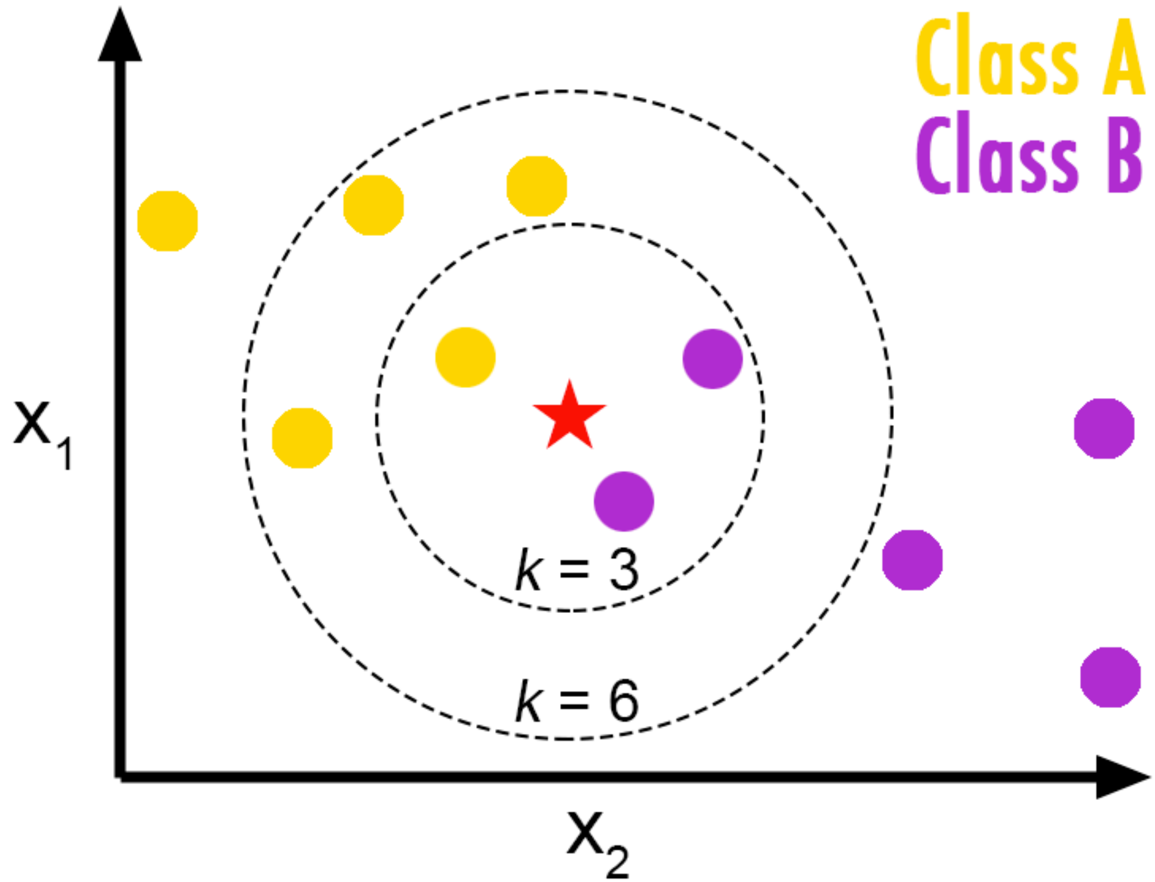
k-Nearest Neighbors
&
Decision Trees

Outline

- k-Nearest Neighbors (kNN)
  - Algorithm
  - Distance metrics
  - kNN tradeoffs

- Decision Trees
  - How to build a decision tree
  - Information gain vs. gini impurity
  - Decision tree tradeoffs
  - Pruning

k-Nearest Neighbors (kNN)



Distance Metrics

- Euclidean distance

$$dist(a,b) = ||a-b|| = \sqrt{(a-b)\cdot(a-b)} = \sqrt{\sum_i (a_i - b_i)^2}$$

- Cosine similarity

$$dist(a,b) = \frac{a \cdot b}{||a||||b||} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Pseudocode - kNN

For datapoint in training set:

    calculate distance from datapoint to new_value

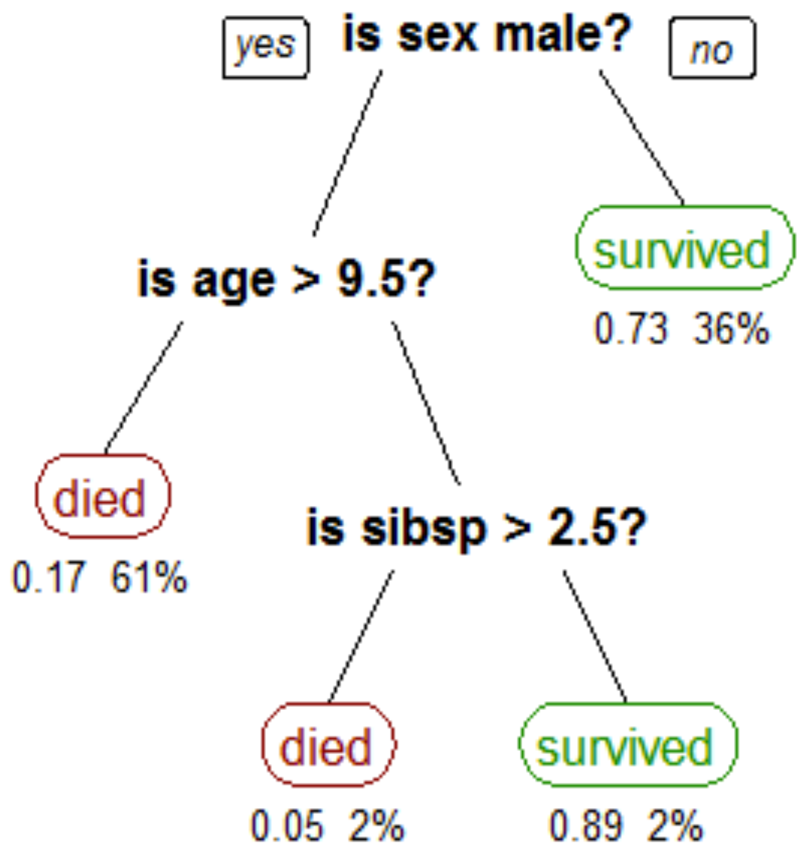
Order distances in increasing order and take the first k

Take the label with the most votes

kNN Tradeoffs

- Pros
  - Really easy to train (just save all the data)
  - Easily works with any number of classes
  - Easy to add new training datapoints
- Con
  - Really slow to predict (especially if you have a lot of features)

Decision Trees



How to Build a Decision Tree

- To do a binary split:
  - For a categorical variable, choose either value or not value (e.g. sunny or not sunny)
  - For a continuous variable, choose a threshold and do > or <= the value (e.g. temperature <75 or >=75)
- To measure how good the split is:
  - Information gain
  - Gini impurity

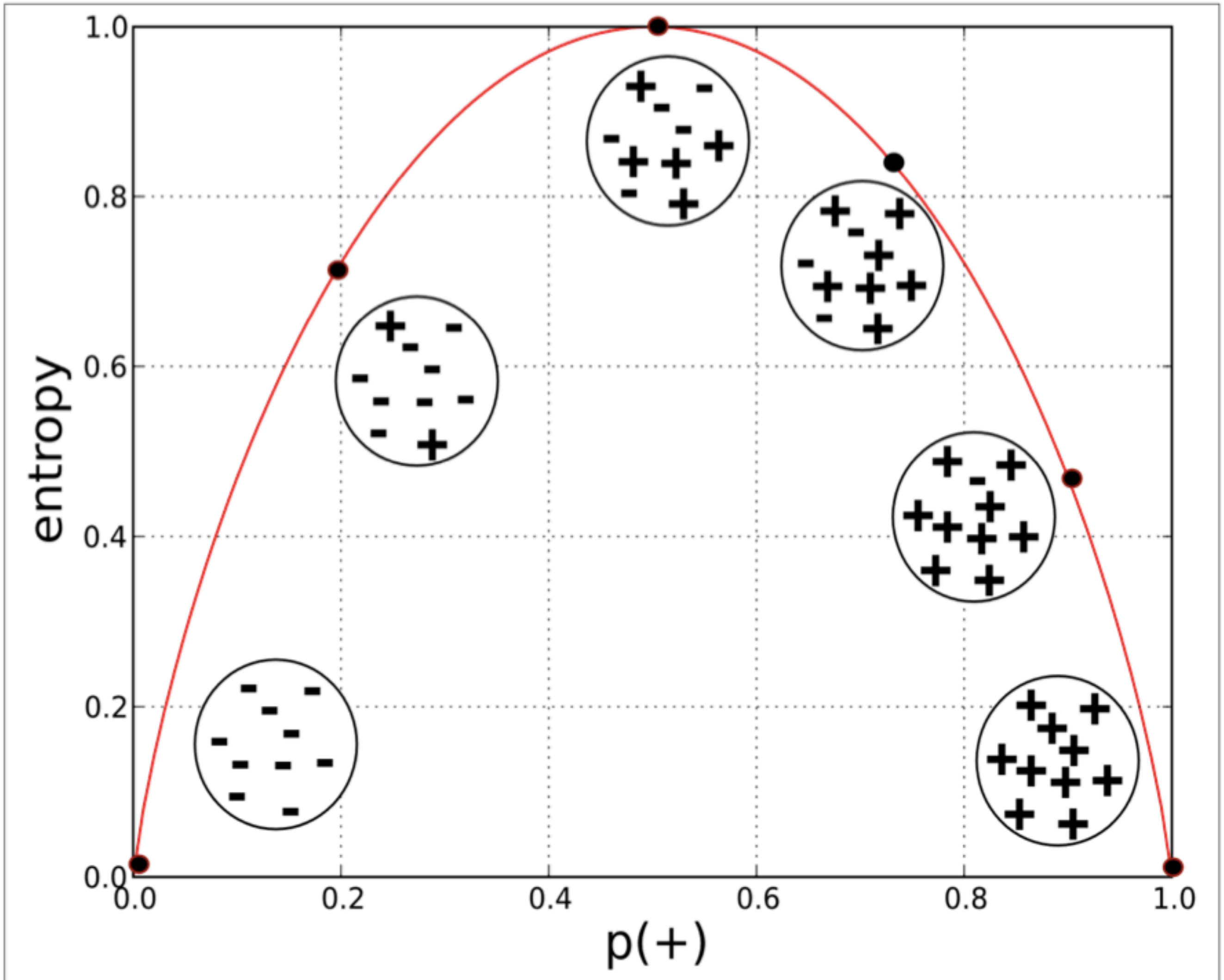
Entropy

- Entropy – a measure of the amount of disorder in a set

$$H(y) = -\sum_{i=1}^{m} P(c_i) \log_2 (P(c_i))$$

P(c) is the percent of the group that belongs to a given class

More on Entropy



Information Gain

$$Gain(S, D) = H(S) - \sum_{V \in D} \frac{|V|}{|S|} H(V)$$

S is the original set and D is the splitting of the set (a partition), each V is a subset of S

Gini Impurity

- It is a measure of this probability:
  - Take random element from the set
  - Label it randomly according to the distribution of labels in the set
  - What is the probability that is it labeled incorrectly?

$$Gini = \sum_{i=1}^{m} P(c_i)(1 - P(c_i)) = 1 - \sum_{i=1}^{m} P(c_i)^2$$

Pseudocode – Decision Trees

```
function BuildTree:
    If every item in the dataset is in the same class or there is
    no feature left to split the data:
        return a leaf node with the class label
    Else:
        find the best feature and value to split the data
        split the dataset
        create a node
        for each split
            call BuildTree and add the result as a child of the node
        return node
```

Decision Tree Tradeoffs

- Pros
  - Easily interpretable
  - Handles missing values and outliers
  - Non-parametric/non-linear/model complex phenomenon
  - Computationally cheap to predict
  - Can handle irrelevant features
  - Mixed data (discrete and continuous)

- Cons
  - Computationally expensive to train
  - Greedy algorithm (local optima)
  - Very easy to overfit

Pruning - Prepruning

- Making the decision tree algorithm stop early
  - Leaf size: stop when the number of data points for a leaf gets below a threshold
  - Depth: stop when the depth of the tree (distance from root to leaf) reaches a threshold
  - Mostly the same: stop when some percent of the data points are the same (rather than all the same)
  - Error threshold: stop when the error reduction (information gain) is not improved significantly

Pruning - Postpruning

- Involves building the tree first and then choosing to cut off some of the leaves
- Psuedocode:

```
function Prune:
    if either left or right is not a leaf:
        call Prune on that split
    if both left and right are leaf nodes:
        calculate error associated with merging two nodes
        calculate error associated without merging two nodes
        if merging results in lower error:
            merge the leaf nodes
```

Decision Tree Variants

- ID3 (Iterative Dichotomiser 3)
  - Designed for only categorical features
  - Splits categorical features completely
  - Uses entropy and information gain to pick the best split
- CART (Classification and Regression Tree)
  - Handles both categorical and continuous data
  - Always uses binary splits
  - Uses gini impurity to pick the best split
- C4.5
  - Handles continuous data
  - Implements pruning to reduce overfitting
- C5.0
  - Propietary, do not have access to the specifics

In Practice

- We always implement pruning to avoid overfitting
- Either gini or information gain is acceptable
- Sometimes fully splitting categorical features is preferred, but generally we air on the side of binary splits

sklearn

- Pruning with max_depth, min_samples_split, min_samples_leaf or max_leaf_nodes
- Gini is default, but you can also choose entropy
- Does binary splits (you would need to binarize categorical features)

Regression Trees

- Decisions trees can also be used for regression
- Can average the values at each leaf node to predict a continuous value
- Can also use a combination of decision trees and linear regression on the leaf nodes (model trees)