Machine Learning

Rudolf Mayer

mayer@ifs.tuwien.ac.at

October 9th, 2025

Outline

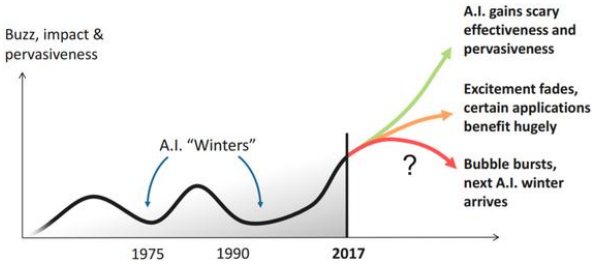
- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

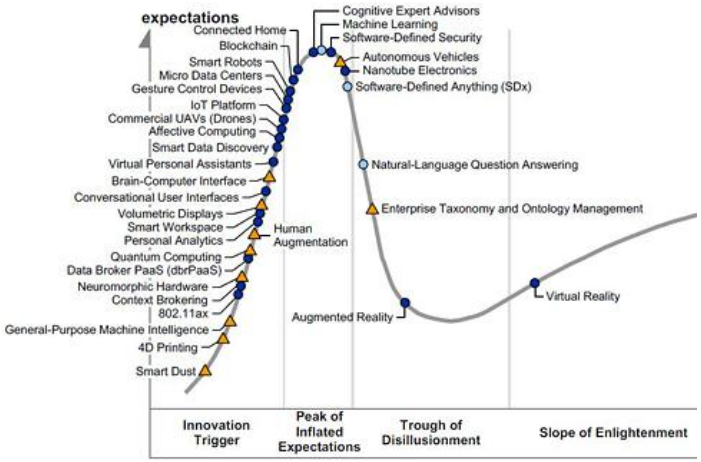
Outline

- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

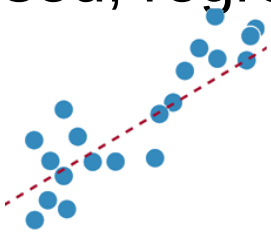
Recap: ML Intro / Scope

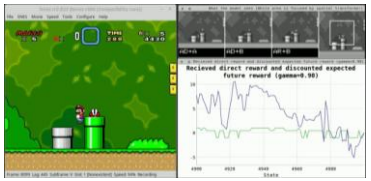
- ML Definitions & setting





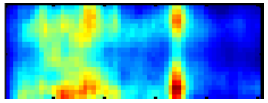
  - Supervised, unsupervised, regression, classification, reinforcement, ..

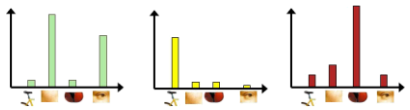




  - Data Science/Mining process

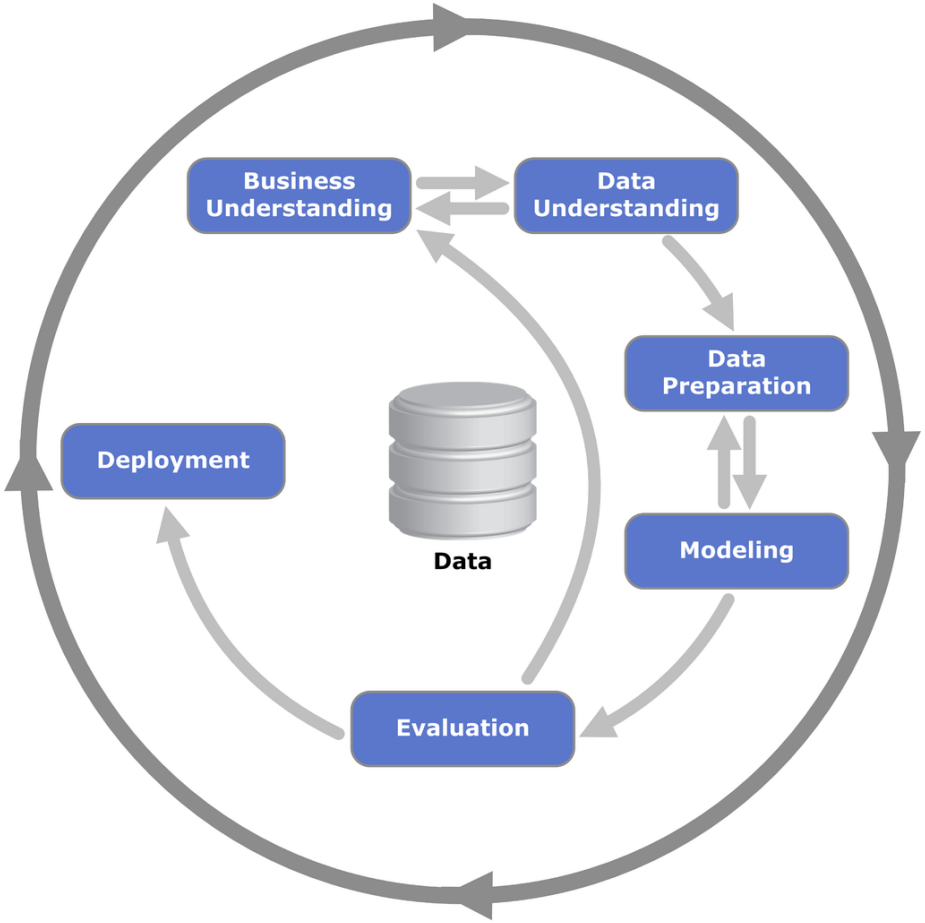
  - Feature extraction



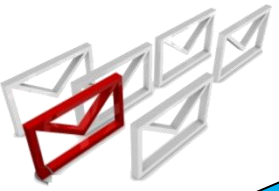


Recap: ML Intro / Scope

- Data Mining Process



Recap: ML Intro / Scope



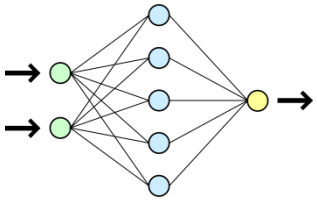
Ground Truth

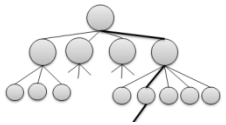
label

Data

extract/ transform

Features



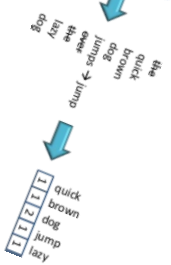


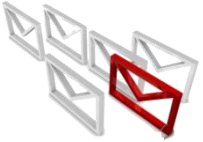
train

ML Model

Features

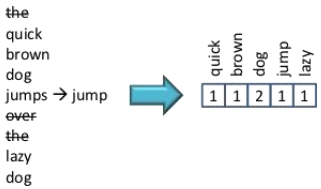
extract/ transform





Unseen Data

apply



Prediction

Continuous (regression)

Binary (one out of two classes)

Multi-class (one out of n classes)

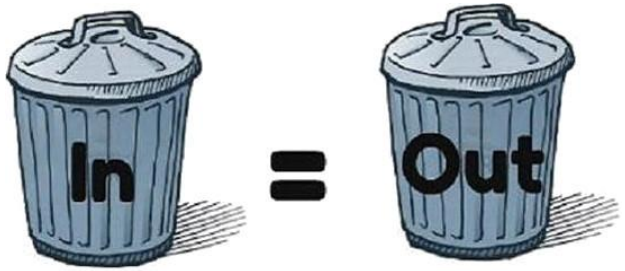
Multi-label (one or more out of n classes)

Outline

- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

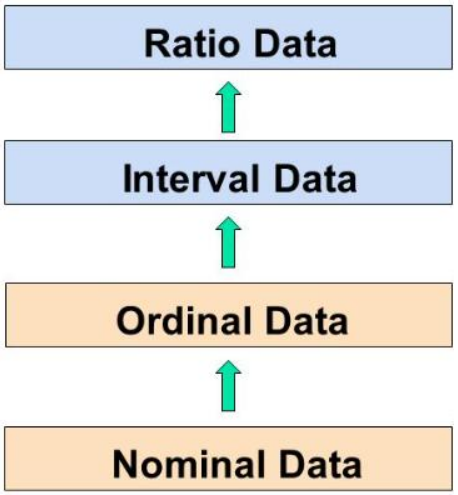
Data Preparation

- Vital step for machine learning (supervised and unsupervised)
- ML algorithm will ***always*** give you a model
  - Quality of that model depends highly on the quality of the input data
- "Garbage in" ➔ "Garbage out"



- One major goal of data preparation:
  - Eliminate "wrong influence" of variables

Data Types



- Possible attribute/feature/variable types (*"levels of measurement"*)
  - Nominal, ordinal, interval and ratio

- Nominal quantities
  - Values are distinct symbols
  - Example: attribute "eye colour": "green","grey", "brown","blue"
    - Values themselves serve only as labels or names
    - Nominal comes from the Latin word for name
    - No relation among values (no ordering or distance measure)
    - Only (in)equality tests can be performed
    - *(You may encode it as number, but order is arbitrary; calculations on it (e.g. mean) are meaningless)*

Data Types

- Ordinal quantities
  - Impose order (rank) on values
  - (In)equality, additionally comparison (<, >) and median
  - But: no distance between values defined
  - Example:
    - attribute “size” in animals data
    - Values: “large” > “medium” > “tiny”
    - Note: addition and subtraction don’t make sense
    - Difference “large” to “medium” and “medium” to “tiny” not necessarily comparable
- *Distinction between nominal and ordinal not always clear*

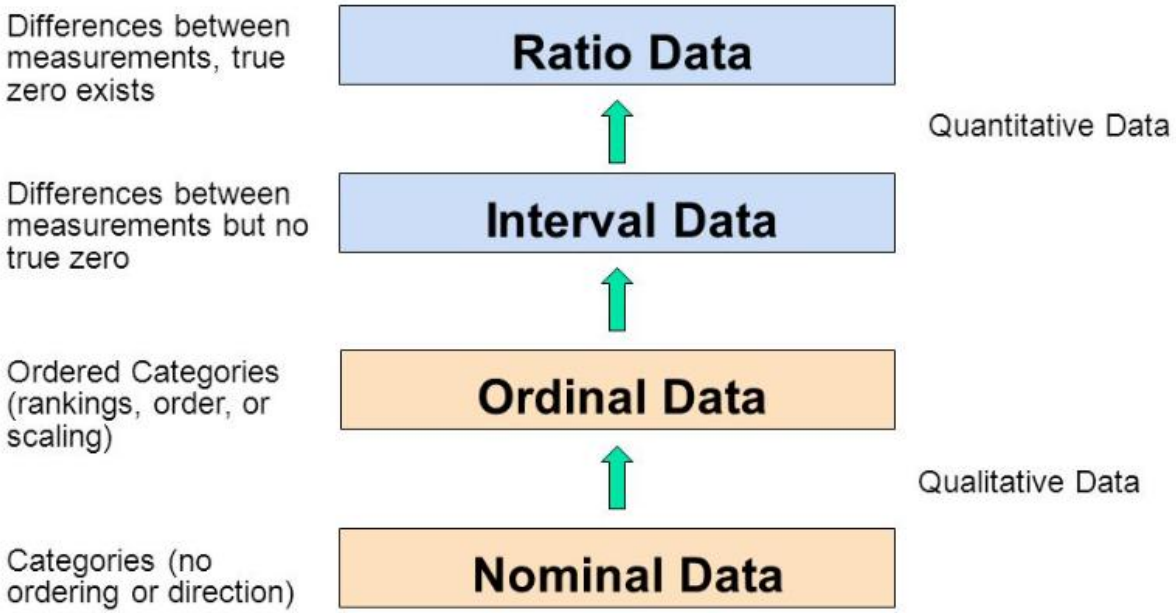
Data Types

- Interval quantities
  - Interval quantities are not only ordered but measured in fixed and equal units
  - Difference between measurements meaningful
  - Example: Temperature, pH values, …
  - Difference of two values makes sense
  - Ratio not (20 is not twice as hot as 10)
  - Sum or product doesn't make sense

Data Types

- Ratio quantities
  - The measurement scheme defines a zero point
  - Example: distance, mass, length
    - Temperature (Kelvin)
  - Distance between an object and itself is zero
  - All mathematical operations are allowed
  - Meaningful to have ratios: “twice as long”

Data Types



Can compute	Nominal	Ordinal	Interval	Ratio
Frequency distribution	Yes	Yes	Yes	Yes
Median and percentiles	No	Yes	Yes	Yes
Add or subtract	No	No	Yes	Yes
Mean, standard deviation, …	No	No	Yes	Yes
Ratios, coefficient of variation	No	No	No	Yes

Data Preparation

- Example data set from earlier (lung cancer)

gender	*age*	*height*	*smoker*	*eye colour*
male	19	170	yes	green
female	44	162	yes	grey
male	49	185	yes	blue
male	12	178	no	brown
female	37	165	no	brown
female	?	157	no	?
male	44	190	no	blue
female	27	178	yes	brown
female	51	162	yes	green
female	81	168	?	grey
male	22	184	yes	brown
male	29	176	no	blue

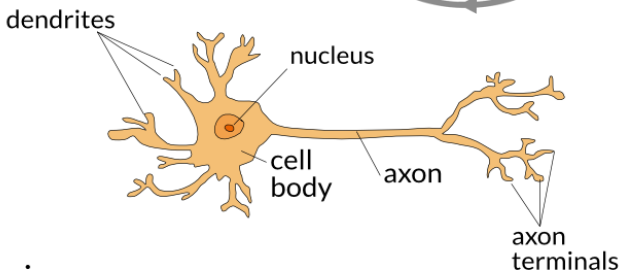
- Potential issues ?
  - Quantitative (continuous) data with different scales
  - Data type not fitting
    - Categorical (nominal, ordinal, ..) vs numerical (interval, ratio)
  - Missing values

➔ More on that later!

Outline

- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

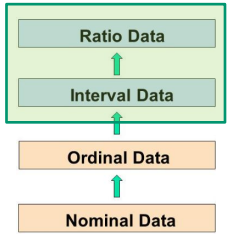
Perceptron

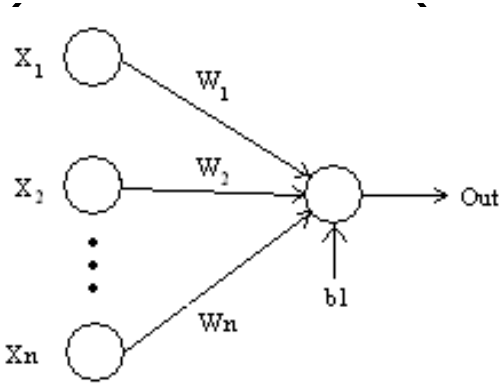


- Proposed in 1957
  - Predecessors already in late 40s
  - *Why is it relevant for you? ☺*

- Artificial (neural) network
  - only one neuron; “single-layer perceptron”

- Input: **continuous values** *X*

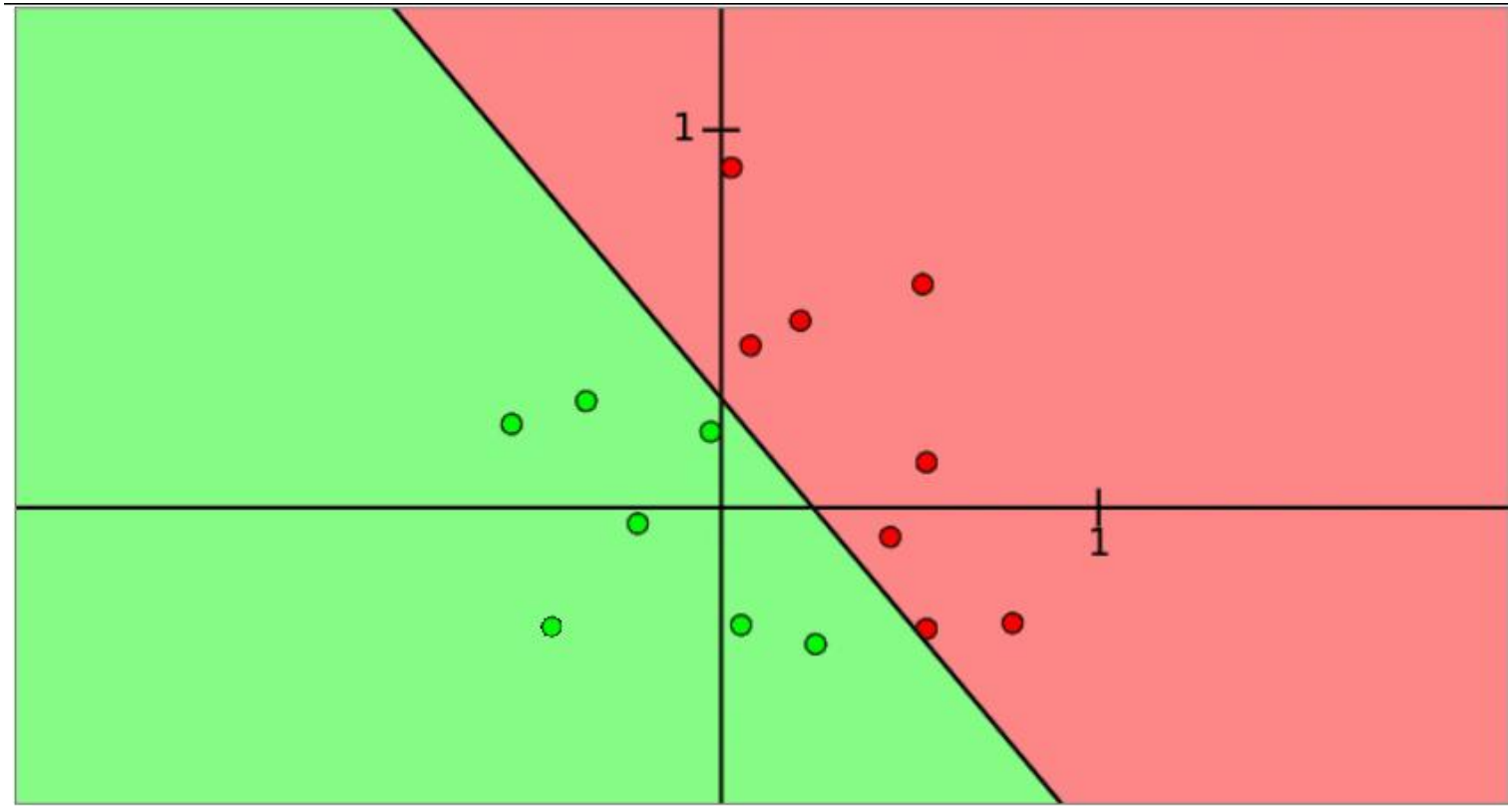




- Output: activation a
  - Boolean value 0 / 1

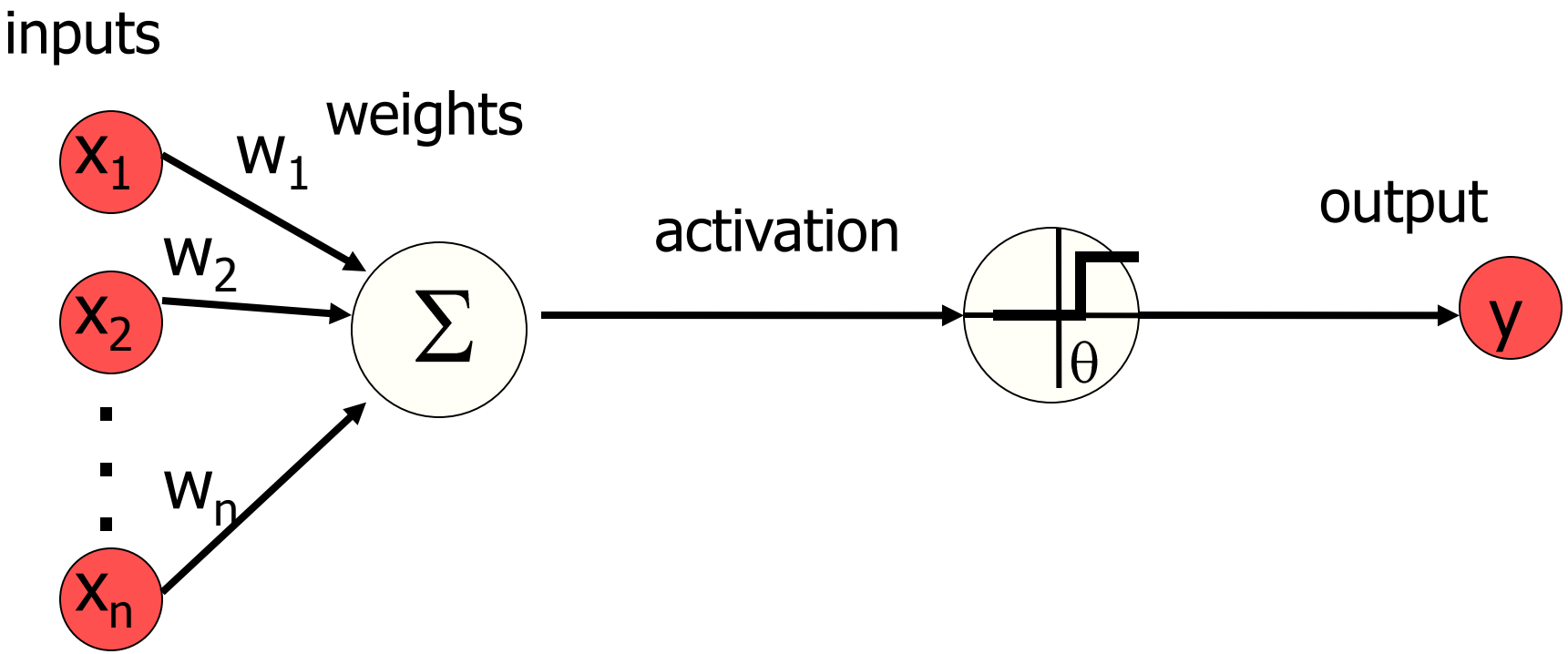
Perceptron (classification): Goal

- What we want to achieve (in classification in general):
  "Partition" input space so it separates classes
  - Here: 2D Input (x, y coordinate), 2 classes (red & green)



Correct: 15/15

Perceptron


inputs
weights
x_1
w_1
w_2
x_2
w_n
x_n
Σ
activation
θ
output
y

Perceptron

- Linear combination of inputs
  - Using weights *W* $a = \sum_{i=1}^{n} w_i x_i$

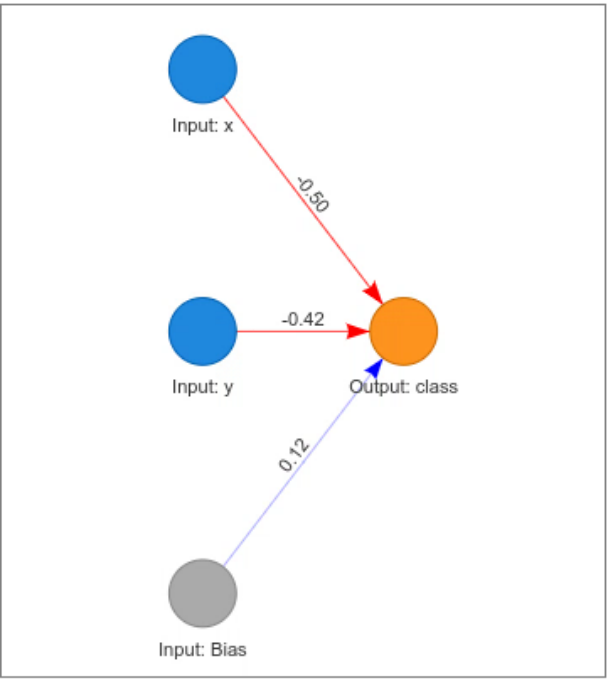
- Pass through threshold activation function with threshold θ

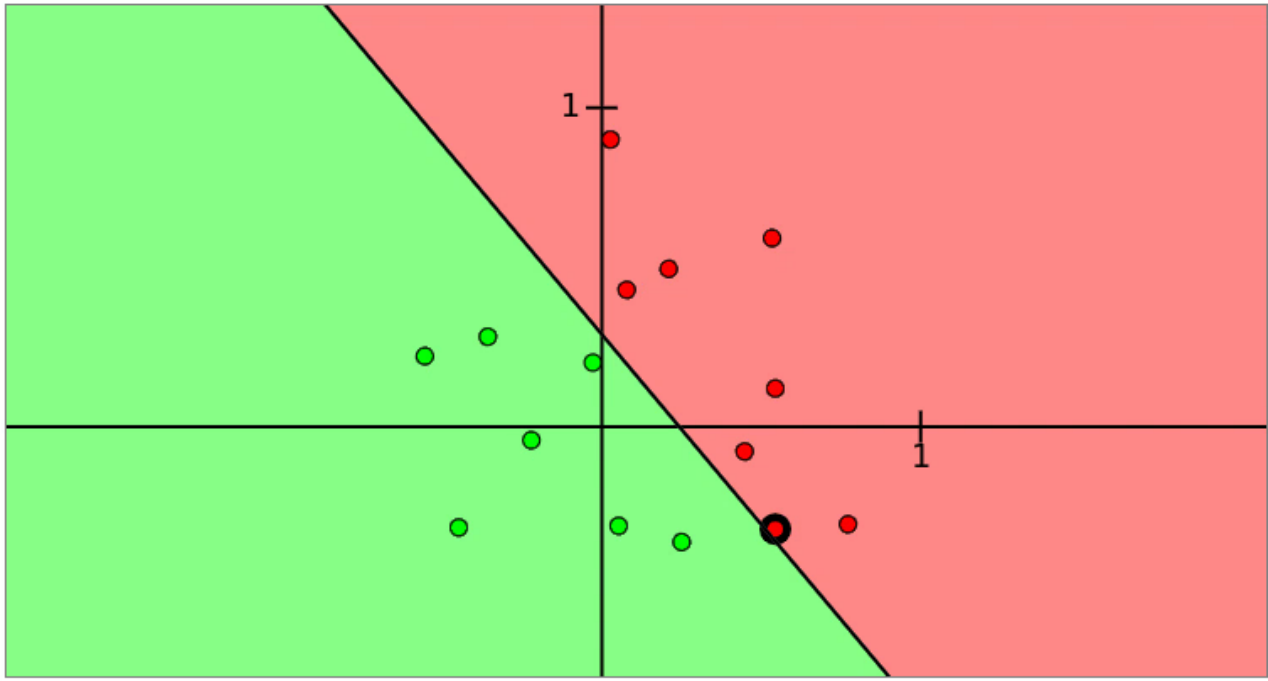
$$y = f(x) = \begin{cases} 1 \text{ if } a \geq \theta \\ 0 \text{ if } a < \theta \end{cases}$$

(Heaviside step function)

- Often: $\theta = 0$

Perceptron: Demo





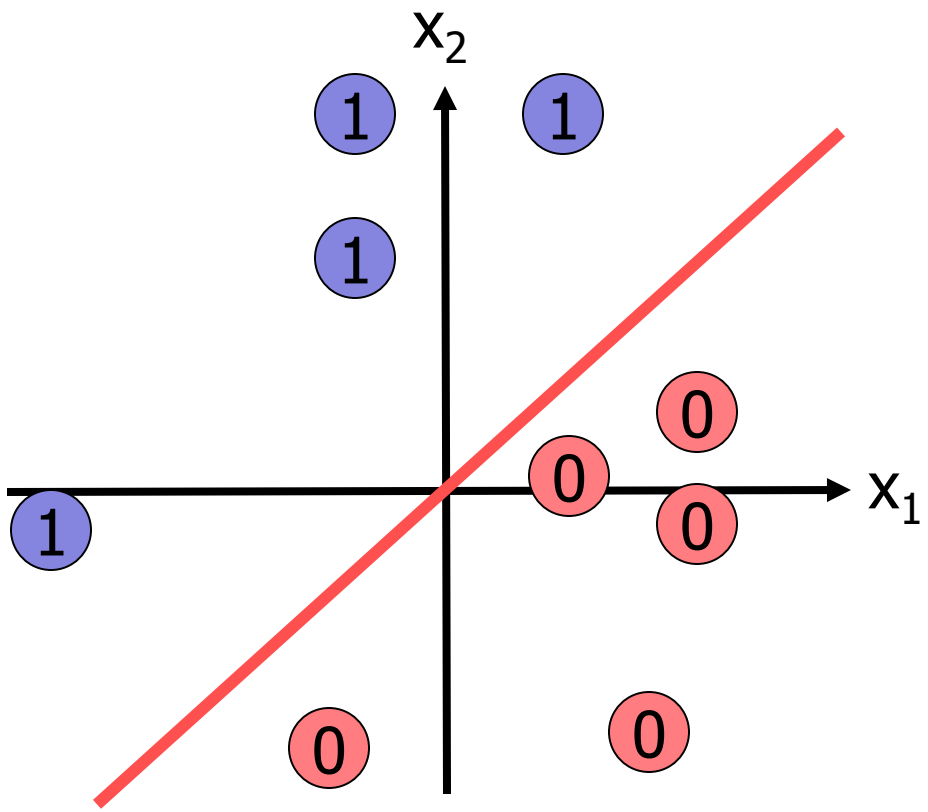
Stop Reset Train All Train Single

Correct: 15/15 — Iteration: 53

Perceptron: simple example

- Two-dimensional data set
  - Variables x_1 & x_2
    - E.g. position on a map
- Weight vector also has two components
  - w_1 & w_2
- Data set can be easily visualised in Cartesian coordinate system
- Two classes: ***0*** and ***1*** (sometimes also -1 & 1)
  - Visualised as red & green

Perceptron visualised



Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron: learning algorithm

- Training the model: learning the weights from labelled samples (label: **y**)

  - Initialise weights

  - Repeat
    - Present training sample **x**
    - Predict sample label: **y'** = f(x)
    - Prediction correct? Compare **y** and **y'**
      - if **y' ≠ y** ➔ Compute new weights **w'** as **w'**=**w** + α (y-y') **x**

  - Until prediction correct (y'=y) for all samples

Perceptron: learning algorithm

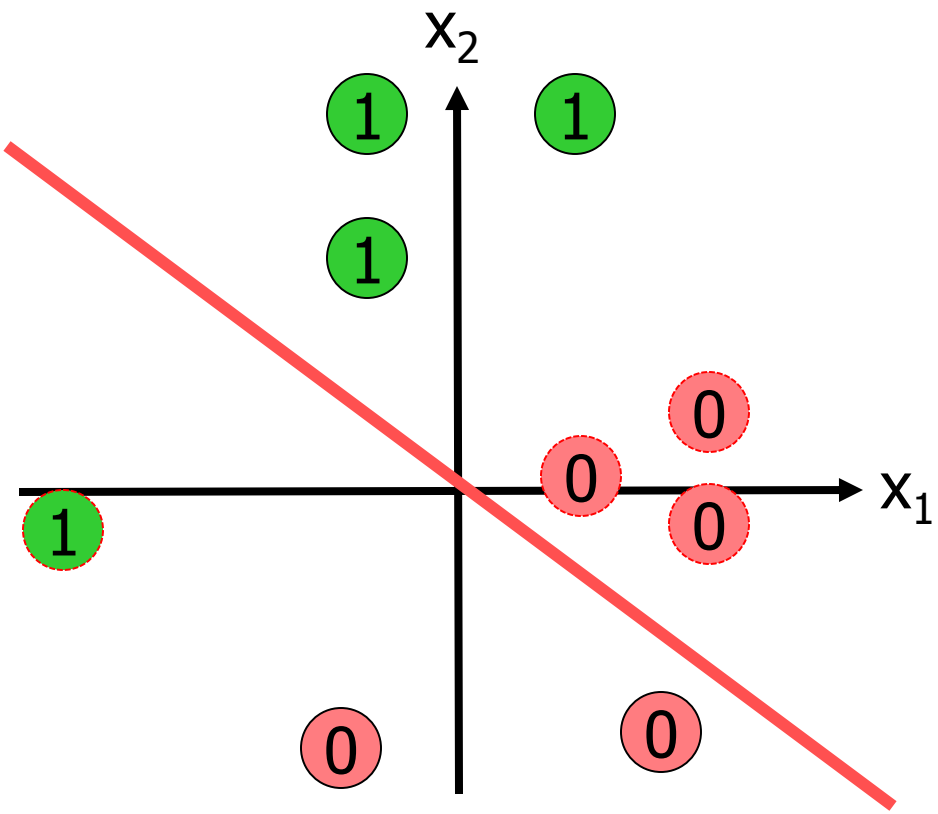
- Parameter α: *learning rate*
  - determines the magnitude of weight updates

- If output is correct (y=y')
  - weights **not** changed

- If output is incorrect (y ≠ y')
  - weights changed such that the output of the for the new weights **w'** is ***closer*** to the input x_i

Perceptron training

- t=0: random weights for initialisation

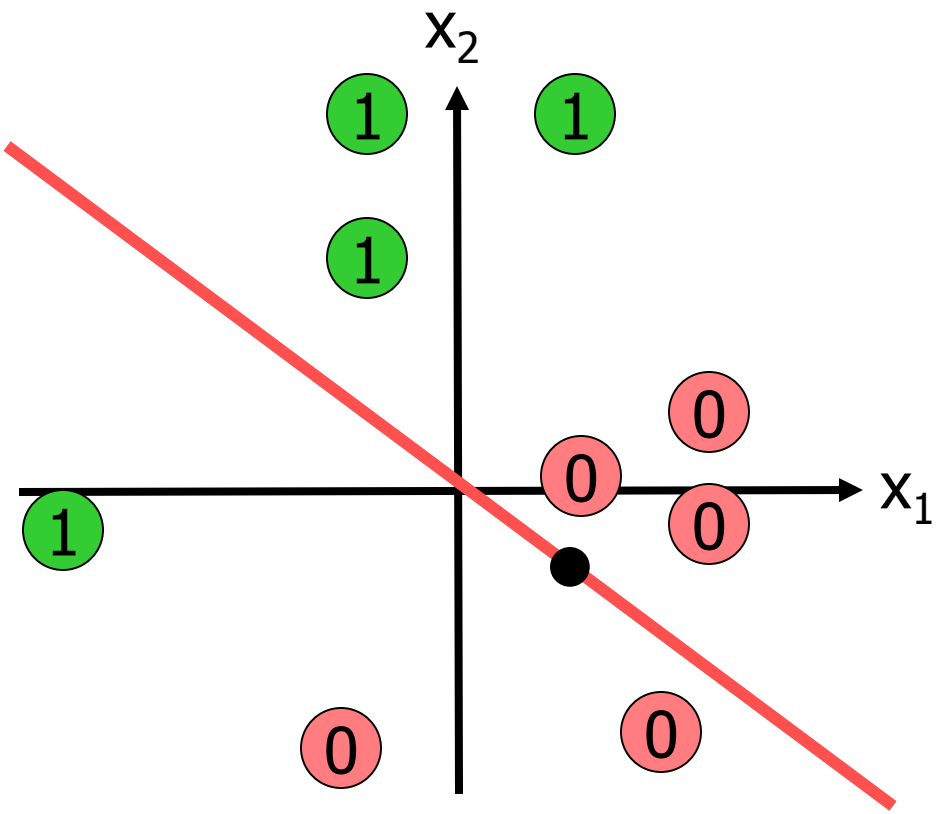


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=0: random weights for initialisation



Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Initial weight vector:

$W_1=1,\ W_2=1$

Point on decision line

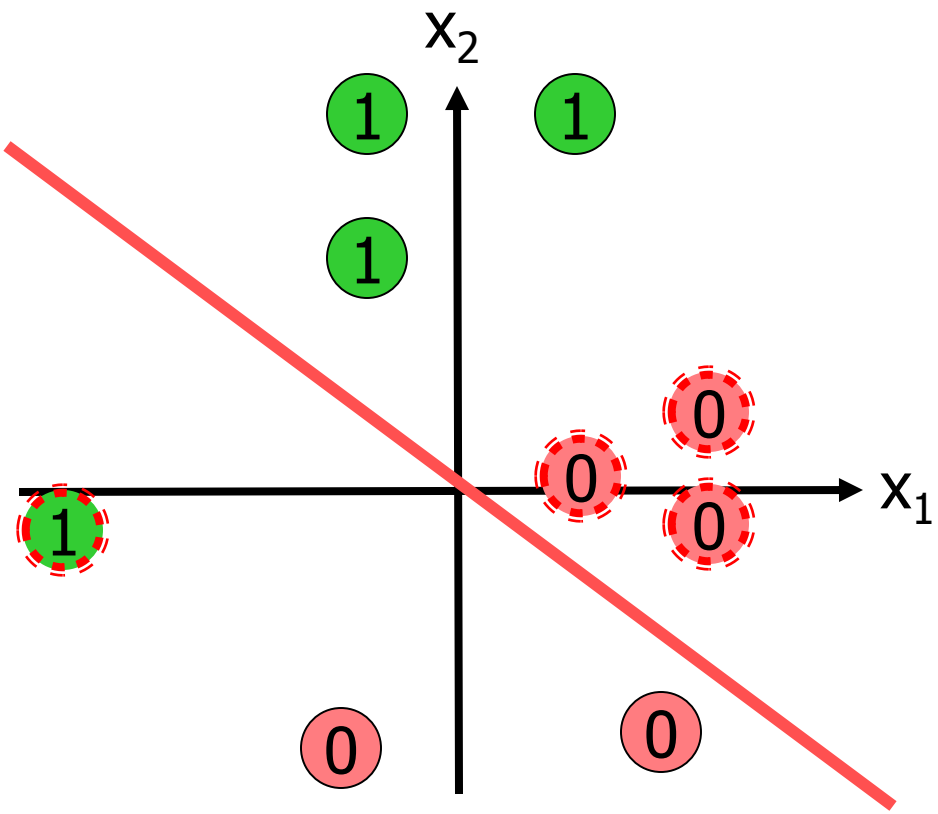
(1,-1)

➔ 1*1 +1*(-1) =

= 1 - 1 = 0 = θ

Perceptron training

- t=1: compute y' = f(x), compare to *y*

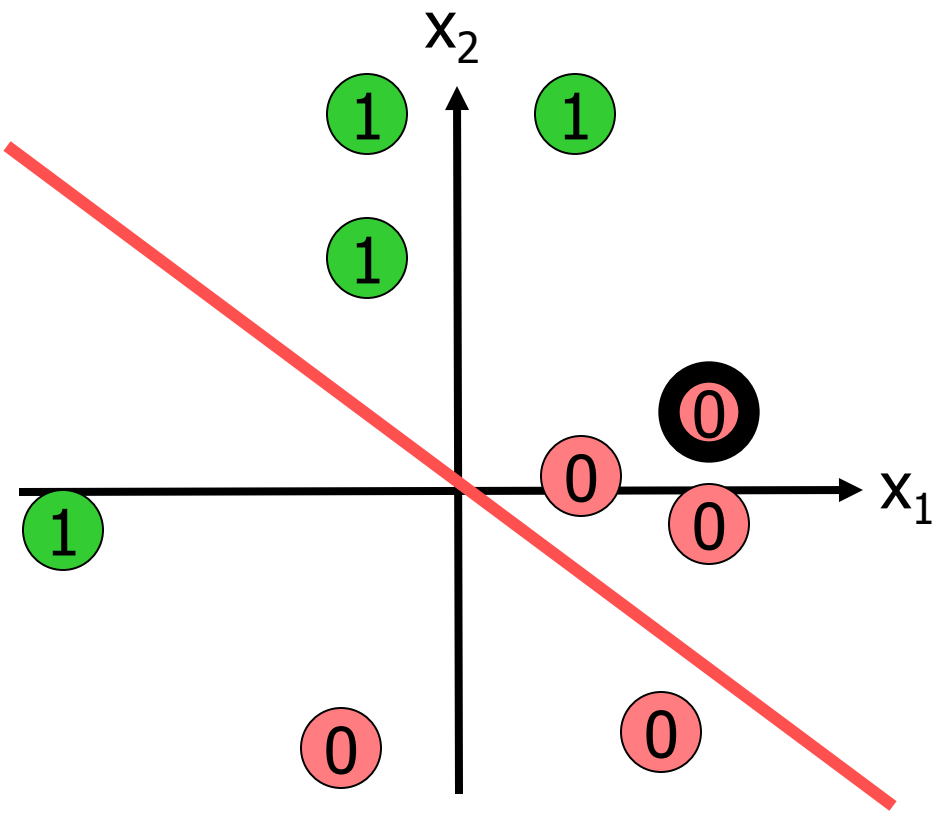


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=1: not all f(x) = y ➔ train
- Pick a point, potentially adapt weights (decision line)

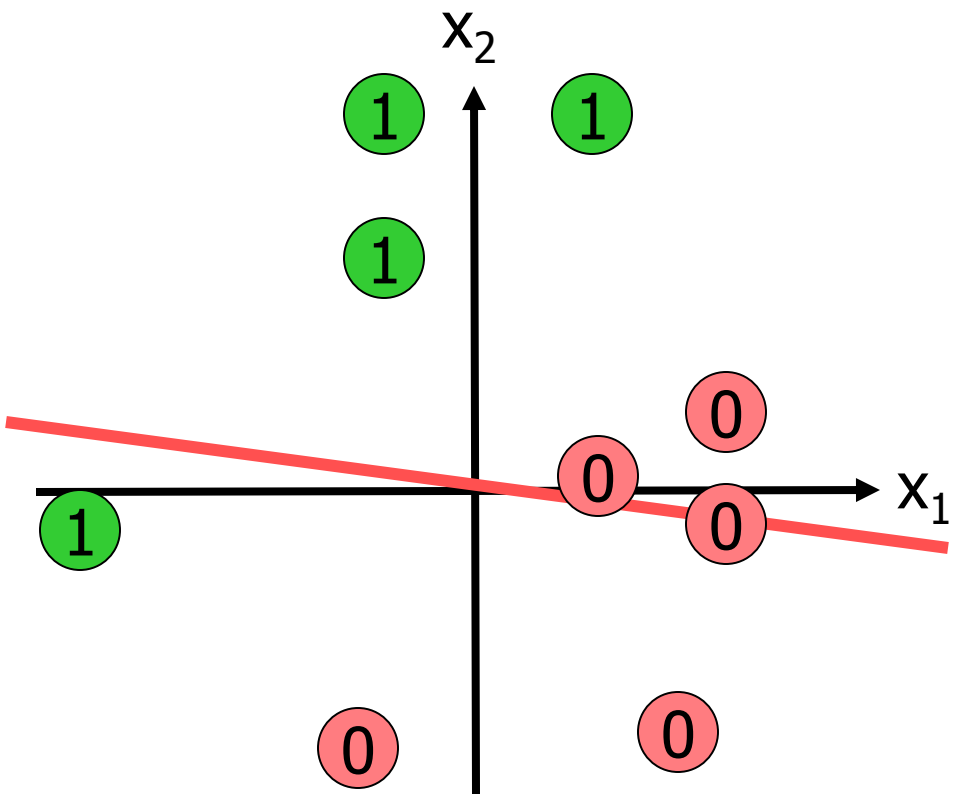


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=1: not all f(x) = y ➔ train
- Pick a point, potentially adapt weights (decision line)

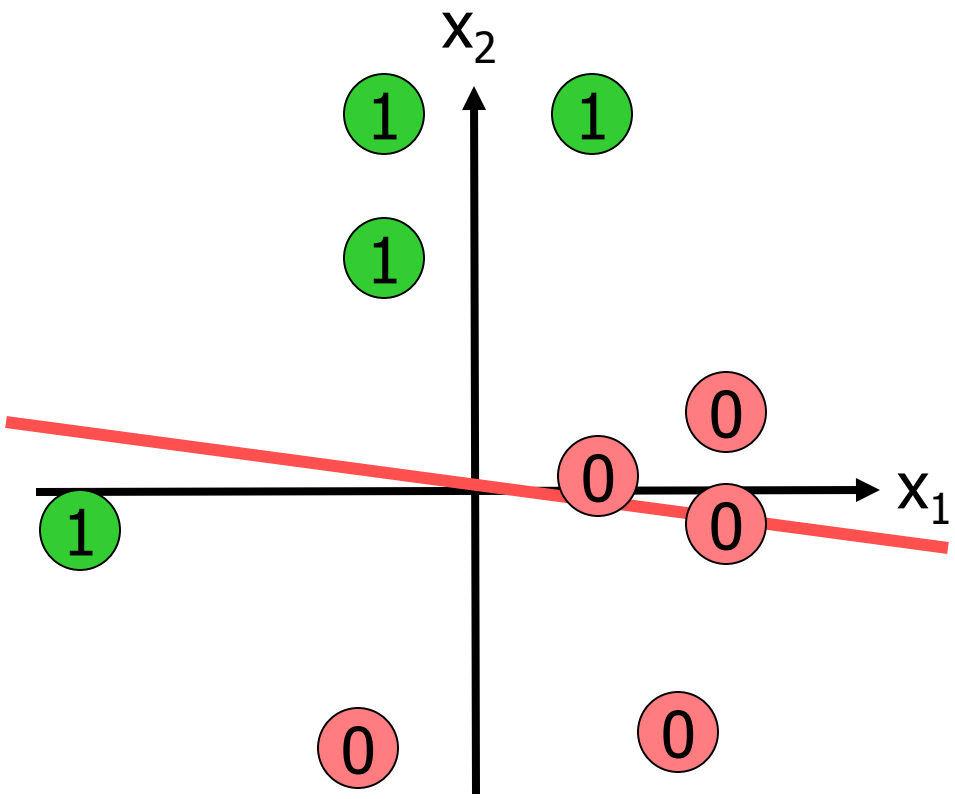


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=1: not all f(x) = y ➔ train
- Pick a point, potentially adapt weights (decision line)

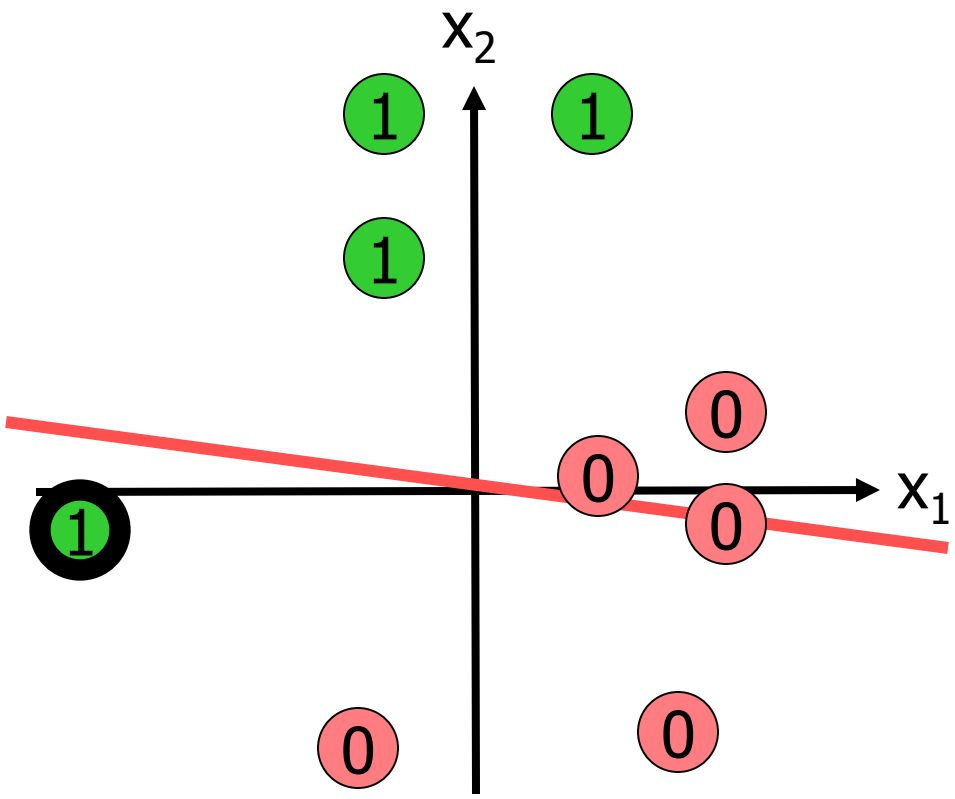


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=2: not all f(x) = y ➔ train
- Pick a point, potentially adapt weights (decision line)

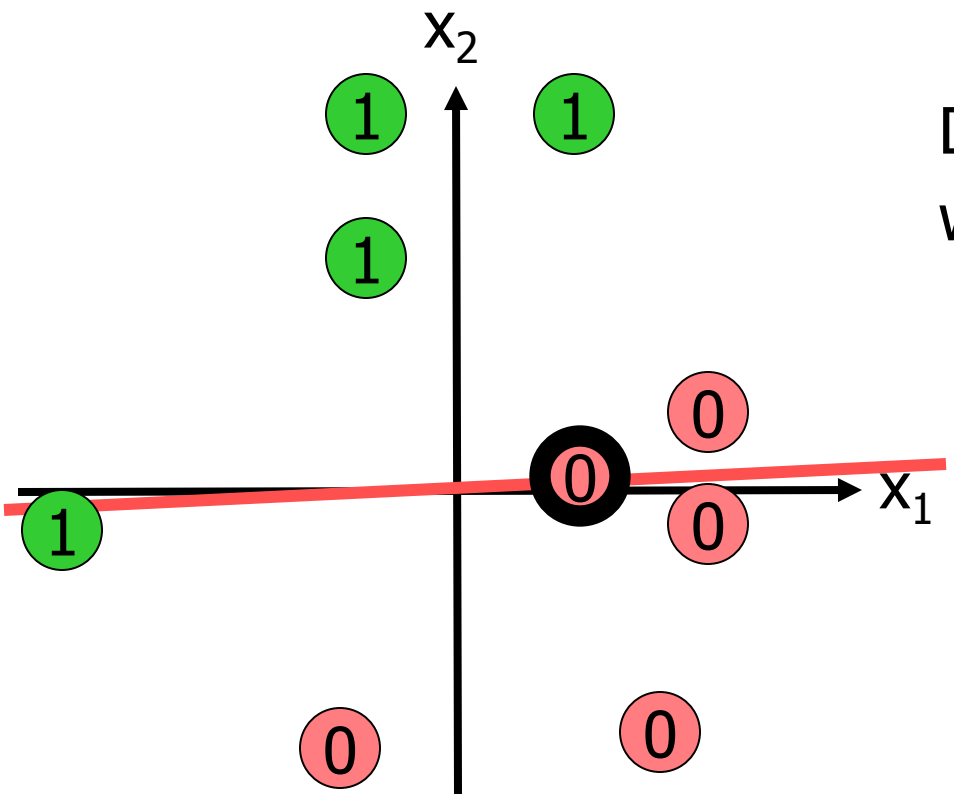


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=3: not all f(x) = y ➔ train
- Pick a point, potentially adapt weights (decision line)

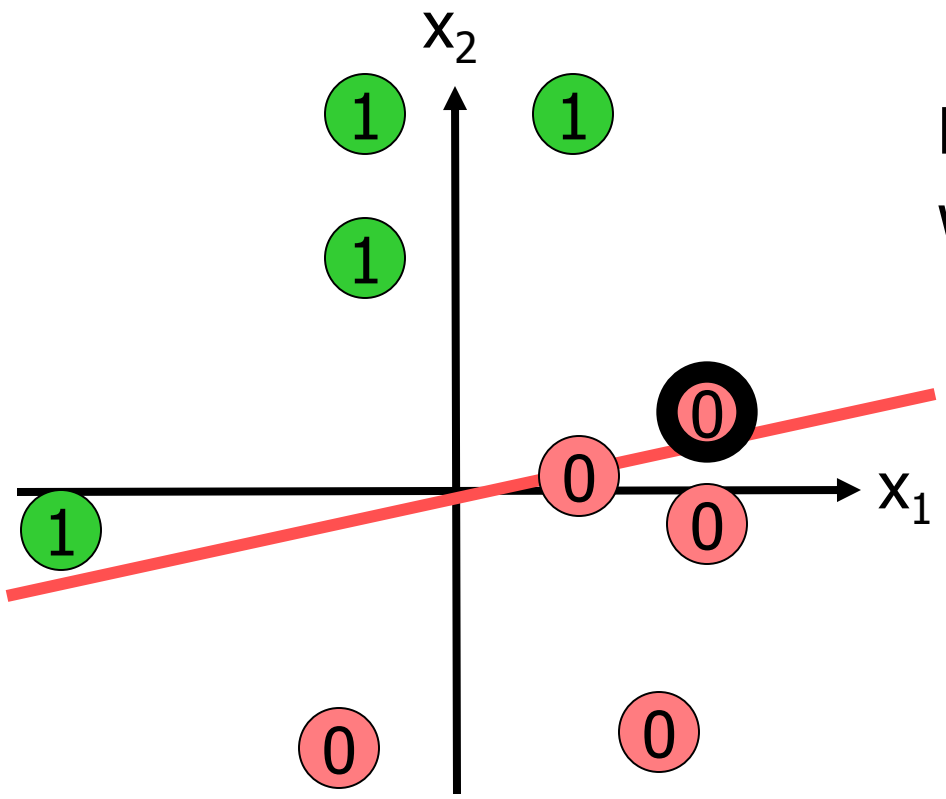


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=4: not all f(x) = y ➔ train
- Pick a point, potentially adapt weights (decision line)

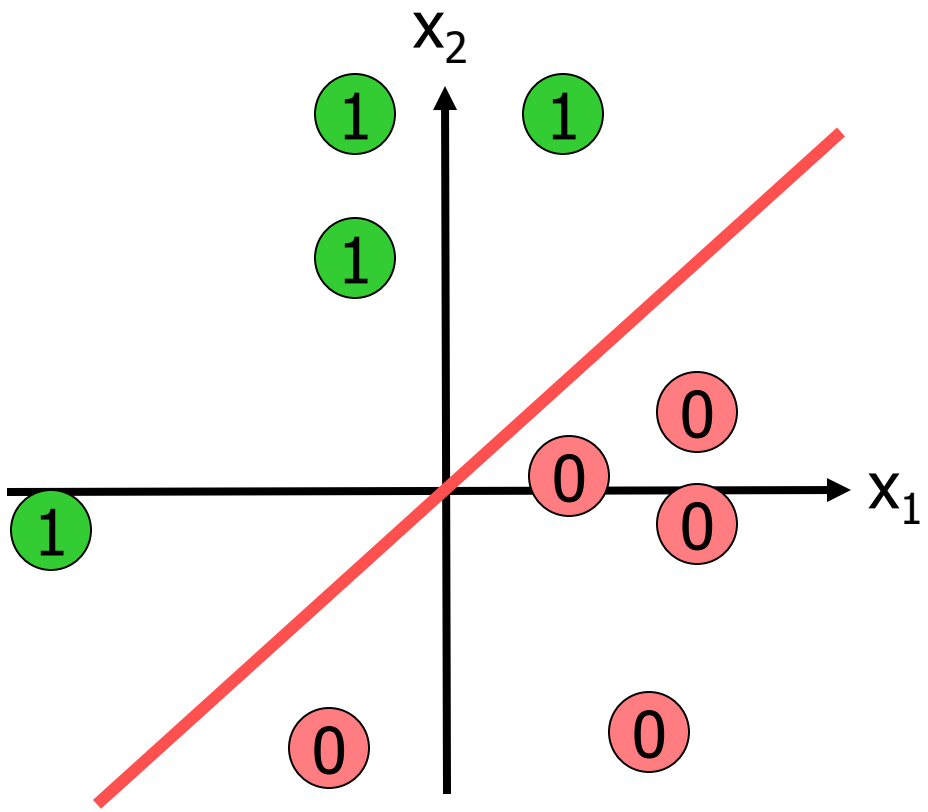


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

- t=5: all f(x) = y ➔ final state

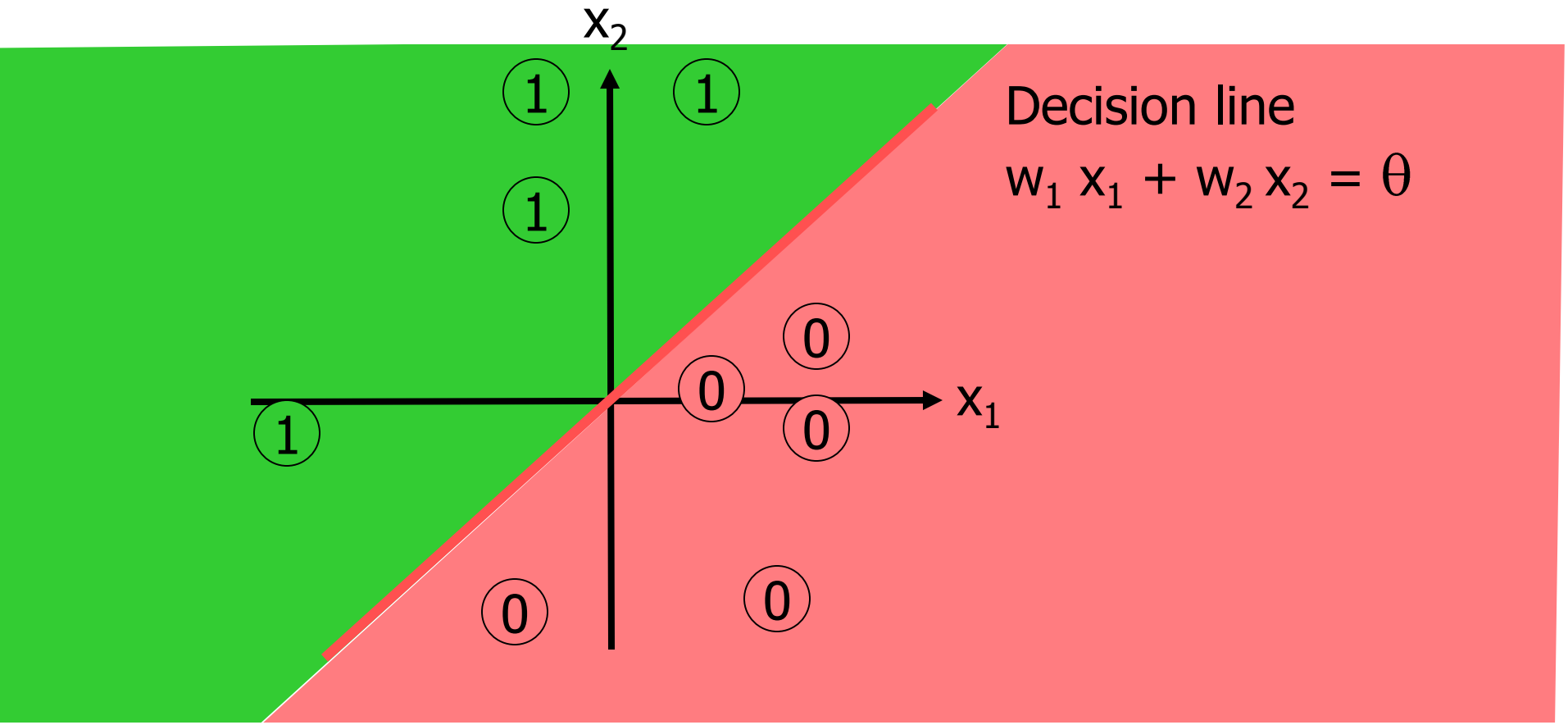


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron training

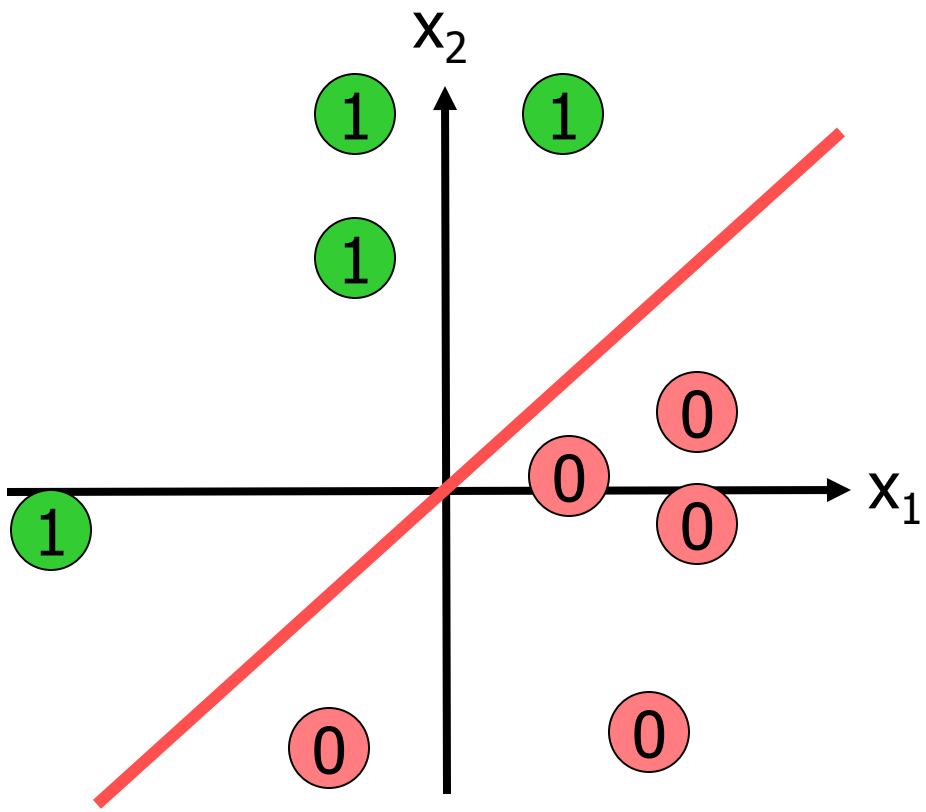
- t=5: all f(x) = y ➔ final state; classification in



(green: correctly , red: wrongly classified)

Perceptron training

- t=5: all f(x) = y ➔ final state



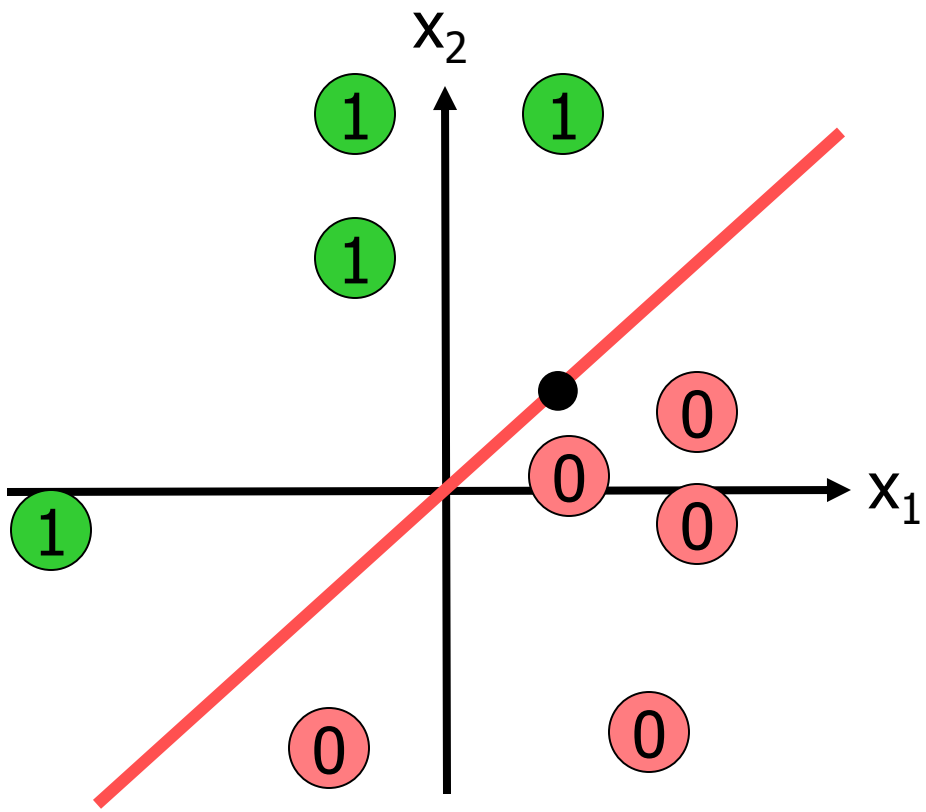
Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Weight vector ?

Perceptron training

- t=5: all f(x) = y ➔ final state



Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Weight vector ?

$W_1 = -1,\ W_2 = 1$

Point on decision line

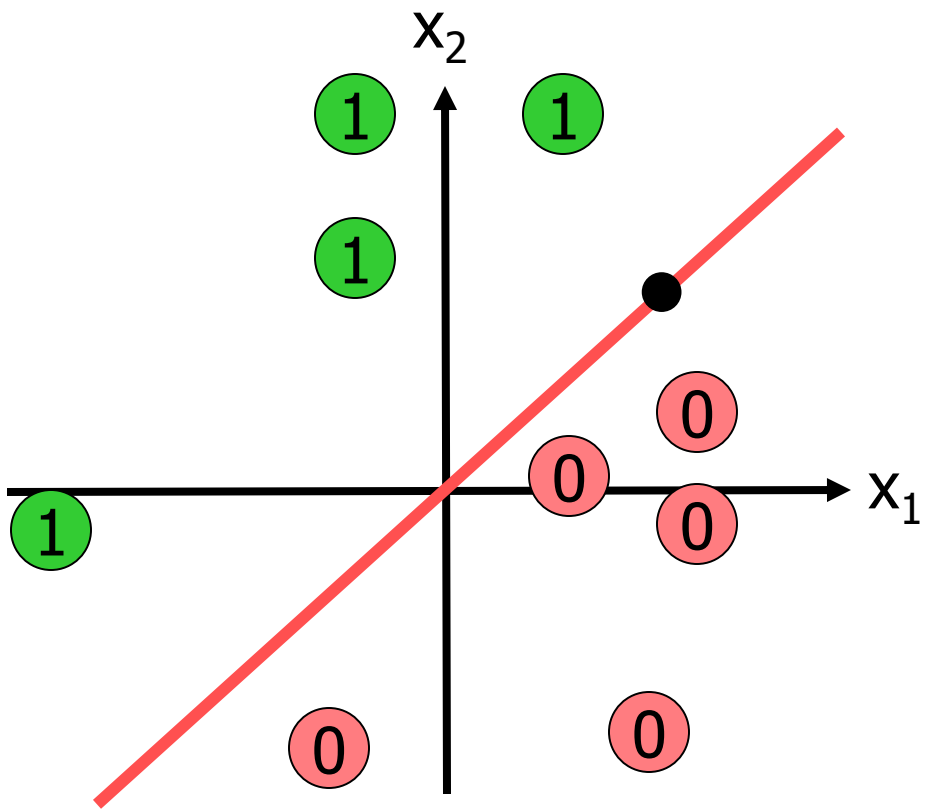
(1,1)

➔ (-1)*1 + 1*1 =

= -1 + 1 = 0 = θ

Perceptron training

- t=5: all f(x) = y ➔ final state



Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Weight vector ?

$W_1 = -1,\ W_2 = 1$

Point on decision line

(2,2)

➔ (-1)*2 + 1*2 =

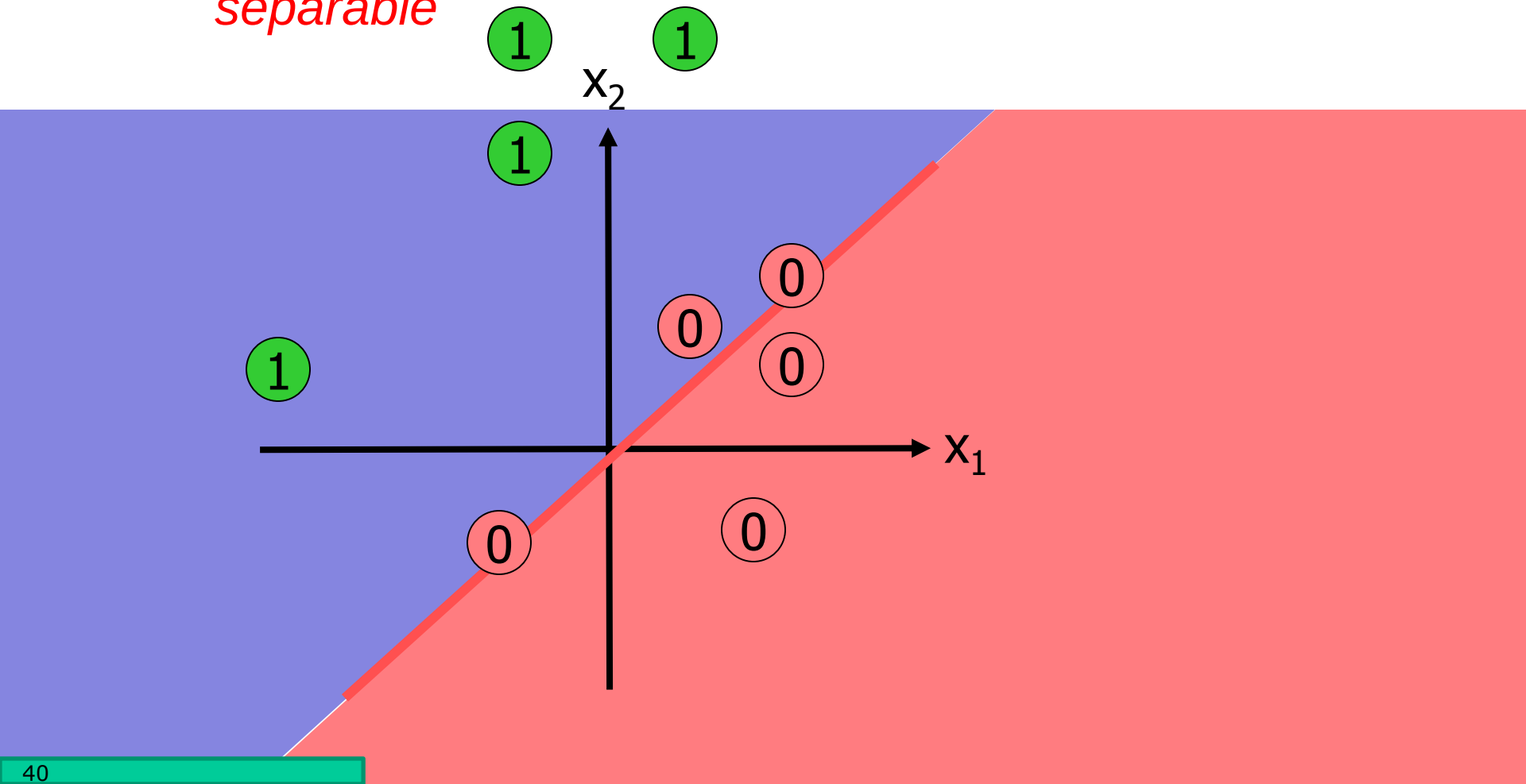
= -2 + 2 = 0 = θ

Perceptron: properties

- Separates linearly
  - Converges to a stable state when data is *linear separable*

Perceptron: properties

- Separates linearly
  - Converges to a stable state when data is *linear separable*

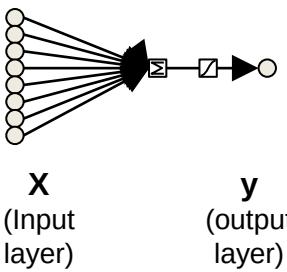


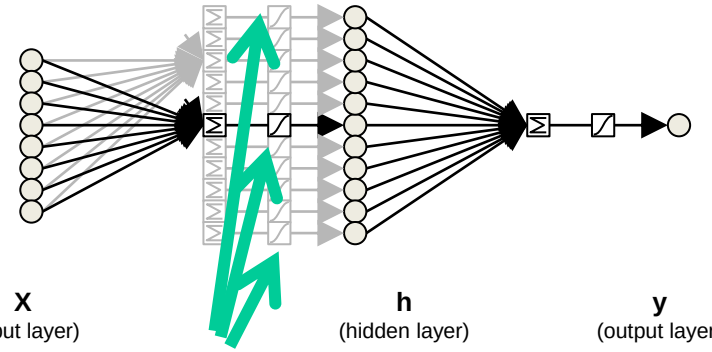
Perceptron: properties

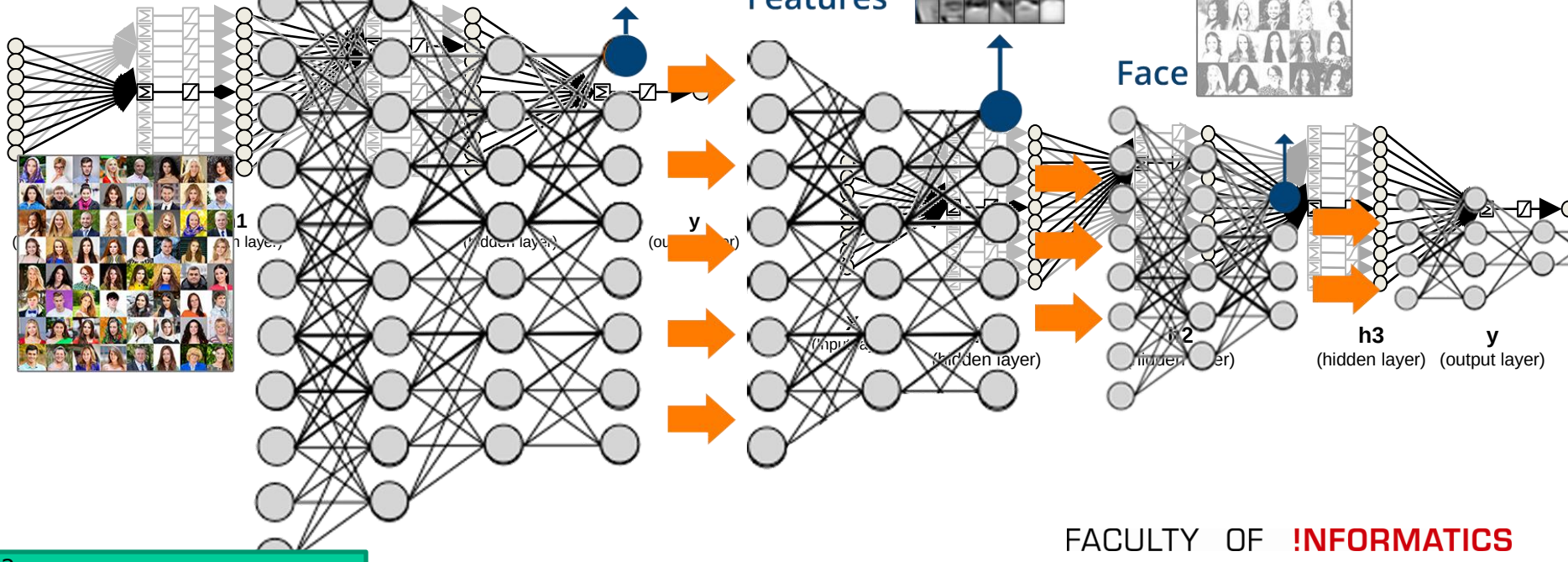
- Very simple model
- Separates linearly
  - Converges to a stable state when data is *linear separable*
- Can predict binary decisions (true/false)
  - Can be extended for multi-class problems
  - *Training rule similar to other online learning algorithms, e.g. Self-Organising Maps*

- *Why is it relevant?*

Perceptron: why is it relevant?

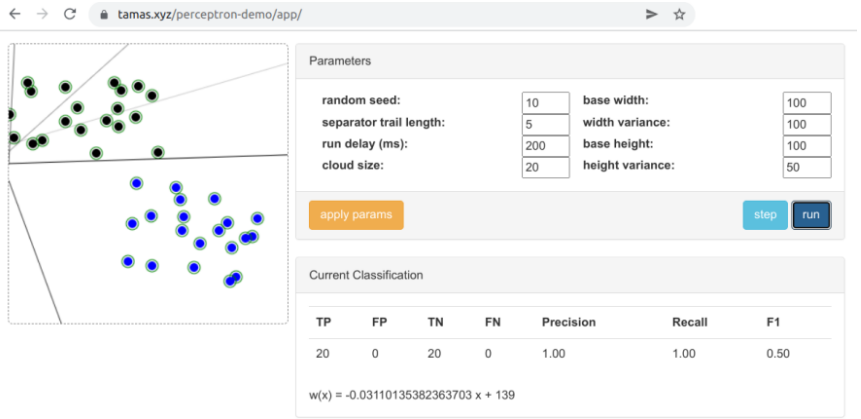




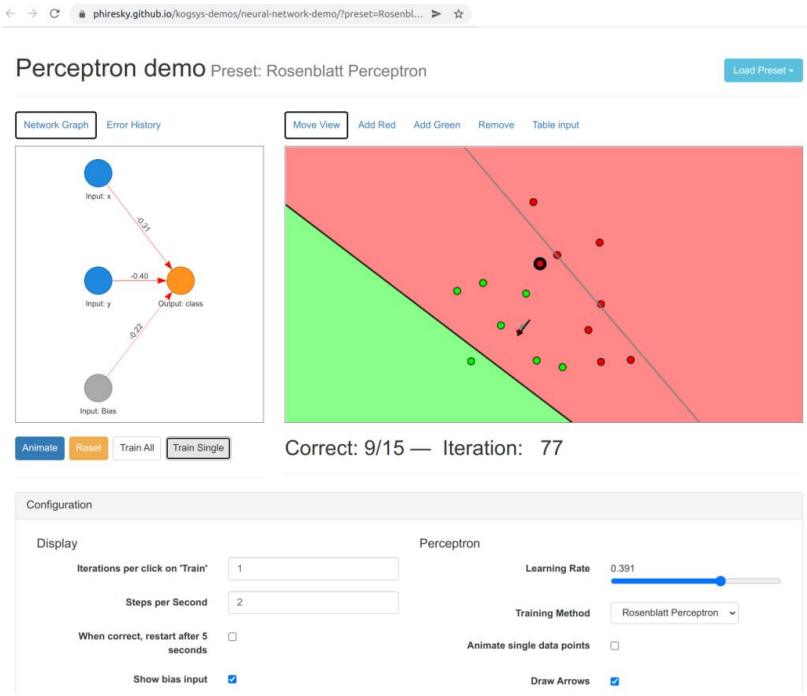


Perceptron demos

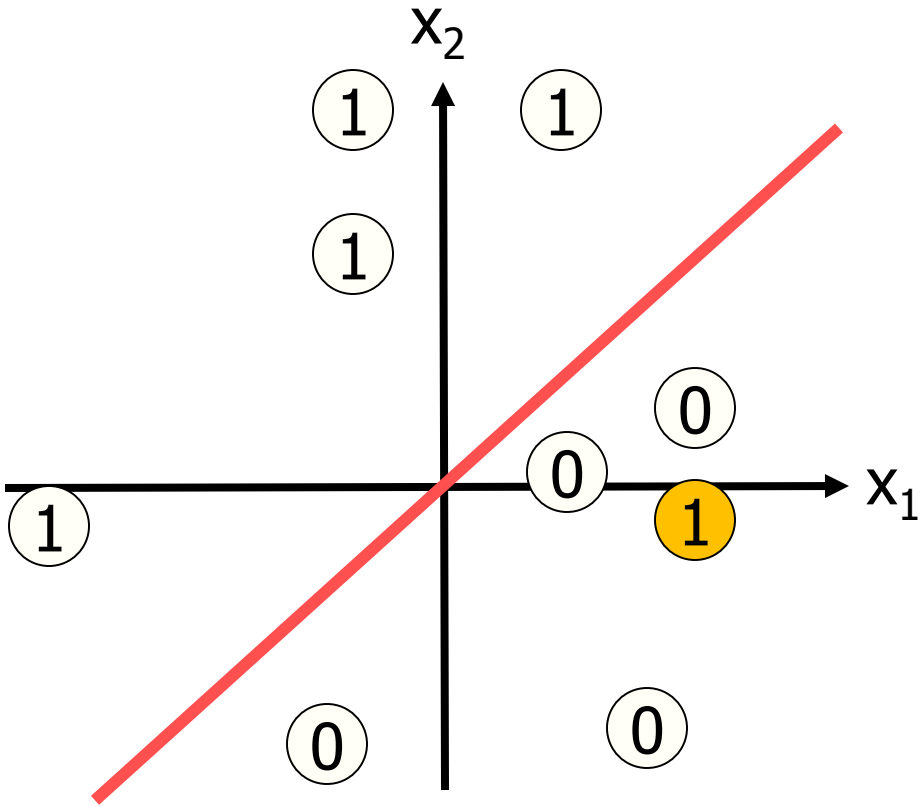
- https://tamas.xyz/perceptron-demo



- https://phiresky.github.io/kogsys-demos/neural-network-demo

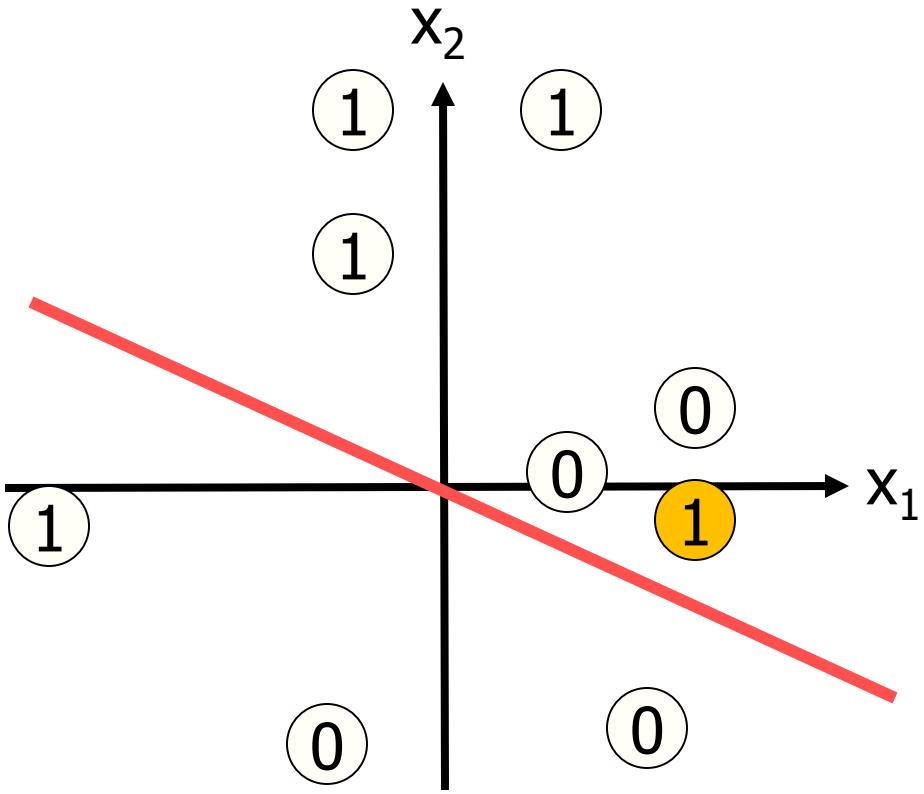


Perceptron: Non-linear separable



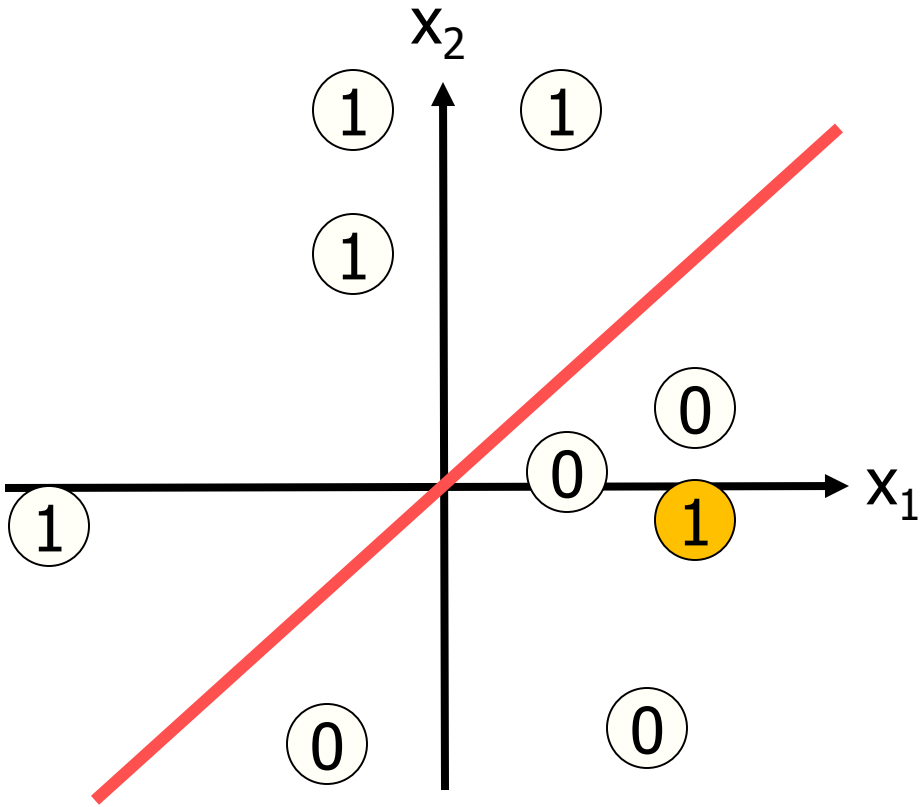
- *Perceptron will not converge to a stable state, but oscilate*

Perceptron: Non-linear separable



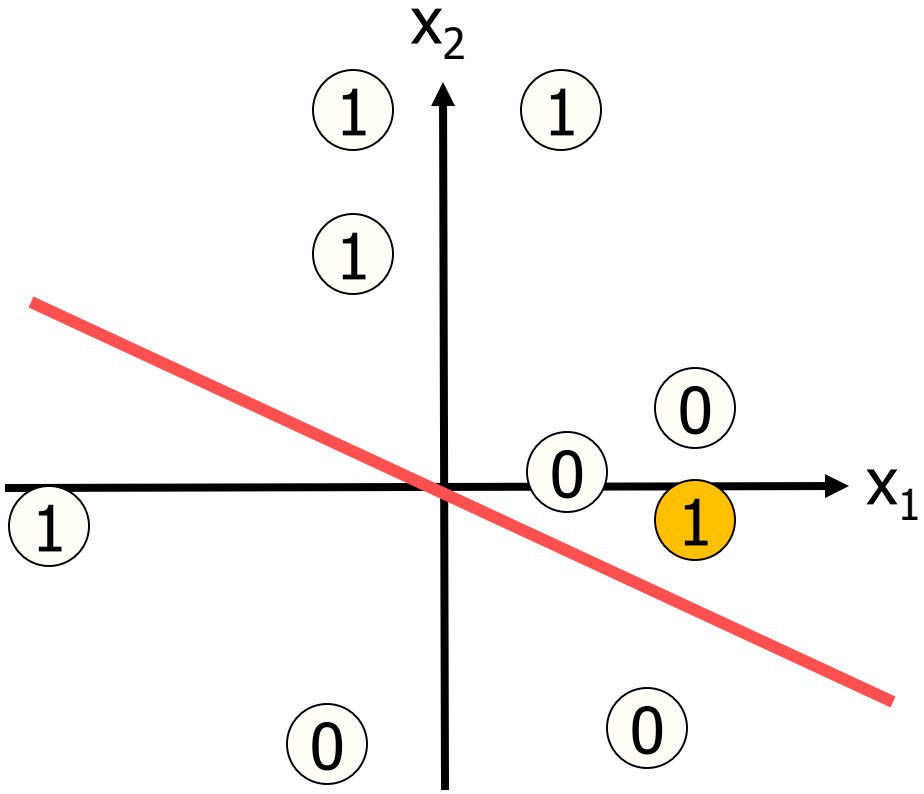
- *Perceptron will not converge to a stable state, but oscilate*

Perceptron: Non-linear separable



- *Perceptron will not converge to a stable state, but oscilate*
- *Solution?*

Perceptron: Non-linear separable



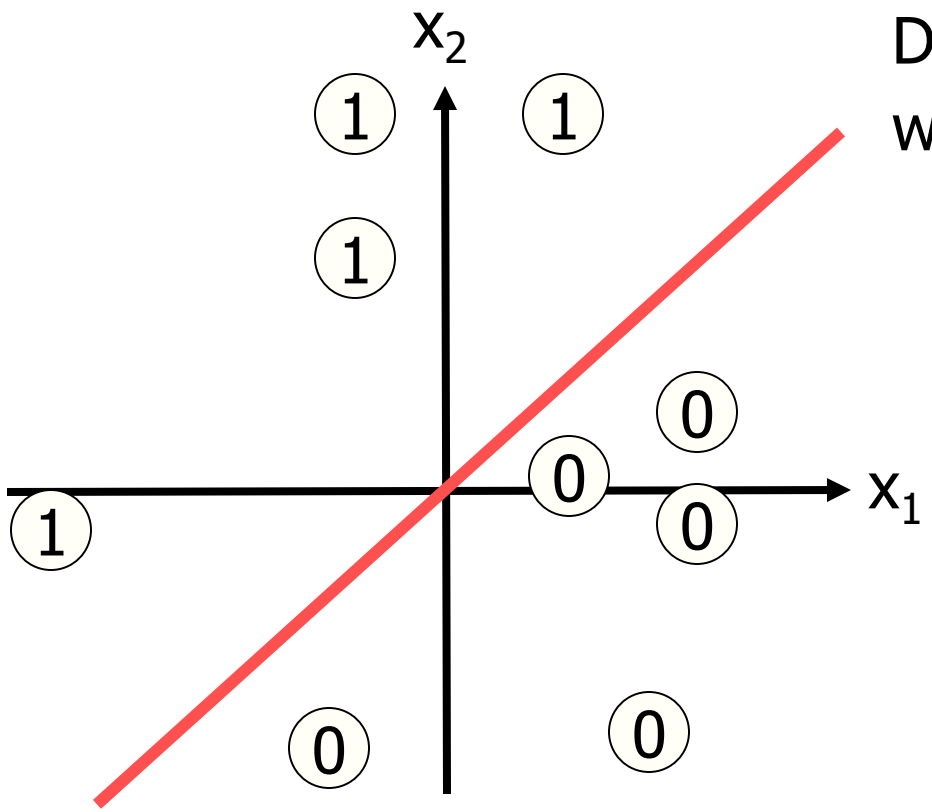
- *Perceptron will not converge to a stable state, but oscilate*

➔ *Need different stopping criterion to end training*

Perceptron: Non-linear separable

- Stopping criteria for training
  - Stop after a maximum number of n training iterations
    - *How to set n?*
  - Stop if there is no more improvement since k iterations
    - Improvement measured e.g. as Accuracy (correctly classified samples)
  - ....
  - *Is that sufficient to find the "best" solution?*
- Pocket algorithm:
  - Keeps the best solution found so far
    - (e.g. the one with highest accuracy)
  - Returns that solution (instead of last state)

Perceptron: bias

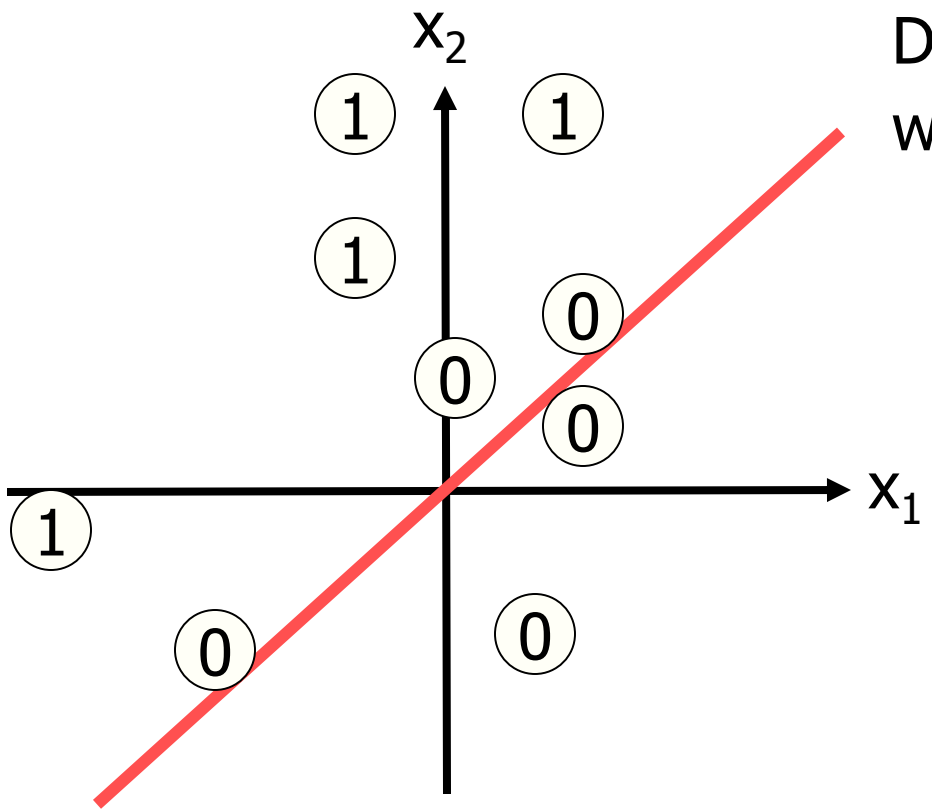


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

(cf. $y = ax$)

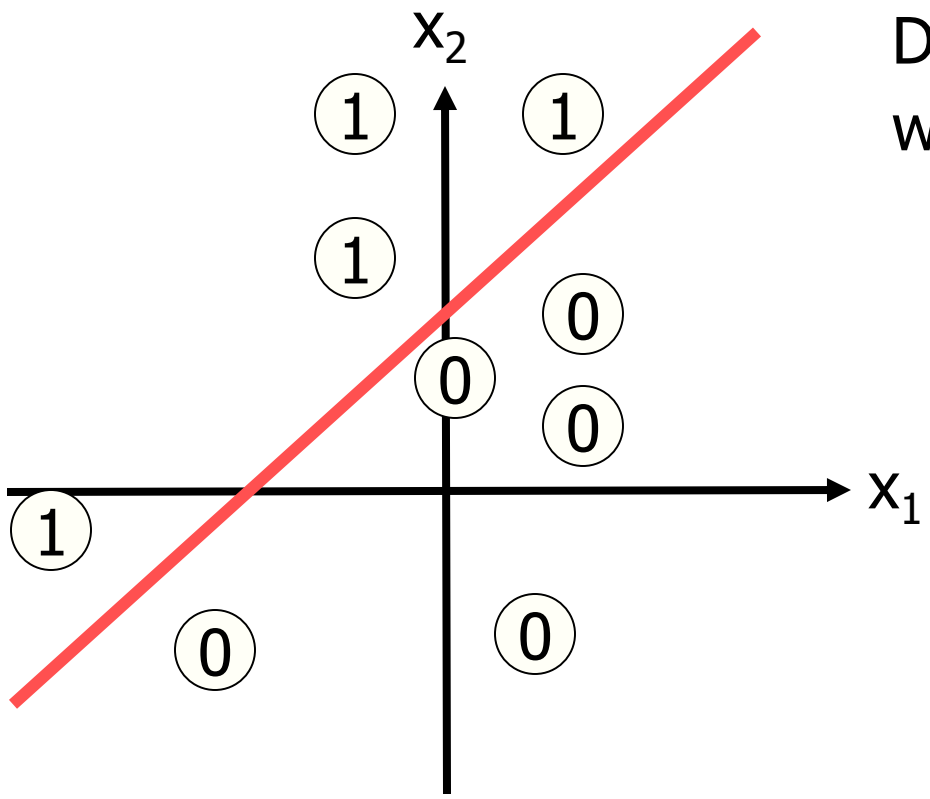
Perceptron: bias



Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron: bias

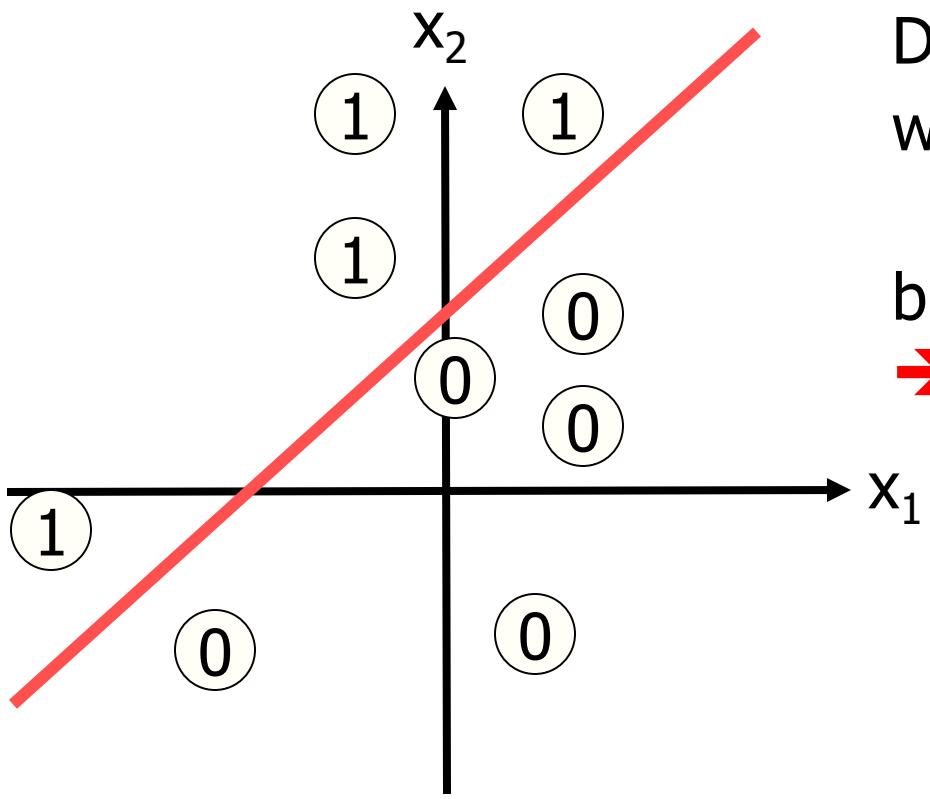


Decision line

$w_1 x_1 + w_2 x_2 + b = \theta$

(cf. $y = ax + b$)

Perceptron: bias



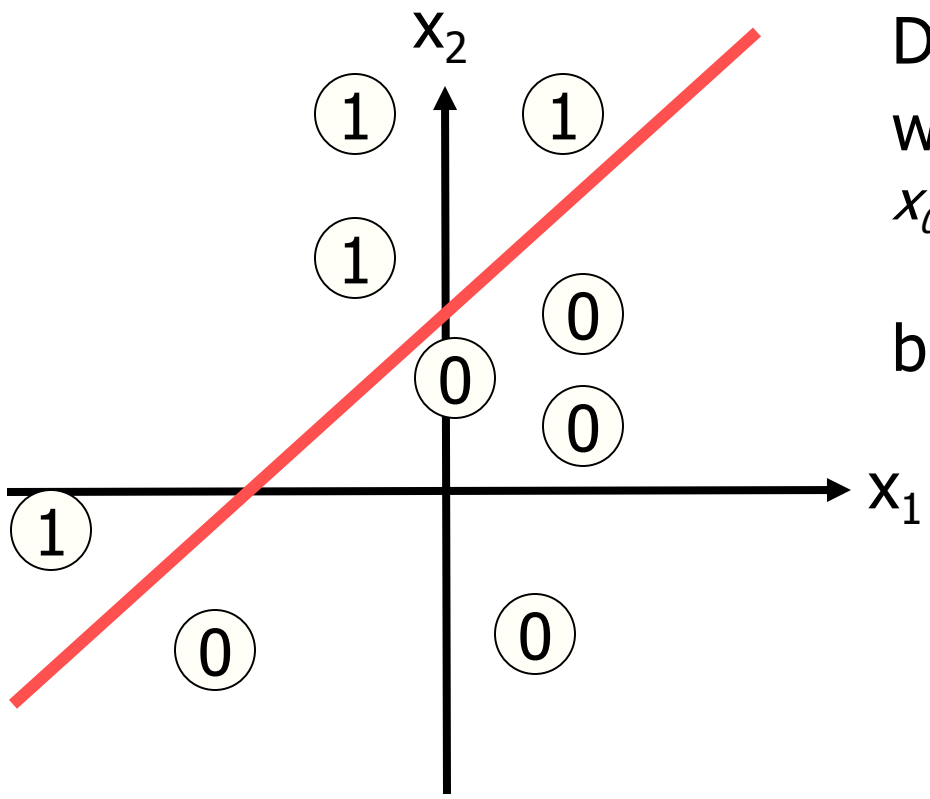
Decision line

$w_1 x_1 + w_2 x_2 + b = \theta$

bias b: constant

➔ how to determine?

Perceptron: bias



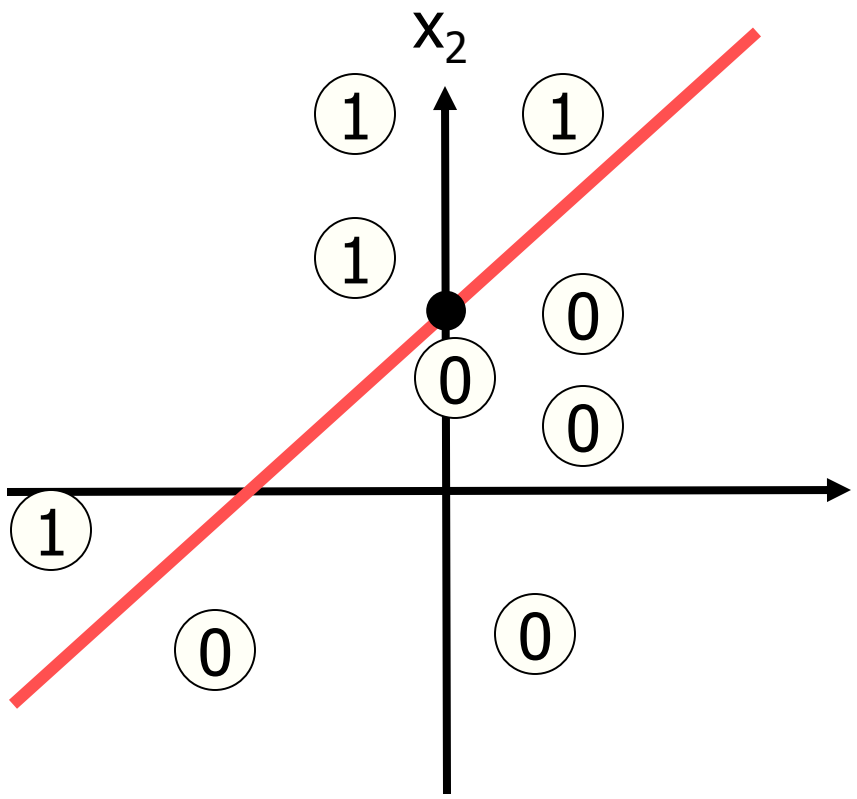
Decision line

$w_0x_0 + w_1 x_1 + w_2 x_2 = \theta$

$x_0 = 1$ (constant)

bias w_0: learned

Perceptron: bias



Decision line

$w_0x_0 + w_1 x_1 + w_2 x_2 = \theta$

$x_0 = 1$ (constant)

W_1=-1, W_2=1, b = -2

Point on decision line

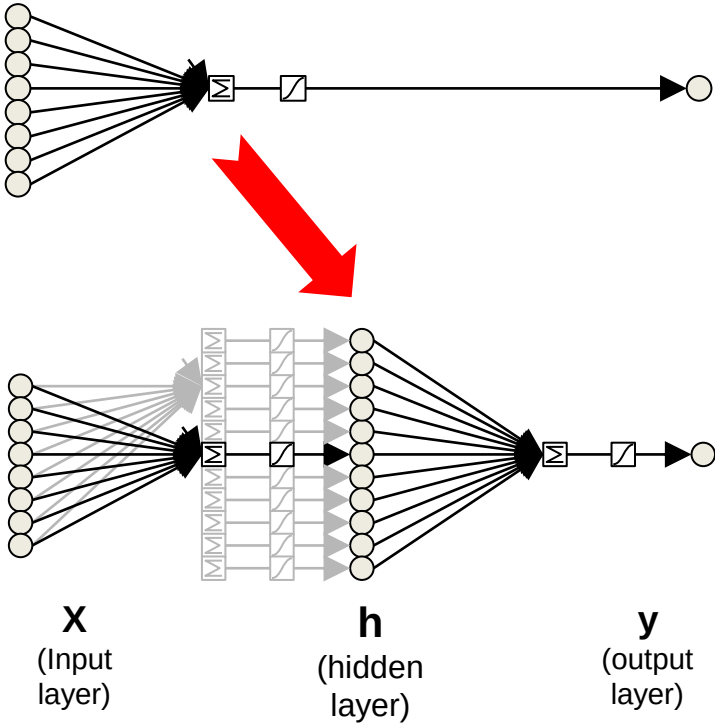
(0,2)

➔-2*1 + -(1)*0 + 1*2 =

= -2 + 0 + 2 = 0 = θ

Extensions to Perceptrons

- Perceptron is a simple, one-neuron neural network
  - Basis for architectures that combine many basic neurons
    - Input layer & output layer
    - *Hidden layer(s)*
  - Initial motivation: model of human brain
    - Neurons connected via synaptic links

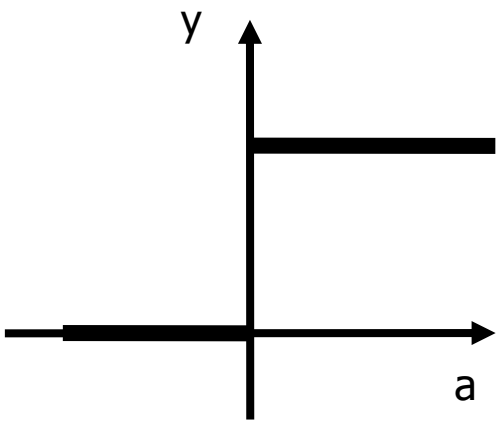


- Extensions for non-linear separable classes
  - Data space projections (Kernels), similar to SVM
    - More on that later ....

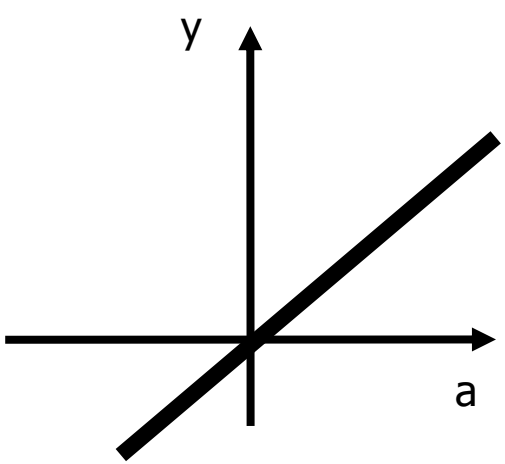
Perceptron: Activation Functions

- Especially in Neural Networks, different activation functions are used

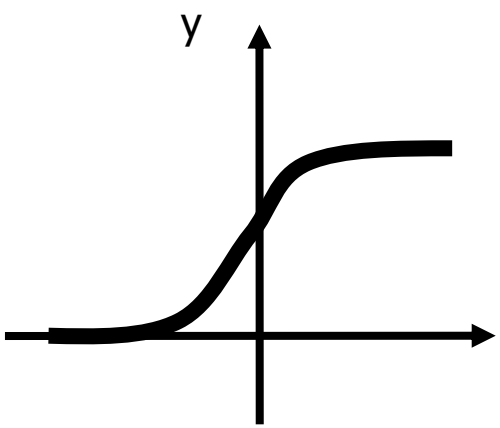
threshold



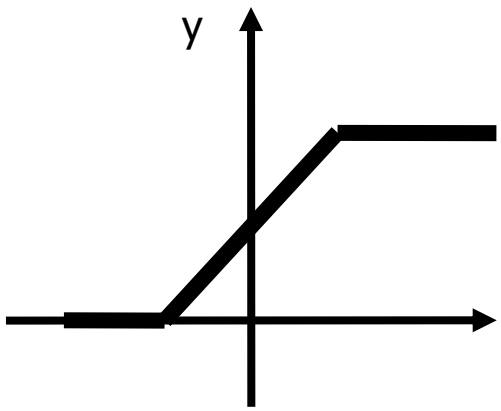
linear



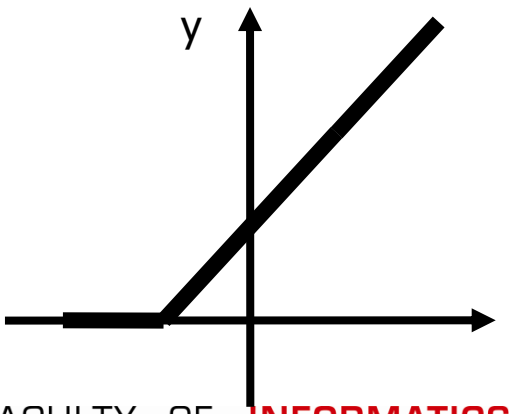
sigmoid



piece-wise linear

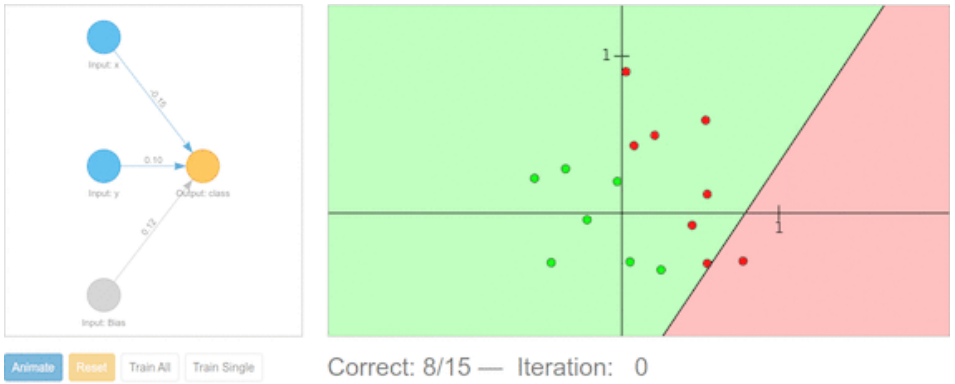


rectified linear (RELU)

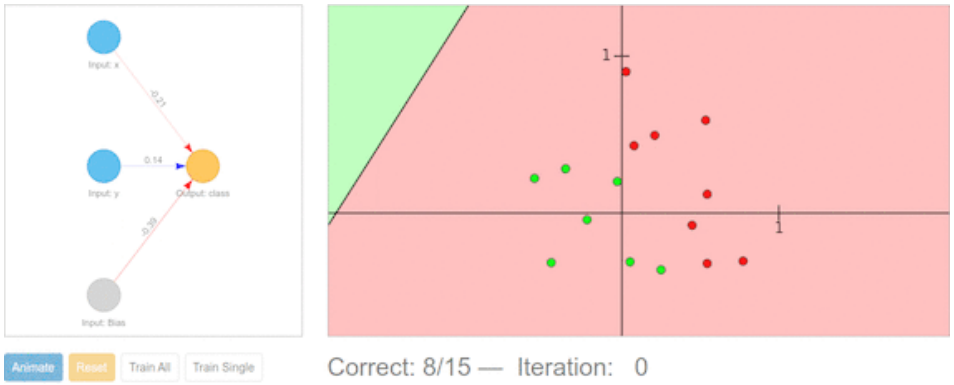


Perceptron: Influence of Learning Rate

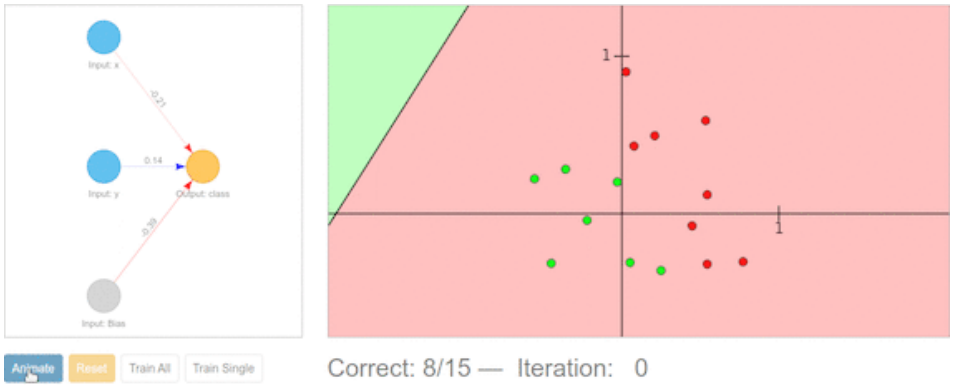
LR = 1.0



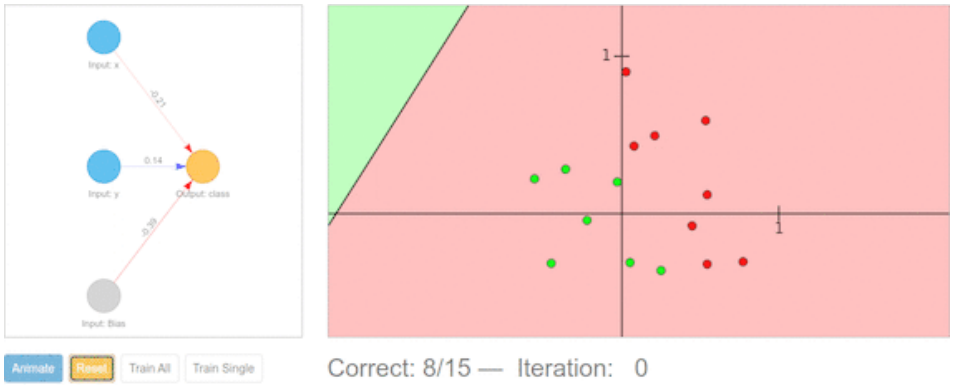
LR = 0.403



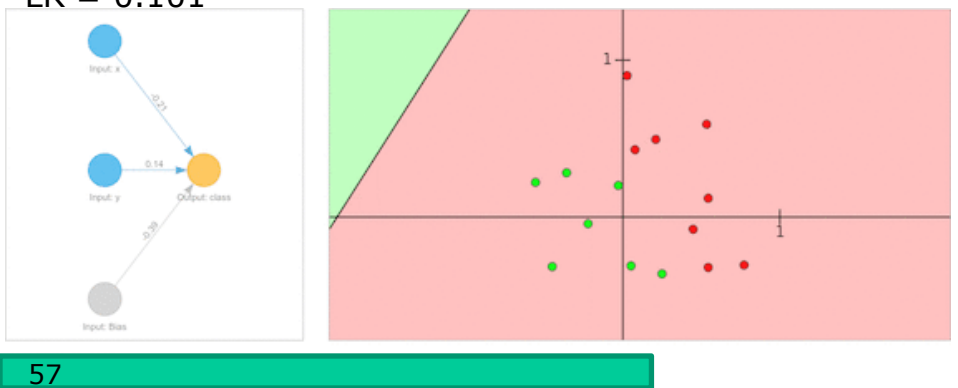
LR = 0.202



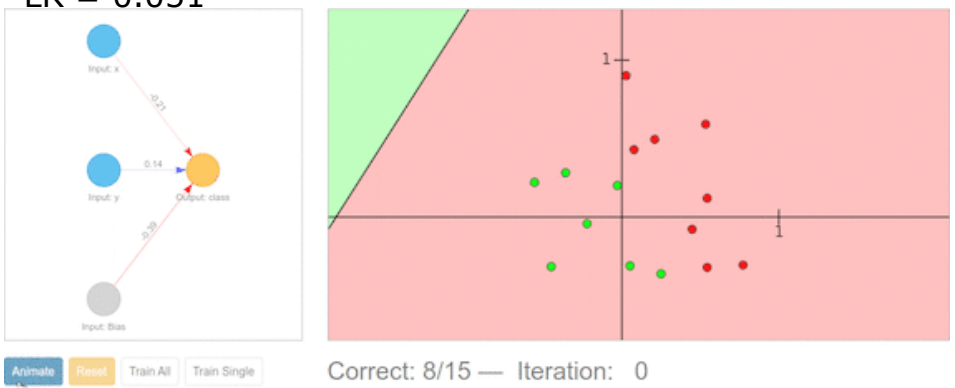
LR = 0.132



LR = 0.101

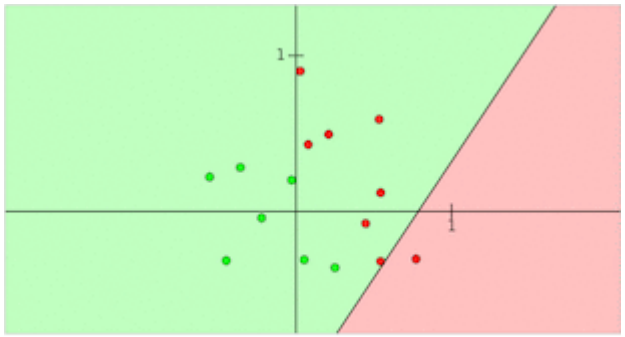


LR = 0.051



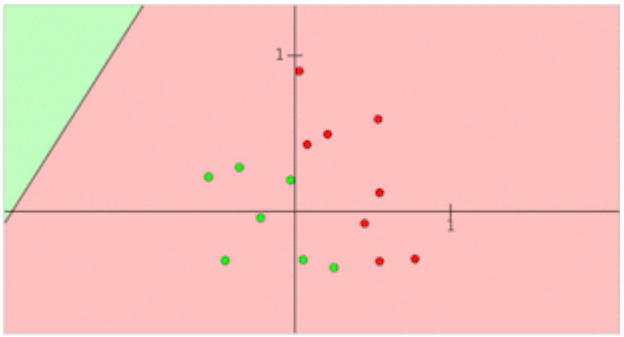
Perceptron: Influence of Learning Rate

LR = 1.0



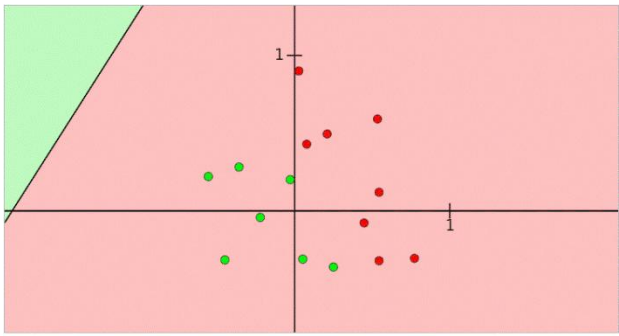
Correct: 8/15 — Iteration: 0

LR = 0.403



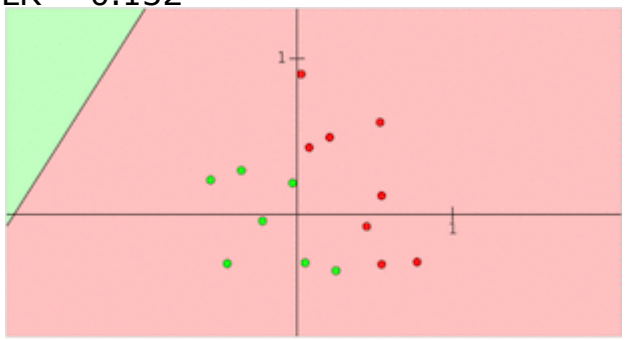
Correct: 8/15 — Iteration: 0

LR = 0.202



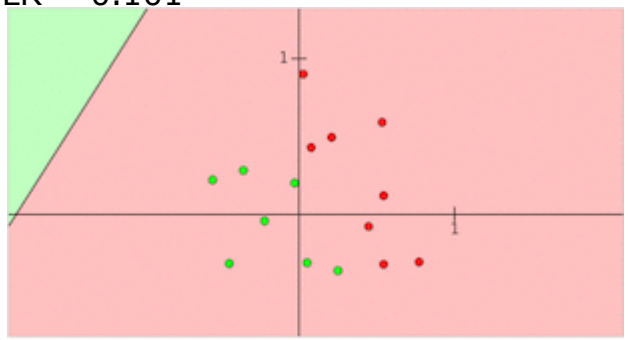
Correct: 8/15 — Iteration: 0

LR = 0.132



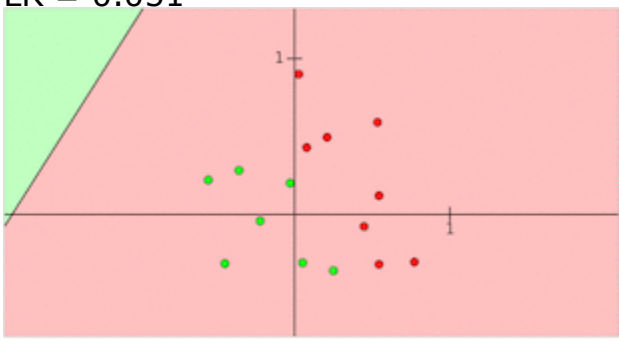
Correct: 8/15 — Iteration: 0

LR = 0.101



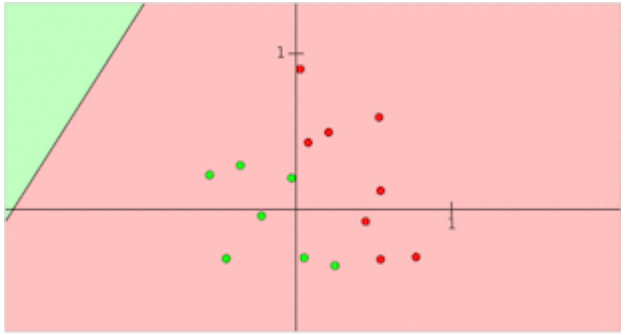
Correct: 8/15 — Iteration: 0

LR = 0.051

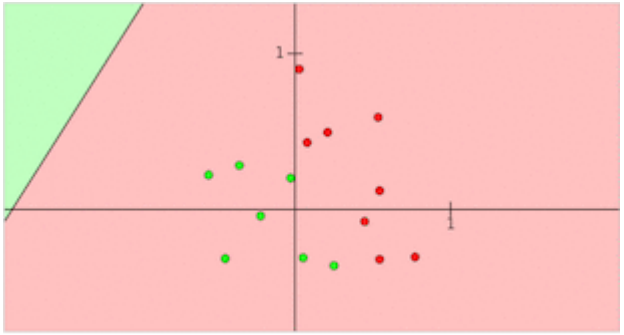


Correct: 8/15 — Iteration: 0

LR = 0.035

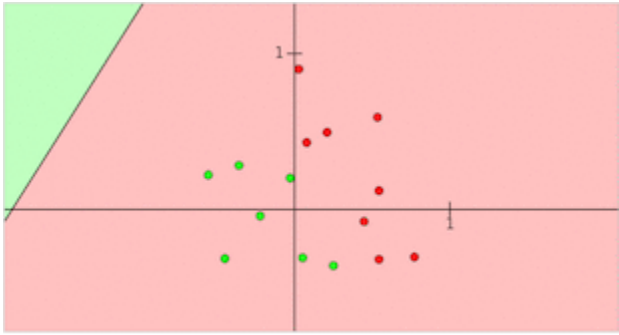


LR = 0.026



Correct: 8/15 — Iteration: 0

LR = 0.011



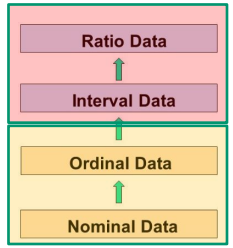
Correct: 8/15 — Iteration: 0

Outline

- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

k-nearest neighbour

- Simple algorithm, well known concepts

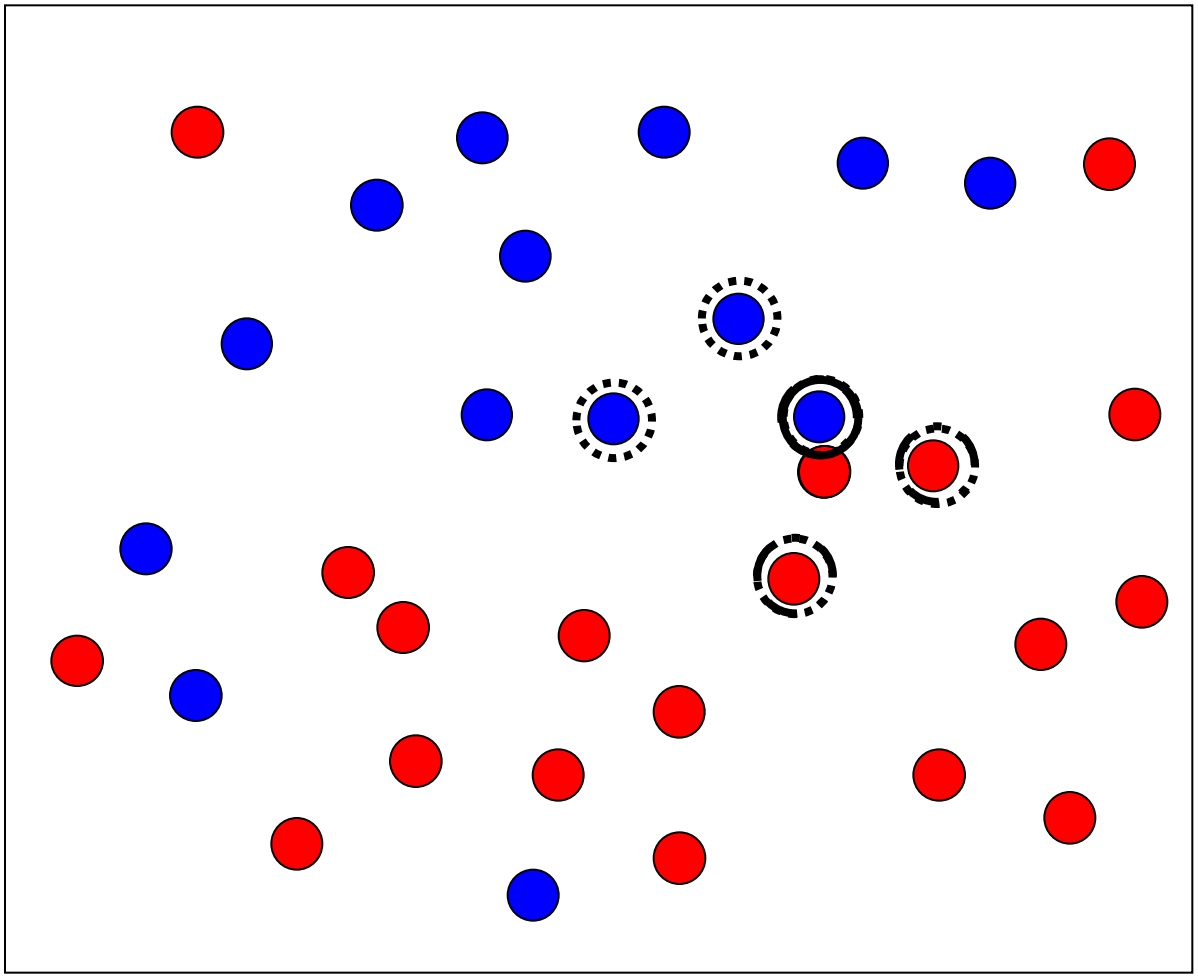


- Classify inputs based on *k* closest training examples

- Two important hyper-parameters
  - *k* to be chosen, high influence
  - Definition of “closest” – distance function

k-NN: Example

- 2-dimensional data, two classes (red & blue)



k-NN – Properties

- Very sensitive to local noise
  - Little abstraction / generalisation
- Larger values of *k* reduce the effect of noise
  - makes boundaries between classes less distinct
- How to determine *k* ?
  - Good values for *k* vary (a lot!) with the data
- *k*=1 is called “nearest neighbour algorithm”

k-NN – Properties

- *What is the maximum value for k?*

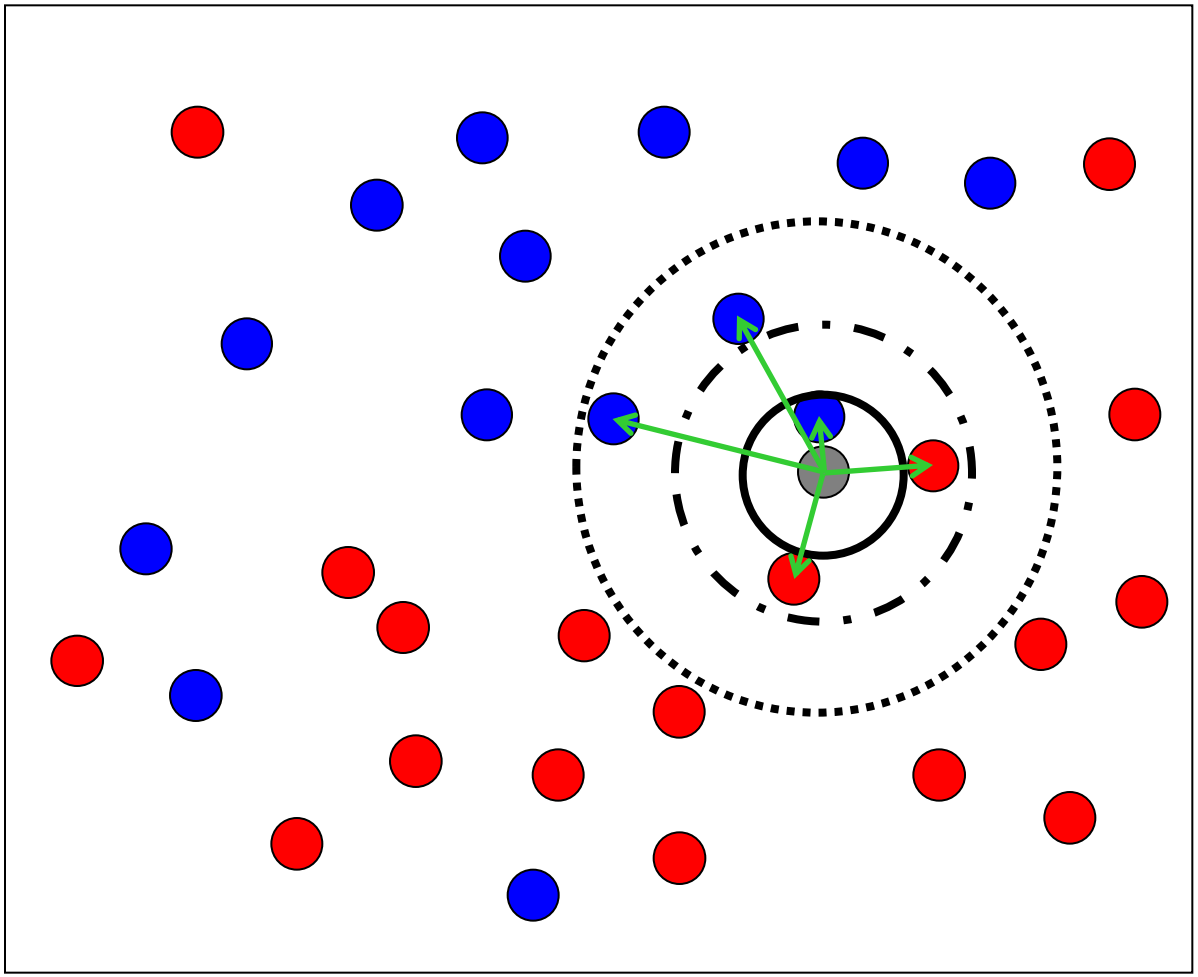
k-NN – Properties

- *What is the maximum value for k?*

- *What happens if k = n (number of samples)*

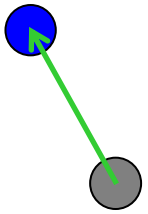
k-NN: distance function

- In this example: Euclidean distance

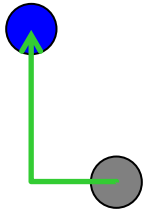


k-NN: distance function

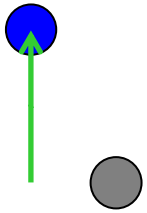
- Previous example: Euclidean distance



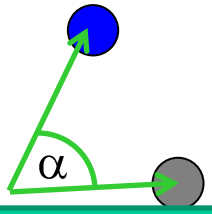
- Any other distance measure for numeric variables can be employed
  - Manhattan/City Block/L1
  - *(Generalised: Minkowski / L_n)*



  - Linfinity



  - Cosine



k-NN vs. Perceptron

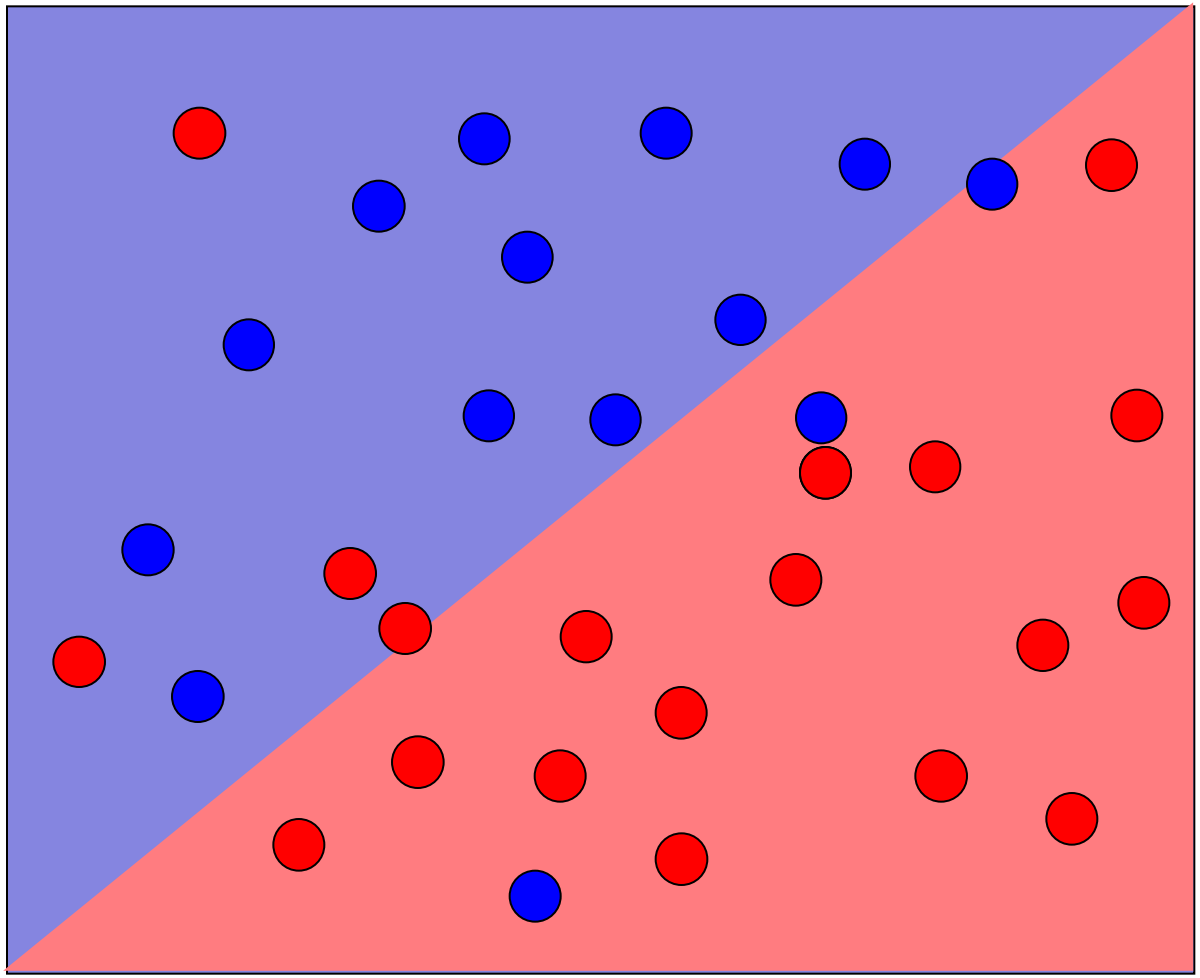
- Major differences between k-NN and perceptron algorithms?

- ***k-NN is a "Lazy learner"***
  - no model built beforehand
    ➔ computation is done at classification step
  - Opposite is called "eager learning"

- k-NN can learn complex (almost arbitrary) **decision boundaries**

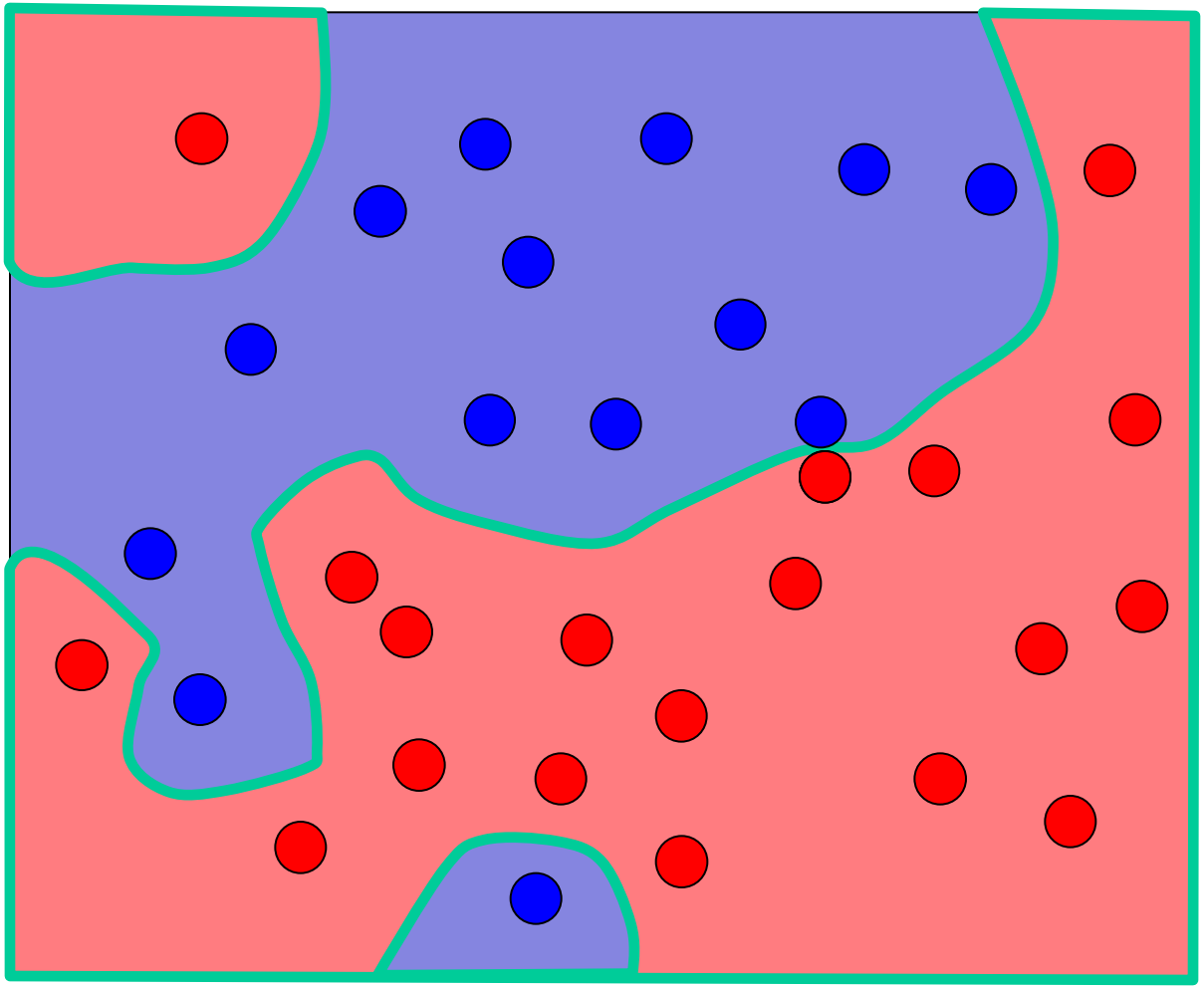
Perceptron: decision boundary

- *Only linear separation!*



k-NN: decision boundary

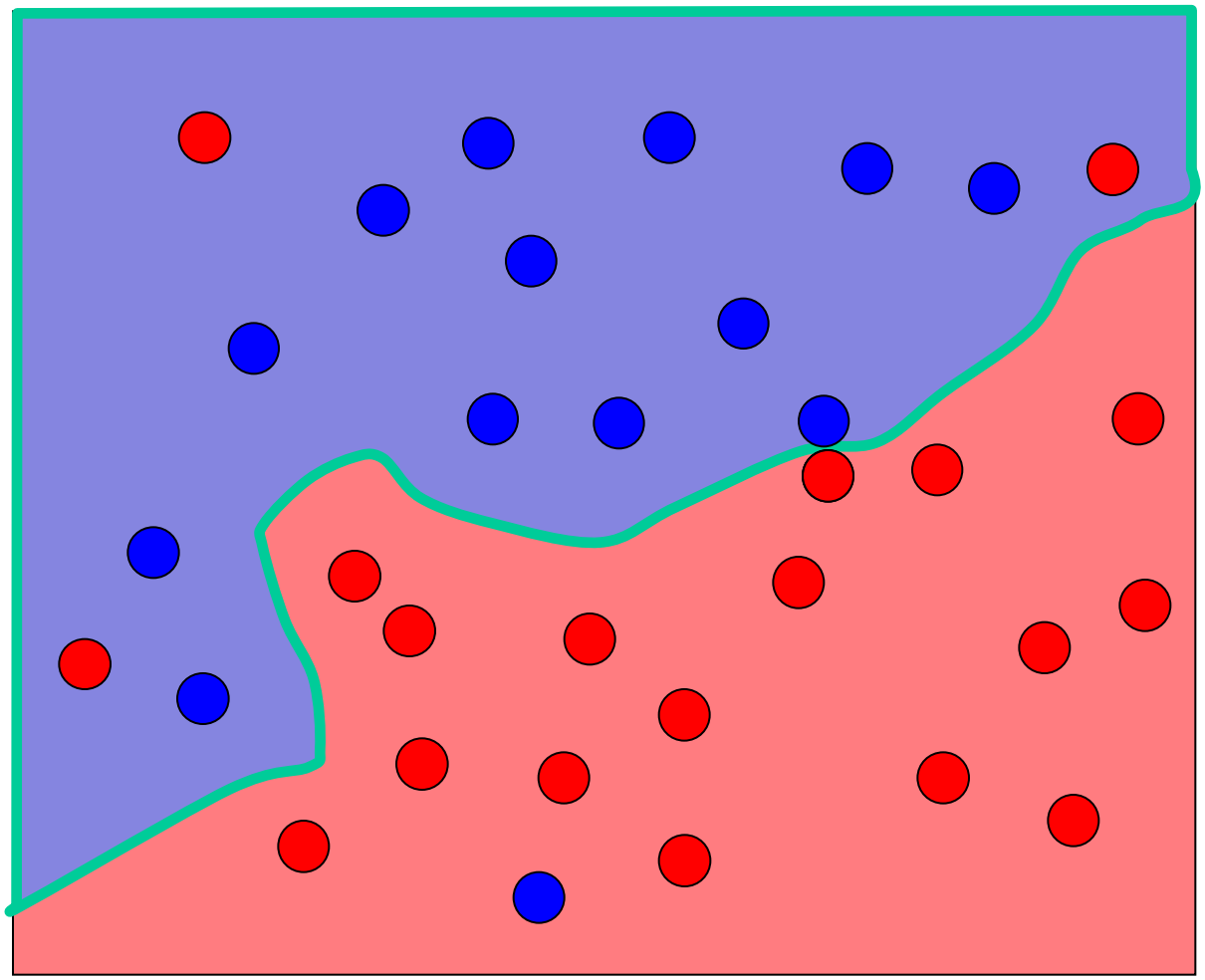
- *(Almost) Arbitrary boundary*



k = 1

k-NN: decision boundary

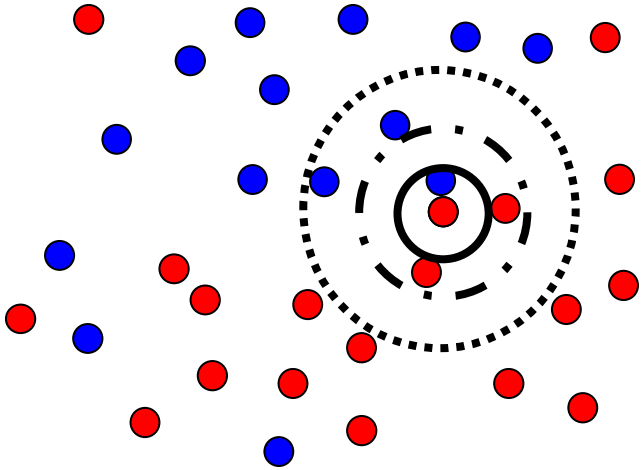
- *(Almost) Arbitrary boundary*



k = 3

k-NN: majority voting

- Simple case: all k neighbours are equally important
  - Majority decides on the class
  - *Is that an issue?*



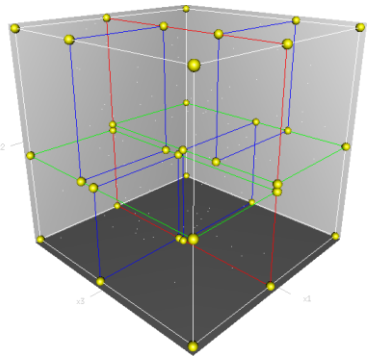
k-NN: majority voting

- Weighted approaches: more influence to closer neighbours
  - Rank weighted: weight determined by rank
    - E.g. each neighbour has a vote 1/r, where r is the rank (1, 2, .., k)
  - Distance weighted: weight determined by distance
    - E.g. each neighbour has a vote 1/d, where d is the (Euclidean) distance from the sample to the neighbour
  - *Differences in these two approaches?*

- *What happens if you chose k=n (number of elements?)*

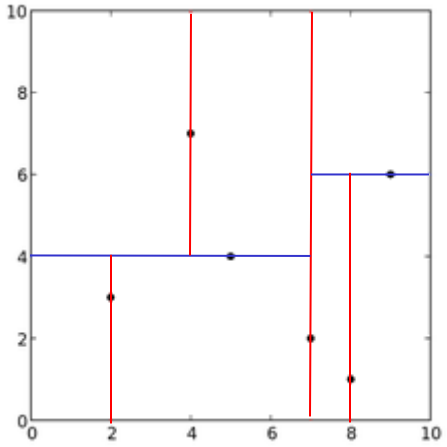
k-NN: other search strategies

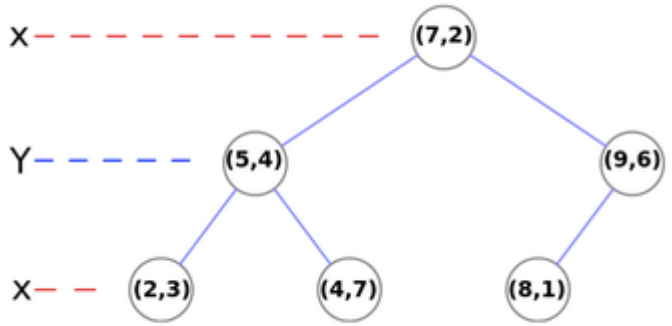
- Becomes computationally expensive with many items to classify
  - Linear (brute-force) search: O (Nd) *N = # samples, d = dimension*
- Search space-partitioning
  - E.g. K-d-Tree
    - Proposed 1975 by Jon Louis Bentley
    - Binary Search Tree (BST)
    - Every node is k-dimensional point
    - Non-leaf node: splitting hyperplane
      - Divides space into two parts



k-NN Search: K-d-Tree

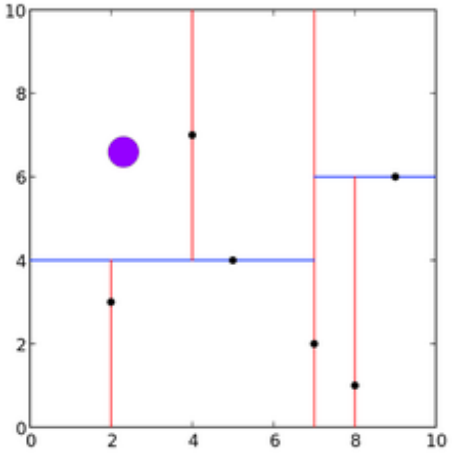
- Recursively split space along points
  - Alternating on the dimensions (axis)

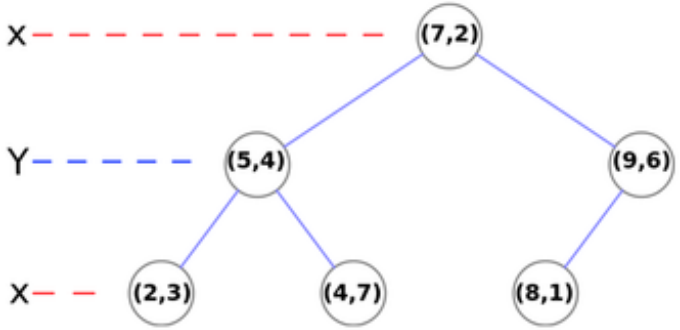




k-NN Search: K-d-Tree

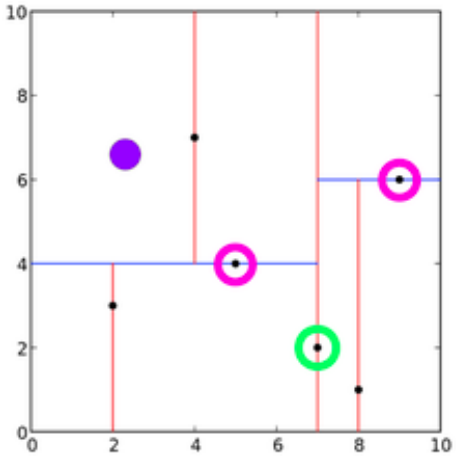
- How to find the neighbours of  ?

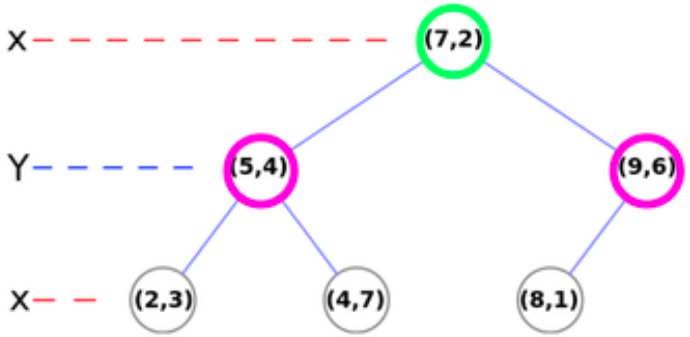




k-NN Search: K-d-Tree

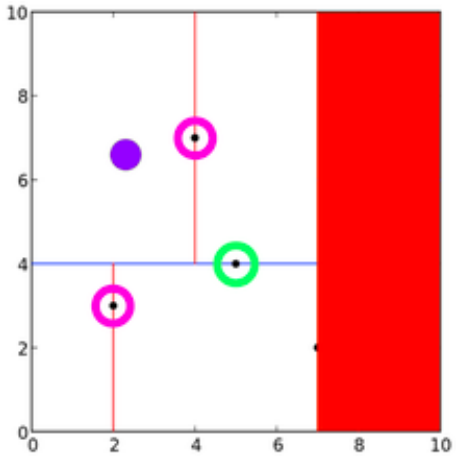
- How to find the neighbours of ● ?
  - Find closest point in first level: (5,4) or (9,6)

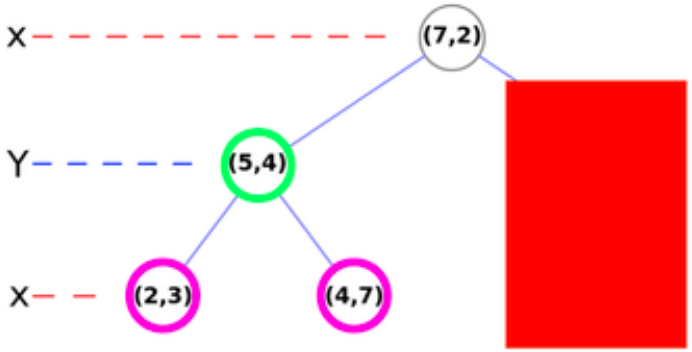




k-NN Search: K-d-Tree

- How to find the neighbours of ● ?
  - Find closest point in first level: (5,4) or (9,6)
    - (5,4) is closer – eliminate other branch
  - Find closest point in second level: (2,3) or (4,7)

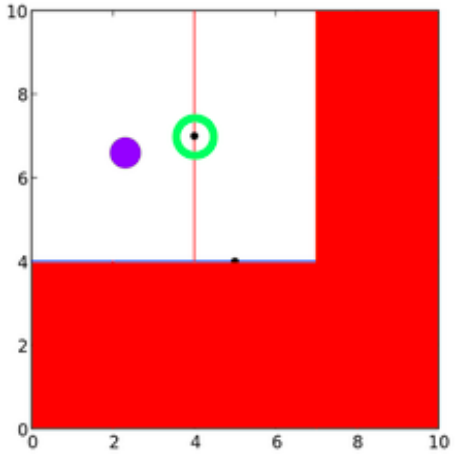


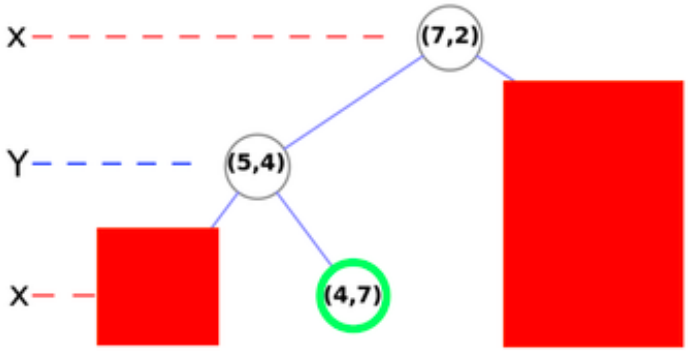


k-NN Search: K-d-Tree

- How to find the neighbours of ● ?
  - Find closest point in first level: (5,4) or (9,6)
    - (5,4) is closer – eliminate other branch
  - Find closest point in second level: (2,3) or (4,7)
    - (4,7) is closer – eliminate other branch

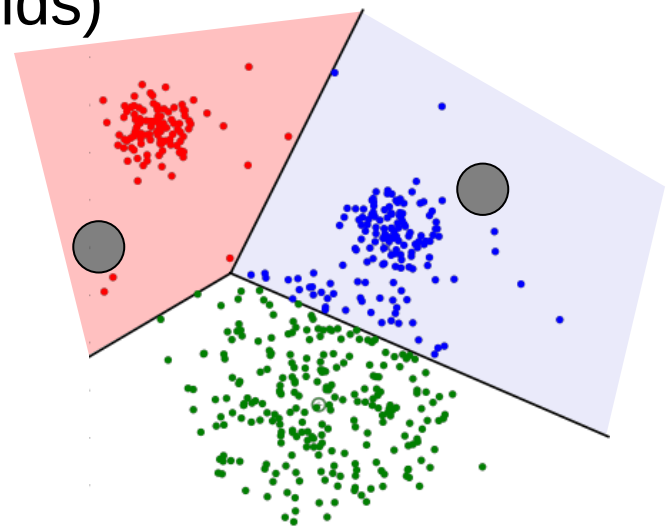
➔ Search in remaining region (might also traverse tree backwards)





k-NN Search: Clustering

- Using Vector Quantisation / Clustering
  - Obtain *prototypes* (cluster centroids)
    - e.g. using k-Means algorithm



  - For a new sample
    1. Find closest prototype (containing cluster)
    2. Then search for neighbours in the same cluster

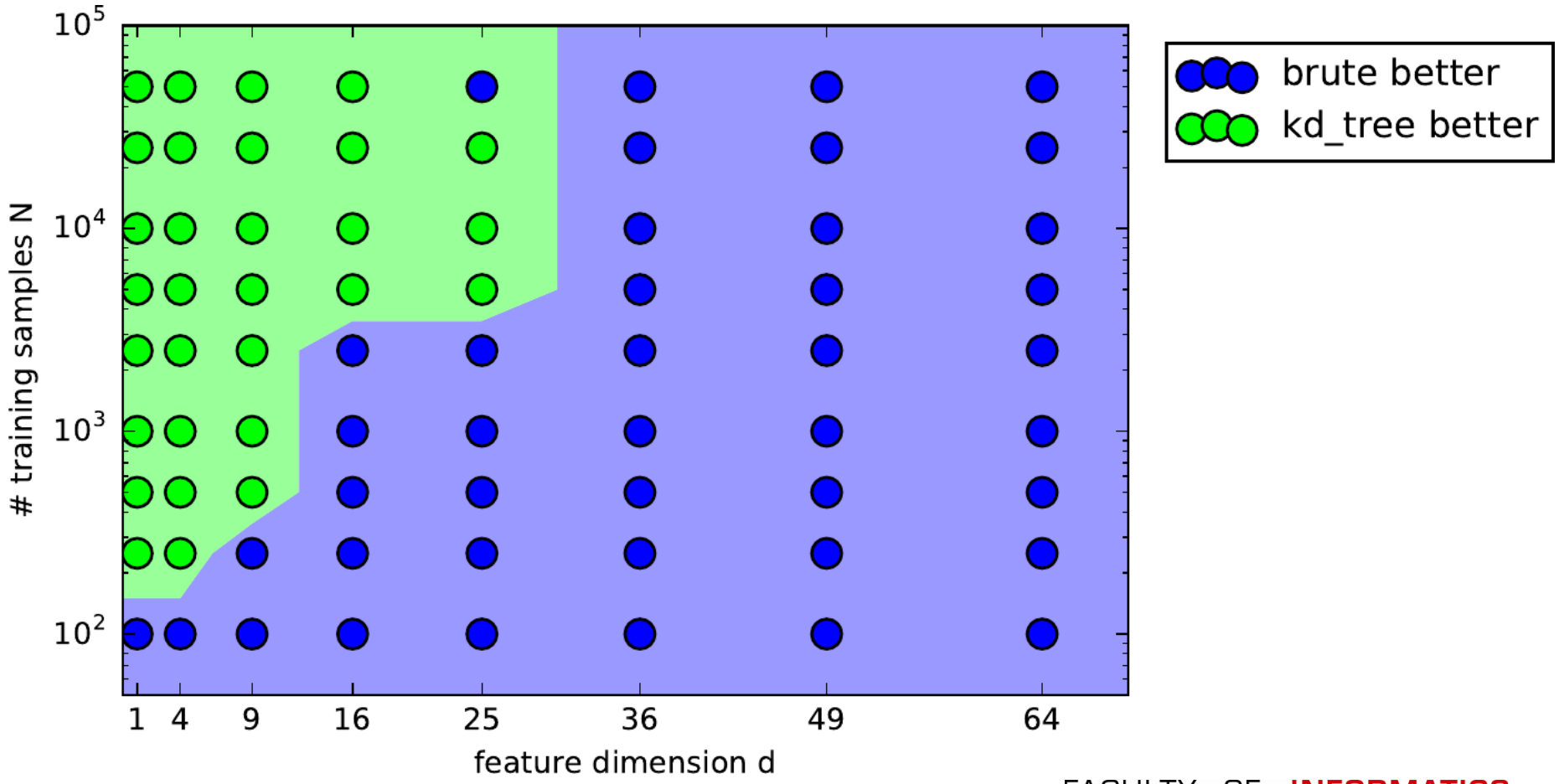
k-NN: other search strategies

- Advantage: can significantly reduce time
  - Practical experiments: up to 10 times
  - Introduces a “training step“ – effort for creating the tree/vector quantisation/....

- *Disadvantages ?*
  - Does ***not*** pay off for small datasets
    - When we consider the time for building the search structure
    - kd-Tree shown to perform worse with high d
  - Depending on the method: may yield different predictions *(why?)*

k-NN: other search strategies

- Kd-Tree evaluation (**combined** training & classification)



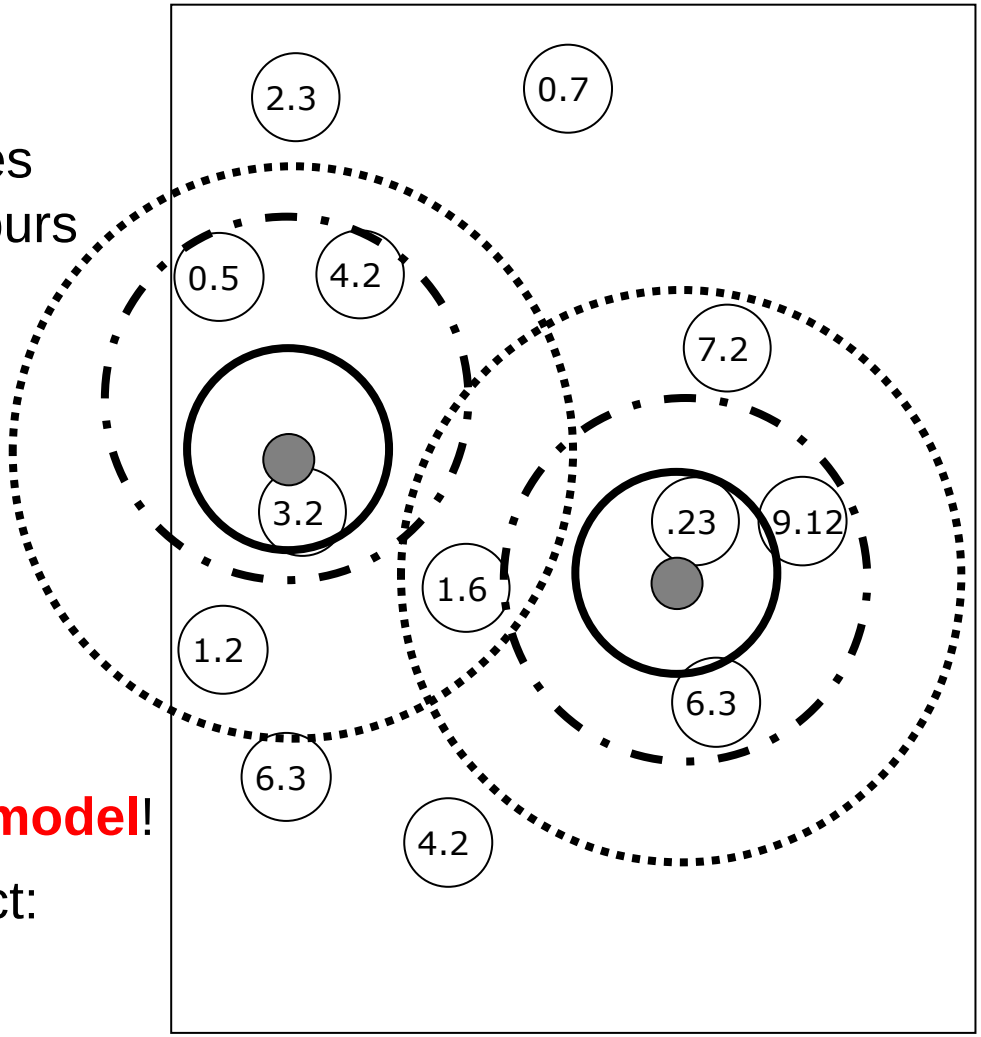
k-NN: Properties

- Easy to understand and interpret
- Easy to implement
- No training time needed…

- Memory requirements depend on training data
- Becomes computationally expensive with many items to classify
  - Linear search: O (Nd), N = # samples, d = dimension
  - Several optimisations proposed

k-NN: regression

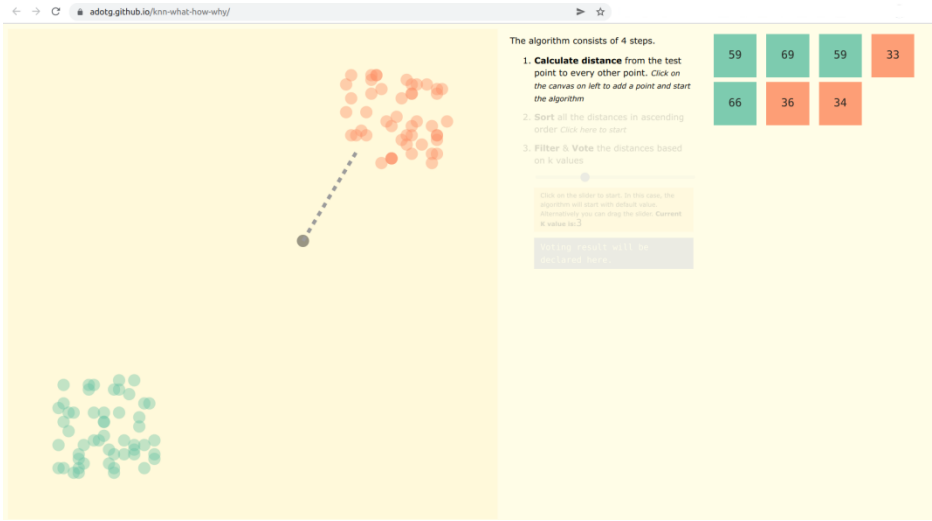
- *Can k-NN be used for regression? (What is regression?)*
- How ?
  - Average of continuous values assigned to nearest neighbours
    - k=1: 0.23 ((0.23) / 1)
    - k=3: 5.22 ((0.23+6.4+9.12) / 3)
    - k=5: 4.89 ((.23+6.4+9.12+1.6+7.2) / 5)
  - Potentially weighted (rank, distance, ...)
- N.b.: lazy learner:
  - we don't build a regression **model**!
  - For each value that to predict: do neighbour search !



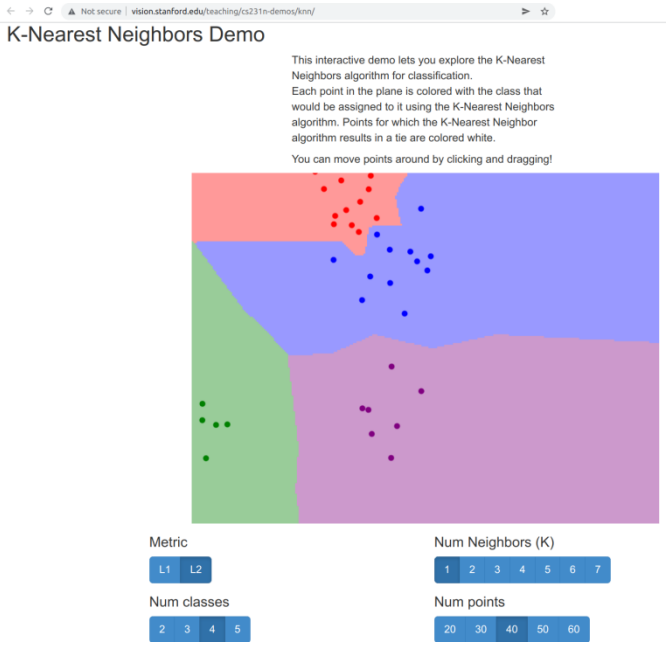
k-NN: Demos

- Interactive demos that illustrate the classification, decision boundary, …

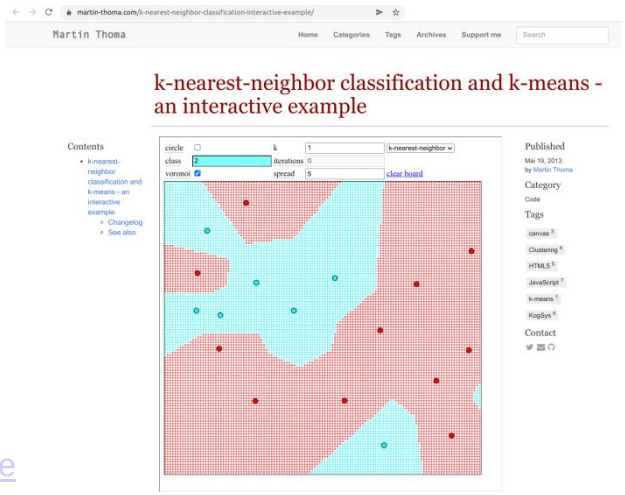
- https://adotg.github.io/knn-what-how-why/



- http://vision.stanford.edu/teaching/cs231n-demos/knn/



- https://martin-thoma.com/k-nearest-neighbor-classification-interactive-example

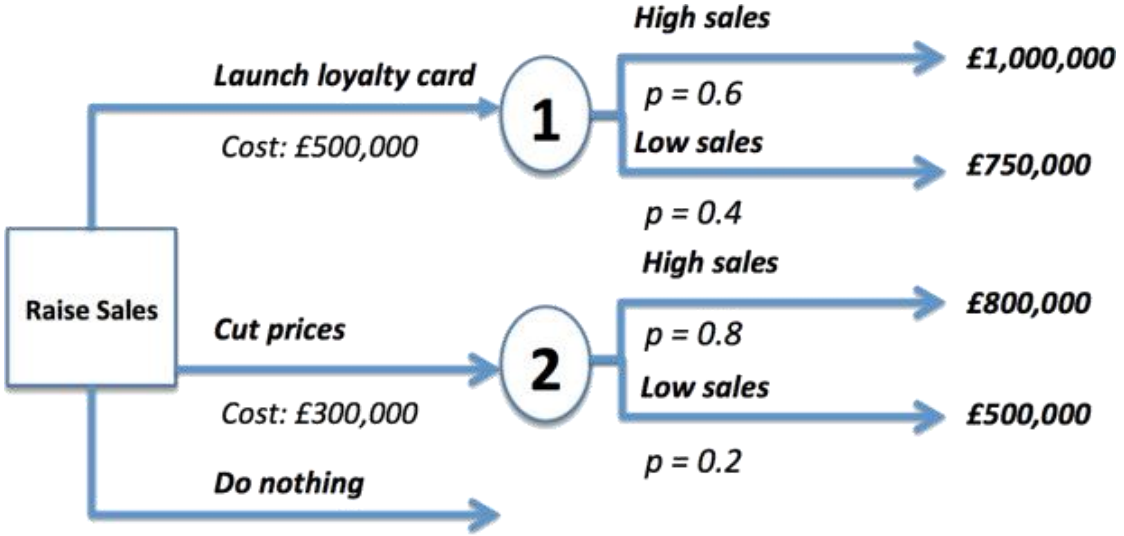


Outline

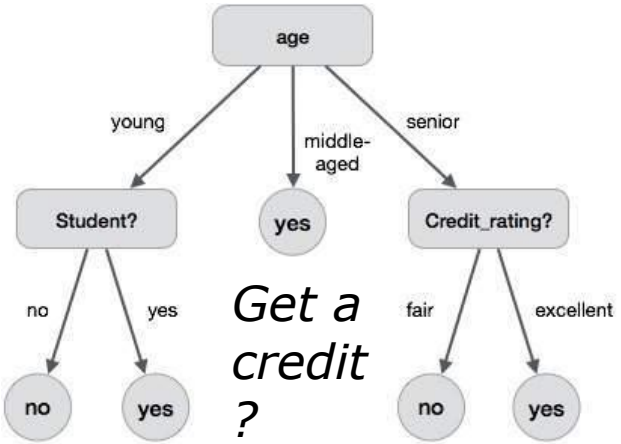
- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees, Episode 1
- Performance evaluation (intro)
- Data preparation (intro)

Decision Trees

- In general: decision support tool
- Tree-like graph, flowchart
- Models decisions and their outcome
  - Potentially including probabilities, costs, …
  - Leaf nodes: events / outcomes
  - Non-leaf nodes: decisions



Decision Trees in ML



- Rather old, simple model

- Easy to interpret model & rules
  ➔ popular in decision support (management, …)

- Classification & regression
- Works with multiple types of data
  - Numerical (size in cm) ***and*** categorical (size: small/medium/large)

- *Simple version?*

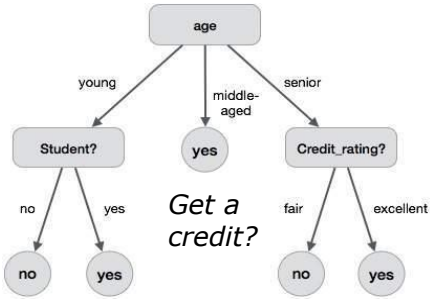
  1R (One Rule; Decision Stump): just one (**root**) node

Simple Baseline Classifier

- *What's even simpler than 1R?*
- ZeroR, Zero Rule
  - Always return the majority class
- *For which purpose is that useful?*
  - As a simple baseline

    ➔Your classifier should be at least as good as that!
- Different names: e.g. DummyClassifier in sk-learn

Decision Trees in ML

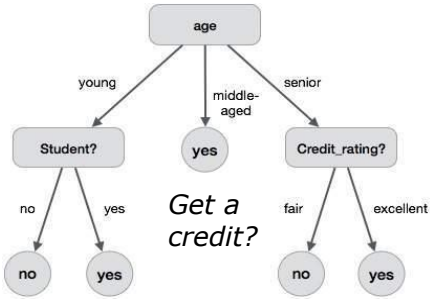


- *Can be manually modelled by experts*
- ML: ***learn*** tree from training data
- Use tree to *predict* classes for unlabeled data

- **Leaf** nodes: define outcome (class / continuous value)

- Inner **nodes**: decisions
  - Based on specific attributes and their values
  - Binary or more (n-ary)
    - I.e. two or more branches from inner nodes

Decision Tree Learning

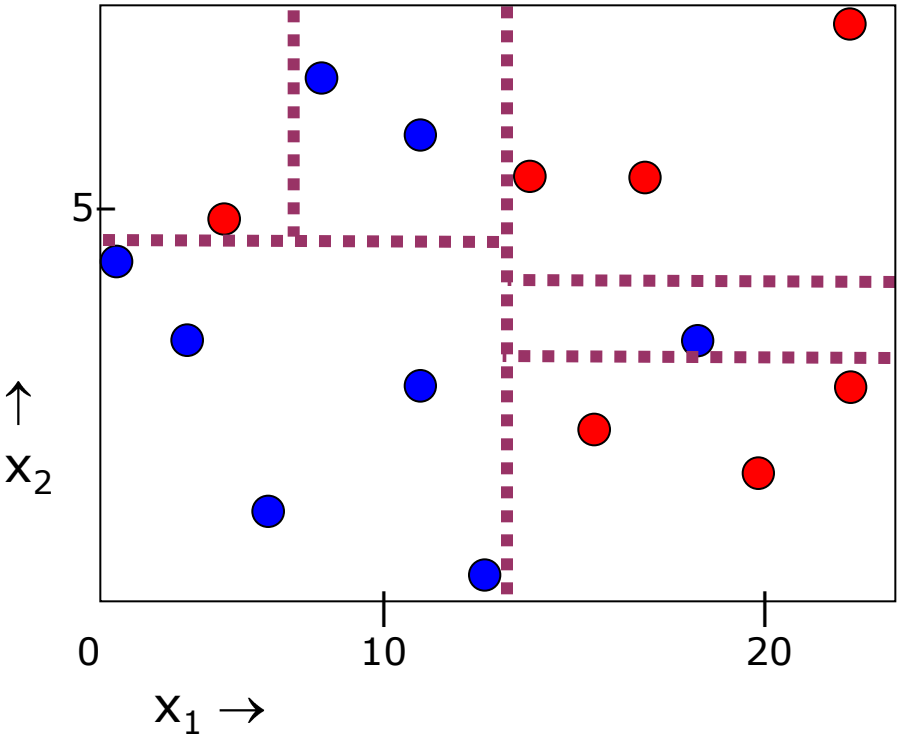


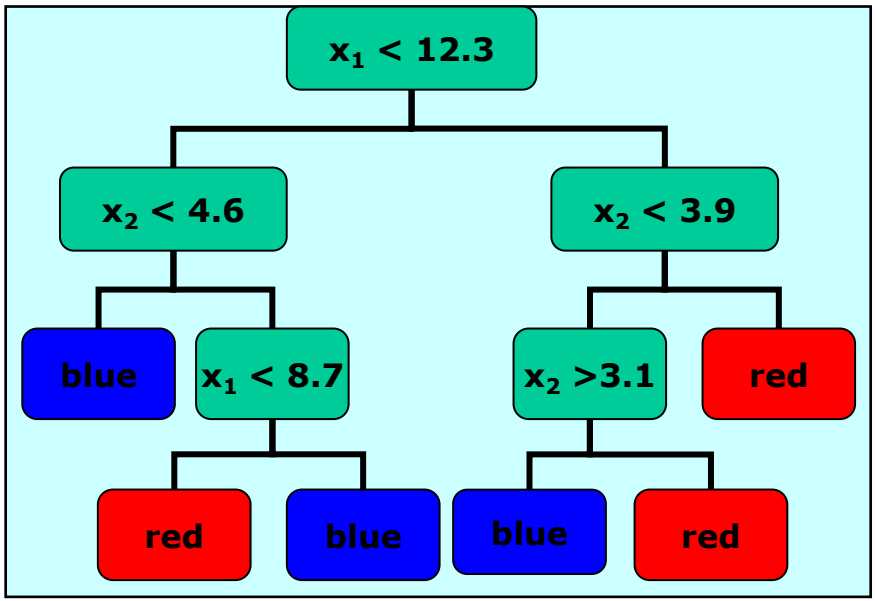
- Training: find rules that recursively **split** data into sub-spaces
  - (Two or more) sub-spaces at each step
  - Until we have reached a stopping criterion
  - Different strategies for evaluating split quality
  - *Well-known algorithms:* CART, ID3 (1986), C4.5 (1996)

- Classification: traverse through tree from root node
  - Until reaching a leaf node
  - Compute prediction based on training data in that leaf node
    - Classification: e.g. majority vote (cf k-nn), i.e. the class that most of the samples in the leaf belong to
    - *Regression: more on that later*

Decision Trees: Example

- Numerical, 2-dimensional data (x_1, $x_{2)}$, two (output) classes
- Training: find rules that recursively split data into sub-spaces (partition)
  - Until each sub-space contains samples from only one class





Decision Trees: Algorithm Outline

- Test data in each leaf node
  - If not all from the same class (or other stopping criterion)
    - For ***each*** attribute
      - Identify possible splits of samples into (two or more) subspaces
    - Compute ***best*** split (over all attributes!)
      - Based on a split goodness measure/criterion
  - Repeat

Decision Trees: Algorithm

- Popular measures to compute best split
  - Error rate
  - Information gain
  - *Gini impurity (Gini index)*
  - *Variance reduction*
  - *...*

Decision Trees: Algorithm

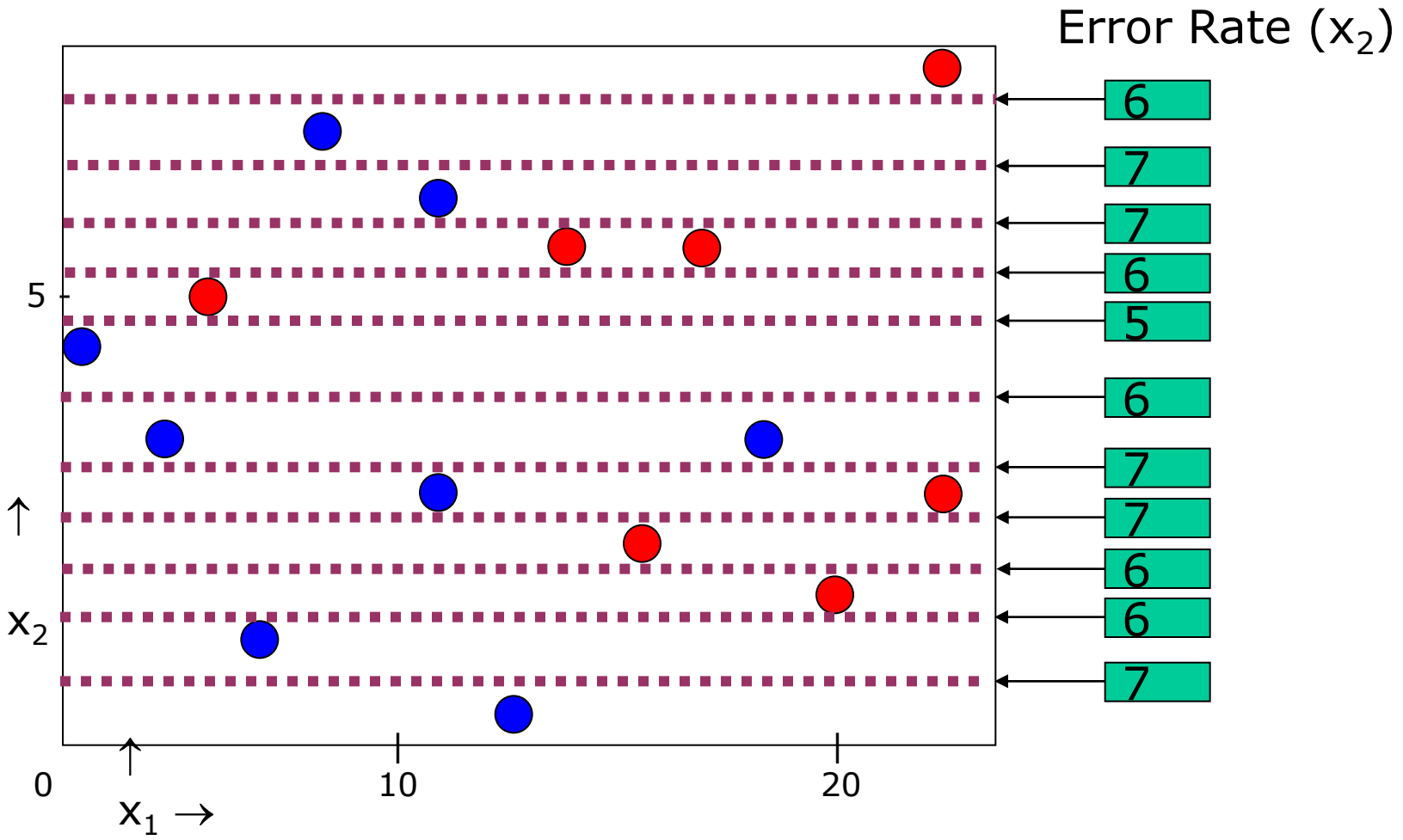
- Popular measures to compute best split
  - Error rate
  - Information gain
  - *Gini impurity (Gini index)*

Decision Trees: Algorithm

- Error rate
  - Absolute error rate: simply count the number of classification errors when performing a split
  - The lower the error, the better
  - Absolute error rate vs. relative error rate
    - Relative error in relation to total number of samples (0..1)
    - Absolute error: 0..n
      - *Semantic difference?*

Decision Trees: error rate

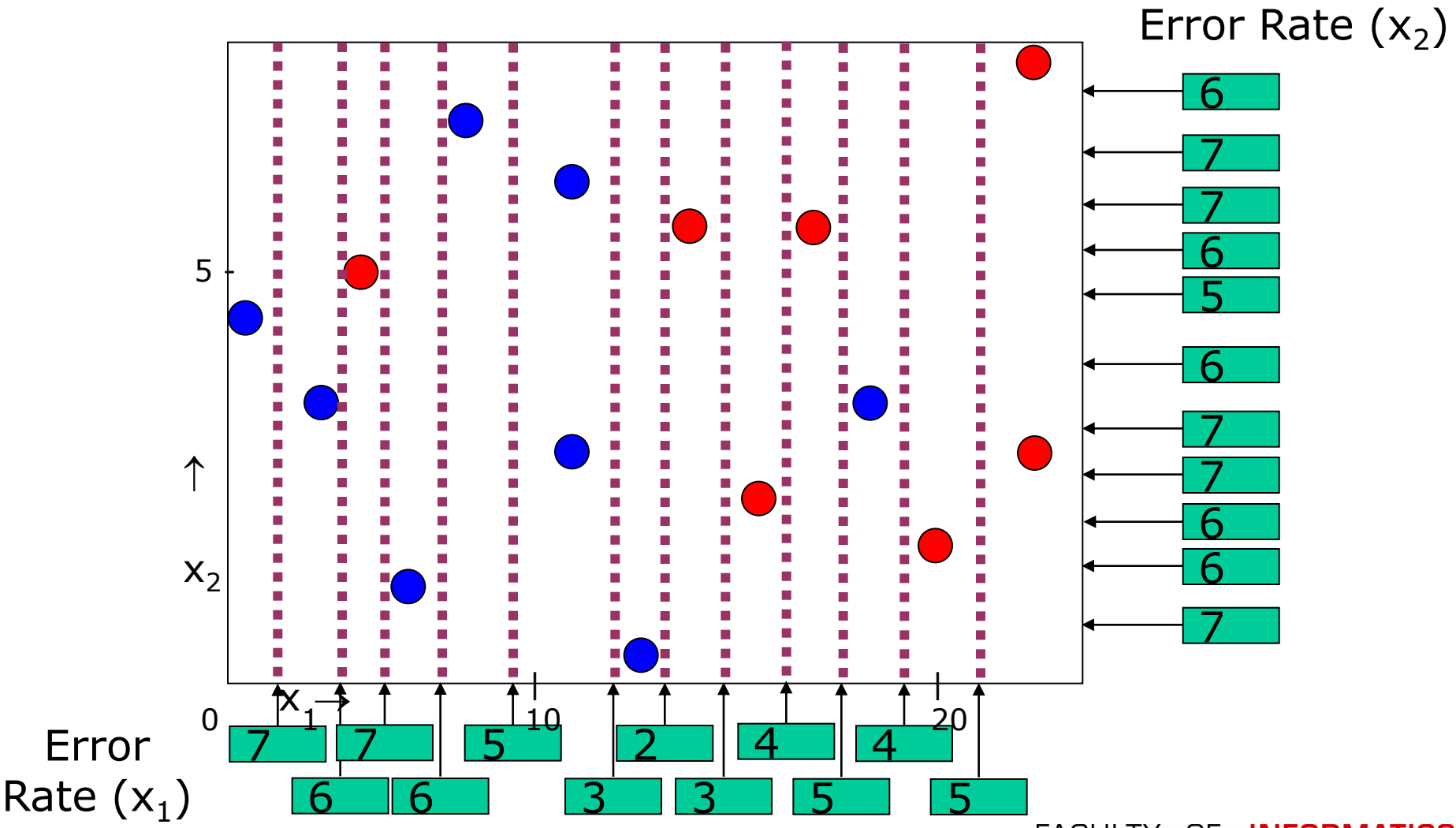
- 2-dimensional data (x_1, x_2), numerical values, two classes; *how to split?*



- Find all possible splits, compute error; *how many splits in this example?*

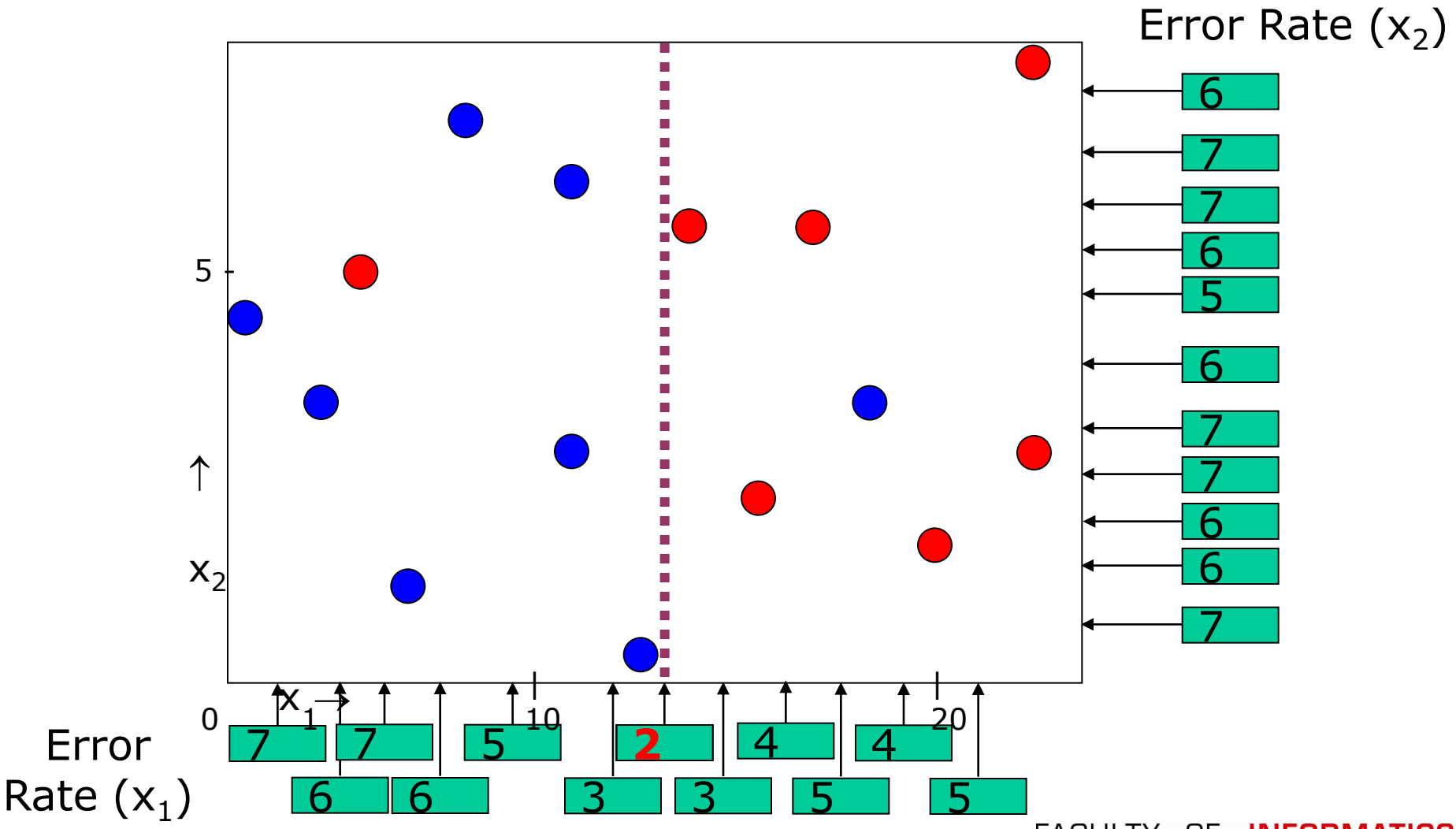
Decision Trees: error rate

- 2-dimensional data (x, y), numerical values, two classes

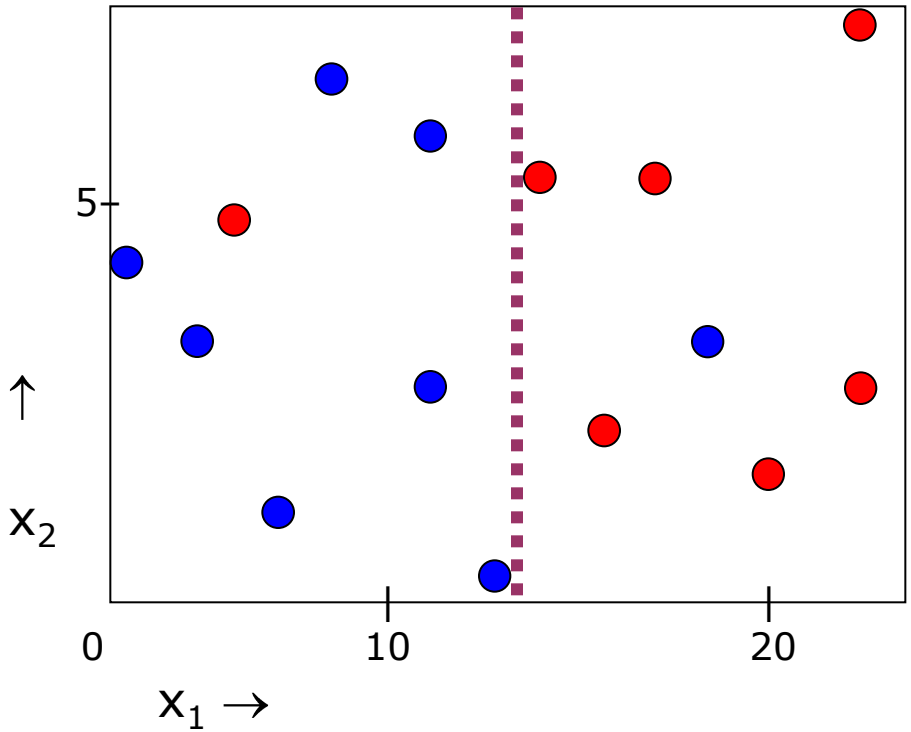


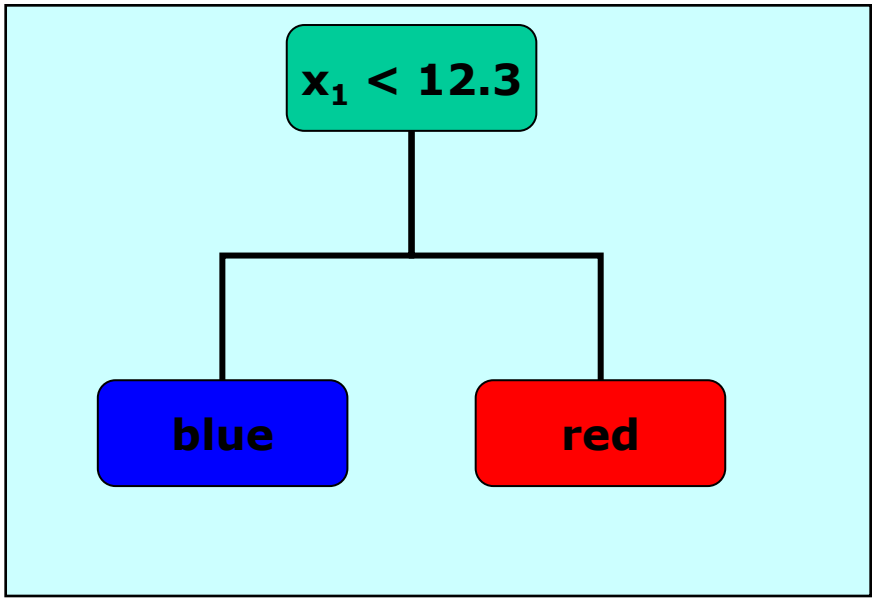
Decision Trees: error rate

- 2-dimensional data (x, y), numerical values, two classes



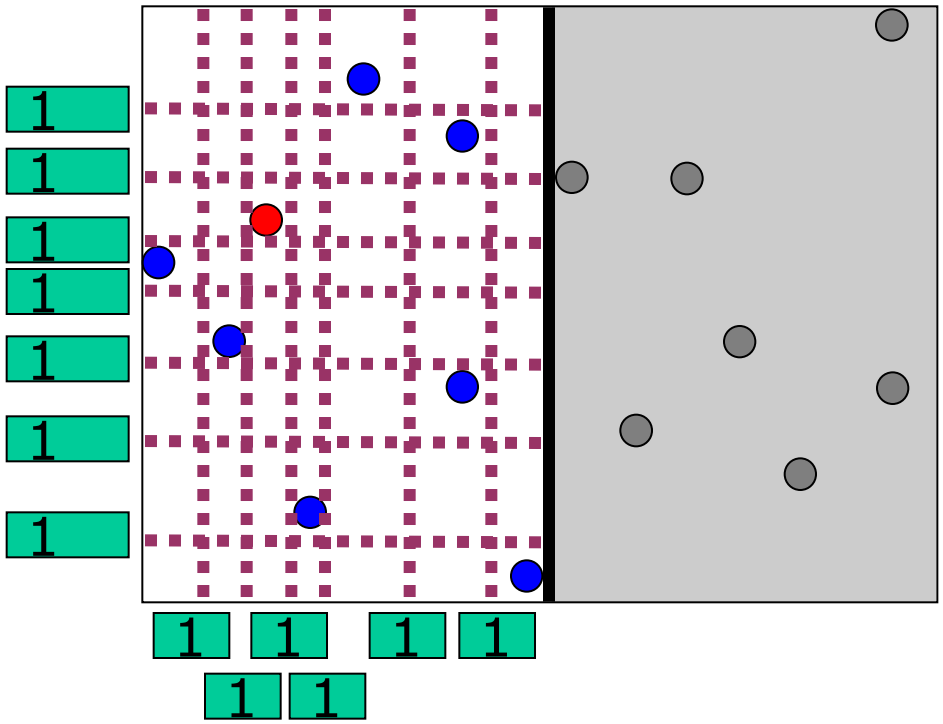
Decision Trees: error rate

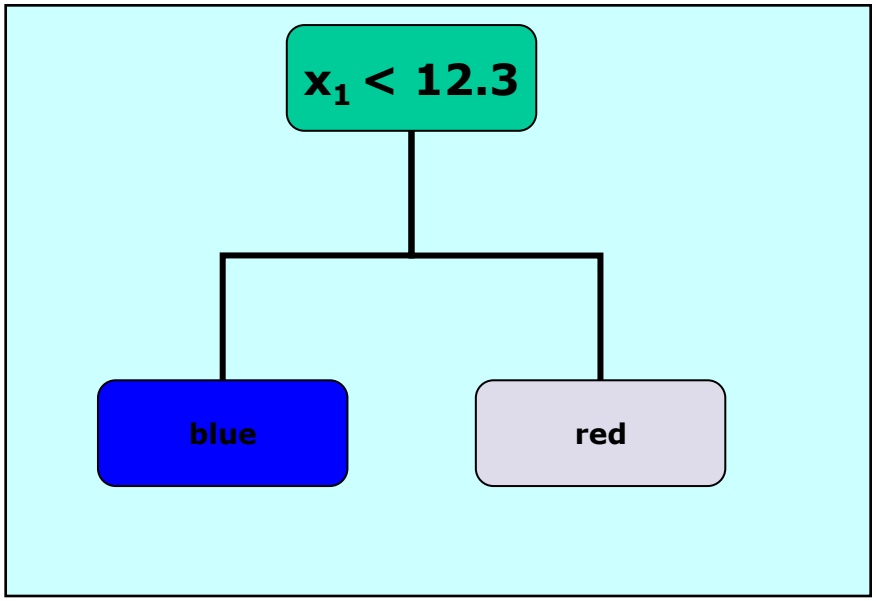




- Split for top-level (root) node found ✓
- *If we stop here – how is this classifier called?*
- *Next step(s)?*

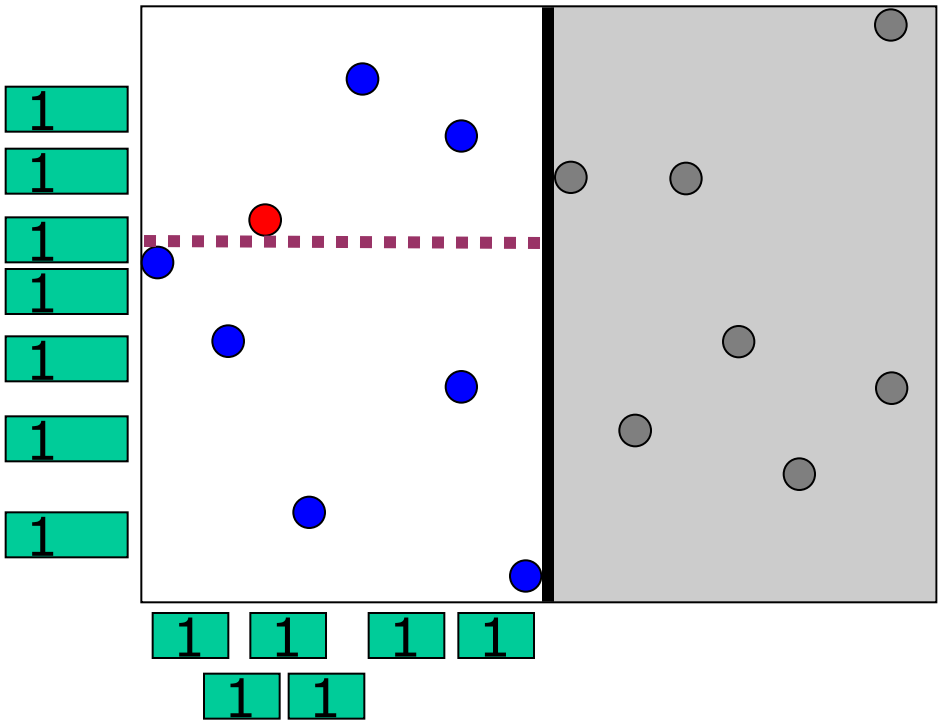
Decision Trees: error rate

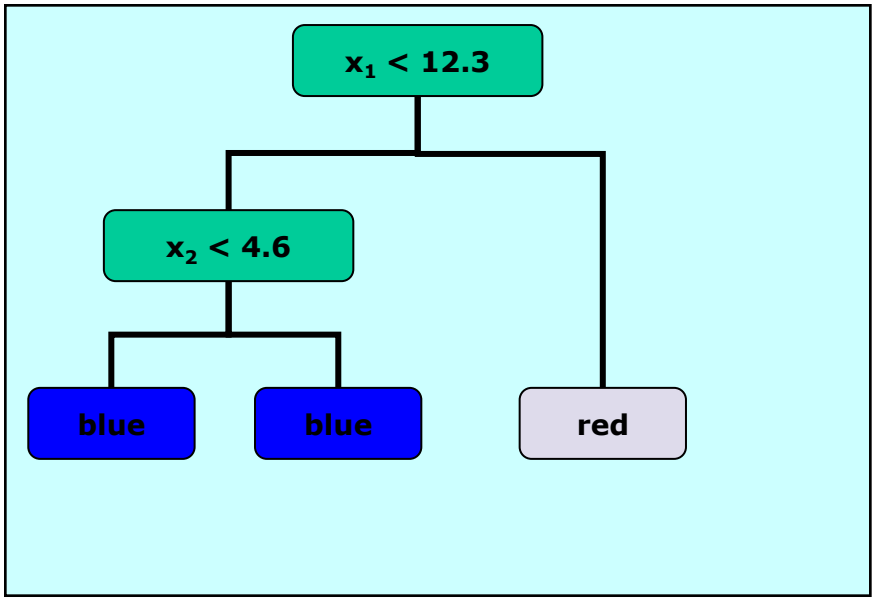




Which split?

Decision Trees: error rate

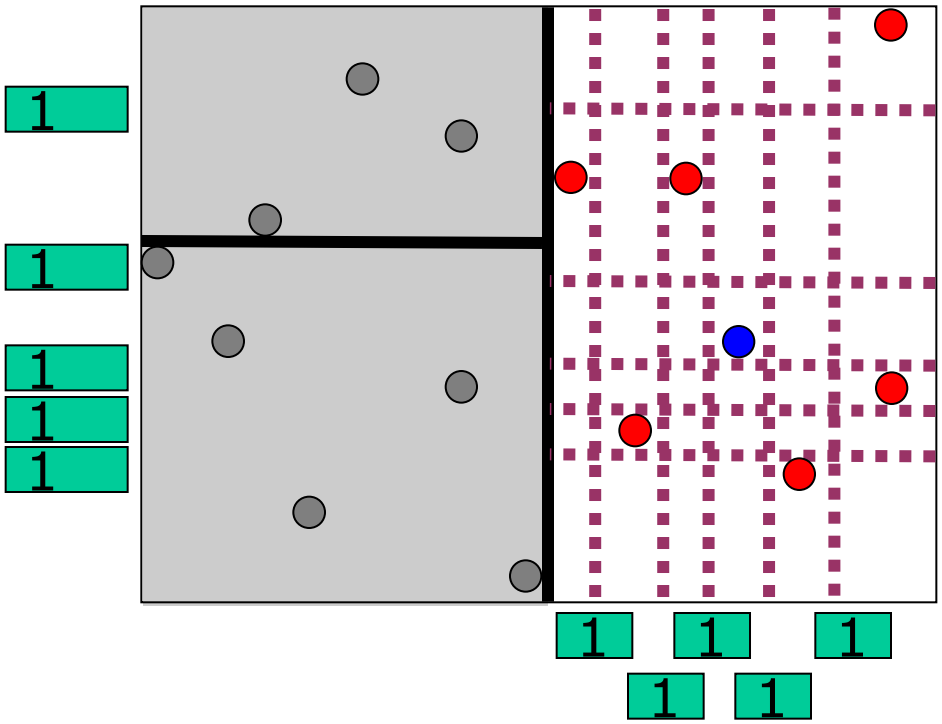


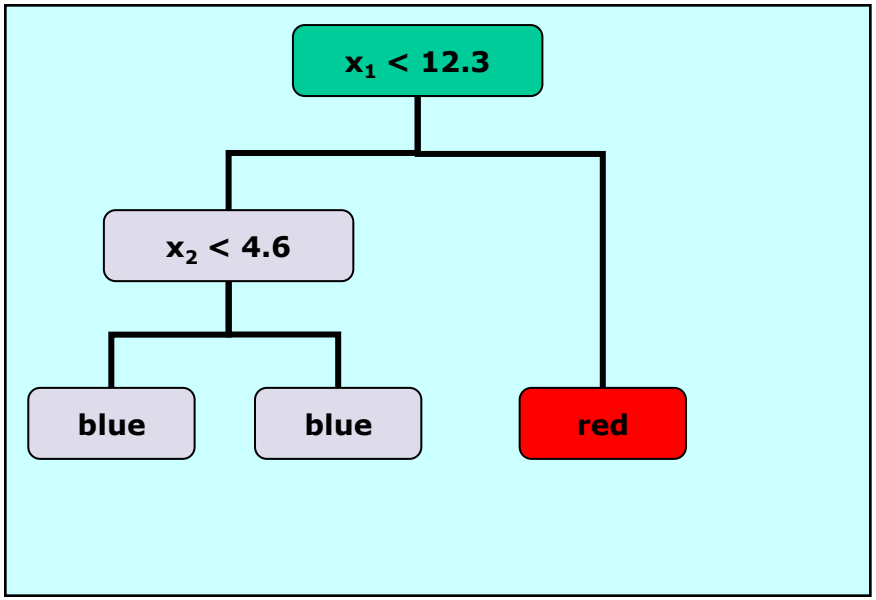


Decision for split in case of equal number of samples?

➔ *e.g. random*

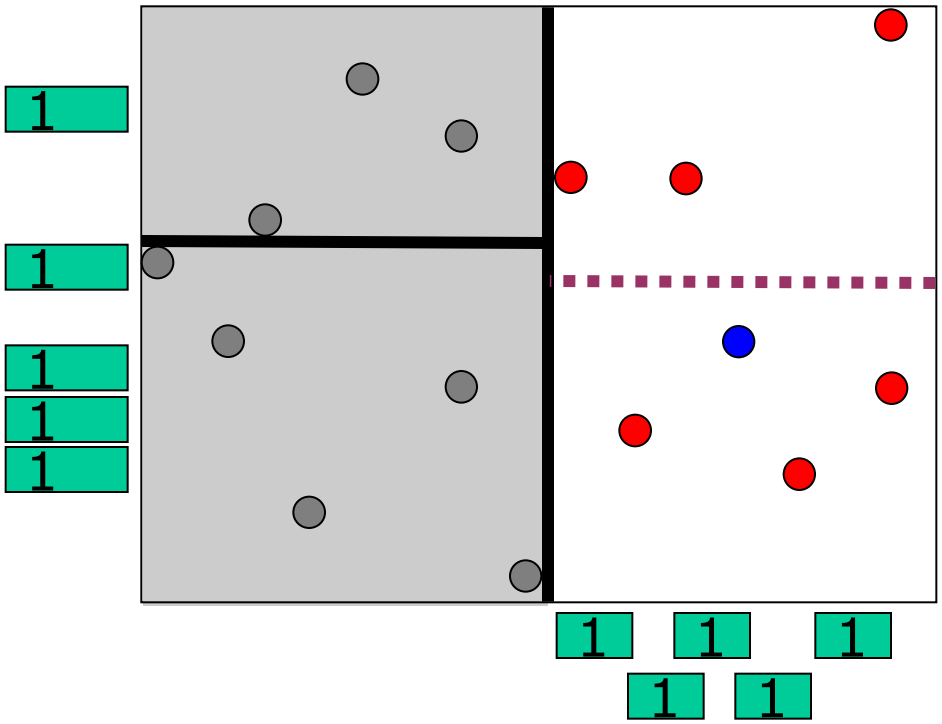
Decision Trees: error rate

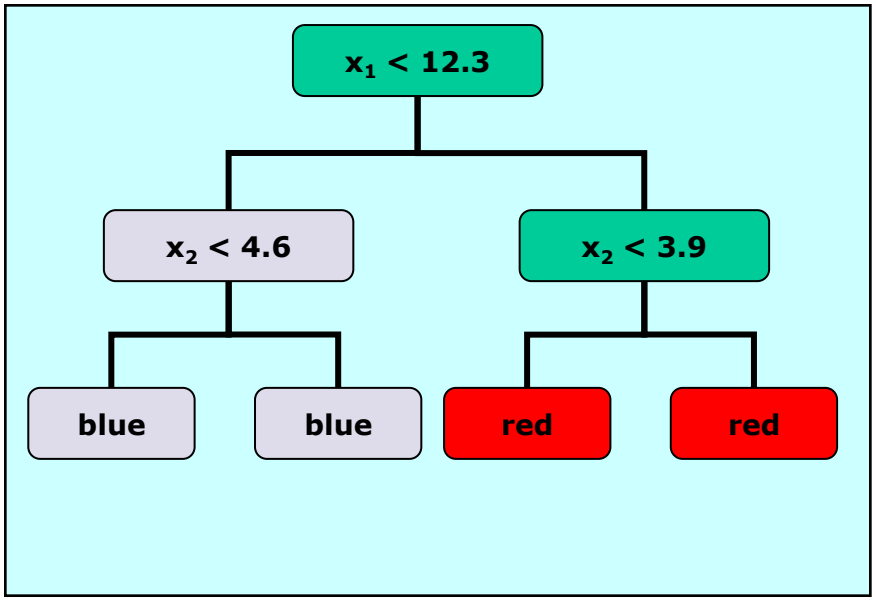




Which split?

Decision Trees: error rate



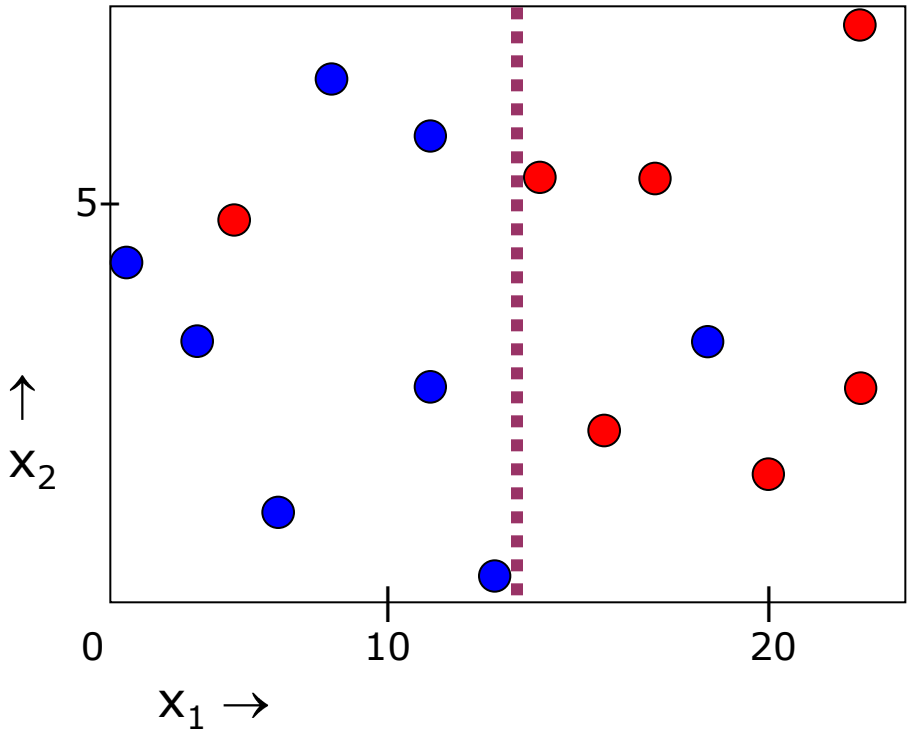


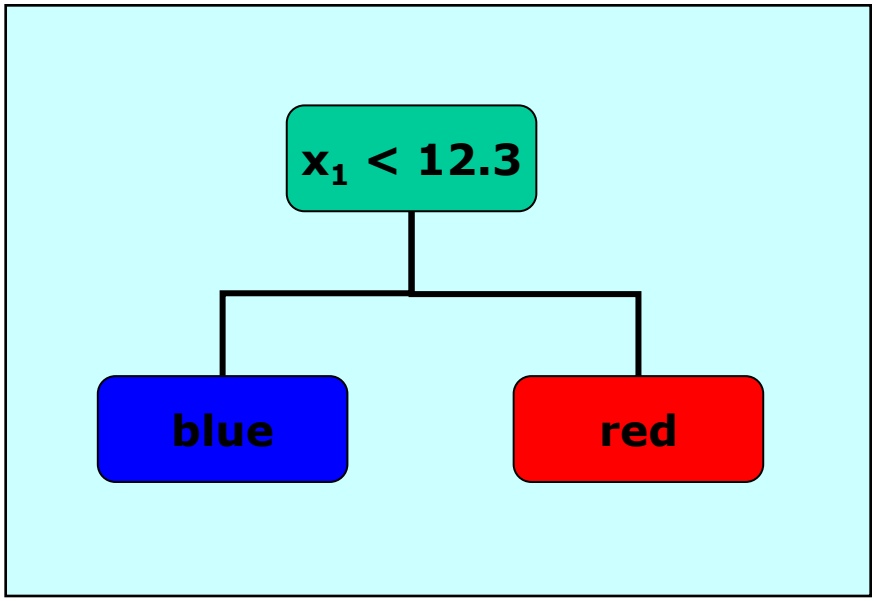
N.b.: each leaf-node's prediction decided by majority

Not influenced by prediction of the other leaf

Decision Trees: error rate

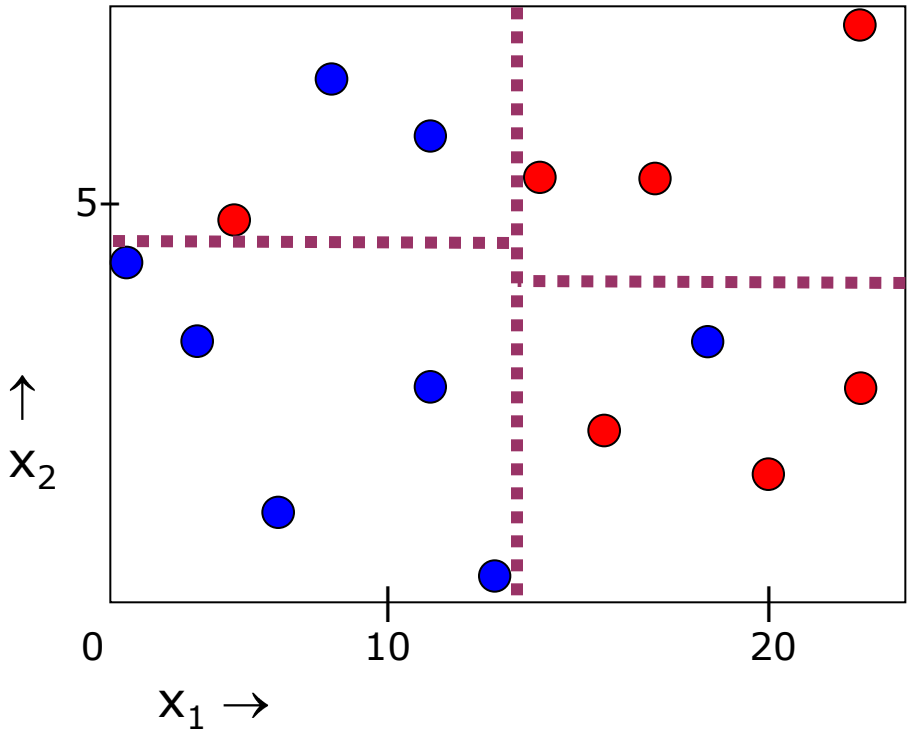
- Tree training, level 1

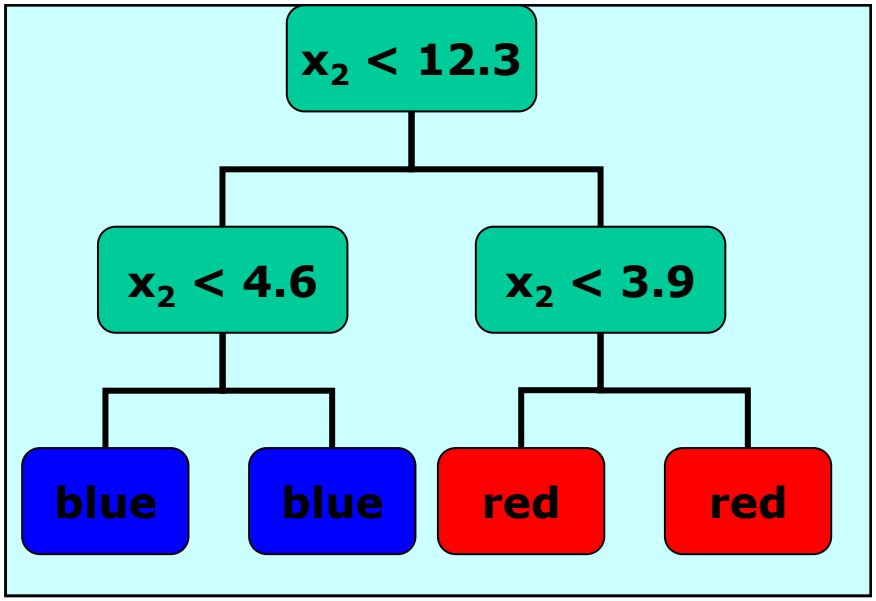




Decision Trees: error rate

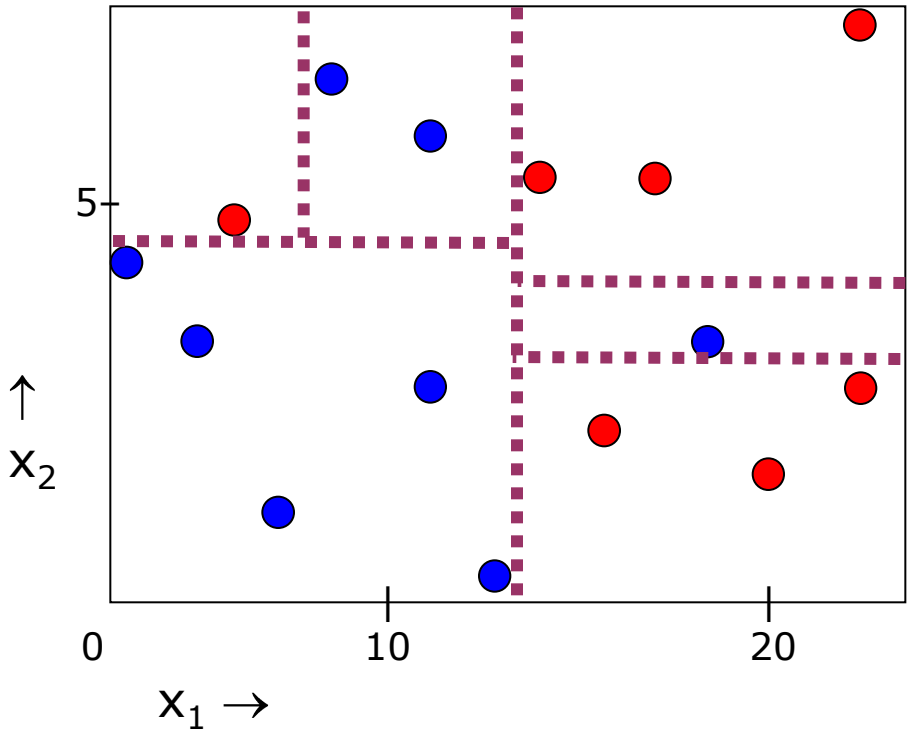
- Tree training, level 2

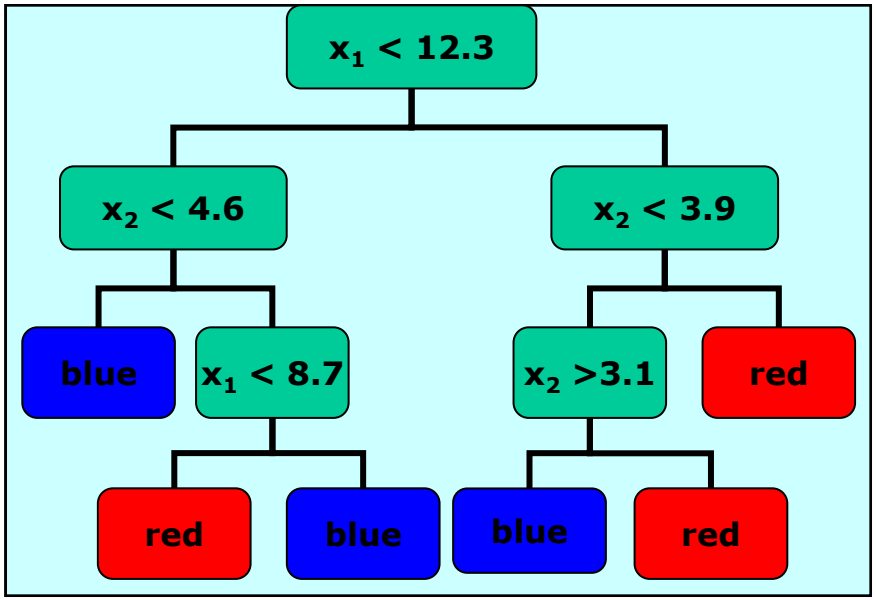




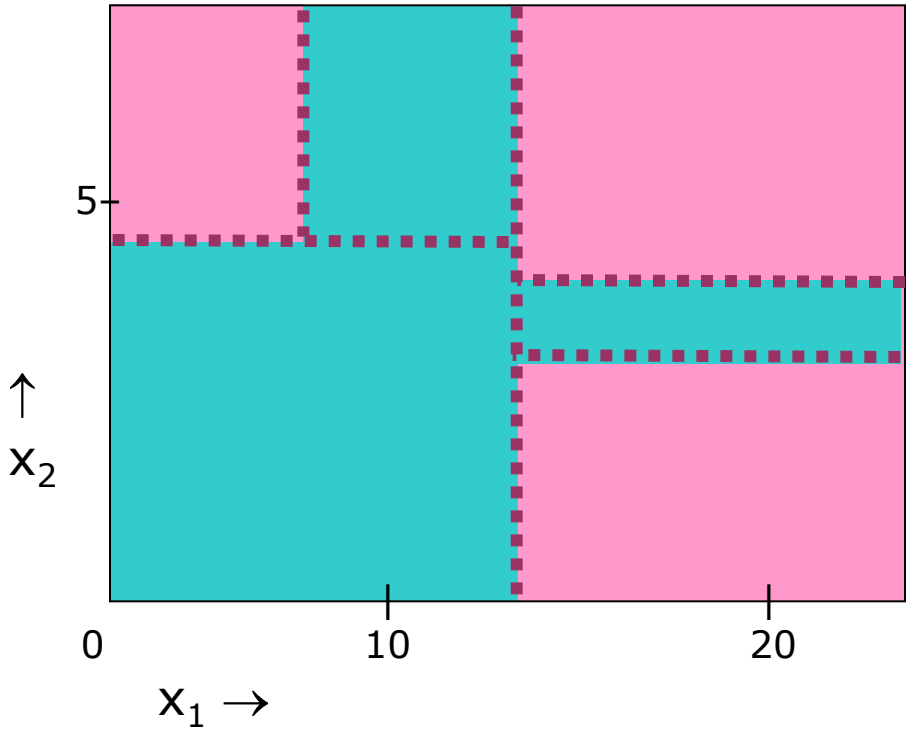
Decision Trees: error rate

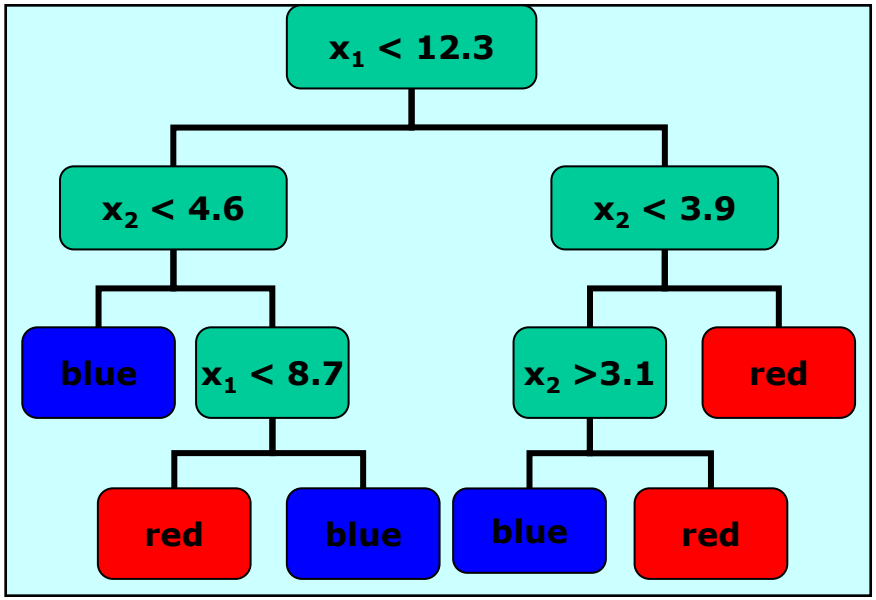
- Tree training, level 3: completely built tree





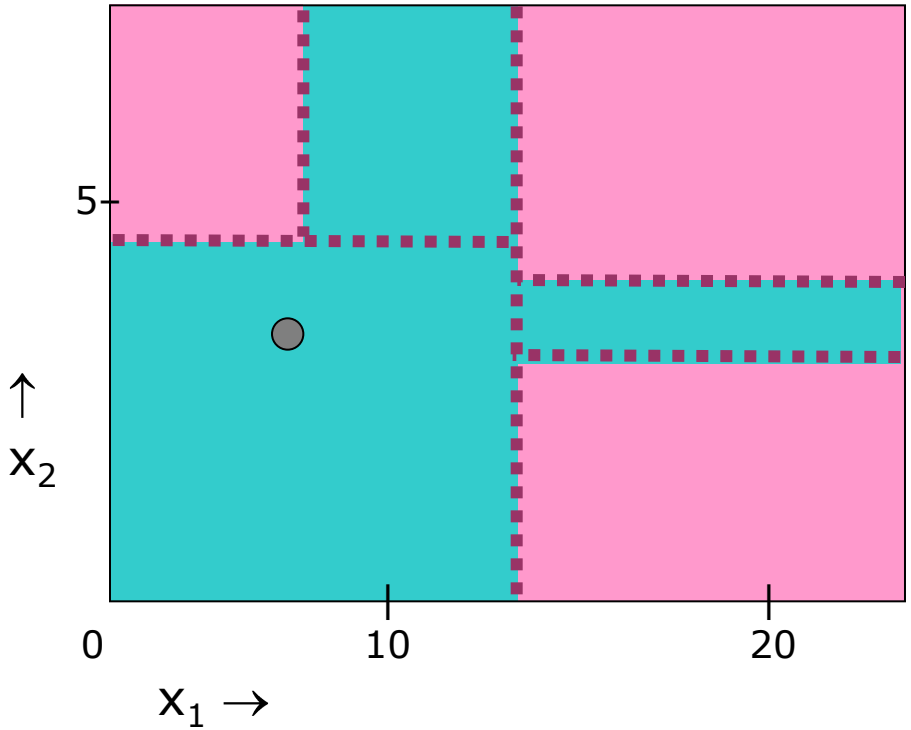
Decision Trees: Classification

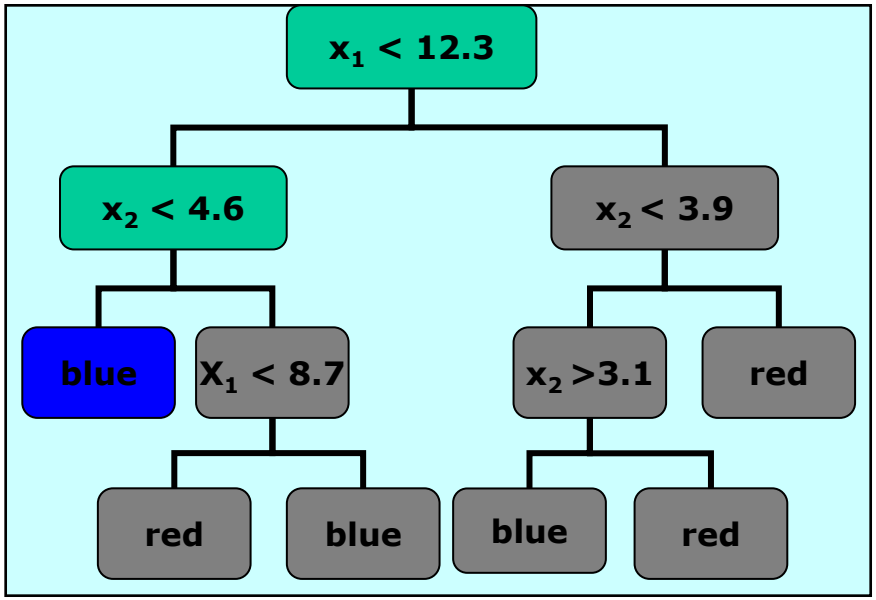




How to classify unlabelled instances?

Decision Trees: Classification

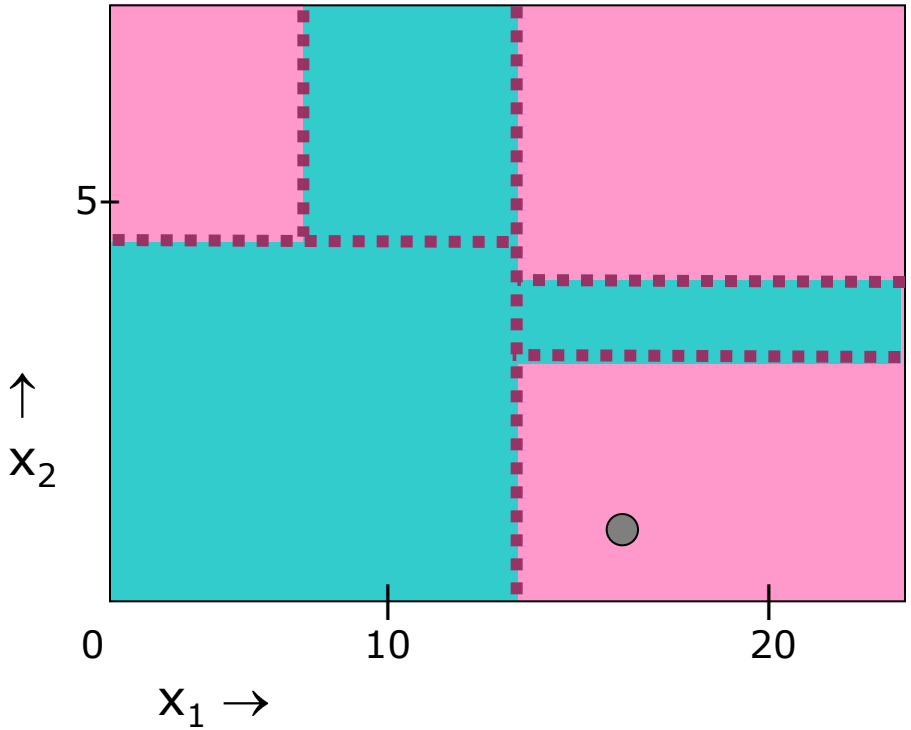


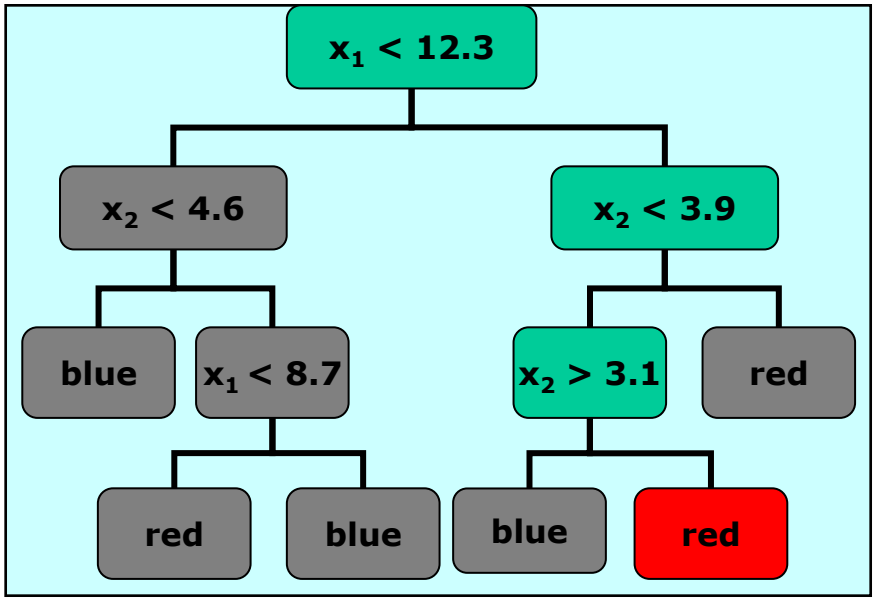


➔ Decend the tree until leaf-node

➔ Use majority of class in that leaf node

Decision Trees: Classification



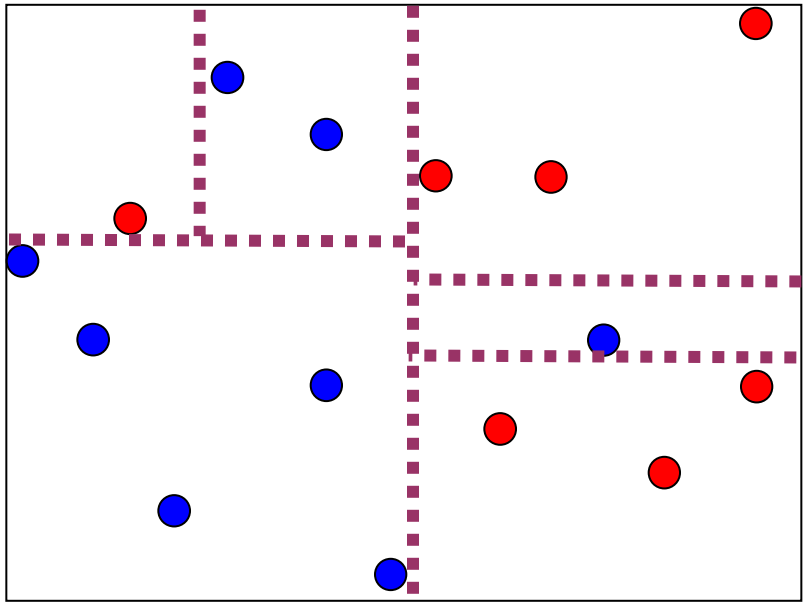


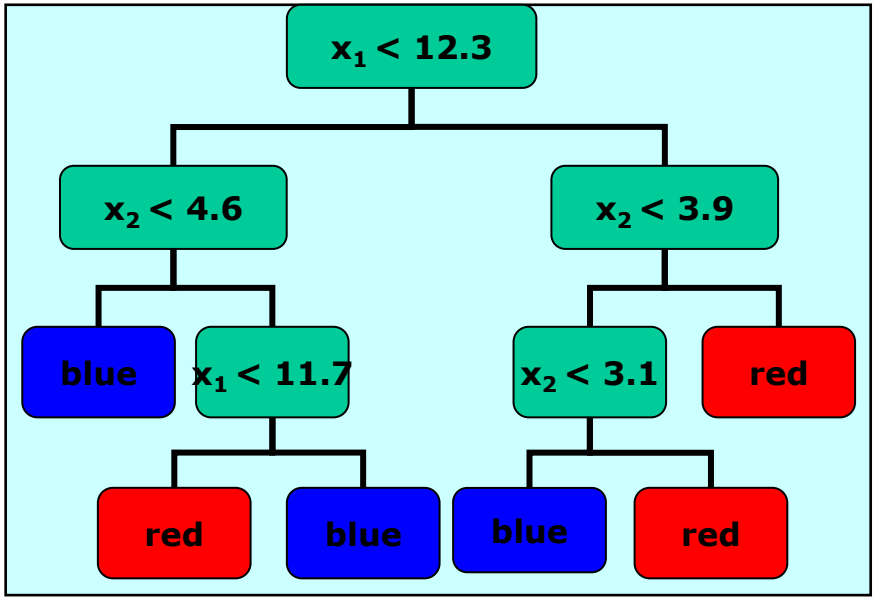
➔ Decend the tree until leaf-node

➔ Use majority of class in that leaf node

Decision Trees: training

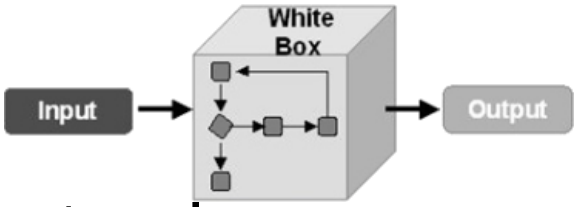
- *How many possible splits?*
  - For numeric data: (number of unique values) – 1
  - In each attribute!
  - Could reduce number of splits, e.g. by binning





Decision Trees – first properties

- Rather old model, well known



- Simple algorithm ➔ easy to understand
  - White box, rather than black-box
  - Used in many non-IT domains
  - Can be used to illustrate expert knowledge

- *Various split criteria; categorical data*
- *Problems with overfitting – (pre)pruning helps*
- *Problems with stability ➔ can be exploited!*

Outline

- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

Evaluation

- When we use a machine learning model, we want to know how good it is (effectiveness)
  - To know how confident we can be in the predictions
  - To know which algorithm (e.g. k-NN vs. Perceptron) or hyper-parameters (k=1, k=3, ...) to use
  - ....
- ➔ *Model validation*
- Need to measure performance of an algorithm
  - Test on (labelled) data
  - Several different measures
- Orthogonal topic: efficiency, i.e. required runtime
  - *More on that later*

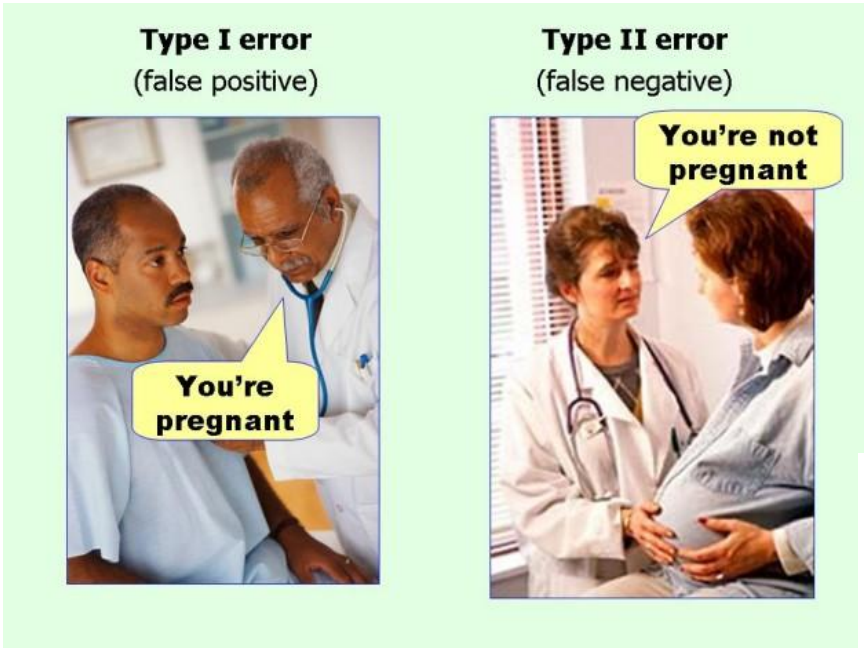
Evaluation (effectiveness)

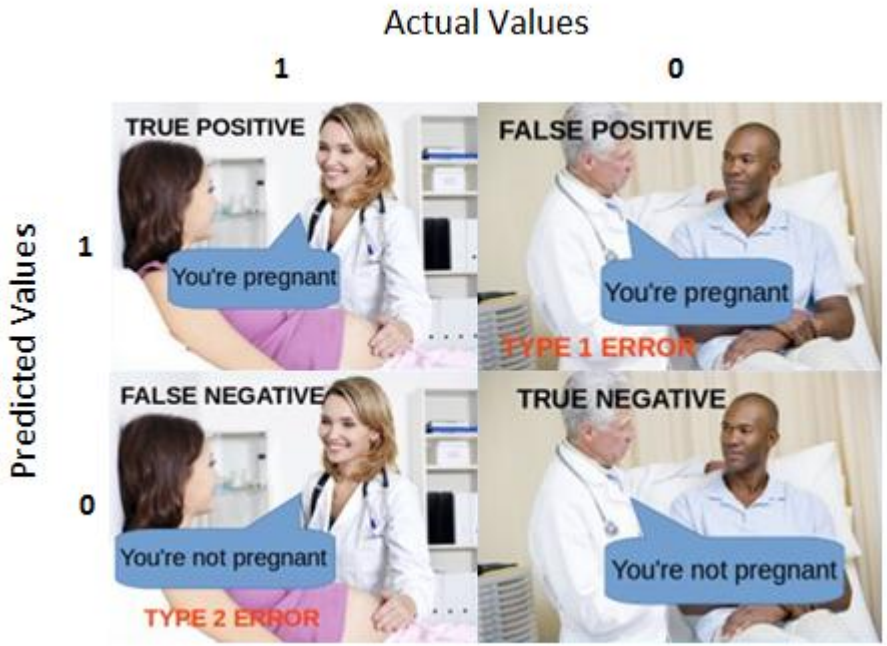
- Binary classification (classes true/false)
- Table of confusion *(contingency table)*

		Actual value (groundtruth)		
		true	false	
Prediction / Test outcome	true	True positive (TP)	False positive (FP, Type I error)	***Precision*** $\frac{TP}{TP + FP}$
	false	False negative (FN, Type II error)	True negative (TN)	
		Recall Sensitivity $\frac{TP}{TP + FN}$	Specificity (True Negative Rate)	***Accuracy*** $\frac{TP + TN}{\#\ samples}$

- *Examples of Type I & II errors?*
  - *Which is worse?*

Evaluation (effectiveness)





Evaluation measures

	true	false
true	True positive (TP)	False positive (FP, Type I error)
false	False negative (FN, Type II error)	True negative (TN)

- Error rate?
- Accuracy: # correctly predicted samples

$$\frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\#\ samples}$$

- Precision $\frac{TP}{TP + FP}$ ➔ *How to optimise?*
- Recall $\frac{TP}{TP + FN}$ ➔ *How to optimise?*
- *More measures in later lectures ...*

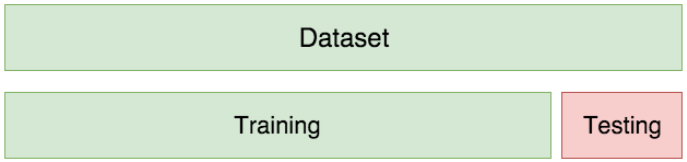
Evaluation: data

- Which data to do evaluation on?
- *The samples used for training?*
  - *Why not?*
  - Would not tell how good the model works for **unknown** data
    - Which is however why we train a model in first place ...
  - *If we test on training data – we are biased*
    - Perceptron on linear separable data: training data will be 100% correctly “predicted”
    - Fully grown decision trees – 100% correct on training data (in normal cases)
    - K-NN, Naïve Bayes: not necessarily 100% correct on training data. Still biased!
    - Similar for other algorithms, though not that extreme, e.g. SVMs, ...
  - *Want to find out:* how will model perform on **unseen** data!

Training & Test set

- Problem: really “unseen” data doesn’t have labels
  - “Simulate” “unseen” data
  - ➔ Holdout method: keep part of your labelled data for testing

- Split data into training and test sets
  - E.g. ~80% training, 20% test
    - 66% - 33%
  - Sorted (linear), random, …

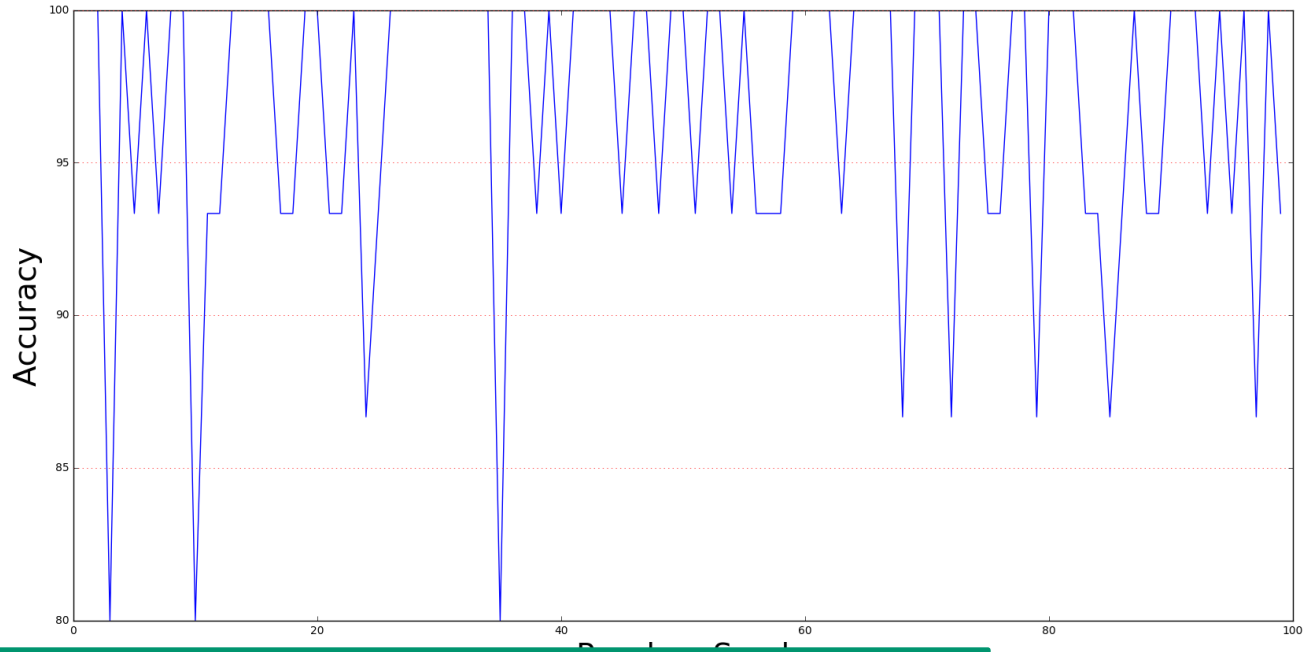


- Performance on test set is an estimate for generalisation ability of our model

- *Results can vary a lot according to how the split is done*

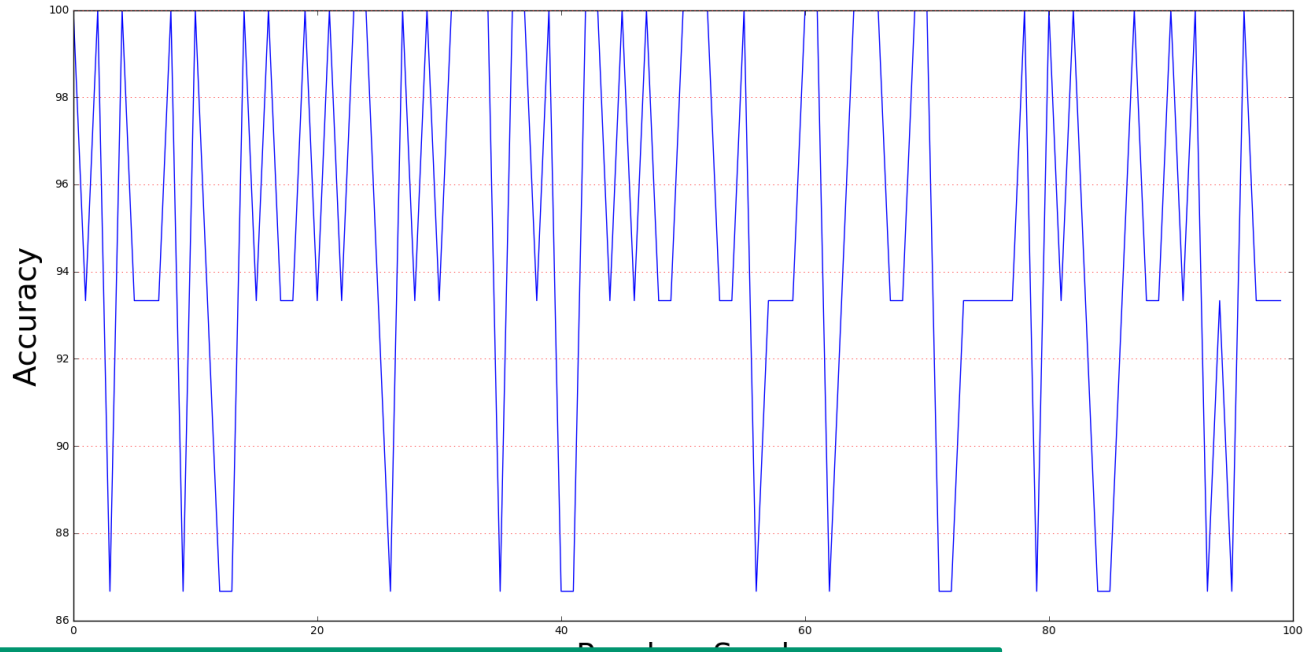
Holdout method

- Random influence in training/test split
  - Example: **Iris Data, *k-NN (k=15)***, random seed 1..100
    - Average Accuracy: 96.67 (+/- 9.8883) [range: 80-100]



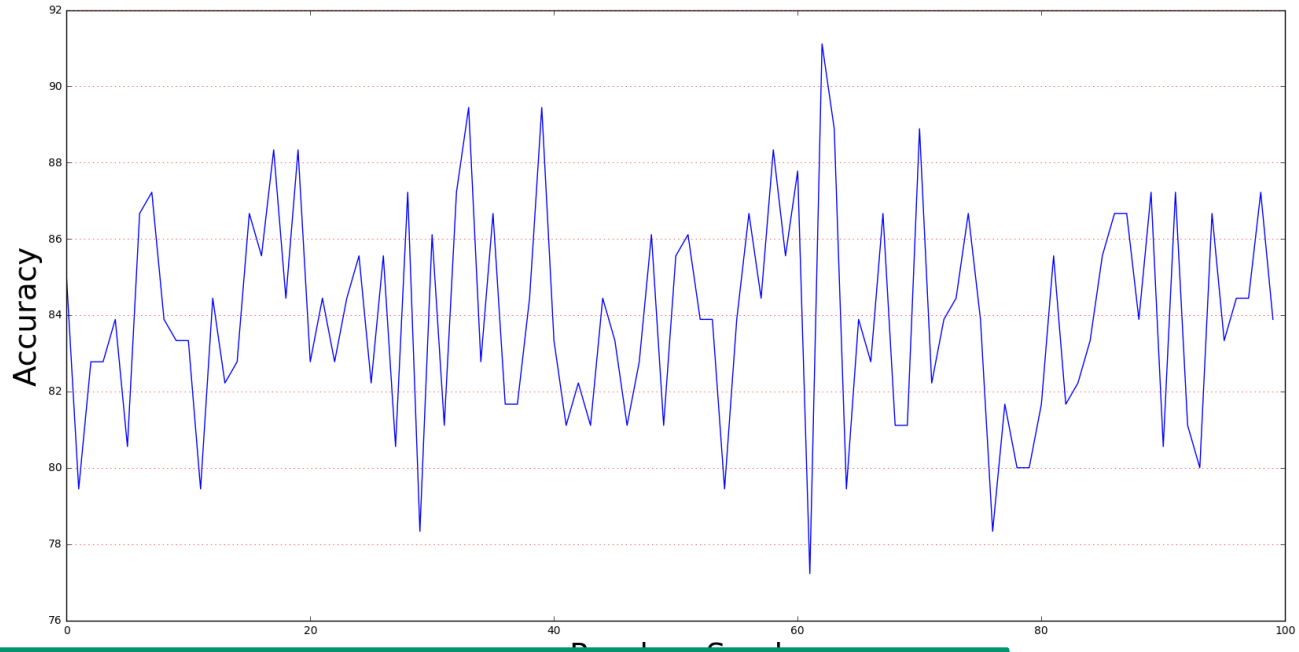
Holdout method

- Random influence in training/test split
  - Example: **Iris Data, *Naive Bayes***, random seed 1..100
    - Average Accuracy: 95.00 (+/- 9.6839) [range: 86.67-100]



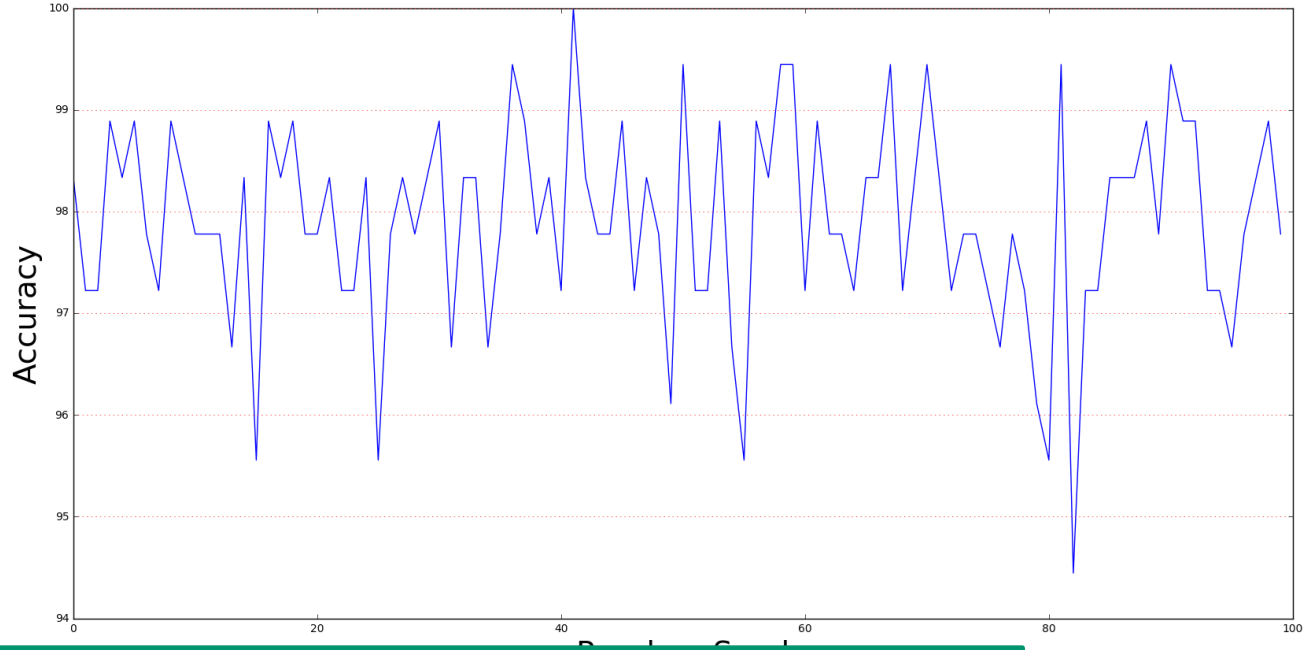
Holdout method

- Random influence in training/test split
  - Example: **Digit Data, *k-NN (k=15)***, random seed 1..100
    - Average Accuracy: 97.90 (+/- 2.0097) [range: 94.44-100]



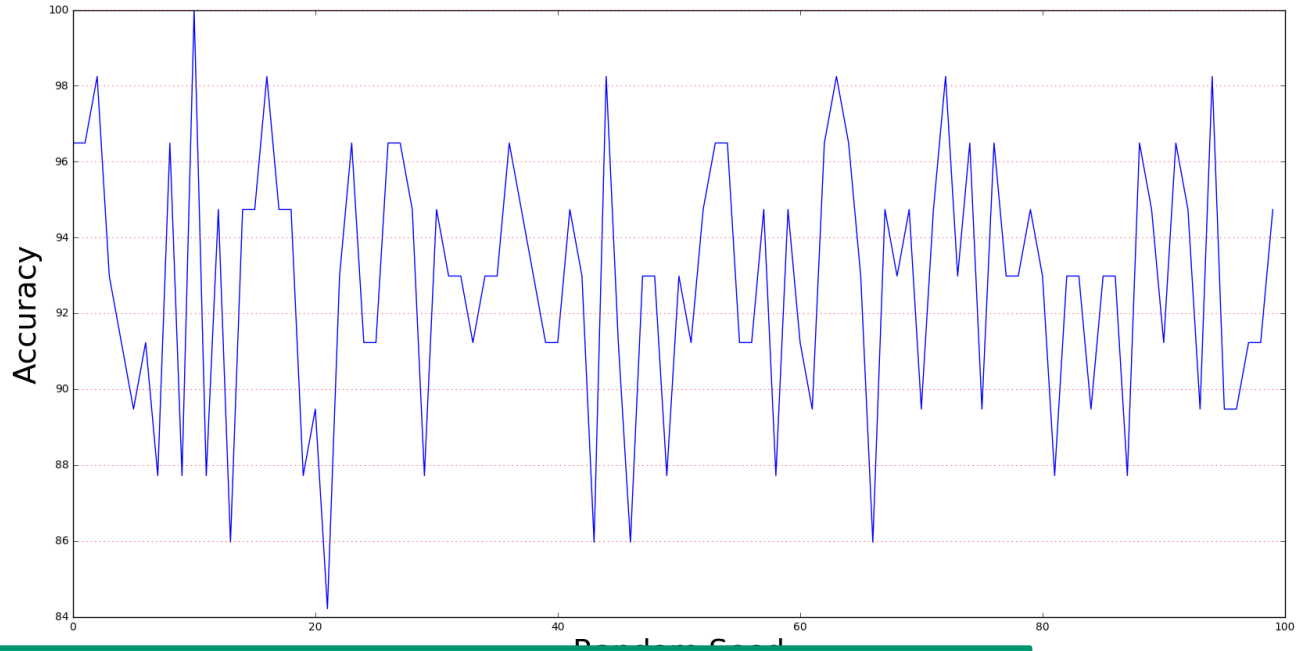
Holdout method

- Random influence in training/test split
  - Example: **Digit Data, *Naive Bayes***, random seed 1..100
    - Average Accuracy: 83.93 (+/- 5.6167) [range: 77.22-91.11]



Holdout method

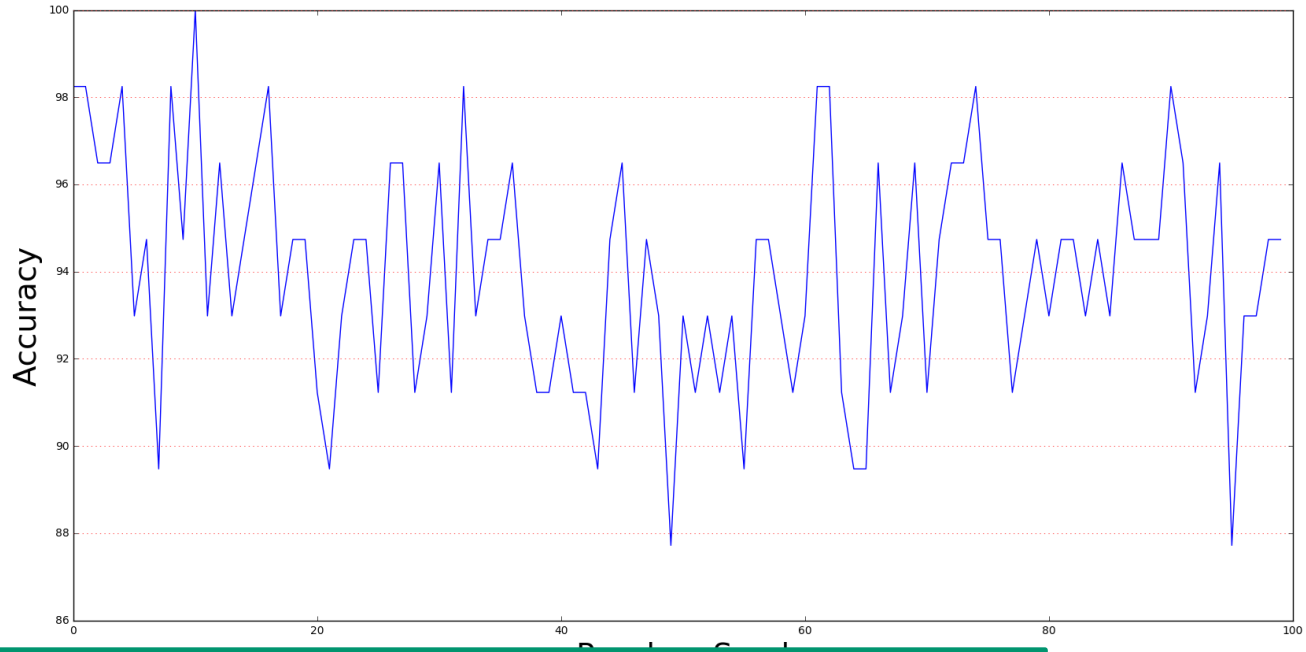
- Random influence in training/test split
  - Example: **Cancer Data, *k-NN (k=15)***, random seed 1..100
    - Average Accuracy: 92.81 (+/- 6.7401) [range: 84.21-100]



Holdout method

- Random influence in training/test split
  - Example: **Cancer Data, *Naive Bayes***, random seed 1..100
    - Average Accuracy: 93.96 (+/- 5.2355) [range: 87.719-100]

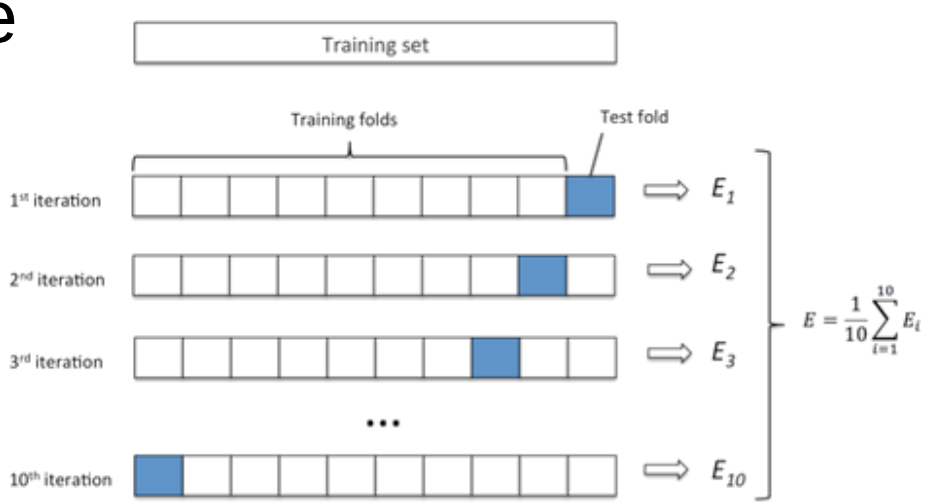
- Solution?



k-fold Cross validation

- Split data into e.g. 10 parts of equal sizes
  - This is called 10-fold cross validation
- Repeat 10 times:
  - use 9 parts for training (training set)
  - calculate performance on remaining part (test set)

- Estimate of performance is average (mean) of the validation set performances



k-fold Cross validation

- Estimate of performance is average (mean)
- In addition to mean, compute standard deviation
  - Indication on how stable the results are in the folds

    ➔ lower standard deviation is better ...

  - Standard deviation to be considered when comparing cross-validation performances from different classifiers

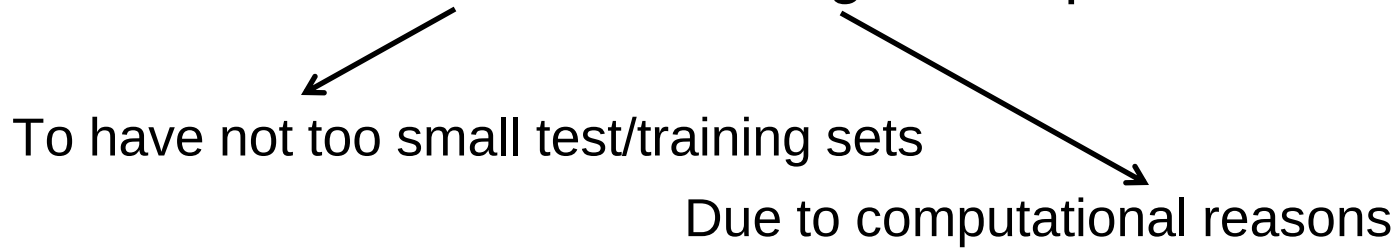
Classifier / Fold	1	2
1	91,80	86,7
2	82,30	87
3	84,40	87,1
4	93,00	85,7
5	81,60	86,8
6	87,40	86,4
7	82,40	87,2
8	92,10	86,5
9	91,90	86,5
10	87,40	86,5
Mean	87,4	86,6
Stdv	4,6	0,4

k-fold Cross validation

- Which classifier is better:
  - Average 87,4%, standard deviation 4,8%
    - *(87,4% 4,8)*
  - Average 86,6%, standard deviation 0,4%
    - *(86,6% 0,4)*
    - ➔ *More on that later: significance testing*

k-fold Cross validation

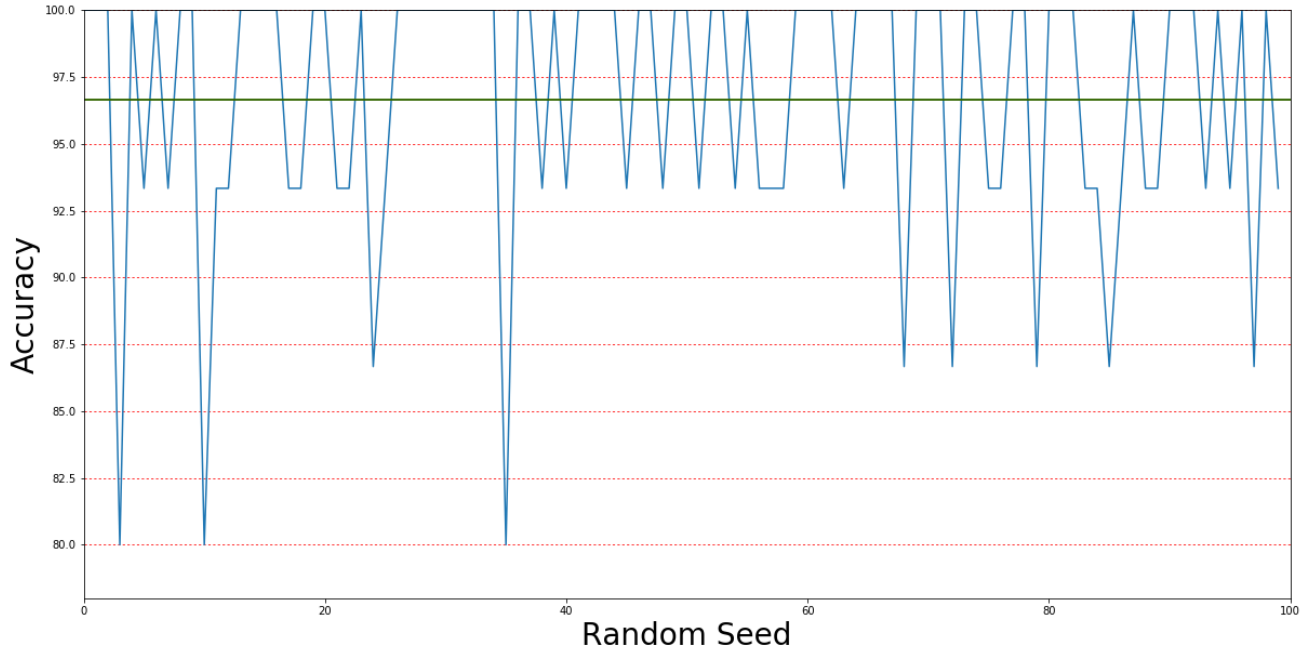
- Results obtained via cross-validation are generally much more reliable
  - Parameter of 10 often used
  - Fewer folds on *smaller and larger* sample size



- Number of folds increases runtime!
  - More-or-less linear with ***n***
  - Might not be that critical – why?
    - Can be parallelised

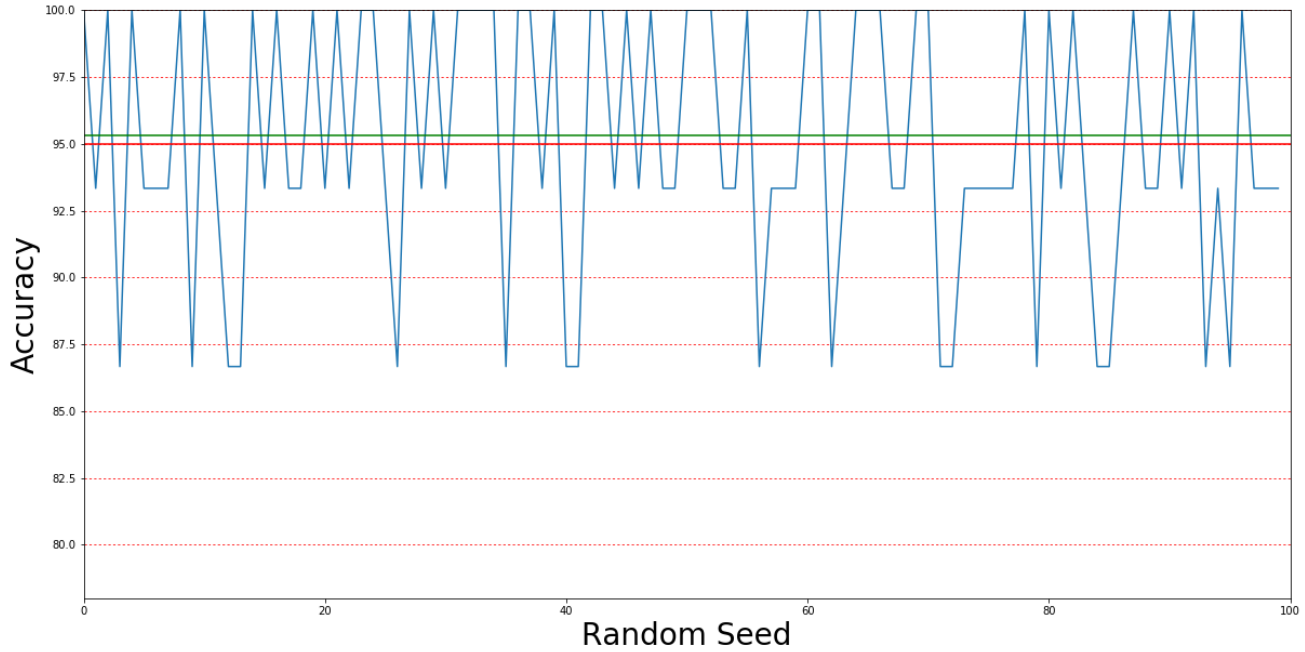
Cross validation

- Cross-validation
  - Smooth effects of random influence in training/test split
  - Example: **Iris Data, *k-NN (k=15)***, random seed 1..100
    - Average Accuracy: 96.67 (+/- 9.8883) [range: 80-100]
    - 10-fold CV: 96.67 (+/- 8.94)



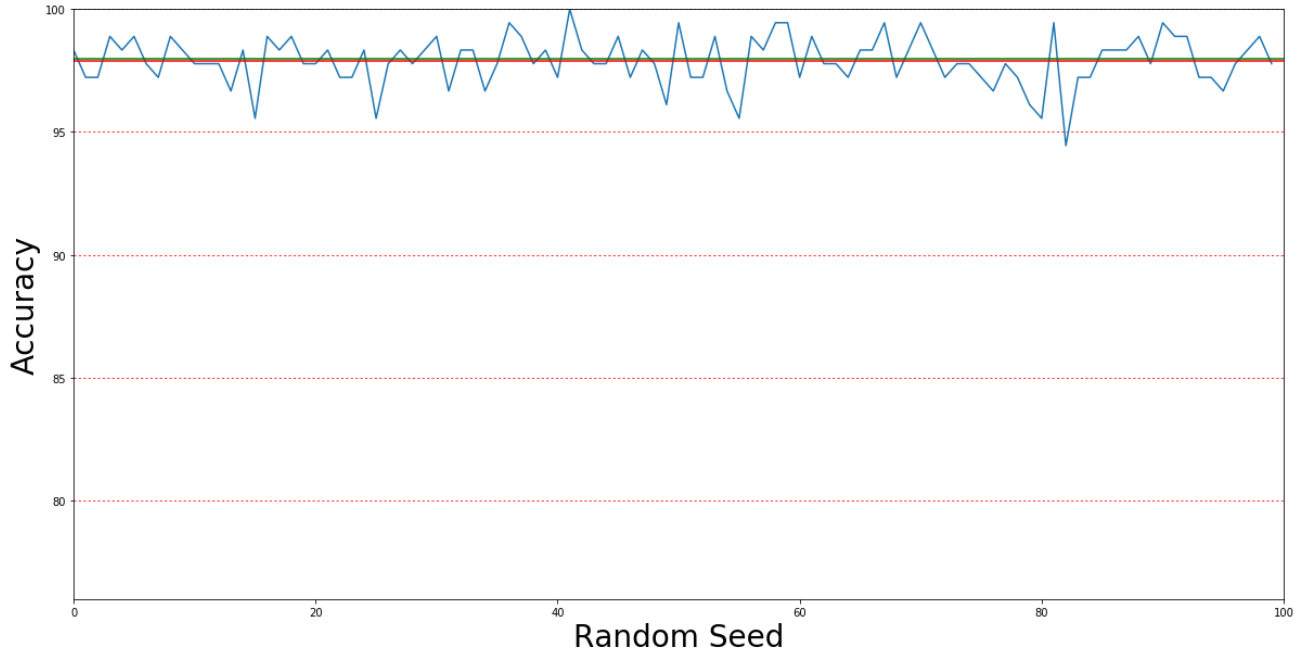
Cross validation

- Cross-validation
  - Smooth effects of random influence in training/test split
  - Example: **Iris Data, *Naive Bayes***, random seed 1..100
    - Average Accuracy: 95.00 (+/- 9.6839) [range: 86.67-100]
    - 10-fold CV: 95.33 (+/- 12.00)



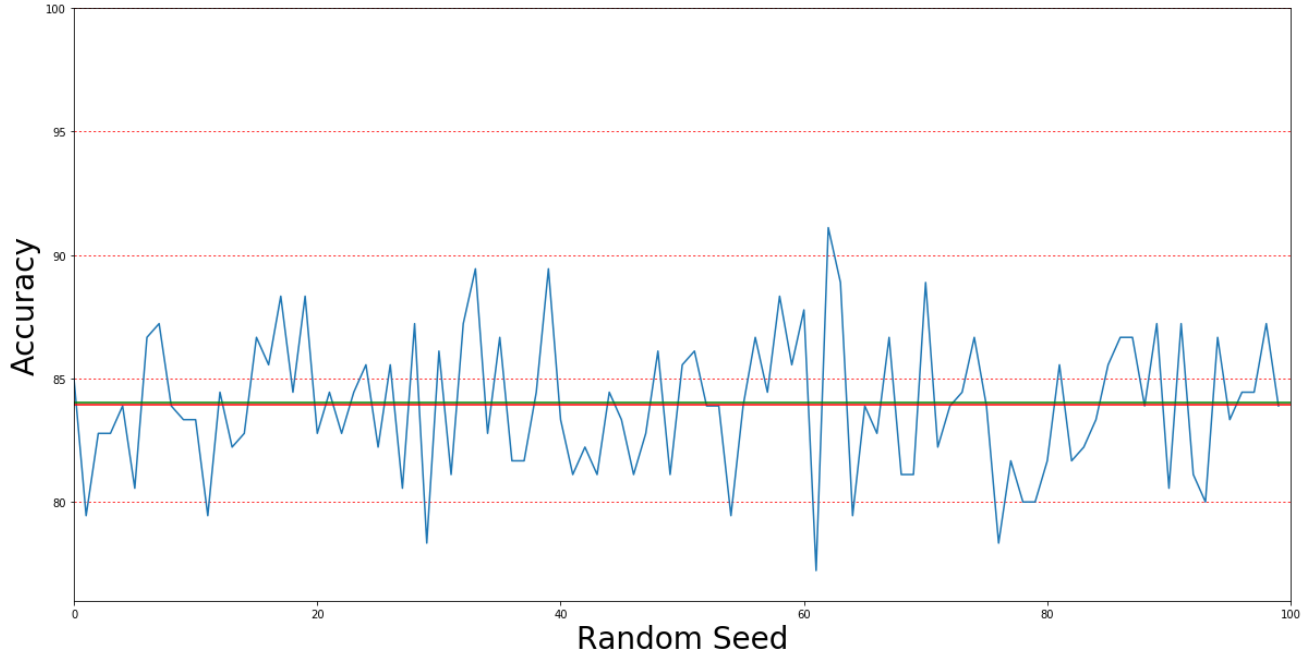
Cross validation

- Cross-validation
  - Smooth effects of random influence in training/test split
  - Example: **Digit Data, *k-NN (k=15)***, random seed 1..100
    - Average Accuracy: 97.90 (+/- 2.0097) [range: 94.44-100]
    - 10-fold CV: 97.99 (+/- 1.96)



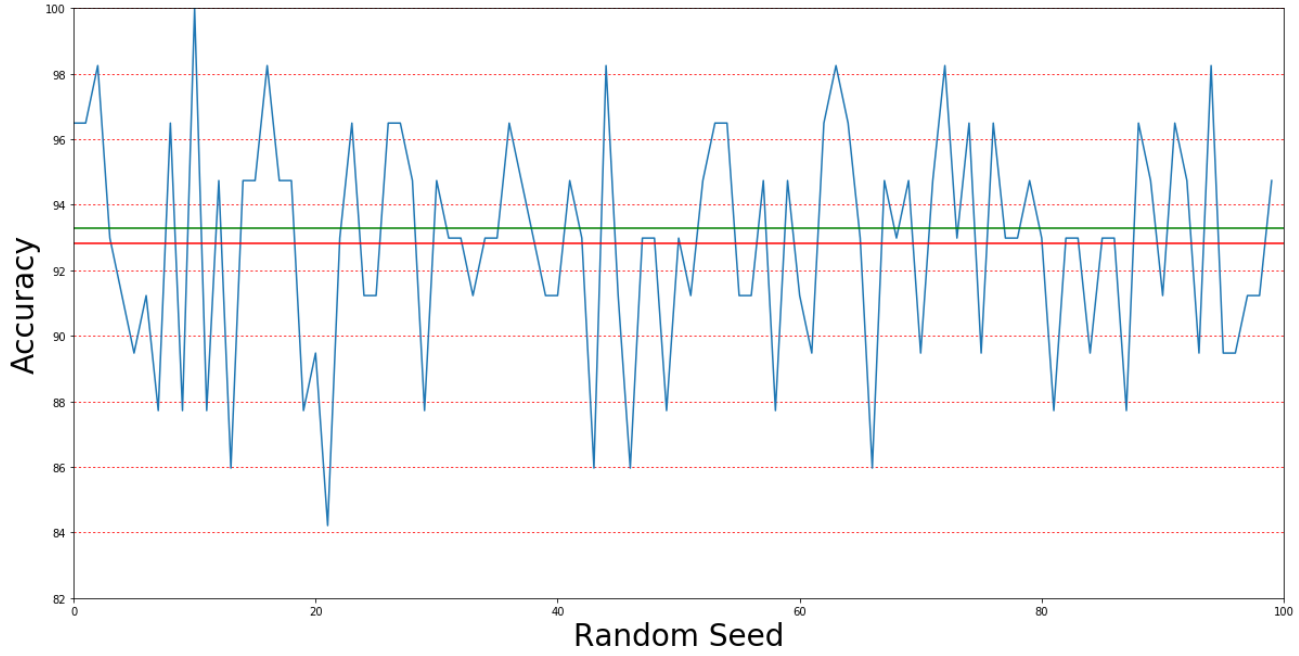
Cross validation

- Cross-validation
  - Smooth effects of random influence in training/test split
  - Example: **Digit Data, *Naive Bayes***, random seed 1..100
    - Average Accuracy: 83.93 (+/- 5.6167) [range: 77.22-91.11]
    - 10-fold CV: 84.03 (+/- 8.02)



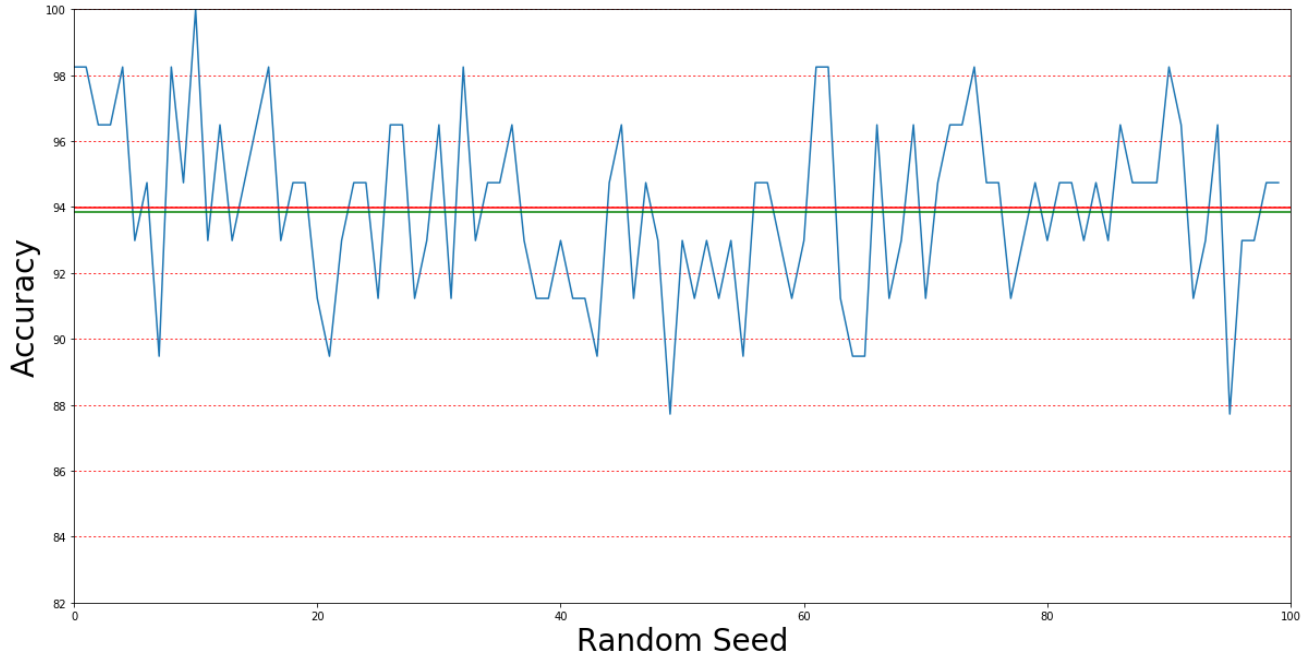
Cross validation

- Cross-validation
  - Smooth effects of random influence in training/test split
  - Example: **Cancer Data, *k-NN (k=15)***, random seed 1..100
    - Average Accuracy: 92.81 (+/- 6.7401) [range: 84.21-100]
    - 10-fold CV: 93.29 (+/- 8.1)



Cross validation

- Cross-validation
  - Smooth effects of random influence in training/test split
  - Example: **Cancer Data, *Naive Bayes***, random seed 1..100
    - Average Accuracy: 93.96 (+/- 5.2355) [range: 87.719-100]
    - 10-fold CV: 93.83 (+/- 4.33)



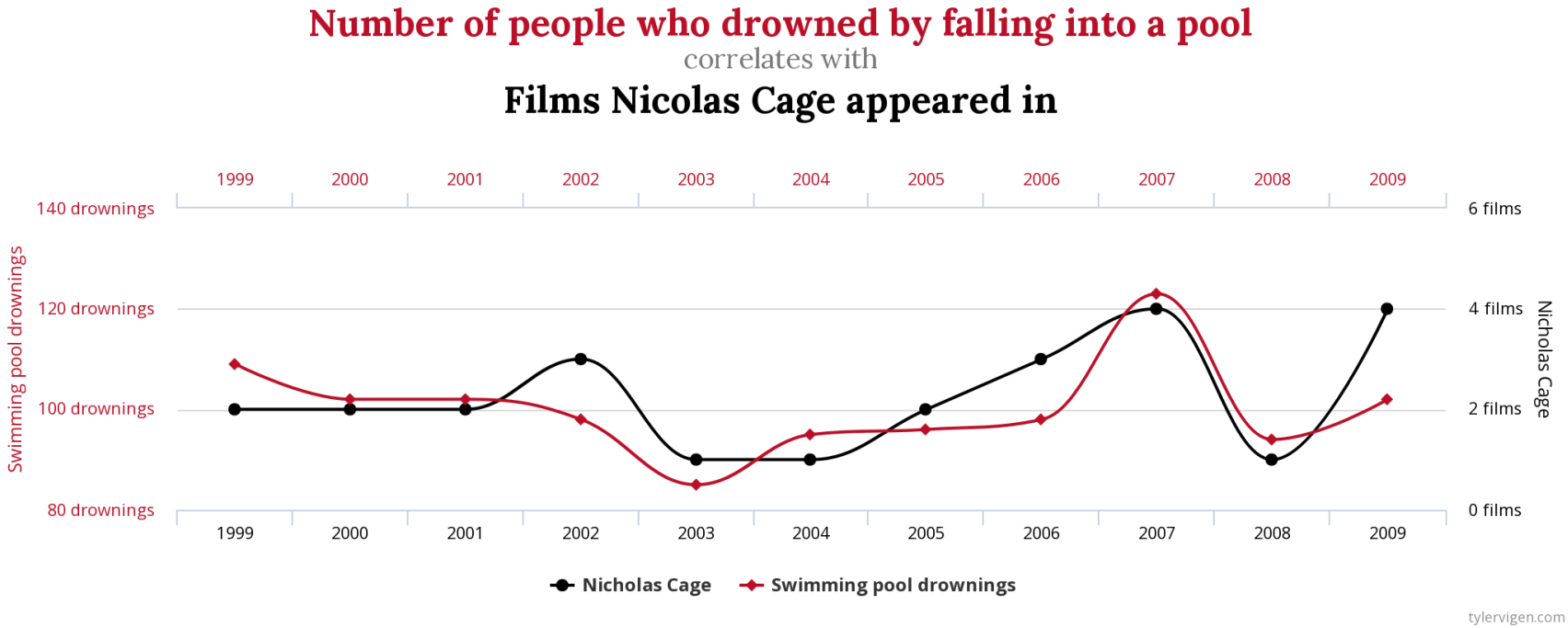
Evaluation: outlook

- Micro vs. macro averaging
  - Confusion matrix
  - Cost functions
- Other evaluation measures
- ROC curves
- Bootstrapping
- Significance testing

- Evaluation measures for regression

Meaningful?

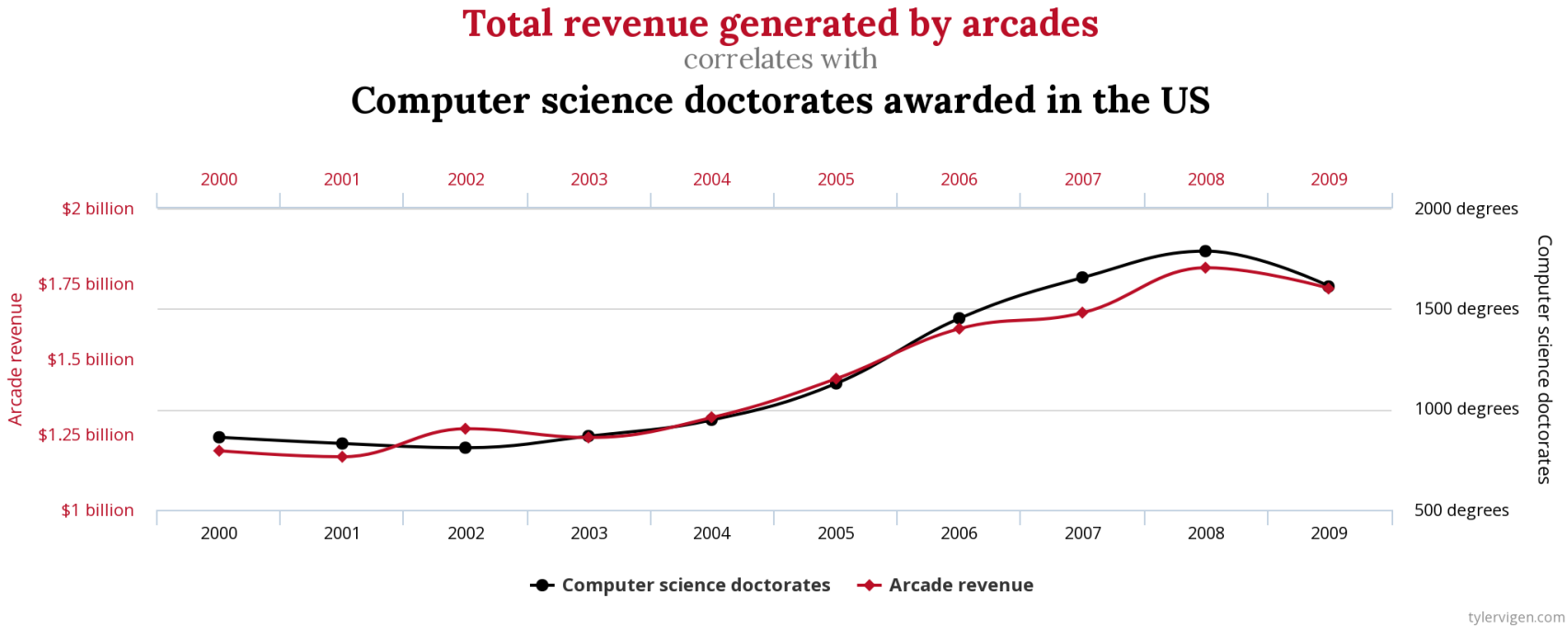
- Not every model is actually use/meaningful ☺
  - Even if some evaluation measure says so...



http://www.tylervigen.com/spurious-correlations

Meaningful?

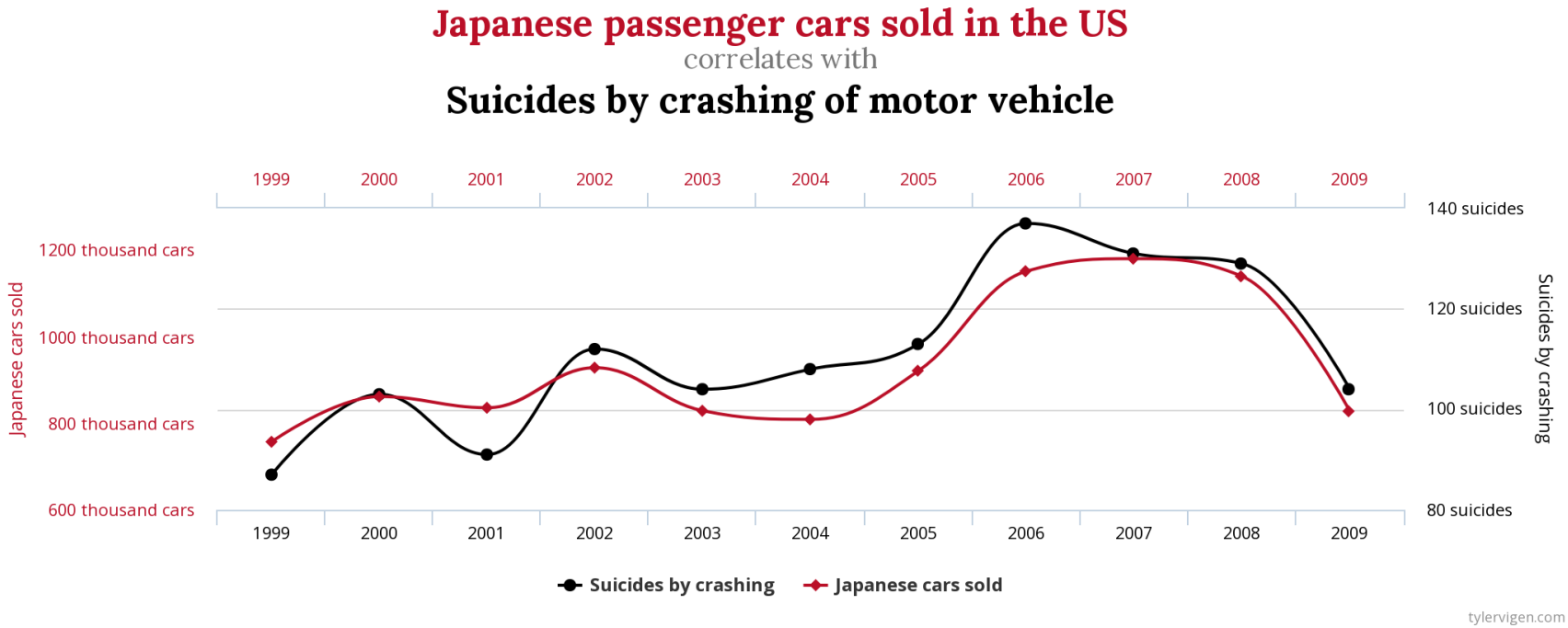
- Not every model is actually use/meaningful ☺
  - Even if some evaluation measure says so...



http://www.tylervigen.com/spurious-correlations

Meaningful?

- Not every model is actually use/meaningful ☺
  - Even if some evaluation measure says so...



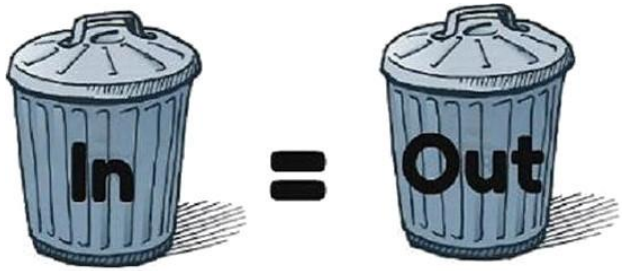
http://www.tylervigen.com/spurious-correlations

Outline

- Short Recap & Data Types
- Perceptron
- k-NN
- Decision Trees
- Performance evaluation (intro)
- Data preparation (intro)

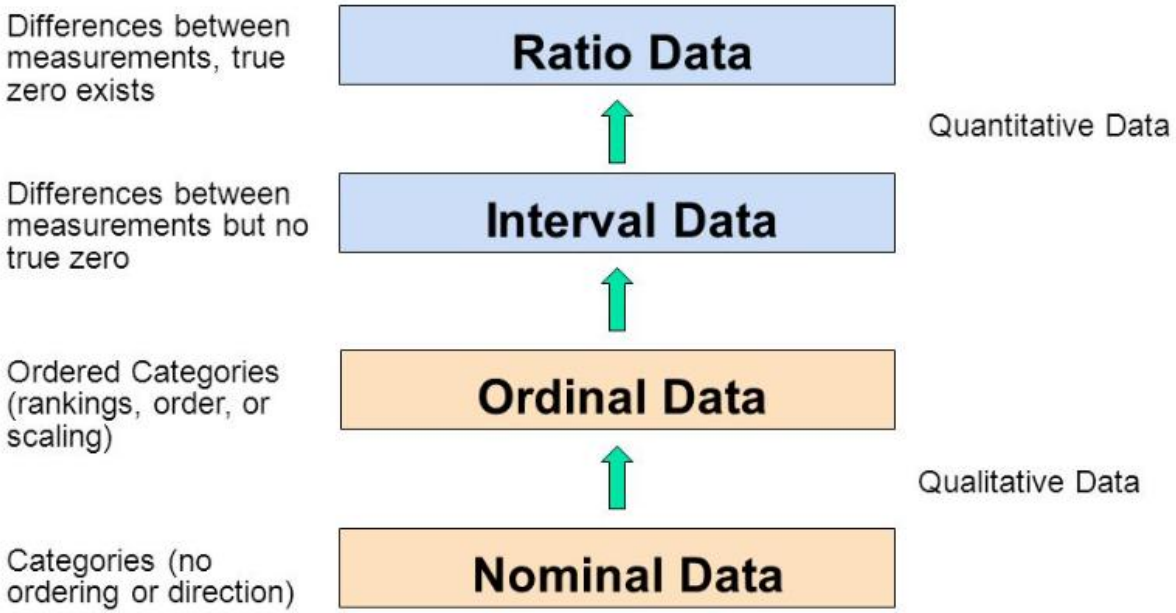
Data Preparation

- Vital step for machine learning (supervised and unsupervised)
- ML algorithm will ***always*** give you a model
  - Quality of that model depends highly on the quality of the input data
- "Garbage in" ➔ "Garbage out"



- One major goal of data preparation:
  - Eliminate "wrong influence" of variables

Data Types



Can compute	Nominal	Ordinal	Interval	Ratio
Frequency distribution	Yes	Yes	Yes	Yes
Median and percentiles	No	Yes	Yes	Yes
Add or subtract	No	No	Yes	Yes
Mean, standard deviation, …	No	No	Yes	Yes
Ratios, coefficient of variation	No	No	No	Yes

Data Preparation

- Example data set from earlier (lung cancer)

gender	age	height	smoker	eye colour
male	19	170	yes	green
female	44	162	yes	grey
male	49	185	yes	blue
male	12	178	no	brown
female	37	165	no	brown
female	?	157	no	?
male	44	190	no	blue
female	27	178	yes	brown
female	51	162	yes	green
female	81	168	?	grey
male	22	184	yes	brown
male	29	176	no	blue

- Potential issues ?
  - Quantitative (continuous) data with different scales
  - Data type not fitting
    - Categorical (nominal, ordinal, ..) vs numerical (interval, ratio)
  - Missing values

Scaling / Normalisation / Standardisation

- Different variables may exhibit significantly different value ranges
  - E.g. a length variable measured in cm, inch, or meters
  - Different types of measurements: length, speed, temperature, …
  - Different types of measuring devices capturing different value ranges
  - …
  - *Why is this a (potential) problem?*

Scaling / Normalisation / Standardisation

- Some ML algorithms rely on *measuring the (numeric) distance* between samples

- ➔ There should be no impact by the value range

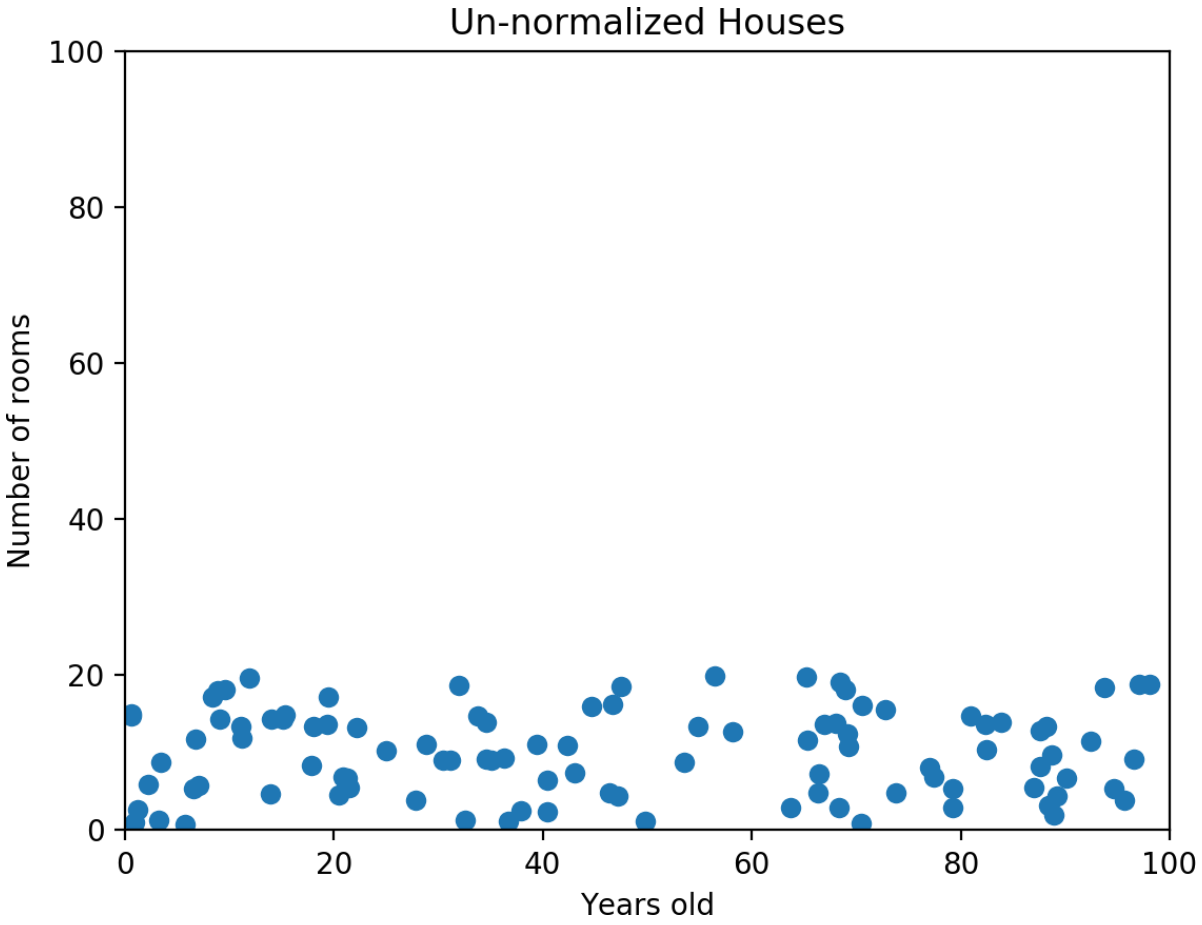
  – higher values in one attribute / variable would have unproportional effect on measure distance
  ➔ would dominate distance metric
  ➔ might thus dominate learning

Scaling / Normalisation / Standardisation

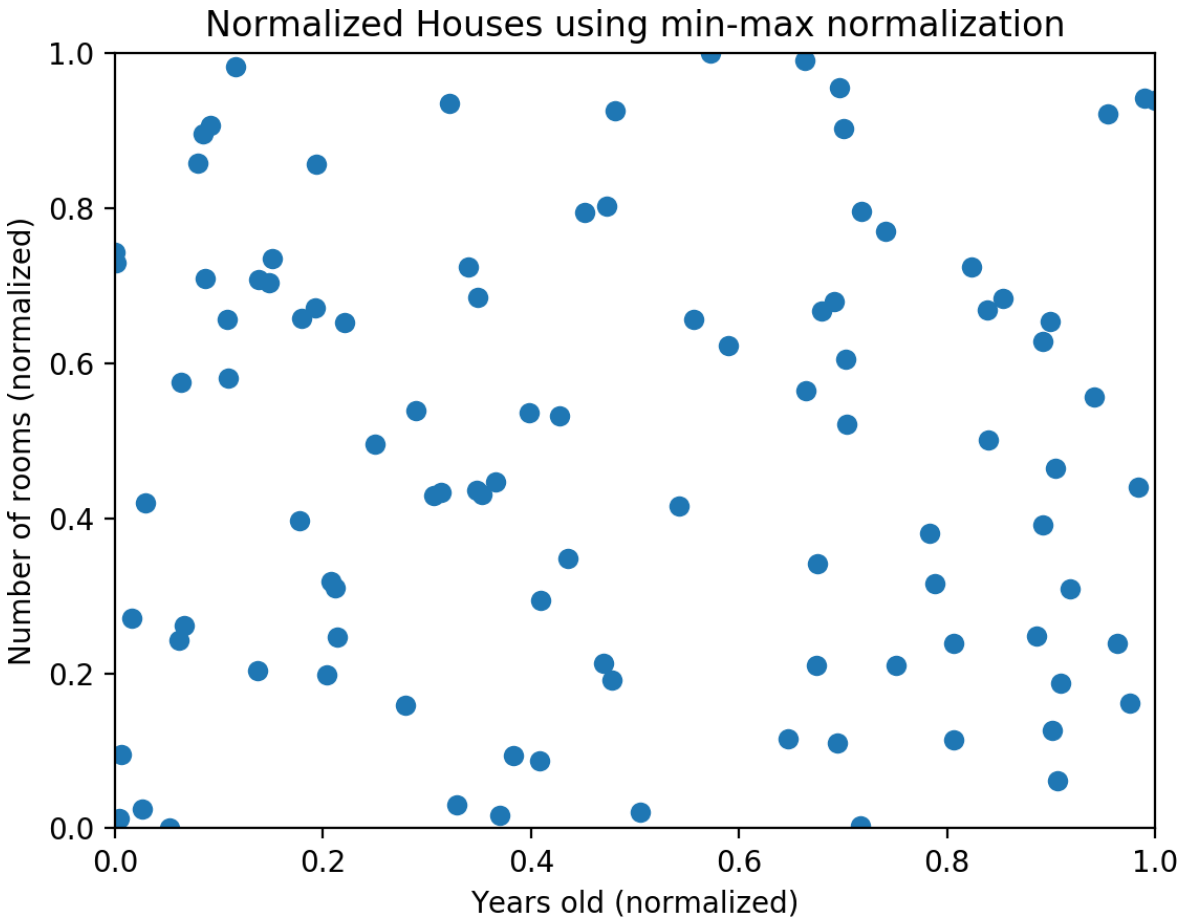
- Remove effects of different value ranges in each attribute/variable

- Common method in statistics and machine learning
  - With many different names...
  - Scaling, Normalisation, Standardisation, ...
  - Often used interchangely, different for each field ...

Scaling / Normalisation / Standardisation



Scaling / Normalisation / Standardisation



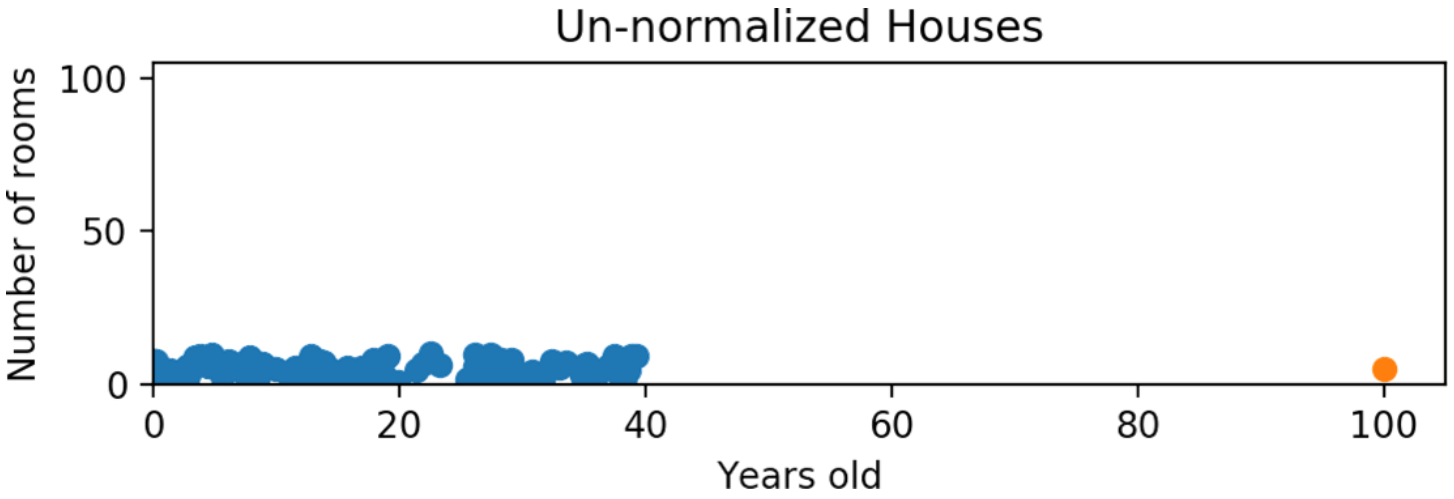
Scaling / Normalisation / Standardisation

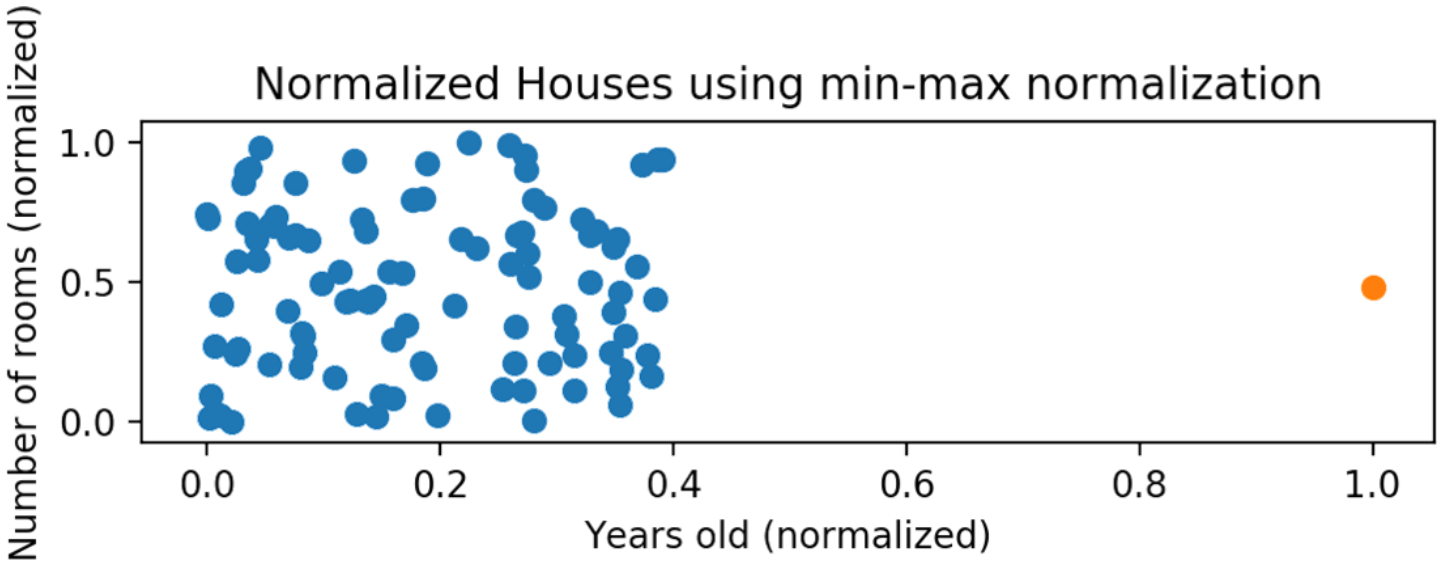
- Min-Max scaling
  - Scale all variables to the same (fixed) range
  - Often between 0 and 1
  - Subtract minimum value for each variable
  - Divide by value range of each variable

$$z_i = \frac{x_i - \min(X)}{\max(X) - \min(X)}$$

  - *Multiply by new range (if different than 0..1)*

Scaling / Normalisation / Standardisation





Scaling / Normalisation / Standardisation

- Z-score standardisation / normalization

- Each variable *x* is converted to have a mean of 0 and a standard deviation of 1
  - subtracting the mean
  - dividing by standard deviation

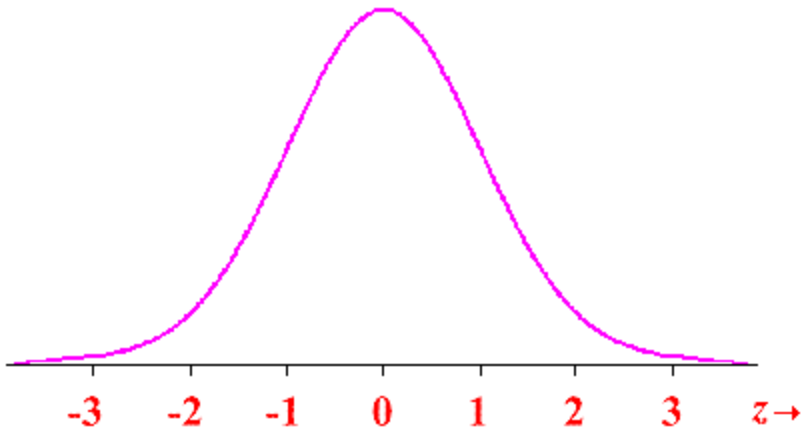
$$z_i = \frac{x_i - \mu}{\sigma}$$

Scaling / Normalisation / Standardisation

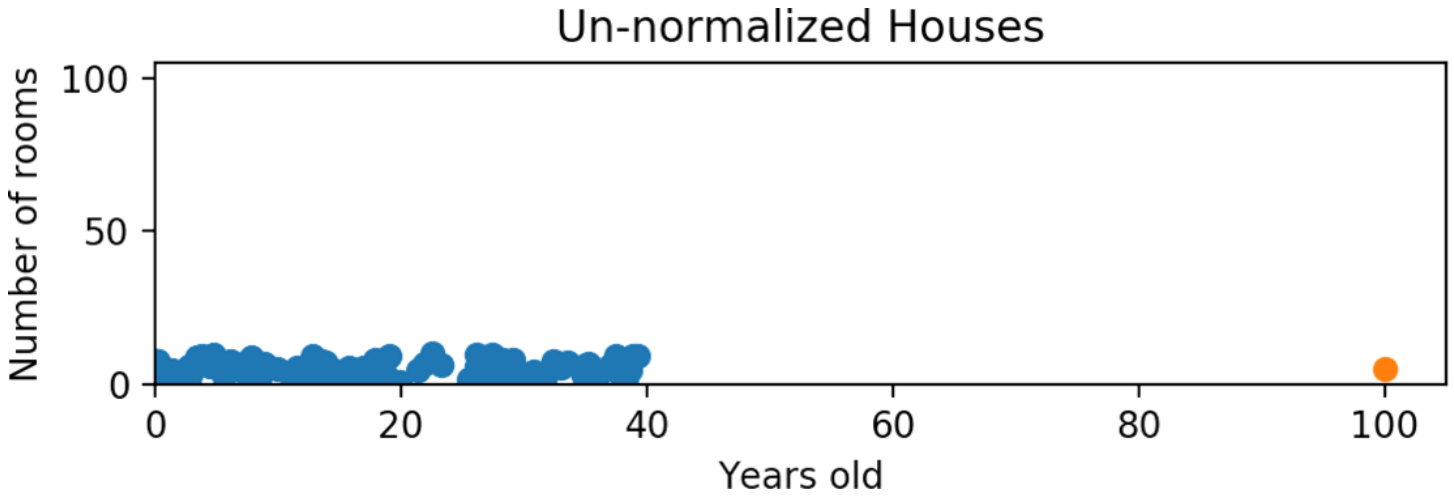
- Z-score standardisation / normalization

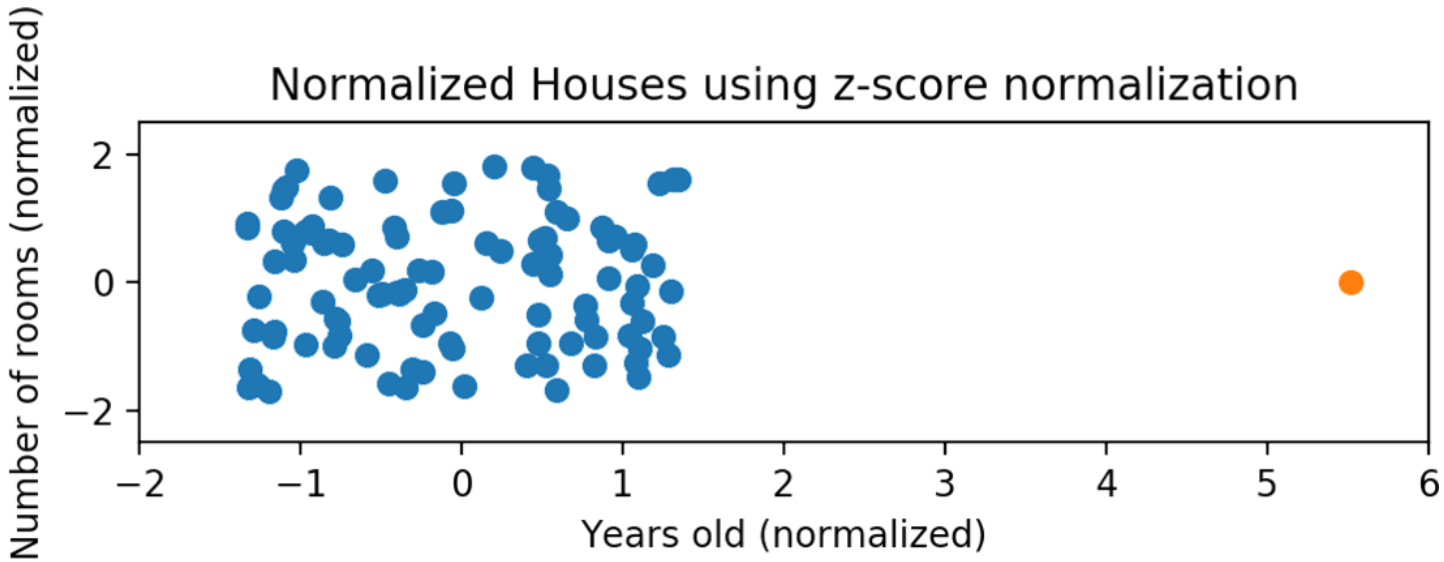
$$z_i = \frac{x_i - \mu}{\sigma}$$

- *What's the new value range after z-score?*



Scaling / Normalisation / Standardisation





Scaling / Normalisation / Standardisation

- *Is scaling an issue for*
  - *k-NN?*
  - *Perceptron?*
  - *Naïve Bayes?*

Scaling / Normalisation / Standardisation

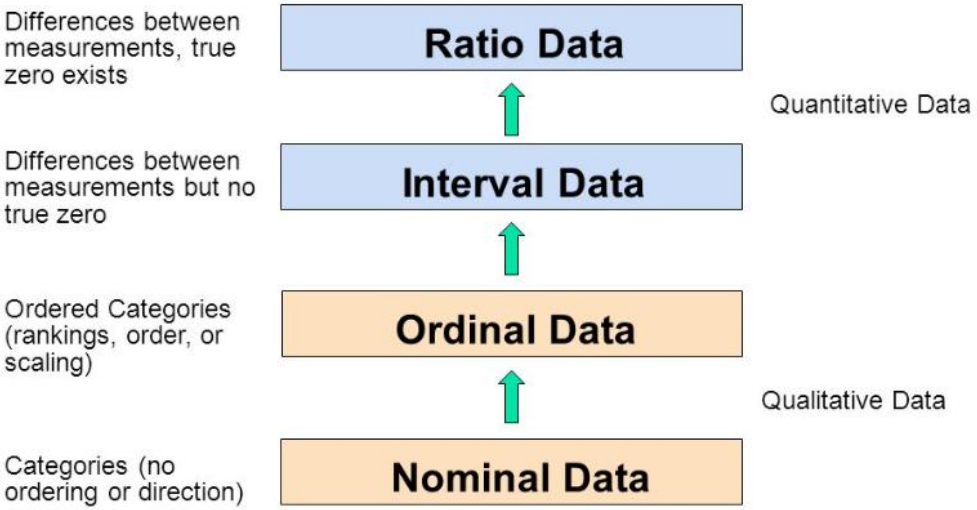
- Especially algorithms relying on distances
  - k-Nearest Neighbours
  - …
- Scaling not needed for algorithms that don't use distances, e.g.
  - Naïve Bayes
  - Decision trees
  - …
- Scaling useful in heuristic/iterative optimisation
  - E.g. Neural Networks via Gradient Descent

Scaling / Normalisation / Standardisation

- Especially algorithms relying on distances
  - k-Nearest Neighbours
  - ...

- *Caveat*
  - Many implementations already do this pre-processing implicitly
  - Check default settings carefully
    - If the implementation already does a z-score normalisation, applying it again before won't change anything….
    - And if you perform min-max before that, your results will be different …

Data Types



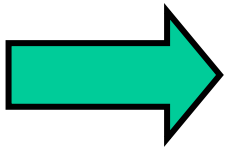
Can compute	Nominal	Ordinal	Interval	Ratio
Frequency distribution	Yes	Yes	Yes	Yes
Median and percentiles	No	Yes	Yes	Yes
Add or subtract	No	No	Yes	Yes
Mean, standard deviation, …	No	No	Yes	Yes
Ratios, coefficient of variation	No	No	No	Yes

- *How to use qualitative / categorical data for Perceptron, … ?*

One hot encoding (1-to-N Coding)

- Replace categorical variables with **n** new binary ones
  - n: number of different values (labels)

Colour
Brown
Blue
Green
Grey
Brown
Green
Blue



Green	Blue	Brown	Grey
0	0	1	0
0	1	0	0
1	0	0	0
0	0	0	1
0	0	1	0
1	0	0	0
0	1	0	0

Categorical data: Animals data set

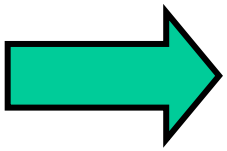
- Animal data set
  - Describes animal by some characteristics
  - Instances: cow, duck, bee, …
- Variables
  - Size (tiny, small, medium, big)
  - Number of legs (2, 4, 6, 8)
  - Feathers (yes/no)
  - Eggs (yes/no)

	size	legs	feathers	eggs
duck	small	2	yes	yes
dog	medium	4	no	no
spider	tiny	8	no	yes
ladybird	tiny	6	no	yes
cow	large	4	no	no
bee	tiny	6	no	no
sparrow	small	2	yes	yes

One hot encoding
Animals data set

- Animal data set: describes animal by some characteristics
  - Instances: cow, duck, bee, …

	size	legs	feathers	eggs
duck	small	2	yes	yes
dog	medium	4	no	no
spider	tiny	8	no	yes
ladybird	tiny	6	no	yes
cow	large	4	no	no
bee	tiny	6	no	no
sparrow	small	2	yes	yes



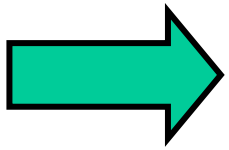
	tiny	small	med	large	legs	feathers	eggs
duck	0	1	0	0	2	1	1
dog	0	0	1	0	4	0	0
spider	1	0	0	0	8	0	1
ladybird	1	0	0	0	6	0	1
cow	0	0	0	1	4	0	0
bee	1	0	0	0	6	0	0
sparrow	0	1	0	0	2	1	1

- Replace “size” with four binary attributes
- Replace feathers & eggs with one binary attribute (0/1)

Label encoding: Animal data set

- Could also encode to numerical value: **label encoding**
  - E.g.: tiny=1, small=2, med=3, large=4 (if sizes are equally distant!)

	size	legs	feathers	eggs
duck	small	2	yes	yes
dog	medium	4	no	no
spider	tiny	8	no	yes
ladybird	tiny	6	no	yes
cow	large	4	no	no
bee	tiny	6	no	no
sparrow	small	2	yes	yes



	Size	legs	feathers	eggs
duck	2	2	1	1
dog	3	4	0	0
spider	1	8	0	1
ladybird	1	6	0	1
cow	4	4	0	0
bee	1	6	0	0
sparrow	2	2	1	1

Categorical data: another example

- *Any other pre-processing needed?*

	size	*legs*
duck	small	2
dog	medium	4
spider	tiny	8
ladybird	tiny	6
cow	large	4
bee	tiny	6
sparrow	small	2

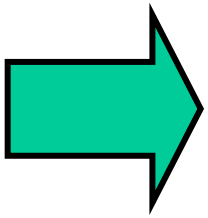
- Variable “legs”
  - If considered categorical: defined order ➔ ordinal data
  - Can compute similarity: 2 closer to 4 than to 6
  - Numerical value, can compute distance directly
  - *Does the number of legs denote similarity?*
    - *Is this an ordinal variable?*
    - *Is this a ratio-scale variable?*

One hot encoding animal data set

- *Does number of legs denote similarity?*
  - i.e., is an animal with 2 legs more similar to one with 4, or with 6?
  - And, is one with 4 equally similar to the one with 2 and 6?
    - Dog to monkey vs. dog to spider
  - One with 6 equally similar to one with 4 and 8?
    - Bee to spider vs. bee to cow

One hot encoding animal data set

	size	*legs*
duck	small	2
dog	medium	4
spider	tiny	8
ladybird	tiny	6
cow	large	4
bee	tiny	6
sparrow	small	2



2 legs	*4 legs*	*6 legs*	*8 legs*
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0
0	1	0	0
0	0	1	0
1	0	0	0

k-NN: distance function

- *What to do with categorical data?*

a) 1-n coding – then apply any distance function mentioned earlier

b) Definition of custom distance functions that applies to categorical data

k-NN: distance function

- Definition of custom distance functions
  - E.g. adapt hamming distance

    Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different

    ➔ count number of different nominal values

	size	*legs*	*feathers*	*eggs*	*distance*
horse	large	4	no	no	
duck	**small**	**2**	**yes**	**yes**	4
dog	**medium**	4	no	no	1
spider	**tiny**	**8**	no	**yes**	3
ladybird	**tiny**	**6**	no	**yes**	3
cow	large	4	no	no	0
bee	**tiny**	**6**	no	no	2
sparrow	**small**	**2**	**yes**	**yes**	4

k-NN: distance function

- Definition of custom distance functions
  - E.g. adapt hamming distance

    Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different

    ➔ count number of different nominal values

  - Define custom distance for **each attribute**, and then aggregate e.g. via sum

Other data preparation aspects

- Missing values
  - For some samples, not all attribute values are known
    - Some ML algorithms can handle missing values
    - For others, we need a special treatment: data imputation

➔ *More on this in upcoming lectures*

Questions?

Upcoming topics:

- Decision trees
- Random Forests

….