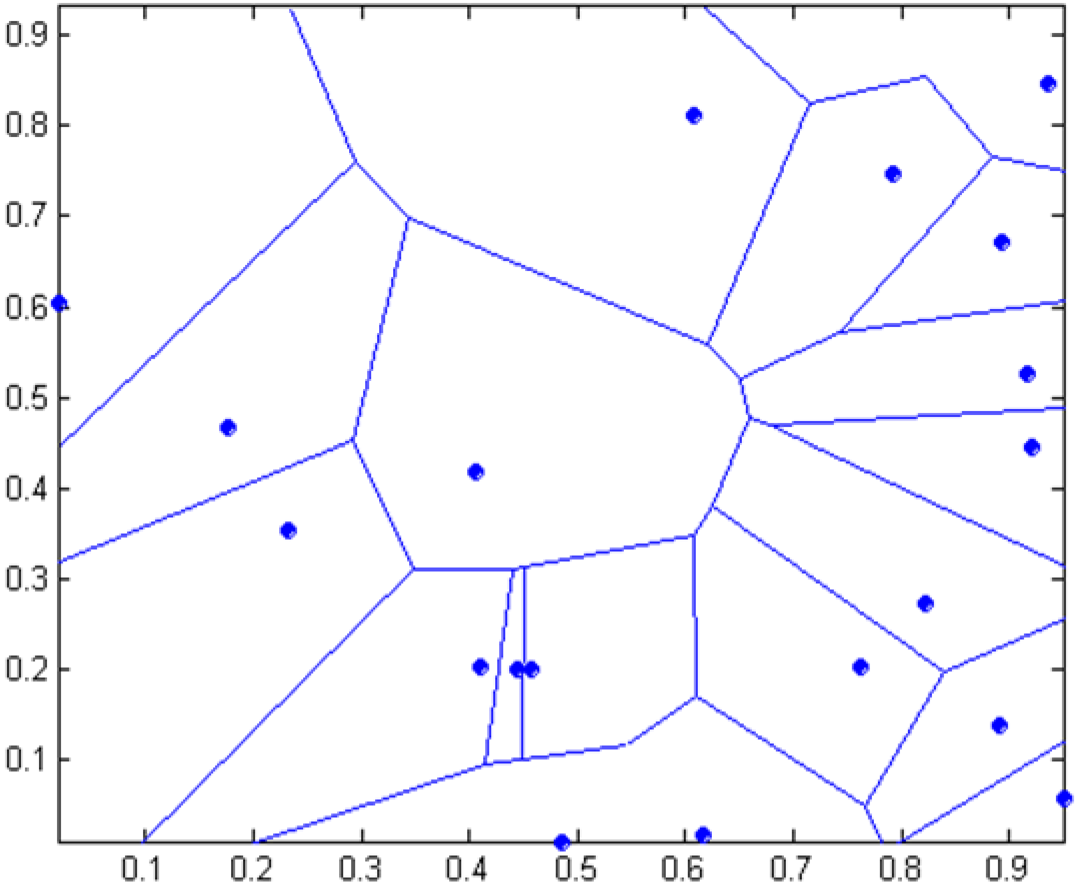
- Majority vote within the k nearest neighbors



k= 1: blue
k= 3: green

The 1 nearest-neighbor decision boundaries

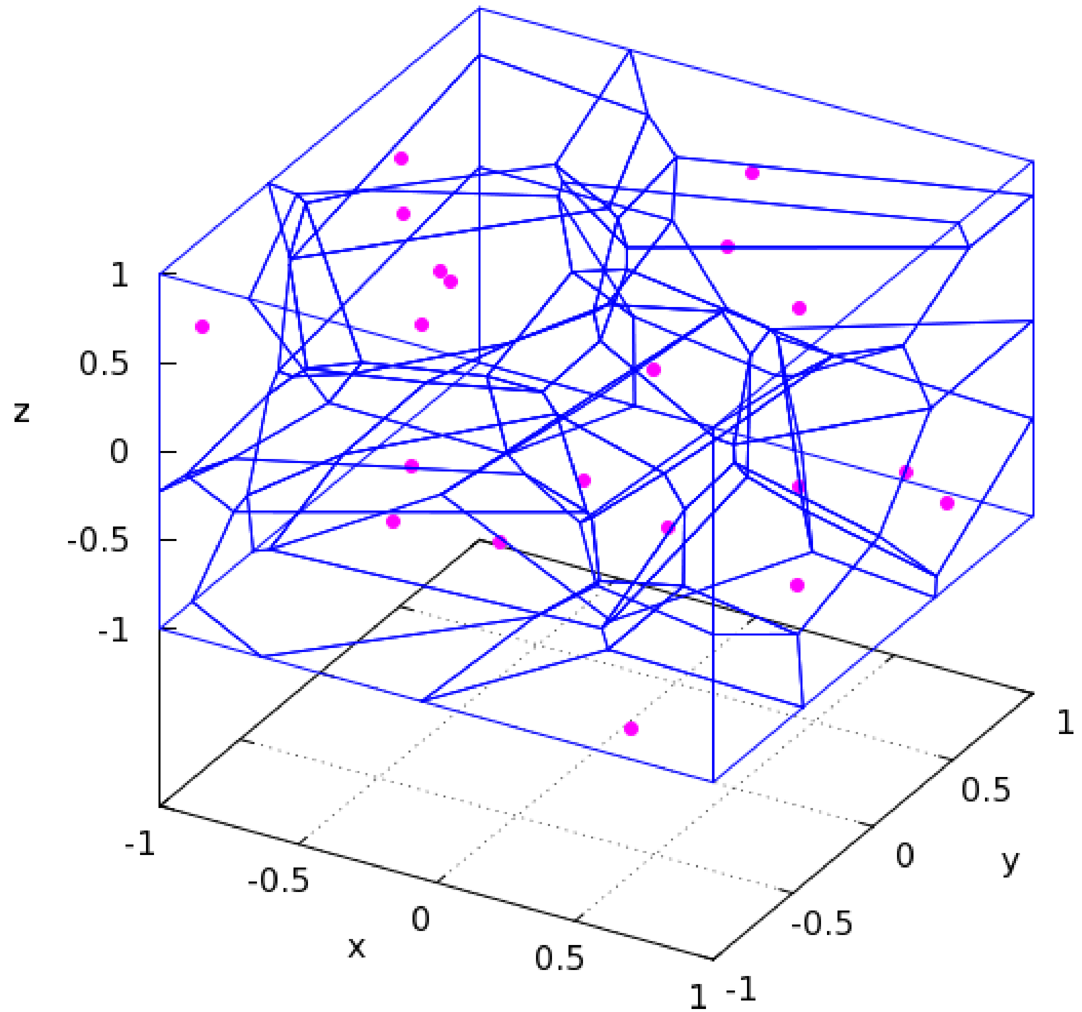
Voronoi Diagram



A partitioning of a plane into regions based on distance to points in a specific subset of the plane

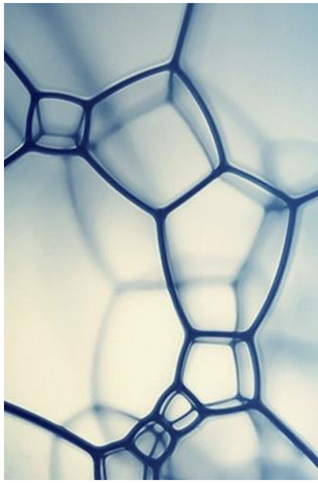
The Voronoi diagram of a set of points is dual to its Delaunay triangulation.

NDimensional Voronois diagram

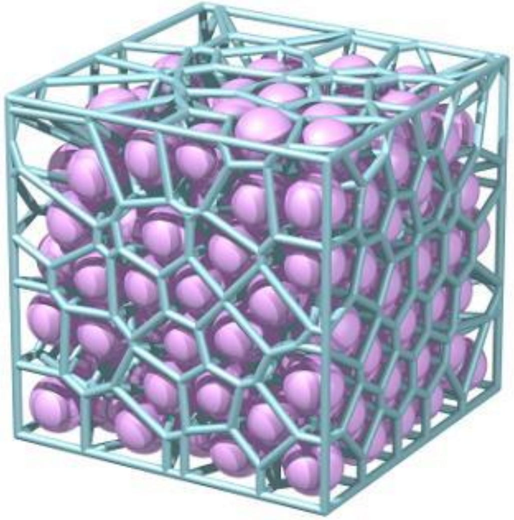


A 3D example

Voronoi usage examples

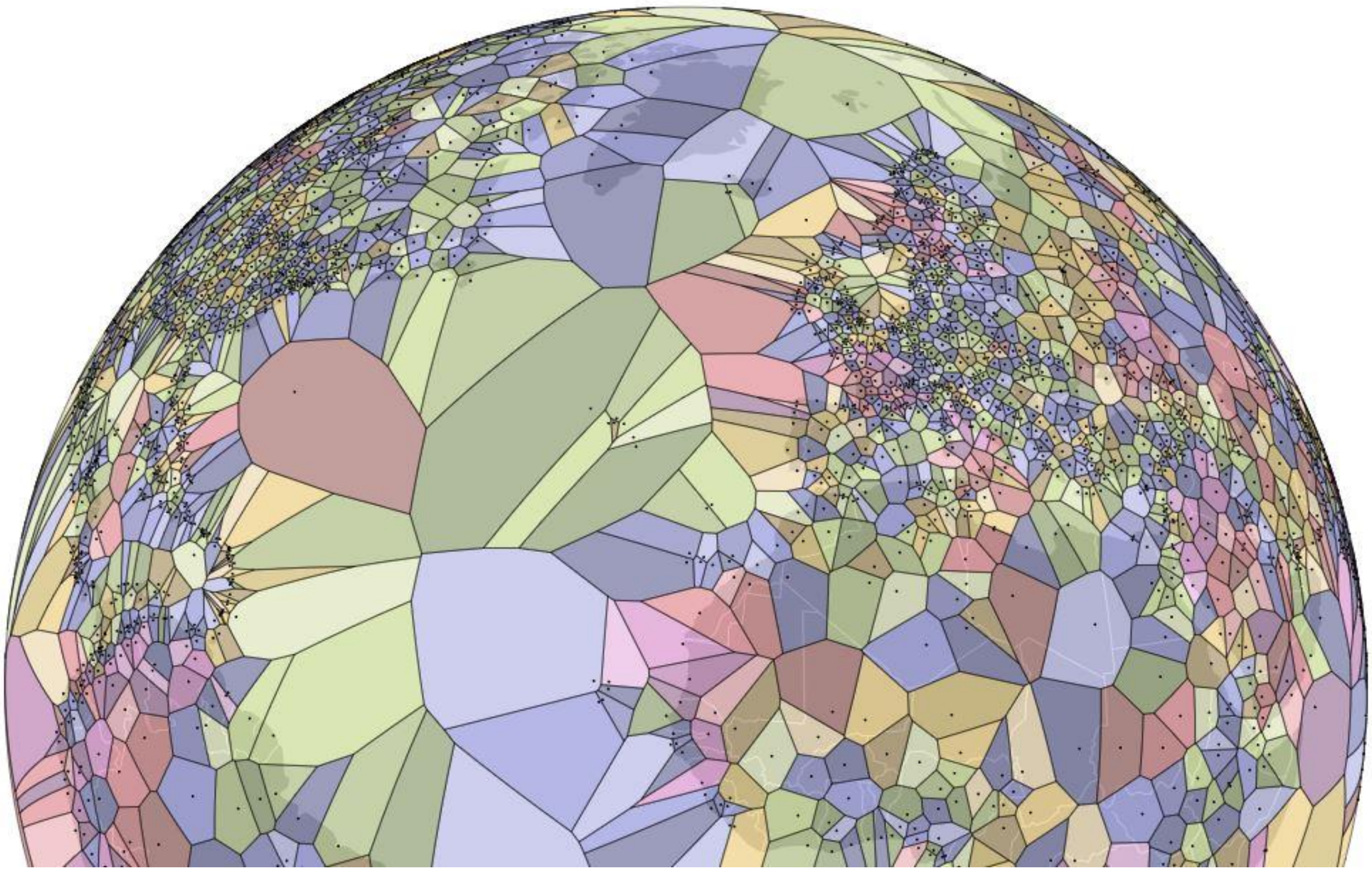


- Natural sciences and biology
  - Voronoi tessellation emerges by radial growth from seeds outward.
- Health
  - Correlate sources of infections in epidemics
- Engineering
  - Free volumes of polymers
- Geometry
- Informatics
- Civics and planning



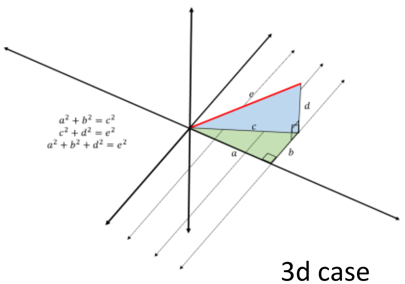
Voronoi diagram usage

A voronoi diagram of the world's airports projected onto an 3D globe (Jason Davies)



Further reading:
An n-dimensional Pythagorean Theorem
William J. Cook
The College Mathematics Journal
Vol. 44, No. 2 (March 2013), pp. 98-101 (4 pages)
Summary: An n-dimensional generalization of the standard cross product leads to an n-dimensional generalization of the Pythagorean theorem

Nearest Neighbor Classification



3d case

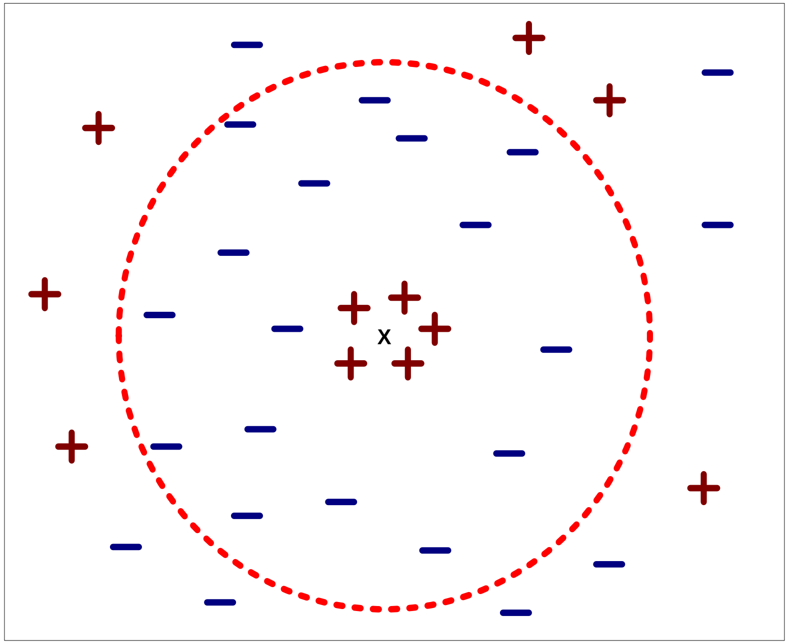
- Compute distance between two points:
  Example: Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

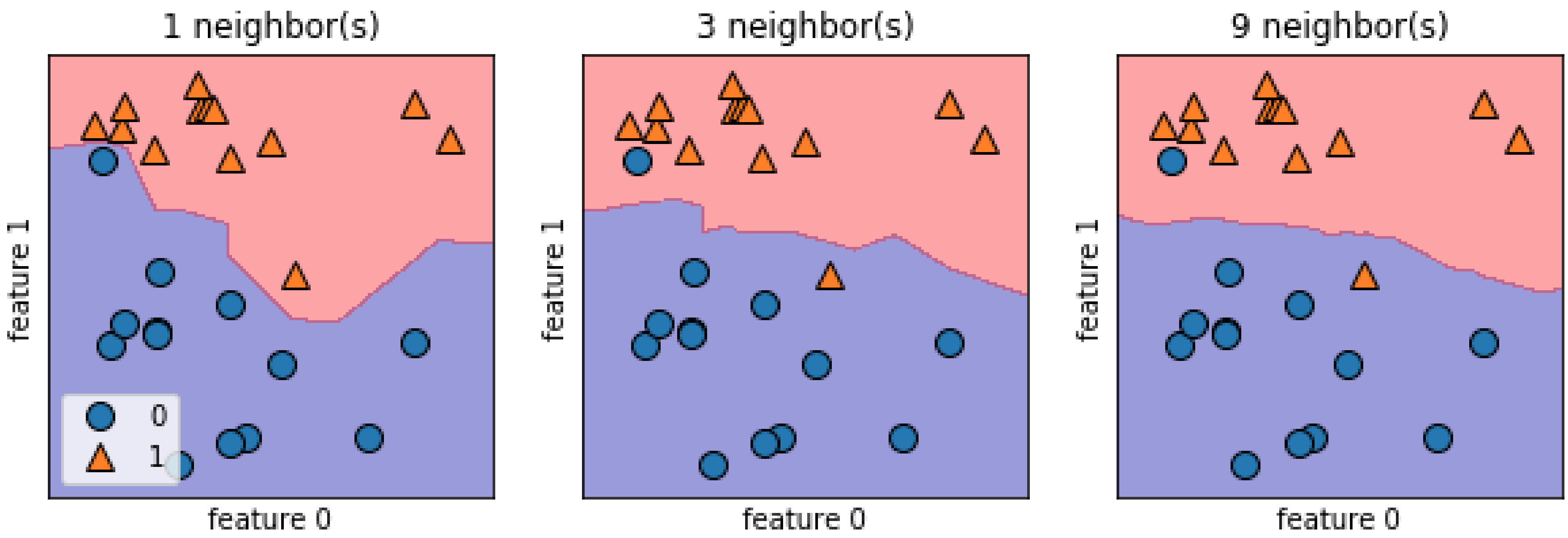
- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Alternative: weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Design of the kNN: choosing the value of k

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes

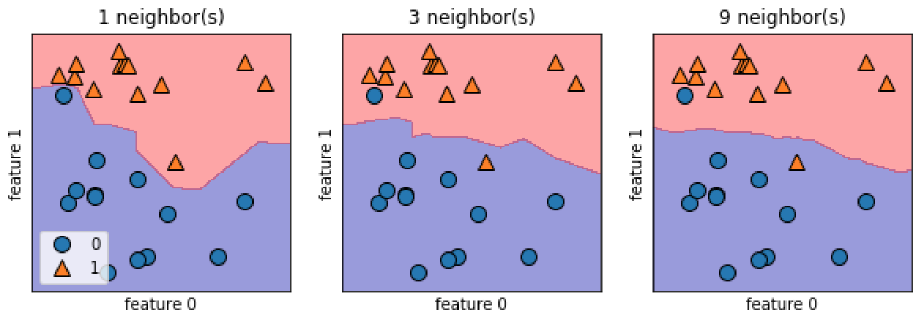


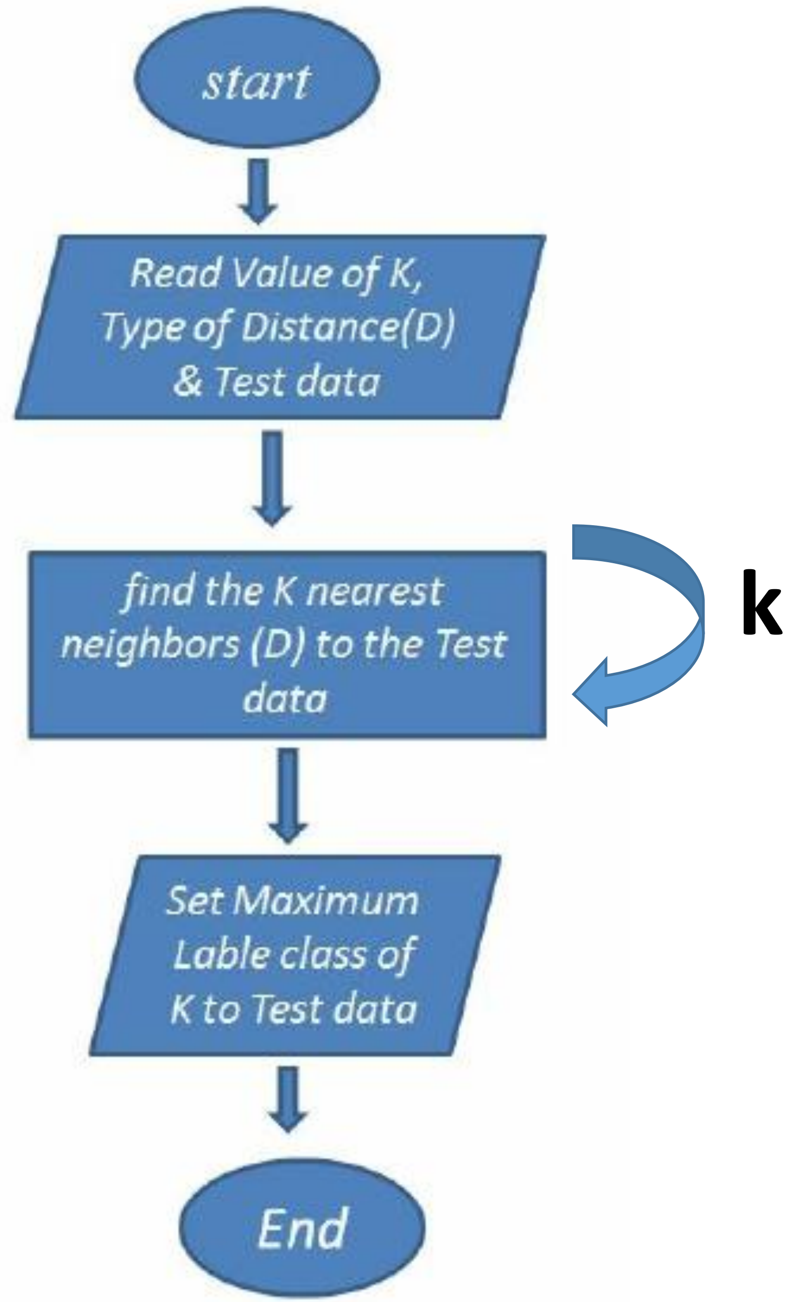
High k → regularization



... but more time is needed to process the stored data

Complexity ←→k





Nearest Neighbor Classification…

- **Scaling issues**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

Discussion on the k-NN Algorithm

(+)

Robust to noisy data by averaging k-nearest neighbors

(-)

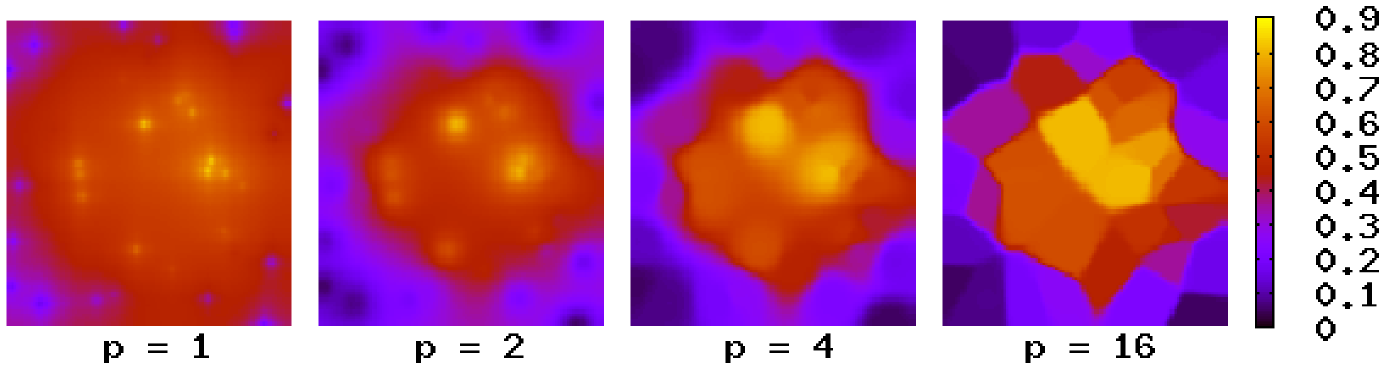
Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.

- To overcome it, axes stretch or elimination of the least relevant attributes (Feature selection)

Distance-Weighted Nearest Neighbor Algorithm

- Assign weights to the neighbors based on their 'distance' from the query point
  - Weight 'may' be inverse square of the distances

➔All training points may influence a particular instance
  - Shepard's method



From smooth partitions → Voronoi

Practical issues when using kNN

SPEED

CURSE OF DIMENSIONALITY

Practical issues when using kNN: speed

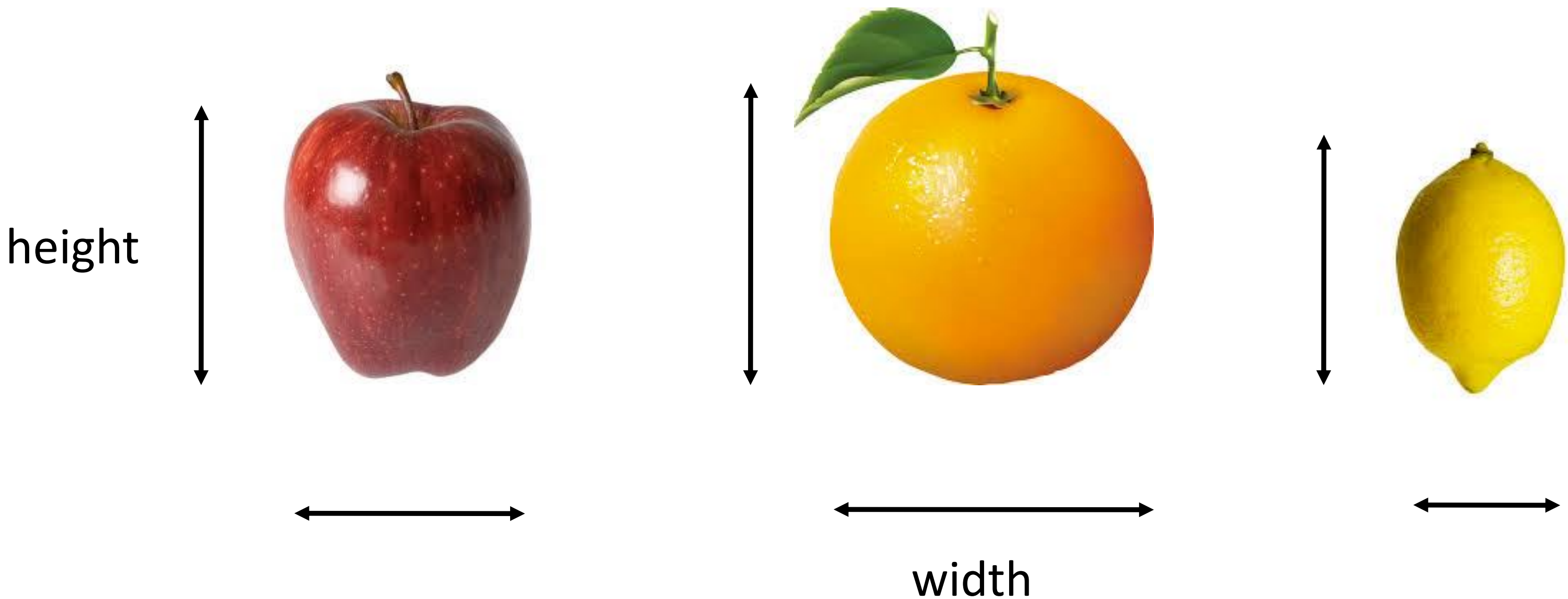
- Speed
  - Time taken by kNN for N points of D dimensions
    - time to compute distances: O(ND)
    - time to find the k nearest neighbor
      - O(k N) : repeated minima
      - O(N log N) : sorting
      - O(N + k log N) : min heap
      - O(N + k log k) : fast median
    - Total time is dominated by distance computation
  - We can be faster if we are willing to sacrifice exactness

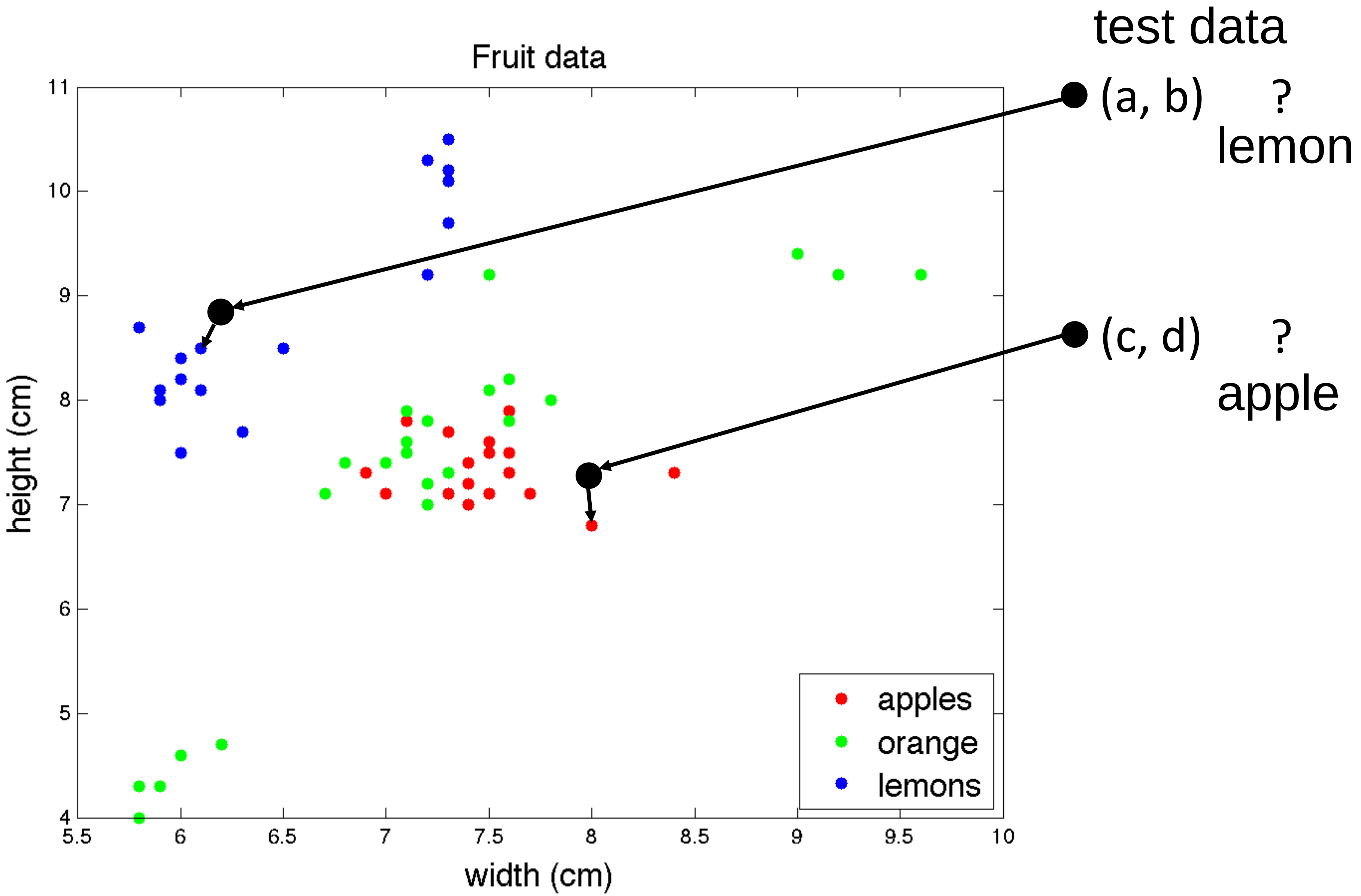



Example of kNN

Nearest neighbor classifier

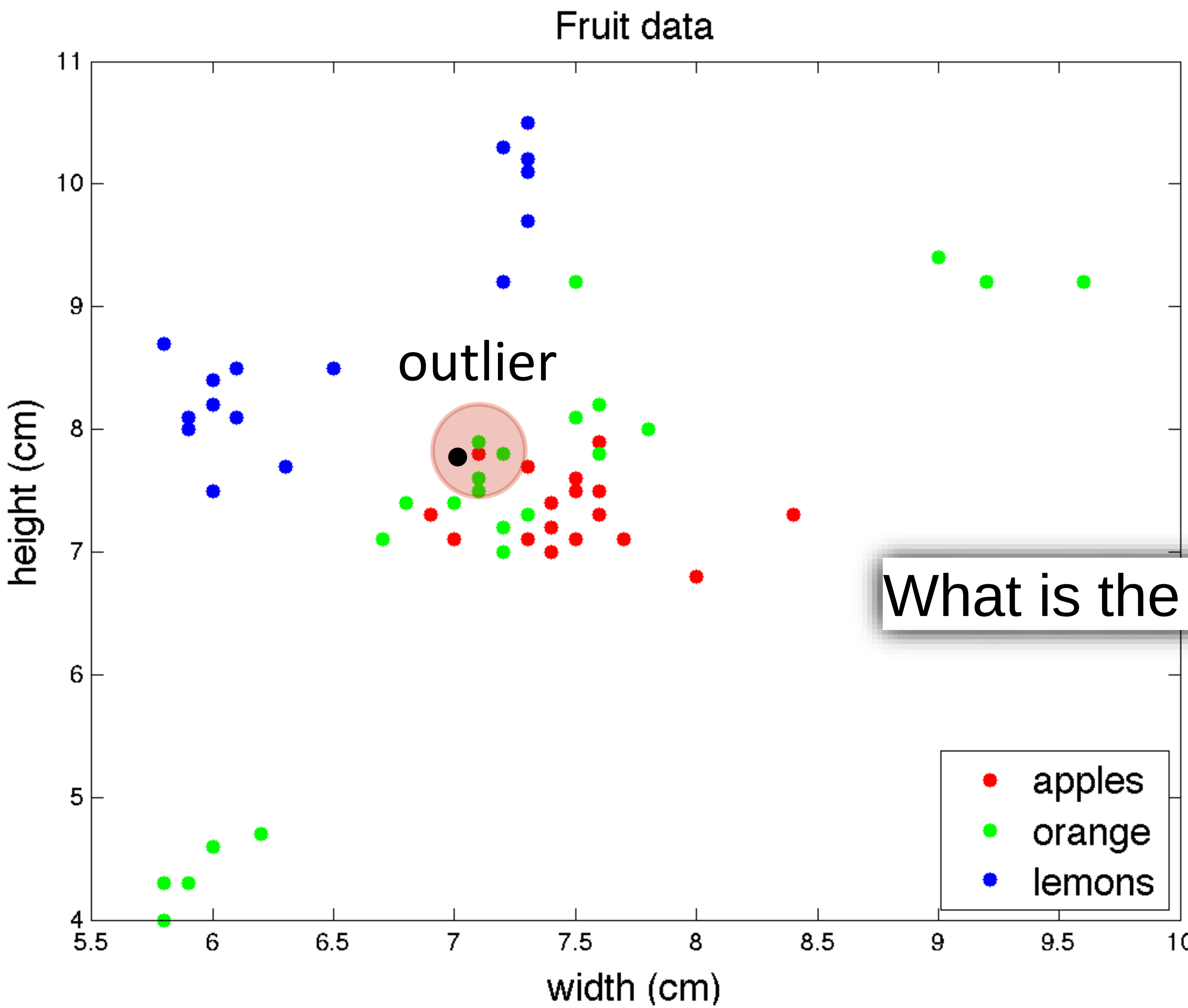
- Training data is in the form of $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- Fruit data:
  - label: {apples, oranges, lemons}
  - attributes: {width, height}




test data
(a, b) ?
lemon
(c, d) ?
apple
Fruit data
height (cm)
width (cm)
apples
orange
lemons

k-Nearest neighbor classifier

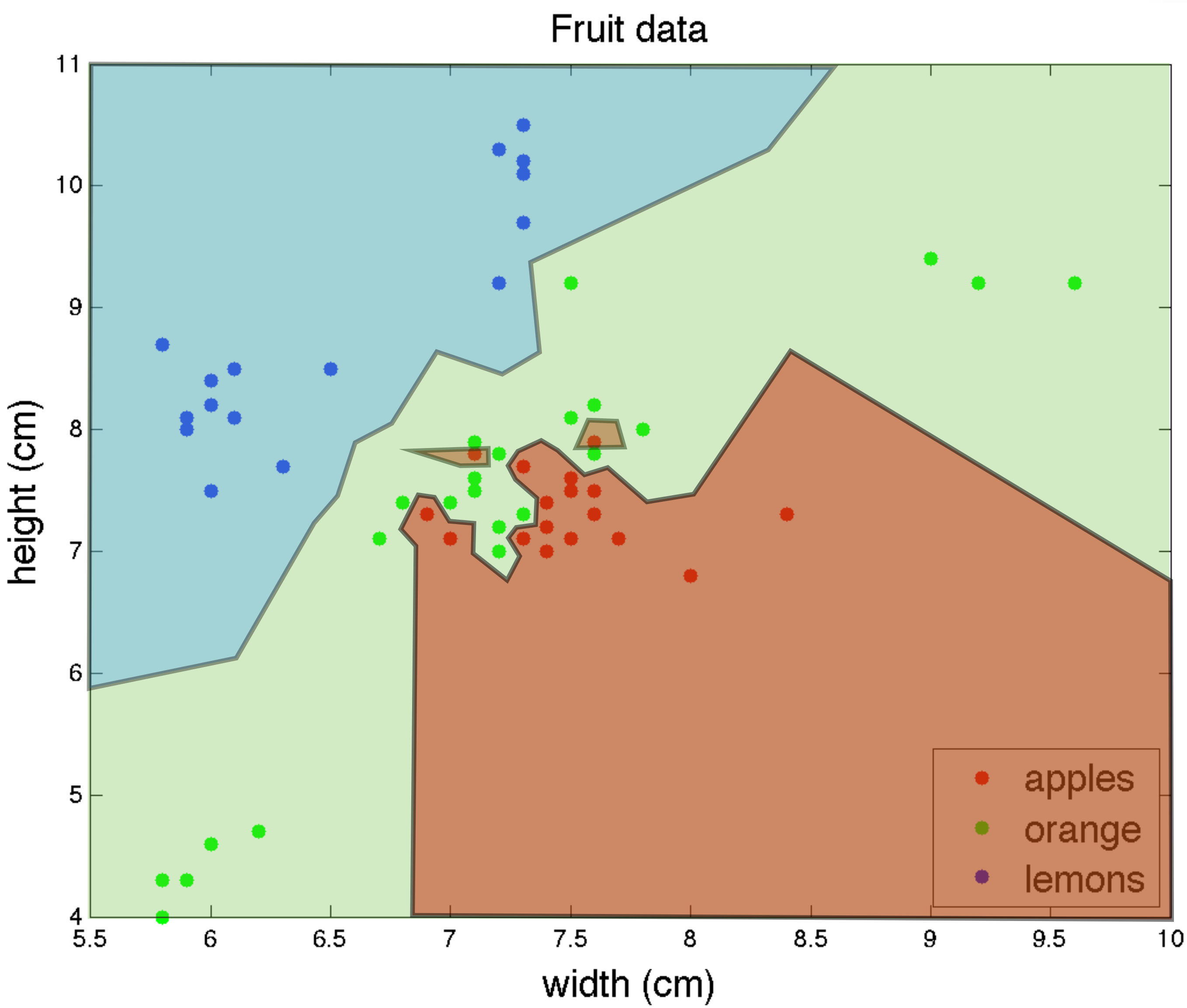
Take majority vote among the k nearest neighbors



What is the effect of k?

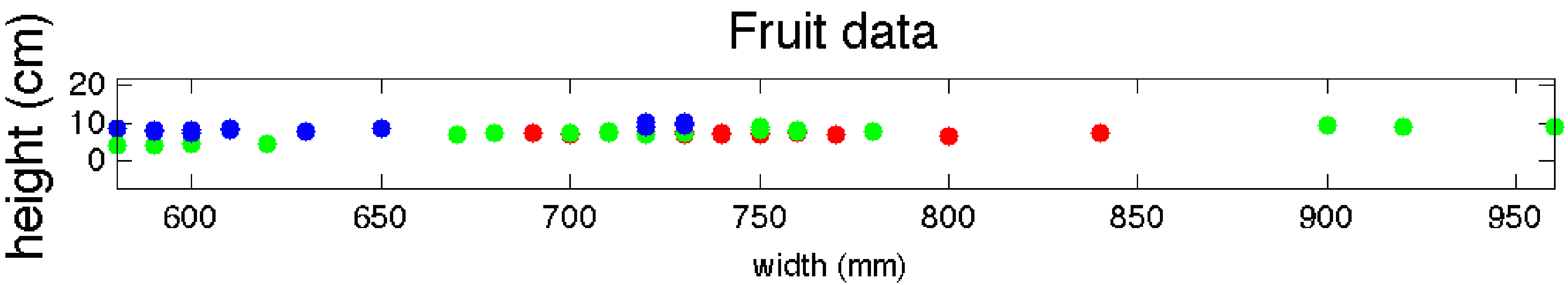
Decision boundaries: 1NN

What is the effect of k?



Inductive bias of the kNN classifier

- Choice of features
  - We are assuming that all features are equally important
  - What happens if we scale one of the features by a factor of 100?
- Choice of distance function
  - Euclidean, cosine similarity (angle), Gaussian, etc …
  - Should the coordinates be independent?
- Choice of k

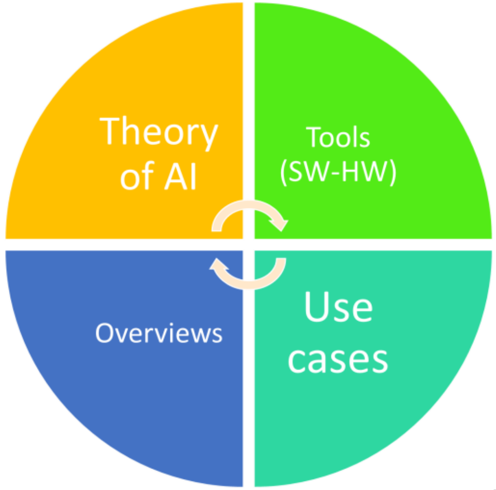





Toolboxes

Coding kNN

Matlab



kNN Coding in Matlab

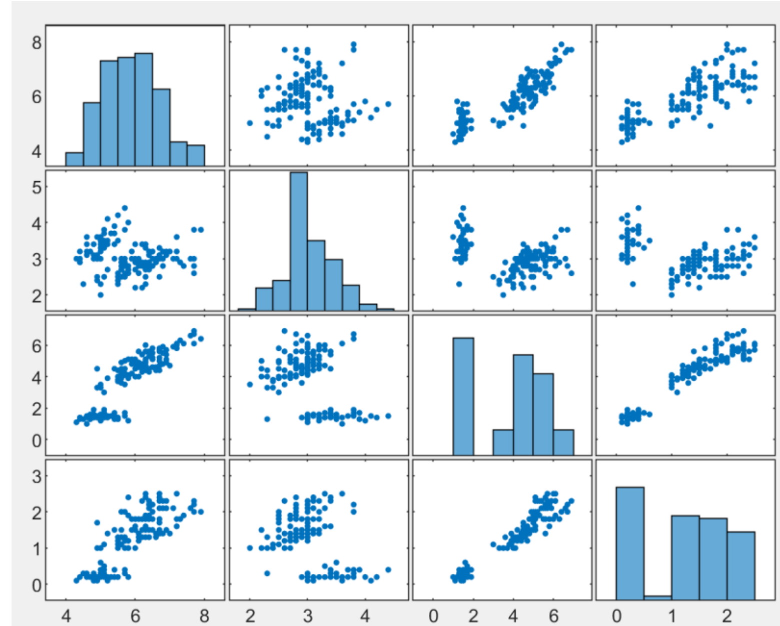
laboratory_MATLAB_Knn.m

```
load fisheriris
X = meas;
Y = species;
% X is a numeric matrix that contains
% four petal measurements for 150 irises.
% Y is a cell array of character vectors
% that contains the corresponding iris species.

% just a simple plot
plotmatrix(X)

% Train a 5-nearest neighbor classifier.
% Standardize the noncategorical predictor data
% --> see lesson about encoding the outputs
kNN_model = fitcknn(X,Y,'NumNeighbors',5,'Standardize’,1)

% Let's try to input a single vector
x = X(1,:)
label = predict(kNN_model,x)
```




```
kNN_model =

  ClassificationKNN
           ResponseName: 'Y'
  CategoricalPredictors: []
             ClassNames: {'setosa'  'versicolor'  'virginica'}
         ScoreTransform: 'none'
        NumObservations: 150
               Distance: 'euclidean'
           NumNeighbors: 5

x =

    5.1000    3.5000    1.4000    0.2000

label =

  1×1 cell array

    {'setosa'}
```

kNN Coding in Matlab

```
% Do a cross-validation test
% The function will create SUBCLASSIFIERS to do a correct CrossValidation
cvmdl_results = crossval(kNN_model, 'KFold',10)
kfoldLoss(cvmdl_results )
```

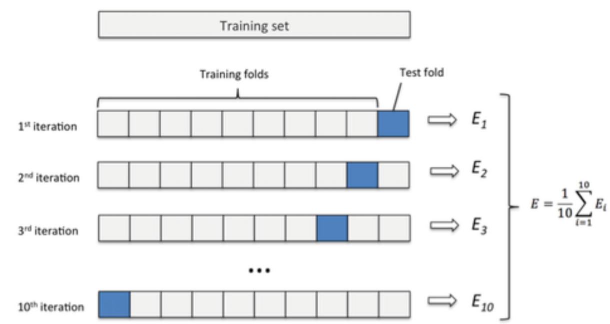
```
ans =

     0.0533
```

```
cvmdl_results =

  classreg.learning.partition.ClassificationPartitionedModel
    CrossValidatedModel: 'KNN'
         PredictorNames: {'x1'  'x2'  'x3'  'x4'}
           ResponseName: 'Y'
        NumObservations: 150
                  KFold: 10
              Partition: [1×1 cvpartition]
             ClassNames: {'setosa'  'versicolor'  'virginica'}
         ScoreTransform: 'none'
```

What happens if we change the validation folders?




```
% Let's do a cross-validation test with different folder
from 10 to 3?

% K = 10
fprintf('KFOLD Validation with K = %d --> Error = %f \n', 10, kfoldLoss(cvmdl_results));
% K = 9...3
for K = [9:-1:3]
   cvmdl_results = crossval(kNN_model, 'KFold',K);
   fprintf('KFOLD Validation with K = %d --> Error = %f \n', K, kfoldLoss(cvmdl_results));
end
```

Warning:
That is the K of the CV
Not the k = 5 of the kNN!

```
KFOLD Validation with K = 10 --> Error = 0.053333
KFOLD Validation with K = 9 --> Error = 0.046667
KFOLD Validation with K = 8 --> Error = 0.053333
KFOLD Validation with K = 7 --> Error = 0.046667
KFOLD Validation with K = 6 --> Error = 0.053333
KFOLD Validation with K = 5 --> Error = 0.060000
KFOLD Validation with K = 4 --> Error = 0.046667
KFOLD Validation with K = 3 --> Error = 0.046667
```

Less information in the training (in general) → accuracy is getting worse, but noise is masking this effect here

Main points

- Bayes Optimal Classification
- Pros and cons of Eager and Lazy Learning Methods
- Course of dimensionality
- Nearest Neighbor Classifiers (kNN)
  - Relevance of the kNN in pattern recognition
    - Approximator of the optimal Bays classifier
  - Definition and coding
  - Problems
    - Speed
- The KNN as “debug tool”
- The concept that classification is a set of boundaries in the feature space (e.g. Voronoi)