Non-Parametric-Learners

Brian J. Mann

November 9, 2015

Objectives

1. Know how to build a k-Nearest-Neighbor (kNN) classifier.
2. Know what a decision tree is and how to create one.
3. Learn a few different ways to build trees and what the concept of *information gain* is.

k-Nearest-Neighbors (kNN)

kNN

- +Simple to train (just save your data)
- +Works with any number of classes
- +Easy to add new data to the model
- -Takes a long time to predict new labels

Algorithm

1. Compute the distance from your un-labeled new data point to each point in your training set.
2. Sort by distance.
3. Take the k-closest and choose the most common label.

In pseudo-python:

```
for point, label in training_set:
    distances.append( (label,
                      distance(point, new_point)) )

distances.sort(key=lambda x: x[1])
closest = distances[:k]
return Counter(closest).most_common(1)
```

Distances

Most common distance metrics are Euclidean and Cosine Similarity:

$$\sqrt{\sum_i (x_i - y_i)^2}$$

or

$$\arccos \frac{x \circ y}{||x||||y||}$$

Examples

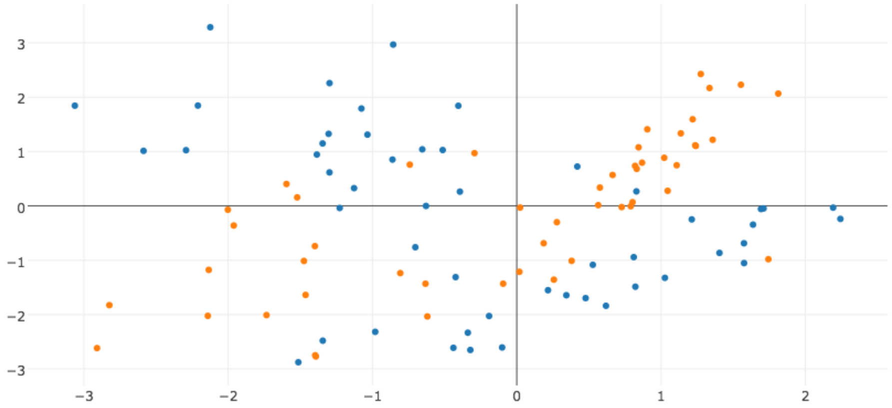

Figure 1:Data

Examples

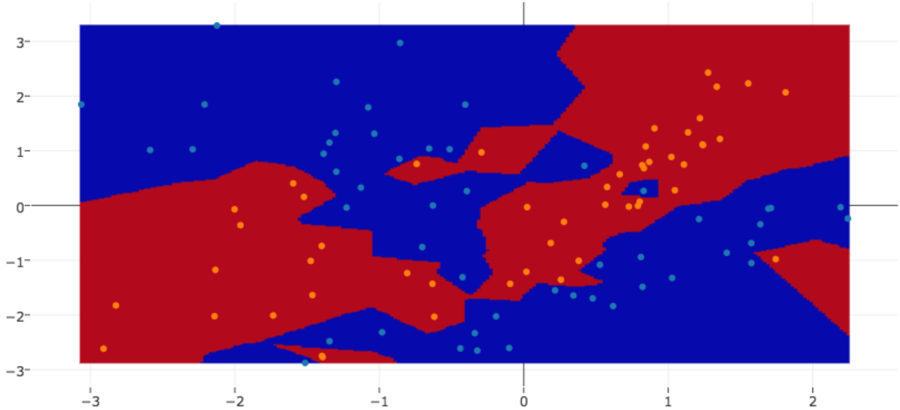

Figure 2: $k = 1$

Examples

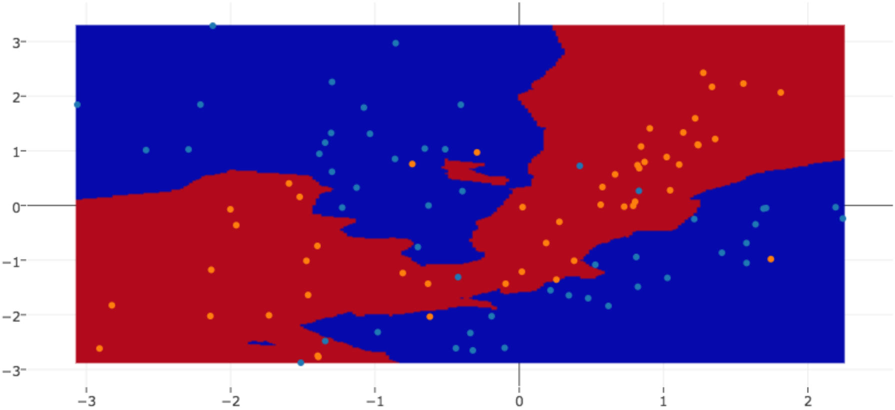

Figure 3: $k = 3$

Examples

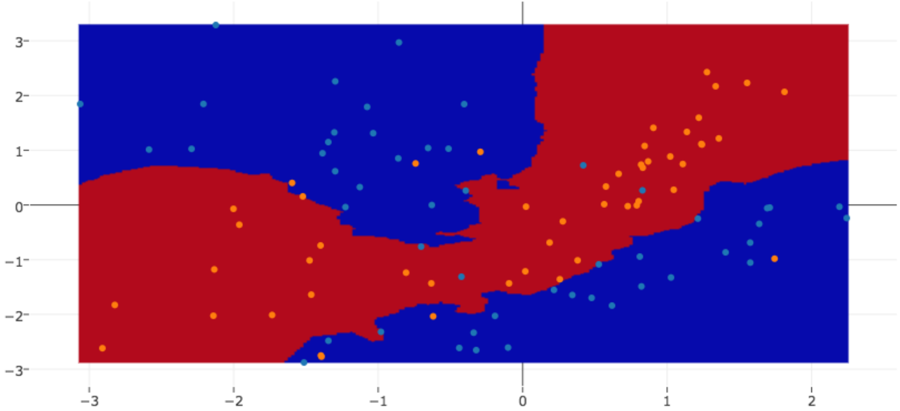

Figure 4: $k = 5$

Questions

1. What are some problems/edge cases you can see with the algorithm as written?
2. How do you know which k to choose?

Decision Trees

Why Decision Trees?

- Popular, flexible models.
- Easy to interpret and explain results.
- Non-parametric and non-linear => can model more complex things.
- Computationally easy to predict.
- Deals easily with irrelevant features.
- Works with mixed (catergorical and continuous) features.

Why Not?

- Computationally slow to train.
- Easy to overfit.

How to build a decision tree

- How to determine which feature to split on or where to split (in the case of continuous features)?
- To start, only look at binary splits = splitting a feature into two parts
  - Value or not-value (categorical)
  - Value $<$ threshold or value $\geq$ threshold (continuous)

Information gain

- In order to understand the concept of which split is "best," understand *entropy* and *information gain*.
- This gives a way to quantitatively measure which feature is best to split on at each step of the tree.

Entropy

- Measure of "disorder" in data.
- Dataset S labeled as classes (or labels) $c_1, \ldots, c_m$.
- The proportion of the data points belonging to the class c_i is $P(c_i)$.
- The *entropy* of the labeled dataset is:

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

.

- Entropy is low when the data belongs mostly to one category, and high when it's "mixed up."

Information Gain

- S is the original set and D is a partition:

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

.

- Want to maximize $Gain(S, D)$ at each step by choosing the best feature and value to split on.

Gini Impurity

- Another way of measuring which split is best
  - Take a random element from S.
  - Label it randomly according the proportions of the labels.
  - Gini Impurity is the probability it was labeled incorrectly.

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

.

How to build a tree

```
def build_tree(data):
    if all observations are in the same class or
    no more features left to split on:
        return a leaf node with the most popular class label
    else:
        find the best split and value
        create a node
        for each subset in the split:
            call build_tree and add result as child node
        return node
```

Pruning

- Prone to overfitting.
- Prepruning and postpruning can help.

Prepruning

- Make the algorithm stop early:
  - Min leaf size
  - Max depth
  - Stop when some % of the data are labeled the same in a split
  - Stop when information gain stops increasing enough

Postpruning

- Build the tree first and then cut off some leaves

```
def prune(node):
    if left or right is not a leaf:
        call prune on non-leaf node
    else:
        calculate error associated with merging the two nodes
        and without
        if error decreases with merging:
            merge left and right
```

Different ways to build trees

- Lots of choices to make:
  - Binary or full splits?
  - Entropy or Gini?
  - Prune?
- Different algorithms make different choices and some have names.

ID3

- Iterative Dichotomiser 3
- Ross Quinlan, 1980's
  - Only categorical features
  - Splits features completely
  - Entropy and information gain

CART

- Breiman, Friedman, Olshen, Stone, 1980's
  - Categorical and continuous features
  - Always binary splits
  - Gini Impurity
- CART is now a catch-all term for tree algorithms

C4.5

- Quinlan improvement to ID3
  - Continuous features ok now
  - Pruning to reduce overfitting

In practice

- Always prune.
- Entropy or Gini Impurity OK.
- Mostly just do binary splits.