ML classification: regression–based

e.g. Discriminant Functions, Logic Regression

calculates feature weights to separate input classes

methods vary in assumptions

- data types: binary, discrete, continuous
- data distributions: binomial, Gaussian, Poisson, etc.
- data inclusion: all, some (‘good’ only), some (random)

Ensembles of functions are (usually) more effective

Eigenvalues may illuminate data properties

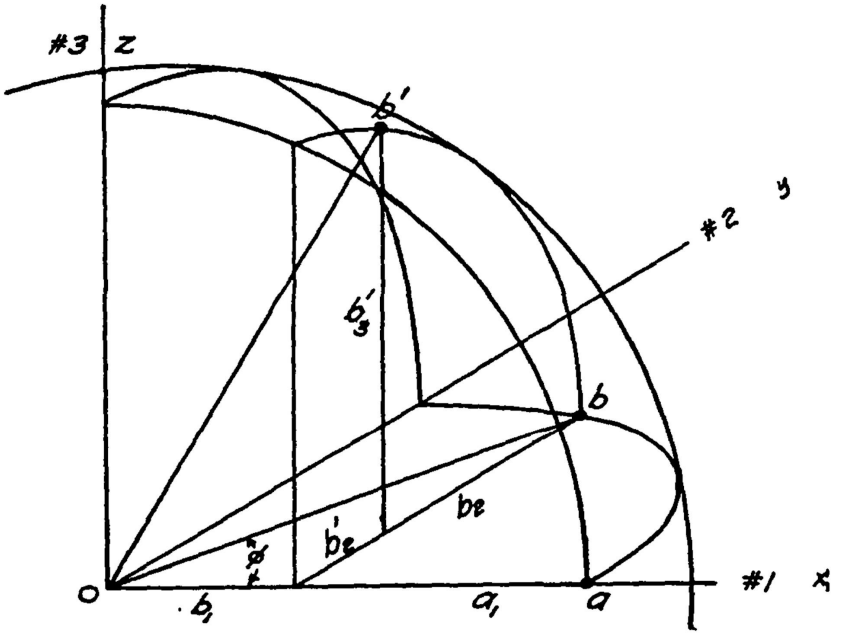


FIG. 1. This diagram shows the relation between the correlation coefficient and the central angle between the two tests or points. Let two points which represent any two tests be designated a and b'. Let the x-axis, representing the first general factor, pass through the point a. Then its coördinates are (1, 0, 0). Therefore $a_1 = 1$ while a_2 and a_3 are both zero. Revolve the sphere about the x-axis so that the point b' is in the horizontal xy-plane at b. Then the z-coördinate of the point b is evidently zero while $b_1^2 + b_2^2 = 1$. Let the angle aob be designated ϕ. Then, clearly, $\cos \phi = b_1$ since the radius of the sphere is unity. The correlation between tests a and b can be written as follows.

$$r_{ab} = a_1 b_1 + a_2 b_2 + a_3 b_3.$$

Since a_2 and a_3 are both zero, the second and third terms vanish and since $a_1 = 1$, it follows that $r_{ab} = b_1$. The angle aob is equal to the angle aob'. Hence the correlation coefficient is equal to the cosine of the central angle between the points that represent the two tests.

(Thurstone 1931)

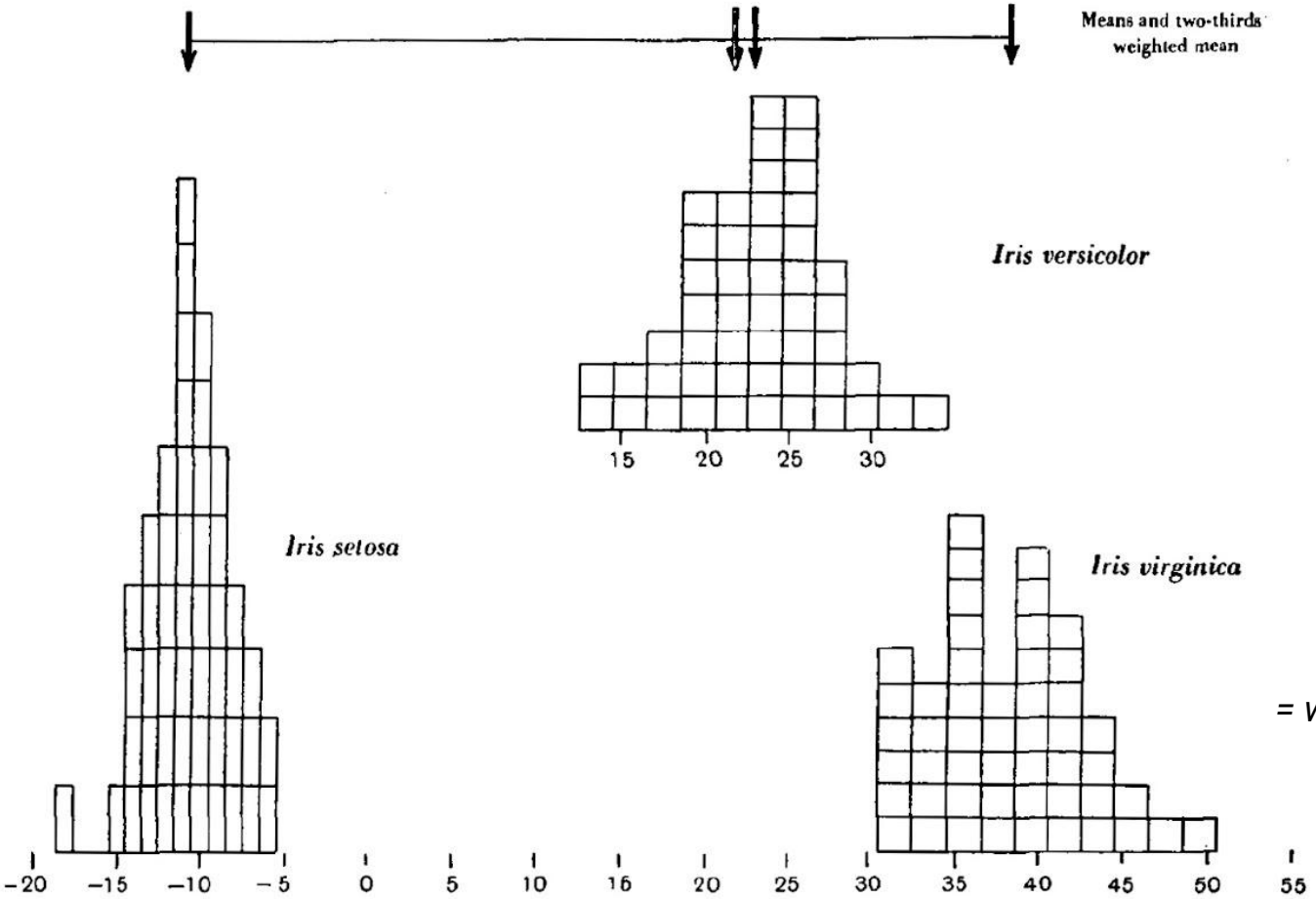


Fig. 1. Frequency histograms of the discriminating linear function, for three species of *Iris*.

(Fisher 1936; https://doi.org/10.1111/j.1469-1809.1936.tb02137.x)

$= w + 5.9037x - 7.1299y - 10.1036z$

ML classification: Naïve Bayes

naïve = assumes independent features

not (usually) a true Bayesian method

$= p(w) \cdot p(x) \cdot p(y) \cdot p(z)$

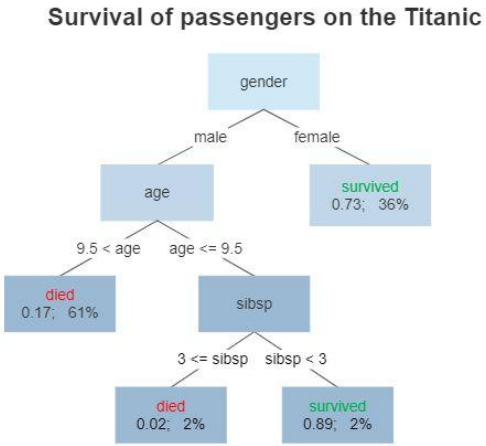
calculates a probability for each feature combination

class with highest probability is output

extremely sensitive to exemplar class frequency distributions used: Gaussian, multinomial, Bernoulli

with discrete data: equivalent to logistic regression

ML classification: Classification And Regression Tree (CART)



discrete data: classification trees

continuous data: regression trees

or a mixture of data types

hierarchic partitions of features to reduce data conflict

NP-complete problem

methods vary in data inclusion: all, random, ‘good’ only

- ensembles of trees are (usually) more effective
- order or frequency of splits may illuminate data properties

https://commons.wikimedia.org/wiki/File:Decision_Tree.jpg

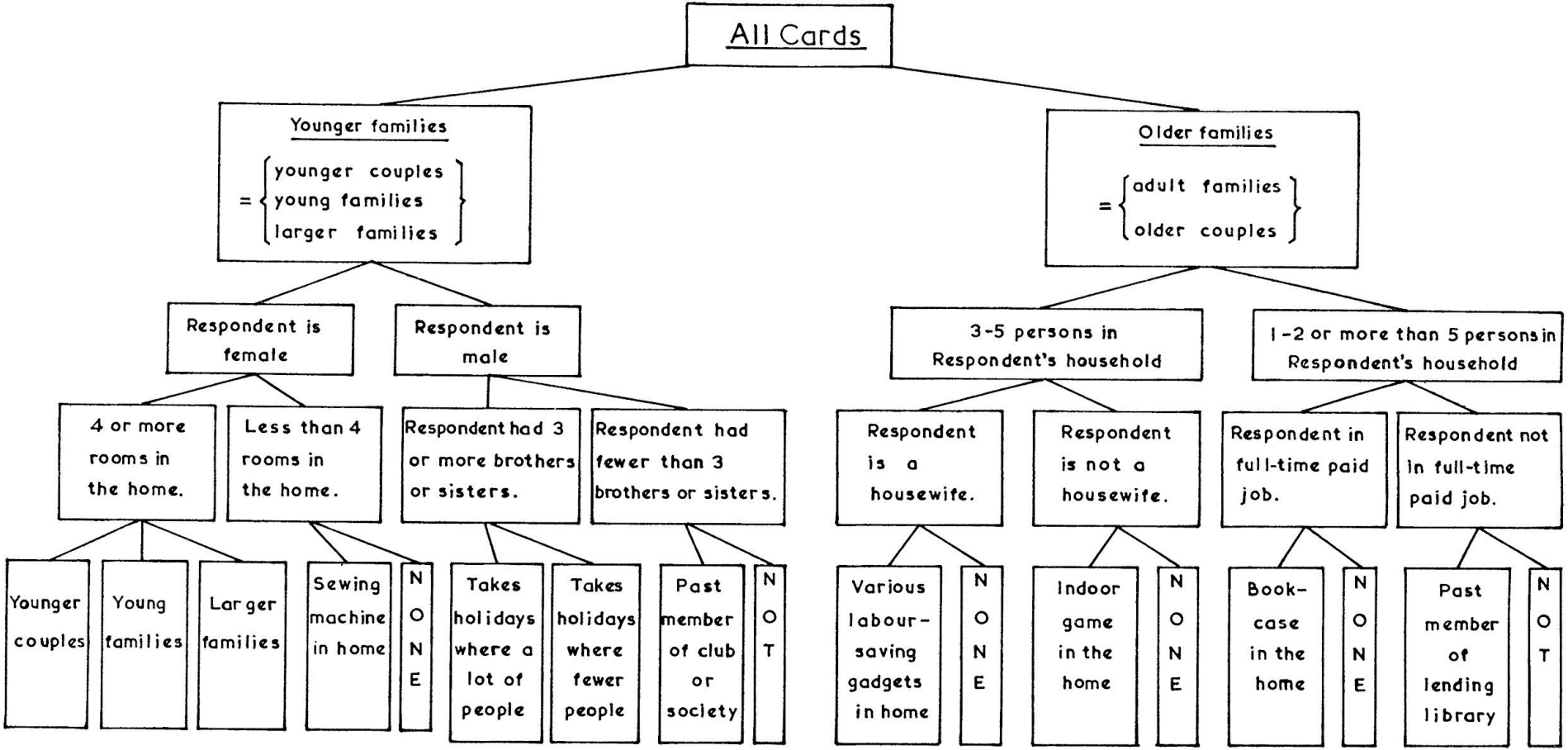
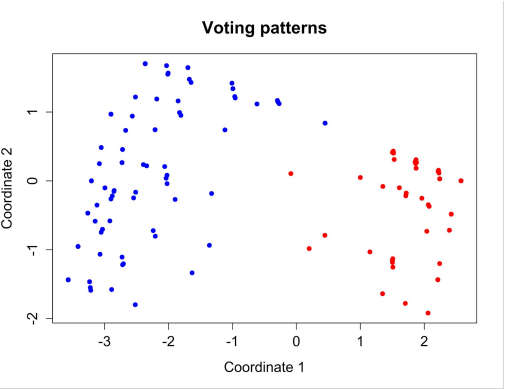


FIG. 2. The derivation of a predictive or matching composite for a study concerning 'Joint Activity' in the home. (To save overcrowding in the bottom row direct negatives of certain classes have been indicated by 'NOT' or 'NONE'.)

(Belson 1959; https://doi.org/10.2307/2985543)

ML classification: clustering algorithms



e.g. MDS, PCA, PCO, SVD, factor analysis

an ‘unofficial’ application that sometimes works

partitions input features into *n* reduced–conflict hyperplanes

methods vary in assumptions

- data types: binary, discrete, continuous
- data distributions: binomial, Gaussian, Poisson, etc.
- data inclusion: all, some (presence only)

eigenvalues may illuminate data properties

https://commons.wikimedia.org/wiki/File:RecentVotes.svg

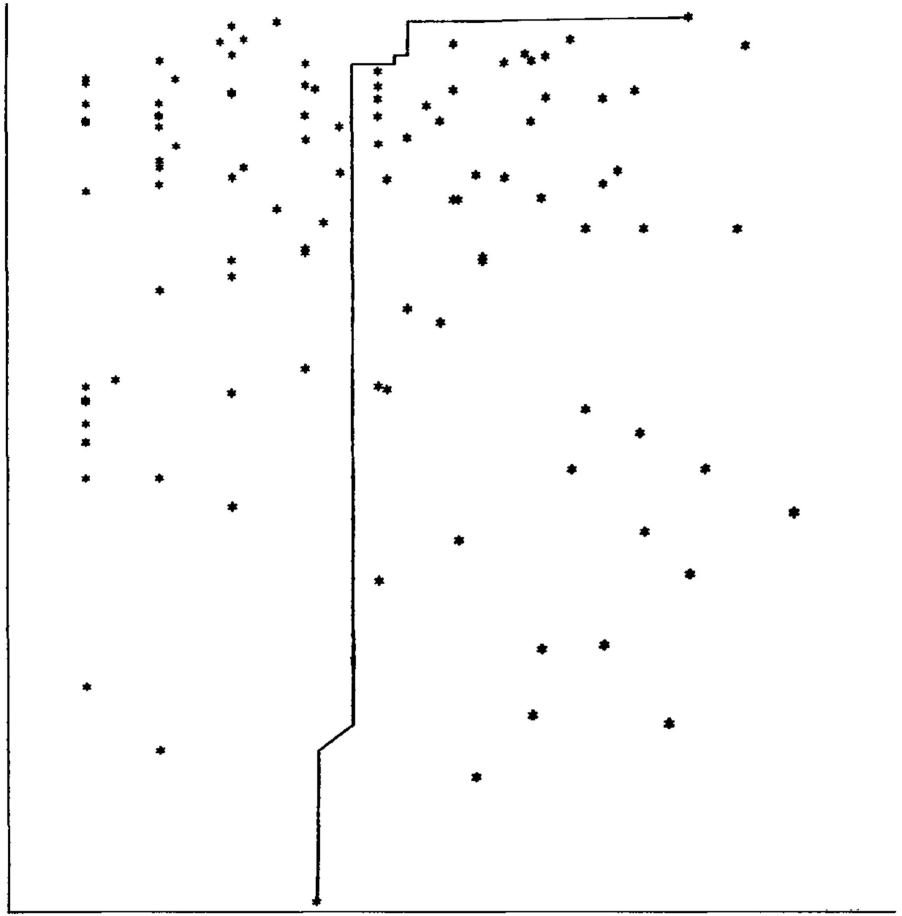


FIGURE 8
Initial Scatter Diagram (Coombs and Kao Data)

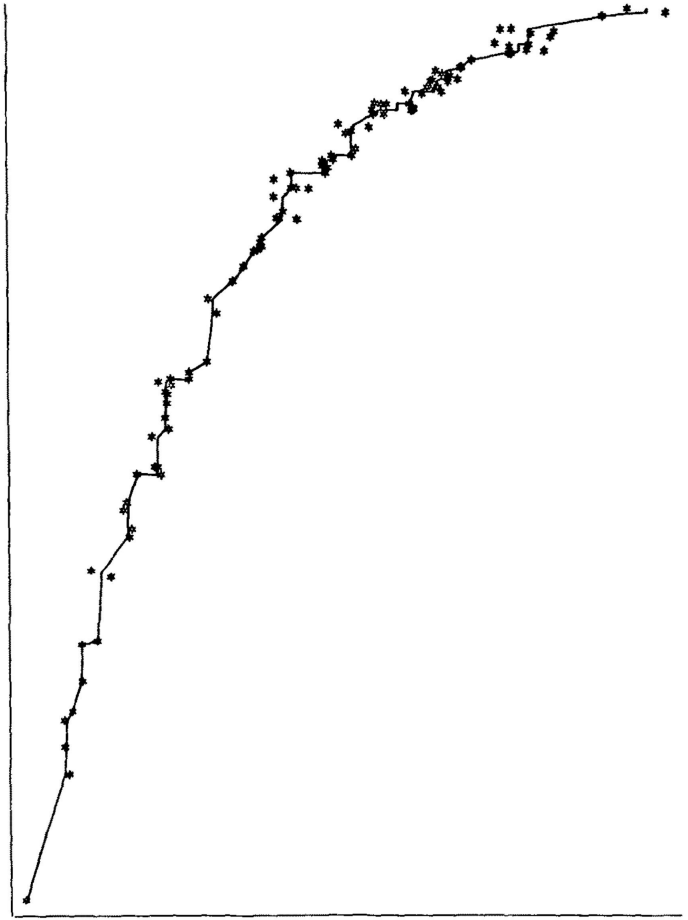
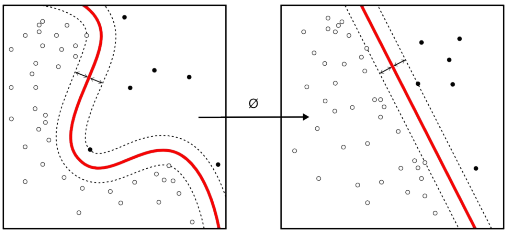


FIGURE 9
Scatter Diagram After 10 Iterations (Coombs and Kao Data)

(Kruskal 1964; https://doi.org/10.1007/BF02289565)

ML classification: Support Vector Machines (SVM)



for classification (or regression)

projects input data into *n*-dimensional space

finds a hyperplane (space) that best separates classes

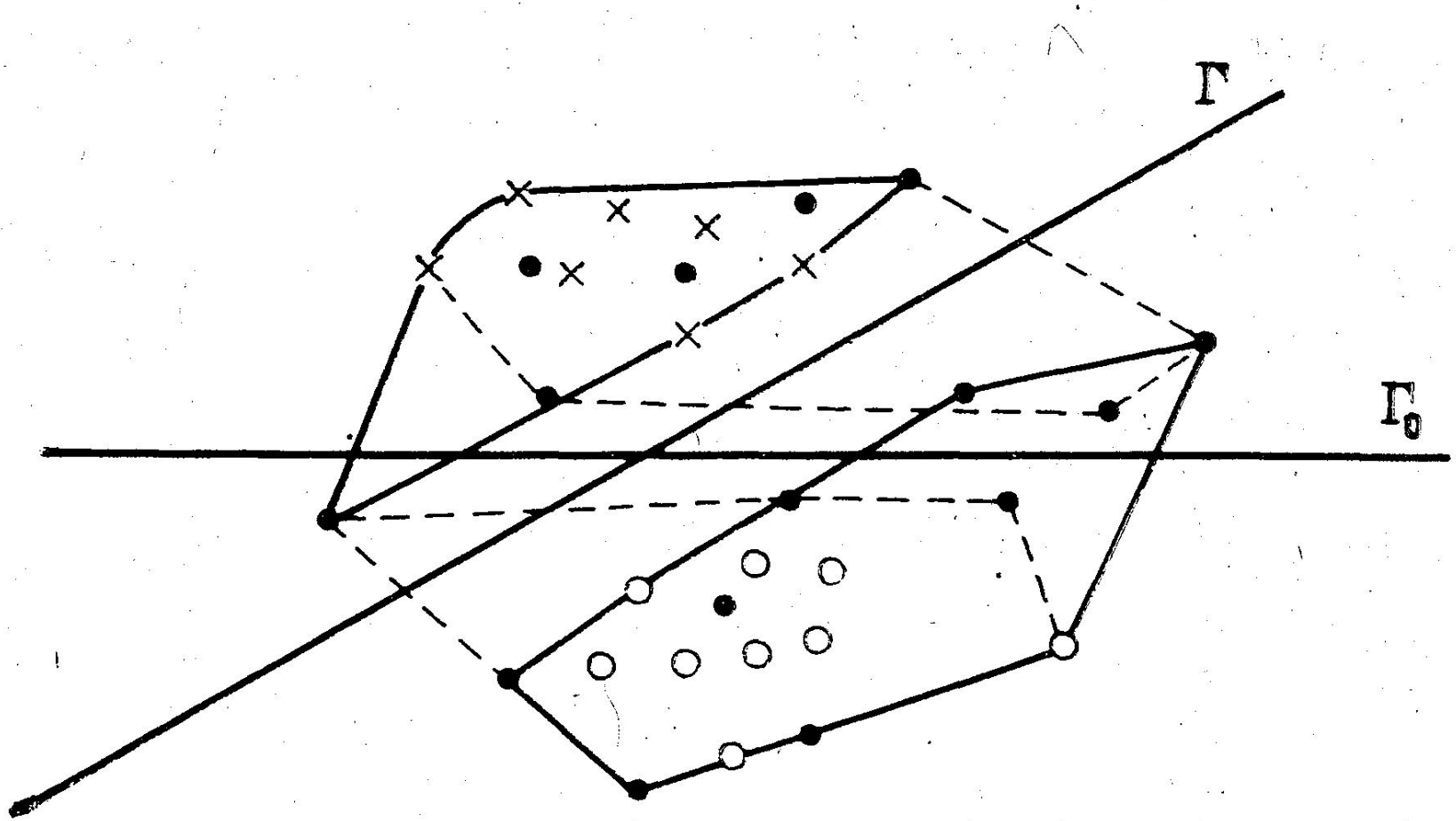
training/validation/testing data required for hyperplane convergence

multiclass are broken into multiple binary problems

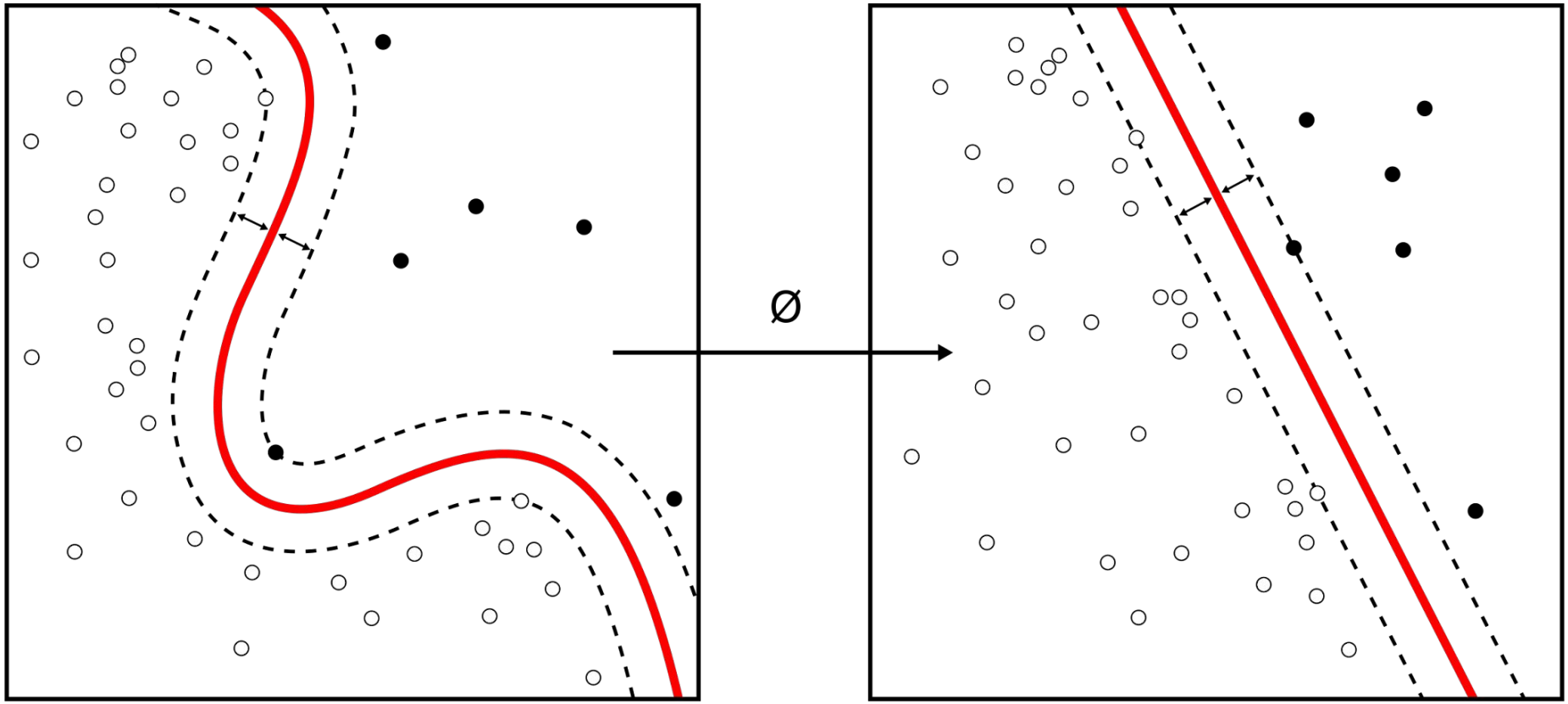
methods vary in assumptions

- data types: discrete (nonlinear), continuous (linear)
- data distributions: homogeneous, heterogeneous

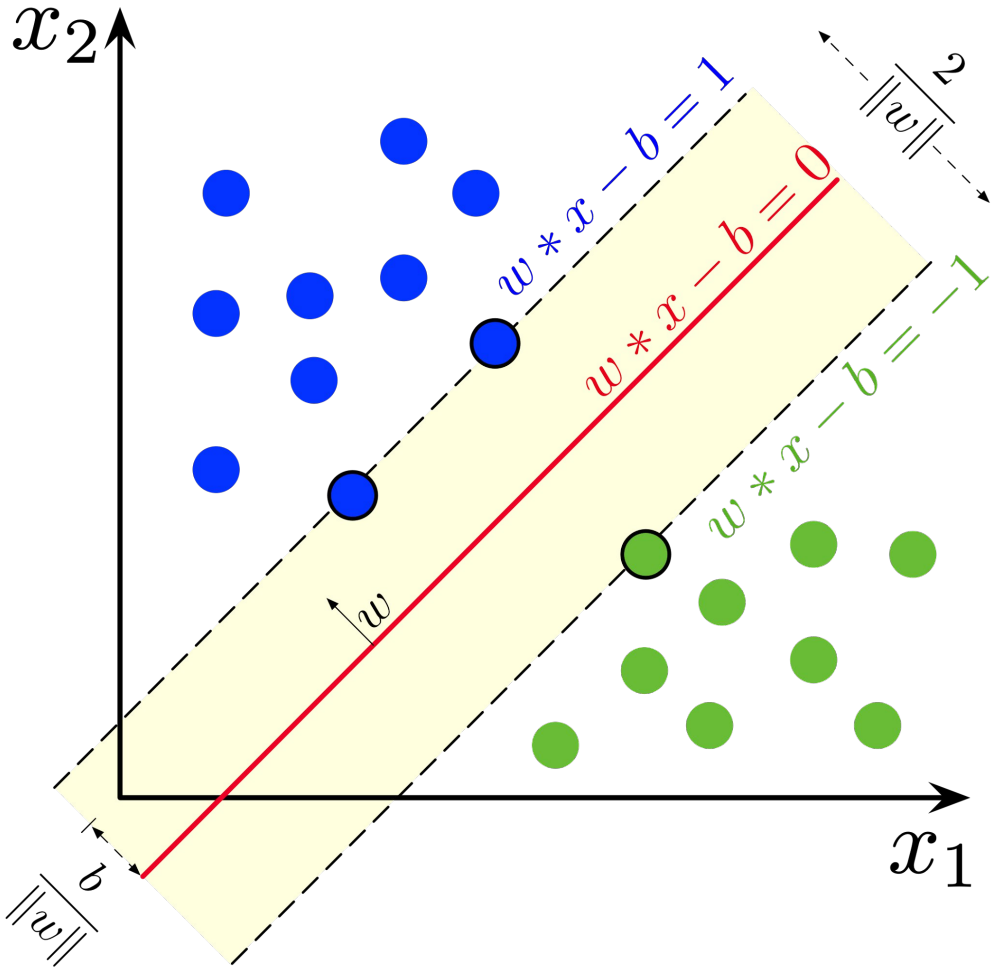
https://commons.wikimedia.org/wiki/File:Kernel_Machine.png



(Vapnik and Chervonenkis 1974)

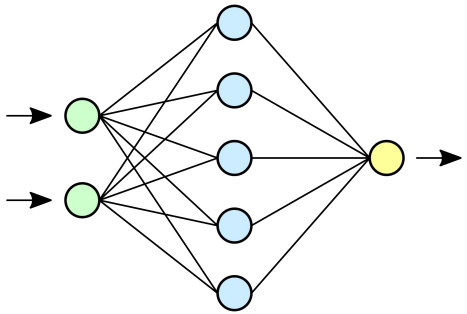


https://commons.wikimedia.org/wiki/File:Kernel_Machine.svg



https://commons.wikimedia.org/w/index.php?curid=73710028

ML classification: Artificial Neural Networks (ANN)



combines many ‘neurons’ into an interconnected web

each ‘neuron’ partitions data into a conflict–‘free’ hyperplane

each ‘neuron’ can be weighted by its reliability

output is rescaled for classification or regression

training/validation/testing data required for weight optimization

ensembles of networks are (usually) more effective

https://commons.wikimedia.org/wiki/File:Neural_network.svg

ML: general assumptions

input classes are actually distinct (and distinguishable)

missing features do not (usually) exist

correctly partitioned exemplars (training/validation/testing)

- otherwise overfitting may occur

exemplars are (usually) independent

- non–independence must be explicitly accounted for

training/validation/testing input order (usually) matters

ML: assumptions about bias

computers (and ML) do exactly what you tell them to do

- garbage in, garbage out

output predictions reflect input (training/validation/testing) data

- if the input data are biased, the output will be too
- more pronounced for some methods
  - e.g. naïve Bayes classifiers

enumerate and quantify the input biases

- mitigate with weights and/or data cleaning
  - input bias assumed to be constant

ML: it’s just regression

most (all?) ML is a (special) type of linear regression

linear regression fails miserably if the data are not linear

- transform to linear using a constant non–linear function
  - e.g. arcsine, log, etc.
- pipe many linear regressions + transformations => ANN
  - represented as a matrix of regression terms
    - (usually) manipulated using matrix algebra

HERBARIUM OF THE
UNIVERSITY OF MINNESOTA
MINNEAPOLIS

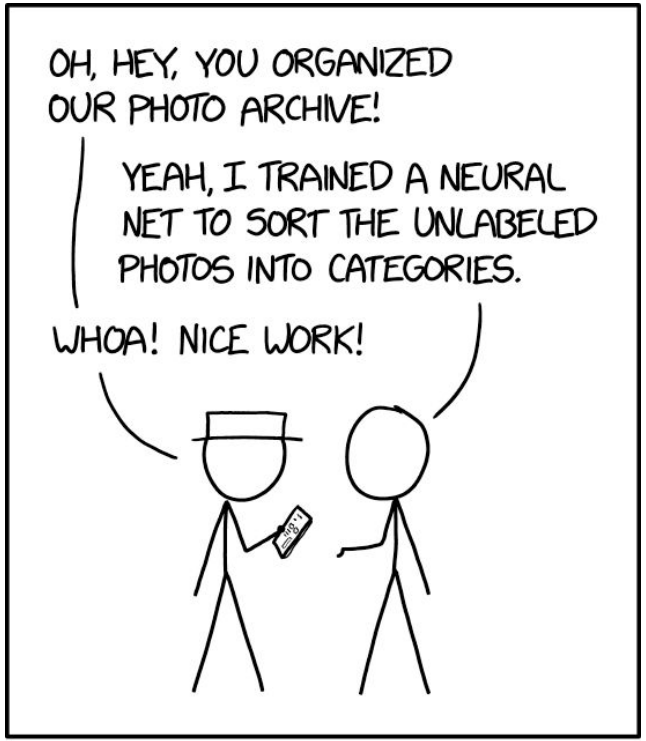
Plants of Minnesota Point

1445. Acer spicatum Lam.

Among A. rubrum on St. Louis Bay side. Sec. 12, R14W, T 49.

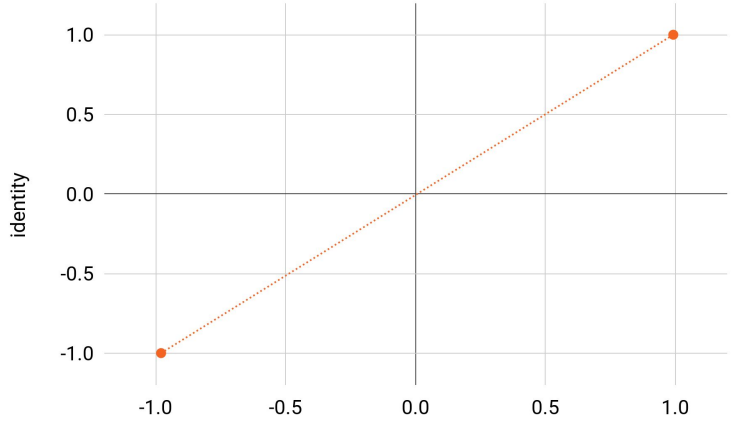
Duluth, Minnesota

June 23, 1936 Olga Lakela



ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

https://xkcd.com/2173



$y = mx + b$

z−score normalize

ANN: untrainable layers/operations…

input receives input data in specified dimensions

e.g. [*None*, 128, 128, 3], [*None*, None, None, None]

reshape modifies tensor shape without changing data

e.g. [*128*, 16, 16, 32] => [*128*, 256, 32]

$16 \times 16 \times 32 = 8192 = 256 \times 32$

permute modifies tensor shape without changing data

e.g. [*128*, 16, 16, 32] => [*128*, 16, 32, 16]

ANN: …untrainable layers/operations…

slice modifies tensor shape with data removal
e.g. [*128*, 16, 16, 32] => [*128*, 16, 16, 8]

concatenate combines tensors along an axis
e.g. [*128*, 16, 16, 32], [*128*, 16, 16, 32]
=> [*128*, 16, 16, 64]

zero padding increase tensor dimensions by adding zeros

up sampling increase tensor dimensions by interpolation
e.g. [*128*, 16, 16, 64] => [*128*, 32, 32, 64]

ANN: …untrainable layers/operations…

max pool resamples using the maximum value
samples within a sliding rectangular window
optionally reduces tensor size (stride > 1)
e.g. [*128*, 16, 16, 32] => [*128*, 8, 8, 32]

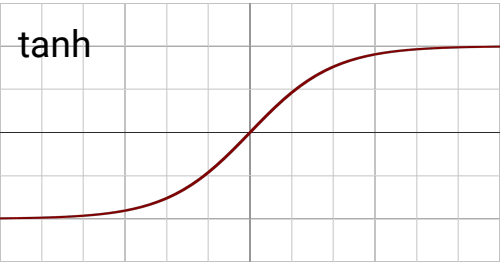
average pool resamples using the average value
samples from within a sliding rectangular window
optionally reduces tensor size (stride > 1)
e.g. [*128*, 32, 32, 64] => [*128*, 16, 16, 64]

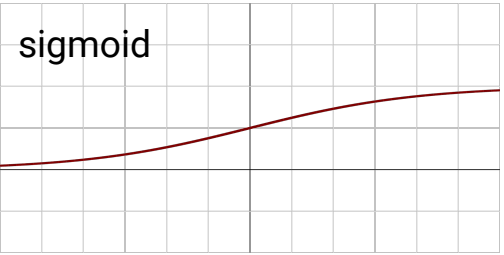
ANN: …untrainable layers/operations…

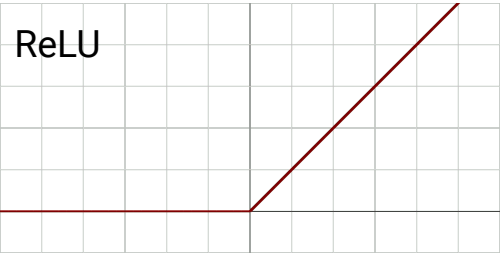
global max pool	resamples using the maximum value samples within each channel reduces tensor size (and shape) e.g. [*128*, 16, 16, 32] => [*128*, 1, 32]
global average pool	resamples using the average value samples within each channel reduces tensor size (and shape) e.g. [*128*, 16, 16, 64] => [*128*, 1, 64]

ANN: …untrainable layers/operations…

add	element–wise addition (‘same’ shape in/out)
subtract	element–wise subtraction (‘same’ shape in/out)
multiply	element–wise multiplication (‘same’ shape in/out)
dot	element–wise multiplication then sum products (‘same’ shape in one value out)
reduce	[sum|mean|min|max|...] to one value
math	e.g. square root, log, absolute value, etc.
trig	e.g. cos, sin, etc.







ANN: …untrainable layers/operations…

activation constant non–linear transformations

e.g. sigmoid = $(1+e^{-x})^{-1}$

inactivates some inputs (zeros)

rescales some inputs (+/- numbers)

amplifies some inputs (+/- numbers)

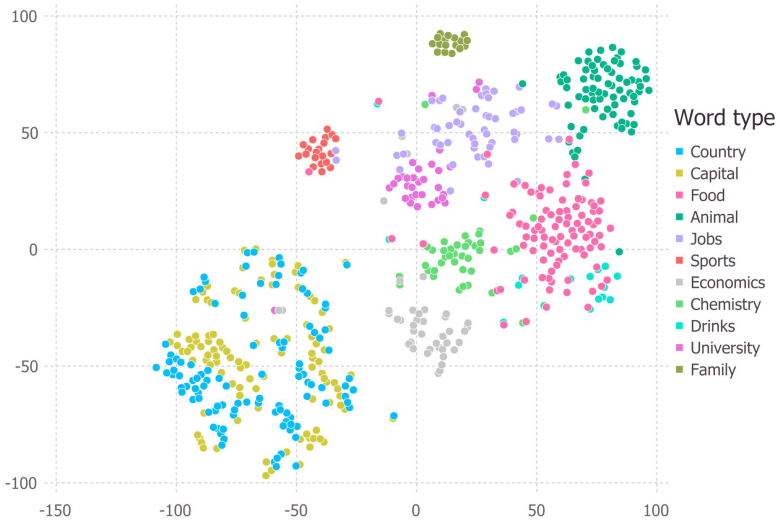
input size = output size

(https://en.wikipedia.org/wiki/File:Activation_{tanh|logistic|rectified_linear}.svg)

ANN: …(un)trainable layers/operations

dropout	randomly changes x% values to zero
noise	injects random (Gaussian) noise
normalization	recenters + rescales (mean = 0, std = 1)
	a.k.a. *z*–score normalization
	by batch or channel
	center and spread are optionally learned

ANN: trainable layers…



embedding converts input integers to vectors of numbers

integers encode text or image fragments

e.g. byte pair encoding ['_', 'Ne', 'w', '_York']

output vectors of numbers are learned

output size > input size

e.g. [*128*, 16] => [*128*, 16, 32]

(Claassen 2020)

ANN: …trainable layers…

dense

every input is connected to every output

learned weights mediate the interconnections

e.g. 32 inputs × 32 outputs = 1024 weights

input (vector) and weight (matrix) multiplied

input•weights•activation = output

input•weights•activation + bias = output

output size >|=|< input size e.g. [*128*, 16] => [*128*, 32]

a.k.a. fully connected, FC, perceptron,

linear (no activation)