Non-Parametric Methods: Nearest Neighbor and Decision Tree Classification and Bagging for Constructing Ensembles

Parametric vs. Non-Parametric Learning

- **Parametric learning**
  - define a space of models parameterized by a fixed number of parameters
  - find model that best fits the data (by searching over the parameters)
  - Example: Perceptron, PA, and SVMs
- **Non-Parametric learning**
  - define a space of models that can grow in size with data
  - find model that best fits the data
  - “Non-parametric” means “Not-fixed”
  - Example: K-nearest neighbor and decision trees

Nearest-Neighbor Classifier

Nearest-neighbor Classifier

- Given a set of training examples $(x_i, y_i): i = 1,2,..N$, and a test example x

- **Prediction Rule:**
  - Find the training point x_j such that the distance between x and x_j is minimum (If tied, break ties uniformly at random)
  - Output y_j

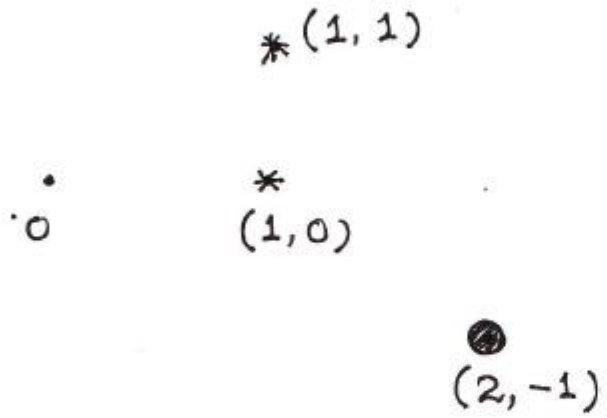
Example 1

Training data:

((1, 0), 0) , ((1, 1), 0) , ((2, -1), 1)

Test points:

(0, 0) , (2, 1) , (1·5, -0·5)



- dist ((0, 0), (1, 0)) = 1

  dist ((0, 0), (1, 1)) = $\sqrt{2}$

  dist ((0, 0), (2, -1)) = $\sqrt{5}$

so: closest point to (0, 0) = (1, 0), Label = 0

Example 1 (contd.)

- dist ((2, 1), (1, 0)) = $\sqrt{2}$

  dist ((2, 1), (1, 1)) = 1

  dist ((2, 1), (2, −1)) = 2.

closest : (1, 1)

output: $y = 0$

- dist ((1·5, −0·5), (1, 0)) = $\frac{1}{\sqrt{2}}$

  dist ((1·5, −0·5), (1, 1)) = $\sqrt{\frac{5}{2}}$

  dist ((1·5, −0·5), (2, −1)) = $\frac{1}{\sqrt{2}}$

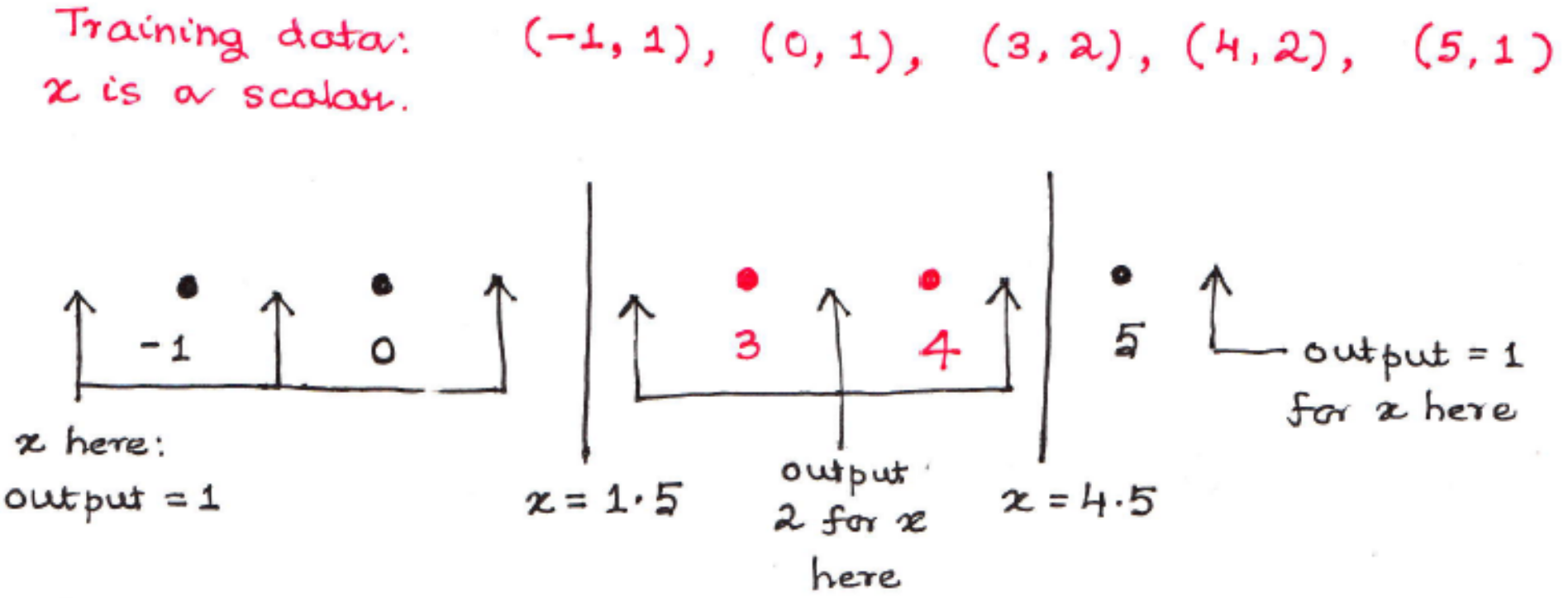
closest: (1, 0), (2, −1)

Break ties at random

(report $y = 0$ w.p. 1/2

$y = 1$ w.p. 1/2)

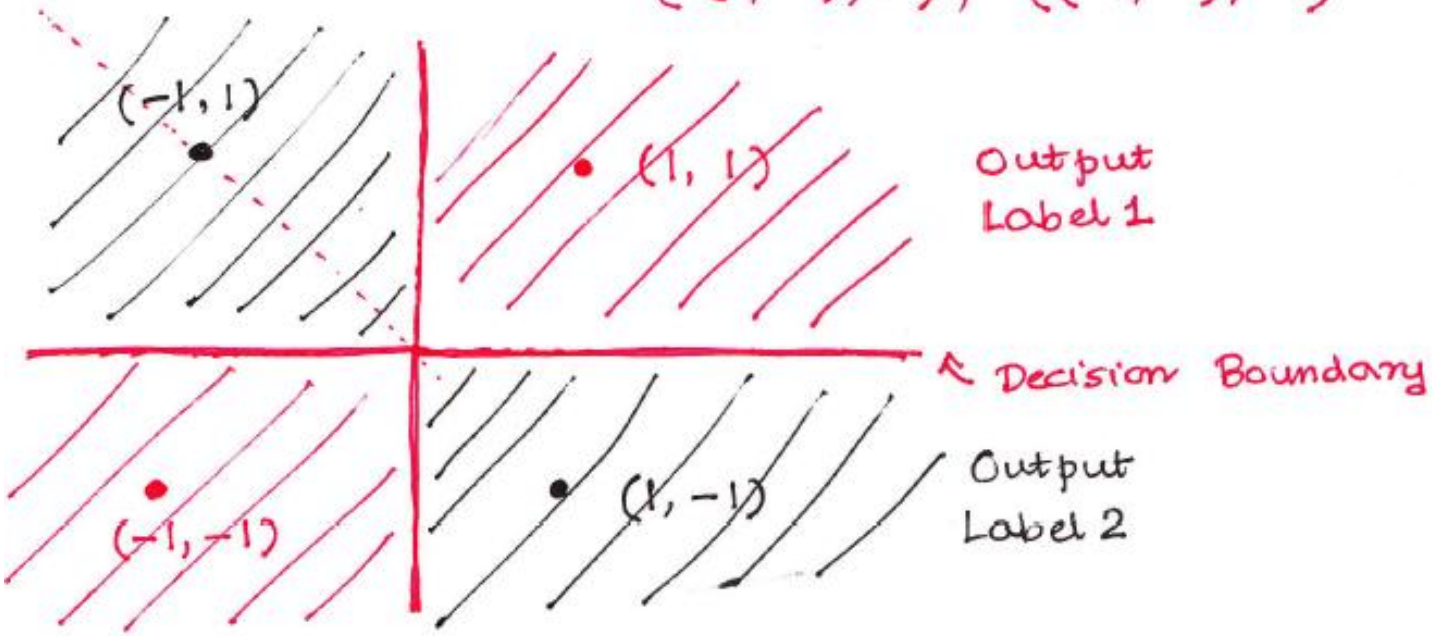
Example 2



- **Decision boundary:**
  - boundary between regions of different classes
  - the output changes at the boundary

Example 3

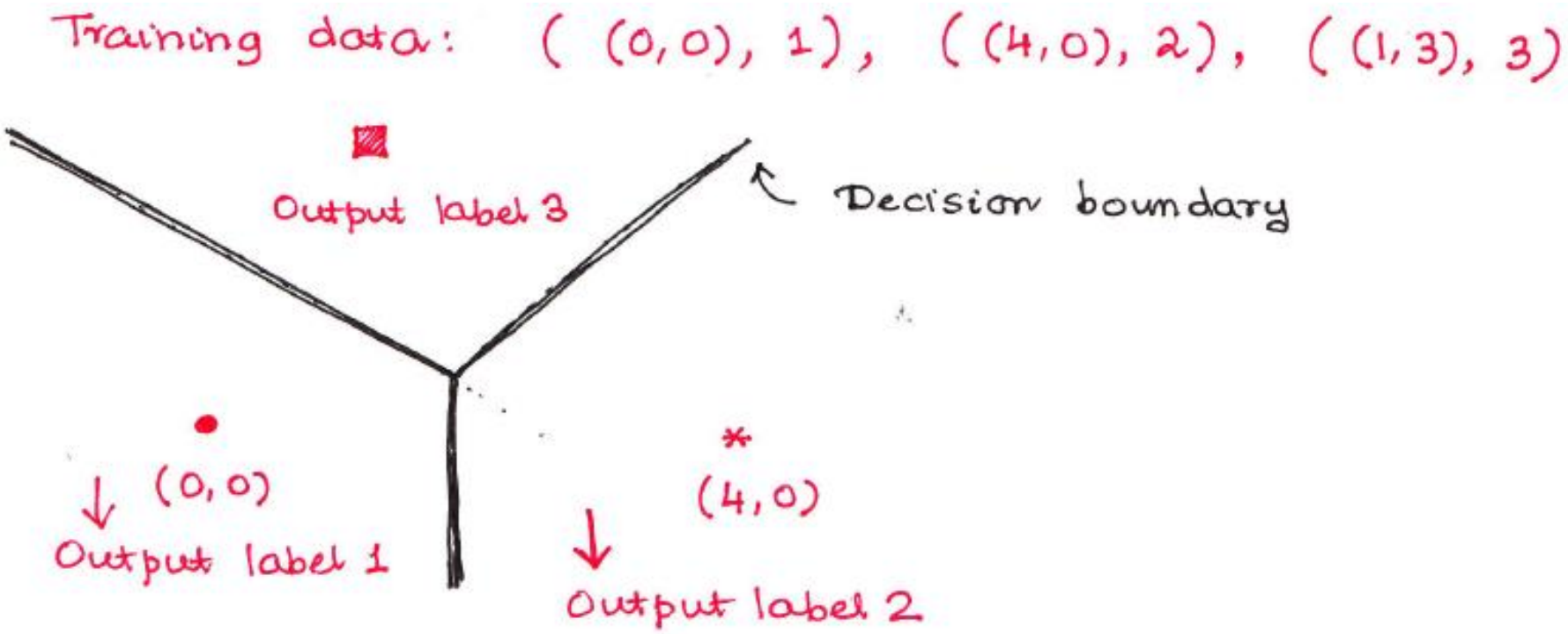
Training data: $((1,1), 1)$, $((-1,-1), 1)$,
$((1,-1), 2)$, $((-1,1), 2)$



- $\|x - (1,1)\| \leq \min [\, \|x - (1,-1)\|, \|x - (-1,-1)\|, \|x - (-1,1)\| \,]$

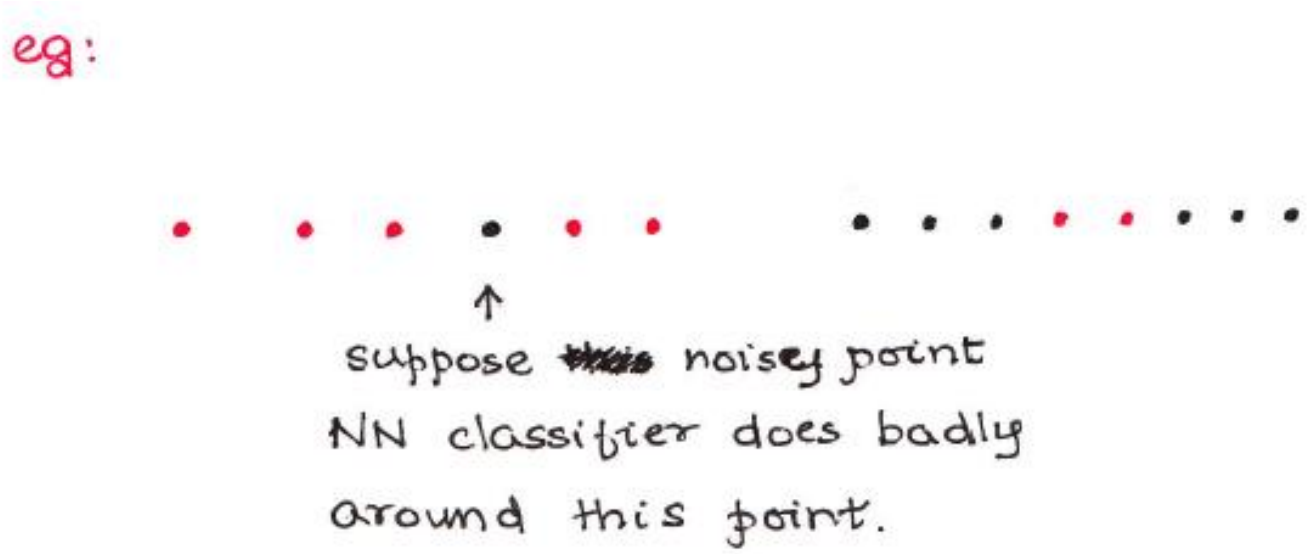
(Equation represents all vectors x which are closer to $(1,1)$ than any other data point.)

Example 4


Training data: ((0,0), 1), ((4,0), 2), ((1,3), 3)
Output label 3
Decision boundary
(0,0)
Output label 1
(4,0)
Output label 2

When does NN work?

- Works well away from the decision boundary
- Not so well at the boundary
- Also does not work well when data is noisy



- How can we make NN more robust to noise?

K-Nearest Neighbor Classifier

- Given a set of training examples $(x_i, y_i): i = 1,2,..N$, and a test example x

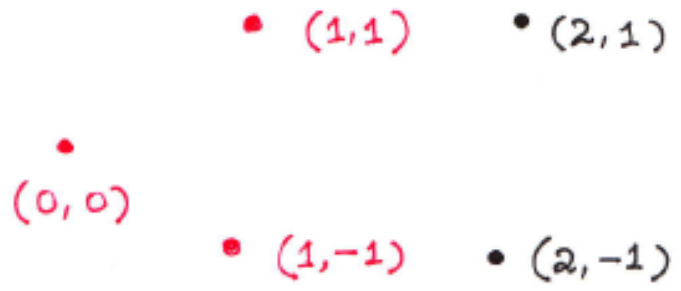
- **Prediction Rule:**
  - Find $j_1, j_2, \ldots, j_k$, the indices of k points closest to x in the training data
  - Output the majority label among $y_{j1}, y_{j2}, \ldots, y_{jk}$
  - Majority label == one that occurs most often
  - If there is tie, resolve uniformly at random

Example 1: 3-NN

Training data:

((0, 0), 0) ((1, 1), 0) ((1, -1), 0)

((2, 1), 1) ((2, -1), 1)



Test points: (1, 0).

dist ((1, 0), (0, 0)) = 1 dist ((1, 0), (2, 1)) = $\sqrt{2}$

dist ((1, 0), (1, 1)) = 1 dist ((1, 0), (2, -1)) = $\sqrt{2}$

dist ((1, 0), (1, -1)) = 1

closest 3 points: (0, 0), (1, 1), (1, -1)

Their labels : 0 0 0

So output = 0.

Example 1: 3-NN

Test point: $(2, 0.5)$

dist $((2, 0.5), (0, 0)) = \frac{\sqrt{17}}{2}$

dist $((2, 0.5), (1, 1)) = \frac{\sqrt{5}}{2}$

dist $((2, 0.5), (1, -1)) = \frac{\sqrt{13}}{2}$

dist $((2, 0.5), (2, 1)) = \frac{1}{2}$

dist $((2, 0.5), (2, -1)) = \frac{5}{2}$

Closest 3 points: $(2, 1)$, $(1, 1)$, $(2, -1)$

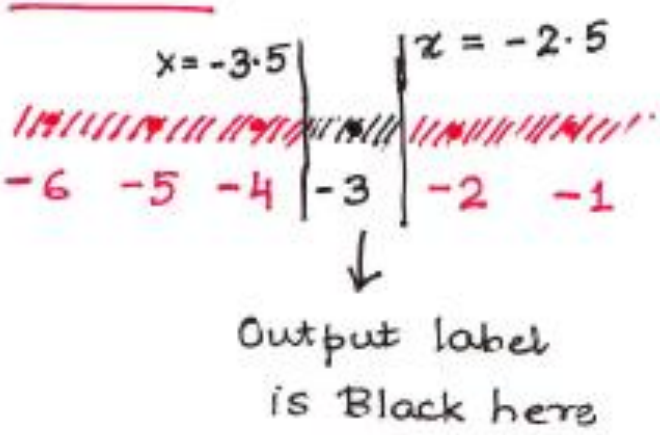
Labels: 1, 0, 1

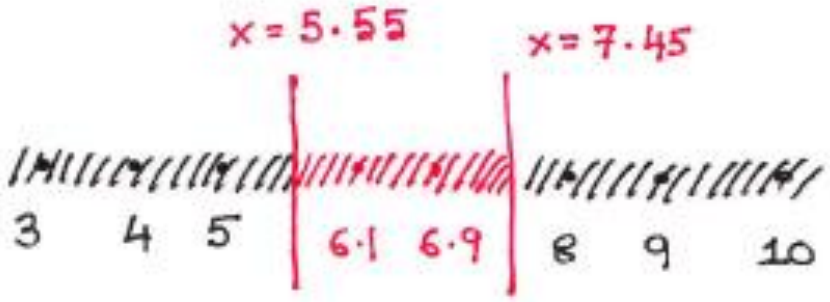
Majority: 1 = output label.

Example 2

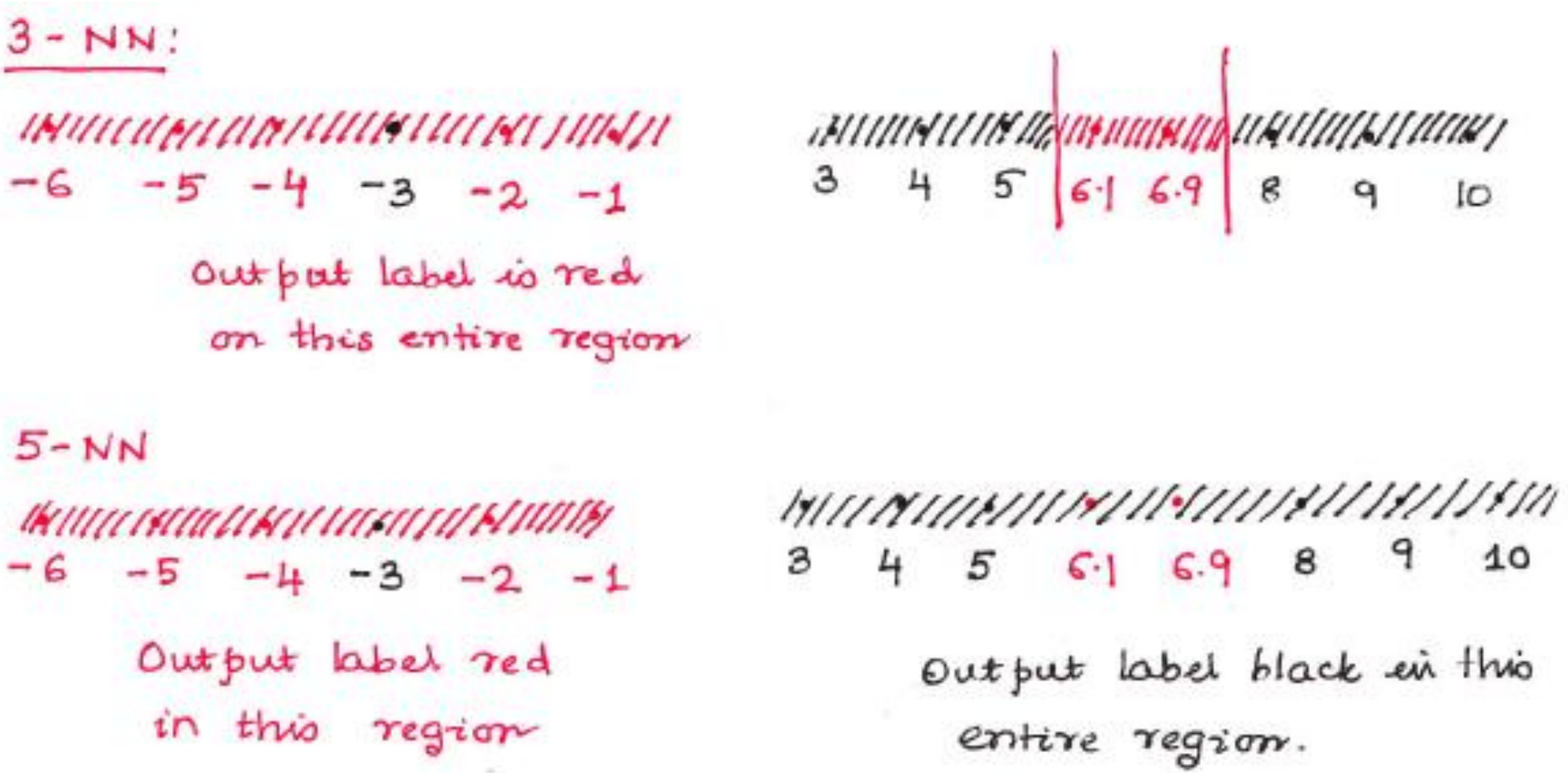
Suppose the points at -3 and 6·1 and 6·9 are noisy.

1-NN:





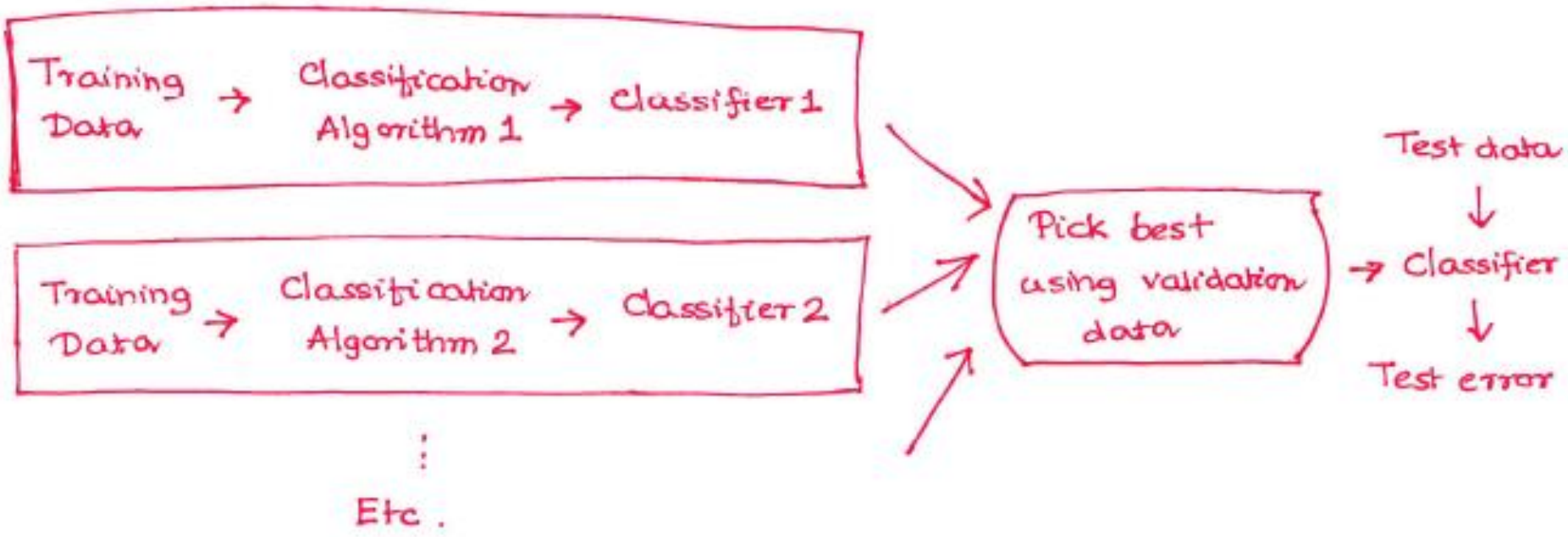
Example 2



- How to choose “K” for K-NN classifier?

Choosing "K" via Cross-Validation

1. Split data into training set and validation set.
2. Train classifier on training set for $k = 1, 3, 5, \ldots$.
3. Evaluate the error of each classifier trained on validation set and pick the one with the lowest error.



Other design choices

- Distance measure?
  - Most common is Euclidean distance
  - Others used too

- How to find NNs?
  - In 1D, binary search will do -- $O(\log n)$
  - In higher-dimensional space, advanced data structures such as Locality Sensitive Hashing (LSH) is used
  - LSH: Hash the data points such that closer points are mapped to the same buckets with very high probability

Summary

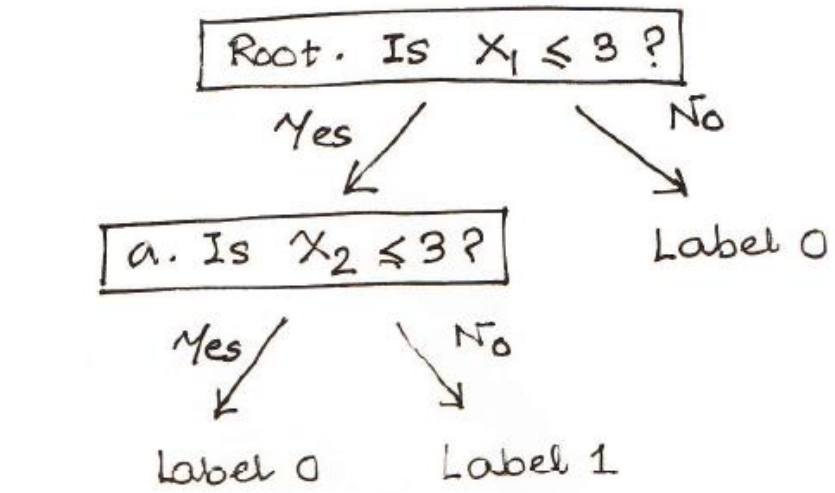
- K-NN is a non-parametric classifier
  - no training needed! Just store all the training data ☺
  - simple and easy to implement
  - flexible hypothesis space: infinitely complex

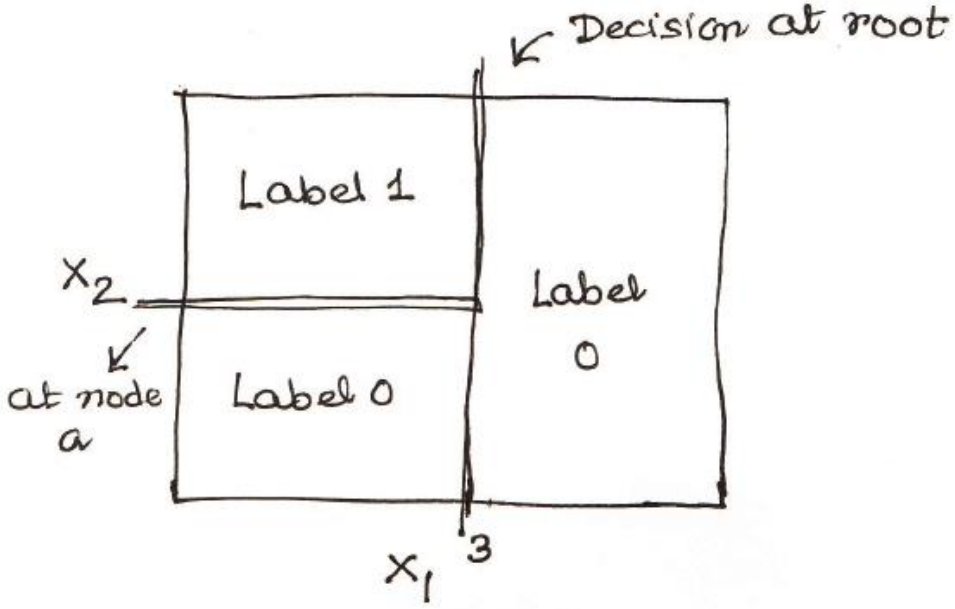
- Disadvantages
  - Classification time is high (NN computation!)
  - Space requirement is high
  - Doesn't work very well in high dimensions

Decision Tree Classifier

Decision Tree: Examples

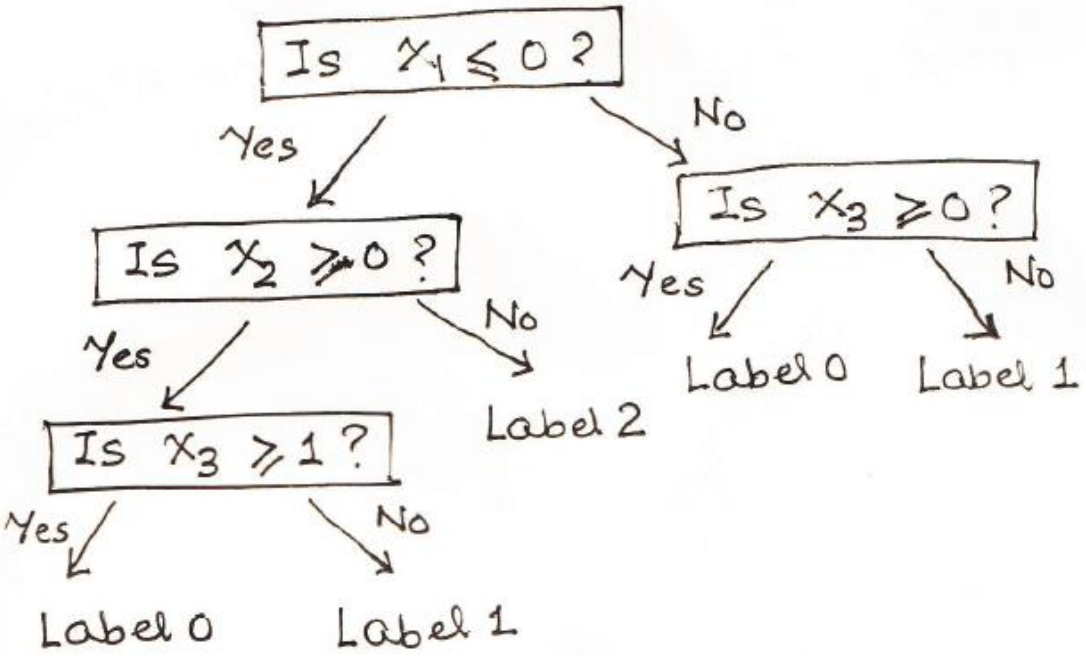
Example 1: Features X_1, X_2





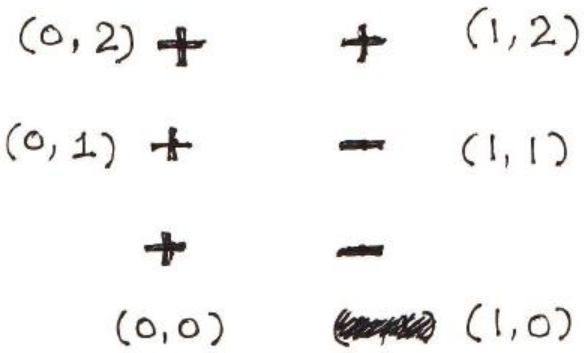
Decision Tree: Examples

Example 2: Features X_1, X_2, X_3

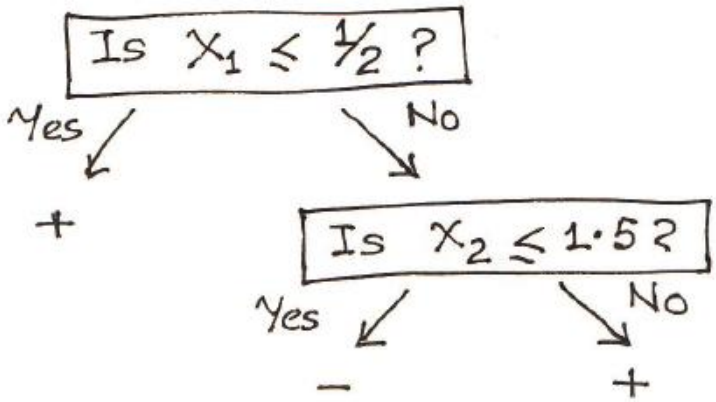


- Each node is based on a feature
- Each node: a decision is made based on the value of this feature
- Each leaf corresponds to a label (same label may appear on many leaves)

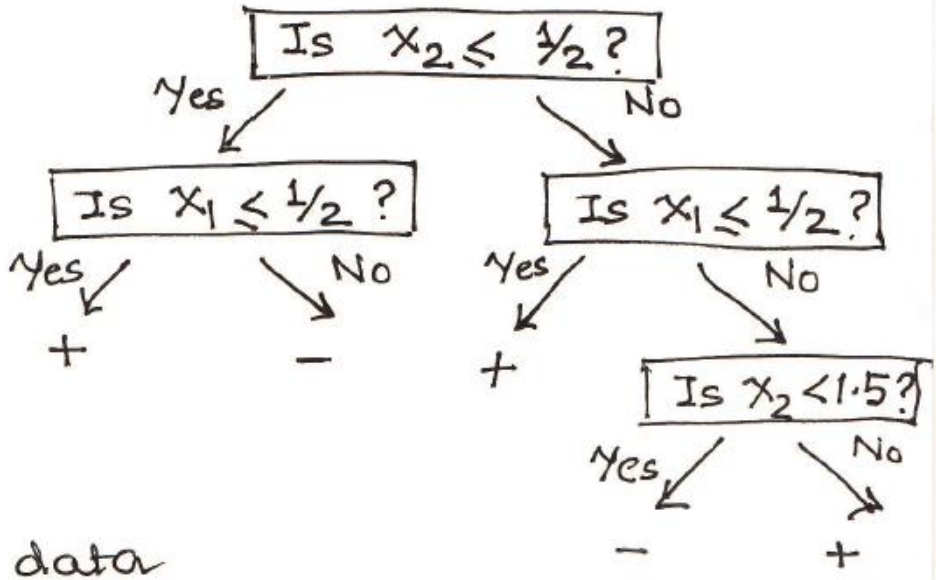
Decision Tree: Examples



Tree 1:



Tree 2:



- Both trees classify the training data correctly; but which one is better?

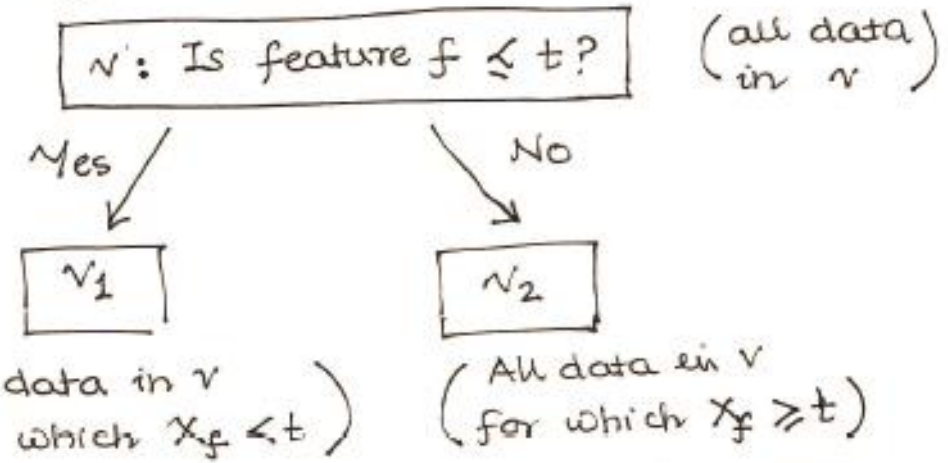
Decision Tree: Examples

- Answer: The simpler one. (here, tree 1)
- In general, finding the smallest decision tree to fit a training data set is computationally difficult.
- A commonly used algorithm that proceeds greedily is the ID3 Decision Tree algorithm.
- Convention: Rule at each node of the form:
  "Is $x_f \leq t$?"

- **Occam's Razor:**
  - Pick the simplest hypothesis that explains the data
  - Simpler solutions generalize well

ID3 Decision Tree Learning Algorithm

1. Initially, Whole training data is at root.
2. While there is an impure node:
   (a) Pick any impure node v
   (b) Pick a feature f and threshold t along which to "split" the data at v. Done according to "Splitting Rule" (to be described later)
   (c) Modify tree as:



If any of v_1 or v_2 is pure (i.e. has data of only one label), then ~~predict~~ make it a leaf that predicts this label.

Impure Node: Node with data points with multiple labels

Each node in the algorithm corresponds to a subset of the training data.

ID3 Algorithm: Running Example

Notes:

1. Convention: Use a threshold "in the middle" of two values
   eg. Node has data points: (0,0), (0,1), (1,1), and if we split along feature 1, use threshold 1/2.
2. More discussion of splitting rule coming up.

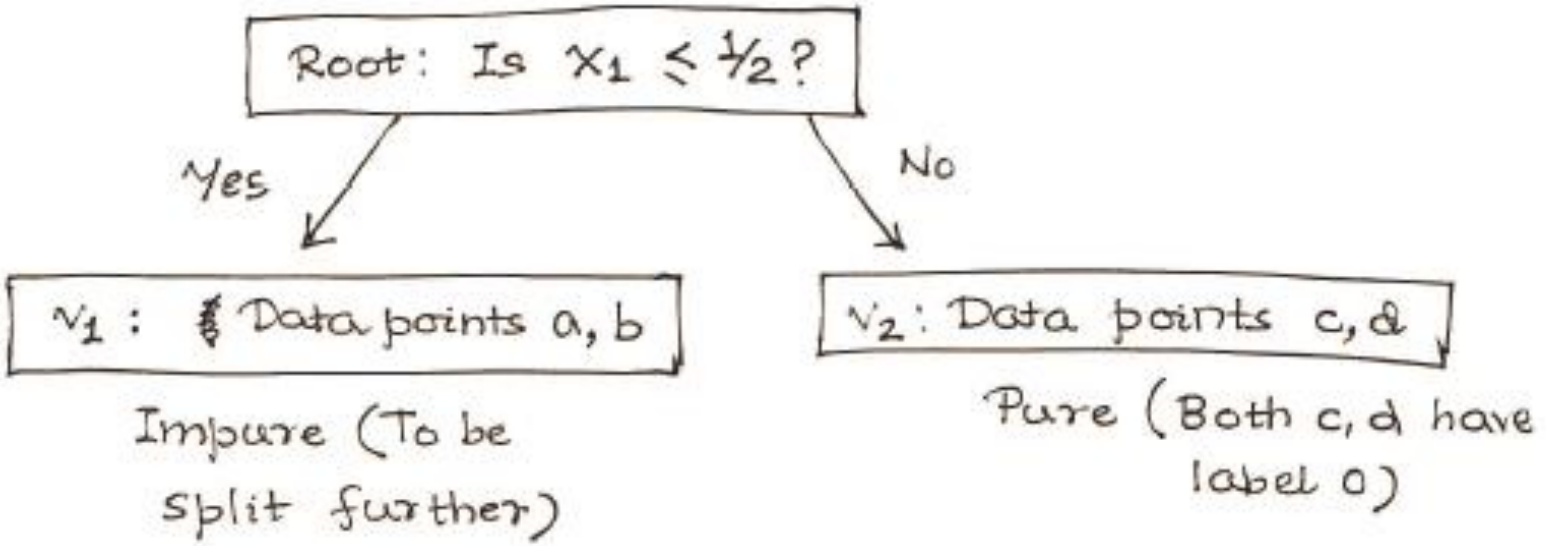
Example: Training data: ((0,0), 1), ((0,1), 0), ((1,0), 0), ((1,1), 0)
(a) (b) (c) (d)

1. Initially: Root has (a,b,c,d)

Root: (a,b,c,d) Impure.

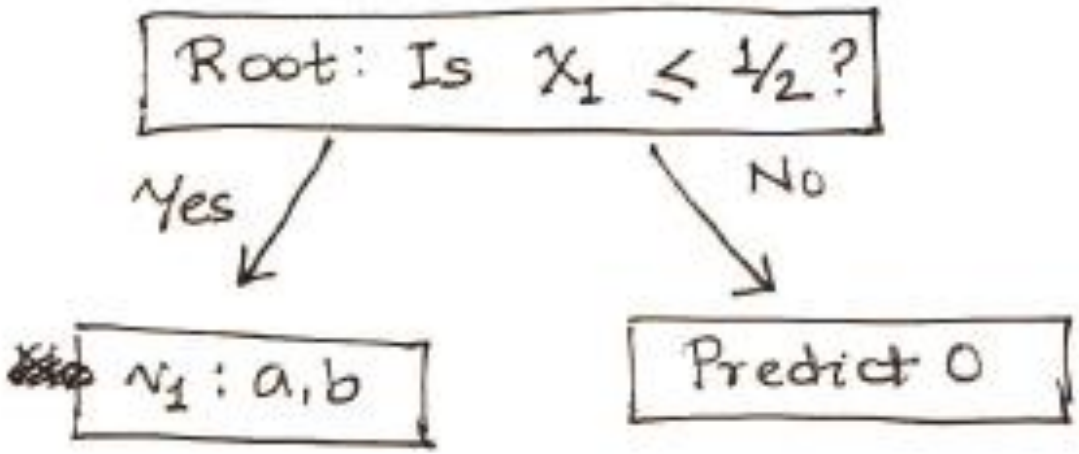
ID3 Algorithm: Running Example

2. Suppose we pick x_1, threshold $1/2$ (thru Selection Rule).



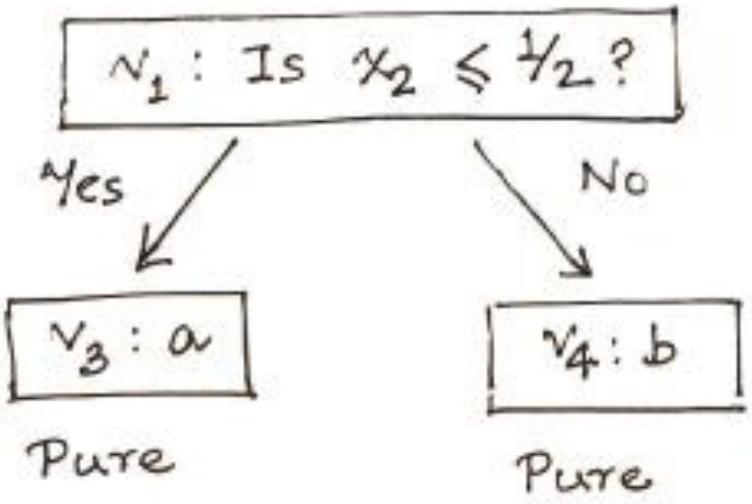
ID3 Algorithm: Running Example

3.



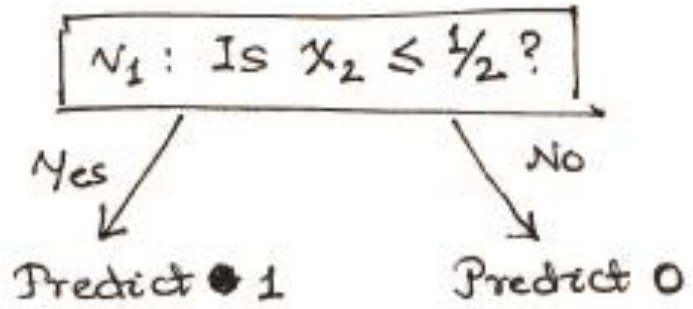
ID3 Algorithm: Running Example

4. Suppose we pick feature x_2, threshold $1/2$ (to split v_1)



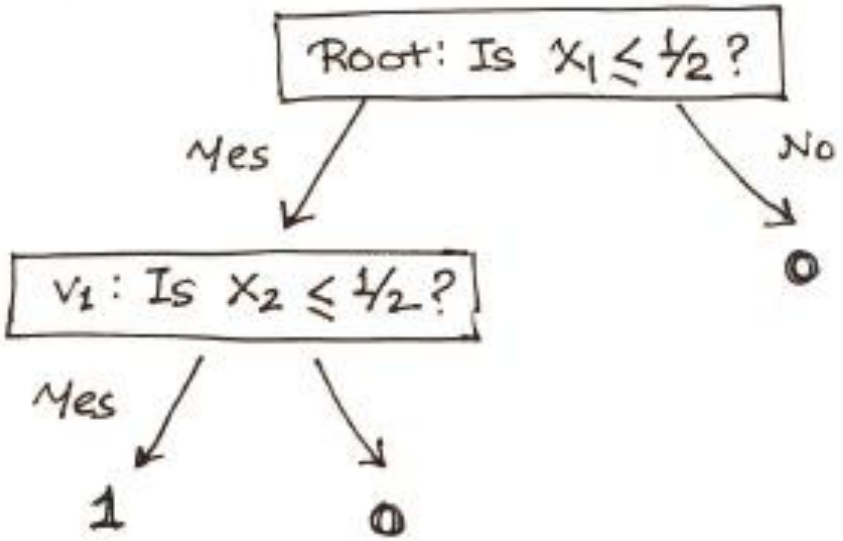
ID3 Algorithm: Running Example

5.



All nodes are pure at this point, so we are done.

Complete Decision Tree:



Splitting Rule

Splitting Rule: How to choose a feature and threshold along which to split a node?

In ID3 Decision Trees, we choose one that reduces uncertainty the most.

How to measure uncertainty? Through a notion in information theory, called entropy

Entropy: Let X be a random variable that takes values $v_1, .., v_K$ with probabilities $p_1, .., p_K$. Then, entropy of X, denoted by $H(X)$ is defined as:

$$H(X) = -p_1 \log p_1 - p_2 \log p_2 - \cdots - p_K \log p_K$$

(Note: Use the convention that $0 \cdot \log 0 = 0$)

(Optional) Entropy: Examples

Some Entropy Values:

1. X: 0/1 random variable (r.v). $Pr(X=1) = 1/2$.

$p_0 = Pr(X=0) = 1/2$

$p_1 = Pr(X=1) = 1/2$

$$H(X) = -p_0 \log p_0 - p_1 \log p_1 = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = \log 2$$

$= 0.69$ (Usually, we use natural logarithm for these calculations.)

2. X: 0/1 r.v. $Pr(X=1) = 0$.

$p_0 = 1, \quad p_1 = 0.$

$$H(X) = -1\log 1 - 0\log 0 = 0$$

3. X is a r.v. which takes values $1, 2, .., k$ w.p. $\frac{1}{k}$ each.

$$H(X) = -\frac{1}{k}\log\frac{1}{k} - \frac{1}{k}\log\frac{1}{k} - \cdots - \frac{1}{k}\log\frac{1}{k}$$

$$= -k \cdot \frac{1}{k}\log\frac{1}{k} = \log k.$$

For 0/1 r.v, closer $Pr(X=0)$ is to a $1/2$, higher is the entropy.

Entropy: Properties

Properties of Entropy:

1. $H(X)$ does not depend on the exact values taken by X, only on the probabilities of distinct values.

2. $H(X) \geq 0$ (always). Why?

   $H(X) = -\sum_{i=1}^{k} p_i \log p_i$, where $\log p_i \leq 0$, as $p_i \leq 1$.

3. If X takes k values, $H(X) \leq \log k$. This maximum is achieved only when each value occurs w.p. $1/k$.

Conditional Entropy

Conditional Entropy:

Let X, Z be two r.v.s. The conditional entropy of X given Z is defined as:

$$H(X|Z) = \sum_{z} \Pr(Z=z)\, H(X|Z=z)$$

(Intuitively, average entropy of X given that we know Z).

Conditional Entropy: Examples

Joint Distribution of X, Z:

Z \ X	0	1
0	$\frac{1}{3}$	$\frac{1}{3}$
1	$\frac{1}{3}$	0

$\Pr(Z=0) = \frac{2}{3}$ $\Pr(Z=1) = \frac{1}{3}$

For any x,

$$\Pr(X=x \mid Z=0) = \frac{\Pr(X=x, Z=0)}{2/3}$$

$$\Pr(X=x \mid Z=1) = \frac{\Pr(X=x, Z=1)}{1/3}$$

So: Conditional distributions are:

$X|Z=0$: $\Pr(X=0|Z=0) = 1/2$

$\Pr(X=1|Z=0) = 1/2$

$X|Z=1$: $\Pr(X=0|Z=1) = 1$

$\Pr(X=1|Z=1) = 0$

$$H(X|Z=0) = -\tfrac{1}{2}\log\tfrac{1}{2} - \tfrac{1}{2}\log\tfrac{1}{2} = \log 2$$

$$H(X|Z=1) = -1\log 1 - 0\log 0 = 0$$

So: $H(X|Z) = \frac{2}{3}\cdot\log 2 + \frac{1}{3}\cdot 0 = \frac{2}{3}\log 2.$

Conditional Entropy: Properties

1. Suppose $X = Z$. What is $H(X|Z)$?

$$H(X|Z) = \sum_z \Pr(Z=z)\, H(X|Z=z).$$

Now, given that $Z=z$, we know that $X = z$ w.p. 1.

So, $H(X|Z=z) = 0$.

Thus, $H(X|Z) = \sum_z \Pr(Z=z) \cdot 0 = 0$.

(This may hold even if $H(X)$ is very large!)

2. Suppose X, Z are independent. What is $H(X|Z)$?

Since X, Z are independent, $\Pr(X=x \mid Z=z)$ for any z and x, is equal to $\Pr(X=x)$.

i.e. $X|Z=z$ has exactly the same distribution as X.

So. $H(X|Z=z) = H(X)$

$$H(X|Z) = \sum_z \Pr(Z=z)\, H(X|Z=z) = H(X) \sum_z \Pr(Z=z) = H(X)$$

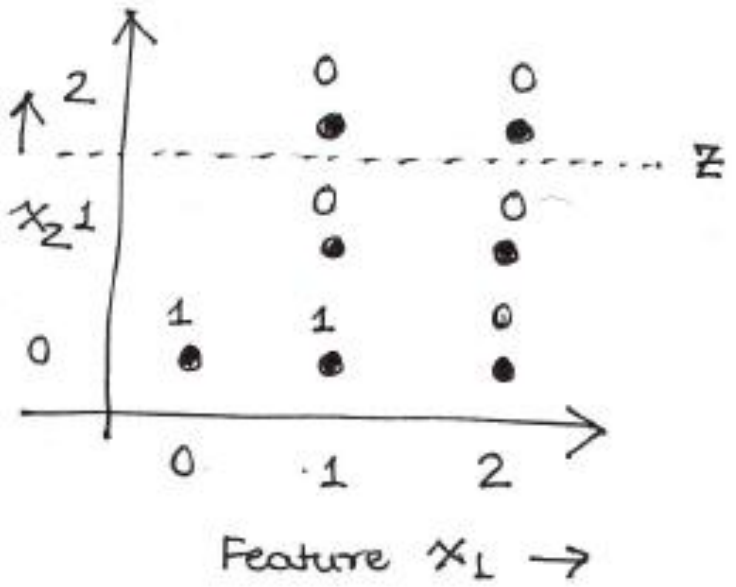
Information Gain

Information Gain$(Z) = H(X) - H(X|Z)$

(essentially, how much entropy of X is reduced because we know Z).

It can be shown that information gain (IG) is ≥ 0 (always)

Information Gain: Example



Suppose Figure shows the training data at a node v. ●

So, v has 5 label 0, 2 label 1 data points, and

$$H(X) = -\frac{5}{7}\log\frac{5}{7} - \frac{2}{7}\log\frac{2}{7}$$

Suppose we would like to split along Z. ($X_1 < 1.5$ or not).

Suppose $Z = 0$ corresponds to $X_1 \geq 1.5$ (1)

$Z = 1$ " " $X_1 < 1.5$ (2)

Then, $X|Z=0$ has 2 label 0s, 0 label 1s.

$Pr(X=0|Z=0) = 1$, $Pr(X=1|Z=0) = 0$.

So, $H(X|Z=0) = -1\log 1 - 0\log 0 = 0$.

Information Gain: Example

$X|Z=1$ has 3 label 0's and 2 label 1's.

$\Pr(X=0|Z=1) = 3/5$, $\Pr(X=1|Z=1) = 2/5$.

$$H(X|Z=1) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}$$

Also:

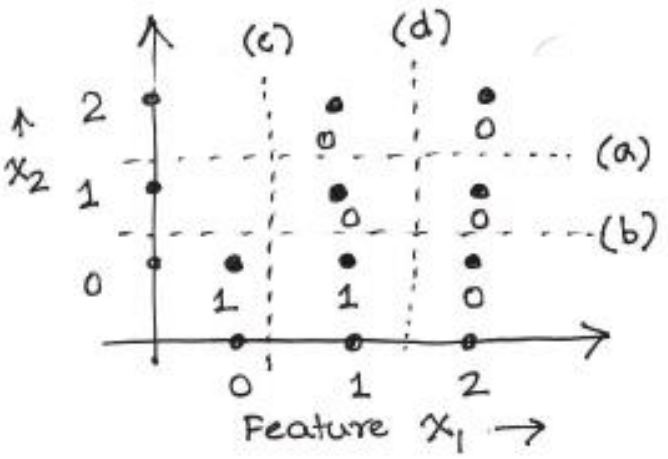
$\Pr(Z=0) = 2/7$ (as 2 out of the total 7 points in X will lie on this branch)

and: $\Pr(Z=1) = 5/7$ (as 5 "-" and 7 "-" and "+")

so: $H(X|Z) = \Pr(Z=0)H(X|Z=0) + \Pr(Z=1)H(X|Z=1)$

$= 0.48$

How to Split a node: Example 1



4 possible (feature, threshold) pairs to split along.

Call them (a), (b), (c), (d).

Initially: (5 0, 2 1)

(5 label 0's, 2 label 1's)

(a): Results: (2 0, 0 1); (3 0, 2 1)

$H(X|Z) = 0.48$

(b): Results: (4 0, 0 1); (1 0, 2 1)

$H(X|Z) = 0.27$

(c): Results: (5 0, 1 1); (0 0, 1 1)

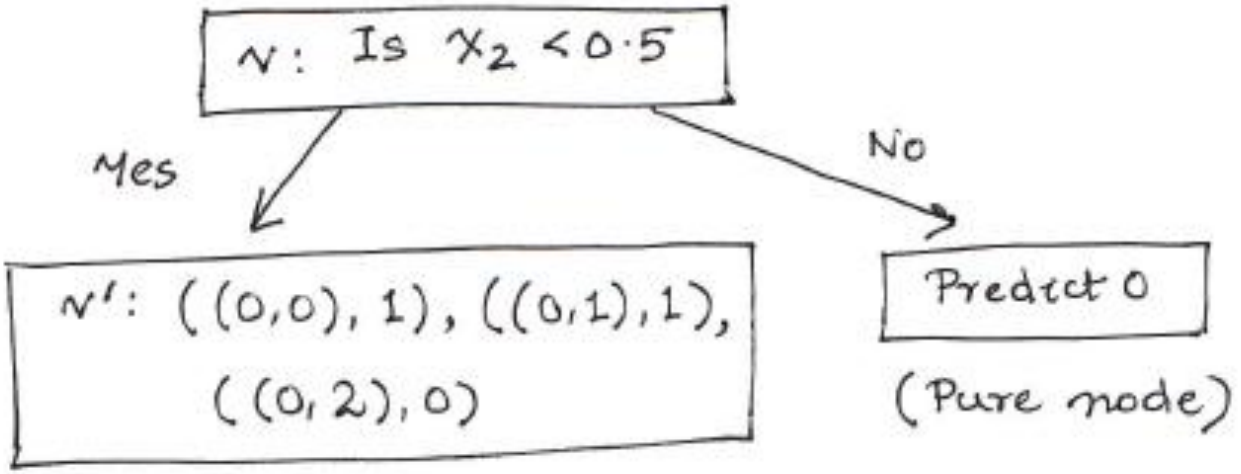
$H(X|Z) = 0.38$

(d): Results: (2 0, 2 1); (3 0, 0 1)

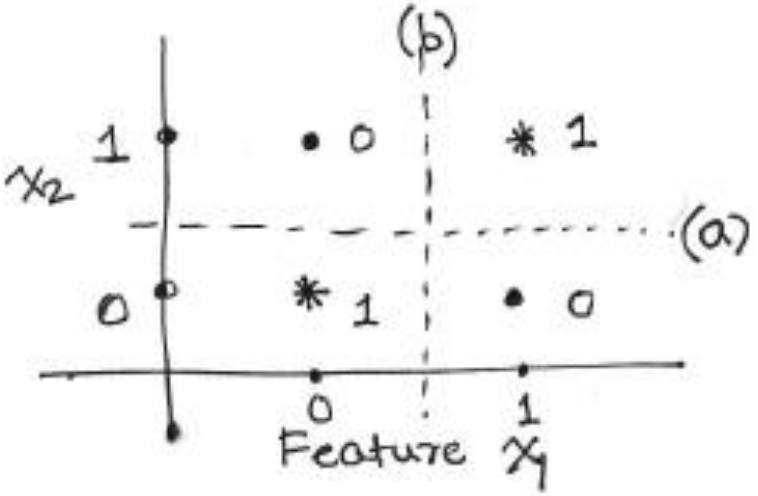
$H(X|Z) = 0.39$

How to Split a node: Example 1

So, ID3 will select (b) to get:



How to Split a node: Example 2



Initially; ●(2 0, 2 1)

$H(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = \log 2$

$= 0.69.$

2 possible ways to split: (a) and (b).

For (a); Result: (1 0, 1 1), (1 0, 1 1)

$H(X|Z) = 0.69$

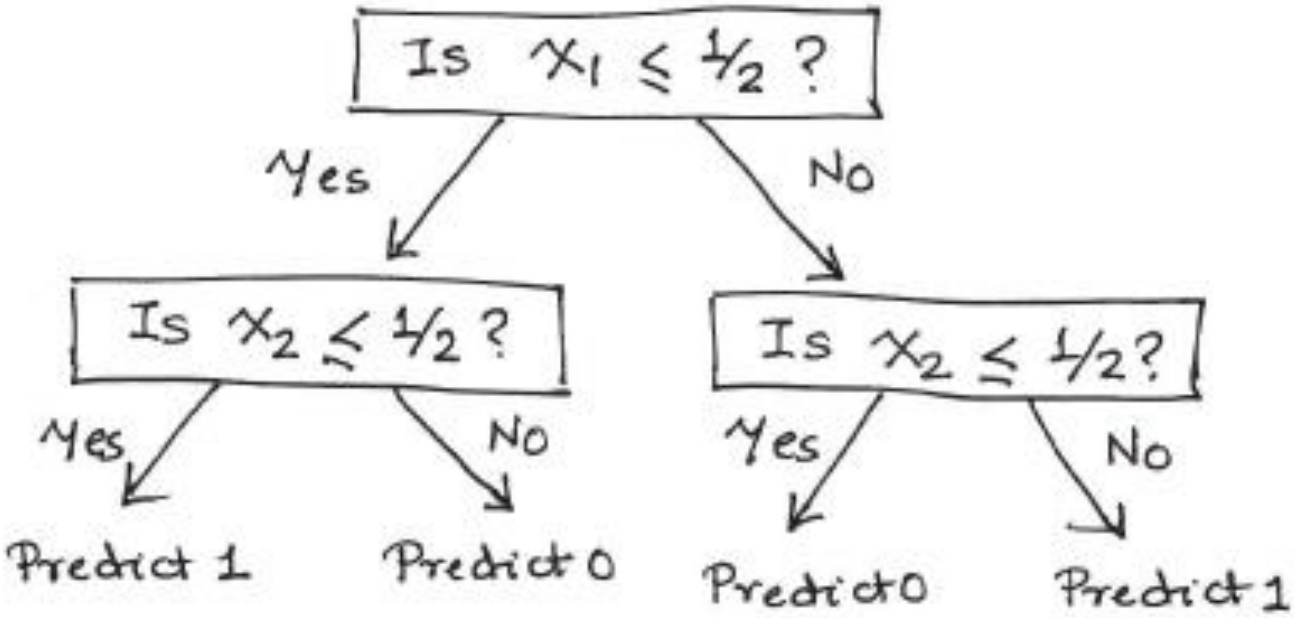
For (b): Result: (1 0, 1 1); (1 0, 1 1)

$H(X|Z) = 0.69$

So, $IG(Z) = 0$.

How to Split a node: Example 2

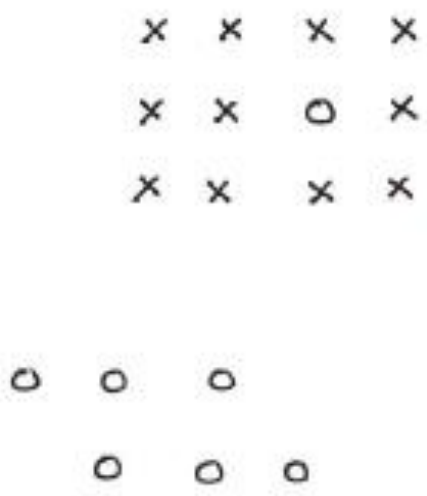
However, we shouldn't stop constructing the tree even if IG = 0. In this case, correct tree is:



Over-fitting: Example

Stopping Rule : Stop when every node is pure.
But this rule may overfit the data

Example:



The o may be an outlier or some noisy data.

If we build a decision tree that stops when all nodes are pure, then we may overfit the training data.

What is Over-fitting?

Data comes from some underlying true distribution D.
Training data, test data → all samples from D.

Training error of a classifier:

$$\hat{err}(h) = \frac{1}{n} \sum_{i=1}^{n} 1(h(x_i) \neq y_i)$$

[$1(\cdot)$ means an indicator variable which is 1 when the condition is true, 0 o/w.

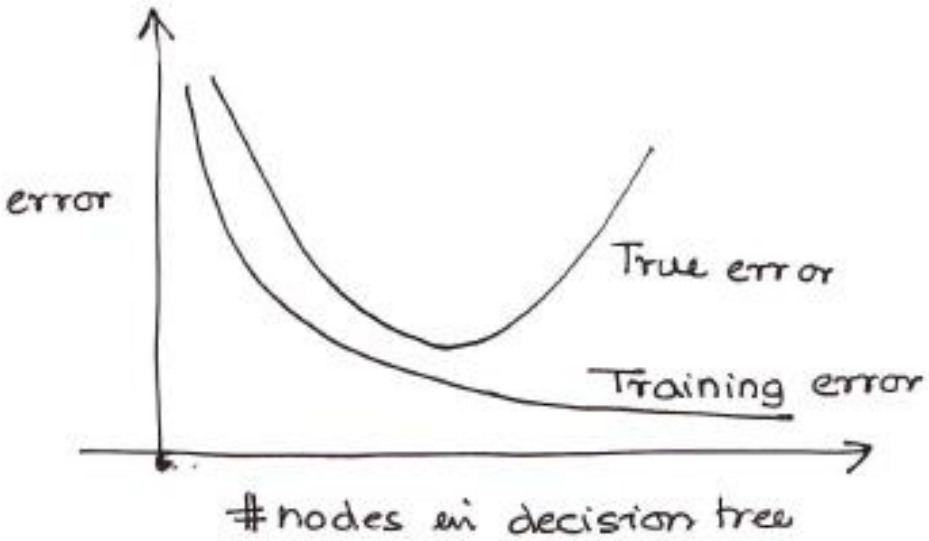
True error of a classifier:

$$err(h) = \Pr_{(x,y) \sim D}(h(x) \neq y)$$

For a fixed classifier, with more data, training error should approach true error.

Over-fitting in Decision Trees

Overfitting happens when:



As we make the tree more and more complicated, training error decreases. At some point, true error stops improving and may get worse.

True data distribution has some structure, which we would like to capture. Training data captures some of this structure, but may have some "chance" structure of its own, which we should not model into a classifier.

Pruning to avoid Over-fitting in Decision Trees

1. Split the training data into training set S and validation set V.
2. Build ID3 tree T using training sets
3. Prune using V:

   Repeat:

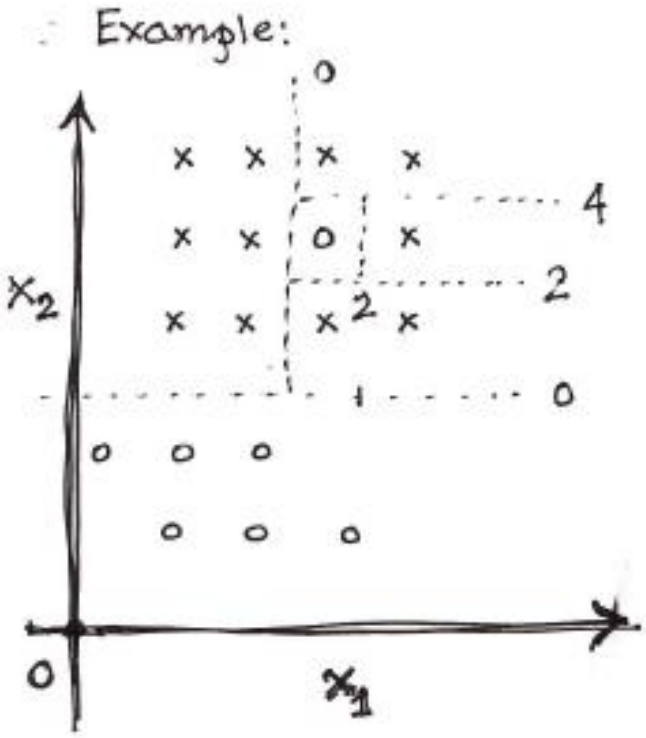
   For each node v in T:

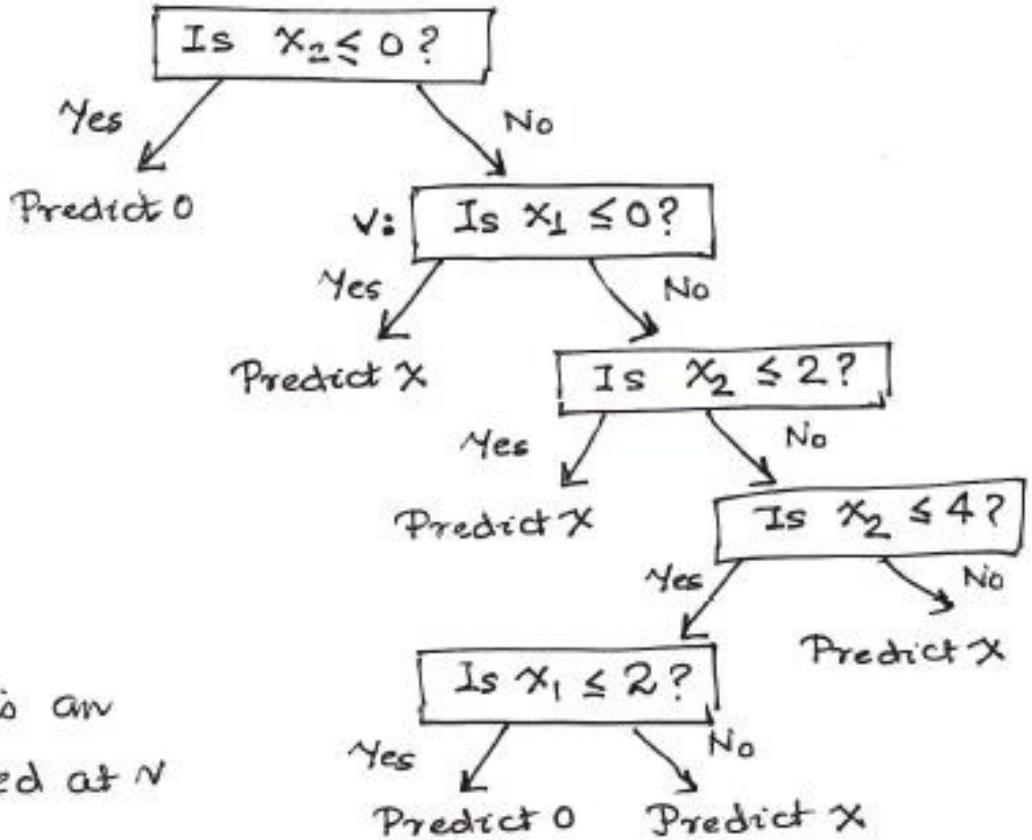
   Replace subtree rooted at v by single node that predicts majority label in v to get tree T'

   If error(T') on V $\leq$ error(T) on V, then T = T'

Continue till there is no such node v.

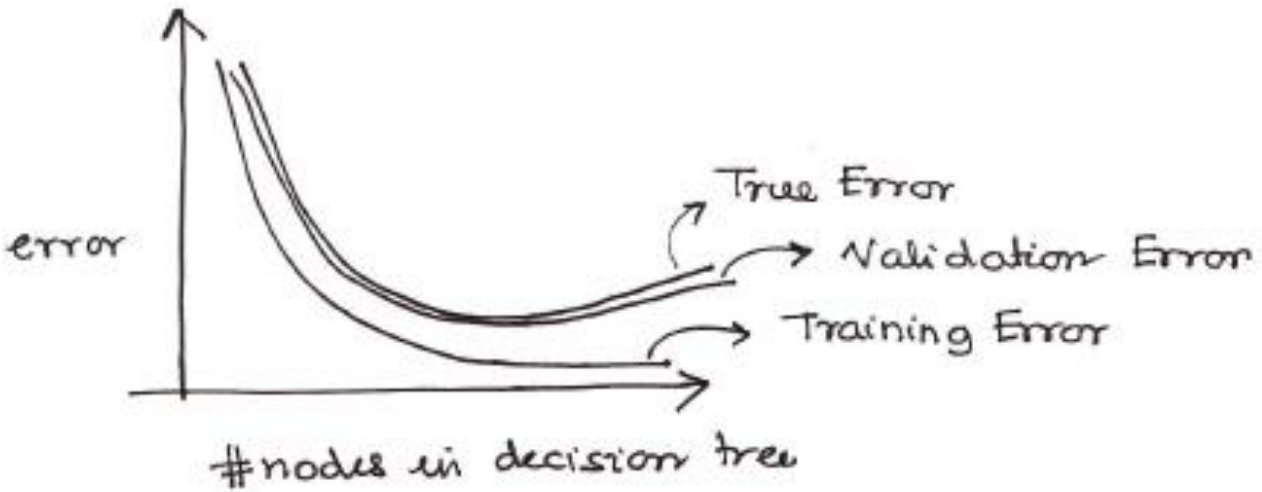
Pruning: Example





If the 0 point in between is an error, then the subtree rooted at V will have higher validation error than predicting X at V.

Pruning



As tree grows more complicated, training error decreases. At some point, true error stops improving and may even get worse. This will be reflected in the validation error, provided we have enough validation data.

Bagging for Constructing Ensembles

What is Ensemble Learning?

In ensemble learning, the idea is to combine multiple classifiers into a single one. Ensemble learning usually works very well in practice.

Two methods (for this class):

1. Bagging
2. Boosting

Bagging

BAGGING: (Bootstrap AGGregatING)

1. Input: n labelled training examples $(x_1, y_1), .., (x_n, y_n)$

2. Algorithm:

   Repeat k times:

   (a) Select m samples out of n with replacement from the training set to get training set S_i

   (b) Train classifier h_i on S_i (usually, h_i's are the same type of classifier)

3. Output: classifiers $h_1, .., h_k$

Testing: Given test example x, output the majority of $h_1(x), h_2(x), .., h_k(x)$ (break ties at random as usual)

Choice Points in Bagging

1. How to pick k?

Higher k is better, but also increases training time, storage requirement and classification time. So pick a k which is feasible.

Choice Points in Bagging

2. How to pick m?

Popular choice for $m = n$.

But this is still very different from working with the entire training set!

$$\Pr(S_i = S) = \frac{n!}{n^n} \quad \left(\begin{array}{l} \#\text{ Ways of choosing } n \text{ samples} \\ \text{with replacement} = n^n \end{array} \right)$$

↦ Only $n!$ of these ways give you the entire training set!

↦ $\frac{n!}{n^n}$ = a very tiny number $\ll 2^{-n/2}$

For any (x_j, y_j), $\Pr\left((x_j, y_j) \text{ is not in } S_i\right) = \left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e}$ (for large n)

$1/e \approx 0.37$; so about 37% of the data is left out of S_i.