Foundations of Machine Learning

Module 2: Linear Regression and Decision Tree

Part D: Overfitting

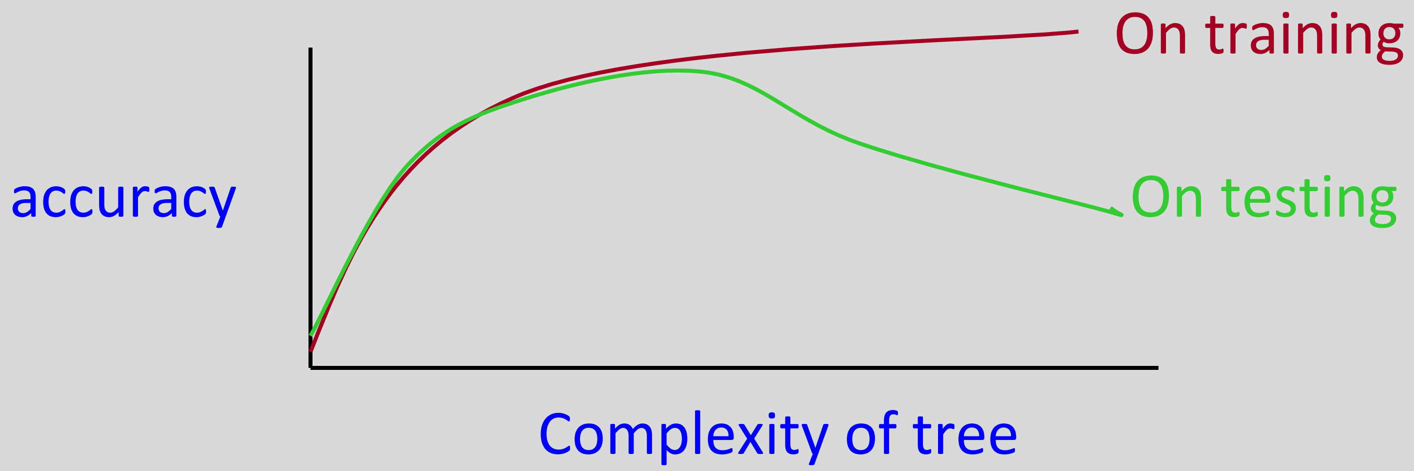
Sudeshna Sarkar
IIT Kharagpur

Overfitting

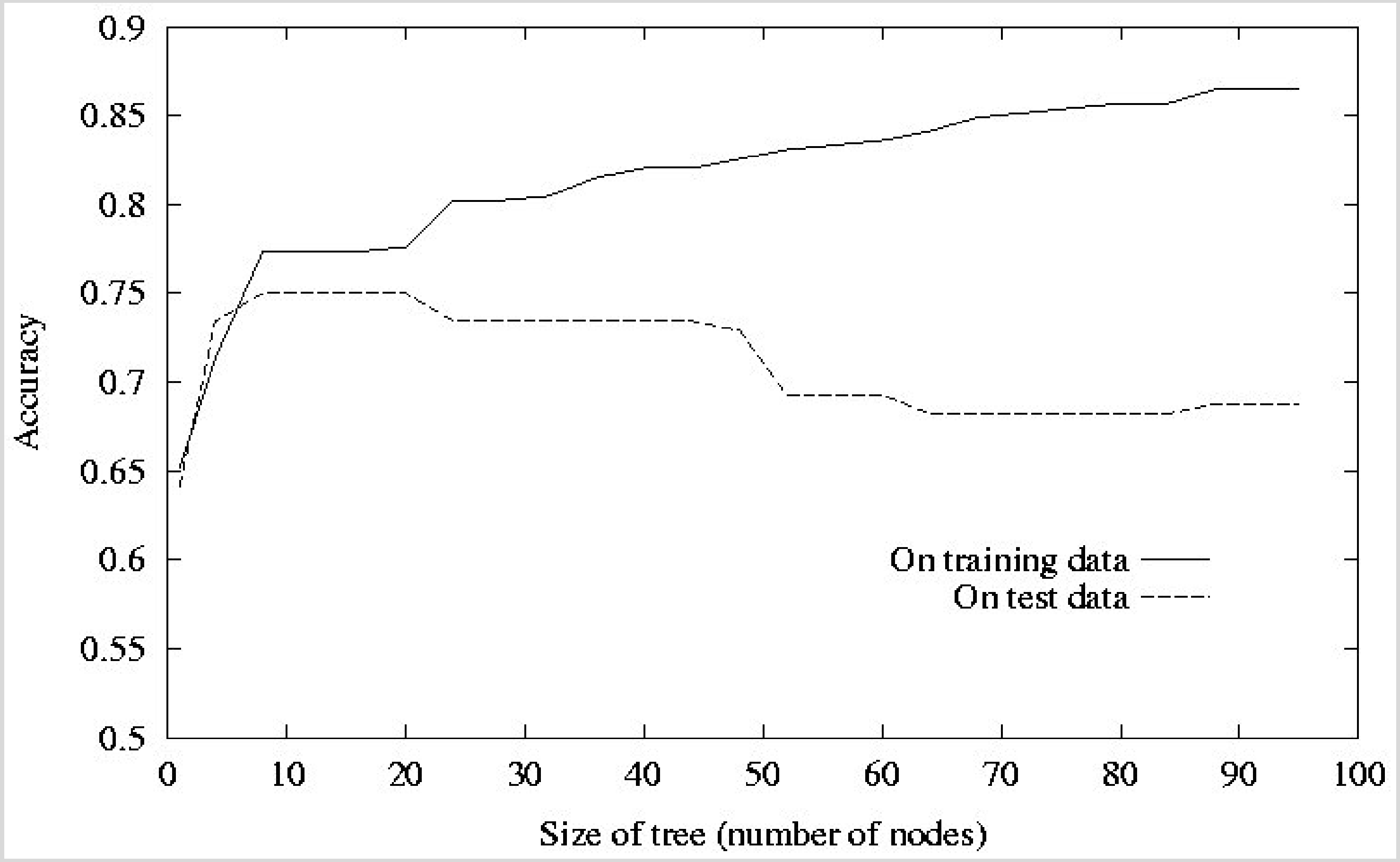
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data
  - May be based on insufficient data
- A hypothesis *h* is said to overfit the training data if there is another hypothesis, h', such that *h* has smaller error than *h*' on the training data but *h* has larger error on the test data than *h'*.

Overfitting

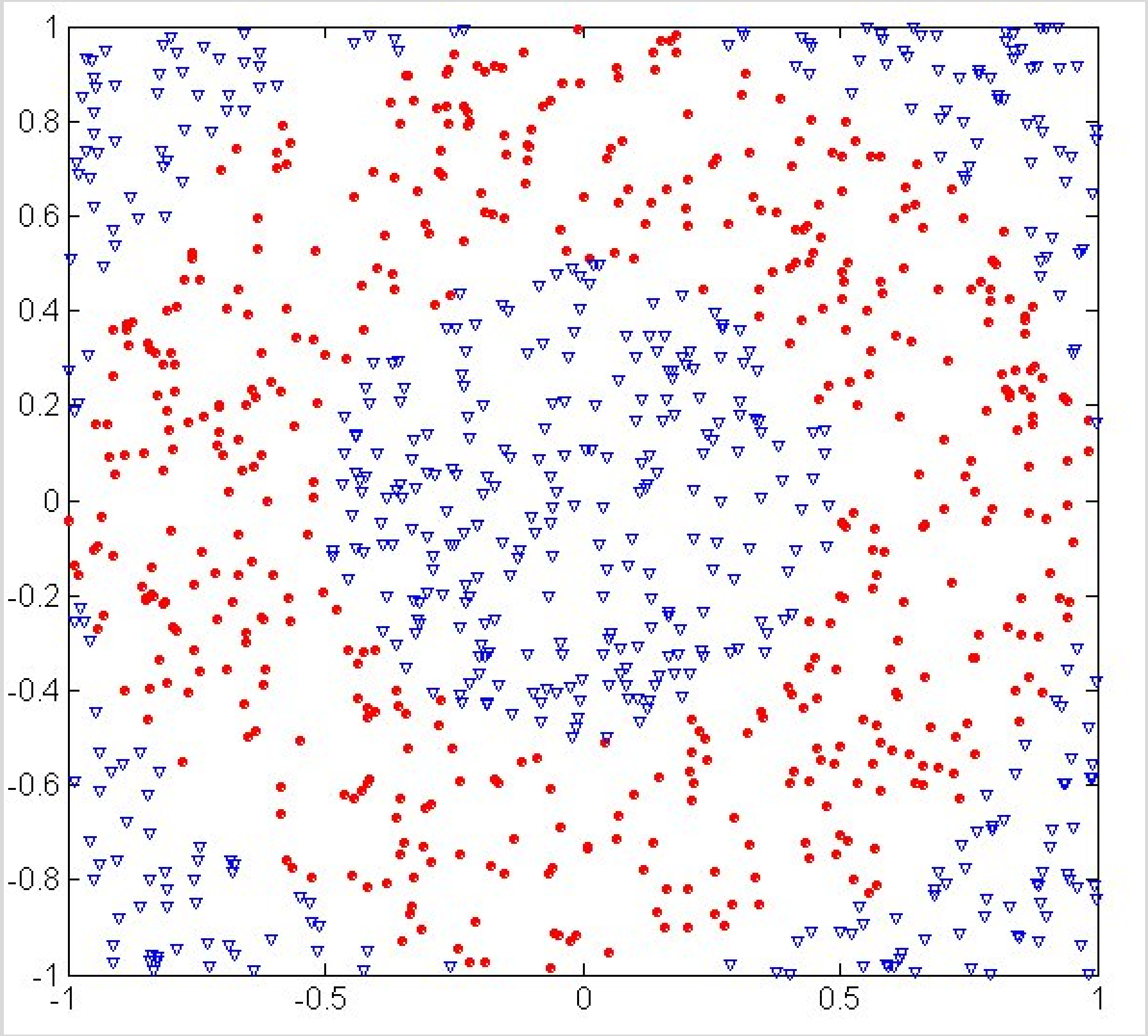
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data
  - May be based on insufficient data
- A hypothesis *h* is said to overfit the training data if there is another hypothesis, h', such that *h* has smaller error than *h*' on the training data but *h* has larger error on the test data than *h'*.



Overfitting



Underfitting and Overfitting (Example)

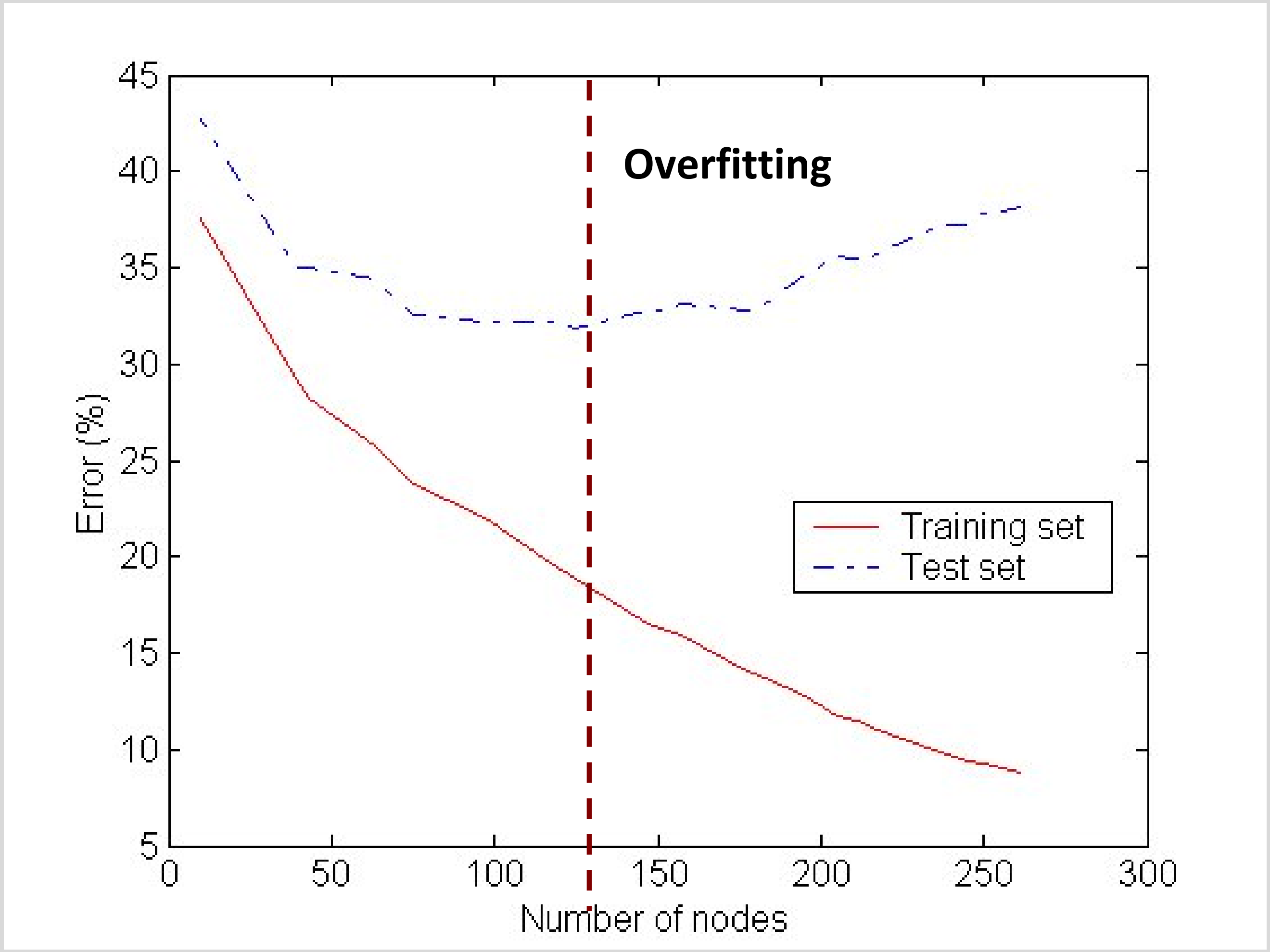


500 circular and 500 triangular data points.

Circular points:
0.5 ◉ sqrt($x_1^2+x_2^2$) ◉ 1

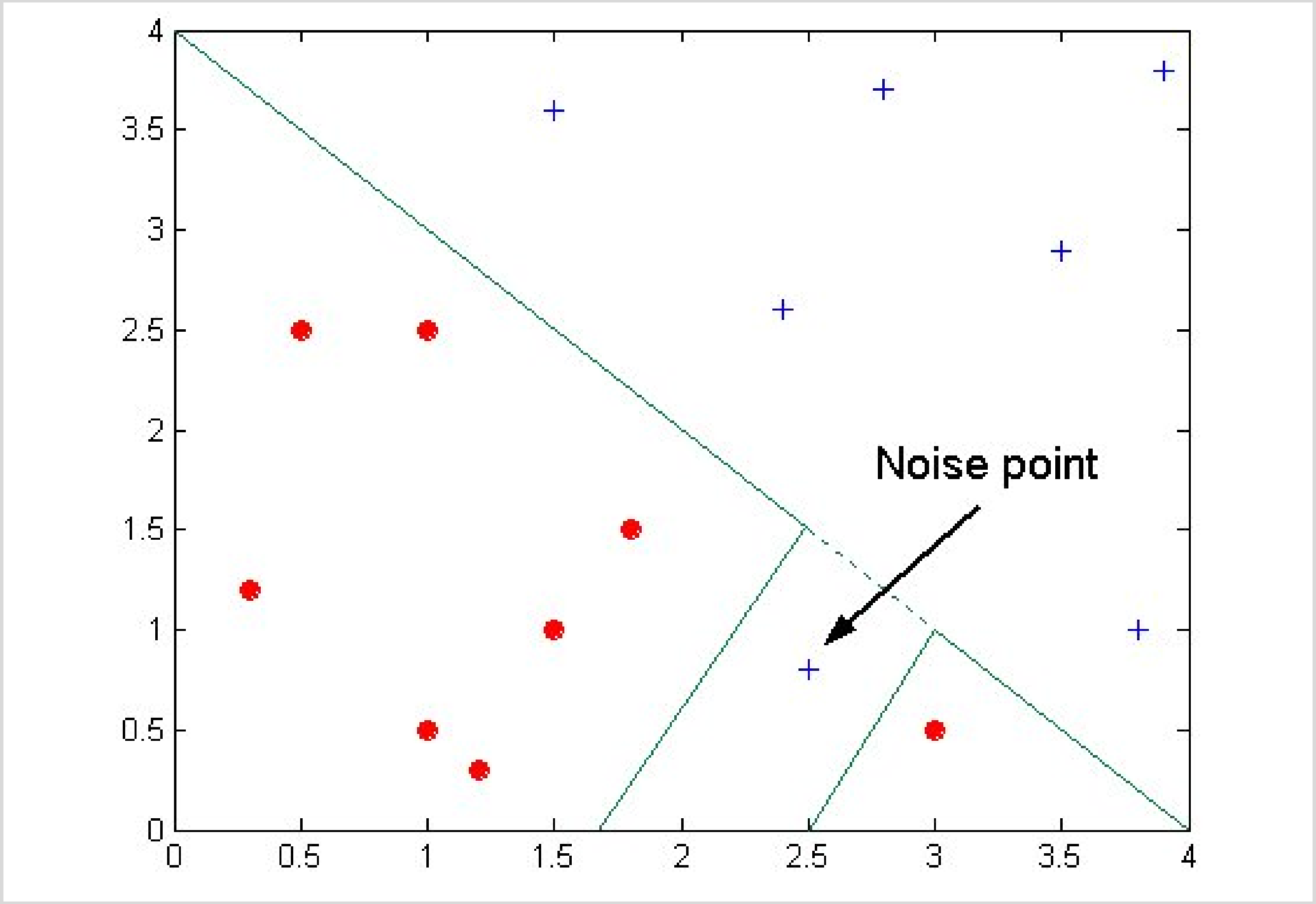
Triangular points:
sqrt($x_1^2+x_2^2$) > 0.5 or
sqrt($x_1^2+x_2^2$) < 1

Underfitting and Overfitting



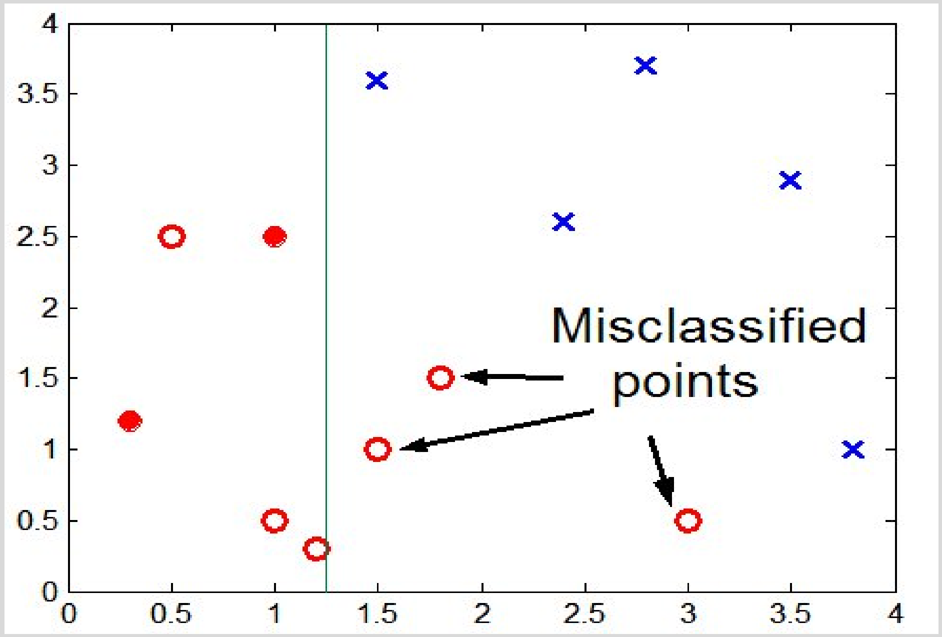
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points makes it difficult to predict correctly the class labels of that region

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

Avoid Overfitting

- How can we avoid overfitting a decision tree?
  - Prepruning: Stop growing when data split not statistically significant
  - Postpruning: Grow full tree then remove nodes

- Methods for evaluating subtrees to prune:
  - Minimum description length (MDL):

  Minimize: size(tree) + size(misclassifications(tree))
  - Cross-validation

Pre-Pruning (Early Stopping)

- Evaluate splits before installing them:
  - Don’t install splits that don’t look worthwhile
  - when no worthwhile splits to install, done

Pre-Pruning (Early Stopping)

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using ♍^{2} test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Reduced-error Pruning

- A post-pruning, cross validation approach
  - Partition training data into "grow" set and "validation" set.
  - Build a complete tree for the "grow" data
  - Until accuracy on validation set decreases, do:

    For each non-leaf node in the tree

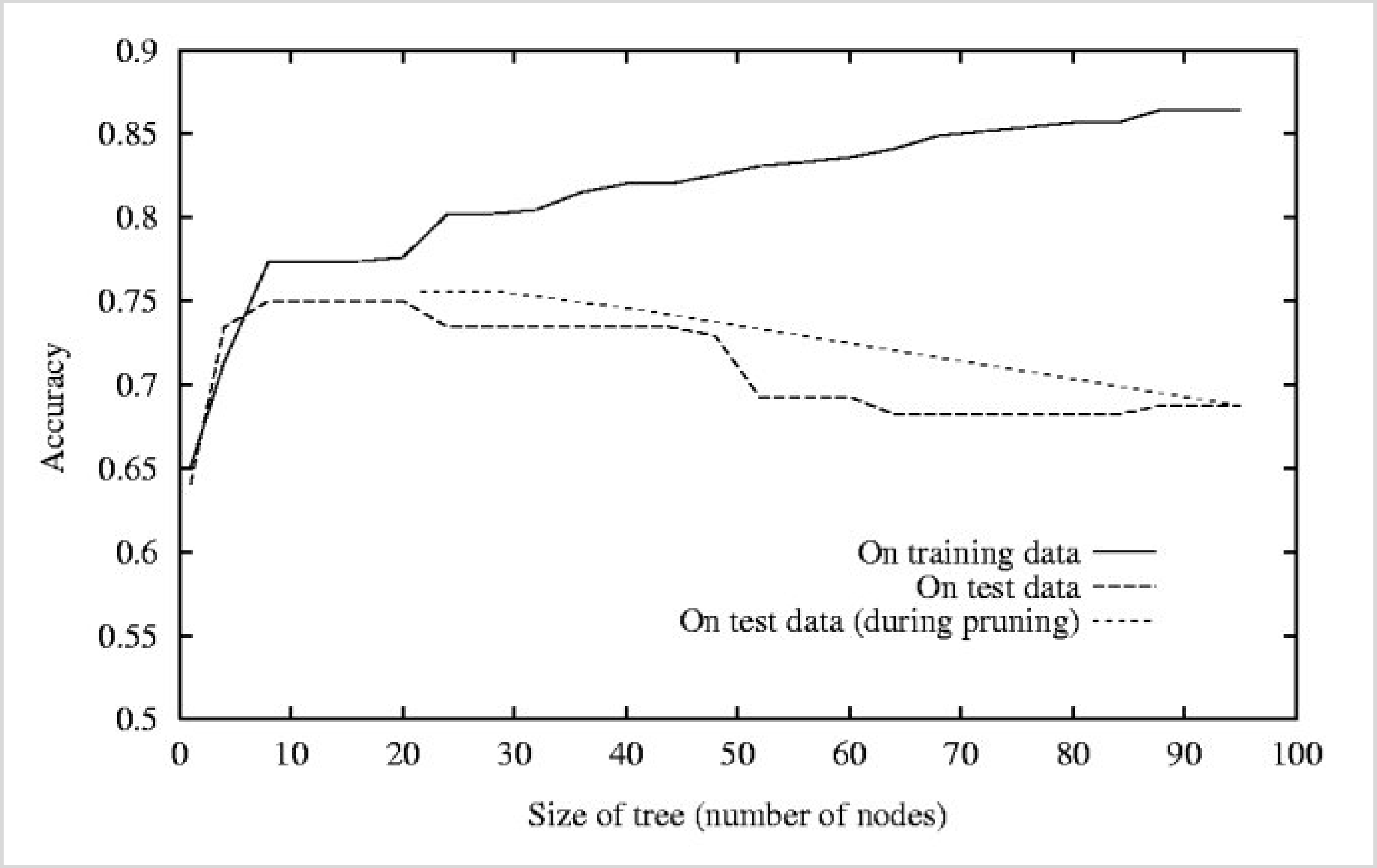
    Temporarily prune the tree below; replace it by majority vote

    Test the accuracy of the hypothesis on the validation set

    Permanently prune the node with the greatest increase in accuracy on the validation test.
- Problem: Uses less data to construct the tree
- Sometimes done at the rules level

General Strategy: Overfit and Simplify

Reduced Error Pruning



Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than *C* or *f*
- Underfitting: H less complex than *C* or *f*

Triple Trade-Off

- There is a trade-off between three factors:
  - Complexity of H, *c* (H),
  - Training set size, *N,*
  - Generalization error, *E* on new data

overfitting

- As *N* *increases*, *E* decreases
- As *c* (H) *increases*, first *E* *decreases* and then *E* *increases*
- As *c* (H) *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)

Notes on Overfitting

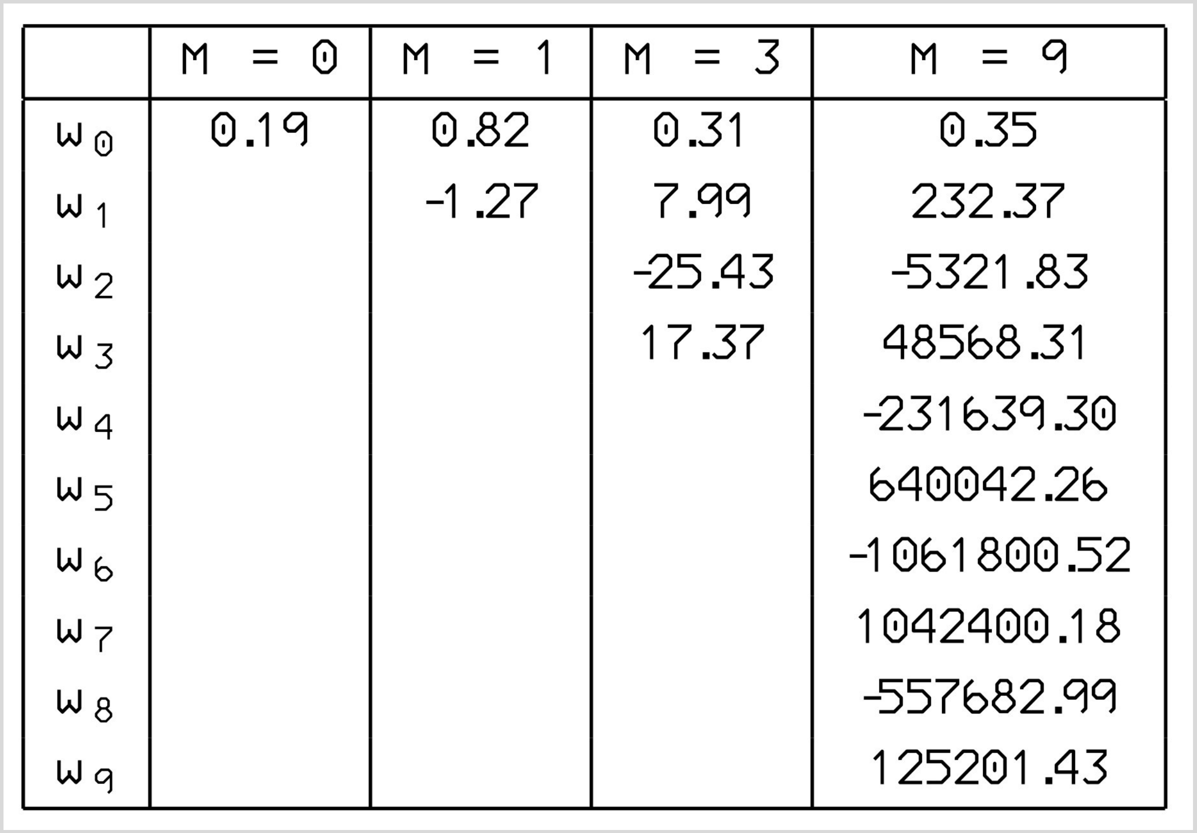
- **overfitting** happens when a model is capturing idiosyncrasies of the data rather than generalities.
  - Often caused by too many parameters relative to the amount of training data.
  - E.g. an order-*N* polynomial can intersect any *N+1* data points

Dealing with Overfitting

- Use more data
- Use a tuning set
- **Regularization**
- Be a Bayesian

Regularization

- In a linear regression model overfitting is characterized by large weights.



	M = 0	M = 1	M = 3	M = 9
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Penalize large weights in Linear Regression

- Introduce a penalty term in the loss function.

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}\{t_n - y(x_n, \vec{w})\}^2$$

Regularized Regression

1. (L2-Regularization or Ridge Regression)

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \frac{\lambda}{2}\|\vec{w}\|^2$$

1. L1-Regularization

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \lambda|\vec{w}|_1$$