Decision Tree (Continued) and K-Nearest Neighbour

Dr. Xiaowei Huang

https://cgi.csc.liv.ac.uk/~xiaowei/

Up to now,

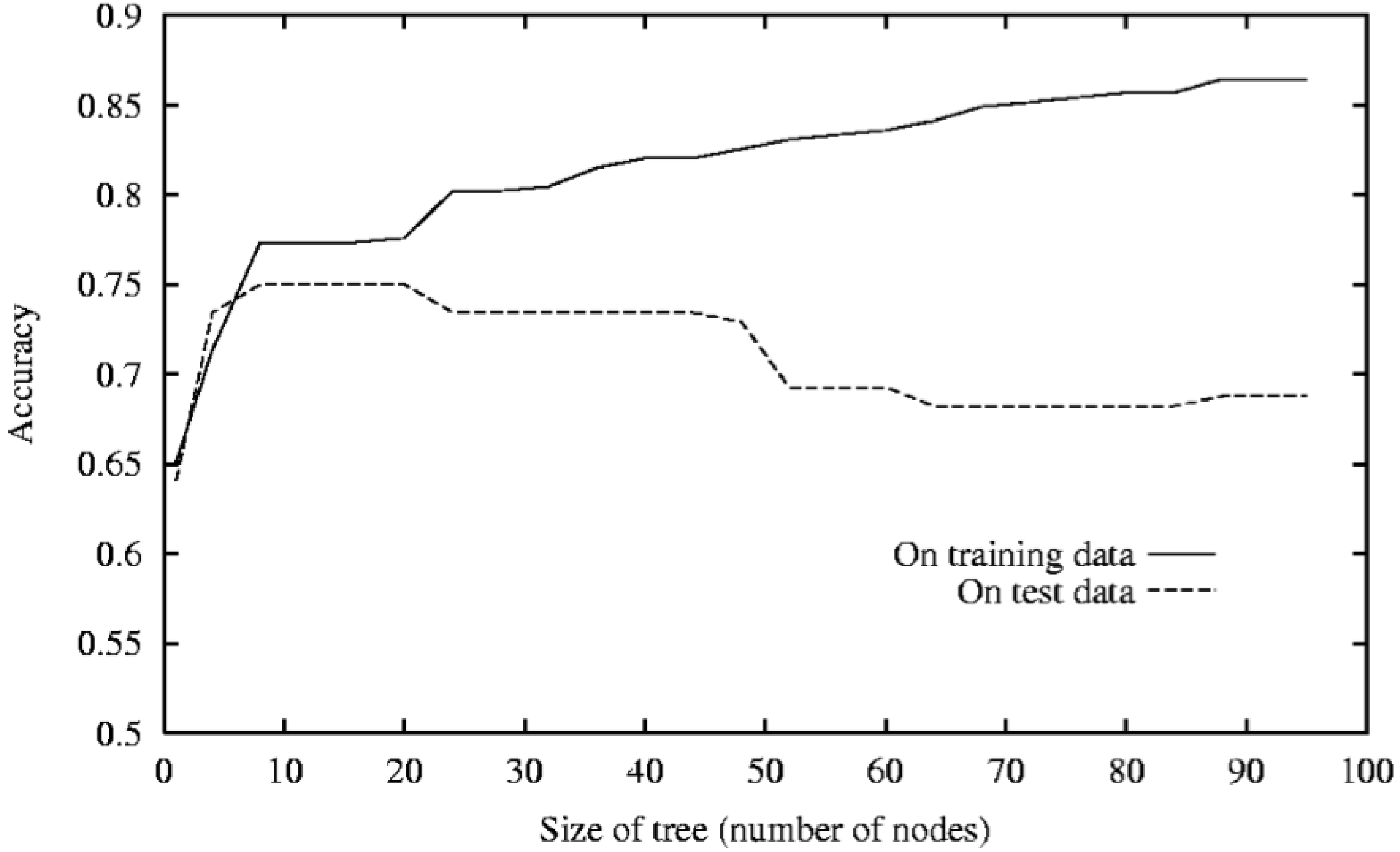
- Recap basic knowledge
- Decision tree learning
  - How to split
  - Identify the best feature to split
  - Accuracy and overfitting

Today’s Topics

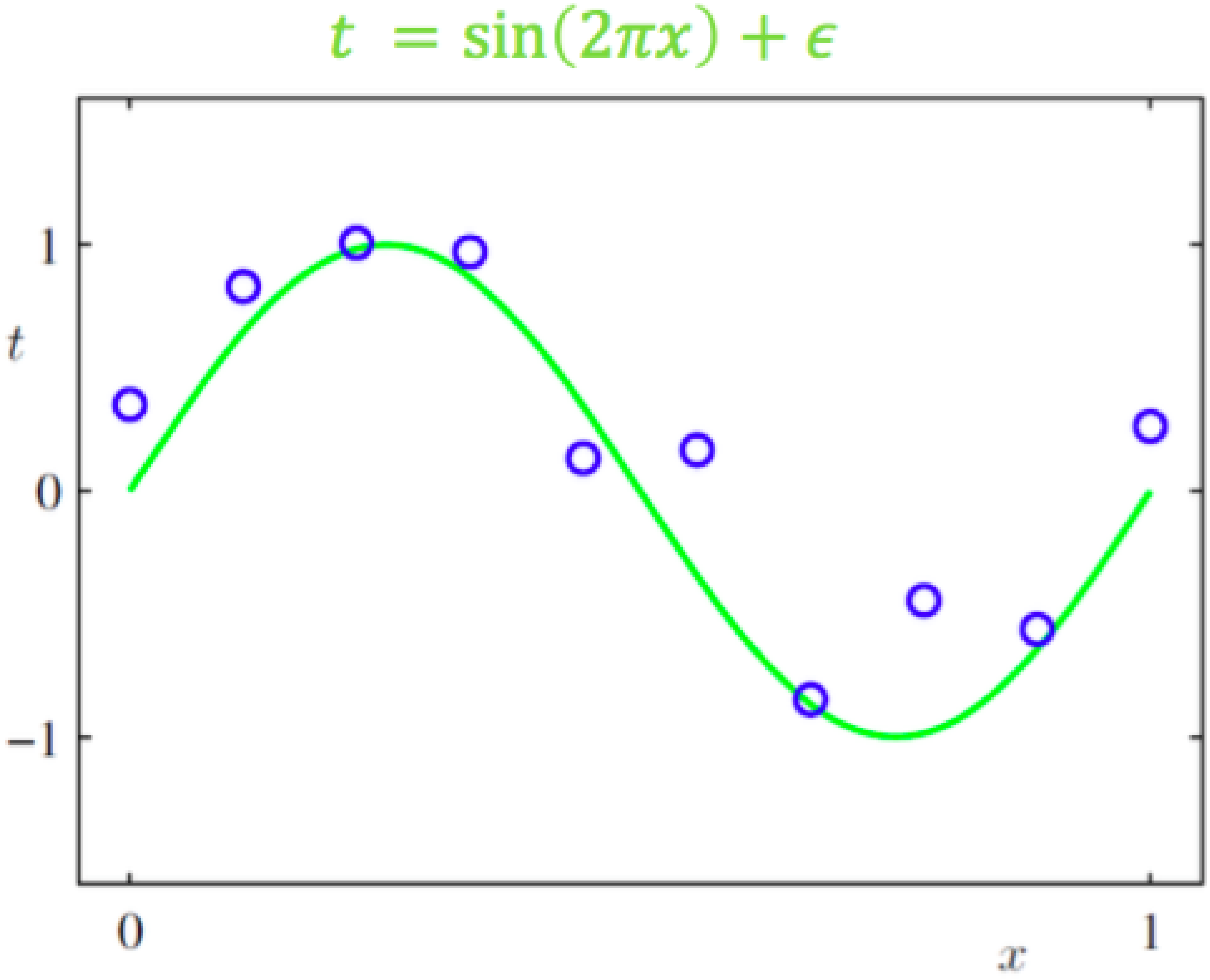
- Decision tree
  - Overfitting (continued) and stopping criteria
- k-NN classification

Overfitting (Continued) and Stopping Criteria

Overfitting in decision trees



Example 3: regression using polynomial

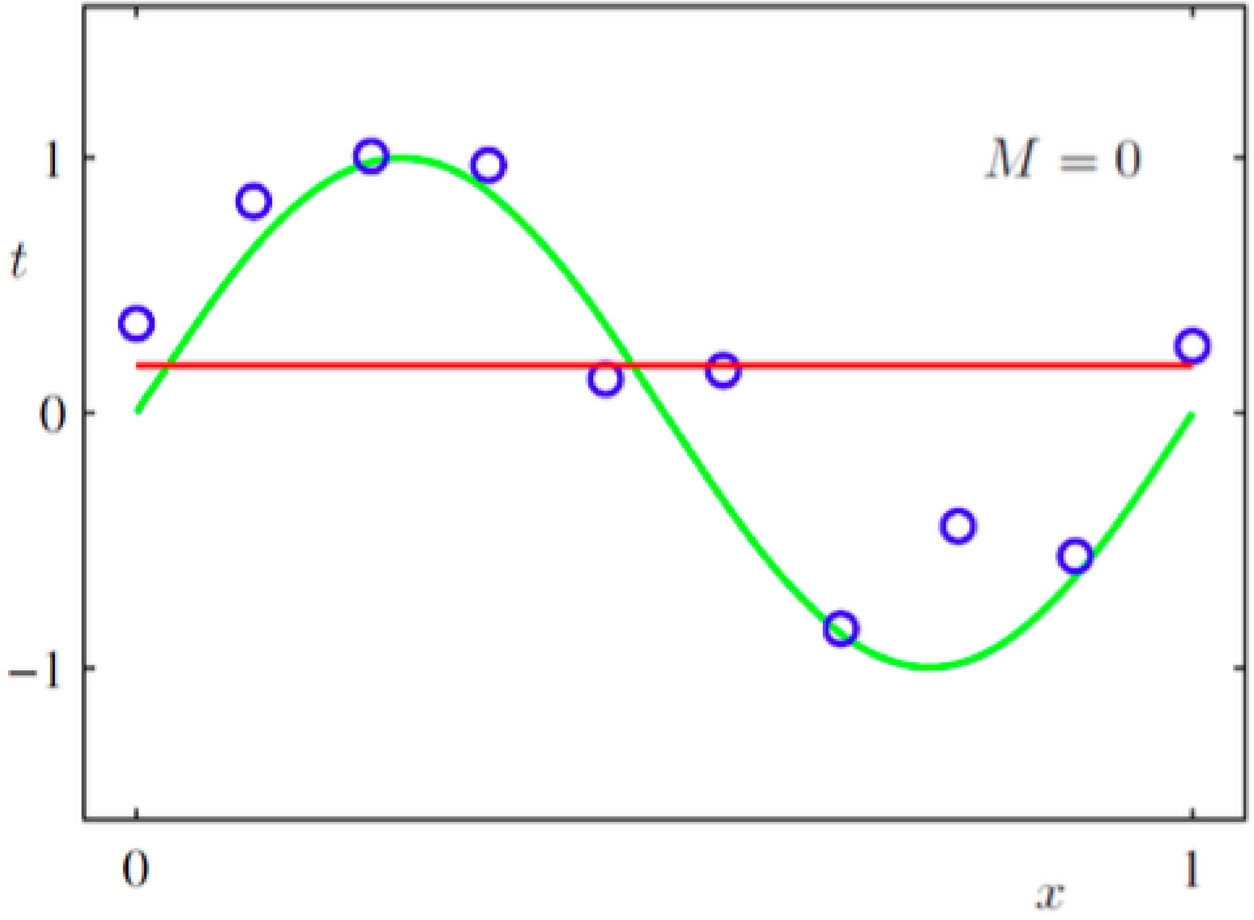


Regression using polynomial of degree M

$y = w_M x^M + w_{M-1} x^{M-1} + \ldots + w_1 x + w_0$

Example 3: regression using polynomial

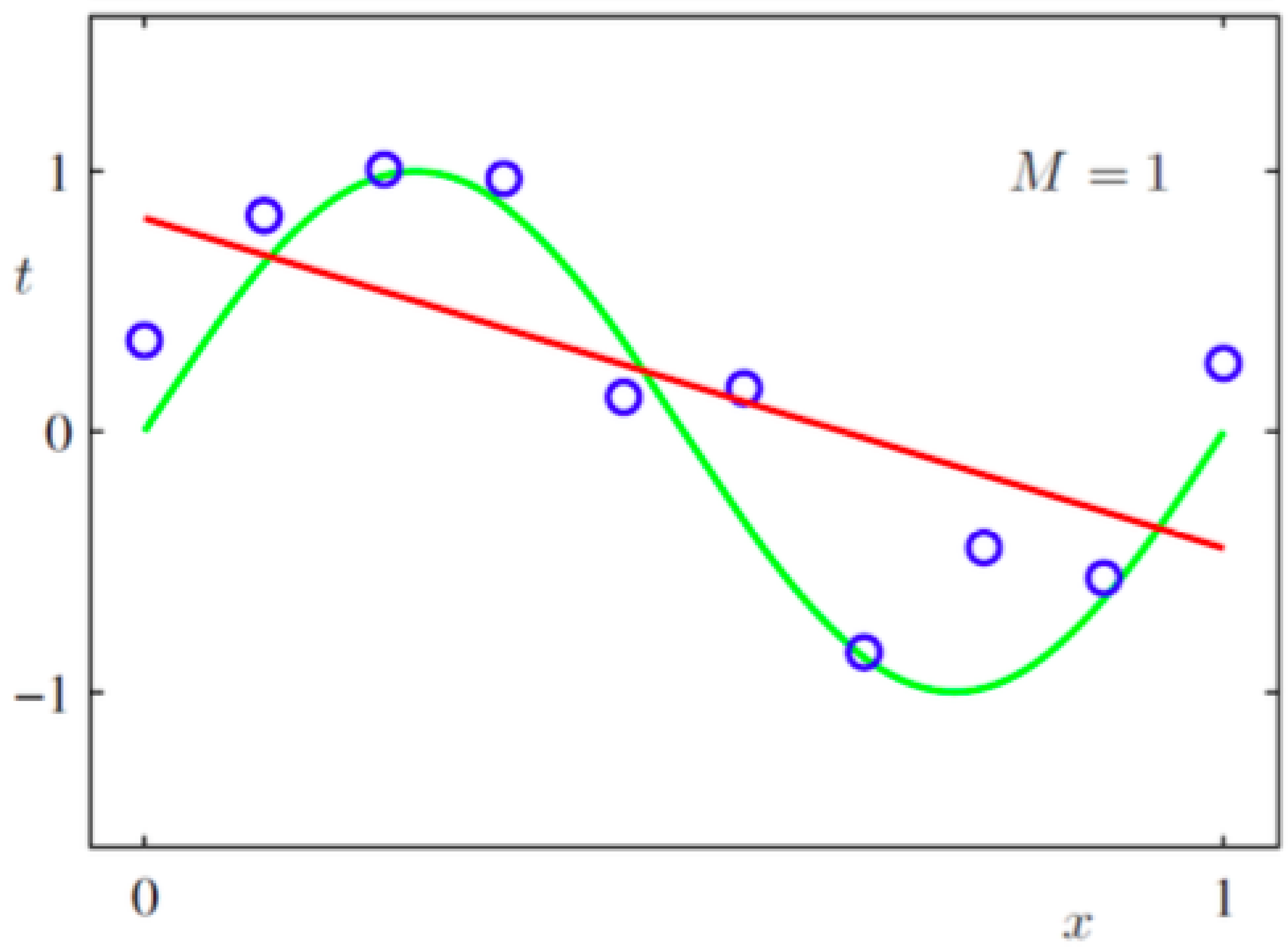
$$t = \sin(2\pi x) + \epsilon$$



y = c

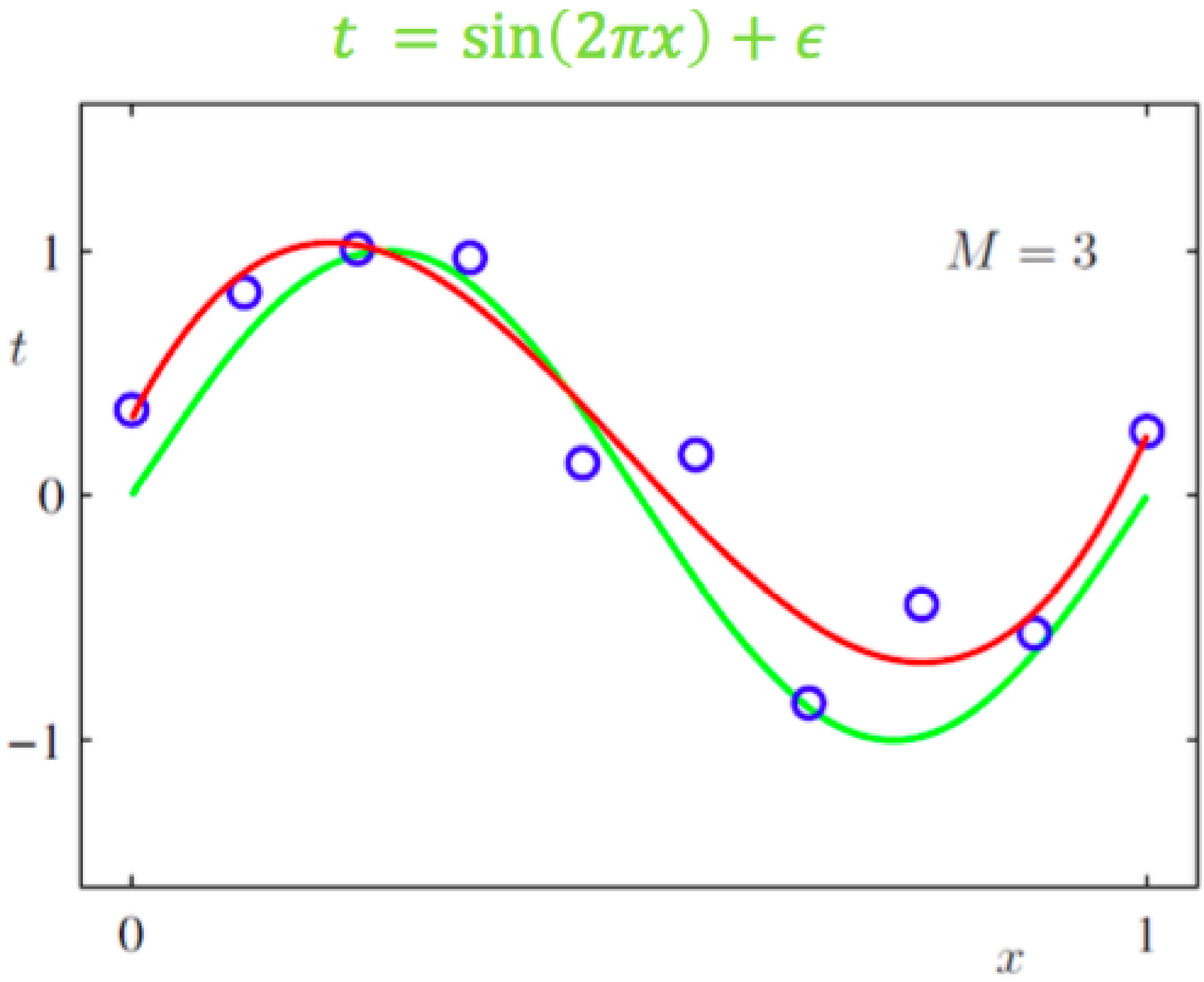
Example 3: regression using polynomial

$$t = \sin(2\pi x) + \epsilon$$



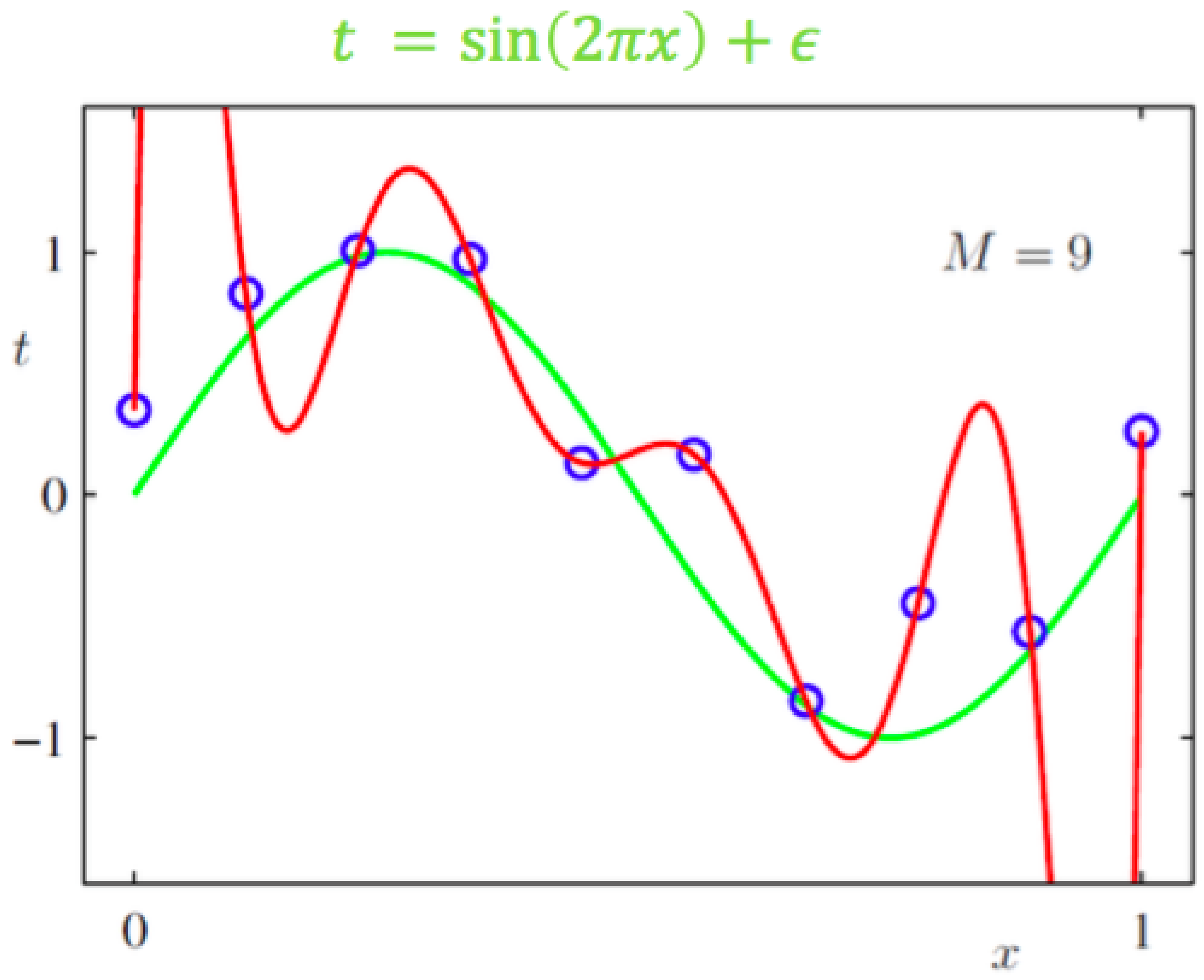
$y = w_1 x + w_0$

Example 3: regression using polynomial



$y = w_3x^3 + w_2x^2 + w_1x + w_0$

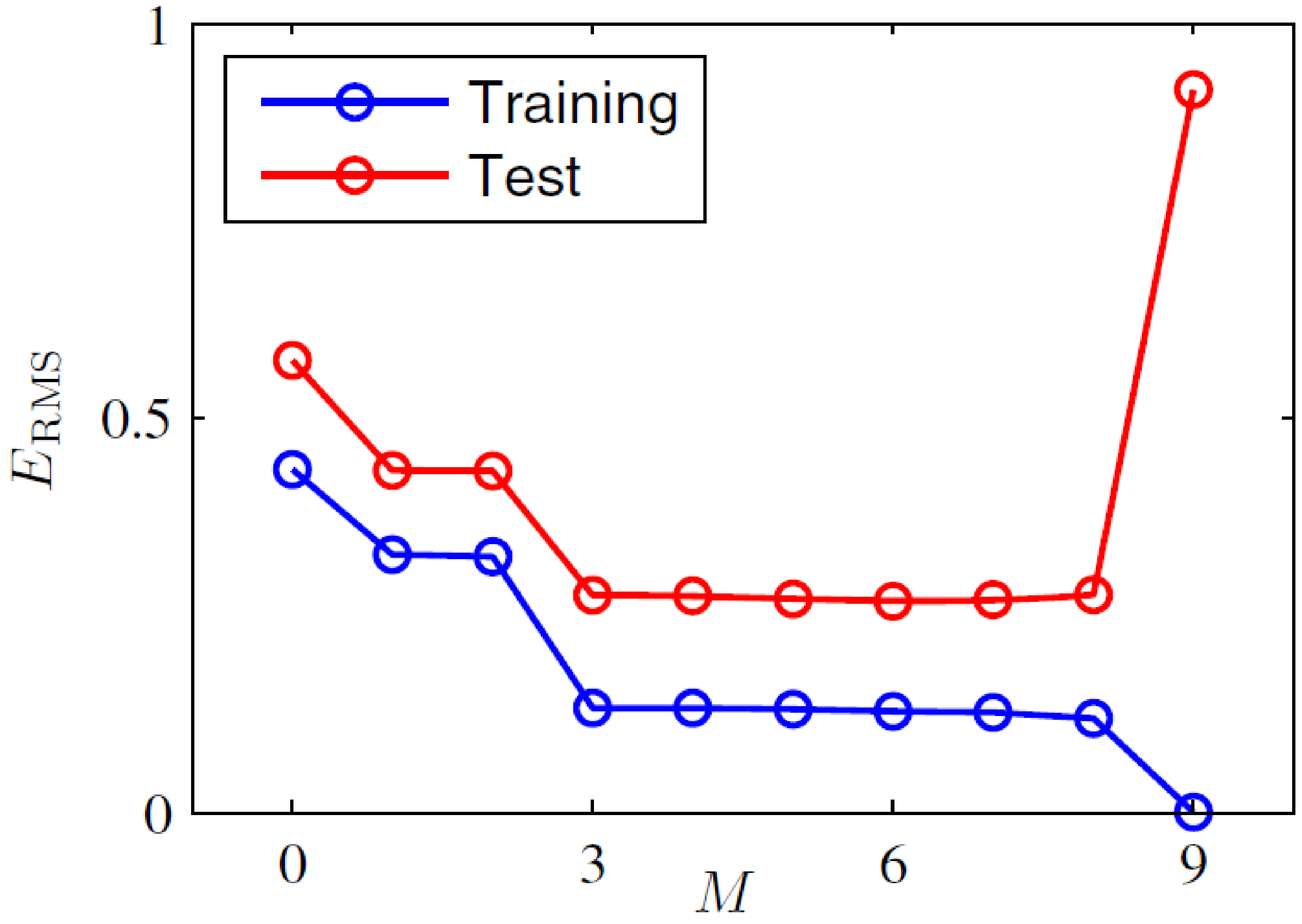
Example 3: regression using polynomial



$y = w_9x^9 + w_8x^8 + \ldots + w_1x + w_0$

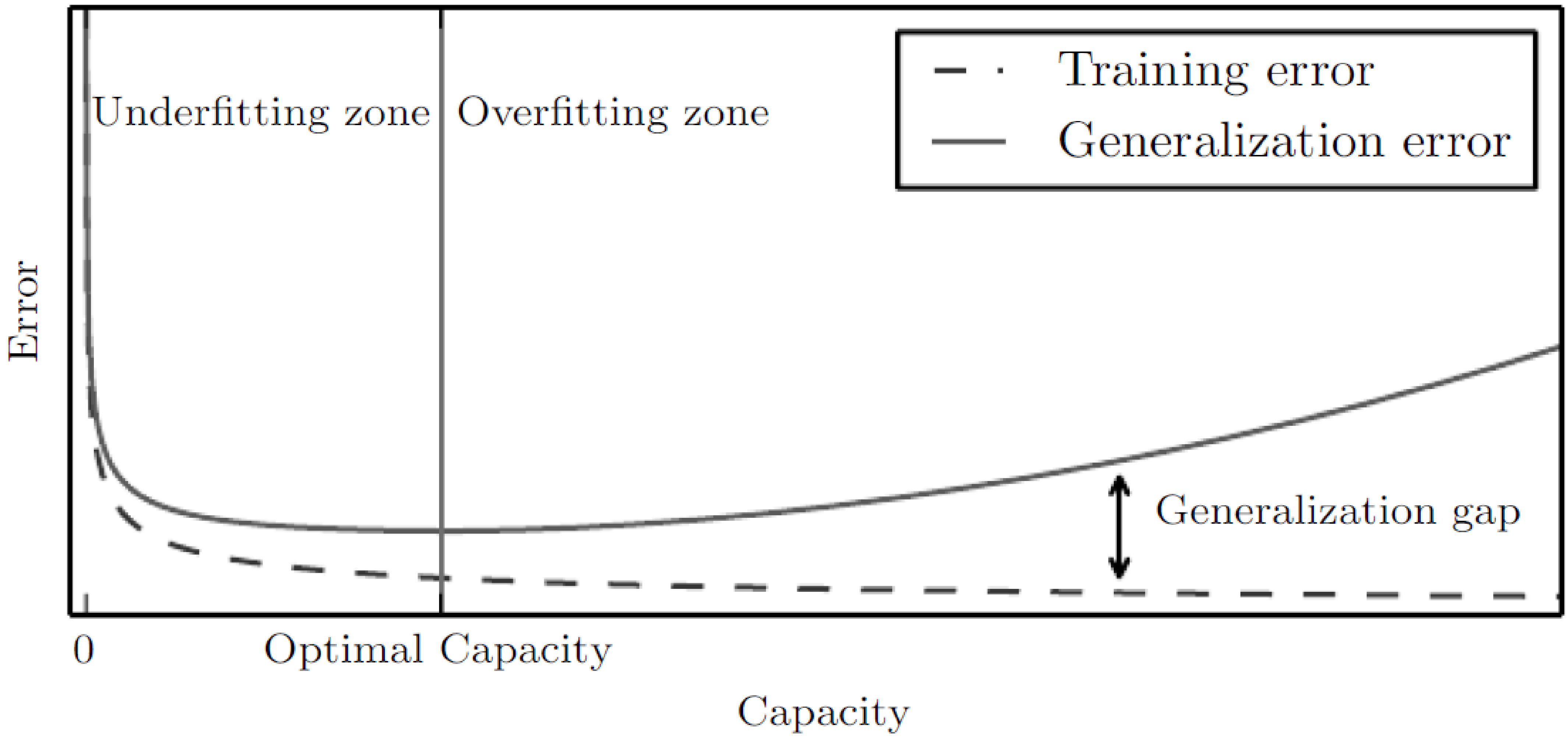
Overfits, why?

Example 3: regression using polynomial



RMS: root mean square, i.e., the square root of the mean square

General phenomenon



Prevent overfitting

- cause: training error and expected error are different
  - there may be noise in the training data
  - training data is of limited size, resulting in difference from the true distribution
  - larger the hypothesis class, easier to find a hypothesis that fits the difference between the training data and the true distribution
- prevent overfitting:
  - cleaner training data help!
  - more training data help!
  - throwing away unnecessary hypotheses helps! (Occam's Razor)

Avoiding overfitting in DT learning

- two general strategies to avoid overfitting
  - *1. early stopping*: stop if further splitting not justified by a statistical test
    - Quinlan’s original approach in ID3
  - *2. post-pruning*: grow a large tree, then prune back some nodes
    - more robust to myopia of greedy tree learning

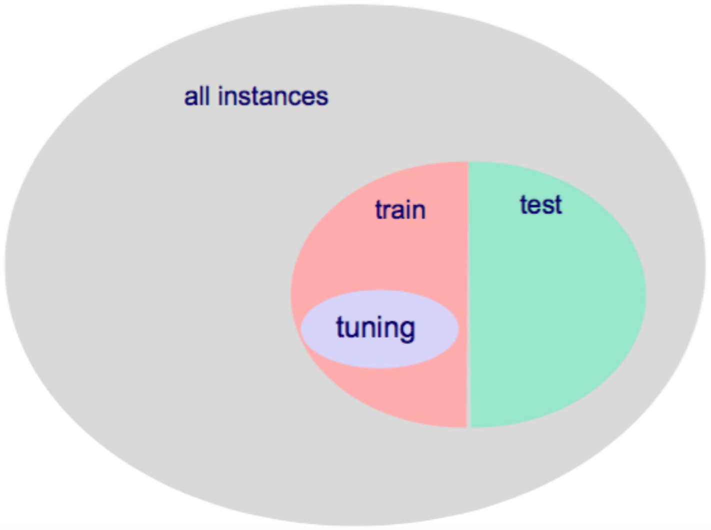
Stopping criteria

Stopping criteria

- We should form a leaf when
  - all of the given subset of instances are of the same class
  - we’ve exhausted all of the candidate splits
- Is there a reason to stop earlier, or to prune back the tree?

Pruning in C4.5

- split given data into training and *validation* (*tuning*) sets
- a *validation set* (a.k.a. *tuning set*) is a subset of the training set that is held aside
  - not used for primary training process (e.g. tree growing)
  - but used to select among models (e.g. trees pruned to varying degrees)



Pruning in C4.5

- split given data into training and *validation* (*tuning*) sets
- Grow a complete tree
- do until further pruning is harmful
  - evaluate impact on tuning-set accuracy of pruning each node
  - greedily remove the one that most improves tuning-set accuracy

Nearest-neighbor classification

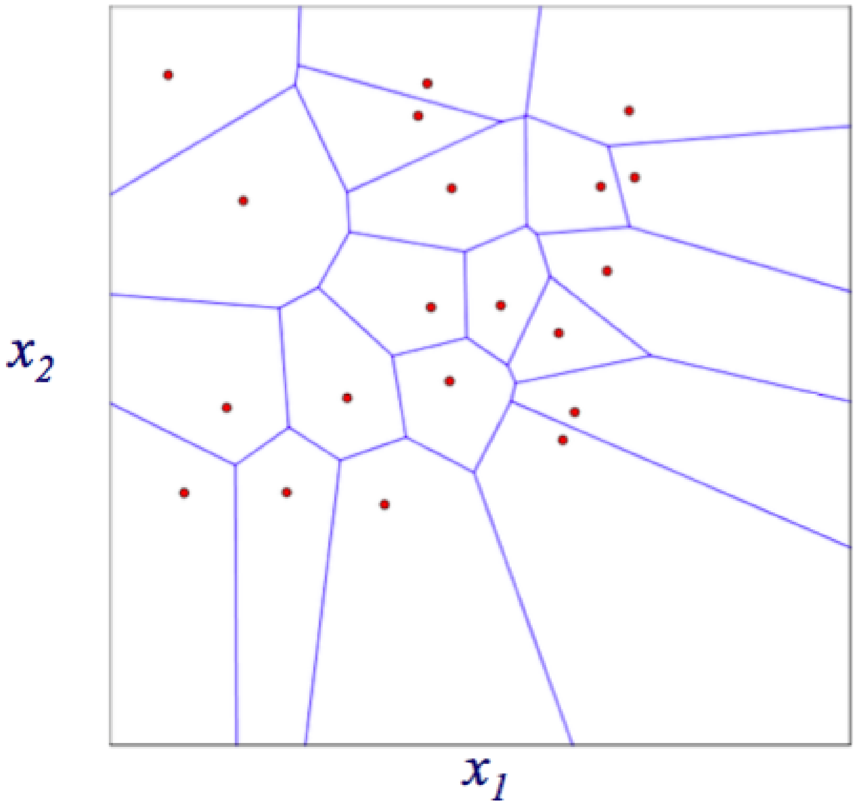
Nearest-neighbor classification

- learning stage
  - given a training set $(\mathbf{x}^{(1)}, y^{(1)}) \ldots (\mathbf{x}^{(m)}, y^{(m)})$, do nothing
    - (it's sometimes called a *lazy learner*)

- classification stage
  - **given**: an instance $x^{(q)}$ to classify
  - find the training-set instance $x^{(i)}$ that is most similar to $x^{(q)}$
  - return the class value $y^{(i)}$

The decision regions for nearest-neighbor classification

- Voronoi diagram: each polyhedron indicates the region of feature space that is in the nearest neighborhood of each training instance



k-nearest-neighbor classification

- classification task
  - **given**: an instance $x^{(q)}$ to classify
  - find the k training-set instances $(\mathbf{x}^{(1)}, y^{(1)})\ldots(x^{(k)}, y^{(k)})$ that are the most similar to $x^{(q)}$
  - return the class value

$$\hat{y} \leftarrow \underset{v \in \text{values}(Y)}{\text{argmax}} \sum_{i=1}^{k} \delta(v, y^{(i)}) \qquad \delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

  - (i.e. return the class that have the most instances)

How can we determine similarity/distance

- suppose all features are discrete
  - Hamming distance (or L^0 norm): count the number of features for which two instances differ

- Example: X = (Weekday, Happy?, Weather) Y = AttendLecture?
  - D : in the table
  - New instance: <Friday, No, Rain>
  - Distances = {2, 3, 1, 2}
  - For 1-nn, which instances should be selected?
  - For 2-nn, which instances should be selected?
  - For 3-nn, which instances should be selected?

v1	v2	v3	y
Wed	Yes	Rain	No
Wed	Yes	Sunny	Yes
Thu	No	Rain	Yes
Fri	Yes	Sunny	No

New datum

Fri	No	Rain	

How can we determine similarity/distance

- Example: X = (Weekday, Happy?, Weather) Y = AttendLecture?
  - New instance: <Friday, No, Rain>
  - For 3-nn, selected instances: {(<Wed, Yes, Rain>, No), (<Thu, No, Rain>, Yes), (<Fri, Yes, Sunny>, No)}
- Classification:

$$\hat{y} \leftarrow \underset{v \in \text{values}(Y)}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, y^{(i)})$$

  - v = Yes. $\sum_{i=1}^{k} \delta(v, y^{(i)}) = 0 + 1 + 0 = 1$
  - v = No. $\sum_{i=1}^{k} \delta(v, y^{(i)}) = 1 + 0 + 1 = 0$

So, which class this new instance should be in?

How can we determine similarity/distance

- suppose all features are continuous
  - Euclidean distance:

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sqrt{\sum_f \left(x_f^{(i)} - x_f^{(j)}\right)^2}$$

where $x_f^{(i)}$ represents the f -th feature of $x^{(i)}$

  - Manhattan distance:

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_f \left|x_f^{(i)} - x_f^{(j)}\right|$$

Recall the difference and similarity with L^p norm

How can we determine similarity/distance

- Example: X = (Height, Weight, RunningSpeed) Y = SoccerPlayer?
  - D: in the table
  - New instance: <185, 91, 13.0>
  - Suppose that Euclidean distance is used.
  - Is this person a soccer player?

v1	v2	v3	y
182	87	11.3	No
189	92	12.3	Yes
178	79	10.6	Yes
183	90	12.7	No

New datum

185	**91**	**13.0**	

How can we determine similarity/distance

- if we have a mix of discrete/continuous features:

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_f \begin{cases} \left| x_f^{(i)} - x_f^{(j)} \right| \text{ if } f \text{ is continuous} \\ 1 - \delta\left(x_f^{(i)}, x_f^{(i)}\right) \text{if } f \text{ is discrete} \end{cases}$$

- typically want to apply to continuous features some type of normalization (values range 0 to 1) or standardization (values distributed according to standard normal)
- many other possible distance functions we could use ...

Standardizing numeric features

- given the training set D, determine the mean and stddev for feature x_i

$$\mu_i = \frac{1}{|D|} \sum_{d=1}^{|D|} x_i^{(d)} \qquad \sigma_i = \sqrt{\frac{1}{|D|} \sum_{d=1}^{|D|} \left(x_i^{(d)} - \mu_i \right)^2}$$

- standardize each value of feature x_i as follows

$$\hat{x}_i^{(d)} = \frac{x_i^{(d)} - \mu_i}{\sigma_i}$$

- do the same for test instances, using the same μ and σ derived from the *training* data