10-301/601: Introduction to Machine Learning Lecture 4 – Overfitting & KNNs

Matt Gormley & Henry Chai

9/9/24

Front Matter

- Announcements:
  - HW2 released 9/4, due 9/16 at 11:59 PM
    - Unlike HW1, you will only have:
      - 1 (graded) submission for the written portion
      - 10 submissions of the programming portion to our autograder

Recall: Decision Tree Pseudocode

```
def train(𝒟):
    store root = tree_recurse(𝒟)
def tree_recurse(𝒟′):
    q = new node()
    base case – if (SOME CONDITION):
    recursion – else:
        find best attribute to split on, x_d
        q.split = x_d
        for v in V(x_d), all possible values of x_d:
            𝒟_v = {(x^(n), y^(n)) ∈ 𝒟′ | x_d^(n) = v}
            q.children(v) = tree_recurse(𝒟_v)
    return q
```

$$\mathcal{D}_v = \left\{\left(x^{(n)}, y^{(n)}\right) \in \mathcal{D}' \mid x_d^{(n)} = v\right\}$$

Recall: Decision Tree Pseudocode

```
def train(𝒟):
    store root = tree_recurse(𝒟)
def tree_recurse(𝒟′):
    q = new node()
    base case – if (𝒟′ is empty OR
        all labels in 𝒟′ are the same OR
        all features in 𝒟′ are identical OR
        some other stopping criterion):
        q.label = majority_vote(𝒟′)


    recursion – else:
    return q
```

Q & A:

Wait, how could we end up calling `tree_recurse` with an empty dataset in the first place?

- Given some subset of our dataset, it could be the case that we choose to split on some feature where not every value that the feature could take on appears in the subset
  - This could happen if we know something about the feature *a priori* or we observe that the feature takes on more values in a different subset/the entire training dataset.
- In this case we would still want to make a branch for it in our decision tree because at inference time, some new data point might come along that goes down that branch

Q & A:

Okay, so what should we predict in leaf nodes with no training data?

- Well, there isn't really a majority label, so we could return a random label or a majority vote over the entire training dataset.
- This is related to the question of "what should we predict if some feature in our test dataset takes on a value we didn't observe in the training dataset?"
  - Going down a branch corresponding to an unseen feature value is like hitting a leaf node with no training data.

Bluntine & Niblett (1992) Splitting Criteria Experiments

Table 1. Properties of the data set

Data Set	Classes	Attr.s	Training Set	Test Set
hypo	4	29	1000	2772
breast	2	9	200	86
tumor	22	18	237	102
lymph	4	18	103	45
LED	10	7	200	1800
mush	2	22	200	7924
votes	2	17	200	235
votesl	2	16	200	235
iris	3	4	100	50
glass	7	9	100	114
xd6	2	10	200	400
pole	2	4	200	1647

Source: https://link.springer.com/content/pdf/10.1023/A:1022686419106.pdf

Bluntine & Niblett (1992) Splitting Criteria Experiments

Table 3. Error for different splitting rules (pruned trees).

Data Set	Splitting Rule		
	GINI	Info. Gain	Random
hypo	1.01 ± 0.29	0.95 ± 0.22	7.44 ± 0.53
breast	28.66 ± 3.87	28.49 ± 4.28	29.65 ± 4.97
tumor	60.88 ± 5.44	62.70 ± 3.89	67.94 ± 5.68
lymph	24.44 ± 6.92	24.00 ± 6.87	32.33 ± 11.25
LED	33.77 ± 3.06	32.89 ± 2.59	38.18 ± 4.57
mush	1.44 ± 0.47	1.44 ± 0.47	8.77 ± 4.65
votes	4.47 ± 0.95	4.57 ± 0.87	12.40 ± 4.56
votes1	12.79 ± 1.48	13.04 ± 1.65	15.62 ± 2.73
iris	5.00 ± 3.08	4.90 ± 3.08	14.20 ± 6.77
glass	39.56 ± 6.20	50.57 ± 6.73	53.20 ± 5.01
xd6	22.14 ± 3.23	22.17 ± 3.36	31.86 ± 3.62
pole	15.43 ± 1.51	15.47 ± 0.88	26.38 ± 6.92

Key takeaway: Gini impurity and Information gain (aka mutual information) are nearly identical

Source: https://link.springer.com/content/pdf/10.1023/A:1022686419106.pdf

Bluntine & Niblett (1992) Splitting Criteria Experiments

Table 4. Difference and significance of error for GINI splitting rule versus others.

Data Set	Splitting Rule: Info. Gain	Splitting Rule: Random
hypo	−0.06 (0.82)	6.43 (1.00)
breast	−0.17 (0.23)	0.99 (0.72)
tumor	1.81 (0.84)	7.06 (0.99)
lymph	−0.44 (0.83)	7.89 (0.99)
LED	0.12 (0.17)	5.41 (0.99)
mush	0.00 (0.00)	7.32 (0.99)
votes	0.11 (0.55)	7.94 (0.99)
votes1	0.26 (0.47)	2.83 (0.99)
iris	−0.10 (0.67)	9.20 (0.99)
glass	1.01 (0.50)	13.64 (0.99)
xd6	0.04 (0.11)	9.72 (0.99)
pole	0.03 (0.11)	10.95 (0.99)

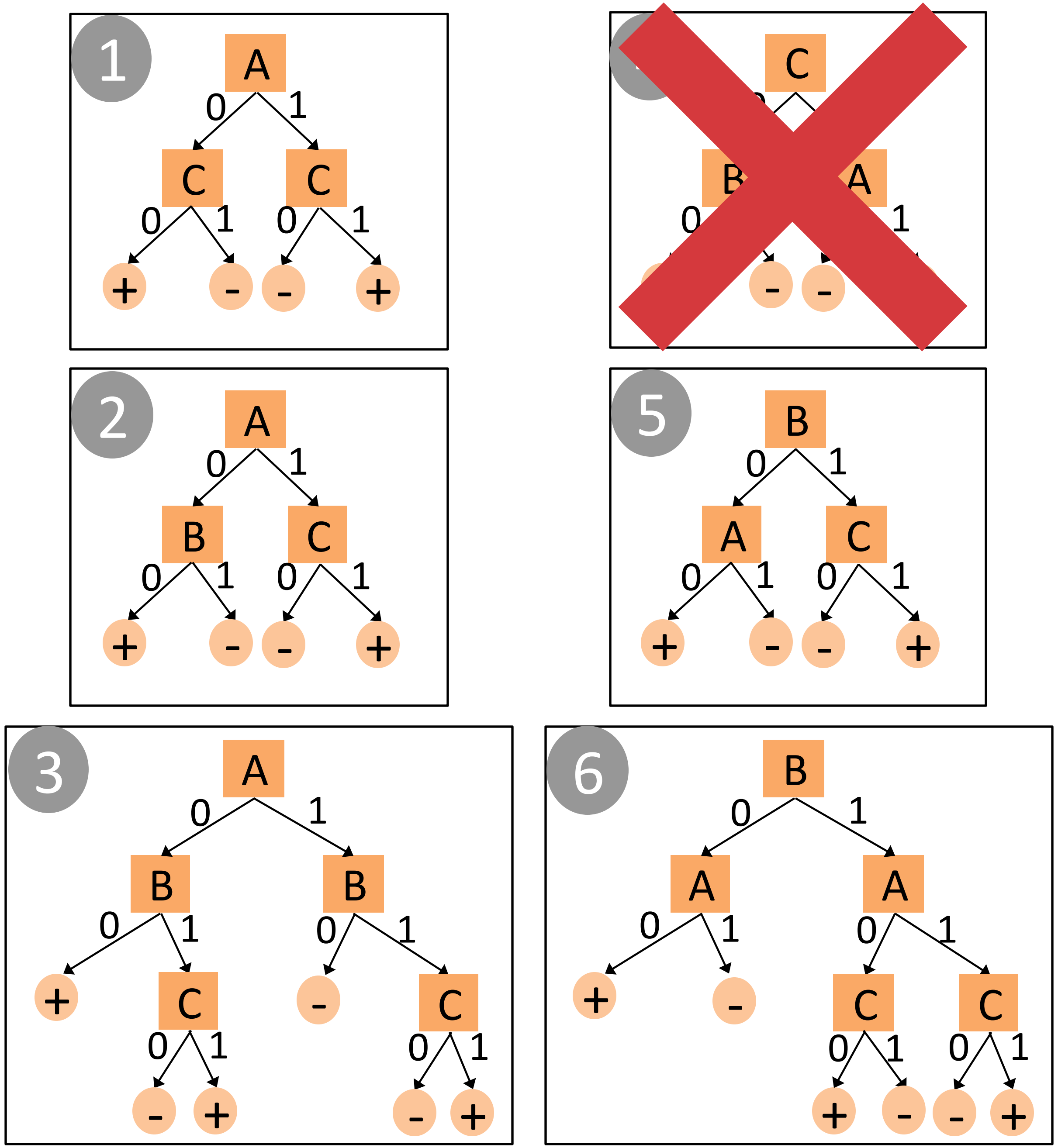
average difference in error between the splitting criteria

statistical significance of the difference according to a two-sided paired t-test

Source: https://link.springer.com/content/pdf/10.1023/A:1022686419106.pdf

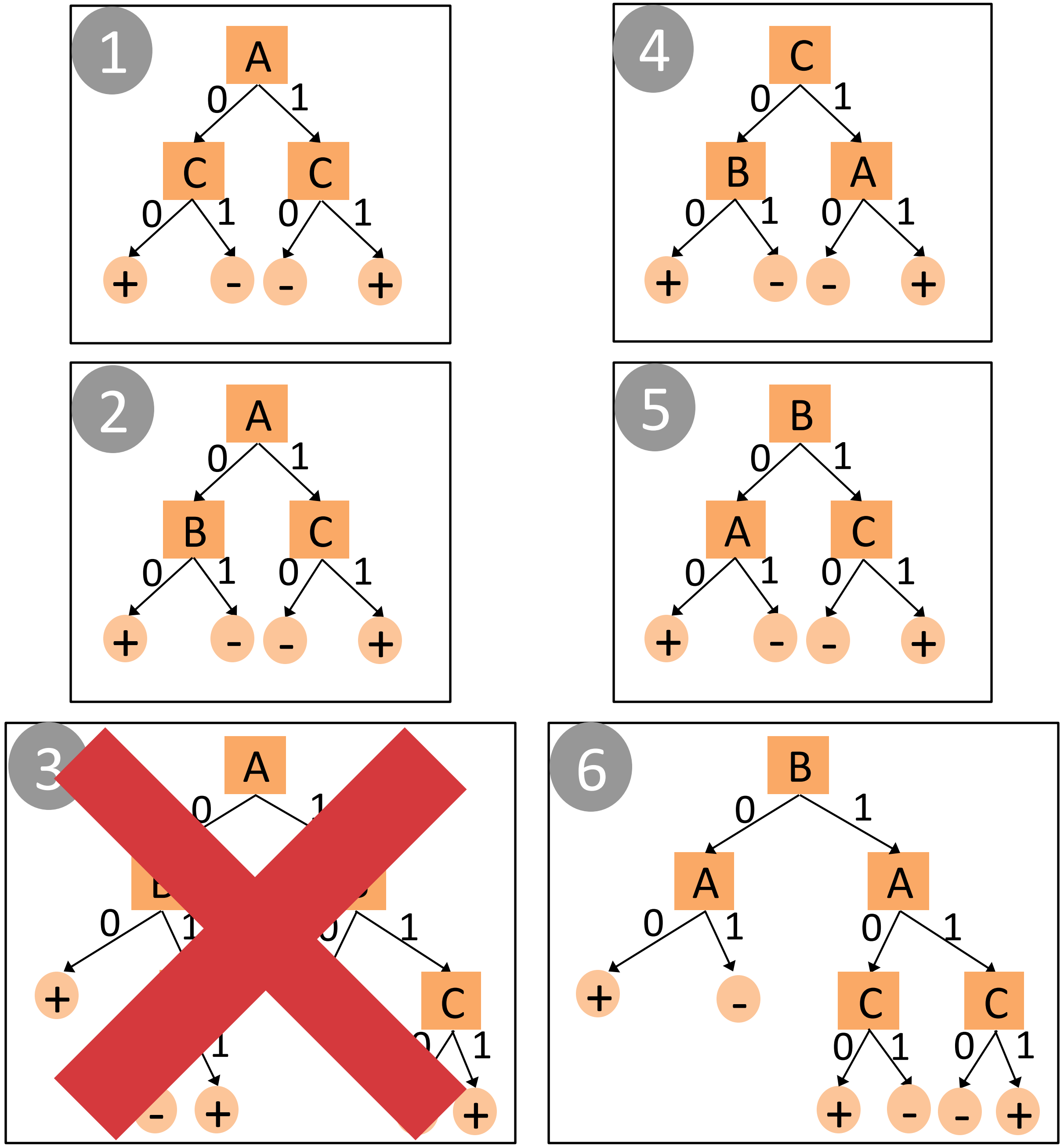
Poll Question 1: Which decision tree would you learn if you used training error rate as the splitting criterion? Break ties alphabetically.

A	B	C	y
0	0	0	+
0	0	1	+
0	1	0	–
0	1	1	+
1	0	0	–
1	0	1	–
1	1	0	–
1	1	1	+



Poll Question 2: Which decision tree is the smallest decision tree that achieves the lowest possible training error?

A	B	C	y
0	0	0	+
0	0	1	+
0	1	0	–
0	1	1	+
1	0	0	–
1	0	1	–
1	1	0	–
1	1	1	+



Decision Trees: Inductive Bias

- The **inductive bias** of a machine learning algorithm is the principal by which it generalizes to unseen examples
- What is the inductive bias of the ID3 algorithm?
  - Try to find the smallest decision tree that achieves low/zero training error with high mutual information features at the top

Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.

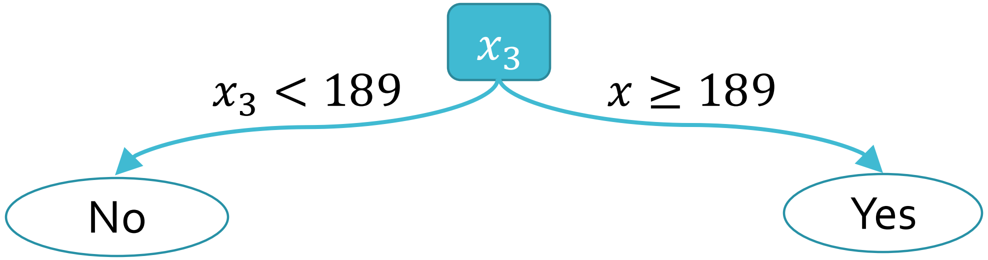
Real-Valued Features: Example

x_3 Cholesterol	y Heart Disease?
174	No
155	No
163	Yes
233	Yes
197	Yes
181	No
244	Yes
245	No
178	No
221	Yes

x_3 Cholesterol	y Heart Disease?
155	No
163	Yes
174	No
178	No
181	No
197	Yes
221	Yes
233	Yes
244	Yes
245	No

$x_3 < 189$

$x_3 \leq 181$



Real-Valued Features: Example

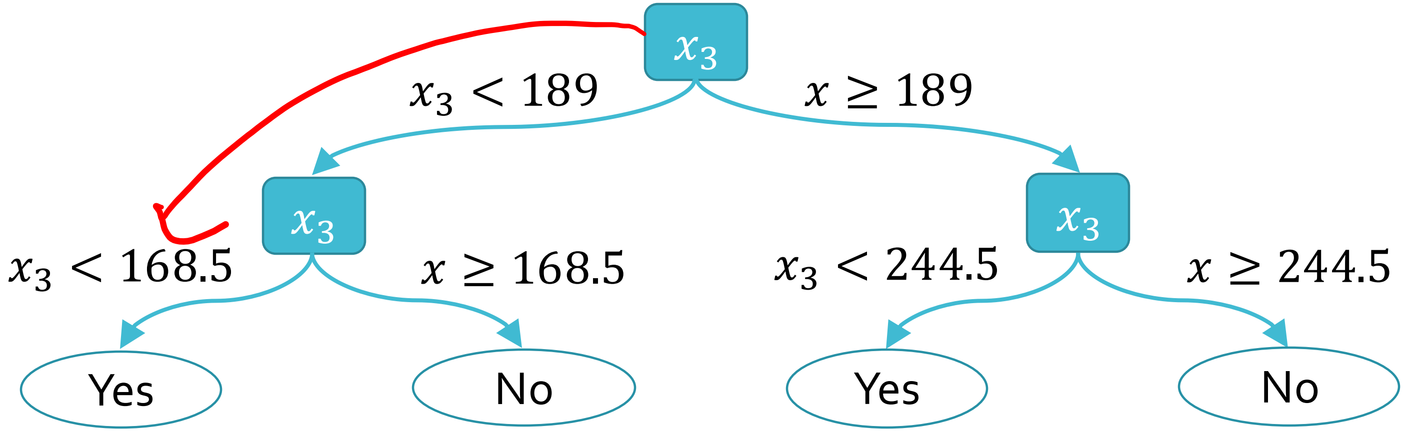
x_3 Cholesterol	y Heart Disease?
174	No
155	No
163	Yes
233	Yes
197	Yes
181	No
244	Yes
245	No
178	No
221	Yes

x_3 Cholesterol	y Heart Disease?
155	No
163	Yes
174	No
178	No
181	No
197	Yes
221	Yes
233	Yes
244	Yes
245	No

$x_3 < 168.5$

$x_3 < 189$

$x_3 < 244.5$



Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Liable to overfit!

Overfitting

- Overfitting occurs when the classifier (or model)…
  - is too complex
  - fits noise or “outliers” in the training dataset as opposed to the actual pattern of interest
  - doesn’t have enough inductive bias pushing it to generalize e.g., the memorizer
- Underfitting occurs when the classifier (or model)…
  - is too simple
  - can’t capture the actual pattern of interest in the training dataset
  - has too much inductive bias e.g., the majority vote classifier

Recall: Different Kinds of Error

- Training error rate = $err(h, \mathcal{D}_{train})$
- Test error rate = $err(h, \mathcal{D}_{test})$
- True error rate = $err(h)$

  = the error rate of h on all possible examples
  - In machine learning, this is the quantity that we care about but, in most cases, it is unknowable.
- Overfitting occurs when $err(h) > err(h, \mathcal{D}_{train})$
  - $err(h) - err(h, \mathcal{D}_{train})$ can be thought of as a measure of overfitting

Overfitting in Decision Trees

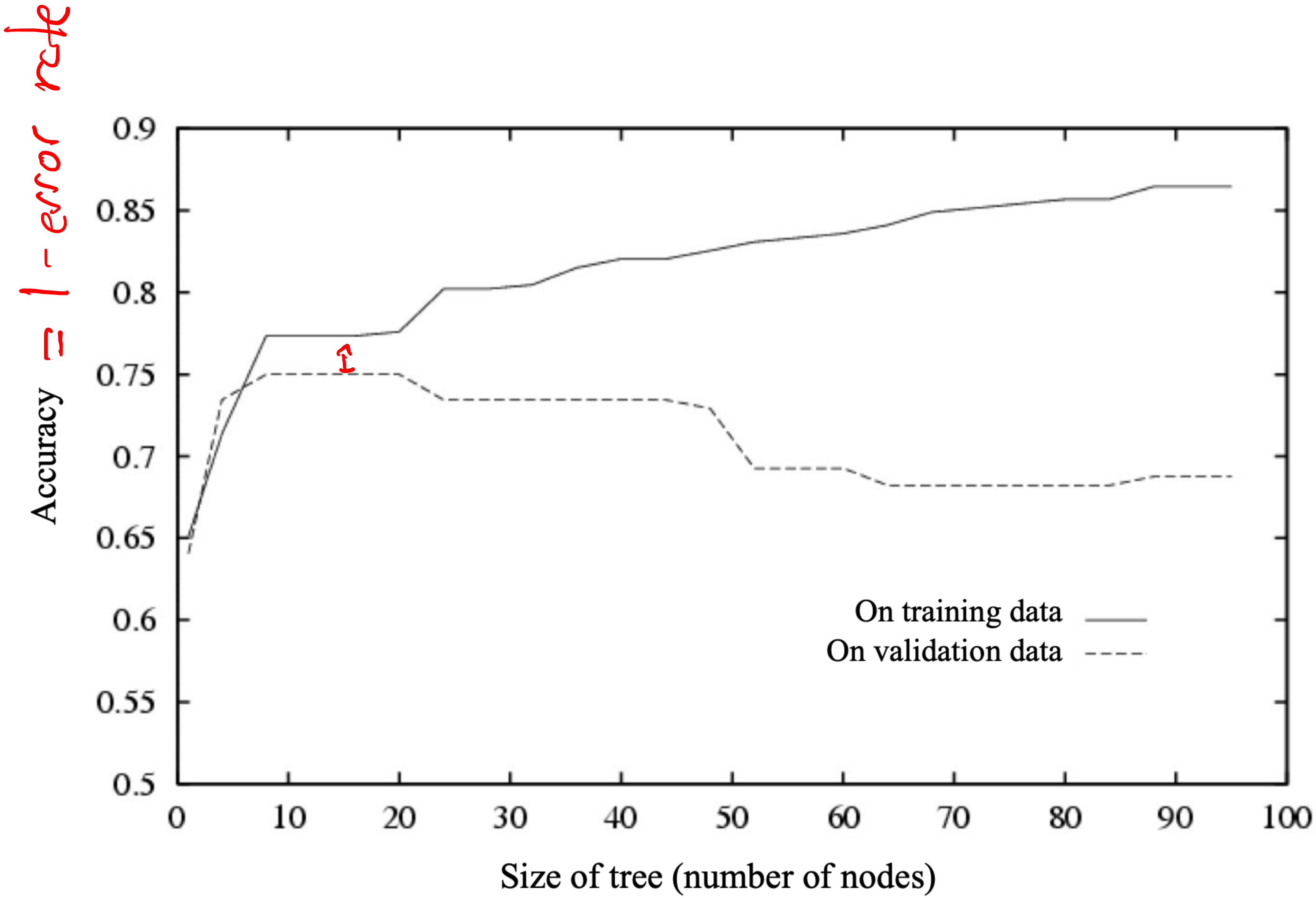


Figure courtesy of Tom Mitchell

Combatting Overfitting in Decision Trees

- Heuristics:
  - Do not split leaves past a fixed depth, δ
  - Do not split leaves with fewer than c data points
  - Do not split leaves where the maximal information gain is less than τ
- Take a majority vote in impure leaves

Combatting Overfitting in Decision Trees

- Pruning:
  - First, learn a decision tree
  - Then, evaluate each split using a “validation” dataset by comparing the validation error rate with and without that split
  - Greedily remove the split that most decreases the validation error rate
  - Stop if no split is removed

Pruning Decision Trees

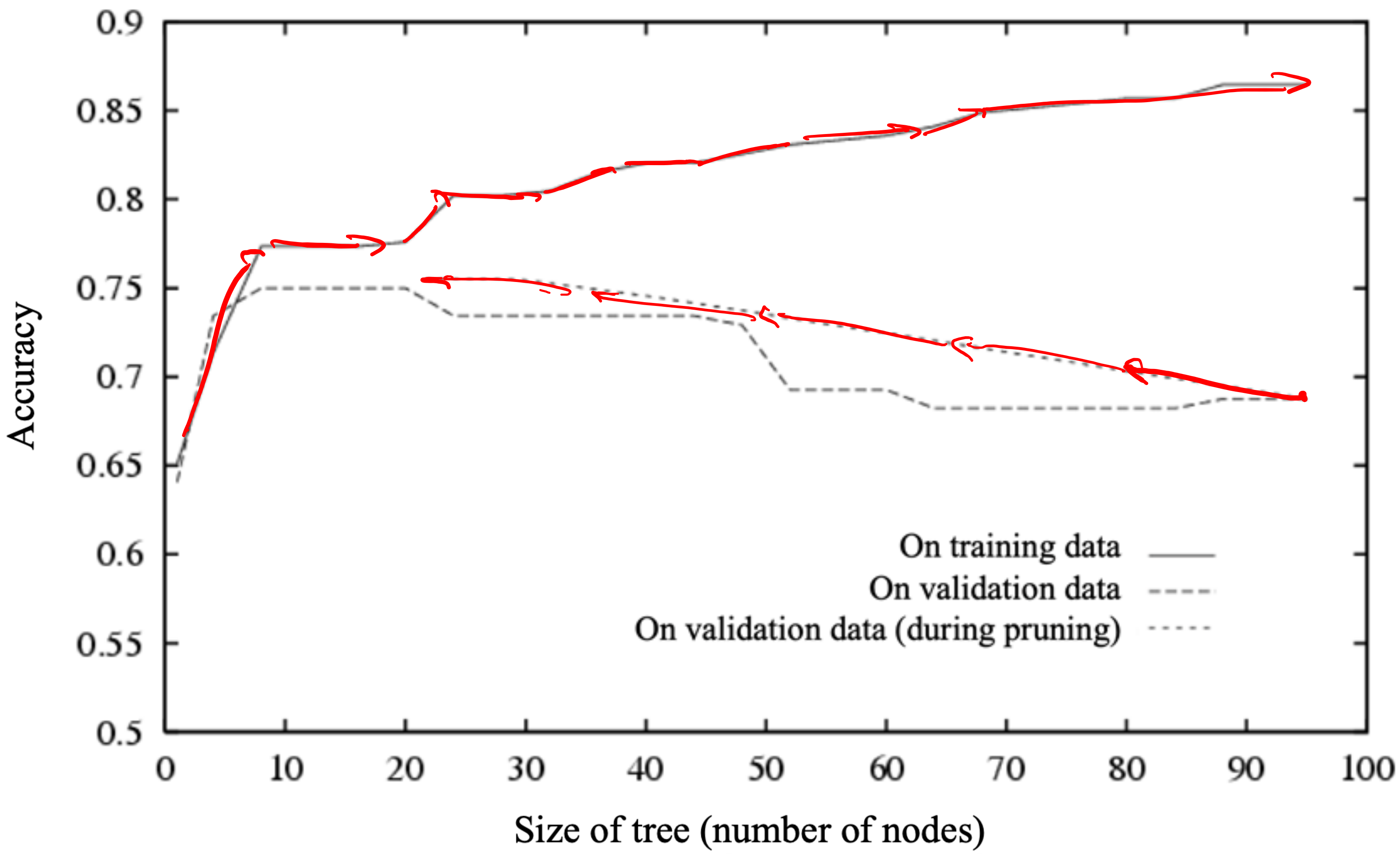


Figure courtesy of Tom Mitchell

Decision Tree Learning Objectives

You should be able to…

1. Implement decision tree training and prediction
2. Use effective splitting criteria for decision trees and be able to define entropy, conditional entropy, and mutual information / information gain
3. Explain the difference between memorization and generalization [CIML]
4. Describe the inductive bias of a decision tree
5. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
6. Explain the difference between true error and training error
7. Judge whether a decision tree is "underfitting" or "overfitting"
8. Implement a pruning or early stopping method to combat overfitting in decision tree learning

Real-valued Features

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

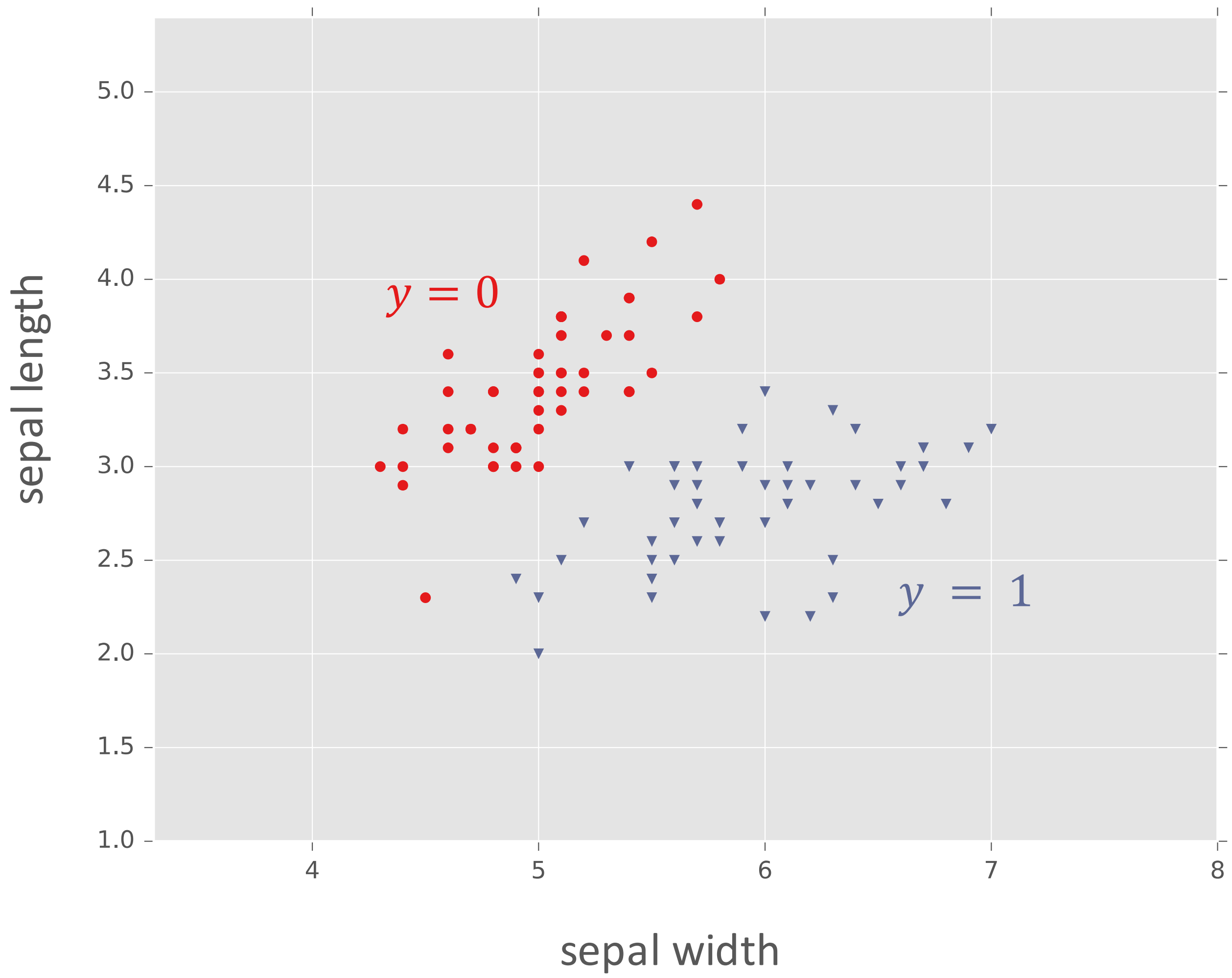
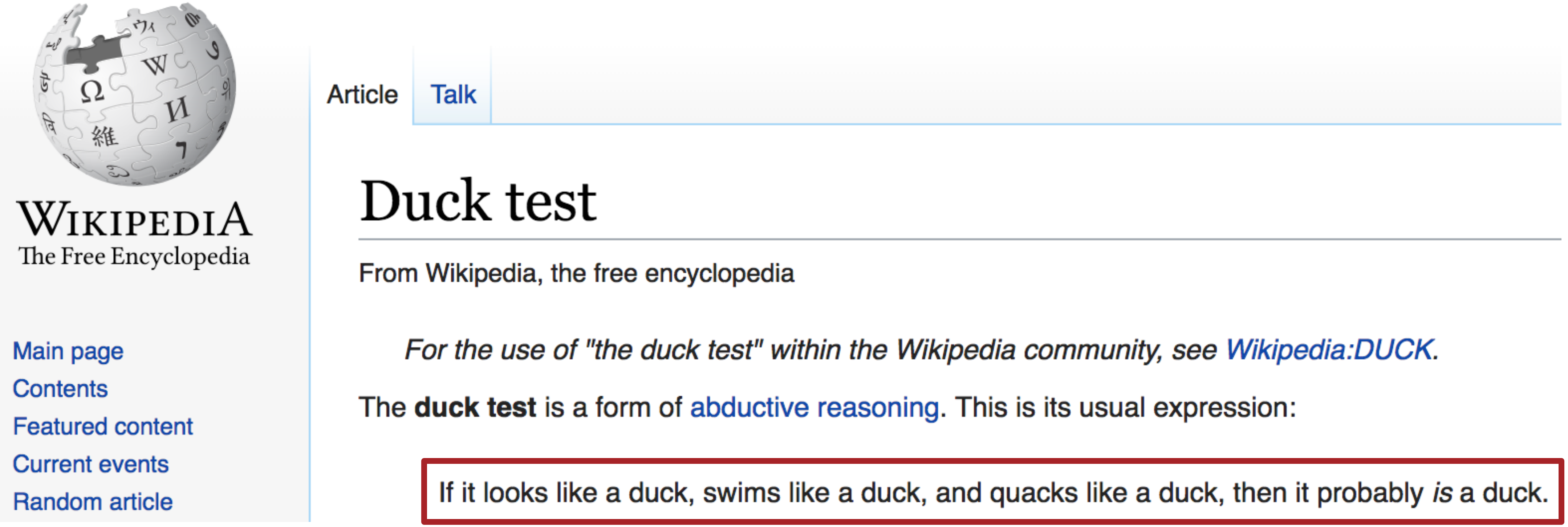


Figure courtesy of Matt Gormley


WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article

Article | Talk

Duck test

From Wikipedia, the free encyclopedia

For the use of "the duck test" within the Wikipedia community, see Wikipedia:DUCK.

The **duck test** is a form of abductive reasoning. This is its usual expression:

If it looks like a duck, swims like a duck, and quacks like a duck, then it probably *is* a duck.

The Duck Test

Source: https://en.wikipedia.org/wiki/Duck_test

kNN: In-class Activity

1. If you have either a red or blue folder, hold it above your head, high in the air
2. Note the location of the manilla folder
3. Raise your hand high in the air if you believe the manilla folder should be classified as red
4. Keep your hand raised until you believe the manilla folder should be classified as blue; then put it down

The Duck Test for Machine Learning

- Classify a point as the label of the "most similar" training point
- Idea: given real-valued features, we can use a distance metric to determine how similar two data points are
- A common choice is Euclidean distance:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2 = \sqrt{\sum_{d=1}^{D} (x_d - x'_d)^2}$$

- An alternative is the Manhattan distance:

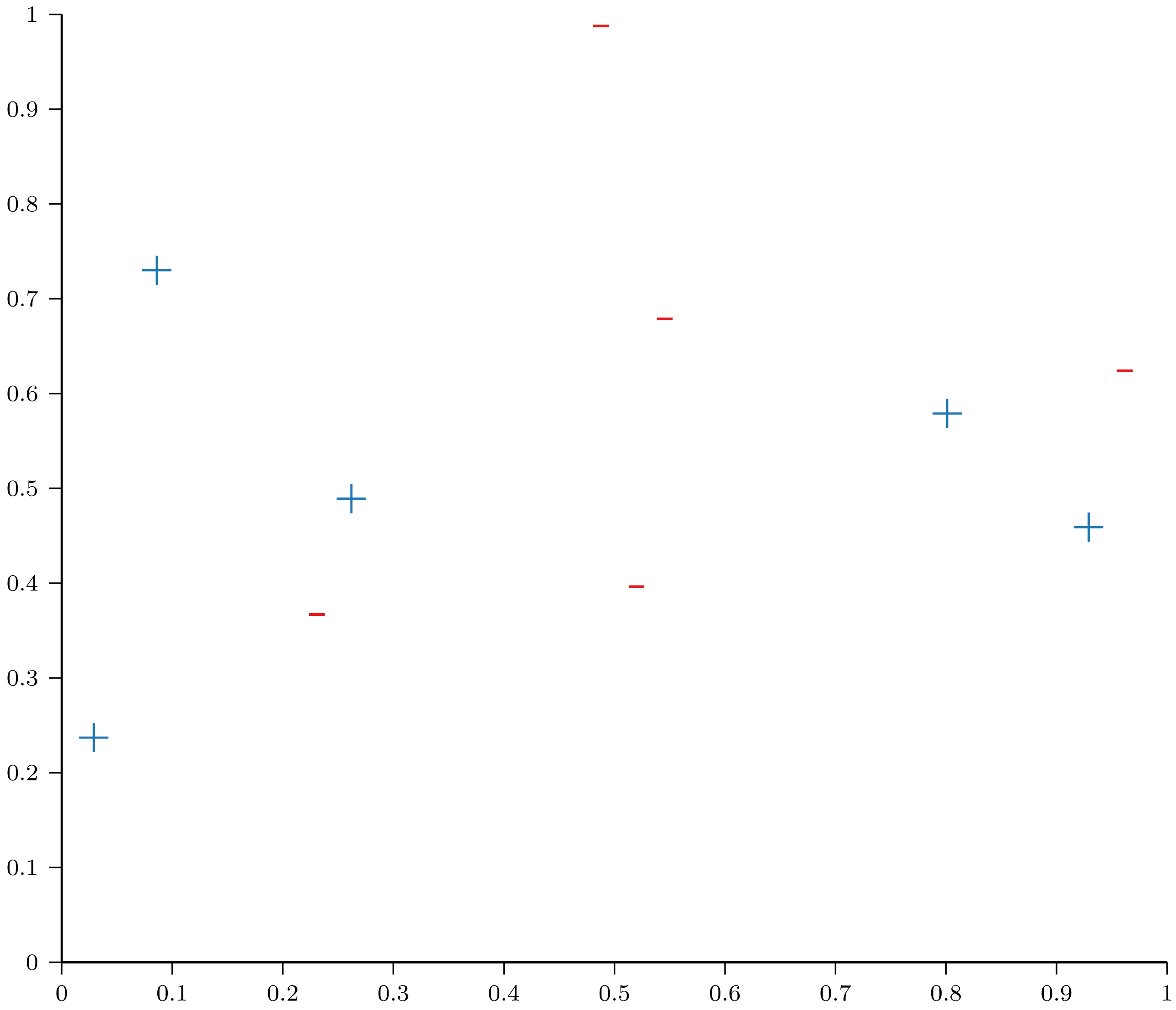
$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \sum_{d=1}^{D} |x_d - x'_d|$$

Nearest Neighbor: Pseudocode

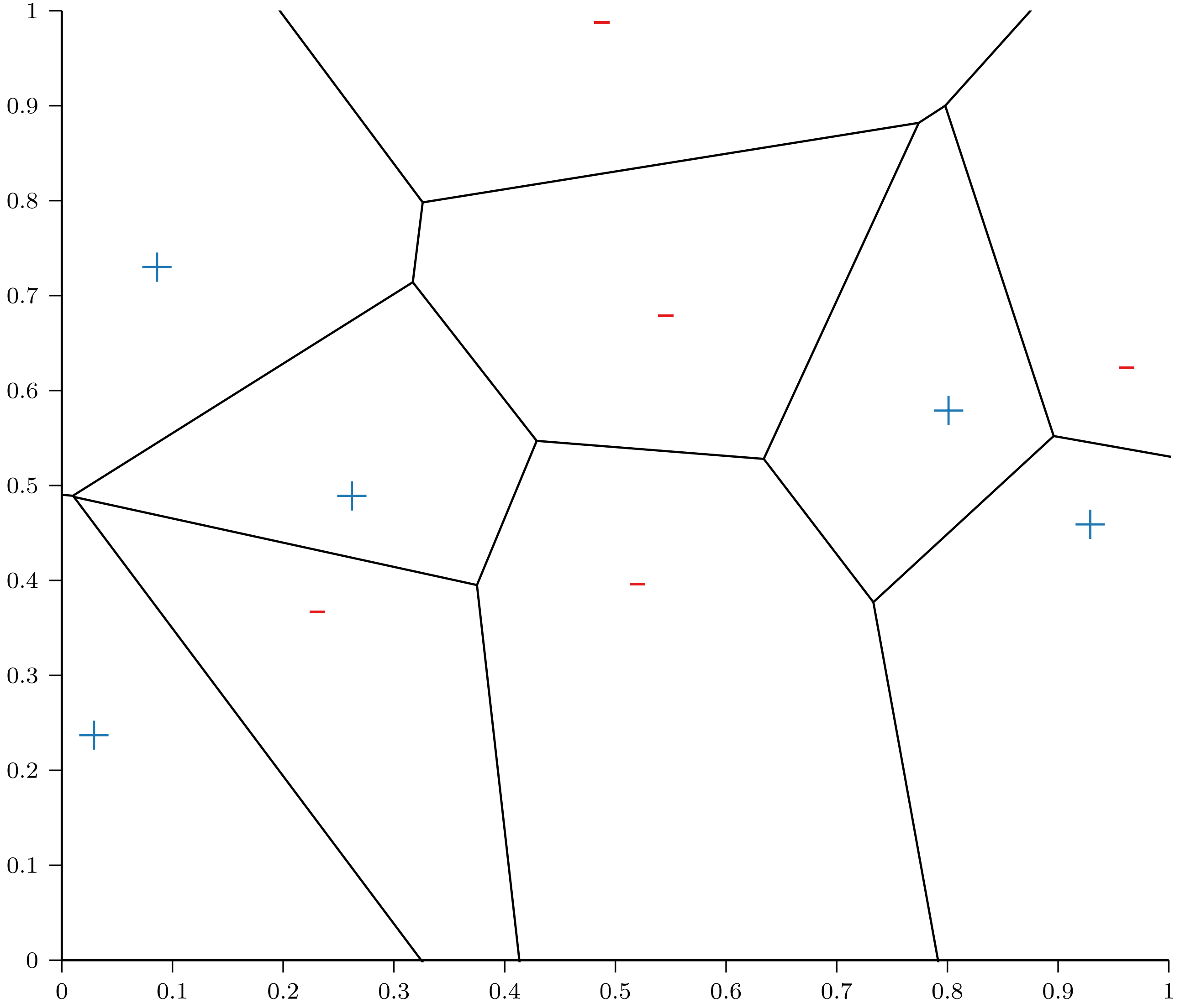
def train(D_{train}):
    store D_{train}

def h($\vec{x}'$):
    find the nearest data point to $\vec{x}'$ in D_{train}, $(\vec{x}^{(i)}, y^{(i)})$
    return $y^{(i)}$

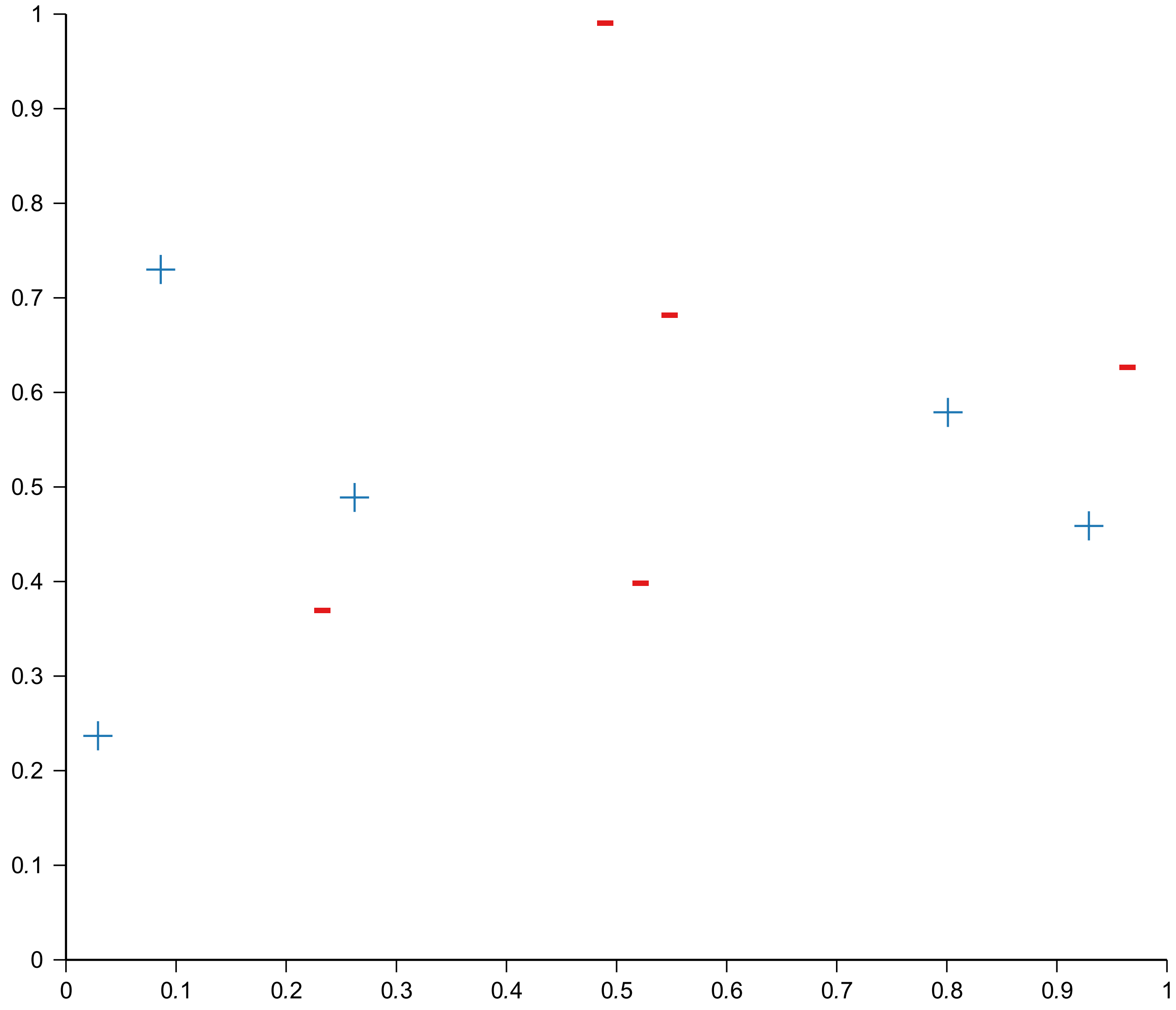
Nearest Neighbor: Example



Nearest Neighbor: Example



Nearest Neighbor: Example



The Nearest Neighbor Model

- Requires no training!
- Always has zero training error!
  - ***A data point is always its own nearest neighbor***