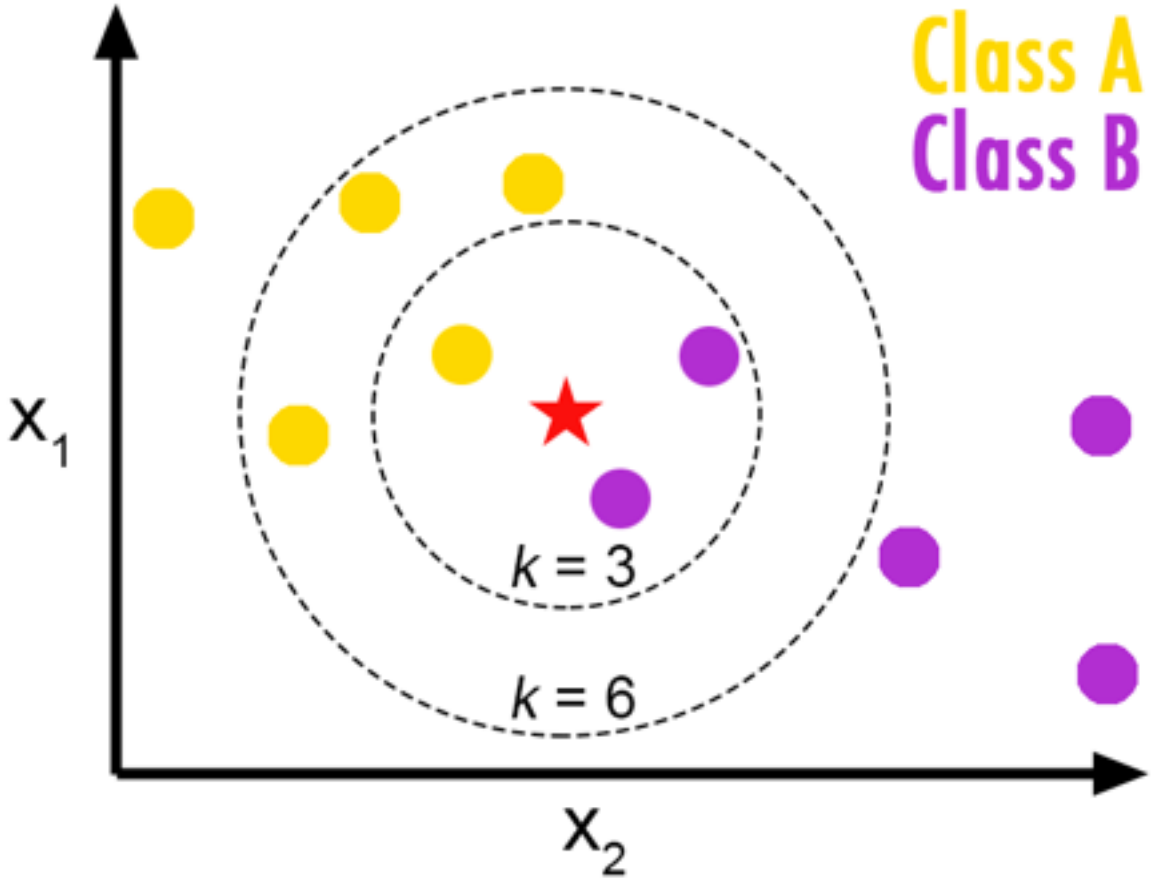
k-Nearest Neighbors
&
Decision Trees

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

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

- Others: Manhattan distance (L^1 norm), L^p norm, L^infinity norm, etc.

Algorithm Summary - kNN

- Find kth nearest neighbors to point of interest
- Count how many of those neighbors are of each class
- Classify the point of interest as the primary class

Psuedocode:

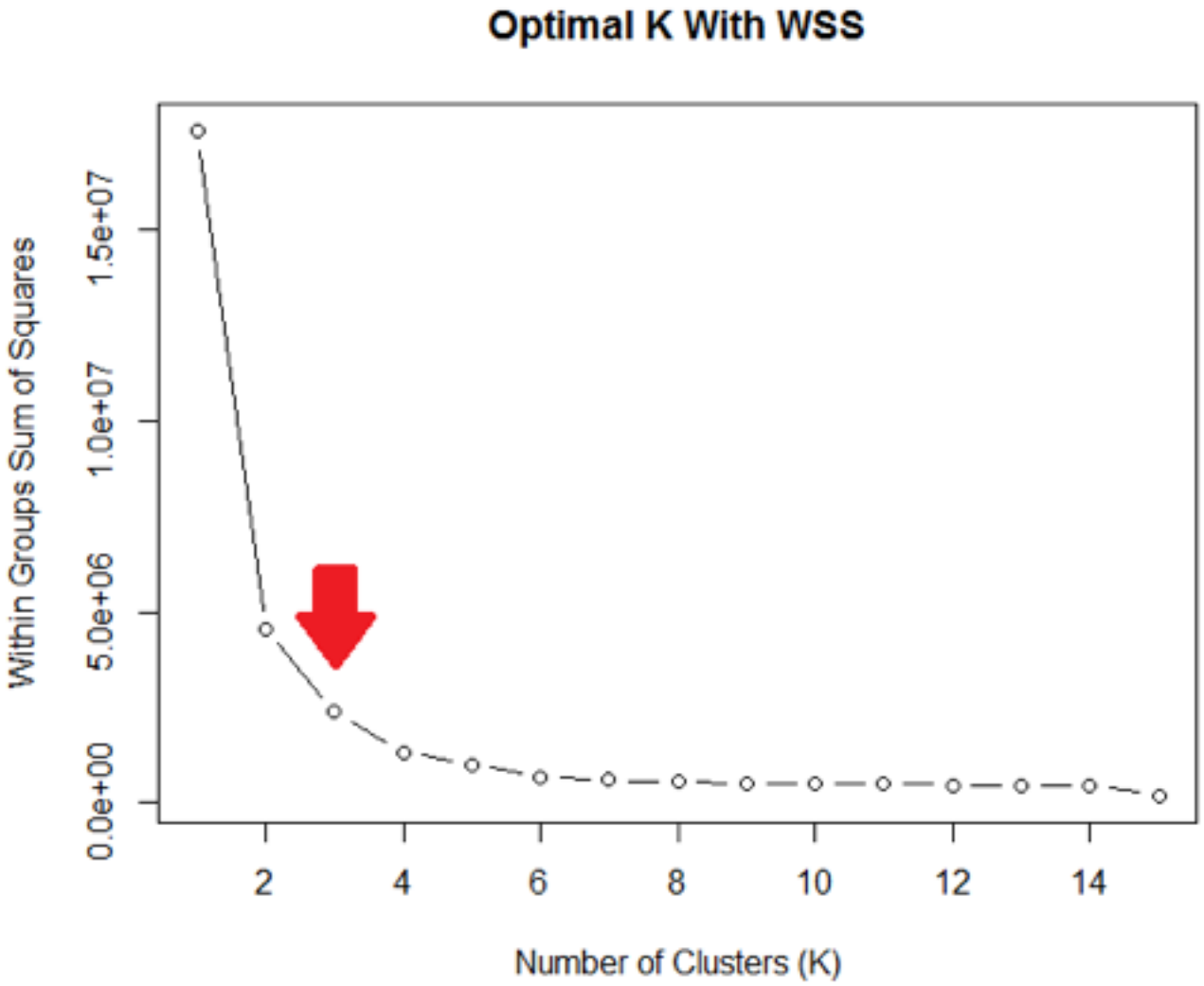
For each datapoint in training set:

    Calculate distance from datapoint to new_value

Order distances in increasing order and take the first k

Assign most frequent label to new_value

How to choose k?
Cross-Validation (k-fold)



kNN Pros

- Can learn a complex function
- No relationships needed between variables (linearity, etc)
- Works with any number of classes
- Easy to store model (data + distance metric)
- Easy to add new training points

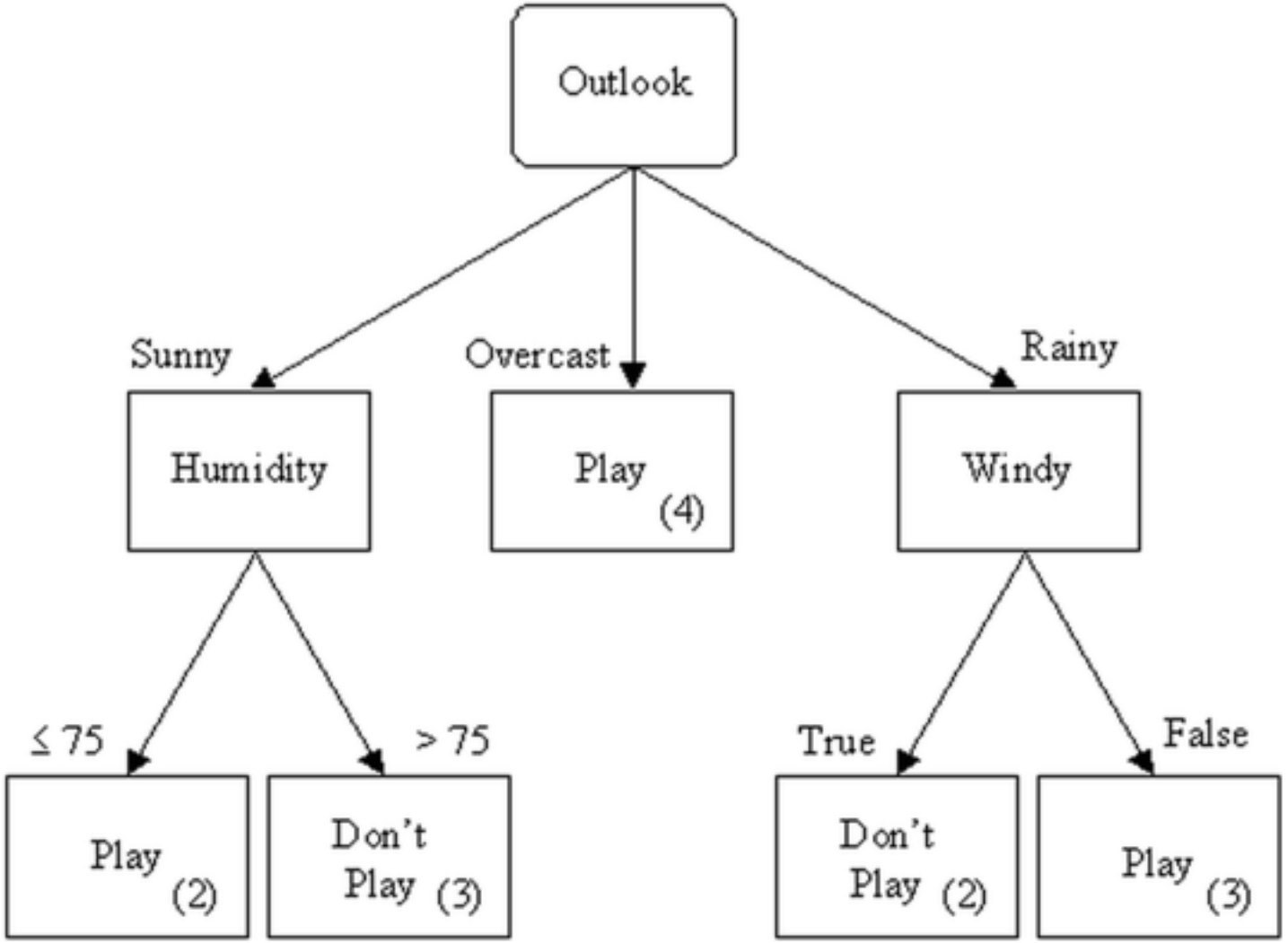
kNN Cons

- Slow to predict
- No way to include a loss function
- Difficulties with imbalanced classes
- Irrelevant attributes can affect results
- Curse of dimensionality*

Uses of k-NN

- Classification
- Imputation
  - Replace missing values with k-NN
  - http://nerds.airbnb.com/overcoming-missing-values-in-a-rfc
  - http://www.icmc.usp.br/~gbatista/files/his2002.pd
- Anomaly Detection
  - outlier has large distances to nearest neighbors
- Regression
  - assign mean value of k nearest neighbors

Decision Trees



Why Decision Trees

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

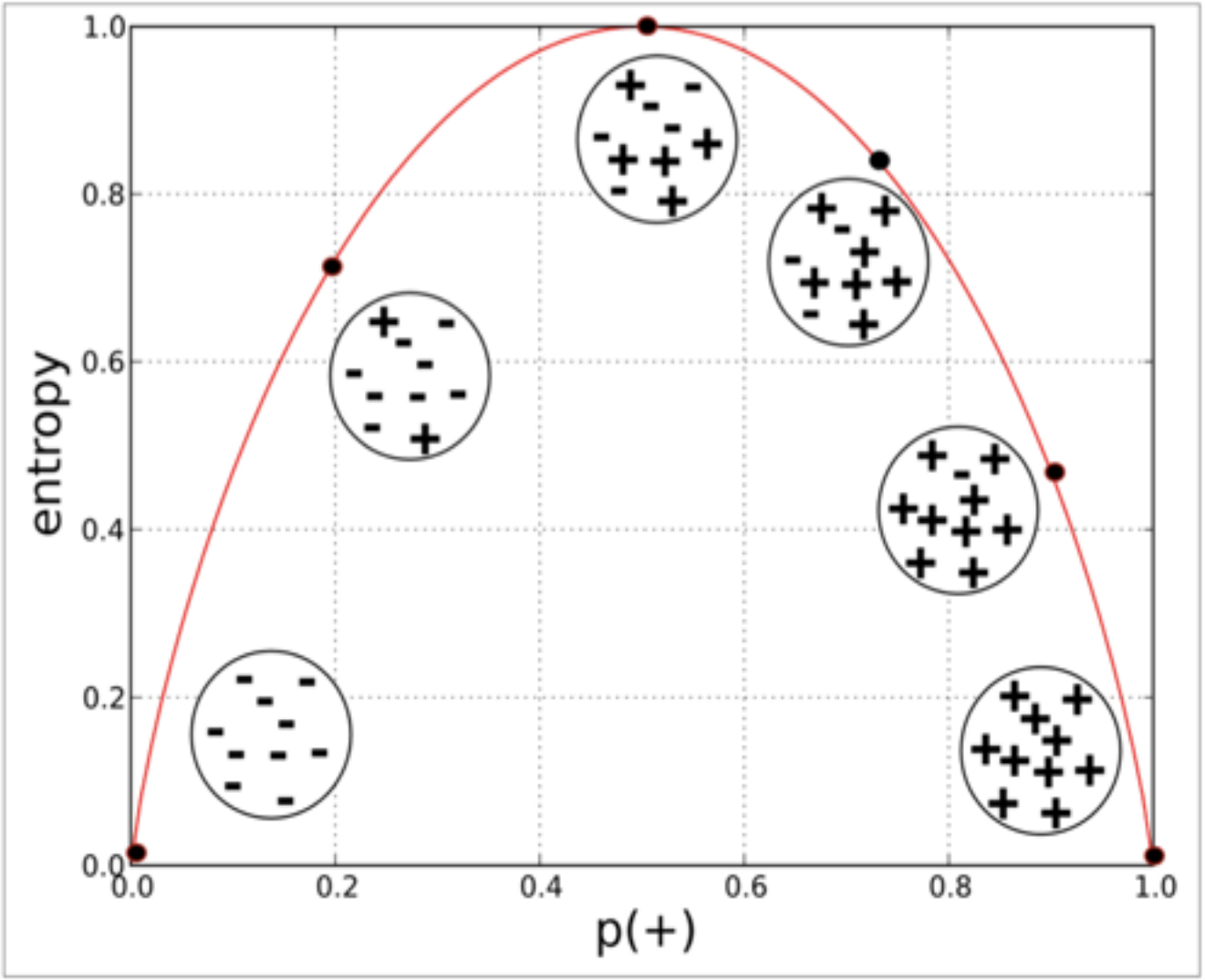
Why Not Decision Trees

- Computationally *expensive* to ***train***
- Very easy to overfit
- Greedy algorithm (local optima)
  - Algorithm makes best splits

How to Build a Decision Tree

- For binary split tree:
  - 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

- 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

Information Gain

Entropy(parent) - Avg(entropy(children))

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

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

Information gain example:
http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

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
```

Pruning - Prepruning

- Making the 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 - CV

- 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
```

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 err 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 Decision Trees

- Can’t use entropy/information gain/gini impurity
- To choose best splits use residual sum of squares (calculate against mean value of each leaf)
- Can also use a combination of decision trees and linear regression on the leaf nodes (model trees)