Decision Tree and KNN

Risman Adnan Mattotorang, Ph.D.

Last Week Material

LINEAR BASIS FUNCTION MODEL FOR CLASSIFICATION

LINEAR DISCRIMINANT MODEL FOR REGRESSION

What We Will Learn

Decision Tree Model

Main Concepts

Hypothesis Set

Learning Algorithm

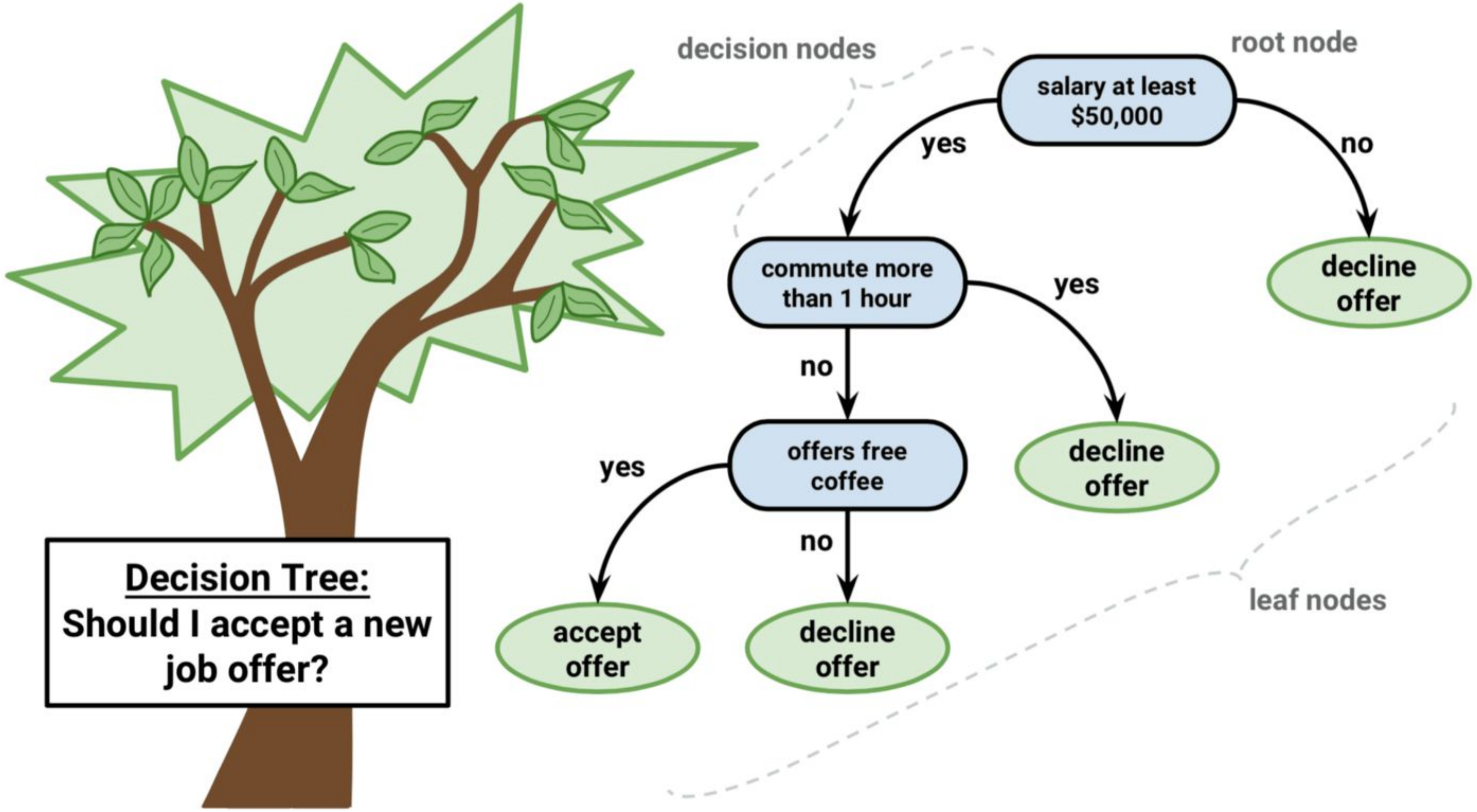
K-Nearest Neighbor Model

Main Concepts

Hypothesis Set

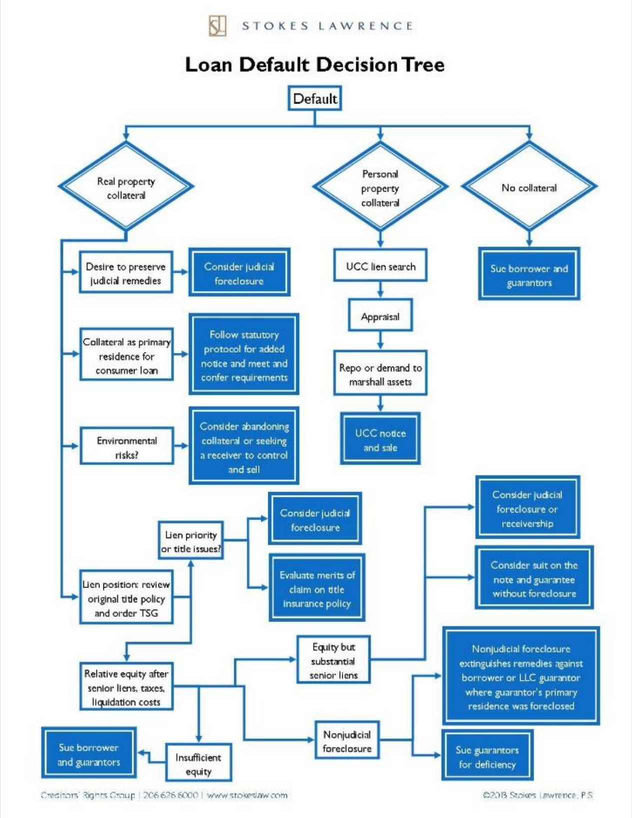
Learning Algorithm

Decision Tree



- Intuitive appeal for users
- Presentation Forms
  - "if, then" statements (decision rules)
  - graphically - decision trees
- Works like a flow chart
- Looks like an upside down tree
- Nodes represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes
- Top or starting node- root node
- Internal nodes - rectangles

Example of Decision Tree



- Bank - loan application
- Classify application
  - approved class
  - denied class
- Criteria - Target Class approved if 3 binary attributes have certain value:
  - Borrower has good credit history (credit rating in excess of some threshold)
  - Loan amount less than some percentage of collateral value (e.g., 80% home value)
  - Borrower has income to make payments on loan
- Possible scenarios = $3^2 = 8$
  - If the parameters for splitting the nodes can be adjusted, the number of scenarios grows exponentially.

How It Works

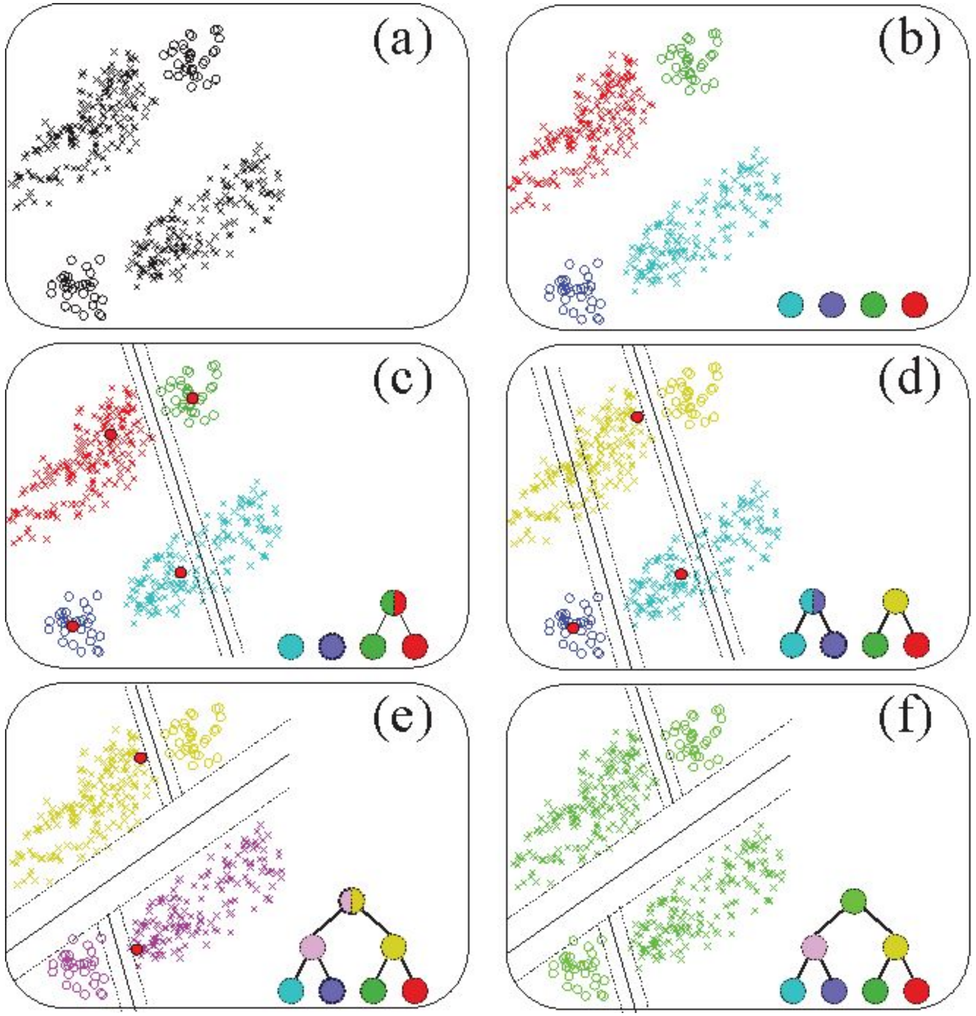
- Decision rules - partition sample of data
- Terminal node (leaf) indicates the class assignment
- Tree partitions samples into mutually exclusive groups
- One group for each terminal node
- All paths
  - start at the root node
  - end at a leaf
- Each path represents a decision rule
  - joining (AND) of all the tests along that path
  - separate paths that result in the same class are disjunctions (ORs)
- All paths - mutually exclusive
  - for any one case - only one path will be followed
  - false decisions on the left branch
  - true decisions on the right branch

Disjunctive Normal Form

- Non-terminal node - model identifies an attribute to be tested
  - test splits attribute into mutually exclusive disjoint sets
  - splitting continues until a node - one class (terminal node or leaf)
- Structure - *disjunctive normal form*
  - limits form of a rule to conjunctions (adding) of terms
  - allows disjunction (or-ing) over a set of rules

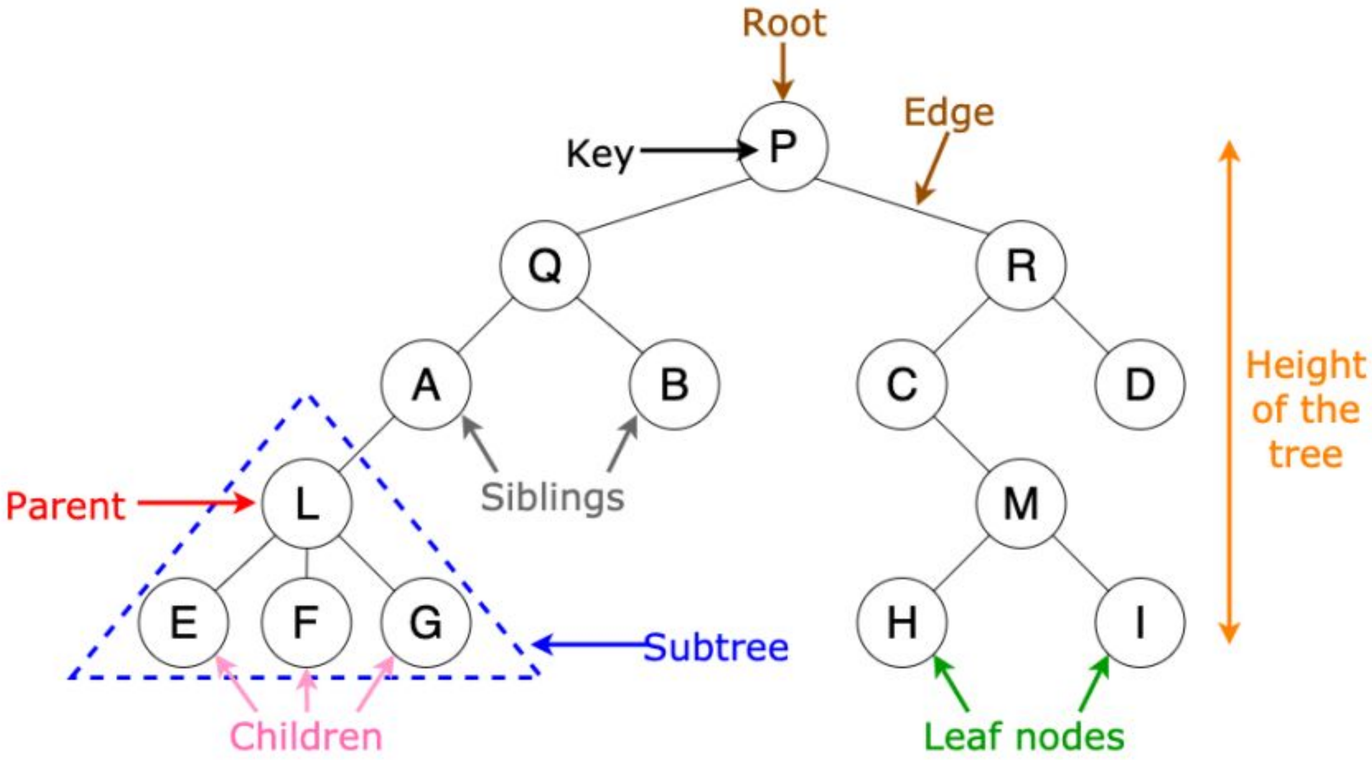
Geometry

- Disjunctive normal form
- Fits shapes of decision boundaries between classes
- Classes formed by lines parallel to axes
- Result - rectangular shaped class regions



Binary Trees

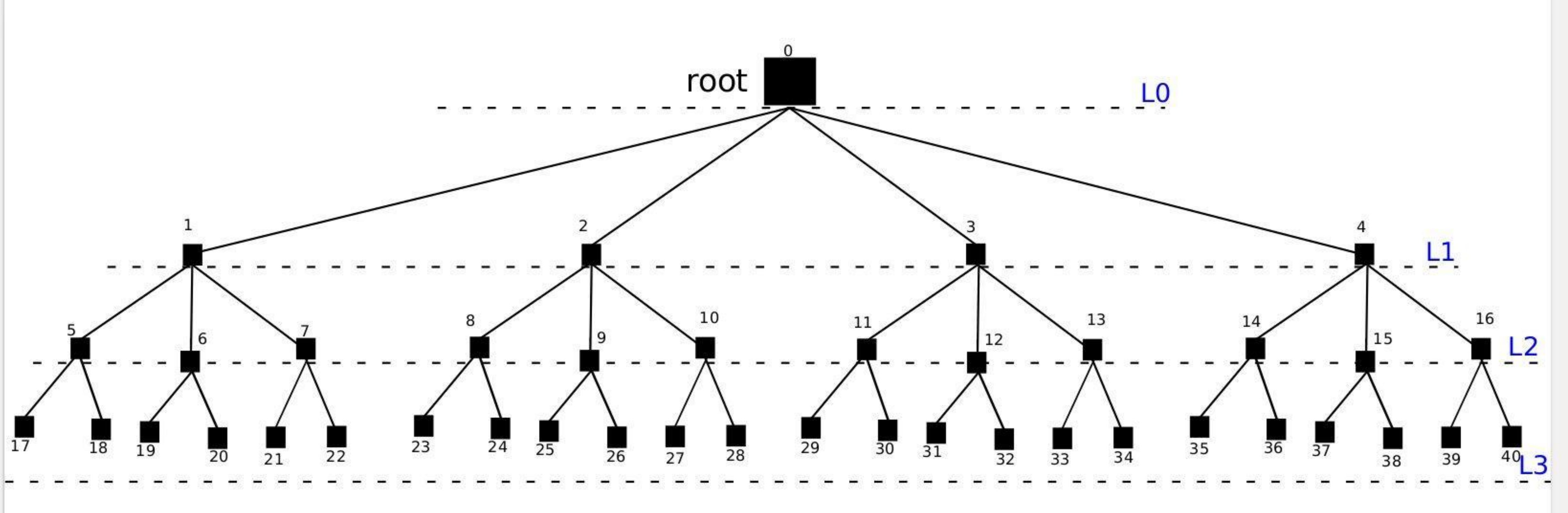
- Characteristics
  - two branches leave each non-terminal node
  - those two branches cover outcomes of the test
  - exactly one branch enters each non-root node
  - there are n terminal nodes
  - there are n-1 non-terminal nodes



Non-Binary Trees

Characteristics

- two or more branches leave each non-terminal node
- those branches cover outcomes of the test
- exactly one branch enters each non-root node
- there are n terminal nodes
- there are n-1 non-terminal nodes



The Goal

- Dual goal - Develop tree that
  - is small
  - classifies and predicts class with accuracy
- Small size
  - a smaller tree more easily understood
  - smaller tree less susceptible to overfitting
  - large tree less information regarding classifying and predicting cases

Rule Induction

Process of building the decision tree or ascertaining the decision rules

- tree induction
- rule induction
- induction

Decision tree algorithms

- induce decision trees recursively
- from the root (top) down - *greedy* approach
- established basic algorithms

Discrete and Continuous Attributes

Continuous variables attributes - problems for decision trees

increase computational complexity of the task

promote prediction inaccuracy

lead to overfitting of data

Convert continuous variables into discrete intervals

“greater than or equal to” and “less than”

optimal solution for conversion

difficult to determine discrete intervals ideal

- size
- number

Making The Split

Models induce a tree by recursively selecting and subdividing attributes

random selection - noisy variables

inefficient production of inaccurate trees

Efficient models

examine each variable

determine which will improve accuracy of entire tree

problem - this approach decides best split without considering subsequent splits

Evaluating the Split

Measures of impurity or its inverse, goodness reduce impurity or degree of randomness at each node popular measures include:

- Entropy Function
- Gini Index
- Twoing Rule

Overfitting

- Error rate in predicting the correct class for new cases
  - overfitting of test data
  - very low apparent error rate
  - high actual error rate

Optimal Size

- Certain minimal size smaller tree
  - higher apparent error rate
  - lower actual error rate
- Goal
  - identify threshold
  - minimize actual error rate
  - achieve greatest predictive accuracy

Ending The Tree Growth

- Grow the tree until
  - additional splitting produces no significant information gain
  - statistical test - a chi-squared test
  - problem - trees that are too small
  - only compares one split with the next descending split

Pruning

- Grow large tree
  - reduce its size by eliminating or pruning weak branches step by step
  - continue until minimum true error rate
- Pruning Methods
  - *reduced-error* pruning
  - divides samples into test set and training set
  - training set is used to produce the fully expanded tree
  - tree is then tested using the test set
  - weak branches are pruned
  - stop when no more improvement

Pruning

Resampling

5 - fold cross-validation

80% cases used for training; remainder for testing

Weakest-link or *cost-complexity* pruning

trim weakest link (produces the smallest increase in the apparent error rate)

method can be combined with resampling

Advanced Decision Trees

- Multivariate or Oblique Trees
  - CART-LC - CART with Linear Combinations
  - LMDT - Linear Machine Decision Trees
  - SADT - Simulated Annealing of Decision Trees
  - OC1 - Oblique Classifier 1

Evaluating Decision Trees

- Method's Appropriateness
- Data set or type
- Criteria
  - accuracy - predict class label for new data
  - scalability
    - performs model generation and prediction functions
    - large data sets
    - satisfactory speed
  - robustness
    - perform well despite noisy or missing data
  - intuitive appeal
    - results easily understood
    - promotes decision making

Decision Tree Limitations

- No backtracking
  - local optimal solution not global optimal solution
  - *lookahead* features may give us better trees
- Rectangular-shaped geometric regions
  - in two-dimensional space
    - regions bounded by lines parallel to the x- and y- axes
  - some linear relationships not parallel to the axes

Conclusions

- <u>Utility</u>
  - analyze classified data
  - produce
  - accurate and easily understood classification rules
  - with good predictive value

<u>Improvements</u>

- Limitations being addressed
- multivariate discrimination - oblique trees
- data mining techniques

What We Will Learn

Decision Tree Model

Main Concepts

Hypothesis Set

Learning Algorithm

K-Nearest Neighbor Model

Main Concepts

Hypothesis Set

Learning Algorithm

Instance-Based Learning

Idea:

Similar examples have similar label.

Classify new examples like similar training examples.

Algorithm:

Given some new example x for which we need to predict its class y

Find most similar training examples

Classify *x* “like” these most similar examples

Questions:

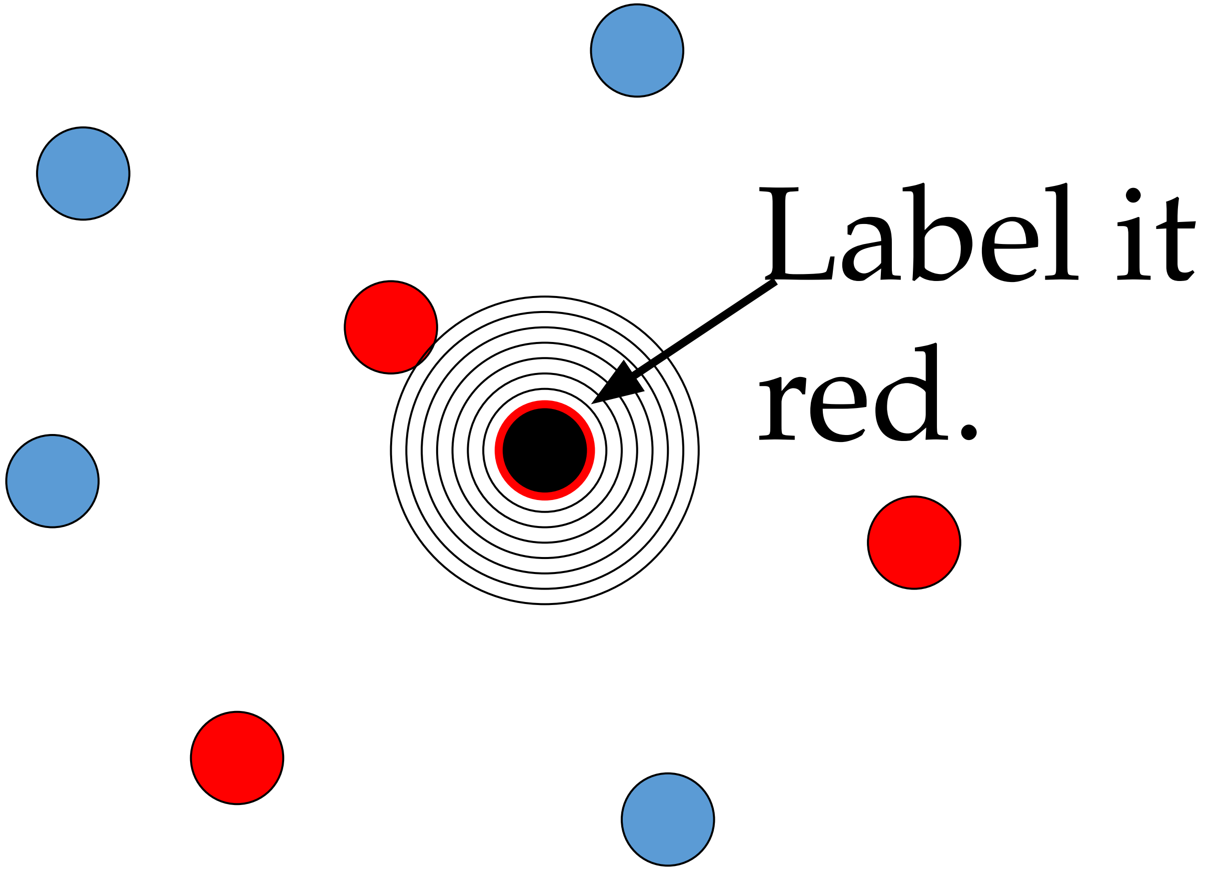
How to determine similarity?

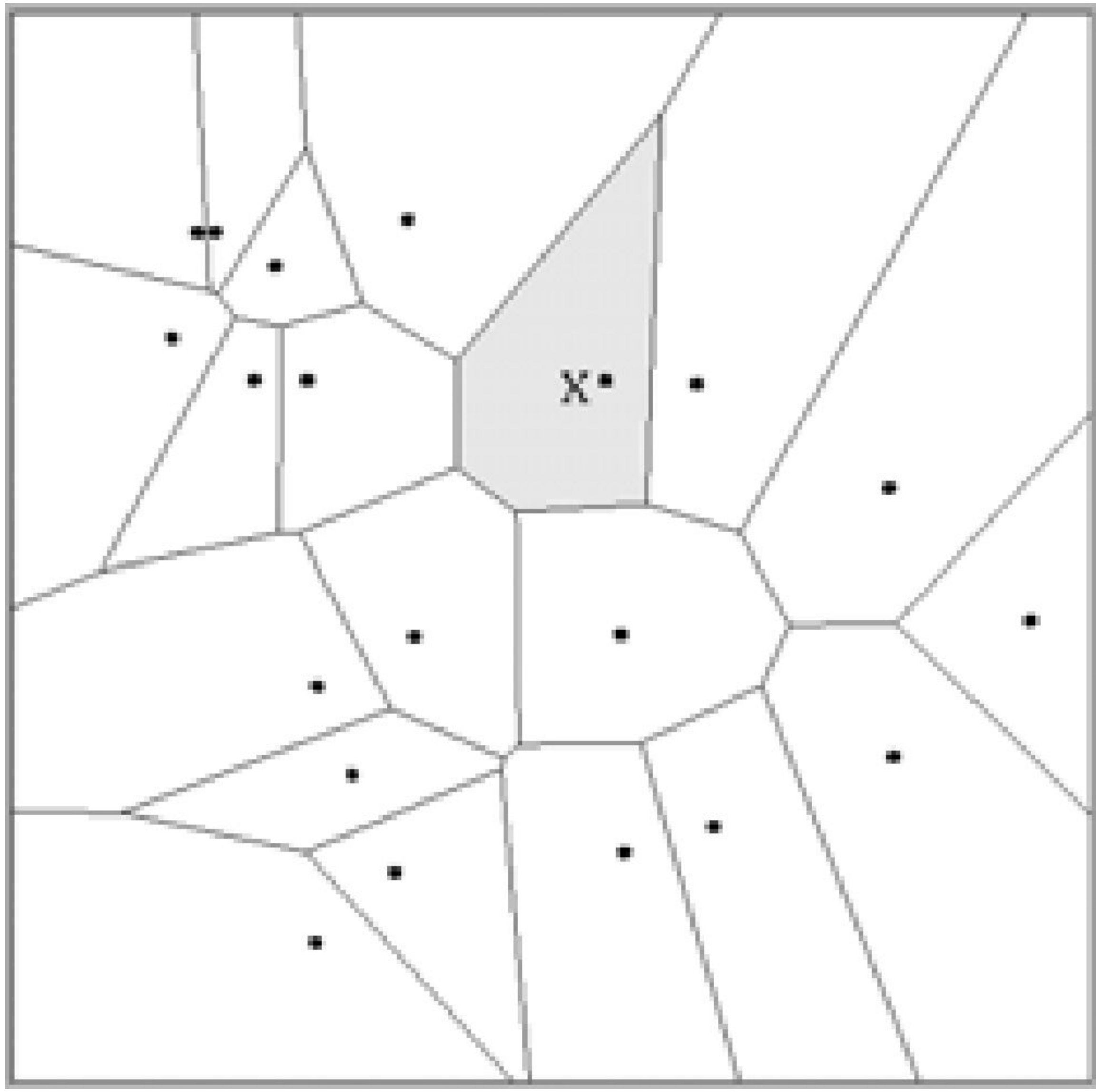
How many similar training examples to consider?

How to resolve inconsistencies among the training examples?

1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- Simple idea: label a new point the same as the closest known point





1-Nearest Neighbor

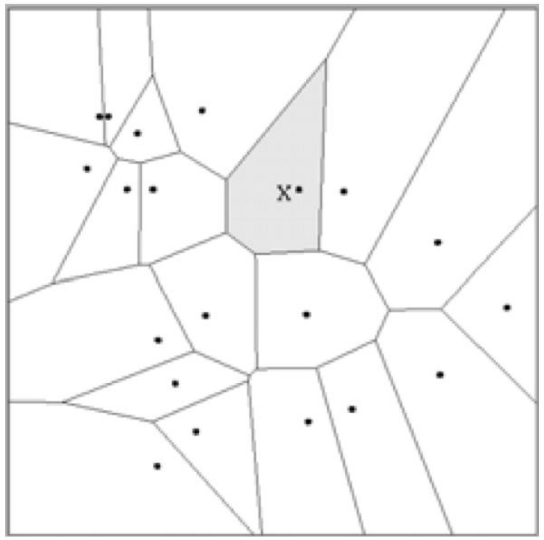
- A type of instance-based learning
  - Also known as “memory-based” learning
- Forms a Voronoi tessellation of the instance space

Distance Metrics

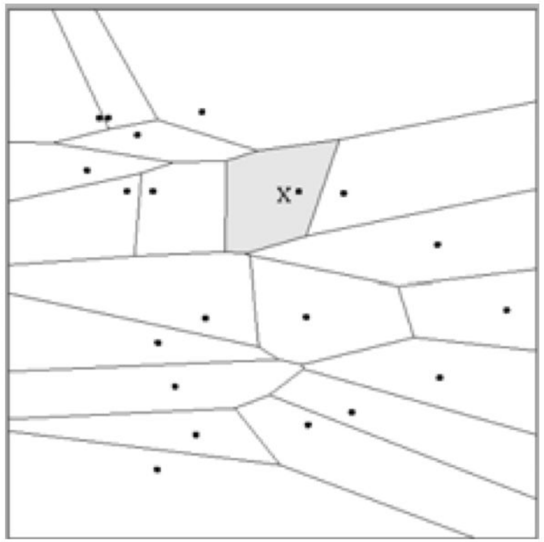
- Different metrics can change the decision surface
- Standard Euclidean distance metric:
  - Two-dimensional:

    $\text{Dist}(a,b) = \text{sqrt}((a_1 - b_1)^2 + (a_2 - b_2)^2)$
  - Multivariate:

    $\text{Dist}(a,b) = \text{sqrt}(\sum (a_i - b_i)^2)$



$\text{Dist}(\mathbf{a},\mathbf{b}) = (a_1 - b_1)^2 + (a_2 - b_2)^2$



$\text{Dist}(\mathbf{a},\mathbf{b}) = (a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

1-NN Aspects As Instance-Based Learning

A distance metric

- Euclidean
- When different units are used for each dimension
   normalize each dimension by standard deviation
- For discrete data, can use hamming distance
   D(x1,x2) =number of features on which x1 and x2 differ
- Others (e.g., normal, cosine)

How many nearby neighbors to look at?

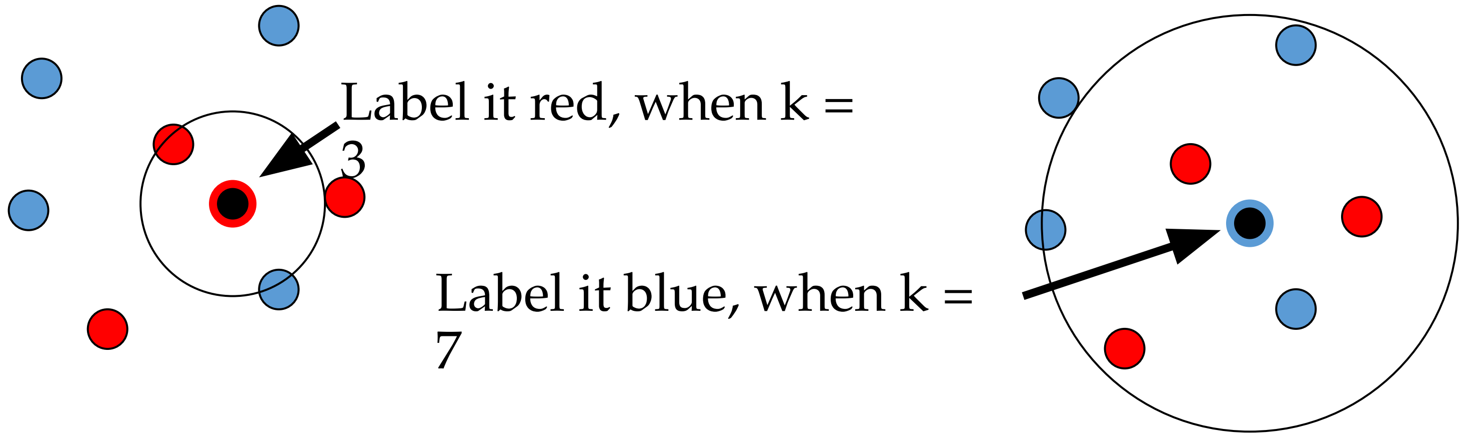
- One

How to fit with the local points?

- Just predict the same output as the nearest neighbor.

K-Nearest Neighbor

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its *k* nearest neighbors



KNN Example

Similarity metric: Number of matching attributes (k=2)

New examples:

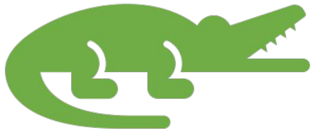
- Example 1 (great, no, no, normal, no)
    - □ most similar: number 2 (1 mismatch, 4 match) □ yes
    - □Second most similar example: number 1 (2 mismatch, 3 match) □ yes
- Example 2 (mediocre, yes, no, normal, no)
    - Most similar: number 3 (1 mismatch, 4 match) □ no
    - □Second most similar example: number 1 (2 mismatch, 3 match) □ yes

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Selecting Number of Neighbor

- Increase k:
  - Makes KNN less sensitive to noise
- Decrease k:
  - Allows capturing finer structure of space
- Pick k not too large, but not too small (depends on data)

Curse of Dimensionality



Prediction accuracy can quickly degrade when number of attributes grows.

Irrelevant attributes easily “swamp” information from relevant attributes

When many irrelevant attributes, similarity/distance measure becomes less reliable

Remedy

Try to remove irrelevant attributes in pre-processing step

Weight attributes differently

Increase k (but not too much)

Advantages and Disadvantages

Need distance/similarity measure and attributes that "match" target function.

For large training sets,

□ Must make a pass through the entire dataset for each classification. This can be prohibitive for large data sets.

Prediction accuracy can quickly degrade when number of attributes grows.

Simple to implement algorithm;
Requires little tuning;
Often performs quite well!
(Try it first on a new learning problem).

Home Work

Decision-Tree Classifier Tutorial

Python · Car Evaluation Data Set

Notebook Data Logs Comments (19)

Run
14.2s

Version 4 of 4

Decision Tree Classifier Tutorial with Python

Hello friends,

In this kernel, I build a Decision Tree Classifier to predict the safety of the car. I build two models, one with criterion `gini index` and another one with criterion `entropy`. I implement Decision Tree Classification with Python and Scikit-Learn.

kNN Classifier Tutorial

Python · UCI_Breast Cancer Wisconsin (Original)

Notebook Data Logs Comments (18)

Run
20.0s

Version 5 of 5

kNN Classifier Tutorial in Python

Hello friends,

kNN or k-Nearest Neighbours Classifier is a very simple and easy to understand machine learning algorithm. In this kernel, I build a k Nearest Neighbours classifier to classify the patients suffering from Breast Cancer.

So, let's get started.

More to Understand:

- StatQuest PCA: https://www.youtube.com/watch?v=_UVHneBUBW0
- StatQuest KNN: https://www.youtube.com/watch?v=HVXime0nQeI
- StatQuest Decision Tree: https://www.youtube.com/watch?v=_L39rN6gz7Y&t=61s