Foundations of Machine Learning

Module 2: Linear Regression and Decision Tree

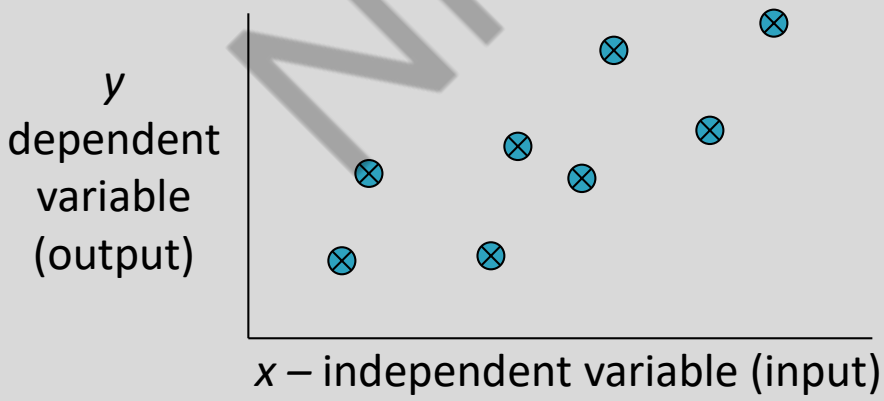
Part A: Linear Regression

Sudeshna Sarkar

IIT Kharagpur

Regression

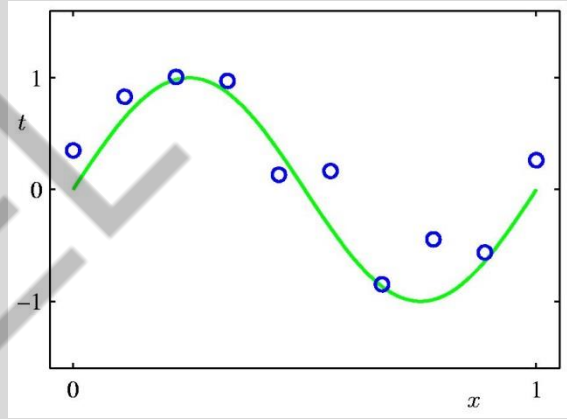
- In regression the output is continuous
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points

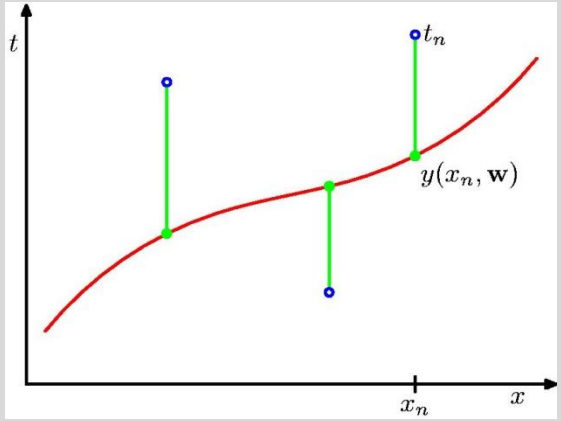


A Simple Example: Fitting a Polynomial

- The green curve is the true function (which is not a polynomial)

- We may use a loss function that measures the squared error in the prediction of y(x) from x.

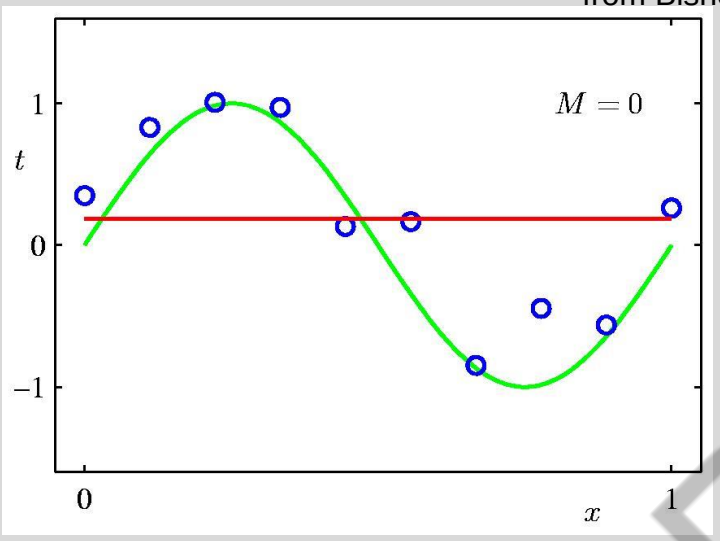


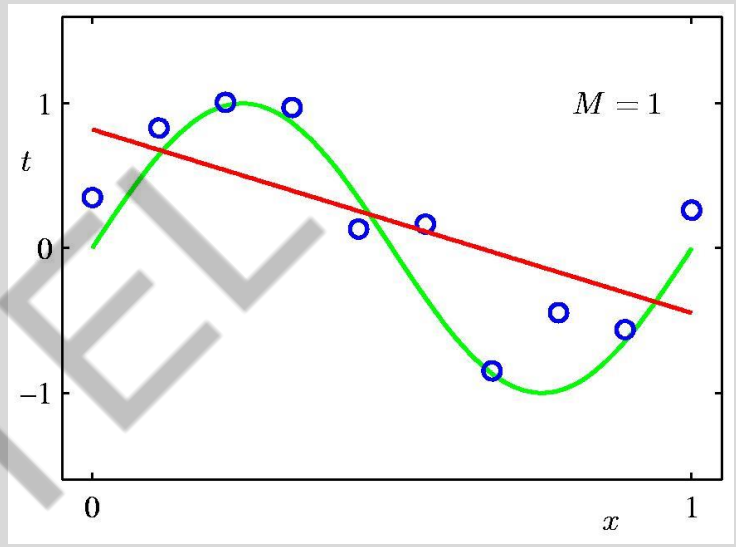


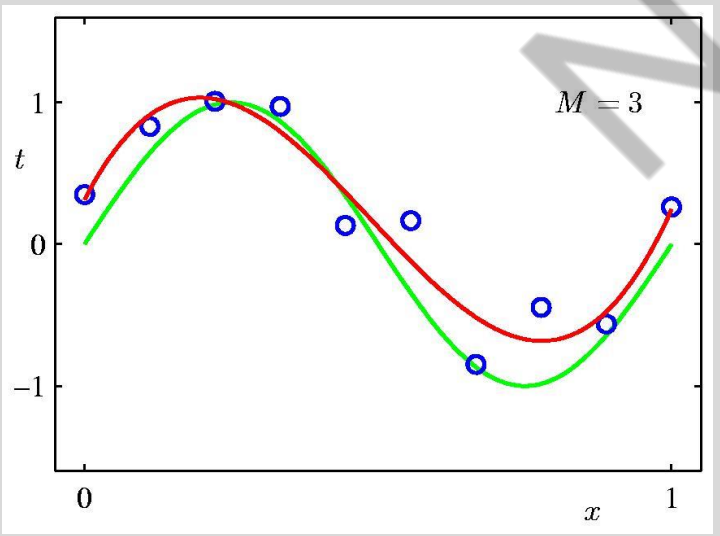
from Bishop's book on Machine Learning

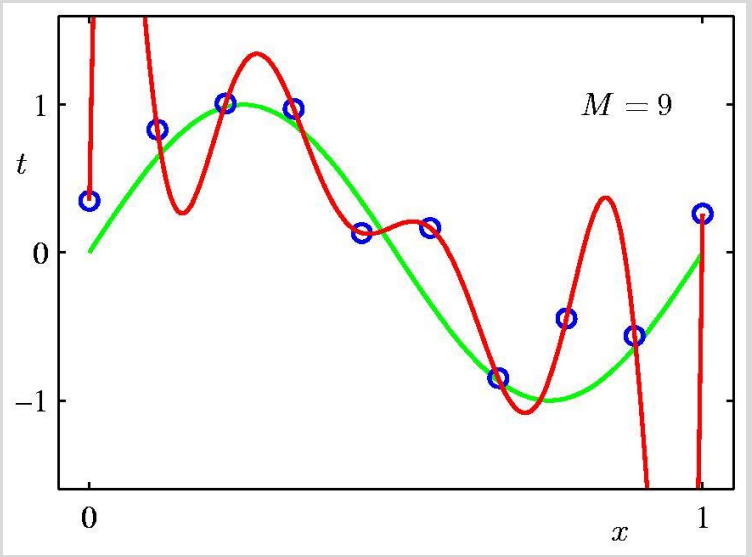
Some fits to the data: which is best?

from Bishop

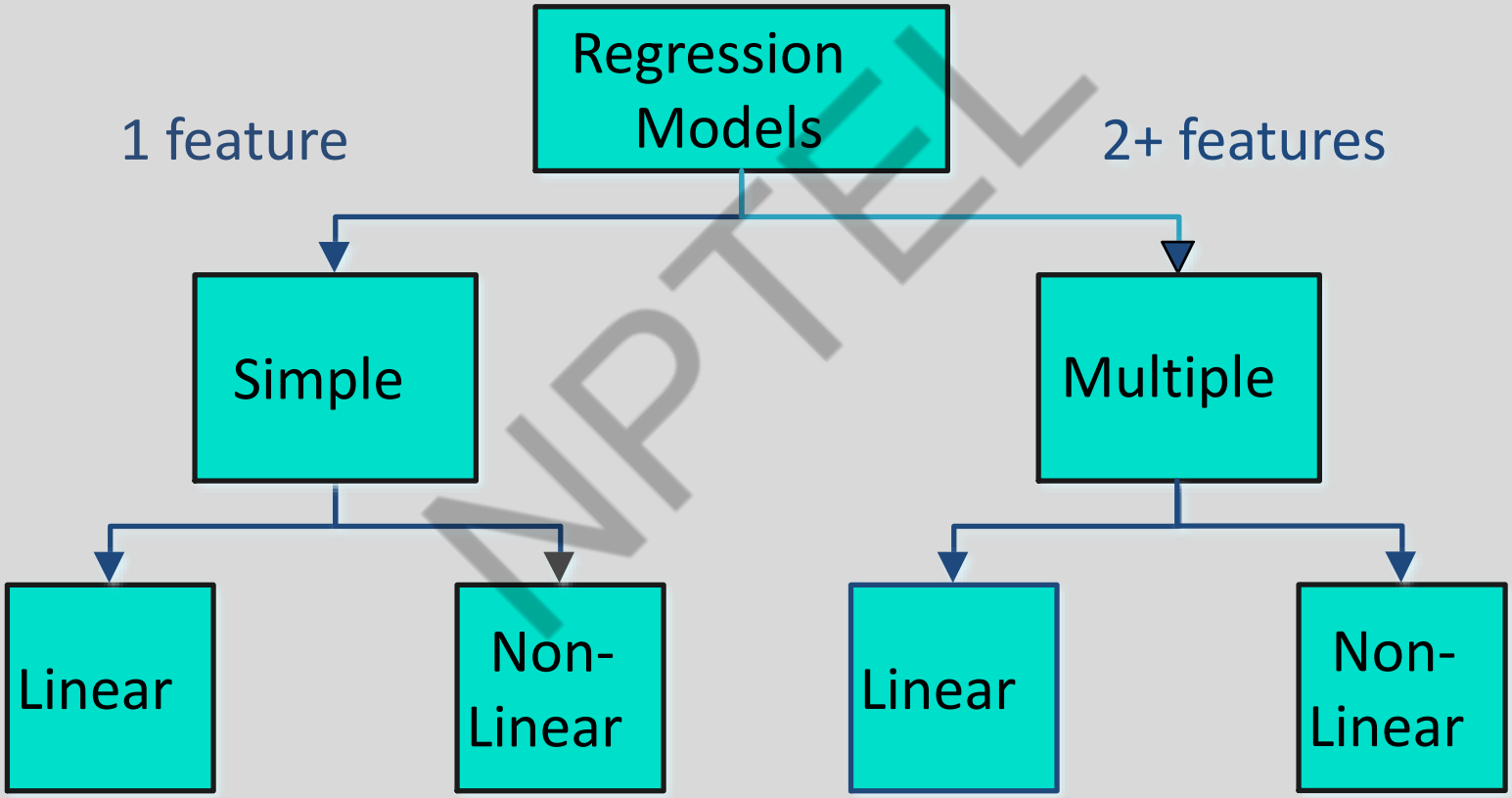








Types of Regression Models



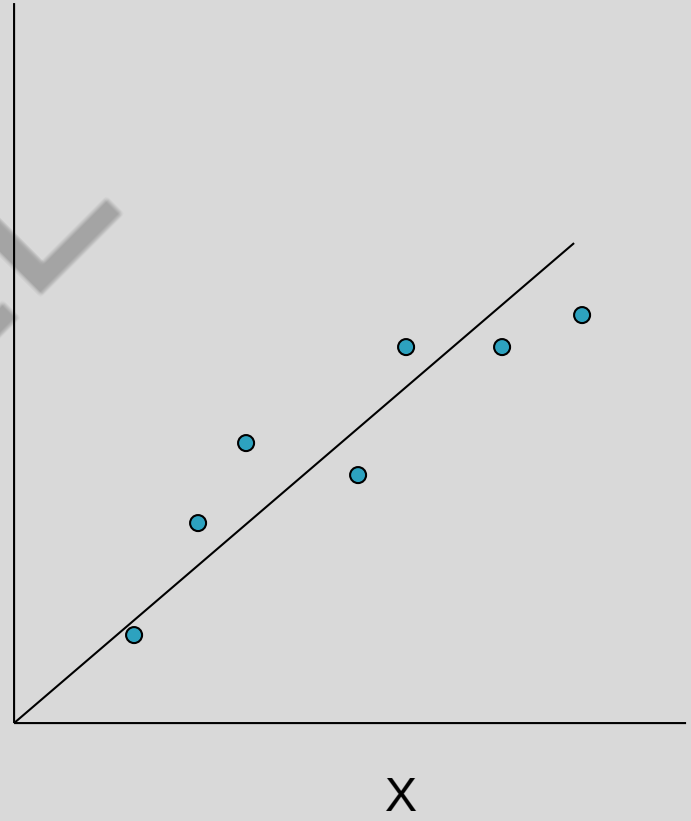
Linear regression

- Given an input x compute an output y
- For example:

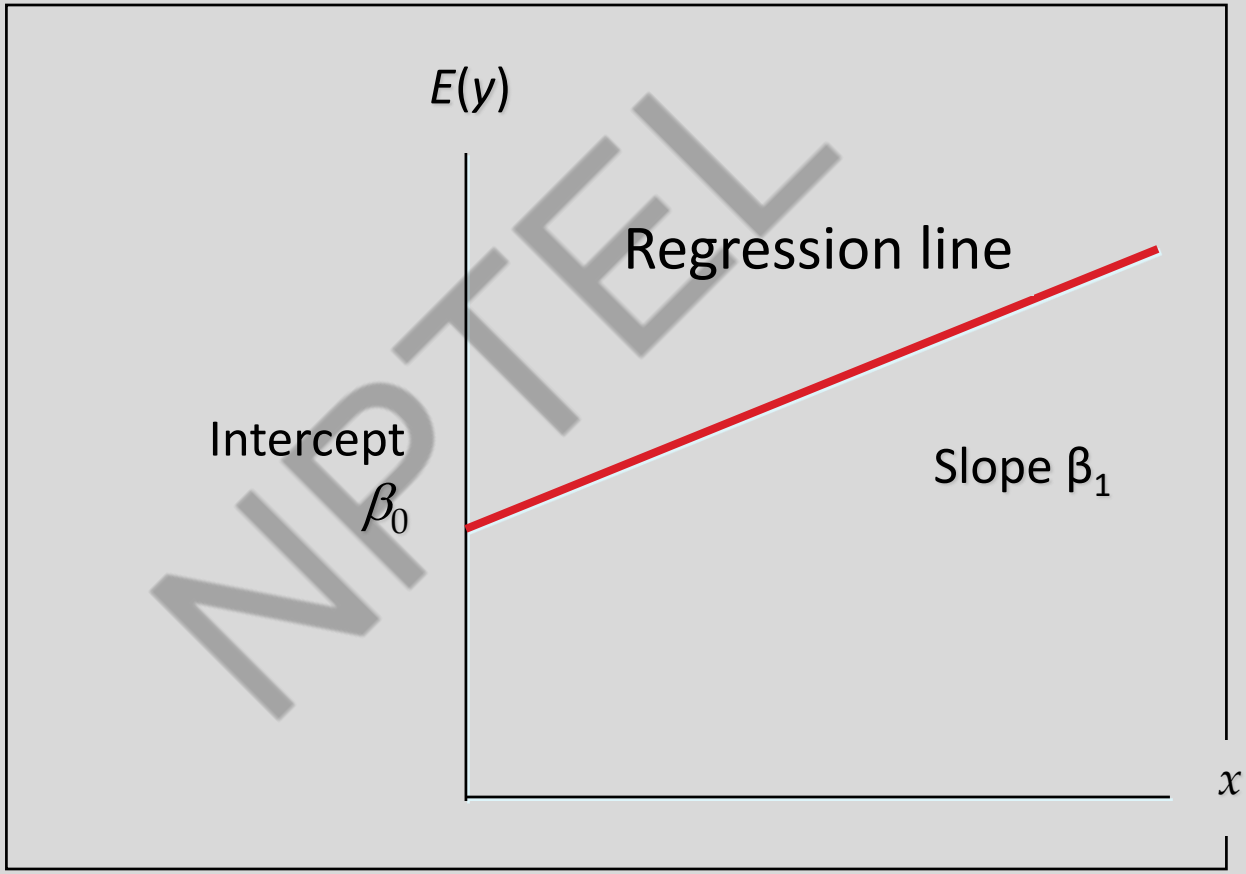
  - Predict height from age

  - Predict house price from house area

  - Predict distance from wall from sensors

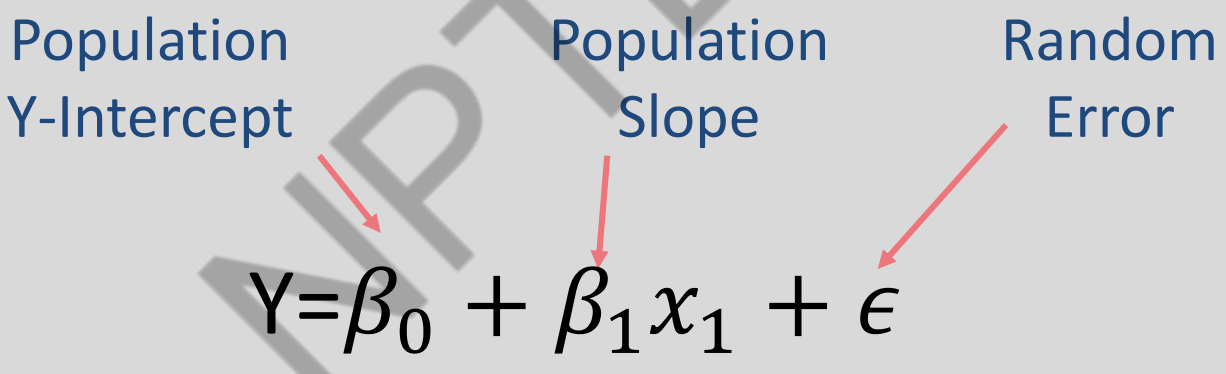


Simple Linear Regression Equation



Linear Regression Model

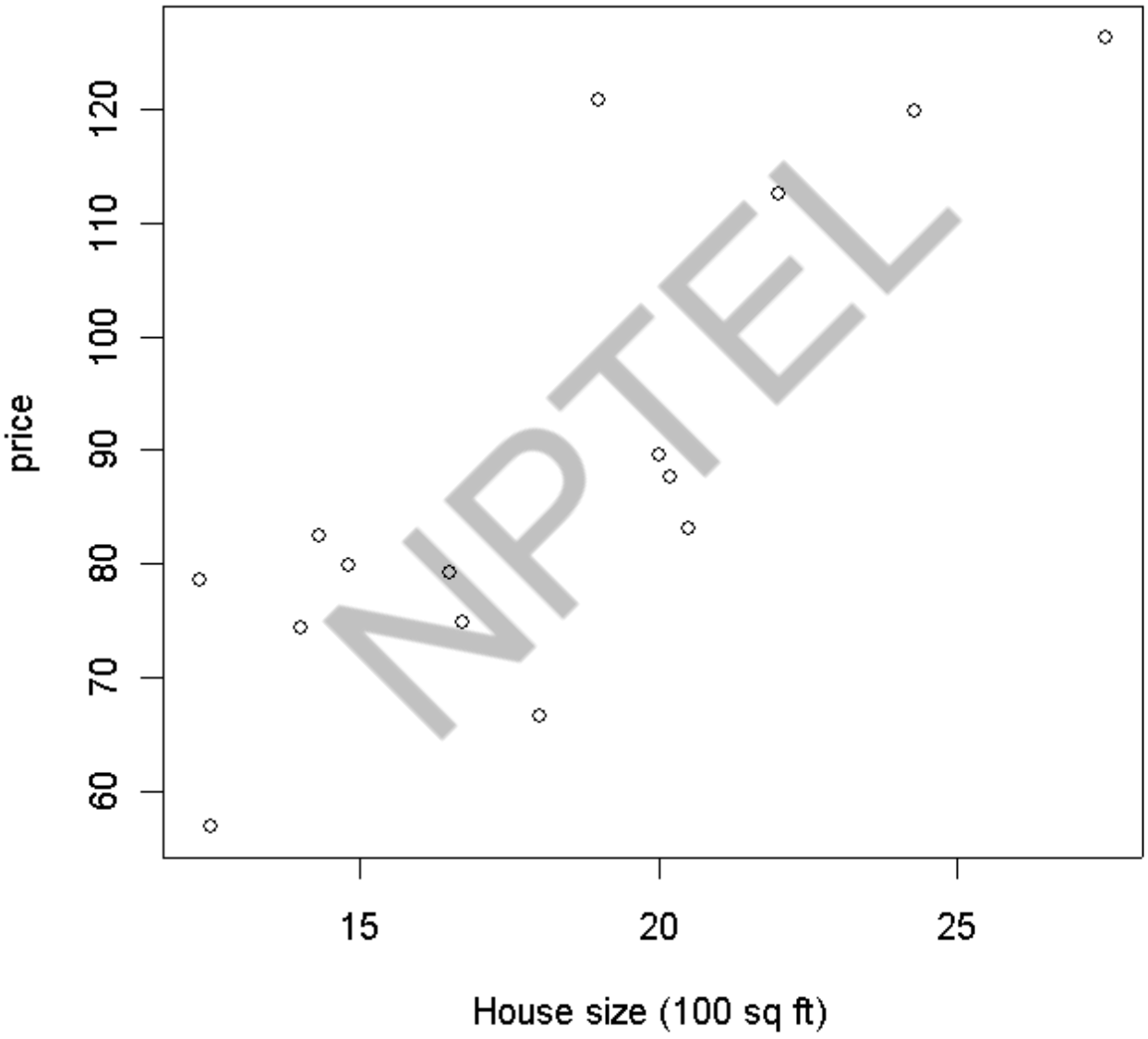
- Relationship Between Variables Is a Linear Function



House Number	*Y:* Actual Selling Price	*X*: House Size (100s ft2)
1	89.5	20.0
2	79.9	14.8
3	83.1	20.5
4	56.9	12.5
5	66.6	18.0
6	82.5	14.3
7	126.3	27.5
8	79.3	16.5
9	119.9	24.3
10	87.6	20.2
11	112.6	22.0
12	120.8	.019
13	78.5	12.3
14	74.3	14.0
15	74.8	16.7
Averages	88.84	18.17

Sample 15 houses from the region.

House price vs size



Linear Regression – Multiple Variables

$$Y_i = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_p X_p + \epsilon$$

- β_0 is the intercept (i.e. the average value for Y if all the X's are zero), β_j is the slope for the *j*th variable X_j

Regression Model

- Our model assumes that

$E(Y \mid X = x) = \beta_0 + \beta_1 x$ (the "population line")

Population line

$$Y_i = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_p X_p + \epsilon$$

Least Squares line

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_1 + \hat{b}_2 X_2 + \cdots + \hat{b}_p X_p$$

We use $\widehat{\beta_0}$ through $\widehat{\beta_p}$ as guesses for β_0 through β_p and $\widehat{Y_i}$ as a guess for Y_i. The guesses will not be perfect.

Assumption

- The data may not form a perfect line.
- When we actually take a measurement (i.e., observe the data), we observe:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i,$$

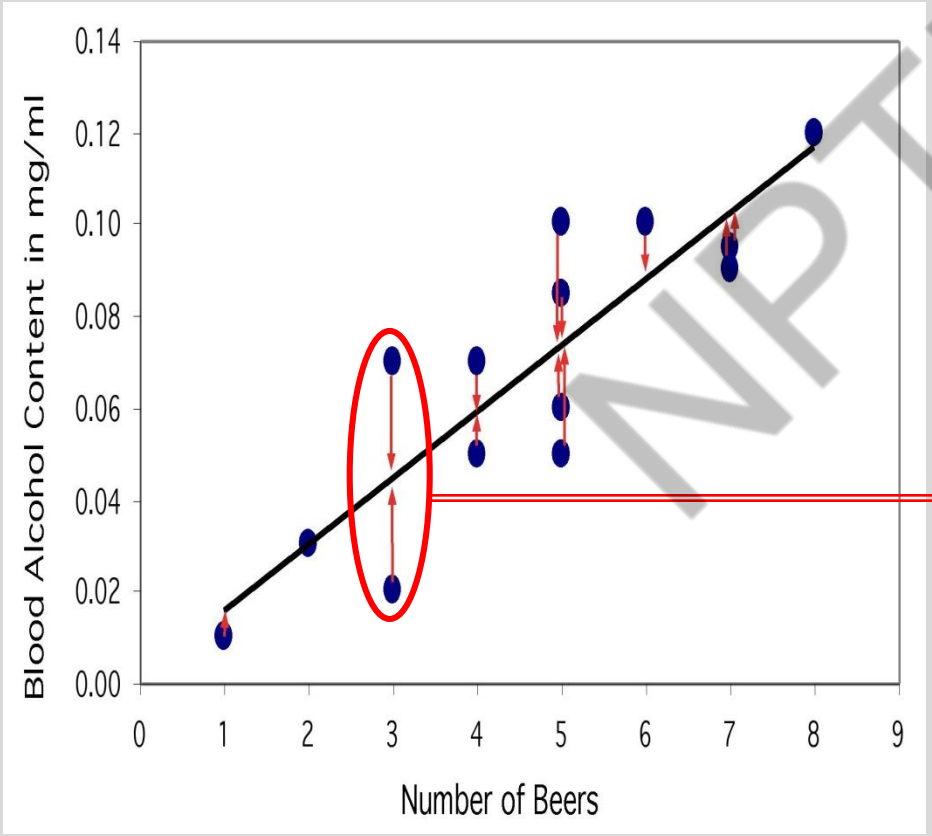
where ε_i is the random error associated with the *i*th observation.

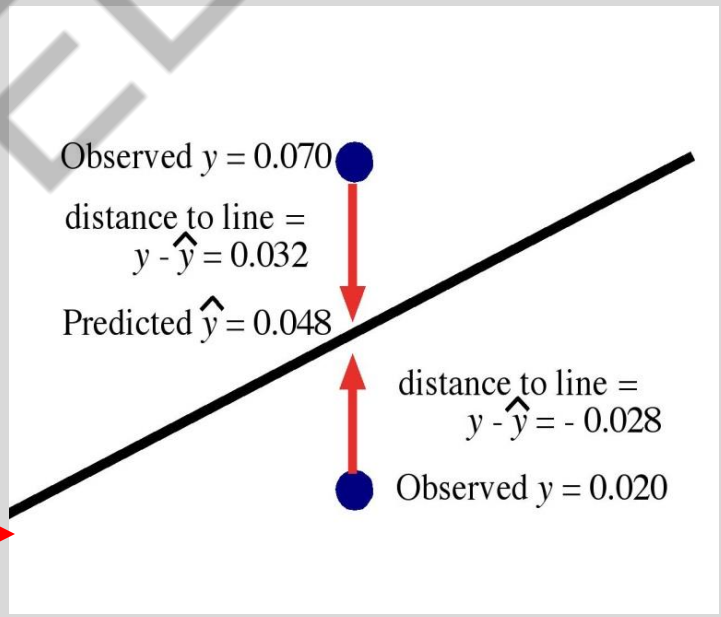
Assumptions about the Error

- $E(\varepsilon_i) = 0$ for $i = 1, 2, \ldots, n$.
- $\sigma(\varepsilon_i) = \sigma_\varepsilon$ where σ_ε is unknown.
- The errors are independent.
- The ε_i are normally distributed (with mean 0 and standard deviation σ_ε).

The regression line

The least-squares regression line is the unique line such that the sum of the squared vertical (y) distances between the data points and the line is the smallest possible.





Criterion for choosing what line to draw: method of least squares

- The method of least squares chooses the line ($\widehat{\beta_0}$ and $\widehat{\beta_1}$) that makes the sum of squares of the residuals $\sum \varepsilon_i^{\,2}$ as small as possible
- Minimizes

$$\sum_{i=1}^{n} [y_i - (b_0 + b_1 x_i)]^2$$

for the given observations (x_i, y_i)

How do we "learn" parameters

- For the 2-*d* problem

$$Y = b_0 + b_1 X$$

- To find the values for the coefficients which minimize the objective function we take the partial derivates of the objective function (SSE) with respect to the coefficients. Set these to 0, and solve.

$$b_1 = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2}$$

$$b_0 = \frac{\sum y - b_1 \sum x}{n}$$

Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

$$h(x) = \sum_{i=0}^{n} \beta_i x_i$$

- There is a closed form which requires matrix inversion, etc.
- There are iterative techniques to find weights
  - delta rule (also called LMS method) which will update towards the objective of minimizing the SSE.

Linear Regression

$$h(x) = \sum_{i=0}^{n} \beta_i x_i$$

To learn the parameters θ (β_i) ?

- Make h(**x**) close to *y*, for the available *training examples*.
- Define a cost function $J(\theta)$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x)^{(i)} - (y)^{(i)})^2$$

- Find θ that minimizes J(θ).

LMS Algorithm

- Start a search algorithm (e.g. gradient descent algorithm,) with initial guess of θ.
- Repeatedly update θ to make J(θ) smaller, until it converges to minima.

$$\beta_j = \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\theta)$$

- *J* is a convex quadratic function, so has a single global minima. gradient descent eventually converges at the global minima.
- At each iteration this algorithm takes a step in the direction of steepest descent(-ve direction of gradient).

LMS Update Rule

- If you have only one training example (x, y)

$$\frac{\partial}{\partial\theta}J(\theta) = \frac{\partial}{\partial\theta_j}\frac{1}{2}(h(x)-y)^2$$

$$= 2.\frac{1}{2}(h(x)-y)\frac{\partial}{\partial\theta_j}(h(x)-y)$$

$$= (h(x)-y).\frac{\partial}{\partial\theta_j}(\sum_{i=0}^{n}\theta_i x_i - y)$$

$$= (h(x)-y)x_j$$

- For a single training example, this gives the update rule:

$$\beta_j = \beta_j + \alpha(y^{(i)} - h(x^i))x_j{}^{(i)}$$

m training examples

Repeat until convergence {

$$\theta_j \underset{\theta}{:=} \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h\left(x^{(i)}\right)\right) x_j{}^{(i)}$$

}

Batch Gradient Descent: looks at every example on each step.

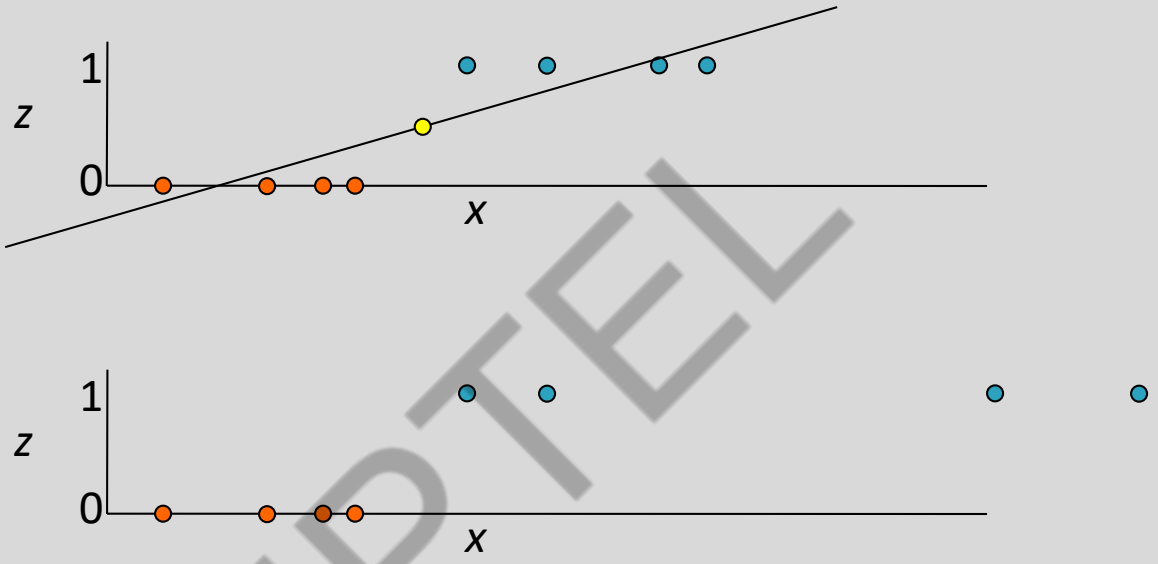
Stochastic gradient descent

- Repeatedly run through the training set.
- Whenever a training point is encountered, update the parameters according to the gradient of the error with respect to that training example only.

Repeat {
 for I = 1 to m do
 $\theta_j := \theta_j + \alpha(y^{(i)} - h(x^{(i)}))x_j{}^{(i)}$ (for every j)
 end for
} until convergence

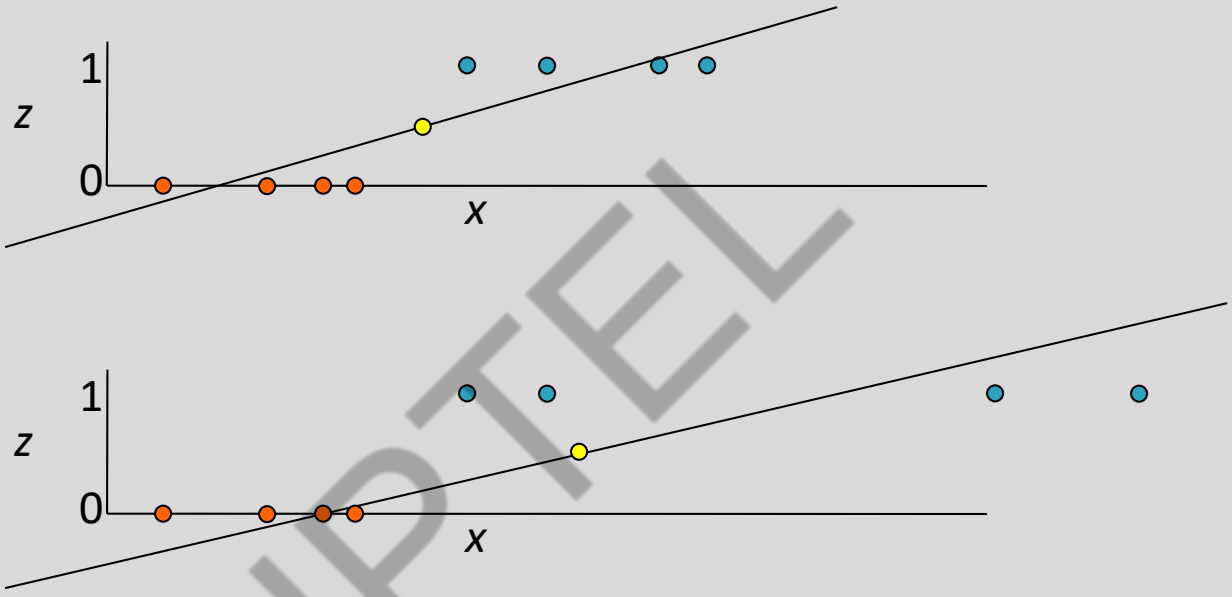
Thank You

Delta Rule for Classification



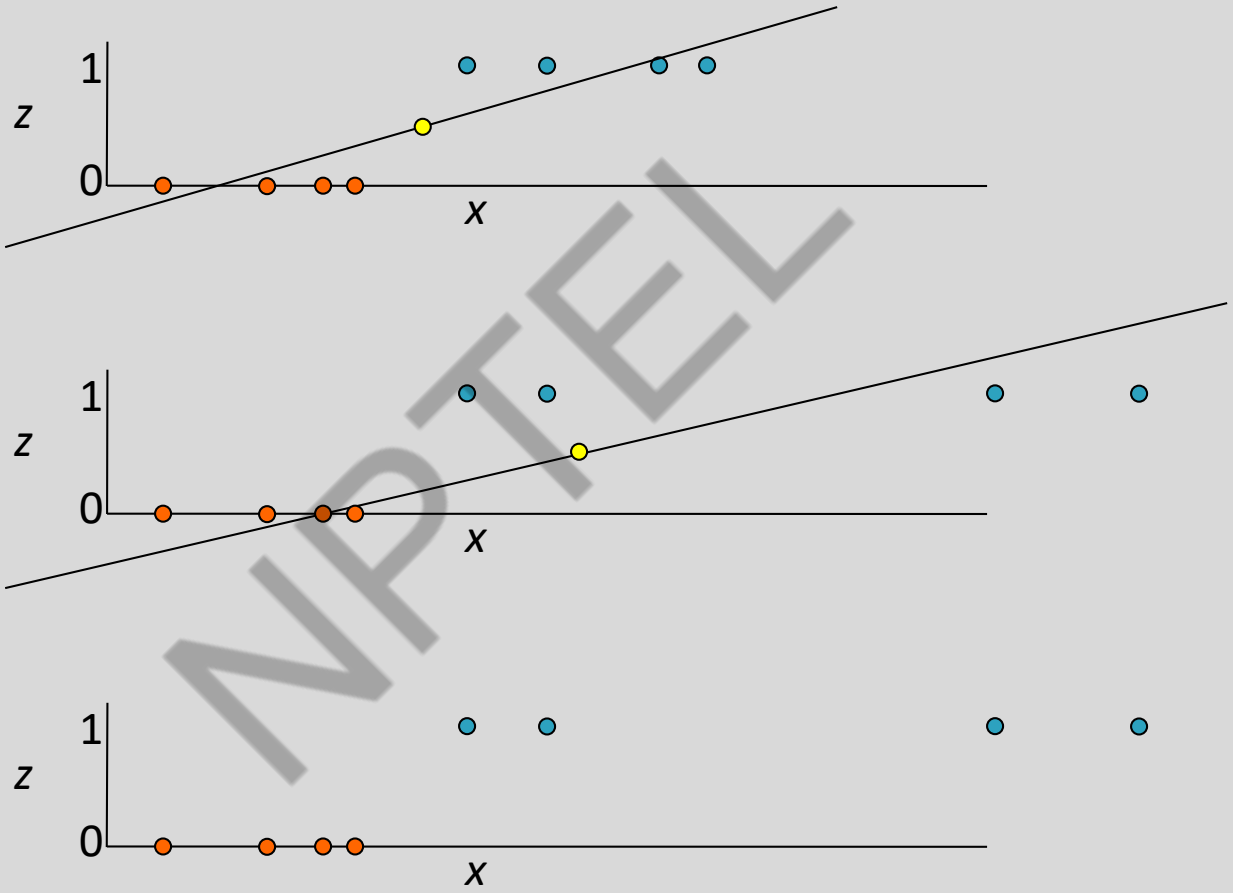
- What would happen in this adjusted case for perceptron and delta rule and where would the decision point (i.e. .5 crossing) be?

Delta Rule for Classification



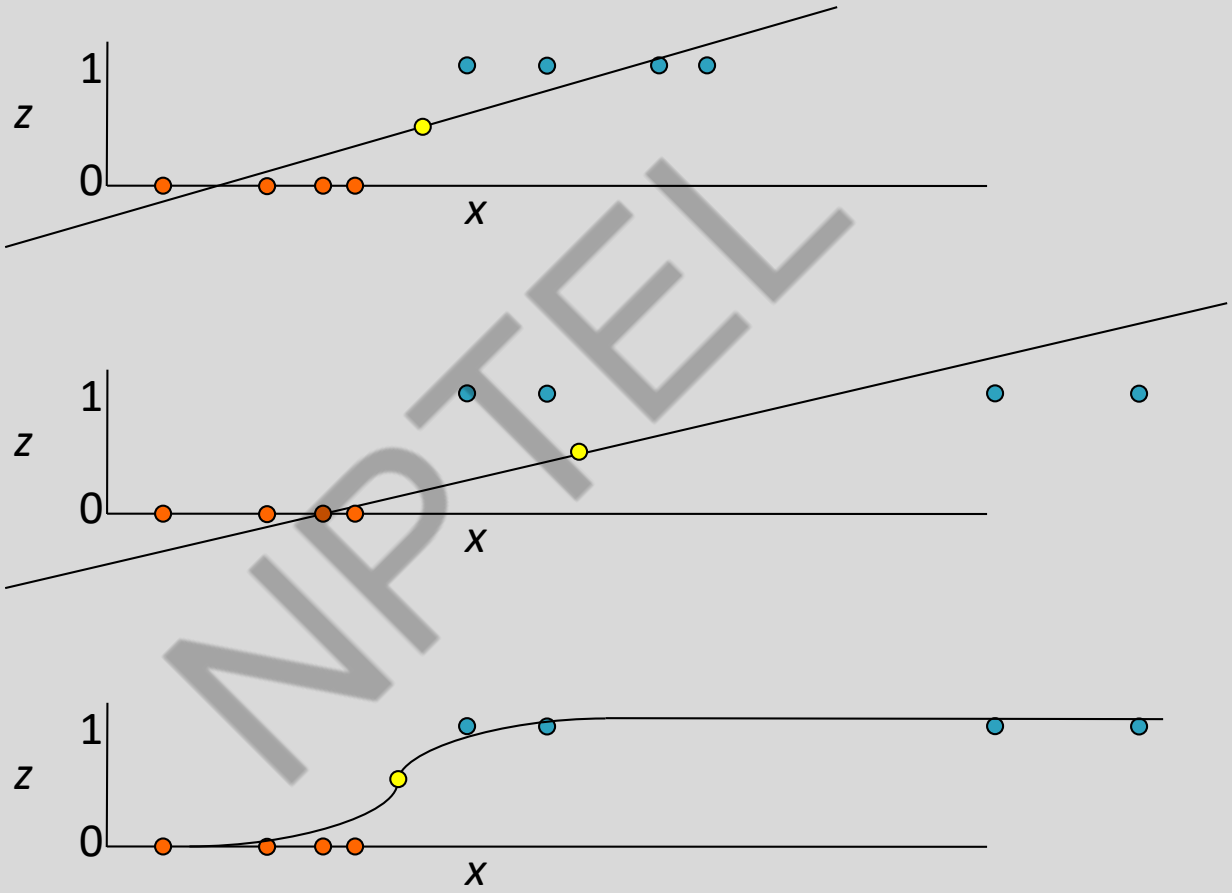
- Leads to misclassifications even though the data is linearly separable
- For Delta rule the objective function is to minimize the regression line SSE, not maximize classification

Delta Rule for Classification



- What would happen if we were doing a regression fit with a sigmoid/logistic curve rather than a line?

Delta Rule for Classification



- Sigmoid fits many decision cases quite well! This is basically what logistic regression does.

Foundations of Machine Learning

Module 2: Linear Regression and Decision Tree

Part B: Introduction to Decision Tree

Sudeshna Sarkar

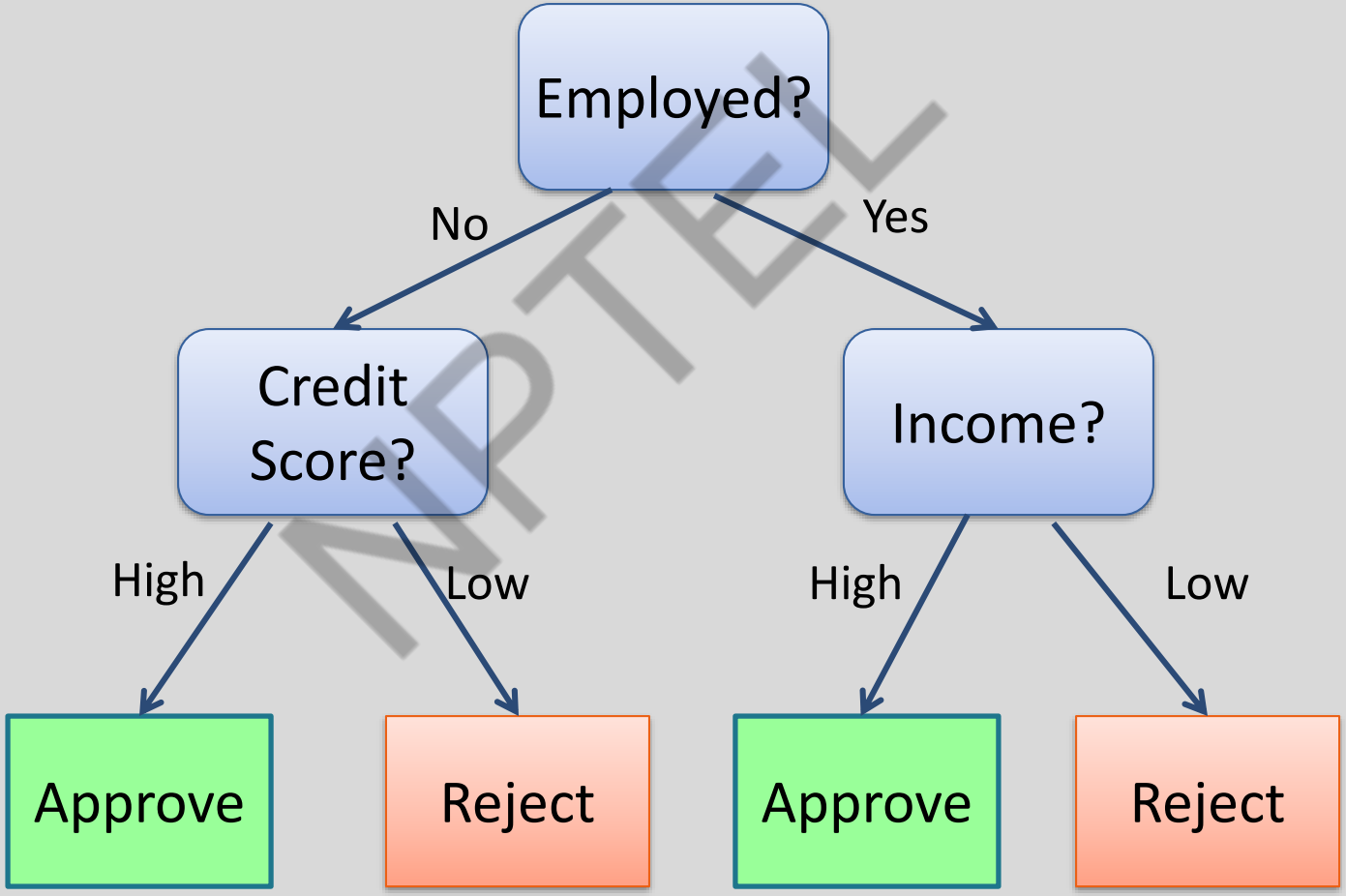
IIT Kharagpur

Definition

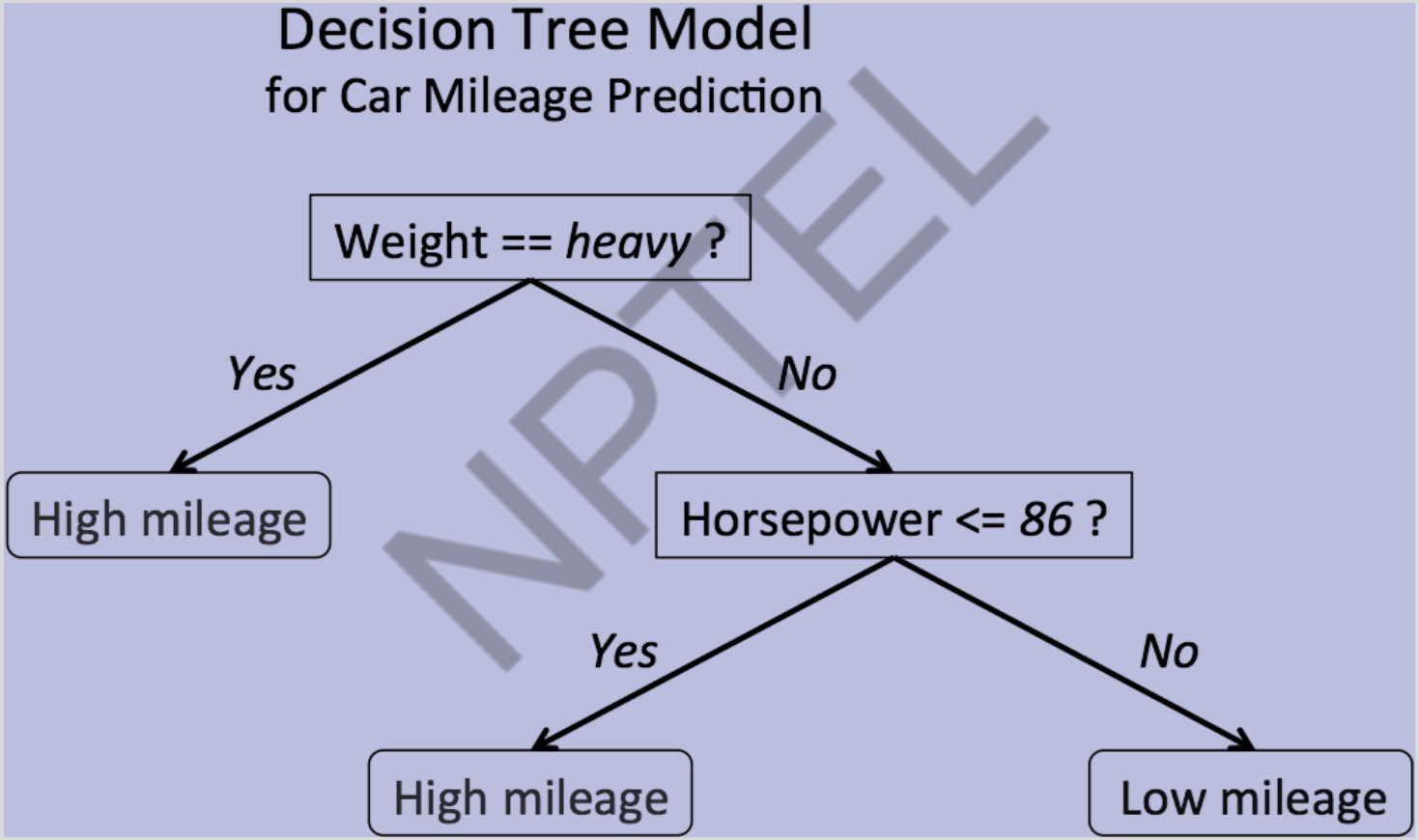
- A decision tree is a classifier in the form of a tree structure with two types of nodes:
  - Decision node: Specifies a choice or test of some attribute, with one branch for each outcome
  - Leaf node: Indicates classification of an example

Decision Tree Example 1

Whether to approve a loan



Decision Tree Example 3



Issues

- Given some training examples, what decision tree should be generated?
- One proposal: prefer the smallest tree that is consistent with the data (Bias)

- Possible method:
  - search the space of decision trees for the smallest decision tree that fits the data

Example Data

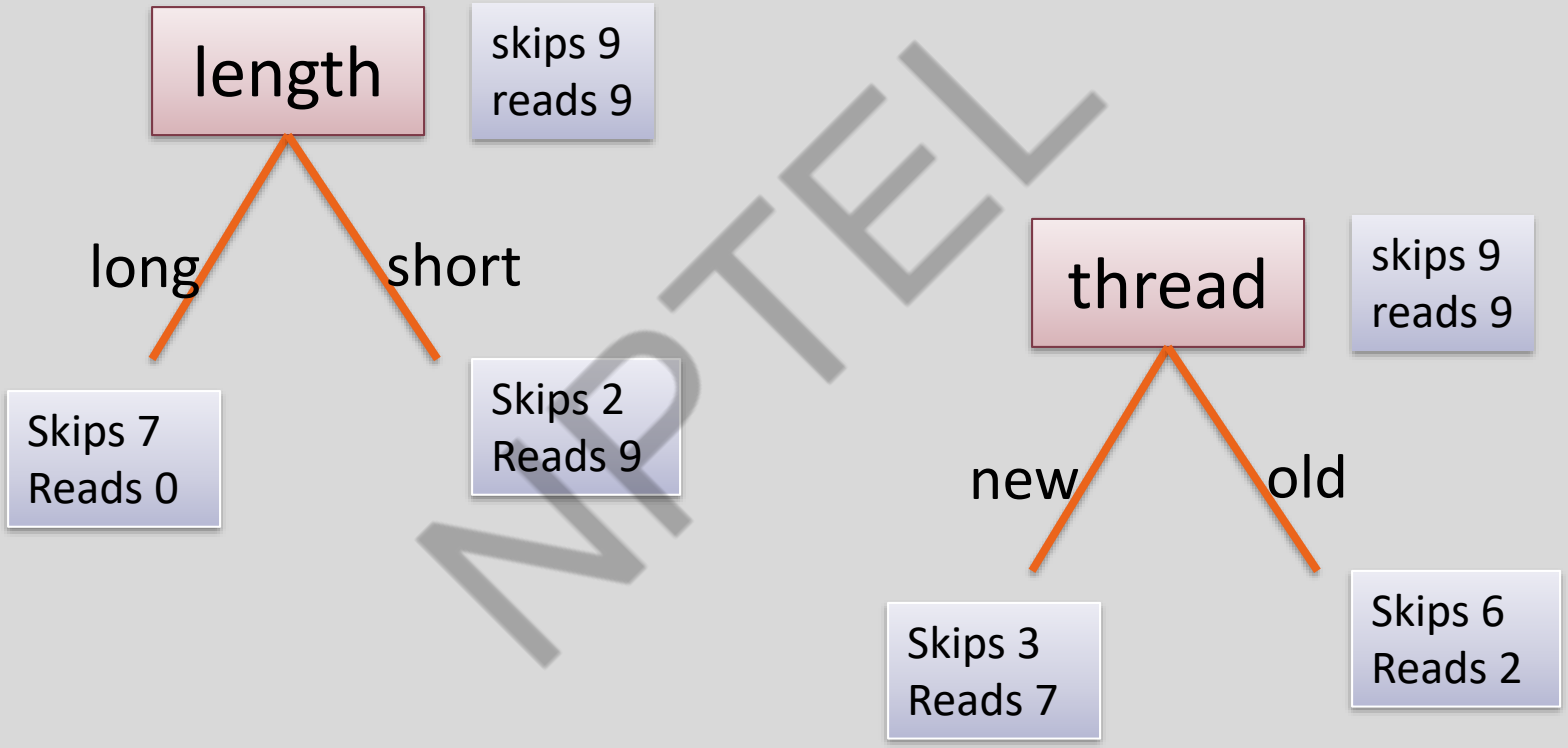
Training Examples:

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	Home
e2	reads	unknown	new	short	Work
e3	skips	unknown	old	long	Work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

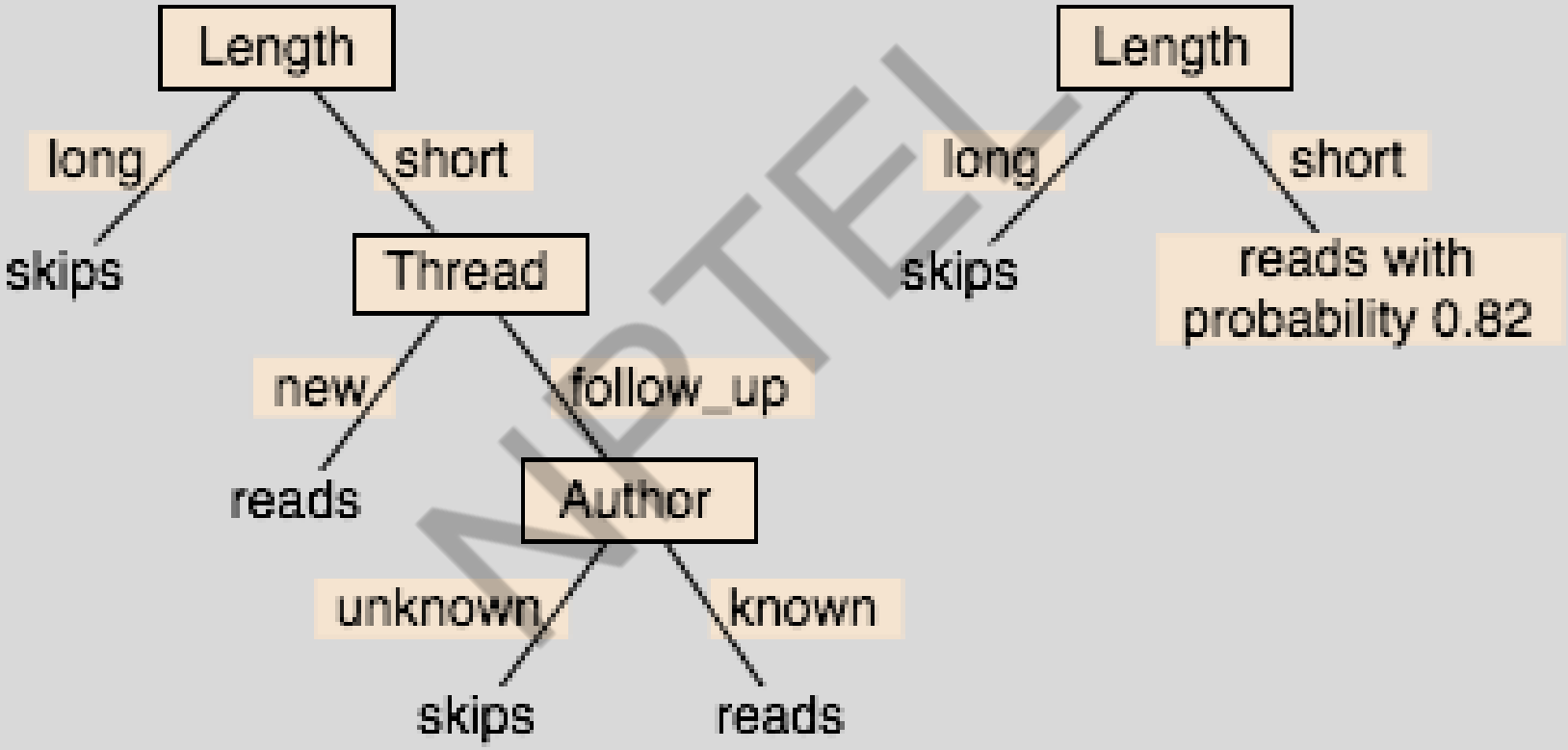
New Examples:

e7	???	known	new	short	work
e8	???	unknown	new	short	work

Possible splits



Two Example DTs

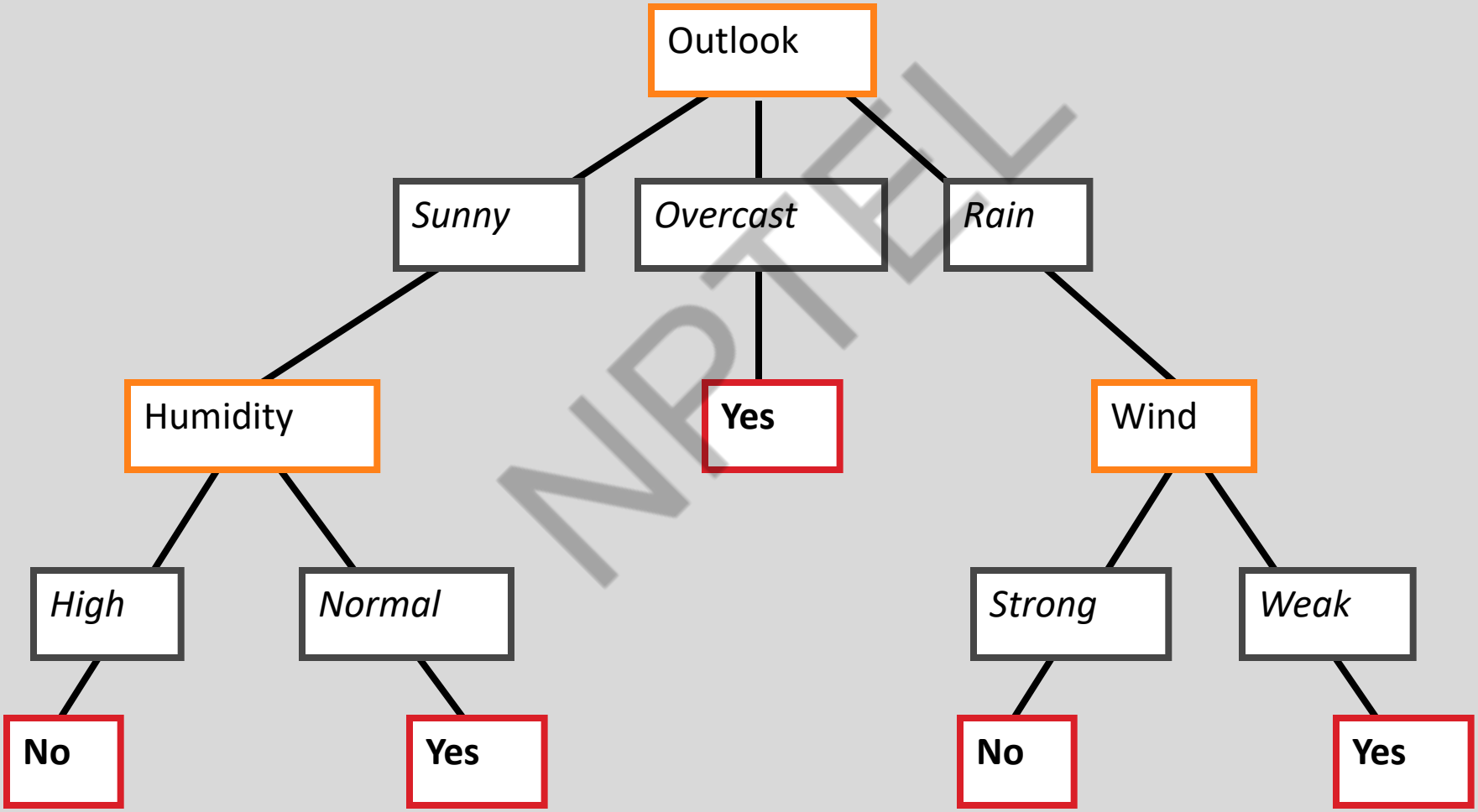


Decision Tree for PlayTennis

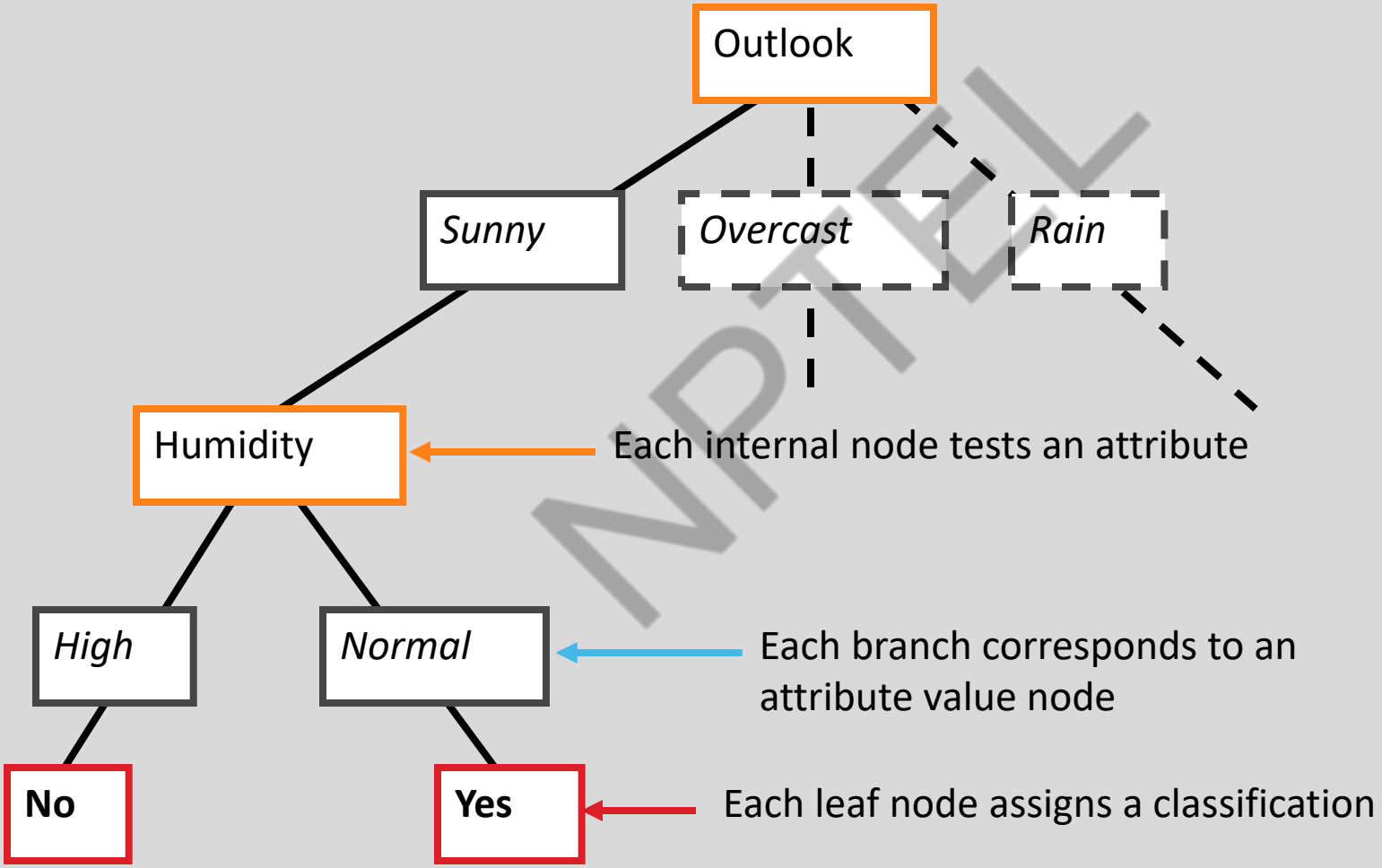
- Attributes and their values:
  - Outlook: *Sunny, Overcast, Rain*
  - Humidity: *High, Normal*
  - Wind: *Strong, Weak*
  - Temperature: *Hot, Mild, Cool*

- Target concept - Play Tennis: *Yes, No*

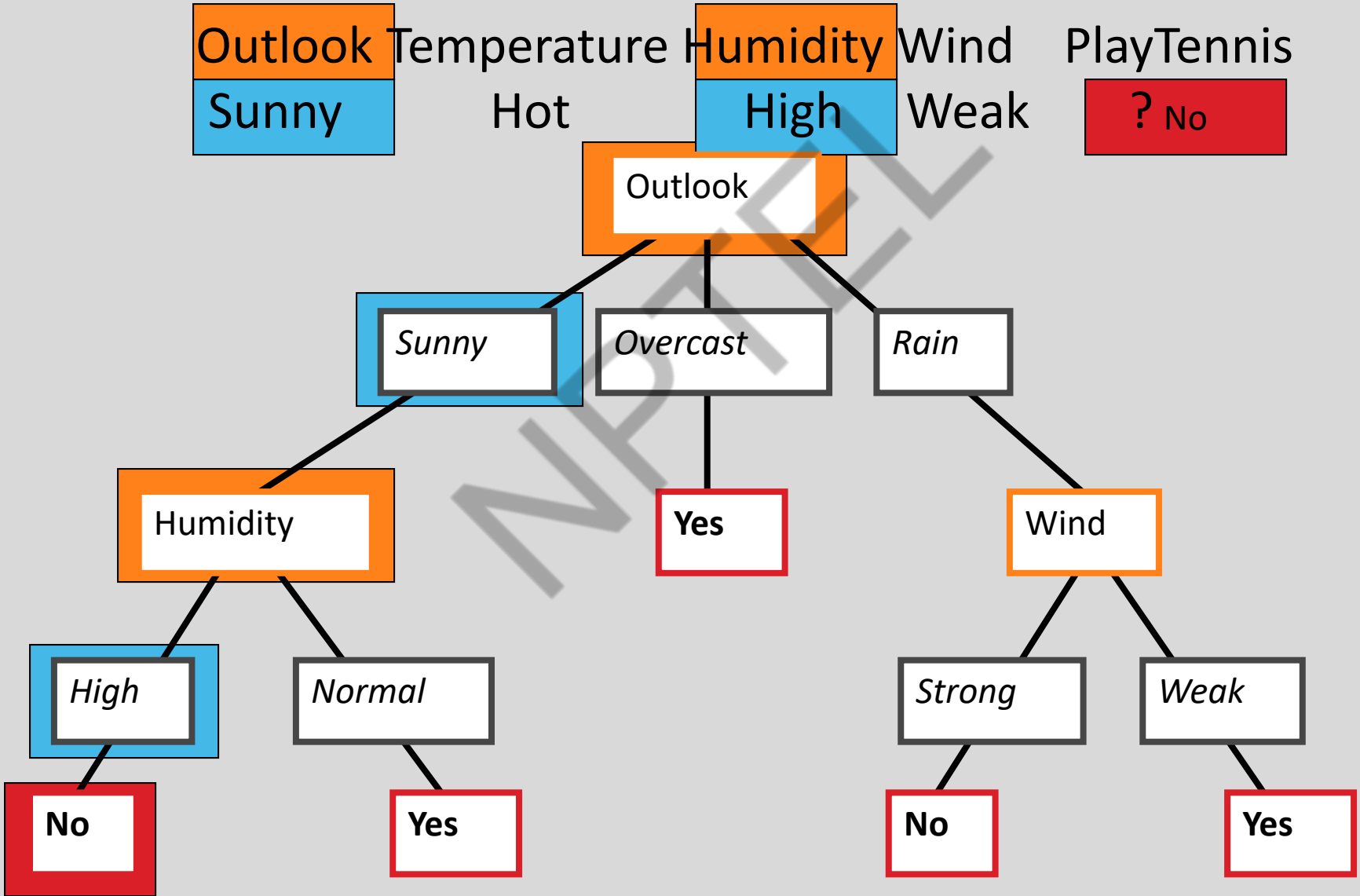
Decision Tree for PlayTennis



Decision Tree for PlayTennis

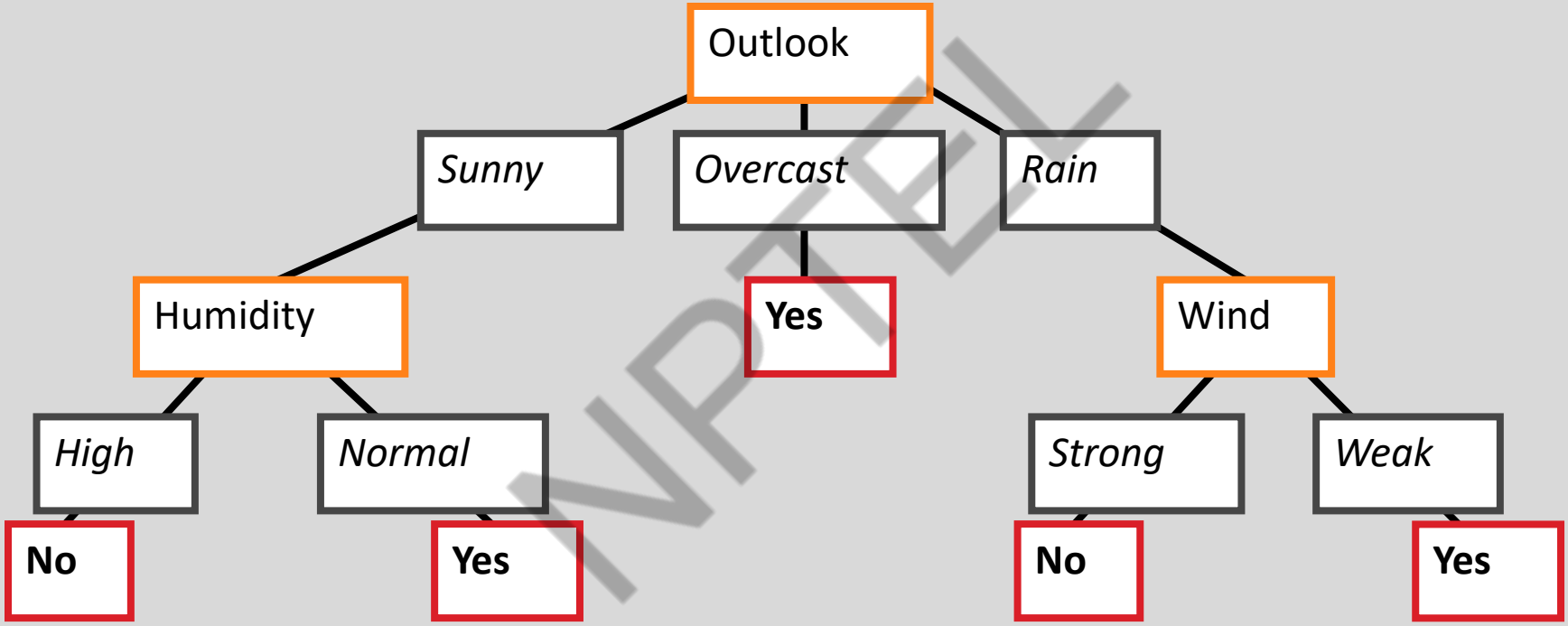


Decision Tree for PlayTennis



Decision Tree

decision trees represent disjunctions of conjunctions



(Outlook=Sunny $\wedge$ Humidity=Normal)
$\vee$ (Outlook=Overcast)
$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

Searching for a good tree

- The space of decision trees is too big for systematic search.

- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values

- Choose a test (e.g. an input feature) to split on.
  - For each value of the test, build a subtree for those examples with this value for the test.

Top-Down Induction of Decision Trees ID3

1. Which node to proceed with?

1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

2. When to stop?

Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split

- Which test to split on
  - split gives smallest error.
  - With multi-valued features
    - split on all values or
    - split values into half.

Foundations of Machine Learning

Module 2: Linear Regression and Decision Tree

Part C: Learning Decision Tree

Sudeshna Sarkar

IIT Kharagpur

Top-Down Induction of Decision Trees ID3

1. Which node to proceed with?

1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

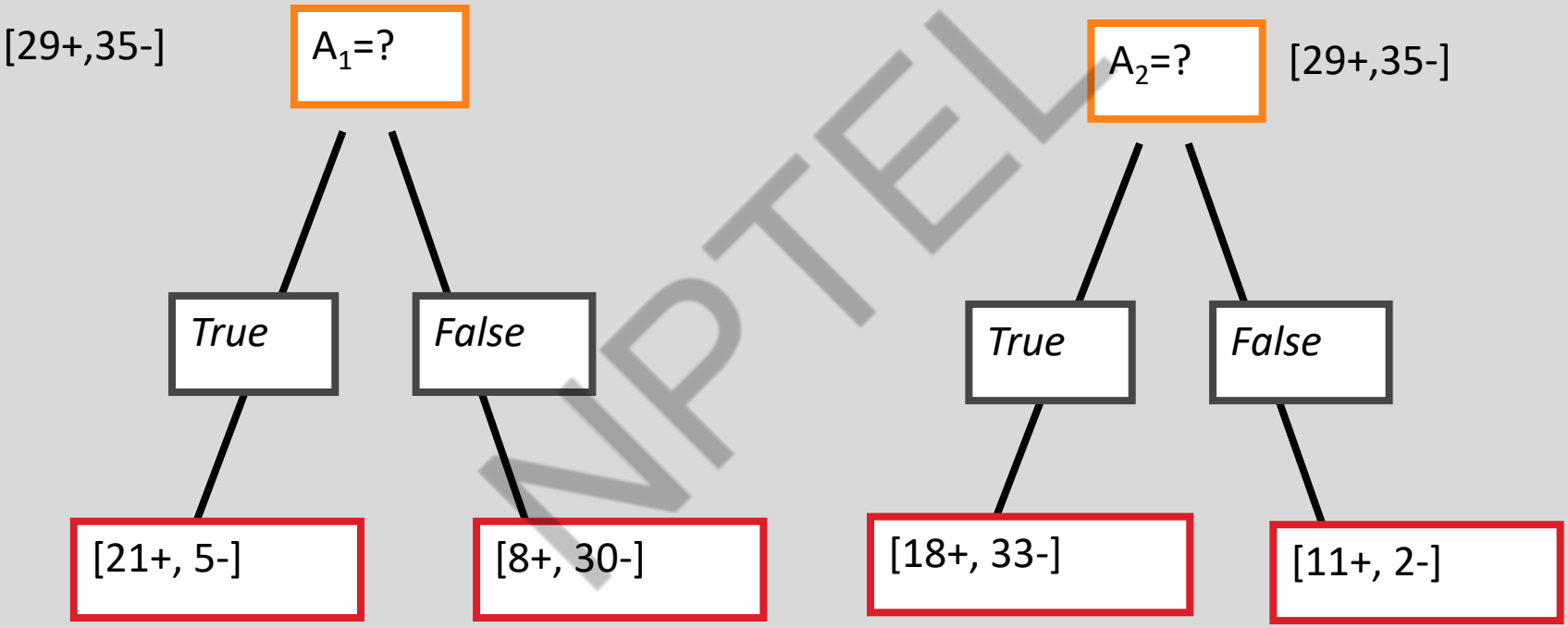
2. When to stop?

Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split

- Which test to split on
  - split gives smallest error.
  - With multi-valued features
    - split on all values or
    - split values into half.

Which Attribute is ”best”?



Principled Criterion

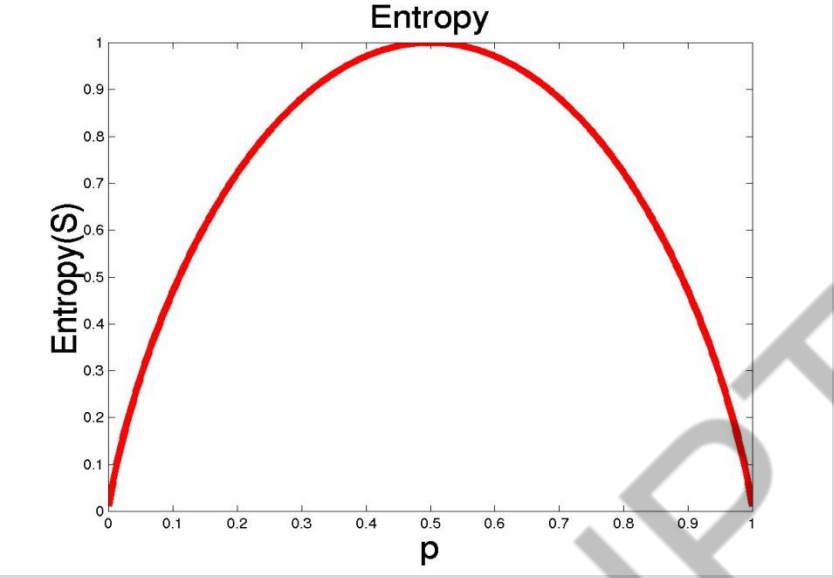
- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)

Entropy

- A measure for
  - uncertainty
  - purity
  - information content
- Information theory: optimal length code assigns $(-\log_2 p)$ bits to message having probability p
- S is a sample of training examples
  - p_+ is the proportion of positive examples in S
  - p_- is the proportion of negative examples in S
- Entropy of S: average optimal number of bits to encode information about certainty/uncertainty about S

$$Entropy(S) = p_+(-\log_2 p_+) + p_-(-\log_2 p_-) = -p_+\log_2 p_+ - p_-\log_2 p_-$$

Entropy



- The entropy is 0 if the outcome is ``certain".
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

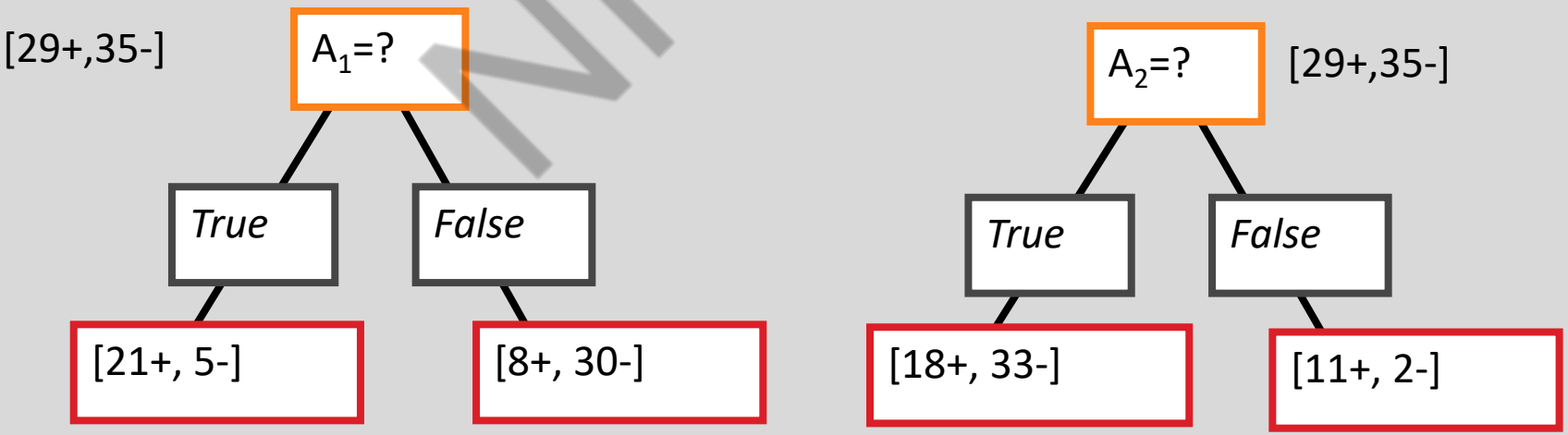
$$\text{Entropy(S)} = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$Gain(S,A)=Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$$

Entropy([29+,35-]) = $-29/64 \log_2 29/64 - 35/64 \log_2 35/64$
= 0.99



Information Gain

Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain(S,A_1)=Entropy(S)

-26/64*Entropy([21+,5-])

-38/64*Entropy([8+,30-])

=0.27

Entropy([18+,33-]) = 0.94

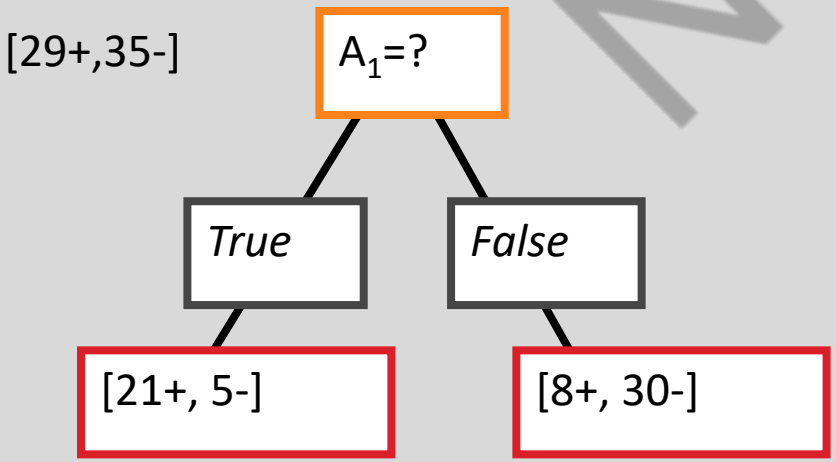
Entropy([8+,30-]) = 0.62

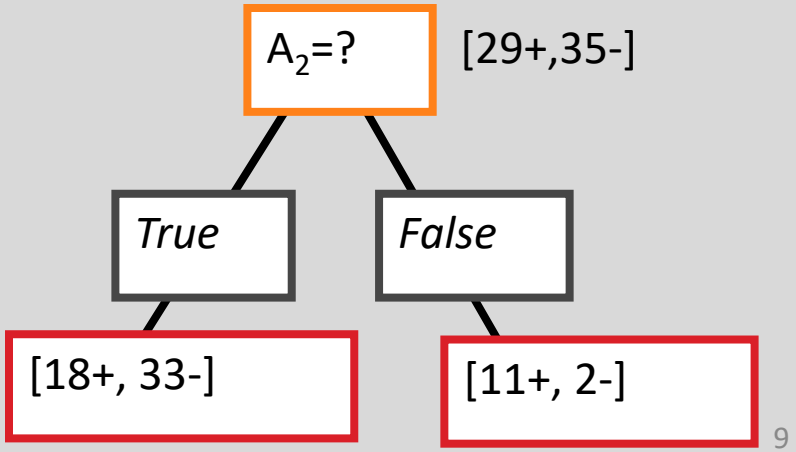
Gain(S,A_2)=Entropy(S)

-51/64*Entropy([18+,33-])

-13/64*Entropy([11+,2-])

=0.12

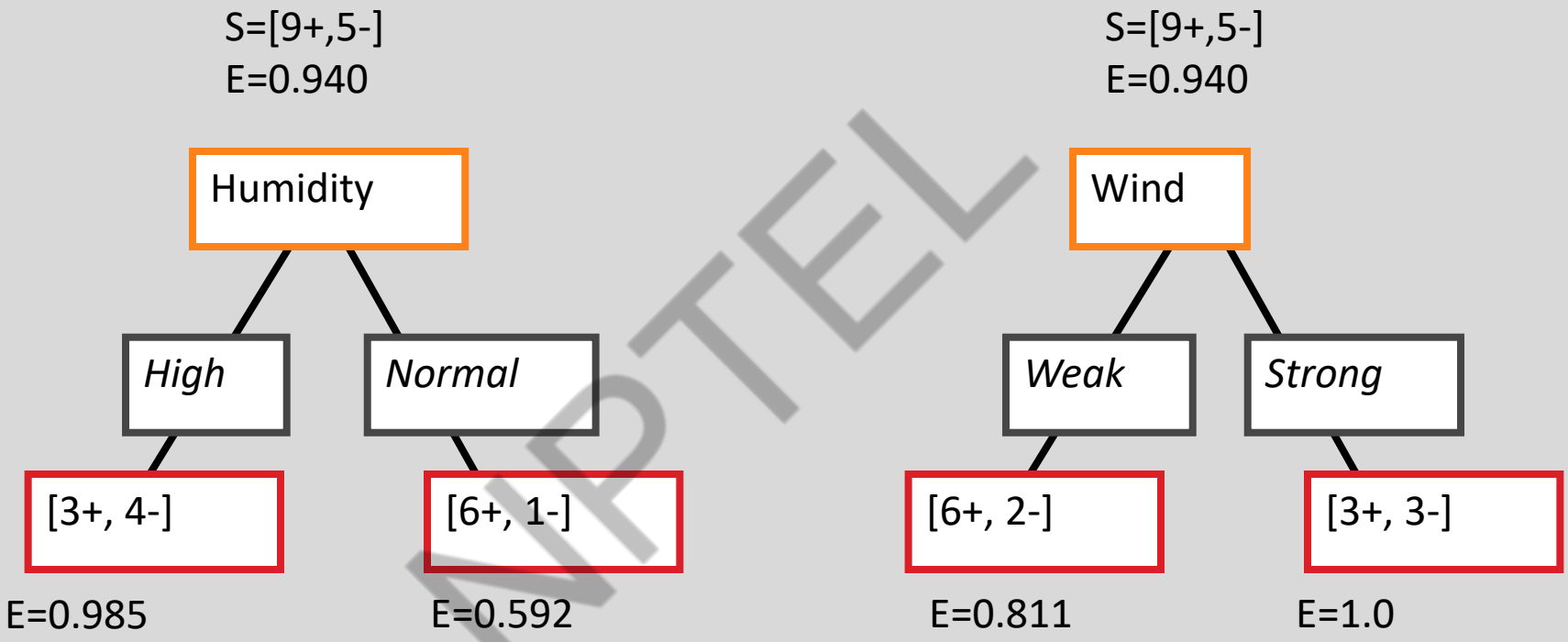




Training Examples

Day	Outlook	Temp	Humidity	Wind	Tennis?
D1	Sunny	Hot	High	Weak	*No*
D2	Sunny	Hot	High	Strong	*No*
D3	Overcast	Hot	High	Weak	*Yes*
D4	Rain	Mild	High	Weak	*Yes*
D5	Rain	Cool	Normal	Weak	*Yes*
D6	Rain	Cool	Normal	Strong	*No*
D7	Overcast	Cool	Normal	Strong	*Yes*
D8	Sunny	Mild	High	Weak	*No*
D9	Sunny	Cool	Normal	Weak	*Yes*
D10	Rain	Mild	Normal	Weak	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*
D14	Rain	Mild	High	Strong	*No*

Selecting the Next Attribute



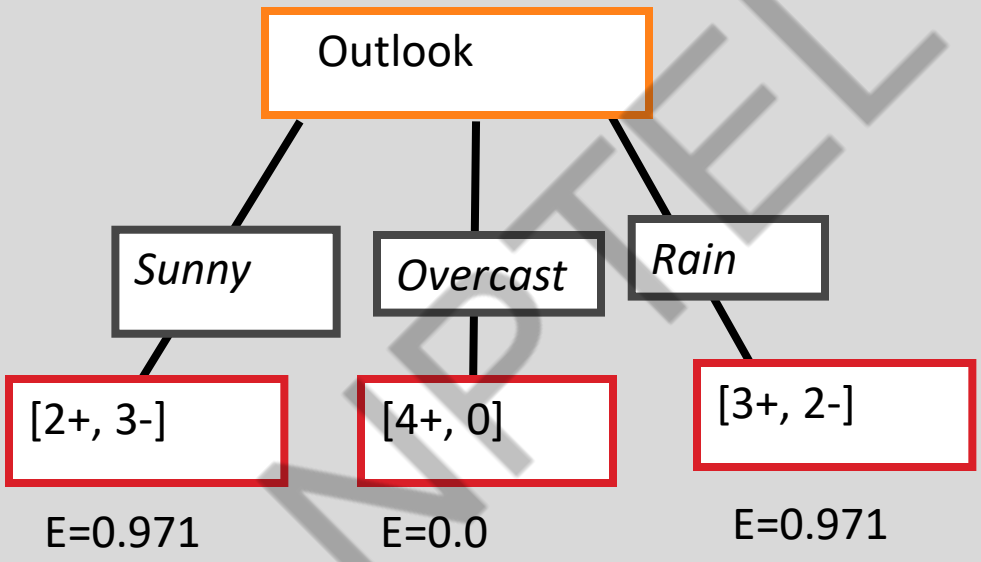
Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Next Attribute

S=[9+,5-]
E=0.940



Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

Selecting the Next Attribute

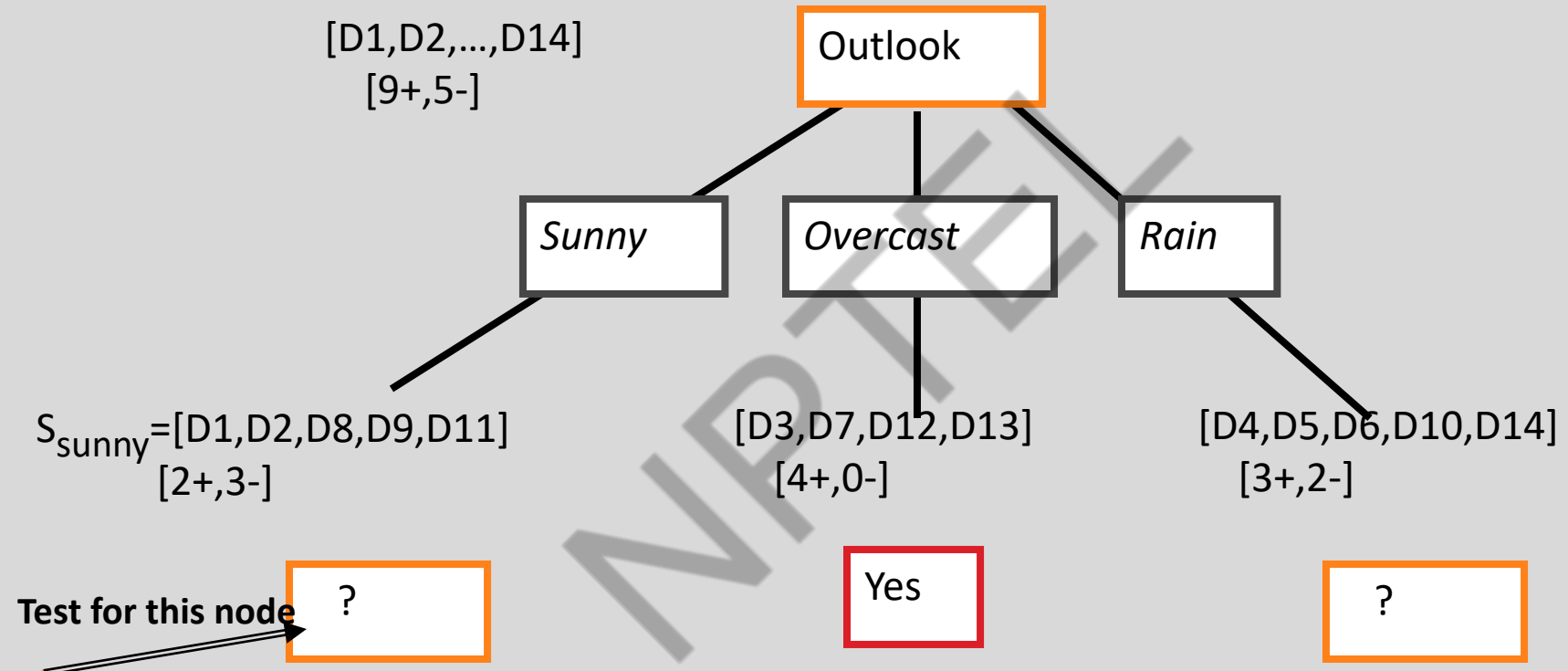
The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

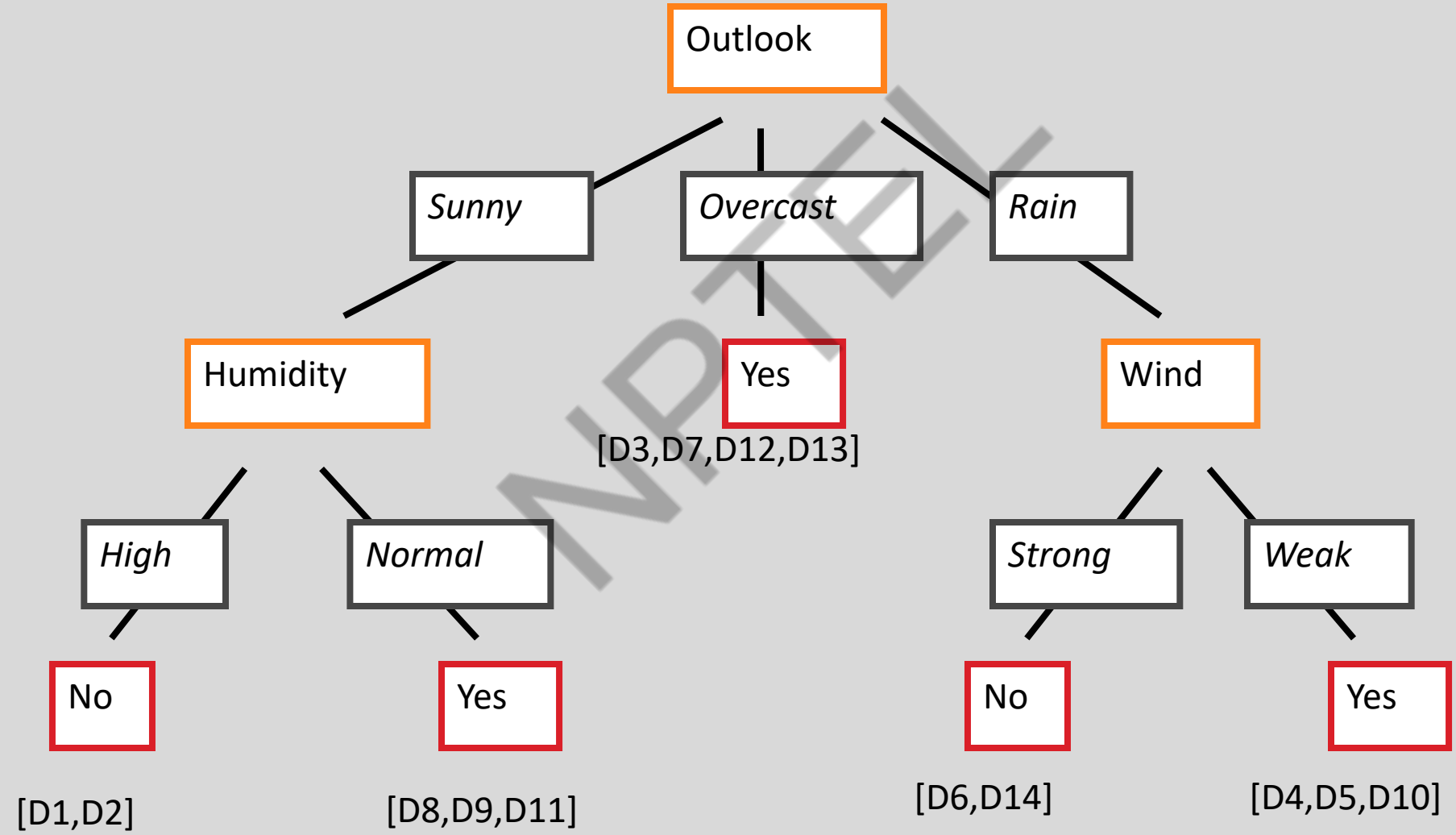
Note: $0\text{Log}_2 0 = 0$

ID3 Algorithm



Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970
Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570
Gain(S_{sunny} , Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



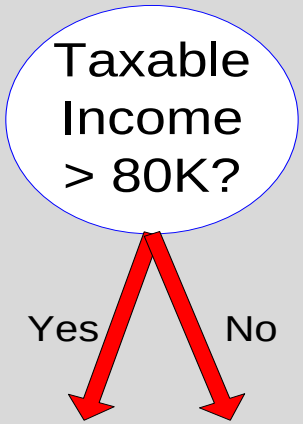
Splitting Rule: GINI Index

- GINI Index
  - Measure of node impurity

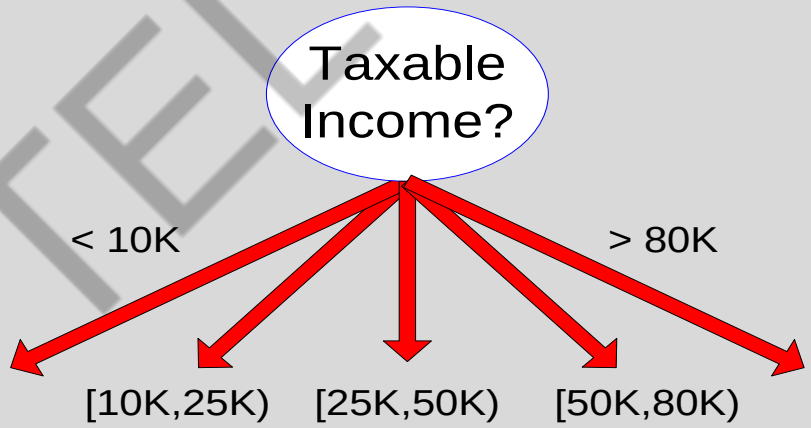
$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

Continuous Attribute – Binary Split

- For continuous attribute
  - Partition the continuous value of attribute A into a discrete set of intervals
  - Create a new boolean attribute A_c , looking for a threshold c,

$$A_c = \begin{cases} true & \text{if } A_c < c \\ false & \text{otherwise} \end{cases}$$

How to choose c ?

- consider all possible splits and finds the best cut

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Hypothesis Space Search in Decision Trees

- Conduct a search of the space of decision trees which can represent all possible discrete functions.
- Goal: to find the best decision tree
- Finding a minimal decision tree consistent with a set of data is NP-hard.
- Perform a greedy heuristic search: hill climbing without backtracking
- Statistics-based decisions using all data

Bias and Occam's Razor

Prefer short hypotheses.

Argument in favor:

- Fewer short hypotheses than long hypotheses
- A short hypothesis that fits the data is unlikely to be a coincidence
- A long hypothesis that fits the data might be a coincidence

Foundations of Machine Learning

Module 2: Linear Regression and Decision Tree

Part D: Overfitting

Sudeshna Sarkar

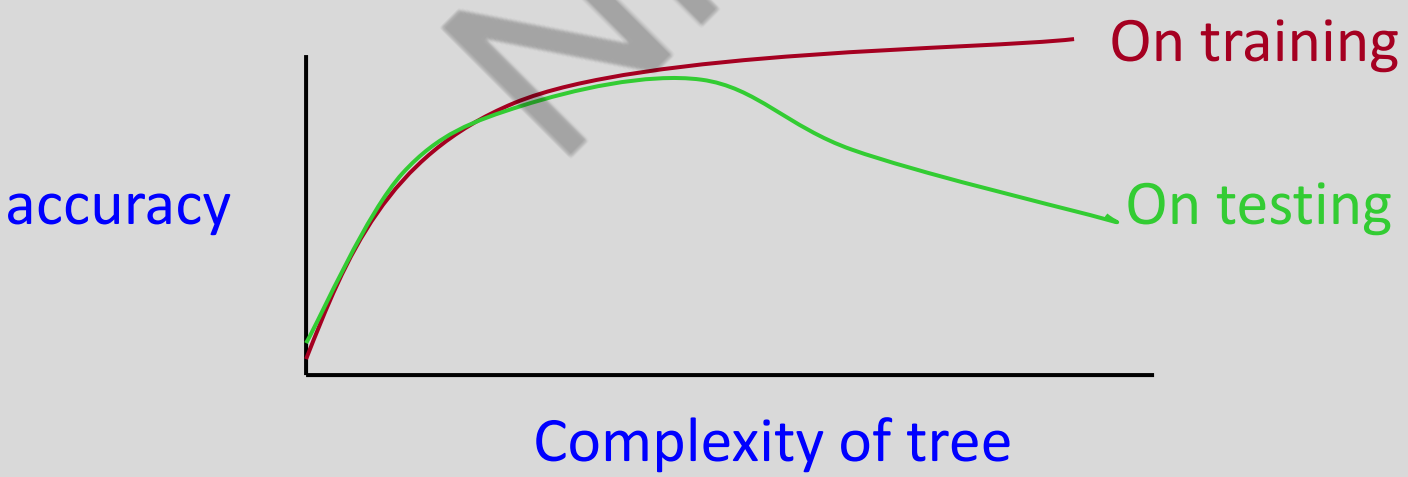
IIT Kharagpur

Overfitting

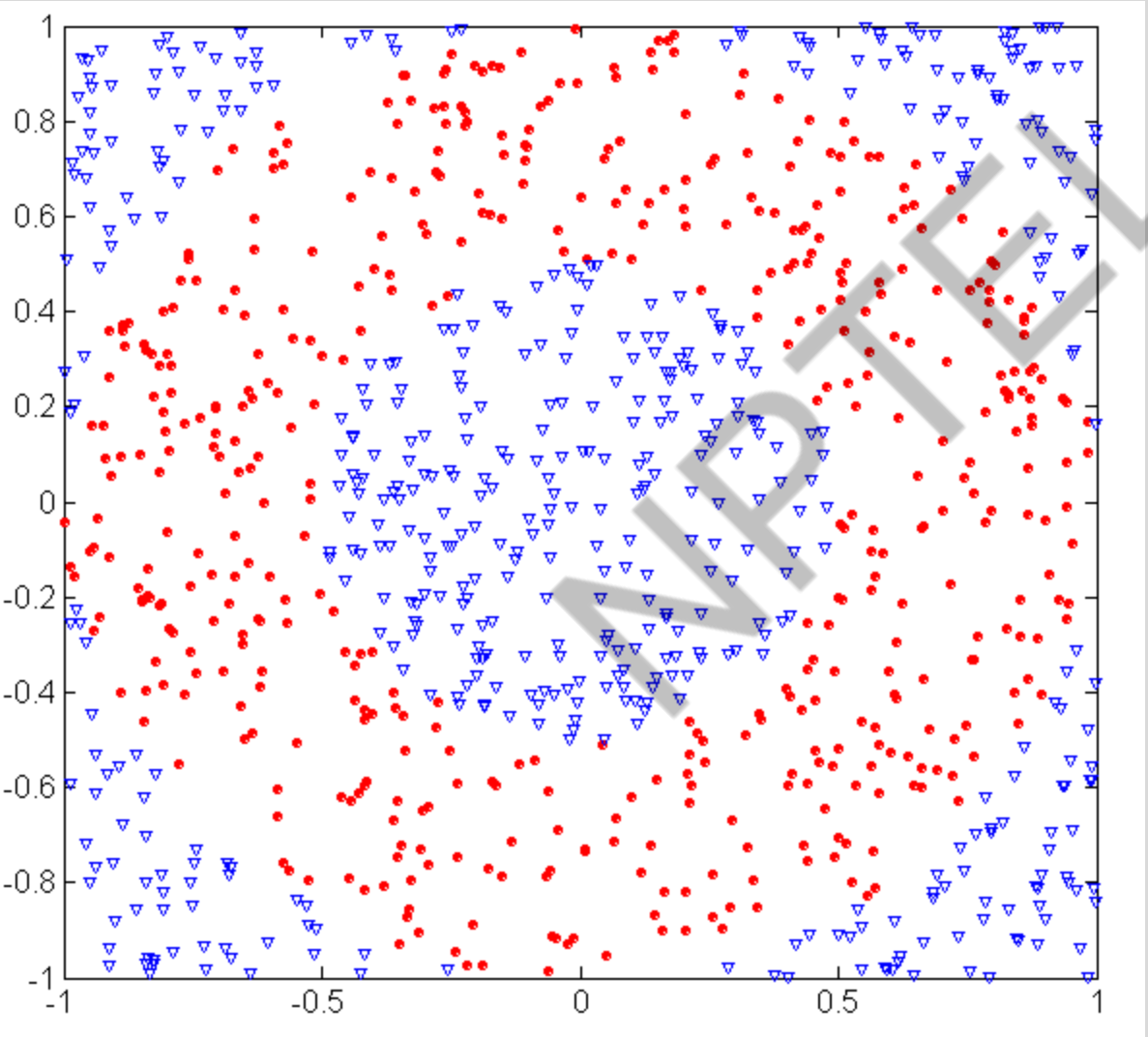
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data
  - May be based on insufficient data
- A hypothesis *h* is said to overfit the training data if there is another hypothesis, h',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



Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

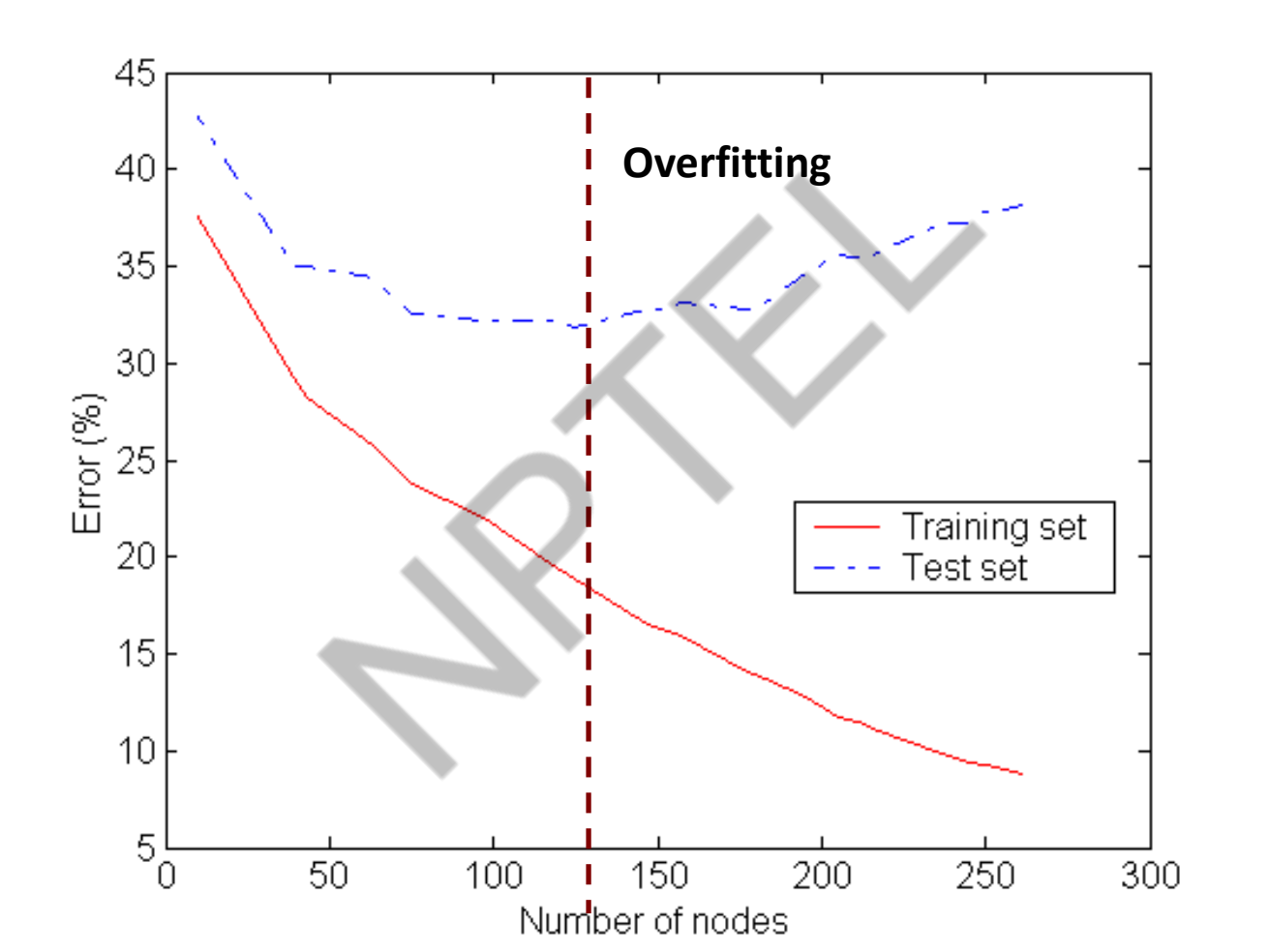
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

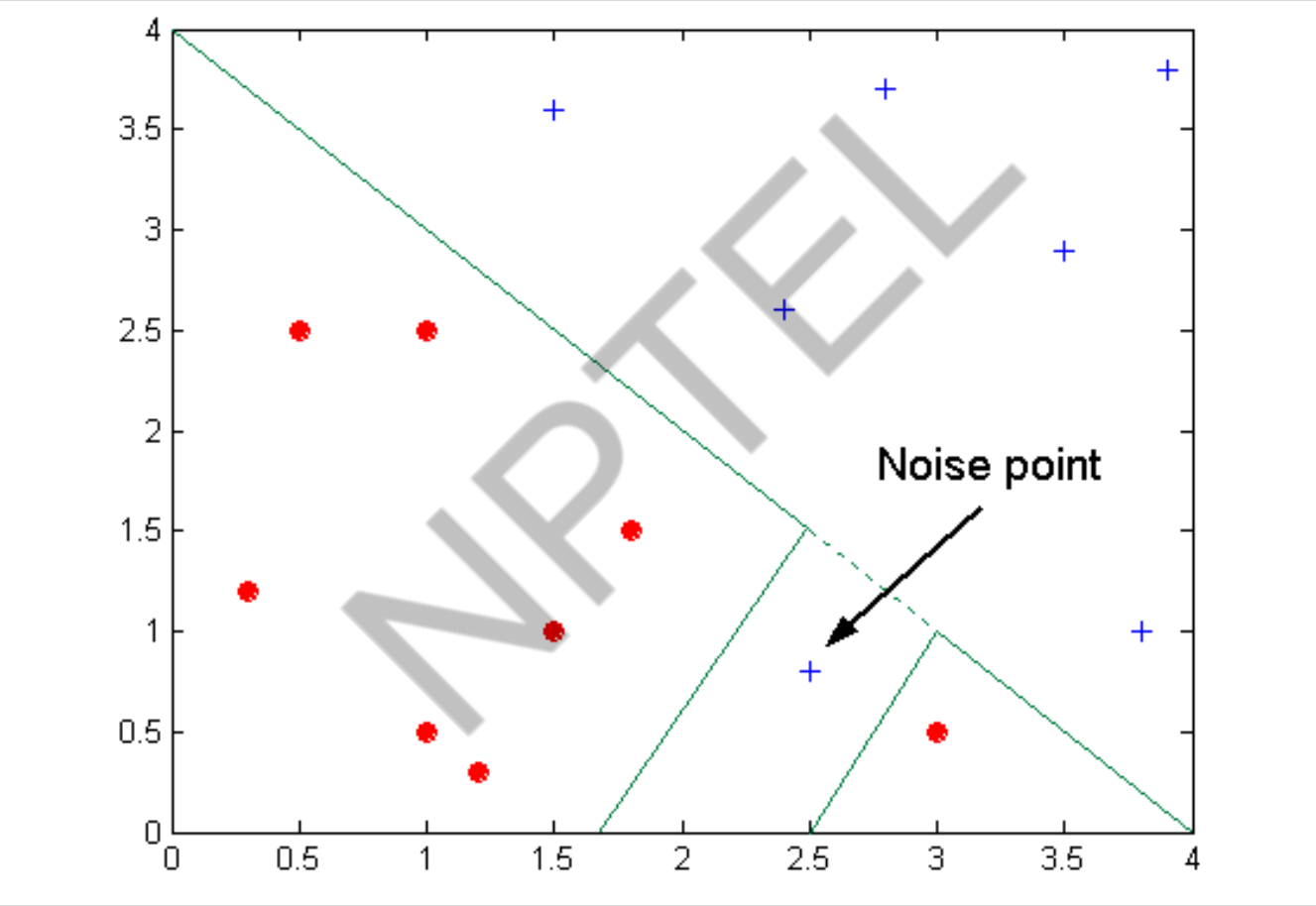
$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



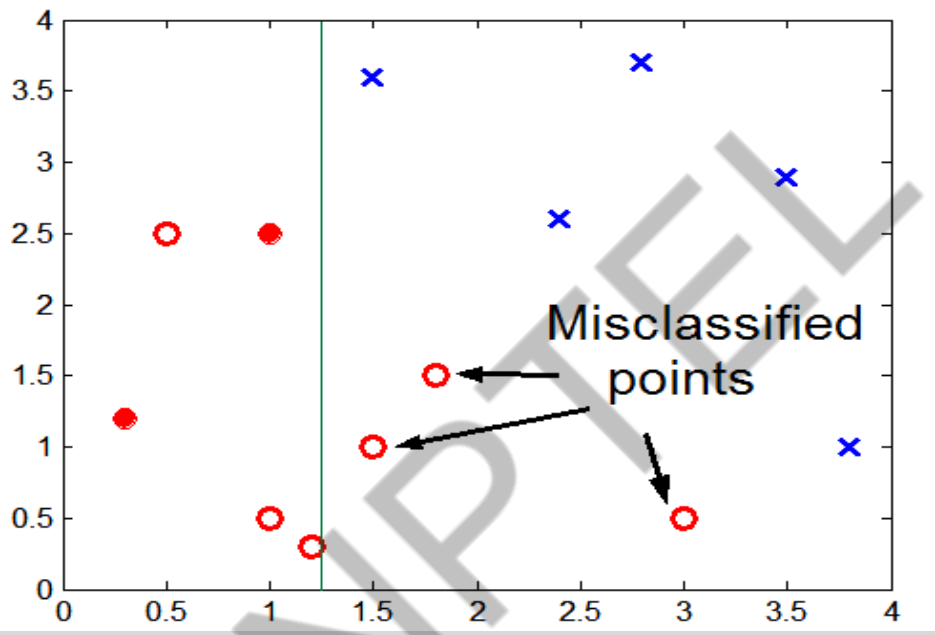
Underfitting: when model is too simple, both training and test errors are large

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 χ^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

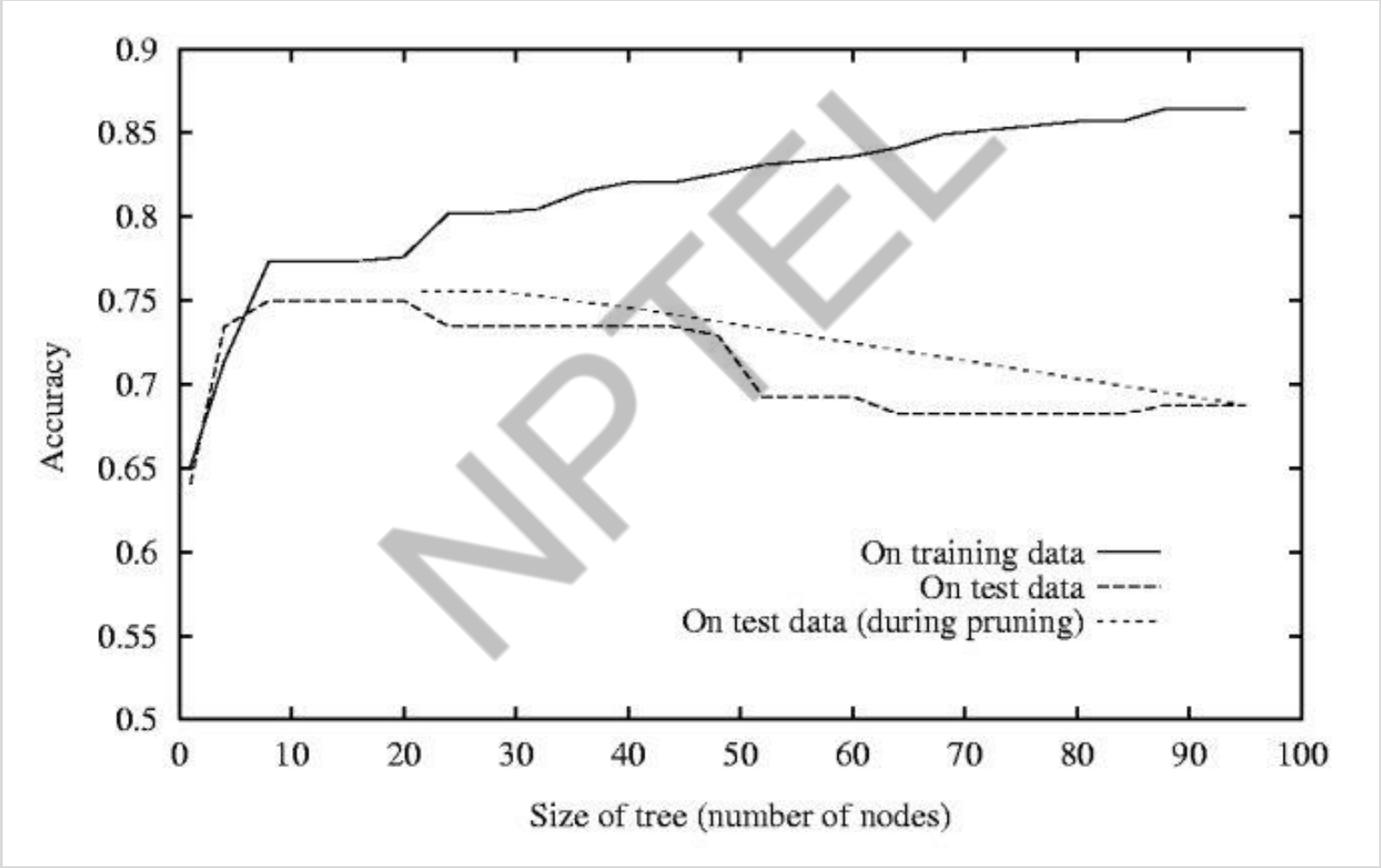
    Temporarily prune the tree below; replace it by majority vote

    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

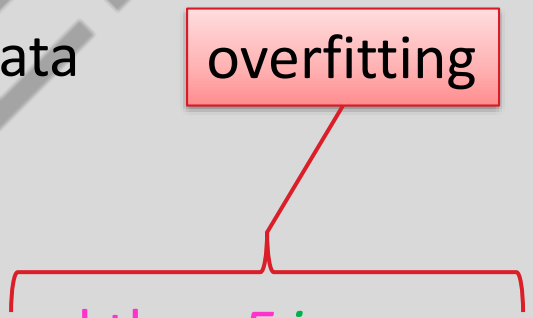


Model Selection & Generalization

- 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



- 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

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