Machine Learning and Intelligent Systems

Decision Trees

Maria A. Zuluaga

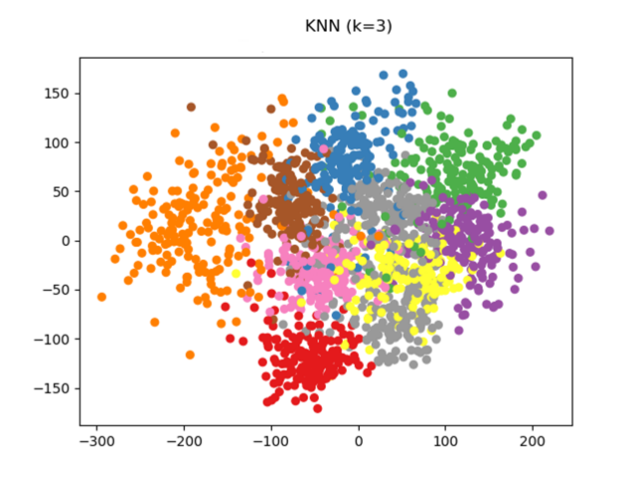
Dec 8, 2023

EURECOM - Data Science Department

Table of contents

Recap

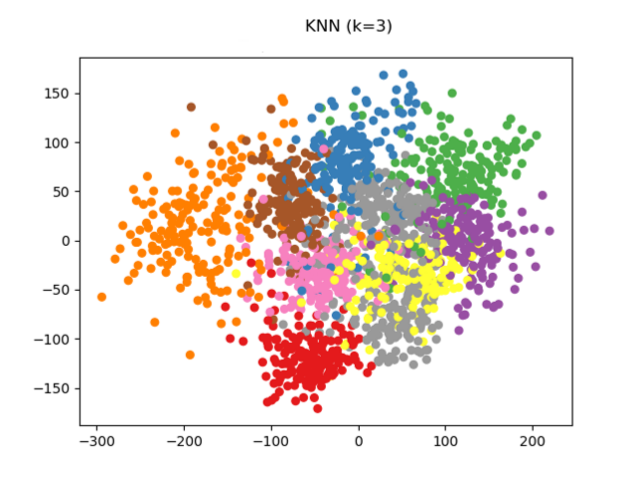
K Nearest Neighbors



Source: adapted from scikit-learn

- Assume this is how your training data looks like
- You have chosen $K = 3$
- A new test point arrives
- What do you do?

K Nearest Neighbors

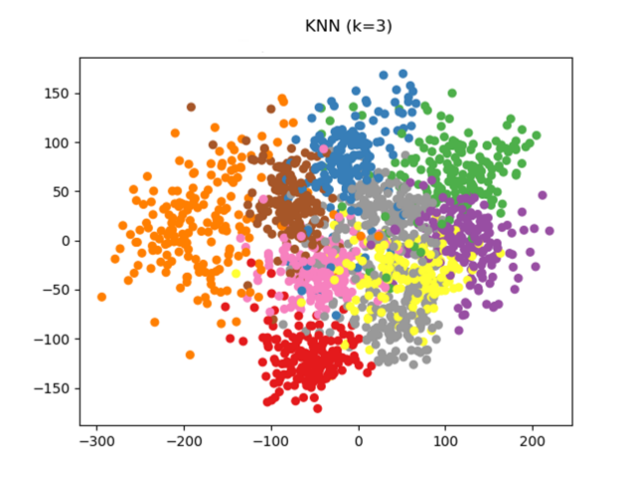


Source: adapted from scikit-learn

- Assume this is how your training data looks like
- You have chosen $K = 3$
- A new test point arrives
- What do you do?

Anything wrong with that?

K Nearest Neighbors



Source: adapted from scikit-learn

- Assume this is how your training data looks like
- You have chosen $K = 3$
- A new test point arrives
- What do you do?

Anything wrong with that?

- Testing time complexity: $\mathcal{O}(ND)$
- Ideally, $N \ggg 0$

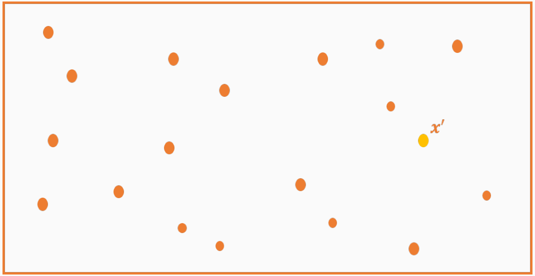
Motivation: Improving k-NN

Goal: How can we make k-NN faster at test time?

Motivation: Improving k-NN

Goal: How can we make k-NN faster at test time?

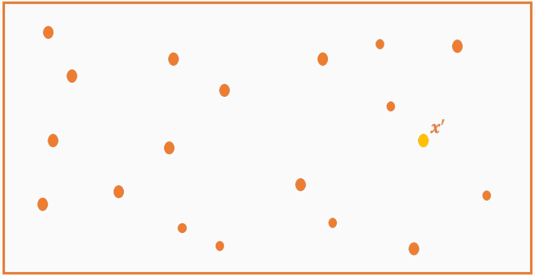
Imagine the following setup: The orange points represent the training points (no labels). We want to predict $\hat{y}$ for the yellow point.

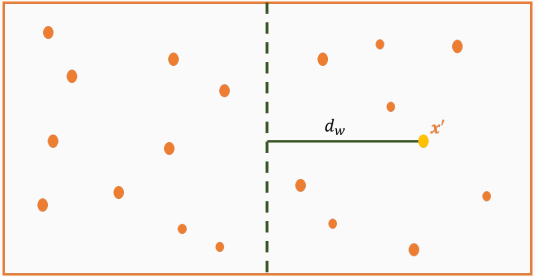


Motivation: Improving k-NN

Goal: How can we make k-NN faster at test time?

Imagine the following setup: The orange points represent the training points (no labels). We want to predict $\hat{y}$ for the yellow point.





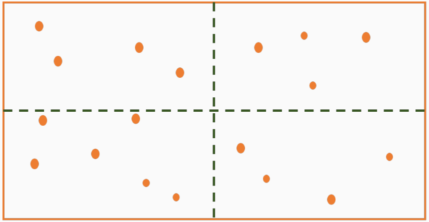
Improved K-NNs: KD Trees

Intuition: Partition of the D-dimensional feature space to make the search of the K nearest neighbors faster

Algorithm ($D = 2$)

Training step:

1. Divide the data along one feature by setting a threshold t_1
2. Divide again along the remaining feature by setting a threshold t_2
3. Record the quadrant each **x** belongs to

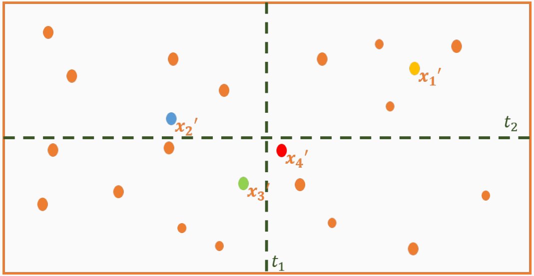


Improved K-NNs: KD Trees

Algorithm $(D = 2)$

Test step (K=1):

1. Find the quadrant Q of $\mathbf{x}'$
2. d_x: distance of $\mathbf{x}'$ to closest point in Q
3. $d_{t_1} = d(\mathbf{x}', t_1)$
4. $d_{t_2} = d(\mathbf{x}', t_2)$

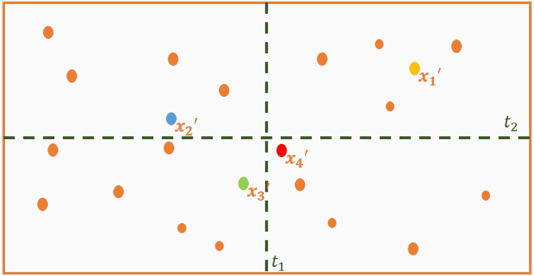


Improved K-NNs: KD Trees

Algorithm $(D = 2)$

Test step (K=1):

1. Find the quadrant Q of $\mathbf{x}'$
2. d_x: distance of $\mathbf{x}'$ to closest point in Q
3. $d_{t_1} = d(\mathbf{x}', t_1)$
4. $d_{t_2} = d(\mathbf{x}', t_2)$
5. **Case 1:** $d_x < d_{t_1}$ & $d_x < d_{t_2}$
   closest point in quadrant



Improved K-NNs: KD Trees

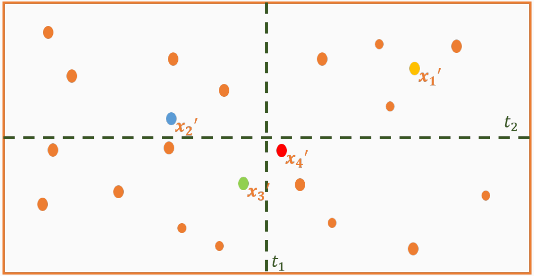
Algorithm $(D = 2)$

Test step (K=1):

1. Find the quadrant Q of $\mathbf{x}'$
2. d_x: distance of $\mathbf{x}'$ to closest point in Q
3. $d_{t_1} = d(\mathbf{x}', t_1)$
4. $d_{t_2} = d(\mathbf{x}', t_2)$
5. **Case 1:** $d_x < d_{t_1}$ & $d_x < d_{t_2}$

   closest point in quadrant
6. **Case 2:** $d_x < d_{t_1}$ & $d_x > d_{t_2}$

   check neighbor quadrant (via t_2)

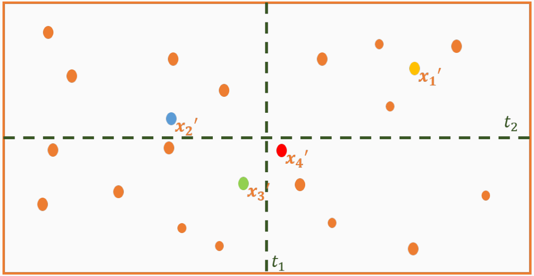


Improved K-NNs: KD Trees

Algorithm $(D = 2)$

Test step (K=1):

1. Find the quadrant Q of $\mathbf{x}'$
2. d_x: distance of $\mathbf{x}'$ to closest point in Q
3. $d_{t_1} = d(\mathbf{x}', t_1)$
4. $d_{t_2} = d(\mathbf{x}', t_2)$
5. **Case 1:** $d_x < d_{t_1}$ & $d_x < d_{t_2}$
   closest point in quadrant
6. **Case 2:** $d_x < d_{t_1}$ & $d_x > d_{t_2}$
   check neighbor quadrant (via t_2)
7. **Case 3:** $d_x > d_{t_1}$ & $d_x < d_{t_2}$
   check neighbor quadrant (via t_1)

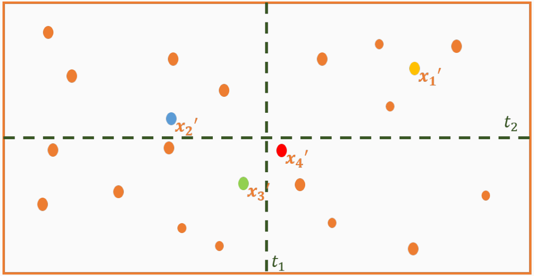


Improved K-NNs: KD Trees

Algorithm $(D = 2)$

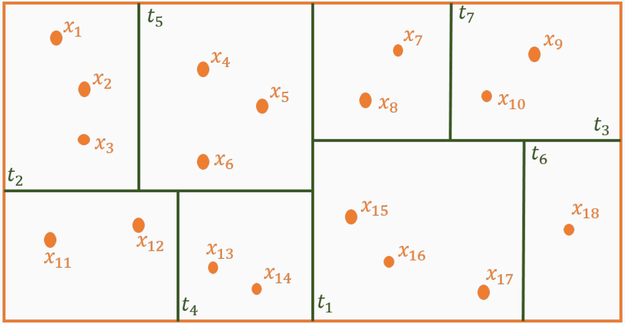
Test step (K=1):

1. Find the quadrant Q of $\mathbf{x}'$
2. d_x: distance of $\mathbf{x}'$ to closest point in Q
3. $d_{t_1} = d(\mathbf{x}', t_1)$
4. $d_{t_2} = d(\mathbf{x}', t_2)$
5. **Case 1:** $d_x < d_{t_1}$ & $d_x < d_{t_2}$
   closest point in quadrant
6. **Case 2:** $d_x < d_{t_1}$ & $d_x > d_{t_2}$
   check neighbor quadrant (via t_2)
7. **Case 3:** $d_x > d_{t_1}$ & $d_x < d_{t_2}$
   check neighbor quadrant (via t_1)

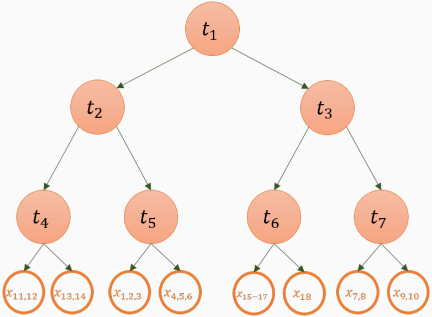


8. **Case 4:** $d_x > d_{t_1}$ & $d_x > d_{t_2}$
   check all other quadrants
   Case 4 is as bad as standard kNN

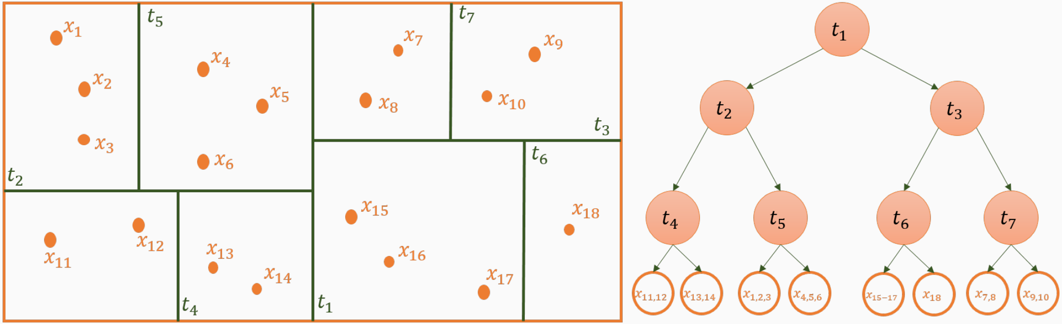
KD Trees



t_5
t_7
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
t_3
t_2
t_6
x_{11}
x_{12}
x_{13}
x_{14}
x_{15}
x_{16}
x_{17}
x_{18}
t_4
t_1



t_1
t_2
t_3
t_4
t_5
t_6
t_7
$x_{11,12}$
$x_{13,14}$
$x_{1,2,3}$
$x_{4,5,6}$
x_{15-17}
x_{18}
$x_{7,8}$
$x_{9,10}$



- The number of elements in a leaf node (leaf size) is a hyper-parameter
- Question: What is the leaf size in the example?

KD Trees: Some open questions

- **Q:** How should the data be split during training?
  A: Split the data in half recursively by rotating through the features.
  When rotating, choosing the feature with maximum variance is a good heuristic

KD Trees: Some open questions

- **Q:** How should the data be split during training?

  A: Split the data in half recursively by rotating through the features.

  When rotating, choosing the feature with maximum variance is a good heuristic
- **Q:** How to define the threshold for the split?

  A: Use the median over the range of values of the feature to be split

KD Trees: Some open questions

- **Q:** How should the data be split during training?
  A: Split the data in half recursively by rotating through the features.
  When rotating, choosing the feature with maximum variance is a good heuristic
- **Q:** How to define the threshold for the split?
  A: Use the median over the range of values of the feature to be split
- **Q:** How to extend the algorithm to $K > 1$?
  A: Instead of considering the distance to the closest point, take K closest distances

KD Trees: Some open questions

- **Q:** How should the data be split during training?

  A: Split the data in half recursively by rotating through the features.

  When rotating, choosing the feature with maximum variance is a good heuristic
- **Q:** How to define the threshold for the split?

  A: Use the median over the range of values of the feature to be split
- **Q:** How to extend the algorithm to $K > 1$?

  A: Instead of considering the distance to the closest point, take K closest distances
- **Q:** How often we may fall into a situation like case 4?

  A: Let's answer to this question by recalling Lab 1

Summary

Advantages

- Easy to build
- Accelerates inference

Disadvantages

- Curse of dimensionality makes it impractical for high values of D

Can we do better?

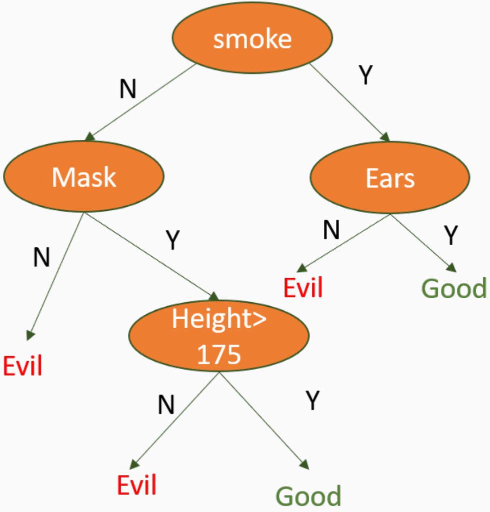
- Ball trees follow a different principle to space partitioning that makes them better for high dimensional spaces (not covered)
- We will exploit the idea behind KD trees to build decision trees

Decision Trees

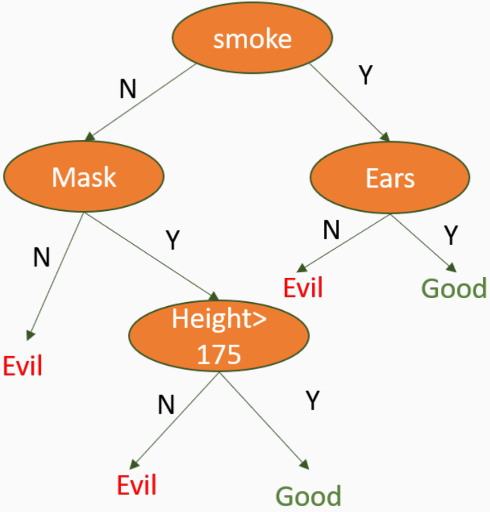
An Example

	Mask	Cape	Tie	Pointy Ears	Smokes	Height	Label
Training set							
Batman	Y	Y	N	Y	N	185	**Good**
Robin	Y	Y	N	N	N	175	**Good**
Alfred	N	N	Y	N	N	180	**Good**
Penguin	N	N	Y	N	Y	145	**Evil**
Catwoman	Y	N	N	Y	N	170	**Evil**
Joker	N	N	Y	N	N	177	**Evil**
Test set							
Batgirl	Y	Y	N	Y	N	165	?
Riddler	Y	N	N	N	N	178	?

A first decision tree

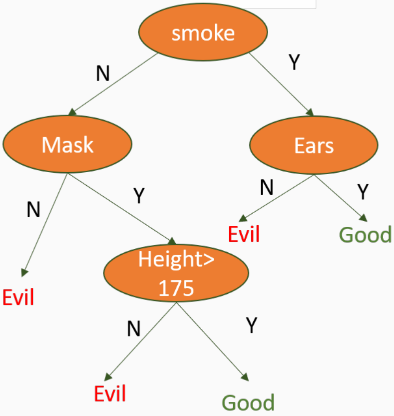


A first decision tree

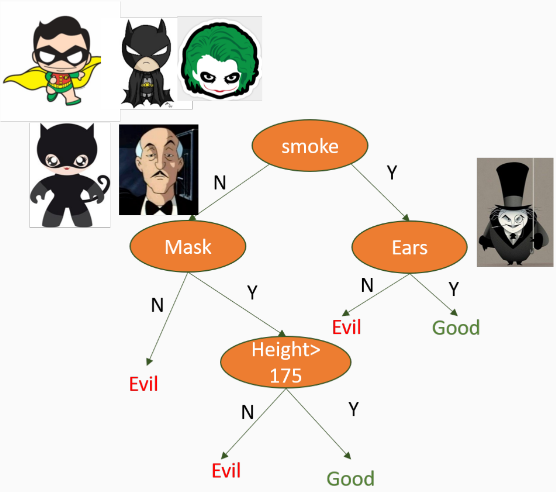


Does it classify properly the training data?

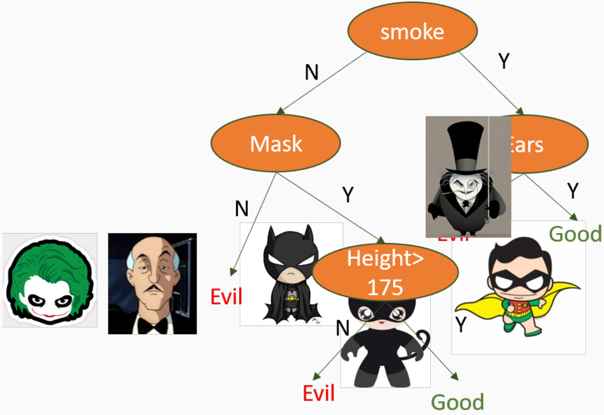
Step 1: All samples at the root node



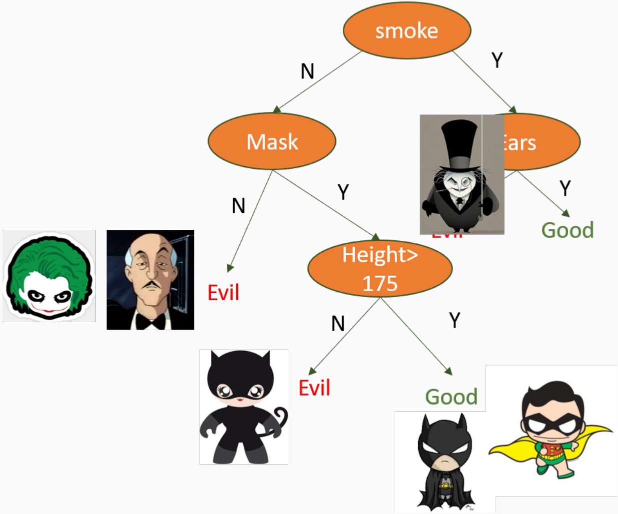
Step 2: Split according to the root node



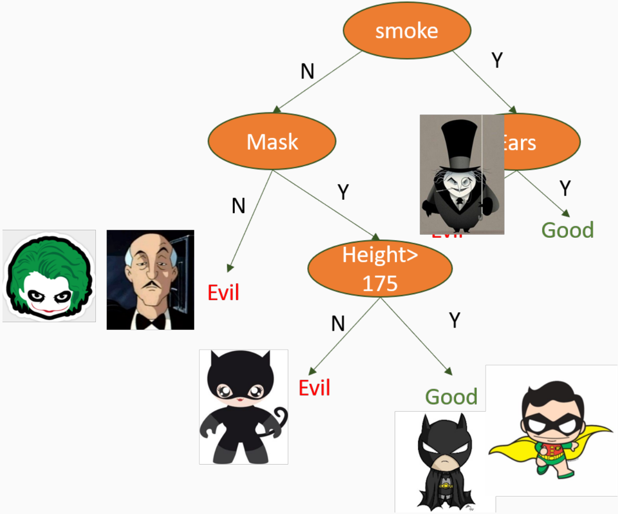
Step 3: Split according to second level nodes



Step 4: Split according to the last node



Step 4: Split according to the last node



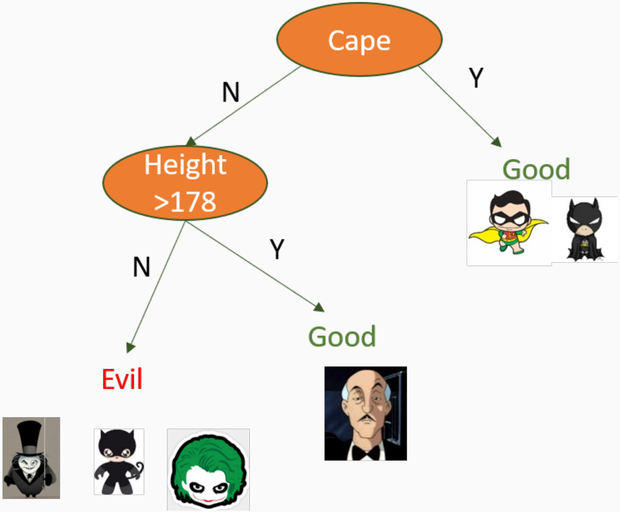
Alfred has been misclassified

Exercise: The smallest tree

Exercise: What is the smallest tree you can construct that properly classifies all the data? How many nodes?

	Mask	Cape	Tie	Pointy Ears	Smokes	Height	Label
Training set							
Batman	Y	Y	N	Y	N	185	**Good**
Robin	Y	Y	N	N	N	175	**Good**
Alfred	N	N	Y	N	N	180	**Good**
Penguin	N	N	Y	N	Y	145	**Evil**
Catwoman	Y	N	N	Y	N	170	**Evil**
Joker	N	N	Y	N	N	177	**Evil**
Test set							
Batgirl	Y	Y	N	Y	N	165	?
Riddler	Y	N	N	N	N	178	?

Exercise: The smallest tree



Decision Trees

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

- Solution - Resort to a greedy approach:
    - Start from empty decision tree
    - Split on next best feature
    - Recurse

Decision Trees

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]
- Solution - Resort to a greedy approach:
  - Start from empty decision tree
  - Split on next best feature
  - Recurse

Goal: Build a maximally compact tree with only pure leaves

Impurity Functions

Greedy strategy: We keep splitting the data to minimize an impurity function that measures label purity among the children.

Impurity Functions

Greedy strategy: We keep splitting the data to minimize an impurity function that measures label purity among the children.

Formalization & Definitions

- Data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}, \quad y_i \in \{1, \ldots, K\}$
- K: Total number of classes

Impurity Functions

Greedy strategy: We keep splitting the data to minimize an impurity function that measures label purity among the children.

Formalization & Definitions

- Data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}, \quad y_i \in \{1, \ldots, K\}$
- K: Total number of classes
- $\mathcal{D}_k \subset \mathcal{D}$, where $\mathcal{D}_k = \{(\mathbf{x}, y) \in \mathcal{D} : y = k\}$

Impurity Functions

Greedy strategy: We keep splitting the data to minimize an impurity function that measures label purity among the children.

Formalization & Definitions

- Data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}, \quad y_i \in \{1, \ldots, K\}$
- K: Total number of classes
- $\mathcal{D}_k \subset \mathcal{D}$, where $\mathcal{D}_k = \{(\mathbf{x}, y) \in \mathcal{D} : y = k\}$
- $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_K$

Gini impurity

Given the previous definitions, the fraction of inputs in $\mathcal{D}$ with label k is:

$$p_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$$

Gini impurity:

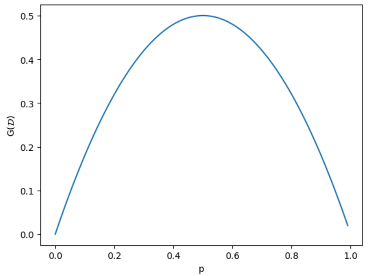
$$G(\mathcal{D}) = \sum_{k=1}^{K} p_k(1 - p_k) \tag{1}$$

Note 1: Often in the literature, they will use S to denote the data, rather than $\mathcal{D}$.
Note 2: The Gini impurity is not the same as the Gini coefficient. The latter measures the level of inequality in a country.

Gini Impurity: Analysis

Let us consider the case where $K = 2$

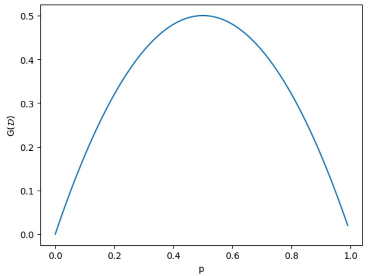
$$\begin{aligned} G(\mathcal{D}) &= p_1(1 - p_1) + p_2(1 - p_2) \\ &= p_1(1 - p_1) + (1 - p_1)p_1 \\ &= 2p(1 - p) \end{aligned}$$



Gini Impurity: Analysis

Let us consider the case where $K = 2$

$$\begin{aligned} G(\mathcal{D}) &= p_1(1 - p_1) + p_2(1 - p_2) \\ &= p_1(1 - p_1) + (1 - p_1)p_1 \\ &= 2p(1 - p) \end{aligned}$$



Interpretation:

- Maximum impurity at p=0.5 since there is 50-50 for both classes (root node case)
- Our goal is to make the Gini impurity zero
- *Low gini:* Dominated by one class.
  High gini: There is no dominating class

Entropy

Let us recall p_k the fraction of inputs for a given label, what is the worst case scenario for the leave of a tree?

Entropy

Let us recall p_k the fraction of inputs for a given label, what is the worst case scenario for the leave of a tree?

Answer: $q_1 = q_2, = \ldots = q_k = \frac{1}{K}$. We do not want to have a uniform distribution

Idea: Let us measure impurity as how close we are to the uniform distribution. We will use the Kullback-Liebler divergence to measure the closeness.

Entropy

Let us recall p_k the fraction of inputs for a given label, what is the worst case scenario for the leave of a tree?

Answer: $q_1 = q_2, = \ldots = q_k = \frac{1}{K}$. <u>We do not want to have a uniform distribution</u>

Idea: Let us measure impurity as how close we are to the uniform distribution. We will use the Kullback-Liebler divergence to measure the closeness.

Kullback-Liebler (KL-)divergence):

$$KL(p||q) = \sum_{k=1}^{K} p_k \log \frac{p_k}{q} \geq 0$$

KL-divergence to measure closeness

$$
\begin{aligned}
KL(p||q) &= \sum_{k=1}^{K} p_k \log \frac{p_k}{q} \\
&= \sum_{k=1}^{K} p_k \log p_k - p_k \log q
\end{aligned}
$$

KL-divergence to measure closeness

$$
\begin{aligned}
KL(p||q) &= \sum_{k=1}^{K} p_k \log \frac{p_k}{q} \\
&= \sum_{k=1}^{K} p_k \log p_k - p_k \log q
\end{aligned}
$$

Our goal is to measure how far p_k are from the uniform distribution. Let us then replace accordingly for such a case

$$
\begin{aligned}
KL\left(p||\frac{1}{K}\right) &= \sum_{k=1}^{K} p_k \log p_k - p_k \log \frac{1}{K} \\
&= \sum_{k=1}^{K} p_k \log p_k + p_k \log K \\
&= \sum_{k=1}^{K} p_k \log p_k + \log K \sum_{k=1}^{K} p_k
\end{aligned}
$$

KL-divergence to measure closeness

We can disregard the term $\log K$ because it is a constant. Moreover, we know that

$$\sum_{k=1}^{K} p_k = 1$$

which leads to

$$KL\left(p || \frac{1}{K}\right) = \sum_{k=1}^{K} p_k \log p_k$$

KL-divergence to measure closeness

We can disregard the term $\log K$ because it is a constant. Moreover, we know that

$$\sum_{k=1}^{K} p_k = 1$$

which leads to

$$KL\left(p||\frac{1}{K}\right) = \sum_{k=1}^{K} p_k \log p_k$$

Our goal is to make sure that p_k is as far as possible from the uniform distribution. This means

$$\max_p KL\left(p||\frac{1}{K}\right) = \max_p \sum_{k=1}^{K} p_k \log p_k \tag{2}$$

Entropy

Along the course we have preferred minimization to maximization. We can transform the problem easily

$$\max_p KL\left(p||\frac{1}{K}\right) = \max_p \sum_{k=1}^{K} p_k \log p_k$$

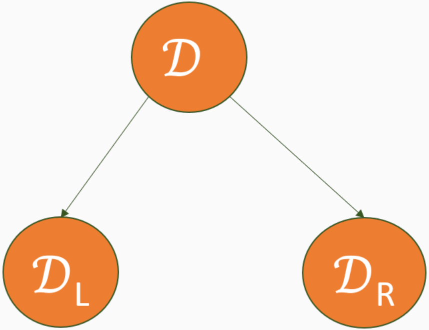
to

$$\min_p KL\left(p||\frac{1}{K}\right) = \min_p -\sum_{k=1}^{K} p_k \log p_k \tag{3}$$

This quantity represents the entropy H(S):

$$\min_p KL\left(p||\frac{1}{K}\right) = \min_p H(\mathcal{D}) \tag{4}$$

Estimating the entropy of a tree



The entropy of a tree can be estimated as:

$$H(\mathcal{D}) = p^L H(\mathcal{D}_L) + p^R H(\mathcal{D}_R) \qquad (5)$$

where:

$$p^L = \frac{|\mathcal{D}_L|}{|\mathcal{D}|}$$
$$p^R = \frac{|\mathcal{D}_R|}{|\mathcal{D}|}$$

The Algorithm

1. Compute the impurity function for every **attribute** of dataset $S = \mathcal{D}$
2. Split the set S into subsets using the attribute for which the resulting impurity function is minimal after splitting (greedy approach)
3. Make a decision tree node containing that attribute
4. Recurse on subsets with remaining attributes

A stopping criterion is required to determine when to stop

Iterative Dichotomiser 3 (ID3) Algorithm

The ID3 algorithm follows the scheme previously presented and uses the following set of rules to decide when to stop:

Rule 1 Every element in the subset belongs to the same class. Node becomes a leaf node

Rule 2 There are no more attributes to be selected, but the examples still do not belong to the same class. Node becomes leaf node labeled with the majority class

Rule 3 There are no examples in the subset, as no example in the parent set matches a specific value of the selected attribute. Create leaf node labeled with the majority class of the parent

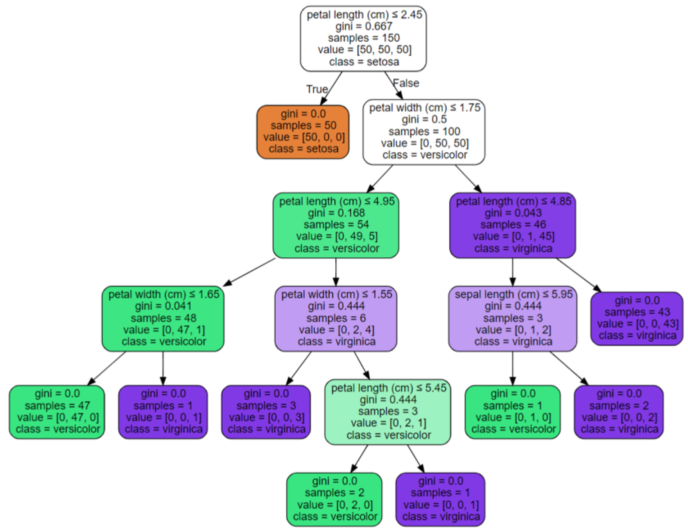
ID3 Algorithm: Splits

The greedy approach to splitting a parent node into sub-nodes requires to try all features/attributes and all possible splits.

The chosen attribute a is the one that will minimize the impurity for a given threshold t defining the split. More formally,

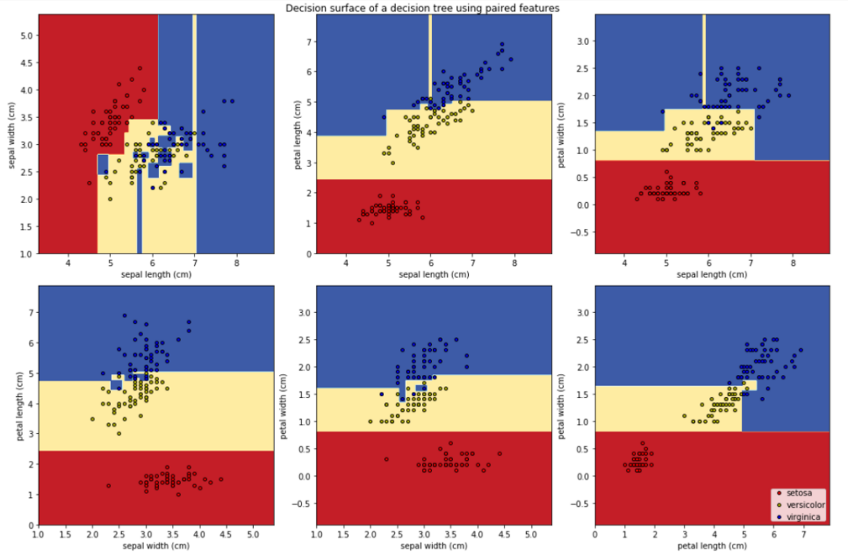
$$S^L = \{(\mathbf{x}, y) \in S : x^{(a)} \leq t\}$$
$$S^R = \{(\mathbf{x}, y) \in S : x^{(a)} > t\}$$

An Example: The Tree



see `06_trees.ipynb`

An Example: Decision Boundary



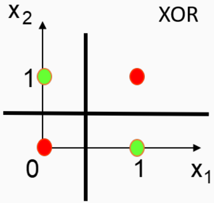
Limitations: Myopia

Let us have a look at the XOR and try to train a decision tree that learns to classify its samples properly

Root node: $S = \mathcal{D}$

$$H(S) = -(0.5\log(2/4) + 0.5\log(2/4)) = 0.69$$

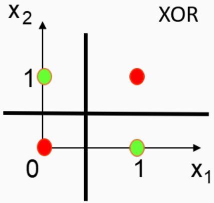
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



Limitations: Myopia

Let us have a look at the XOR and try to train a decision tree that learns to classify its samples properly

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



Root node: $S = \mathcal{D}$

$H(S) = -(0.5\log(2/4) + 0.5\log(2/4)) = 0.69$

Try split with feature a:

$S^L = \{(\mathbf{x} = [0,0], y = 0), (\mathbf{x} = [0,1], y = 1)\},\ P^L = 0.5$

$S^R = \{(\mathbf{x} = [1,0], y = 1), (\mathbf{x} = [1,1], y = 0)\},\ P^R = 0.5$

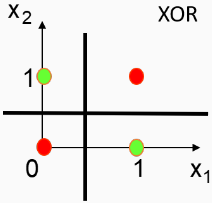
$H(S^L) = -(0.5\log(1/2) + 0.5\log(1/2)) = 0.69$

$H(S^R) = 0.69$

Limitations: Myopia

Let us have a look at the XOR and try to train a decision tree that learns to classify its samples properly

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



Root node: $S = \mathcal{D}$

$H(S) = -(0.5\log(2/4) + 0.5\log(2/4)) = 0.69$

Try split with feature a:

$S^L = \{(\mathbf{x} = [0,0], y = 0), (\mathbf{x} = [0,1], y = 1)\},\ P^L = 0.5$

$S^R = \{(\mathbf{x} = [1,0], y = 1), (\mathbf{x} = [1,1], y = 0)\},\ P^R = 0.5$

$H(S^L) = -(0.5\log(1/2) + 0.5\log(1/2)) = 0.69$

$H(S^R) = 0.69$

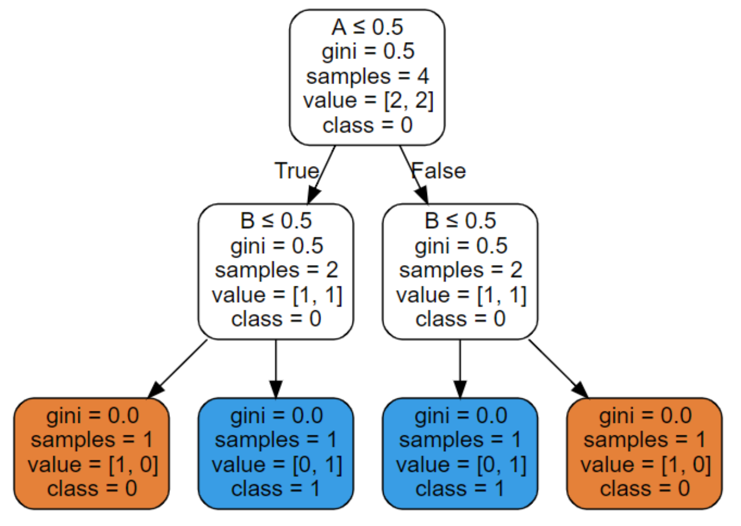
Try split with feature b:

It leads to the same case as with feature a: **No improvement**

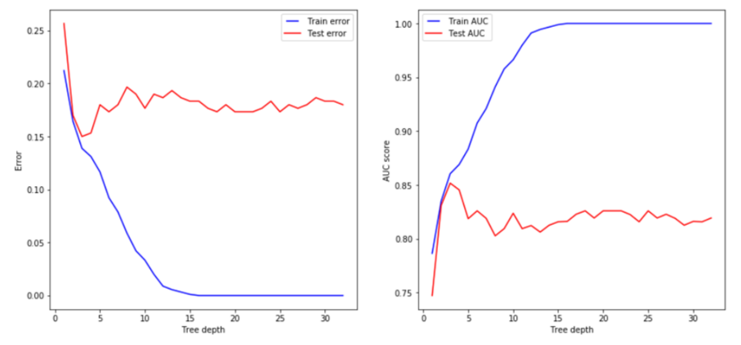
Limitations: Myopia

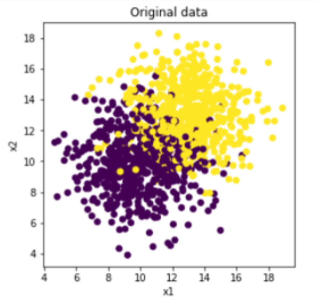
- Decision trees suffer from myopia
- They cannot see further beyond the current node
- Therefore, in practice, it is recommended to "keep trying" and let the tree grow
- Once the tree has grown, tree pruning is done

Example: XOR solution

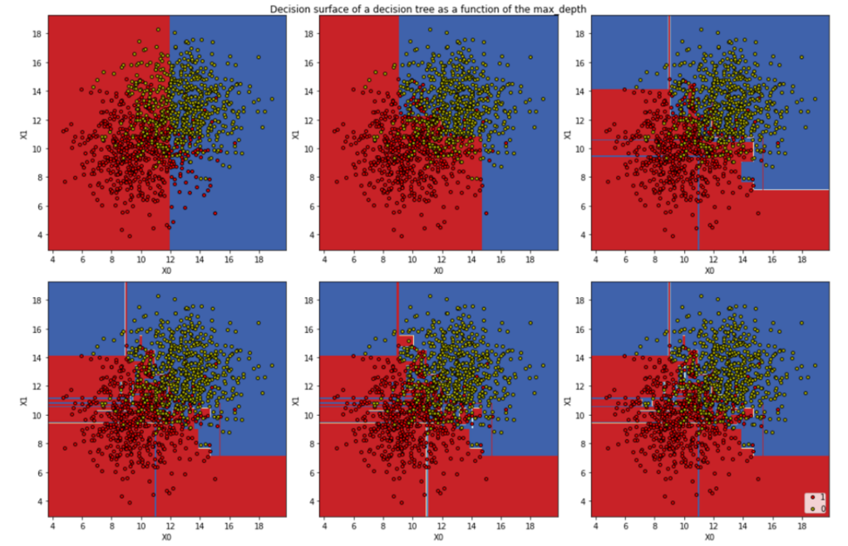


Limitations: Overfitting





Limitations: Overfitting



Overfitting: How to fix it

- Must use tricks to find simple trees
- Fixed depth/Early stopping
- Pruning
- Use ensembles of different trees: Random forests

CART: Classification and Regression Trees

Decision trees can also be used for regression, i.e. $y \in \mathbb{R}$. In such a case, the impurity needs to change

Impurity: Squared Loss

$$L(R) = \frac{1}{|R|} \sum_{(\mathbf{x},y)\in R} (y - \bar{y}_R)^2 \tag{6}$$

where R denotes a region

$$\bar{y}_R = \frac{1}{|R|} \sum_{(\mathbf{x},y)\in R} y$$

is the average label of the region R.

CART: Classification and Regression Trees

Decision trees can also be used for regression, i.e. $y \in \mathbb{R}$. In such a case, the impurity needs to change

Impurity: Squared Loss

$$L(R) = \frac{1}{|R|} \sum_{(\mathbf{x},y)\in R} (y - \bar{y}_R)^2 \tag{6}$$

where R denotes a region

$$\bar{y}_R = \frac{1}{|R|} \sum_{(\mathbf{x},y)\in R} y$$

is the average label of the region R.

Important: Finding the best binary partition in terms of minimum sum of squares is generally computationally infeasible. A greedy approach is preferred, using the same principle as for classification.

Wrap-up

Wrap-up

- We introduced decision trees
- These are light-weight classifiers that can be very fast
- They are not competitive in accuracy and prone to overfitting
- Using ensemble techniques allows them to become strong and competitive

Key Concepts

- Gini Impurity
- Entropy
- KL-Divergence
- Prunning
- Decision trees

References

Further Reading and Useful Material

Source	**Notes**
The Elements of Statistical Learning	Sec. 9, 10, 15, 16