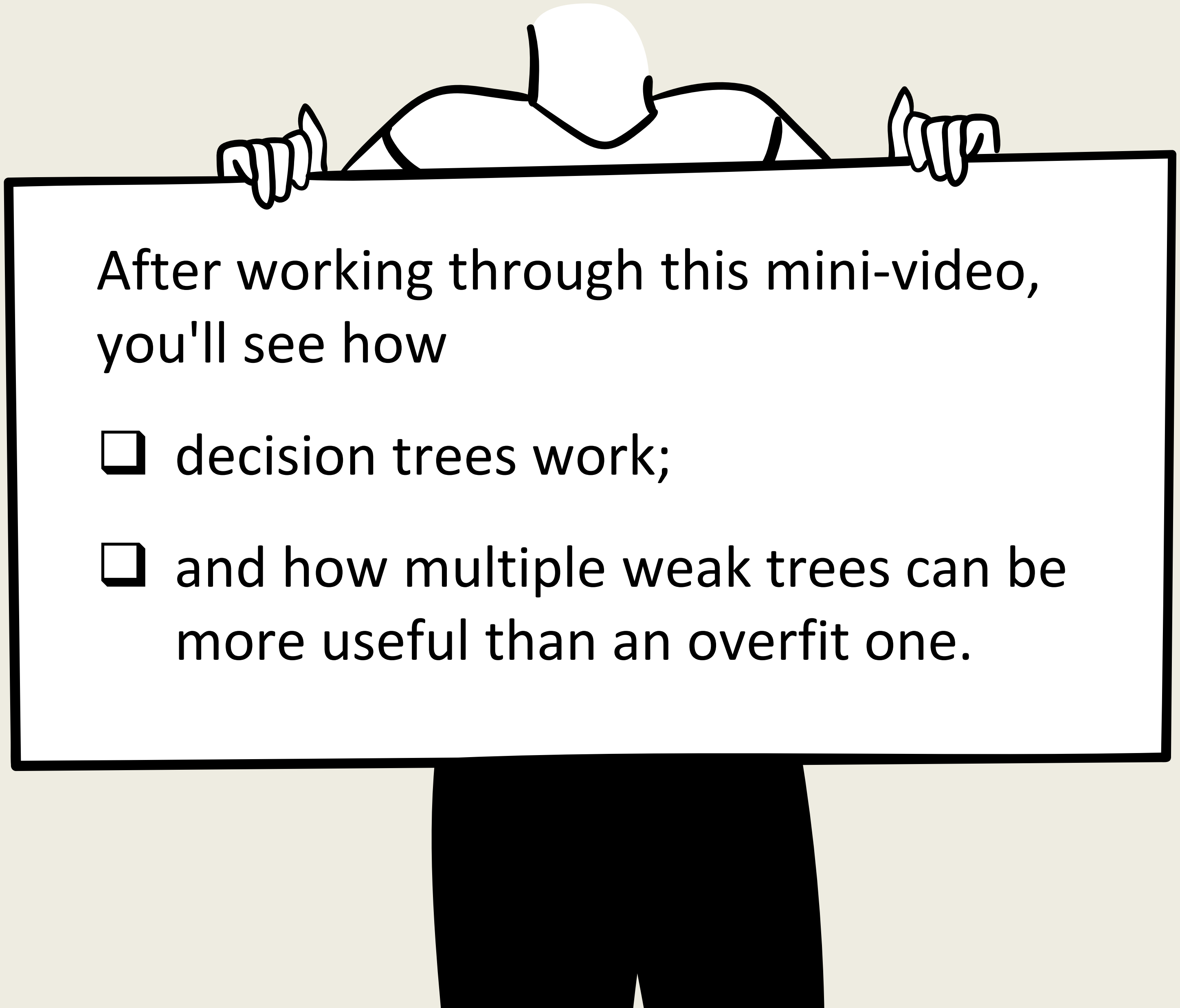
A tree-based model

MACHINE LEARNING

Dr. Temitayo Olugbade

UNIVERSITY OF SUSSEX

Learning outcome



After working through this mini-video, you'll see how

- ❑ decision trees work;
- ❑ and how multiple weak trees can be more useful than an overfit one.

Lecture outline

- ❑ Decision trees

- ❑ Ensemble learning

Recall from Week 1

1. The most basic element of **machine learning** is a **model** that learns from data.
2. The linear regression model is $f(\boldsymbol{x}) = \hat{\boldsymbol{y}} = \boldsymbol{x}\boldsymbol{w} + b$.
3. Training a ML model involves optimizing the model parameters based on a **loss function**.
4. An important goal in machine learning is **generalizability** to data not seen by the model during its training.
5. Achieving the goal of generalizability to unseen data is trade-off between underfitting (**weak learning**) & **overfitting**.
6. Overfitting can be addressed with **regularization** or **data augmentation**.

Decision trees

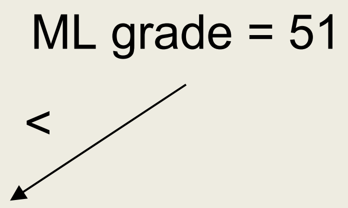
- ❑ **Decision trees**
- ❑ Ensemble learning

Toy data to illustrate decision trees

$\boldsymbol{x}$		y
Experience	**ML grade**	**Hiring decision**
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired
50	80	?

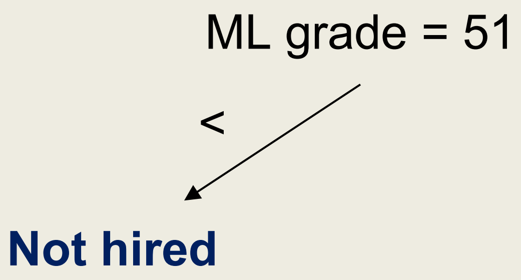
Decision tree creation example

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired



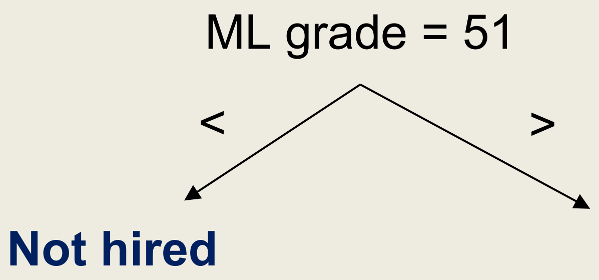
Decision tree creation example (2)

Experience	**ML grade**	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired



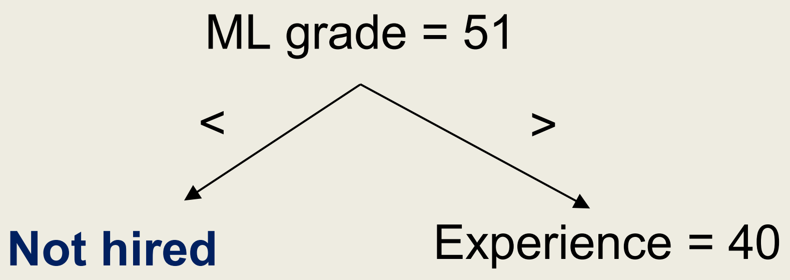
Decision tree creation example (3)

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired



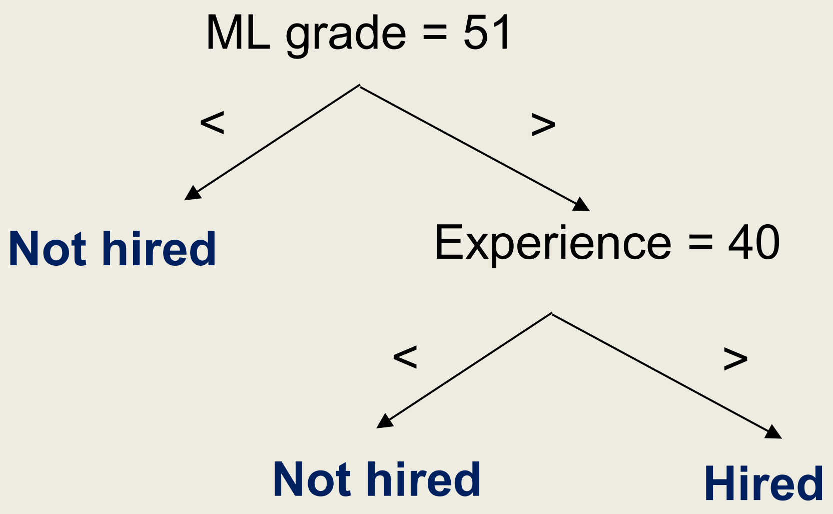
Decision tree creation example (4)

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired



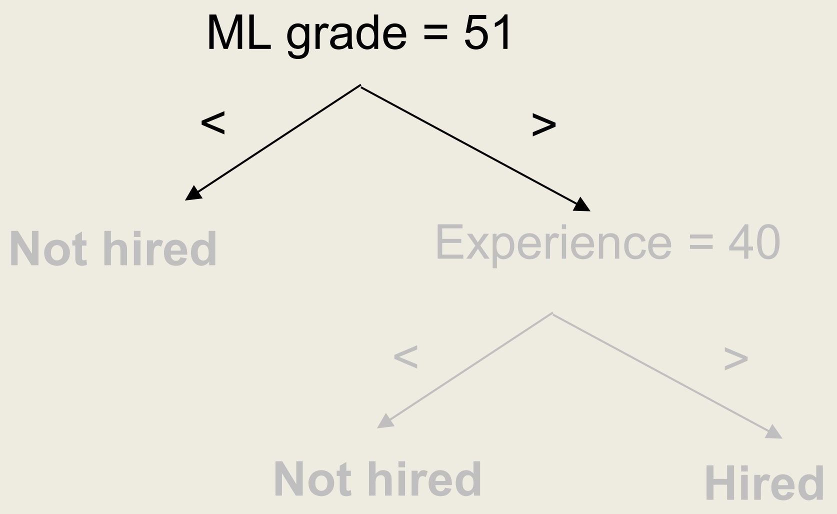
Decision tree creation example (5)

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

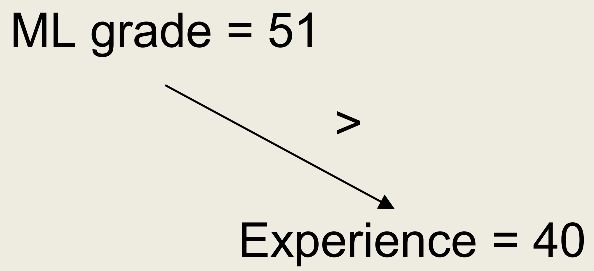


Making inference with the decision tree (1)

x		y
Experience	**ML grade**	**Hiring decision**
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired
50	80	?

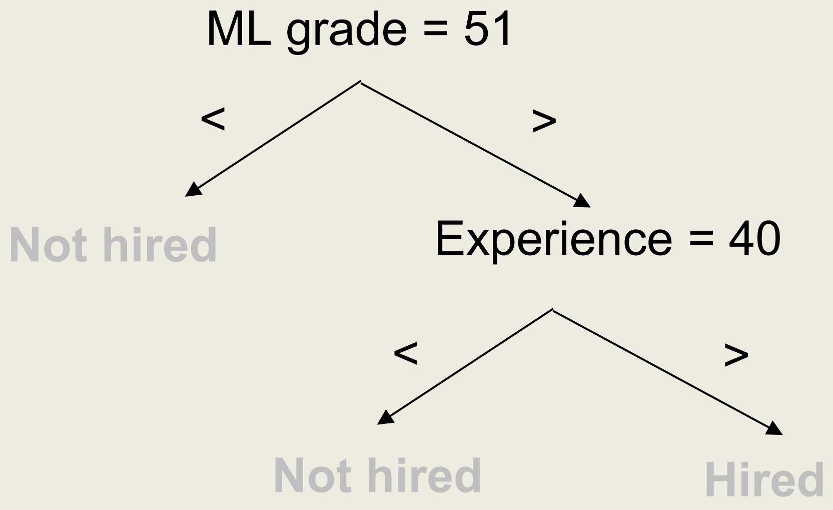


1. **Is ML grade < or > 51?**

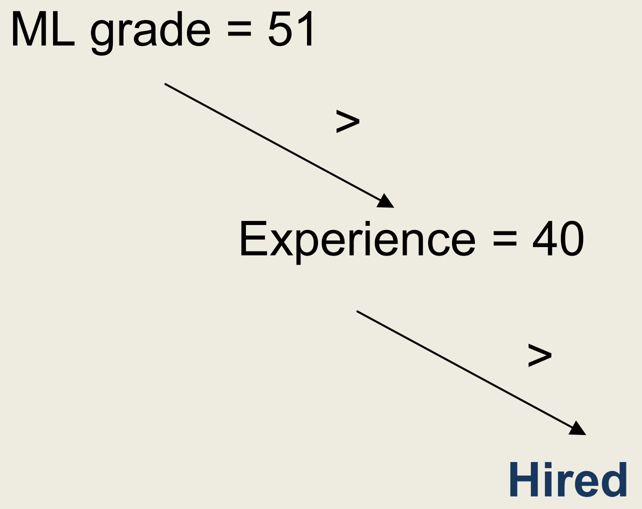


Making inference with the decision tree (2)

x		y
Experience	**ML grade**	**Hiring decision**
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired
50	80	?

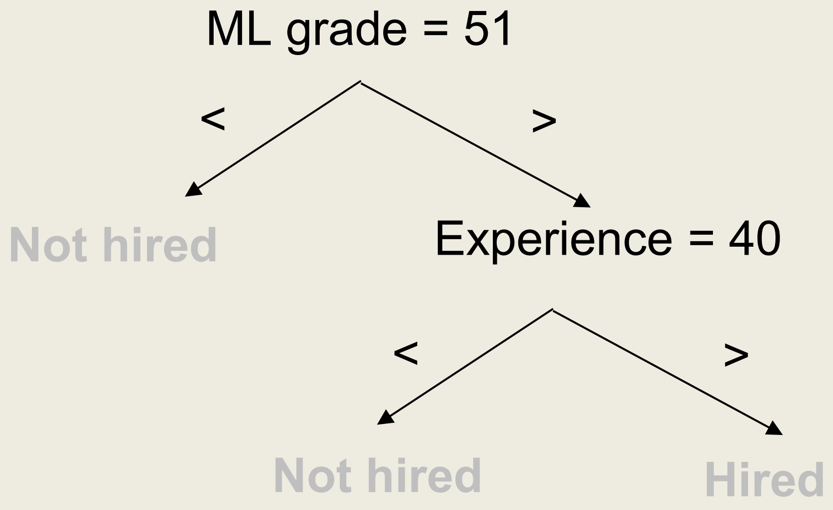


2. Is Experience < or > 40?



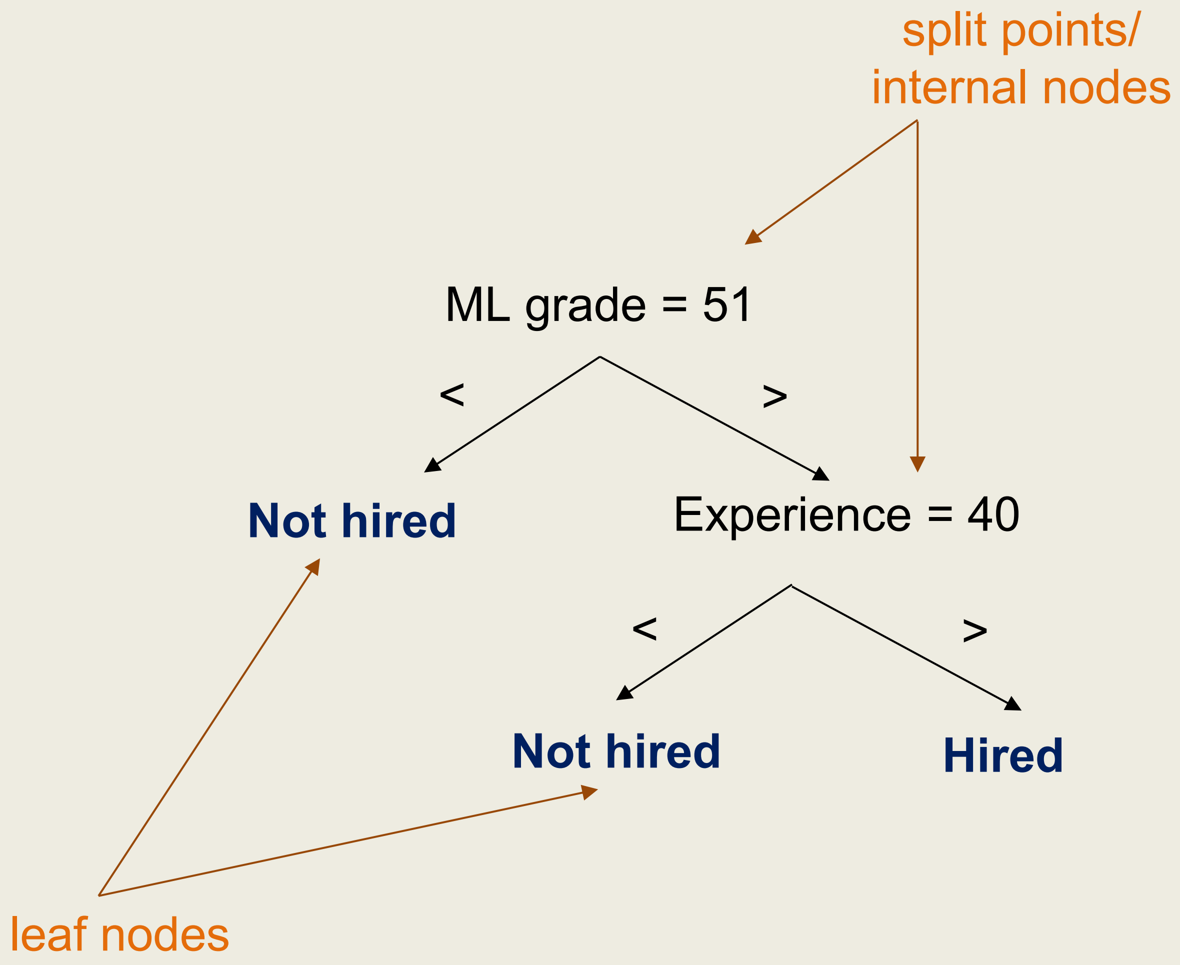
Making inference with the decision tree (3)

x		y
Experience	**ML grade**	**Hiring decision**
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired
50	80	?



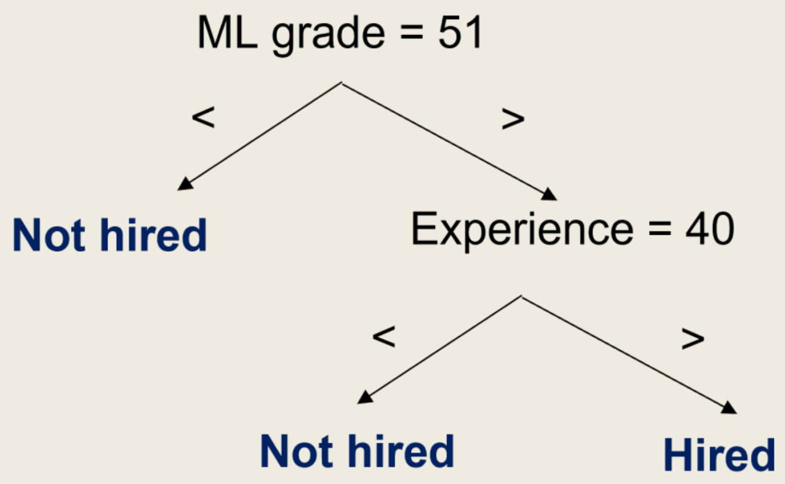
3. **Leaf node → 'Hired' decision**

Anatomy of a decision tree



Decision tree

- Works by repeatedly partitioning the data space into two



- Each split point is defined by a hyperplane that separates partitions in the data space

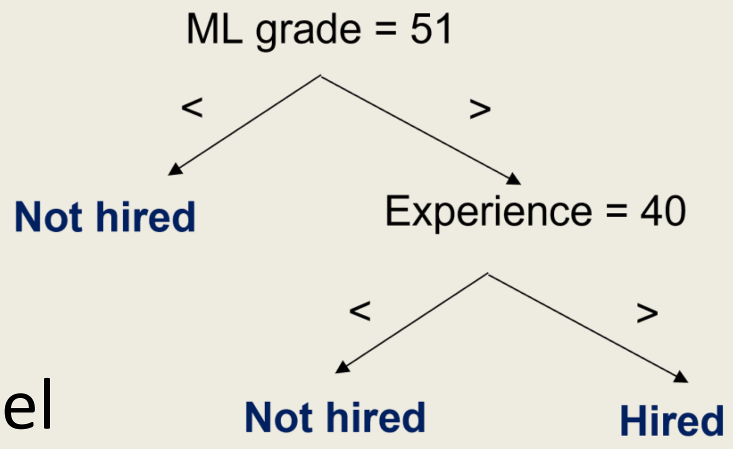
- Leaf nodes represent labels for data instances which share the same label region

- Can be used for both classification and regression

Knowing when to split

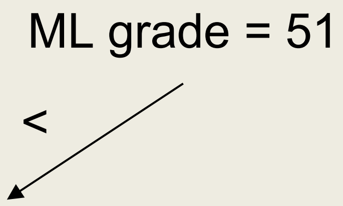
- Further split of a partition is based on **purity** of that partition m

- Purity is proportion of instances of the majority label



see math details in last slide pages

Purity example A



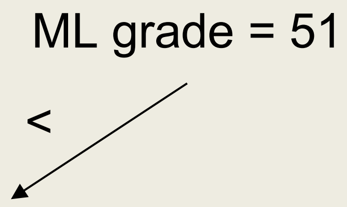
Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

Purity is proportion of instances of the majority label

$$purity = \frac{3}{3} = 1$$

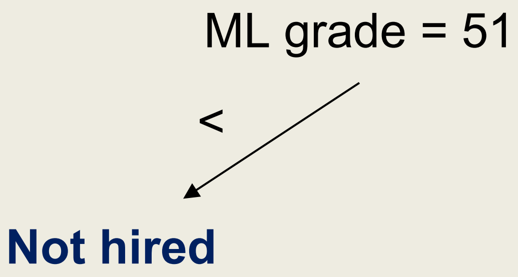
see math details on last slide pages

Purity example A (2)



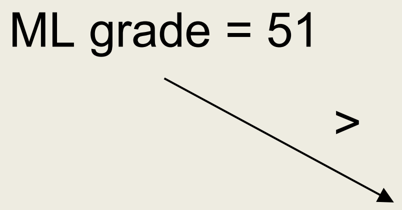
$$purity = 1$$

→ no need to split further



Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

Purity example B



Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

$$purity = \frac{3}{5} = 0.67$$

see math details on last slide pages

Purity example B (2)

$$\begin{tikzcd}\text{ML grade} = 51 \arrow[dr, ">"] & \\ & {}\end{tikzcd}$$

$purity < 1$

→ needs further split

$$\begin{tikzcd}\text{ML grade} = 51 \arrow[dr, ">"] & \\ & \text{Experience} = 40\end{tikzcd}$$

Experience	**ML grade**	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

Knowing how to split (1)

1. For a given feature, sort the values for that feature in the dataset

x	
Experience	**ML grade**
20	15
35	50
32	78
67	83
46	90
12	40
13	66
70	58

ML grade
15
40
50
58
66
78
83
90

Knowing how to split (2)

2. For each value in the sorted list:
   - Compute potential split points – Midpoint between each value and the next, if different

     27.5, 45, 54, 62, 72, 80.5, 86.5

ML grade	
15	Not hired
40	Not hired
50	Not hired
58	Hired
66	Not hired
78	Not hired
83	Hired
90	Hired

Knowing how to split (3)

3. Evaluate each potential split point

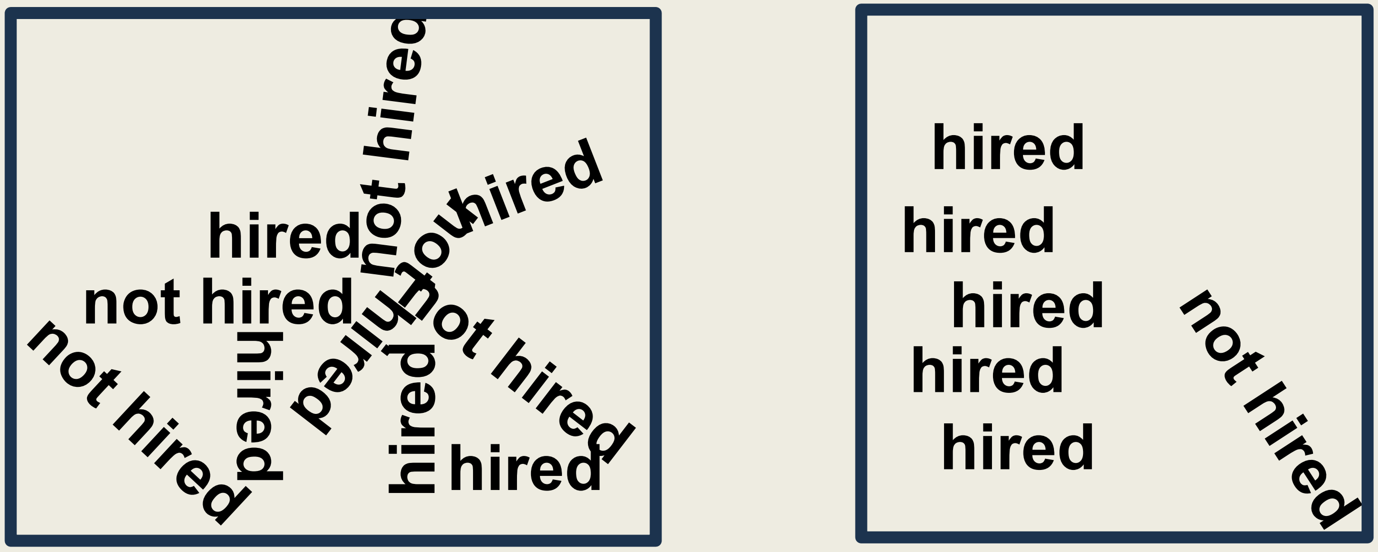
Evaluating a split point

The value of a split point is determined by a **split criterion**:

- Information gain or entropy; or
- Gini index

Label disorder/uncertainty in a partition

Entropy and Gini index are measures of disorder or uncertainty.



Choosing the best split point: Info. gain

- **Information gain** – reduction in uncertainty (entropy) about the label for the partition region
- The split point s with the highest information gain is chosen

$$Gain(m, m_{<s}, m_{>s}) = H_m - H_{m_{<s}/m_{>s}}$$

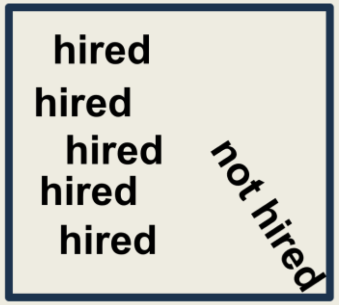
where

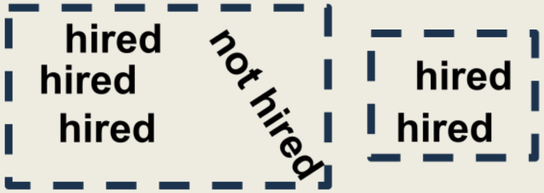
$$H_{m_{<s}/m_{>s}} = \frac{n_{m_{<s}}}{n_m} H_{m_{<s}} + \frac{n_{m_{>s}}}{n_m} H_{m_{>s}}$$

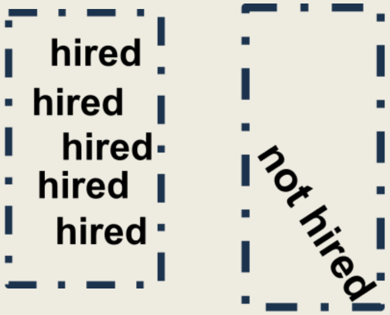
entropy $H_m = -\sum_{k=1}^{K} p(c_k|m) \log_2 p(c_k|m)$

$p(c_k|m)$ = probability of class c_k given data region m.

$p(c_k|m)$ can be computed from the data as $\frac{n_{m_k}}{n_m}$.







Choosing the best split point: Gini index

- **Gini index** – uncertainty
- The split point with the lowest index is chosen

$$G_{m_{<s}/m_{>s}} = \frac{n_{m_{<s}}}{n_m} G_{m_{<s}} + \frac{n_{m_{>s}}}{n_m} G_{m_{>s}}$$

where

$$G_m = 1 - \textstyle\sum_{k=1}^{K} p(c_k|m)^2$$

$$= 1 - \sum_{k=1}^{K} \left(\frac{n_{m_k}}{n_m}\right)^2$$

n_m = number of data instances in partition region m
n_{m_k} = number of data instances that belong to class c_k in partition region m

see the last slide pages for a worked out example

Knowing how to split : Categorical features

1. Potential split points – Each value is a potential split point
2. Count the number of occurrences of each class for the partition region ~~up to~~ for each given split point, and then beyond it
3. Evaluate each potential split point using a split criterion
4. Return the best split point for the feature

Decision tree algorithm: Optimal model

1. Given purity threshold, max number of leaf nodes, and initial region, compute the purity of the region

2. For the region, if purity >= threshold:
   - Stop splitting
   - Make majority class in the region the leaf node

3. Else:
   - For each feature, find the best split point based on a split criterion
   - Then, find the feature with the best split point for the current region
   - Split the region using that feature and its split point
   - For each region in the split, repeat from (1)

initial region

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

▭ purity threshold=1

♣ maximum number of leaf nodes=∞

Work through on your own: Decision tree

Using the (optimised) decision tree algorithm on the previous page, create an optimised decision tree for the given dataset.

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

Use *purity threshold*=1, *maximum number of leaf nodes*=∞, *initial region*=above data

Overfitting & Pruning

- Overfitting in decision trees manifests as a large tree with many nodes
- Pruning is a strategy to reduce overfitting in decision trees by removing nodes
- A pruning criterion v_T is used to determine if a given node should be pruned

$$v_T = \sum_{\tau=1}^{|T|} \varepsilon_\tau + \alpha|T|$$

where $|T|$ = total number of leaf nodes in tree T
α = pruning hyperparameter
ε_τ = uncertainty measure for τ (e.g. Gini index or entropy)

Any questions???

scan the QR code to ask questions

Ensemble learning

- ❑ Decision trees
- ❑ **Ensemble learning**

Ensemble learning

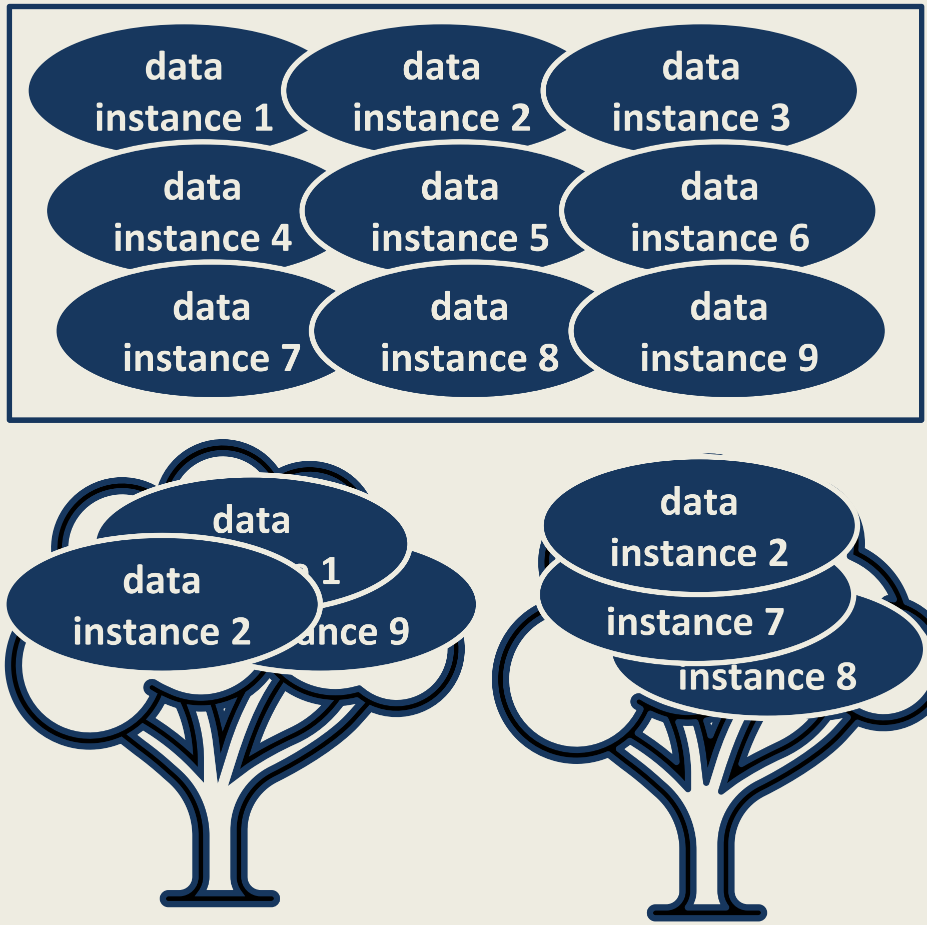
- Creating a community model made up of multiple ML models

- The model type for the constituent models
  - could be any – more types of models (**learning algorithms**) to be covered in coming weeks
  - is most commonly decision trees

- Common ensemble learning approaches
  - Bagging (<u>B</u>ootstrap <u>agg</u>regat<u>ing</u>)
  - Boosting

Bagging

- Bagging – Bootstrap aggregating
  - <u>Bootstrap</u> – Each model is trained on a randomly selected subset of the training data
  - <u>Aggregate</u> – Prediction of the community model is average prediction of its constituents



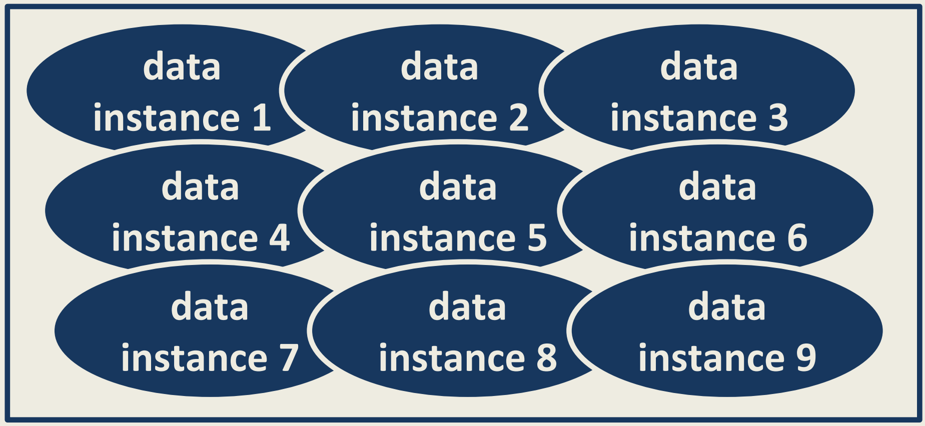
Merits of Bagging

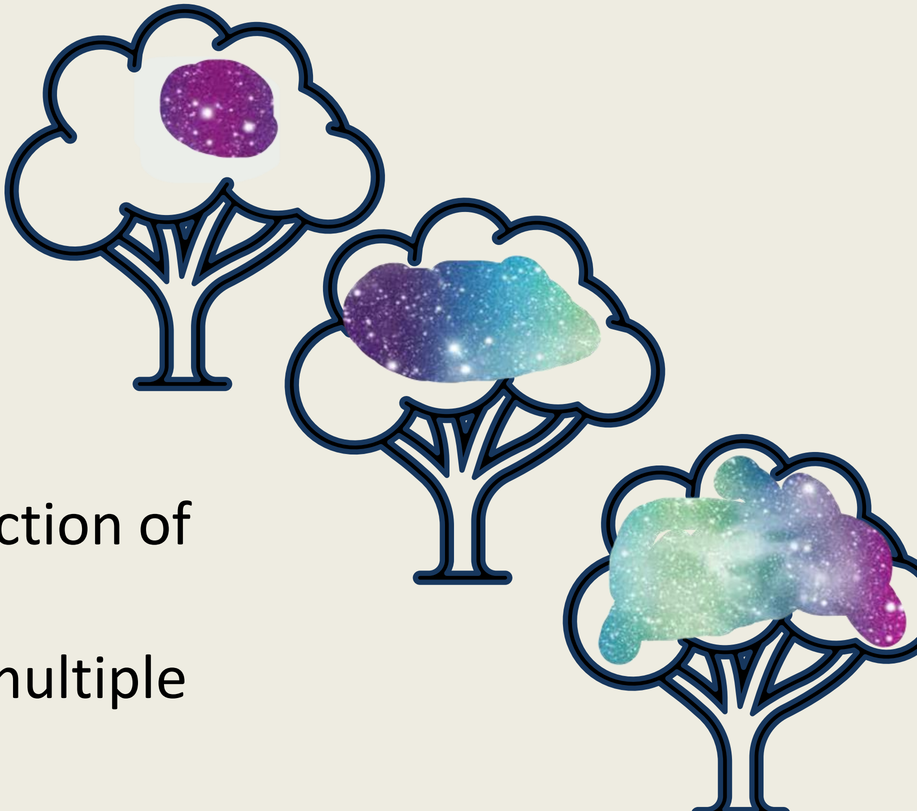
- Averaging of multiple functions captures complexity in the classification/regression problem
- Expected error can be less than the average expected error of the constituents
- Inherently robust to overfitting since each constituent is not fully fit to the training data

Boosting

AdaBoost (Adaptive Boosting) – common boosting method

- Cumulative training
  - Multiple models are created by training a base model cumulatively
  - Data with worse performance in one model is weighted for higher prediction penalty for the subsequent model
- Prediction weighting – Prediction of the community model is the weighted average over the multiple





see math details on last slide pages

Summary

1. Training the **decision tree** involves repeatedly partitioning the data region until each single partition has one single label.

2. Overfitting here can be addressed with **pruning**.

3. **Ensemble learning**, e.g. **bagging** or **boosting**, enables use of multiple weak models to make a model that generalizes better and would not overfit.

Any questions???

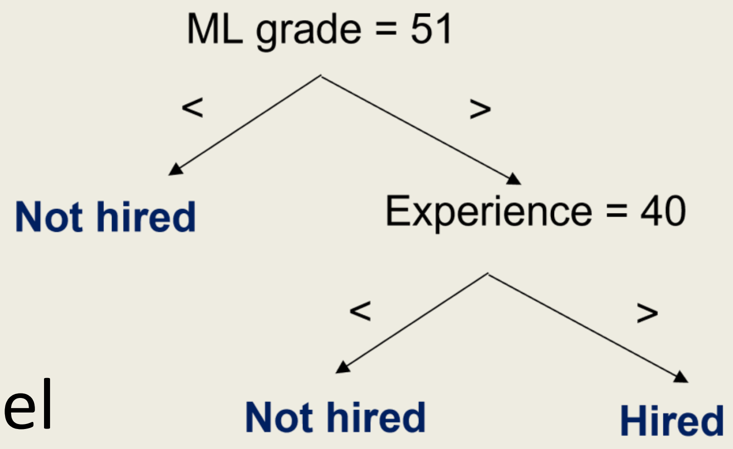
scan the QR code to ask questions

Math details and proofs

Knowing when to split – MATH DETAILS

- Further split of a partition is based on **purity** of that partition m

- Purity is proportion of instances of the majority label



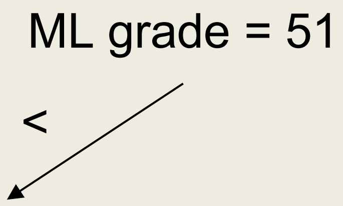
$$purity_m = \max_{c_k} \frac{n_{m_k}}{n_m}$$

where

n_m = number of data points in the partition region m

n_{m_k} = number of data points that belong to label class c_k in the partition region m

Purity example A – MATH DETAILS



Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

$$purity_m = \max_{c_k} \frac{n_{m_k}}{n_m}$$

$$= \max_{c_k} \left\{ \frac{3}{3} \; for\ c_1 = Not\ Hired, \qquad \frac{0}{3} \; for\ c_2 = Hired \right\}$$

$$= \frac{3}{3} = 1$$

Purity example B – MATH DETAILS

ML grade = 51

$>$

Experience	ML grade	
20	15	Not hired
35	50	Not hired
32	78	Not hired
67	83	Hired
46	90	Hired
12	40	Not hired
13	66	Not hired
70	58	Hired

$$purity_m = \max_{c_k} \frac{n_{mk}}{n_m}$$

$$= \max_k \left\{ \frac{2}{5}\ for\ c_1 = Not\ Hired, \quad \frac{3}{5}\ for\ c_2 = Hired \right\}$$

$$= \frac{3}{5} = 0.67$$

Knowing how to split – MATH DETAIL (1)

1. For a given feature, sort the values for that feature in the dataset

x	
Experience	**ML grade**
20	15
35	50
32	78
67	83
46	90
12	40
13	66
70	58

ML grade
15
40
50
58
66
78
83
90

Knowing how to split – MATH DETAIL (2)

2. For each value in the sorted list:
   - Compute potential split points – Midpoint between each value and the next, if different

     27.5, 45, 54, 62, 72, 80.5, 86.5

   - Count number of occurrences – for each class for the partition region up to each given split point, and then beyond it

     e.g. for split 27.5:

     $n_{m_{Hired_{<27.5}}} = 0, n_{m_{Not\_Hired_{<27.5}}} = 1$

     $n_{m_{Hired_{>27.5}}} = 3, n_{m_{Not\_Hired_{>27.5}}} = 4$

ML grade	
15	Not hired
40	Not hired
50	Not hired
58	Hired
66	Not hired
78	Not hired
83	Hired
90	Hired

Knowing how to split – MATH DETAIL (3a)

3. Evaluate each potential split point using a split criterion

Split points	$n_{m_{Hired_<}}$	$n_{m_{Not\_Hired_<}}$	$n_{m_{Hired_>}}$	$n_{m_{Not\_Hired_>}}$	
27.5	0	1	3	4	
45	0	2	3	3	
54	0	3	3	2	
62	1	3	2	2	
72	1	4	2	1	
80.5	1	5	2	0	
86.5	2	5	1	0	

Knowing how to split – MATH DETAIL (3b)

$$G_{m_{<s}/m_{>s}} = \frac{n_{m_{<s}}}{n_m}\left(1 - \sum_{k=1}^{K}\left(\frac{n_{m_{k,<s}}}{n_{m_{<s}}}\right)^2\right) + \frac{n_{m_{>s}}}{n_m}\left(1 - \sum_{k=1}^{K}\left(\frac{n_{m_{k,>s}}}{n_{m_{>s}}}\right)^2\right)$$

e.g. for split 27.5 = $\frac{1}{8}\left(1 - \left(\left(\frac{0}{1}\right)^2 + \left(\frac{1}{1}\right)^2\right)\right) + \frac{7}{8}\left(1 - \left(\left(\frac{3}{7}\right)^2 + \left(\frac{4}{7}\right)^2\right)\right)$

Split points	$n_{m_{Hired_<}}$	$n_{m_{Not\_Hired_<}}$	$n_{m_{Hired_>}}$	$n_{m_{Not\_Hired_>}}$	**Gini indices**
27.5	0	1	3	4	0.43
45	0	2	3	3	
54	0	3	3	2	
62	1	3	2	2	
72	1	4	2	1	
80.5	1	5	2	0	
86.5	2	5	1	0	

Work through on your own: Gini index

$$G_{m_{<s}/m_{>s}} = \frac{n_{m_{<s}}}{n_m}\left(1 - \sum_{k=1}^{K}\left(\frac{n_{m_{k,<s}}}{n_{m_{<s}}}\right)^2\right) + \frac{n_{m_{>s}}}{n_m}\left(1 - \sum_{k=1}^{K}\left(\frac{n_{m_{k,>s}}}{n_{m_{>s}}}\right)^2\right)$$

Which split point is the most optimal?

Split points	$n_{m_{Hired_<}}$	$n_{m_{Not\_Hired_<}}$	$n_{m_{Hired_>}}$	$n_{m_{Not\_Hired_>}}$	Gini indices
27.5	0	1	3	4	0.43
45	0	2	3	3	
54	0	3	3	2	
62	1	3	2	2	
72	1	4	2	1	
80.5	1	5	2	0	
86.5	2	5	1	0	

AdaBoost data weighting – MATH DETAIL

Given a binary classification task, total error for model iteration i is

$$\varepsilon_i = \frac{\sum_{n=1}^{N} \alpha_n^{(i)} I(f_i(x_n) \neq y_n)}{\sum_{n=1}^{N} \alpha_n^{(i)}}$$

where

$I(f_i(x_n) \neq y_n)$ = 1 if model i misclassifies x_n, 0 otherwise

$\alpha_n^{(i)}$ = weight for data point x_n with model i

$$\alpha_n^{(i)} = \begin{cases} 1/_N, & for\ i = 1 \\ \alpha_n^{(i-1)} e^{\beta_{i-1} \cdot I(f_{i-1}(x_n) \neq y_n)}, & for\ i > 1 \end{cases}$$

such that $\beta_{i-1} = \ln \frac{\varepsilon_{i-1}}{1-\varepsilon_{i-1}}$

AdaBoost prediction weighting – MATH

For binary classification given classes *+1* and *-1*, the prediction across all models over all iterations is

$$\hat{y} = sign\left(\sum_{i=1}^{M}\left(\log\frac{1-\varepsilon_i}{\varepsilon_i}\right)\hat{y}_i\right)$$