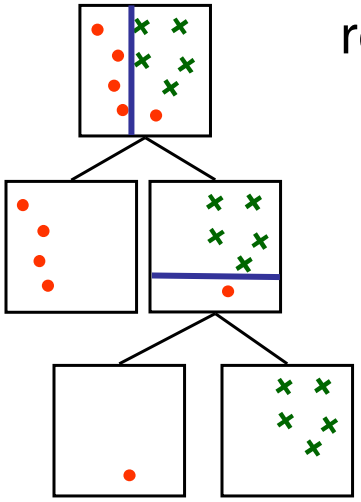
Partial Review

Decision Trees: Things to remember



1. Training: Find (variable,threshold) pairs to split the node until a “pure” node (one class) is reached
2. Test: Given new data, follow the tree until a leaf node is reached and return corresponding output value

Decision Trees

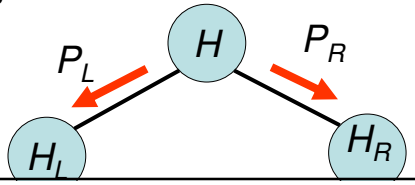
3. Measure of purity of a node → Entropy

$$\boldsymbol{H(Y) = -\sum_{i=1}^{n} P_i \log_2 P_i}$$

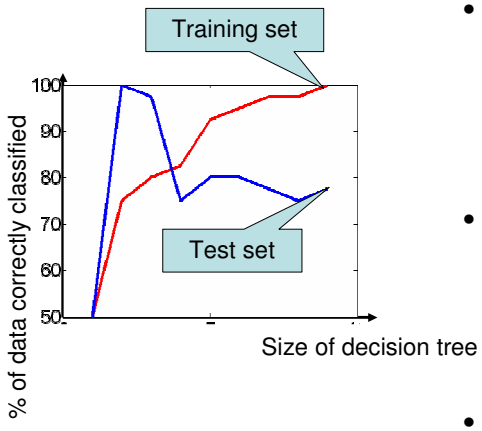
4. Measure of usefulness of a split → Information gain

$$IG(Y|X_j,t) = H(Y) - H(Y|X_j,t)$$

$$H(Y|X_j,t) = (H_L \text{x } P_L + H_R \text{x } P_R)$$



Decision Trees



- 6. Overfitting problem: DT can predict exactly *Y* from *X* from the training data but may do poorly new data
- 7. Pruning:
  - Evaluate performance on validation data
  - Remove splits if they are not statistically significant
- 8. Using pruned tree: Return the "majority" value in the leaf node reached after traversing the tree

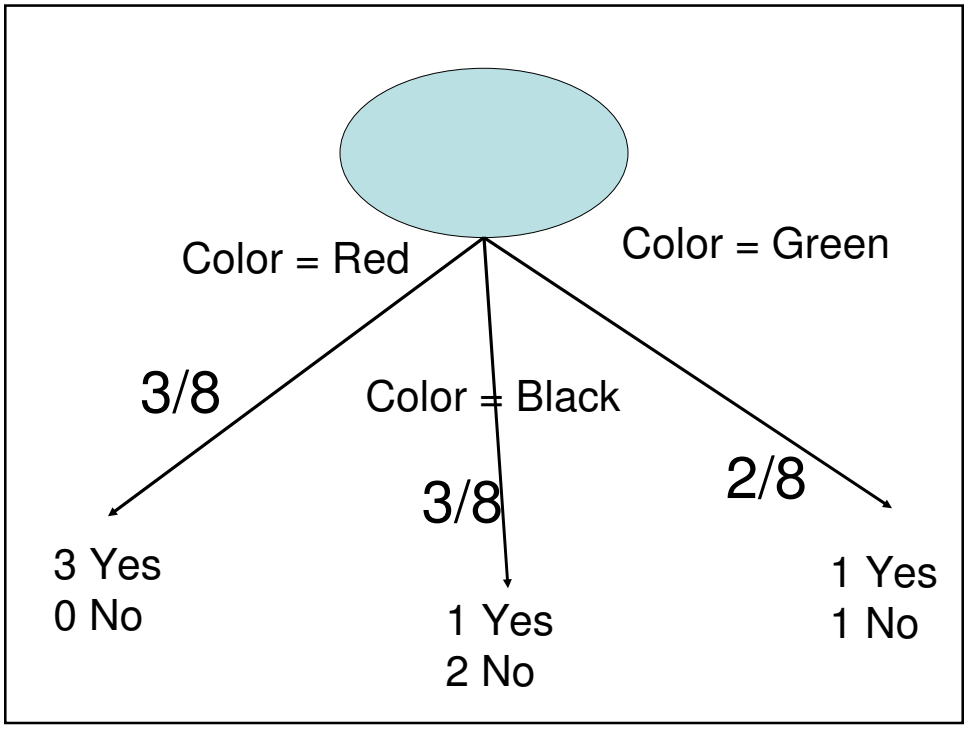
Example

Object	Color	Number of eyes	Alive
A	Red	4	Yes
B	Black	42	No
C	Red	13	Yes
D	Green	3	Yes
E	Black	27	No
F	Red	2	Yes
G	Black	1	Yes
H	Green	11	No

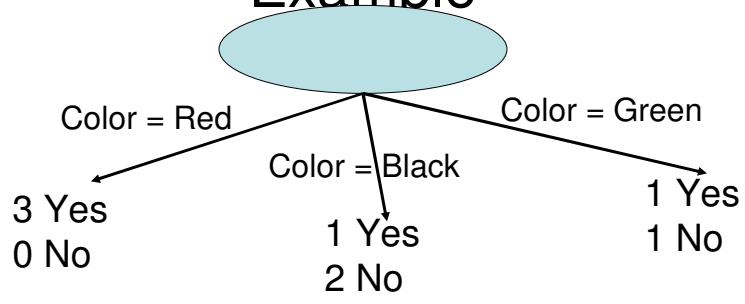
- H(Alive) = ?
- IG(Alive|Color) = ?

Example

- $H(Alive) = -P_{Yes} \log_2(P_{Yes}) - P_{No} \log_2(P_{No})$
- $P_{Yes} = 5/8$
- $P_{No} = 3/8$
- H(Alive) ~ 0.95



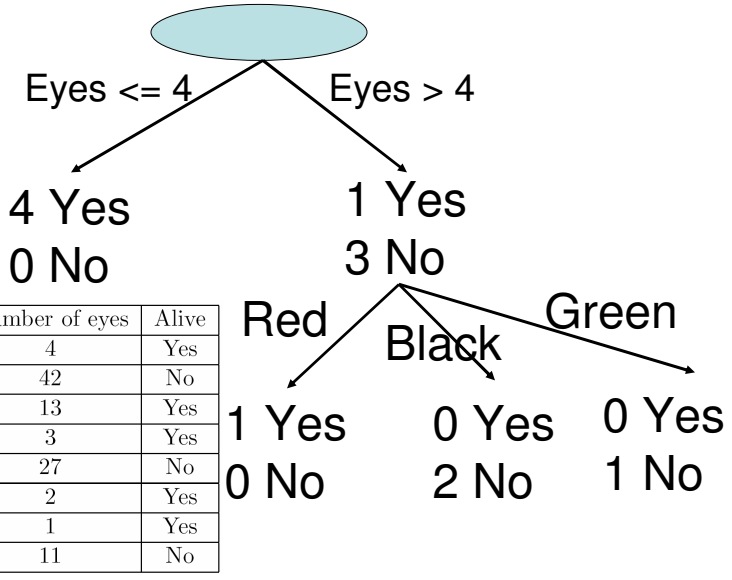
Example



- H(Alive|Color = Red) = 0
- H(Alive|Color = Black)=-1/3 log(1/3)–2/3log(2/3)
- H(Alive|Color = Green)=-1/2 log(1/2)–1/2log(1/2) = 1
- IG(Alive|Color) = H(Alive) – 3/8 x 0 – 3/8 x H(Alive|Color = Black) – 2/8 x 1 ~ 0.36

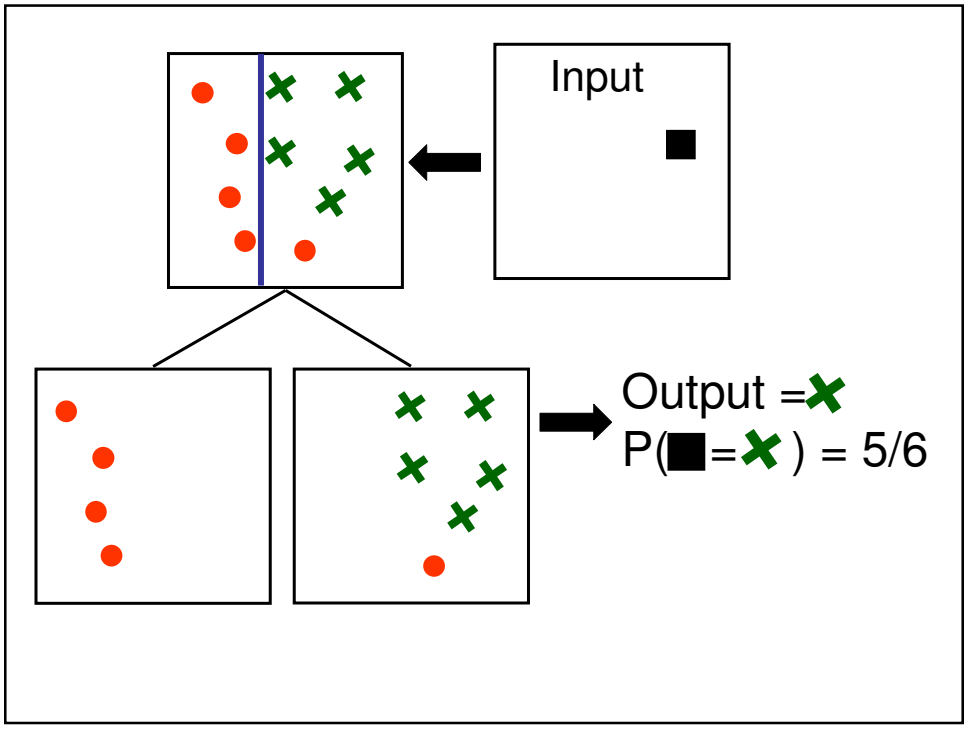
Example

- Tree when splitting first on (Eyes > 4) and then on color?
- Classification for data = (red,23eyes)?
- Classification for data = (black,2eyes)?

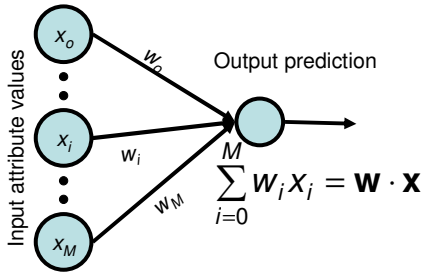


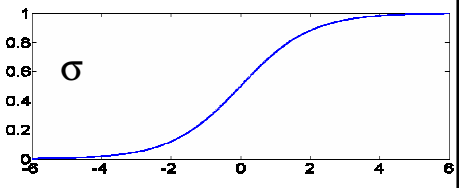
Object	Color	Number of eyes	Alive
A	Red	4	Yes
B	Black	42	No
C	Red	13	Yes
D	Green	3	Yes
E	Black	27	No
F	Red	2	Yes
G	Black	1	Yes
H	Green	11	No

- (red,23eyes) → Yes
- (black,2eyes) → Yes



Neural Nets: Things to remember





1. Output prediction is either **w**.**x** or σ(**w**.**x**)
2. Can solve any *linear* problem
3. Training by minimizing:

$$\boldsymbol{E} = \sum_{\text{Training Data}} \left(\boldsymbol{y}^{\boldsymbol{k}} - \mathbf{w} \cdot \mathbf{x}^{\boldsymbol{k}} \right)^2 = \sum_{\boldsymbol{k}=1}^{\boldsymbol{N}} \delta_{\boldsymbol{k}}^2$$

Neural Nets

4. Incremental update rule during training (guaranteed convergence):

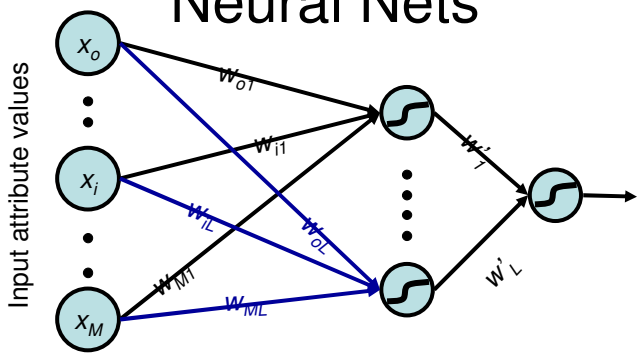
$$w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i} \quad \Rightarrow \quad w_i \leftarrow w_i + \alpha \delta_k x_i^k$$

OR

$$w_i \leftarrow w_i + \alpha \delta_k x_i^k \sigma'\left(\mathbf{w} \cdot \mathbf{x}^k\right)$$

5. α = learning rate (low = smooth convergence but slow, high = chaotic convergence but fast)

Neural Nets



6. Can represent any function of the input attributes (given enough layers and units)
7. Convergence is *not* guaranteed

Example

- Inputs: *M* binary variables $x_1,..,x_M$ (*M odd*)
- Output: *Y* = majority value of the inputs, i.e., *Y* = 1 if there are more 1s than zeros, *Y* = 0 otherwise
- Design a neural networks that computes *Y* from $x_1,..,x_M$

Answer