Machine Learning: Introduction and Unsupervised Learning

Chapter 18.1, 18.2, 18.8.1
and "Introduction to Statistical Machine Learning"

1

What is Learning?

- *"Learning is making useful changes in our minds"* – Marvin Minsky
- *"Learning is constructing or modifying representations of what is being experienced"* – Ryszard Michalski
- *"Learning denotes changes in a system that ... enable a system to do the same task more efficiently the next time"* – Herbert Simon

3

Why do Machine Learning?

- Solve classification problems
- Learn models of data ("data fitting")
- Understand and improve efficiency of human learning (e.g., Computer-Aided Instruction (CAI))
- Discover new things or structures that are unknown to humans ("data mining")
- Fill in skeletal or incomplete specifications about a domain

4

Major Paradigms of Machine Learning

- Rote Learning
- Induction
- Clustering
- Discovery
- Genetic Algorithms
- Reinforcement Learning
- Transfer Learning
- Learning by Analogy
- Multi-task Learning

5

Inductive Learning

- Generalize from a given set of (***training***) examples so that accurate predictions can be made about ***future*** examples
- Learn unknown function: $f(x) = y$
  - x: an input **example** (aka **instance**)
  - y: the desired output
    - Discrete or continuous scalar value
  - h (hypothesis) function is learned that approximates f

6

Representing "Things" in Machine Learning

- An **example** or **instance**, ***x***, represents a specific object ("thing")
- ***x*** often represented by a *D*-dimensional **feature vector** ***x*** = (x_1, . . . , x_D)
- Each dimension is called a **feature** or **attribute**
- Continuous or discrete valued
- ***x*** is a point in the ***D*-dimensional feature space**
- Abstraction of object. Ignores all other aspects (e.g., two people having the same weight and height may be considered identical)

7

Feature Vector Representation

- Preprocess raw data
  - extract a feature (attribute) vector, ***x***, that describes all attributes relevant for an object
- Each ***x*** is a list of (*attribute*, *value*) pairs
  x = [(*Rank*, *queen*), (*Suit*, *hearts*), (*Size*, *big*)]
  - number of attributes is fixed: Rank, Suit, Size
  - number of possible values for each attribute is fixed (if discrete)
    Rank: 2, …, 10, jack, queen, king, ace
    Suit: diamonds, hearts, clubs, spades
    Size: big, small

8

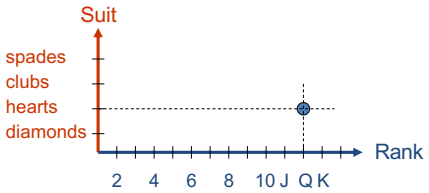
Types of Features

- **Numerical feature** has discrete or continuous values that are measurements, e.g., a person's weight
- **Categorical feature** is one that has two or more values (categories), but there is no intrinsic ordering of the values, e.g., a person's religion (aka **Nominal** feature)
- **Ordinal feature** is similar to a categorical feature but there is a clear ordering of the values,
  e.g., economic status, with three values: low, medium and high

9

Feature Vector Representation

Each example can be interpreted as a point in a *D*-dimensional feature space, where *D* is the number of features/attributes



10

Feature Vector Representation Example

- Text document
  - Vocabulary of size D (~100,000): aardvark, …, zulu
- "bag of words": counts of each vocabulary entry
  - To marry my true love ➔ (3531:1 13788:1 19676:1)
  - I wish that I find my soulmate this year ➔ (3819:1 13448:1 19450:1 20514:1)
- Often remove "stopwords:" the, of, at, in, …
- Special "out-of-vocabulary" (OOV) entry catches all unknown words

11

More Feature Representations

- Image
  - Color histogram
- Software
  - Execution profile: the number of times each line is executed
- Bank account
  - Credit rating, balance, #deposits in last day, week, month, year, #withdrawals, …
- Bioinformatics
  - Medical test1, test2, test3, …

12

Training Set

- A **training set** (aka **training sample**) is a collection of **examples** (aka **instances**), $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, which is the ***input*** to the learning process
- $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iD})$
- Assume these instances are all sampled *independently* from the same, unknown (population) distribution, $P(\boldsymbol{x})$
- We denote this by $\boldsymbol{x}_i \overset{\text{i.i.d.}}{\sim} P(\boldsymbol{x})$, where i.i.d. stands for **independent and identically distributed**
- Example: Repeated throws of dice

13

Training Set

- A training set is the "experience" given to a learning algorithm
- What the algorithm can learn from it varies
- Two basic learning paradigms:
  - ***unsupervised learning***
  - ***supervised learning***

14

Inductive Learning

- **Supervised** vs. **Unsupervised** Learning
  - supervised: "teacher" gives a set of (***x***, *y*) pairs
    - Training examples have *known* outcomes
  - unsupervised: only the ***x***'s are given
    - Training examples have *unknown* outcomes
- In either case, the goal is to estimate ***f*** so that it *generalizes well* to "correctly" deal with "future examples" in computing ***f***(***x***) = *y*
  - That is, find ***f*** that minimizes some measure of the error over a set of samples

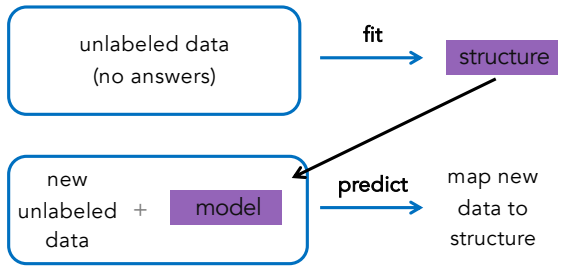
15

Unsupervised Learning

- Training set is $\mathbf{x}_1, \ldots, \boldsymbol{x}_n$, that's it!
- **No "teacher"** providing supervision as to how individual examples should be handled
- Common tasks:
  - **Clustering**: separate the *n* examples into groups
  - **Discovery**: find hidden or unknown patterns
  - **Novelty detection**: find examples that are very different from the rest
  - **Dimensionality reduction**: represent each example with a lower dimensional feature vector while maintaining key characteristics of the training samples

16

Unsupervised Learning Overview

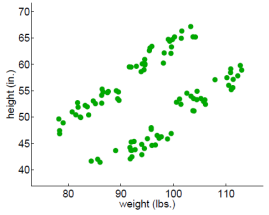


Slide by Intel Software

17

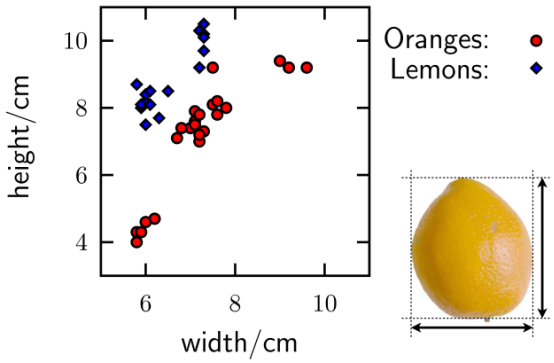
Clustering

- Goal: Group training sample into clusters such that examples in the same cluster are *similar*, and examples in different clusters are *different*
- How many clusters do you see?
- Many clustering algorithms



19

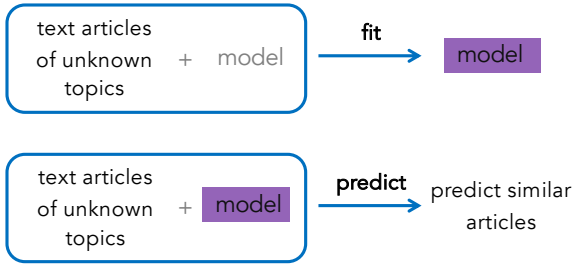
Oranges and Lemons



(from Iain Murray http://homepages.inf.ed.ac.uk/imurray2/)

20

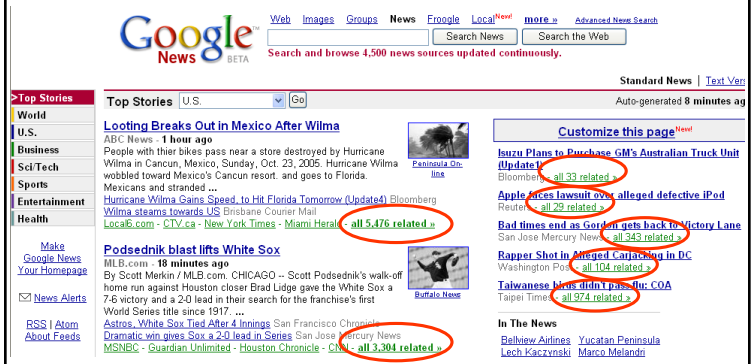
Clustering Application: Topic Modeling



Slide by Intel Software

21

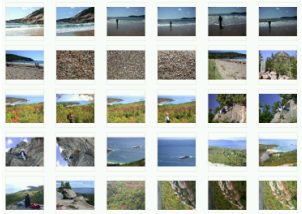
Google News



22

Digital Photo Collections

- You have 1000s of digital photos stored in various folders
- Organize them better by grouping into clusters
  - Simplest idea: use image creation time (EXIF tag)
  - More complicated: extract image features



23

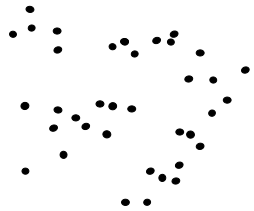
Three Frequently Used Clustering Methods

- **Hierarchical Agglomerative Clustering**
  - Build a binary tree over the dataset by repeatedly *merging* clusters
- **K-Means Clustering**
  - Specify the desired number of clusters and use an iterative algorithm to find them
- **Mean Shift Clustering**

29

Hierarchical Agglomerative Clustering

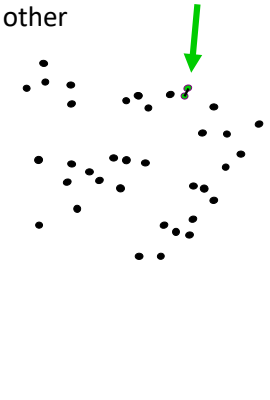
- Initially every point is in its own cluster



30

Hierarchical Agglomerative Clustering

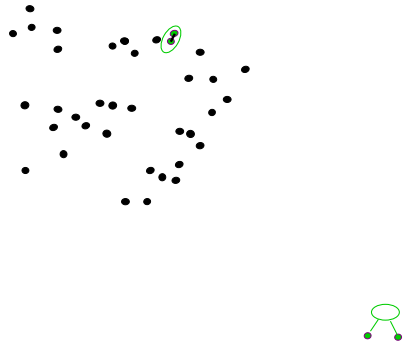
- Find the pair of clusters that are the closest to each other



32

Hierarchical Agglomerative Clustering

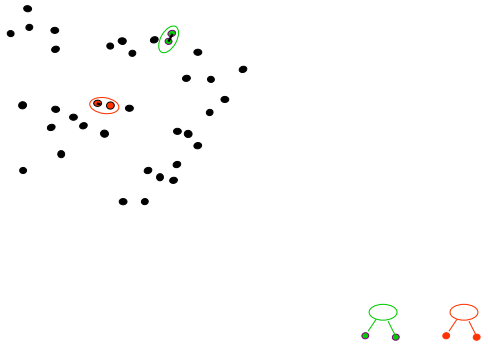
- Merge the two into a single cluster



33

Hierarchical Agglomerative Clustering

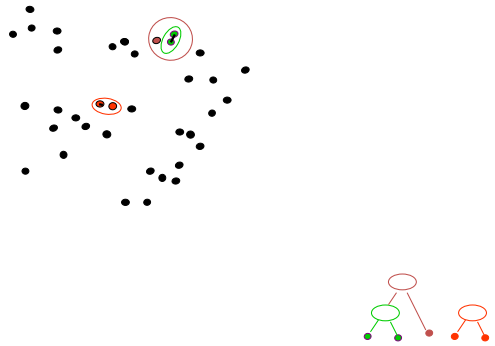
- Repeat …



34

Hierarchical Agglomerative Clustering

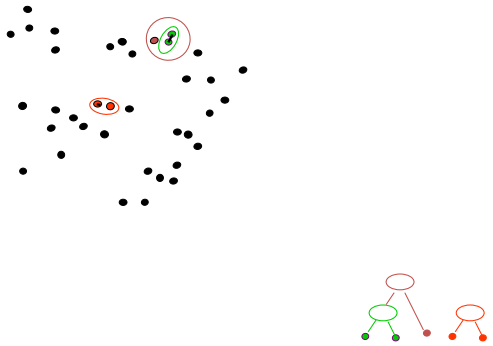
- Repeat …



35

Hierarchical Agglomerative Clustering

- Repeat … until the whole dataset is one giant cluster
- You get a binary tree (not shown here)



36

Hierarchical Agglomerative Clustering Algorithm

Input: a training sample $\{\mathbf{x}_i\}_{i=1}^n$*; a distance function* $d()$.
1. *Initially, place each instance in its own cluster (called a singleton cluster).*
2. ***while*** *(number of clusters* > 1*)* ***do:***
3. *Find the closest cluster pair* A, B*, i.e., they minimize* $d(A, B)$.
4. *Merge* A, B *to form a new cluster.*

Output: *a binary tree showing how clusters are gradually merged from singletons to a root cluster, which contains the whole training sample.*

37

Hierarchical Agglomerative Clustering

How do you measure the closeness between two clusters?

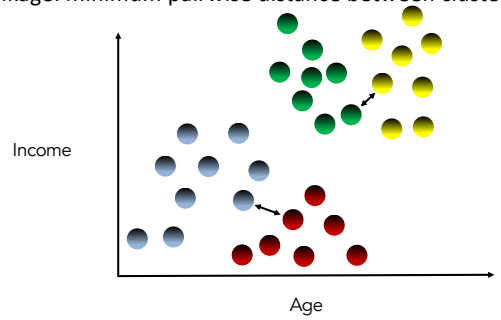
At least three ways:

- **Single-linkage**: the **shortest distance** from any one member of one cluster to any one member of the other cluster
- **Complete-linkage**: the **largest distance** from any one member of one cluster to any one member of the other cluster
- **Average-linkage**: the **average distance** between *all pairs* of members, one from each cluster

38

Hierarchical Linkage Types

Single linkage: minimum pairwise distance between clusters

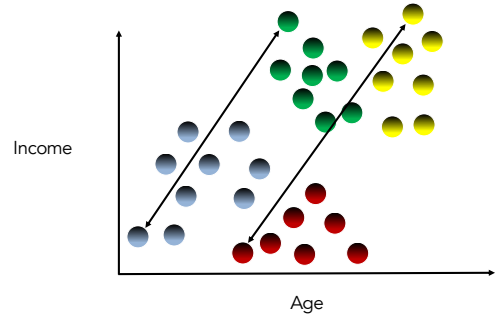


Slide by Intel Software

39

Hierarchical Linkage Types

Complete linkage: maximum pairwise distance between clusters

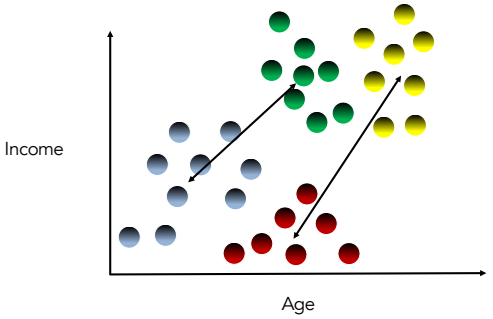


Slide by Intel Software

40

Hierarchical Linkage Types

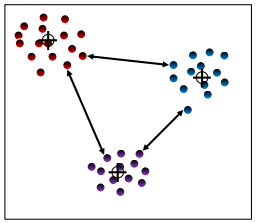
Average linkage: average pairwise distance between clusters



Slide by Intel Software

41

Distance Metric Choice



- Choice of distance metric is extremely important to clustering success
- Each metric has strengths and most appropriate use cases
- but sometimes choice of distance metric is also based on empirical evaluation

Slide by Intel Software

42

Distance

- How to measure the distance between a pair of examples, $\mathbf{X} = (x_1, \ldots, x_n)$ and $\mathbf{Y} = (y_1, \ldots, y_n)$?
  - Euclidean

$$d(\mathbf{X},\mathbf{Y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

  - Manhattan / City-Block

$$d(\mathbf{X},\mathbf{Y}) = \sum_i |x_i - y_i|$$

  - Hamming
    - Number of features that are different between the two examples (useful for categorical data)
  - And many others

43

Hierarchical Agglomerative Clustering

- The binary tree you get is often called a **dendrogram**, or taxonomy, or a hierarchy of data points
- The tree can be cut at any level to produce different numbers of clusters: if you want *k* clusters, just cut the (*k*-1) longest links

44

Hierarchical Agglomerative Clustering Example

- 6 Italian cities
- **Single-linkage**

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0

Example created by Matteo Matteucci

45

Iteration 1: Merge MI and TO

	BA	FI	MI/TO	NA	RM
BA	0	662	877	255	412
FI	662	0	295	468	268
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	268	564	219	0

Recompute **min** distance from MI/TO cluster to all other cities

46

Iteration 2: Merge NA and RM

	BA	FI	MI/TO	NA/RM
BA	0	662	877	255
FI	662	0	295	268
MI/TO	877	295	0	564
NA/RM	255	268	564	0

47

Iteration 3: Merge BA and NA/RM

	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI	268	0	295
MI/TO	564	295	0

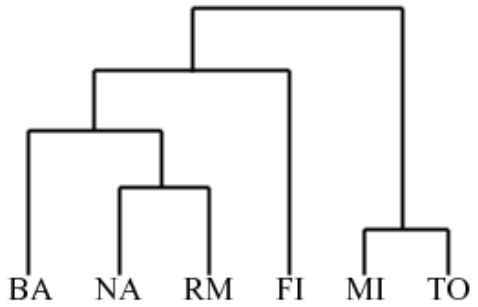
48

Iteration 4: Merge FI and BA/NA/RM

	BA/FI/NA/RM	MI/TO
BA/FI/NA/RM	0	295
MI/TO	295	0

49

Final Dendrogram



50

What Factors Affect the Outcome of **Hierarchical Agglomerative Clustering**?

- Features used
- Range of values for each feature
- Linkage method
- Distance metric used
- Weight of each feature

51

Issues

- When to stop / how many clusters?
- What if there are different ranges for the possible values of each feature?
- How to measure distance for categorical features?
- What if features are not of equal importance?

52

Agglomerative Clustering Stopping Criteria

Method 1	the correct number of clusters is reached
Method 2	minimum average intra-cluster distance is greater than a threshold

Slide by Intel Software

53

Hierarchical Agglomerative Clustering Applet

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletH.html

54

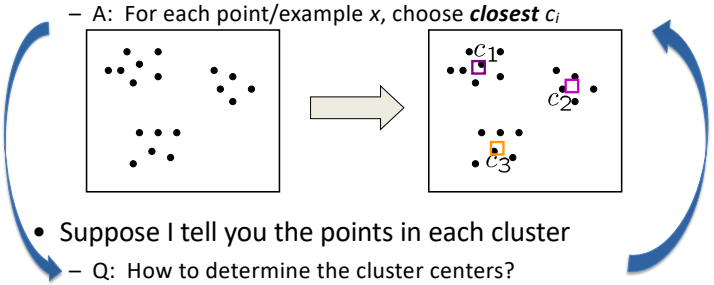
Three Frequently Used Clustering Methods

- **Hierarchical Agglomerative Clustering**
  - Build a binary tree over the dataset
- **K-Means Clustering**
  - Specify the desired number of clusters and use an iterative algorithm to find them
- **Mean Shift Clustering**

56

K-Means Clustering

- Suppose I tell you the cluster centers, c_i
  - Q: How to determine which points to associate with each c_i?
  - A: For each point/example *x*, choose ***closest*** c_i

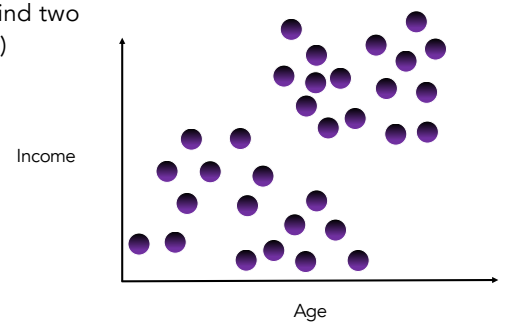


- Suppose I tell you the points in each cluster
  - Q: How to determine the cluster centers?
  - A: Choose c_i to be the ***mean / centroid*** of all points/examples in the cluster

58

K-Means Algorithm

K = 2 (find two clusters)

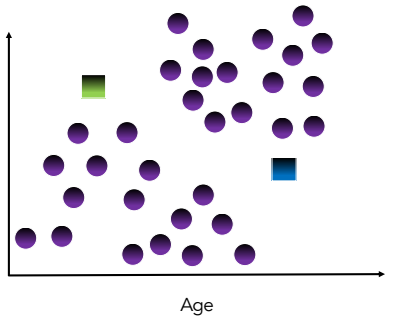


Slide by Intel Software

59

K-Means Algorithm

K = 2, Randomly assign cluster centers

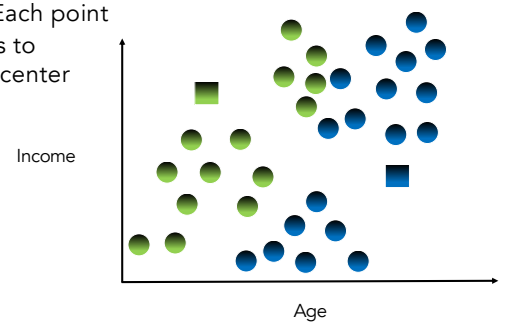


Slide by Intel Software

60

K-Means Algorithm

K = 2, Each point belongs to closest center

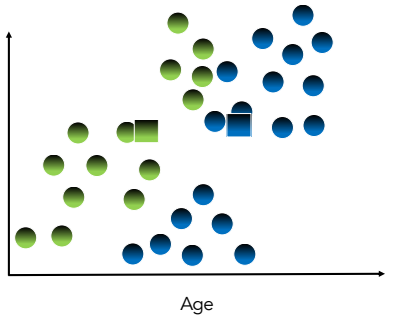


Slide by Intel Software

61

K-Means Algorithm

K = 2, Move each center to cluster's mean

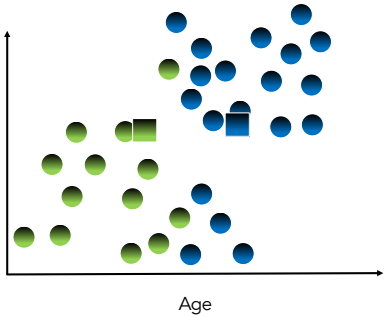


Slide by Intel Software

62

K-Means Algorithm

K = 2, Each point belongs to closest center

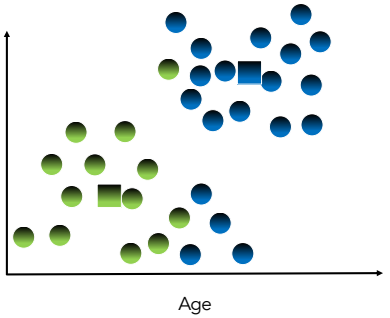


Slide by Intel Software

63

K-Means Algorithm

K = 2, Move each center to cluster's mean

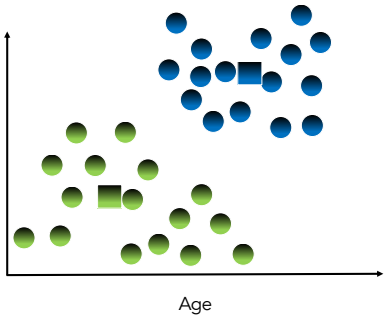


Slide by Intel Software

64

K-Means Algorithm

K = 2, Points don't change → Converged

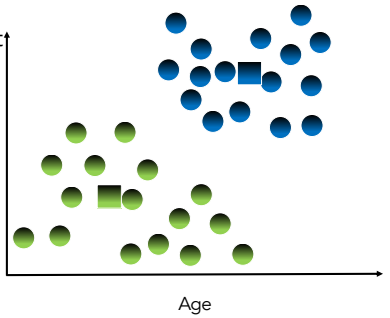


Slide by Intel Software

65

K-Means Algorithm

K = 2, Each point belongs to closest center

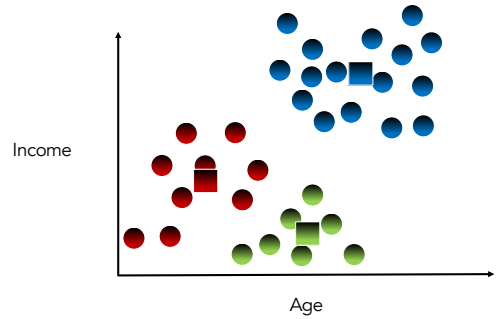


Slide by Intel Software

66

K-Means Algorithm

K = 3

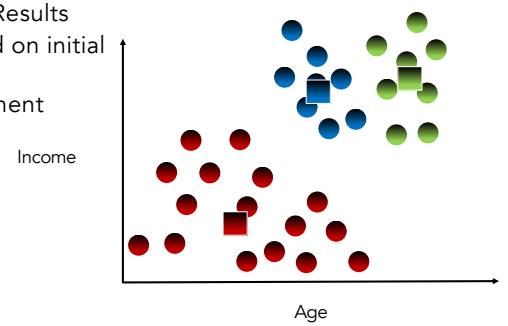


Slide by Intel Software

67

K-Means Algorithm

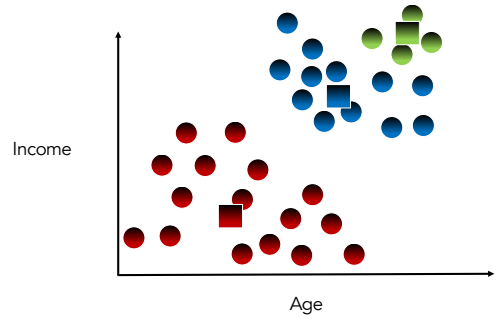
K = 3, Results depend on initial cluster assignment



Slide by Intel Software

68

Which Model is Best?



Slide by Intel Software

69

Which Model is Best?

- **Distortion:** Sum of squared distance from each point (x_i) to its cluster (C_k)

$$\sum_{i=1}^{n} (x_i - C_k)^2$$

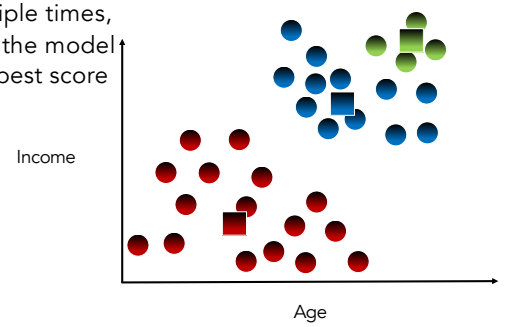
- Smaller value corresponds to tighter clusters
- Other metrics can also be used

Slide by Intel Software

70

Which Model is Best?

Run multiple times, and take the model with the best score

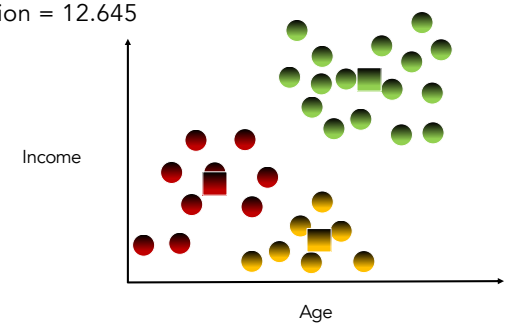


Slide by Intel Software

71

Which Model is Best?

Distortion = 12.645

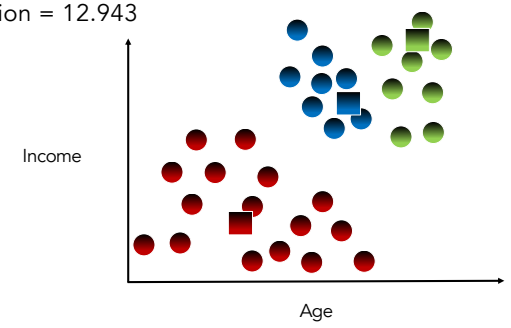


Slide by Intel Software

72

Which Model is Best?

Distortion = 12.943

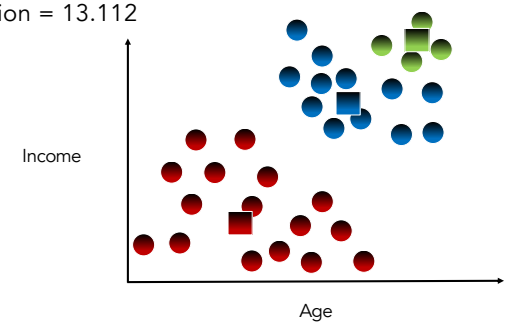


Slide by Intel Software

73

Which Model is Best?

Distortion = 13.112



Slide by Intel Software

74

K-Means Algorithm

- Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n, k$ where each $\mathbf{x}_i$ is a point/example in a *d*-dimensional feature space
- **Step 1**: Select *k* cluster centers, $\mathbf{c}_1, \ldots, \mathbf{c}_k$
- **Step 2**: For each point $\mathbf{x}_i$, determine its cluster: Find the closest center (using, say, Euclidean distance)
- **Step 3**: Update all cluster centers as the centroids

$$\mathbf{c}_i = \frac{1}{num\_pts\_in\_cluster\_i} \sum_{\mathbf{x} \in \text{cluster } i} \mathbf{x}$$

- Repeat steps 2 and 3 until cluster centers no longer change

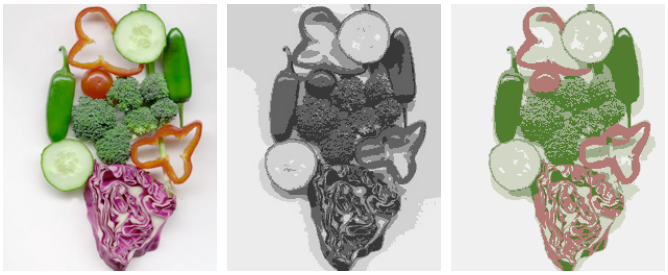
75

K-Means Demo

- http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

81

Example: Image Segmentation



Input image | Clusters on intensity | Clusters on color

83

K-Means Properties

- Will it always terminate?
  - Yes (finite number of ways of partitioning a finite number of points into *k* groups)
- Is it guaranteed to find an "optimal" clustering?
  - No, but each iteration will reduce the distortion of the clustering

84

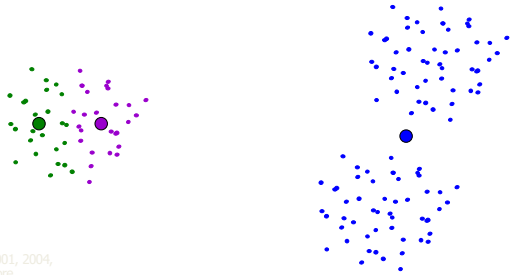
Non-Optimal Clustering

Say *k*=3 and you are given the following points:

85

Non-Optimal Clustering

Given a poor choice of the initial cluster centers, the following result is possible:



86

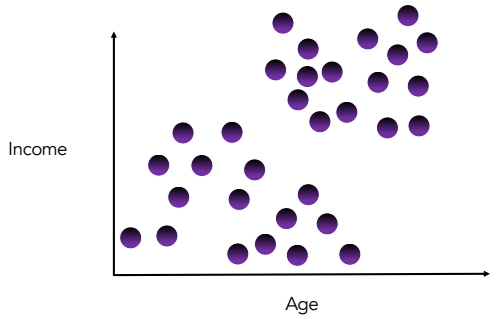
Picking Starting Cluster Centers

Which local optimum *k*-Means goes to is determined solely by the starting cluster centers

- **Idea 1**: Run *k*-Means multiple times with different starting, random cluster centers (hill climbing with random restarts)
- **Idea 2**: Pick a random point x_1 from the dataset
  1. Find a point x_2 far from x_1 in the dataset
  2. Find x_3 far from both x_1 and x_2
  3. … Pick *k* points like this, and use them as the starting cluster centers for the *k* clusters

87

Smarter Initialization of K-Means Clusters

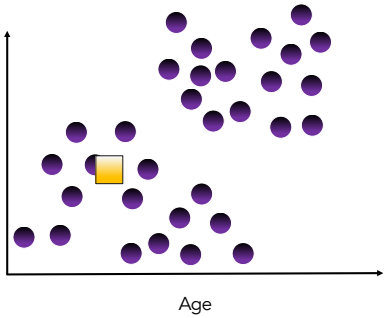


Slide by Intel Software

88

Smarter Initialization of K-Means Clusters

Pick one point at random as initial point

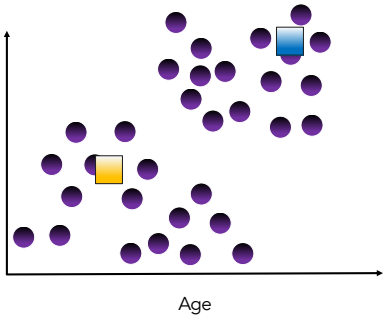


Slide by Intel Software

89

Smarter Initialization of K-Means Clusters

Pick next point by weighting each by $1/\text{distance}^2$

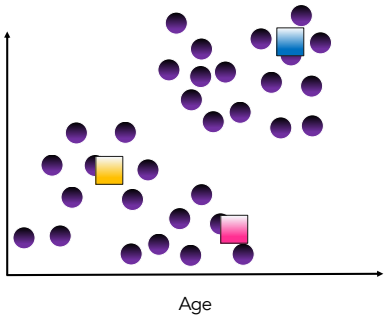


Slide by Intel Software

90

Smarter Initialization of K-Means Clusters

Pick next point by weighting each by $\sum 1/\text{distance}^2$

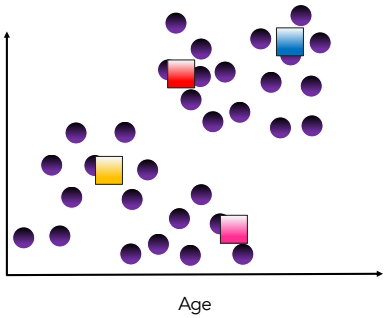


Slide by Intel Software

91

Smarter Initialization of K-Means Clusters

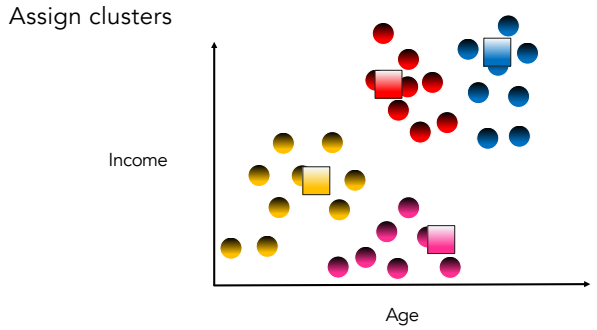
Pick next point by weighting each by $\sum 1/\text{distance}^2$



Slide by Intel Software

92

Smarter Initialization of K-Means Clusters



Slide by Intel Software

93

Picking the Number of Clusters

- Difficult problem
- Heuristic approaches depend on the number of points and the number of dimensions

94

Picking the Number of Clusters

- Sometimes the problem has a known *k*
- Clustering similar jobs on 4 CPU cores (*k* = 4)
- A clothing design in 10 different sizes to cover most people (*k* = 10)
- A navigation interface for browsing scientific papers with 20 disciplines (*k* = 20)

Slide by Intel Software

95

Measuring Cluster Quality

- **Distortion** = Sum of squared distances of each data point to its cluster center:

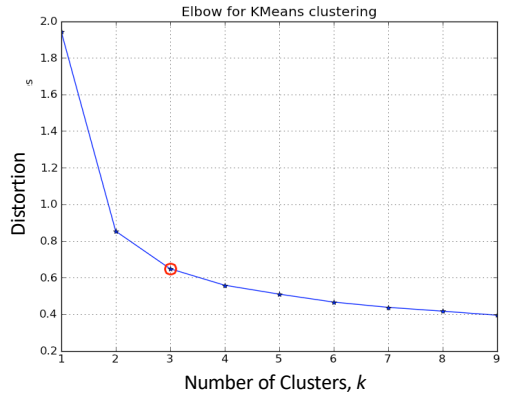
$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

- The "optimal" clustering is the one that minimizes distortion (over all possible cluster center locations and assignment of points to clusters)

96

How to Pick the Number of Clusters, *k*?

Try multiple values of *k* and pick the one at the "elbow" of the distortion curve



97

Uses of *K*-Means

- Often used as an exploratory data analysis tool
- In one-dimension, a good way to quantize real-valued variables into *k* non-uniform buckets
- Used on acoustic data in speech recognition to convert waveforms into one of *k* categories (known as **Vector Quantization**)
- Also used for choosing color palettes on graphical display devices

99

Three Frequently Used Clustering Methods

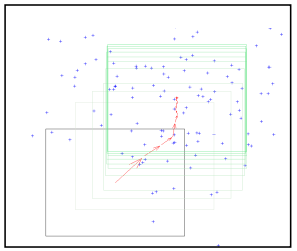
- **Hierarchical Agglomerative Clustering**
  - Build a binary tree over the dataset
- **K-Means Clustering**
  - Specify the desired number of clusters and use an iterative algorithm to find them
- **Mean Shift Clustering**

100

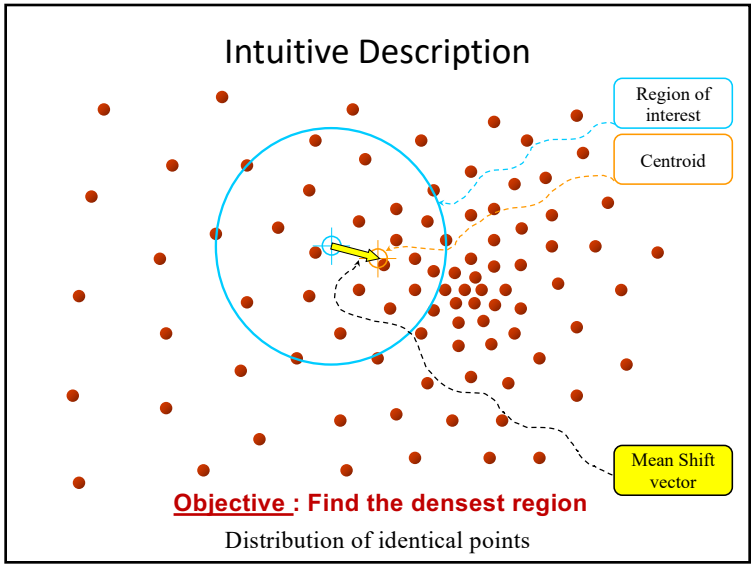
Mean Shift Clustering

1. Choose a search window size
2. Choose the initial location of the search window
3. Compute the mean location (centroid of the data) in the search window
4. Center the search window at the mean location computed in Step 3
5. Repeat Steps 3 and 4 until convergence

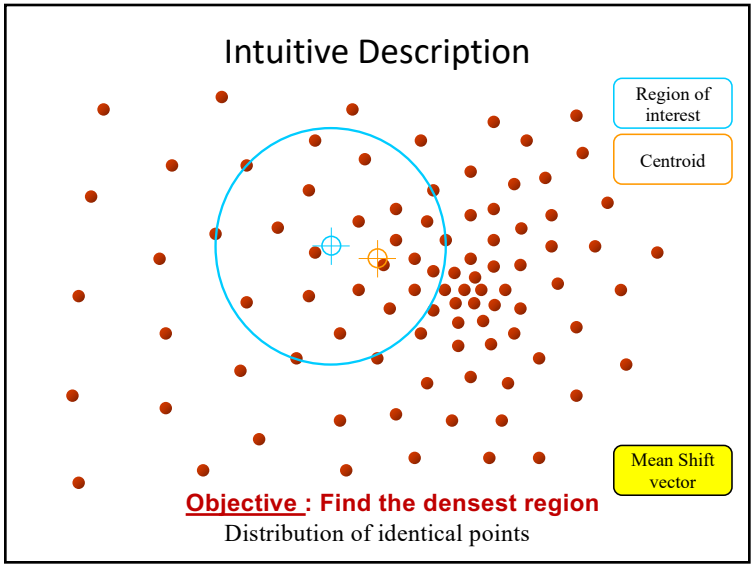
The mean shift algorithm seeks the ***mode***, i.e., point of highest density of a data distribution:



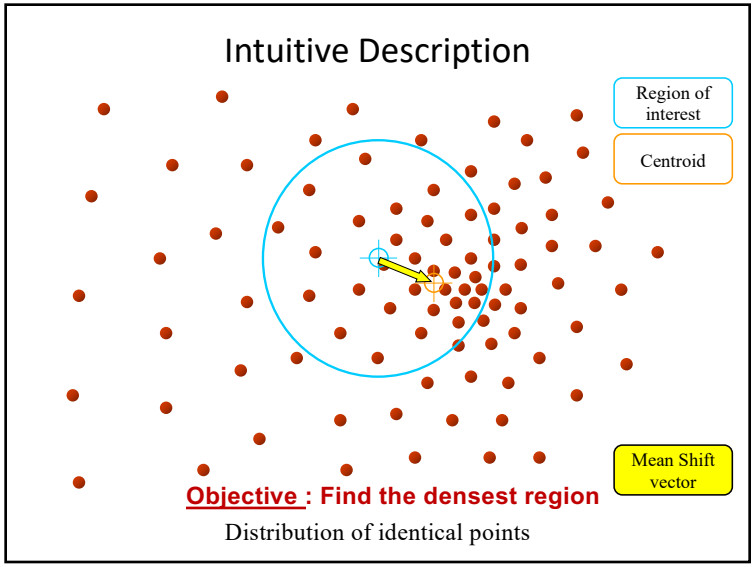
101



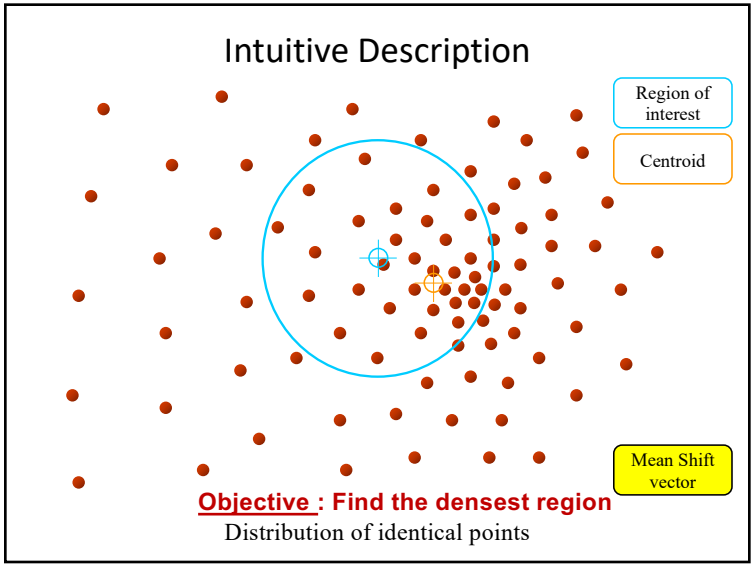
102



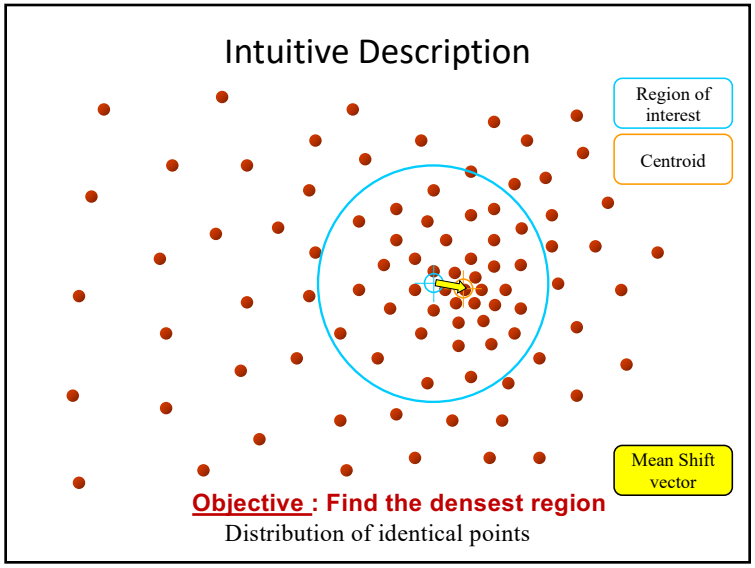
103



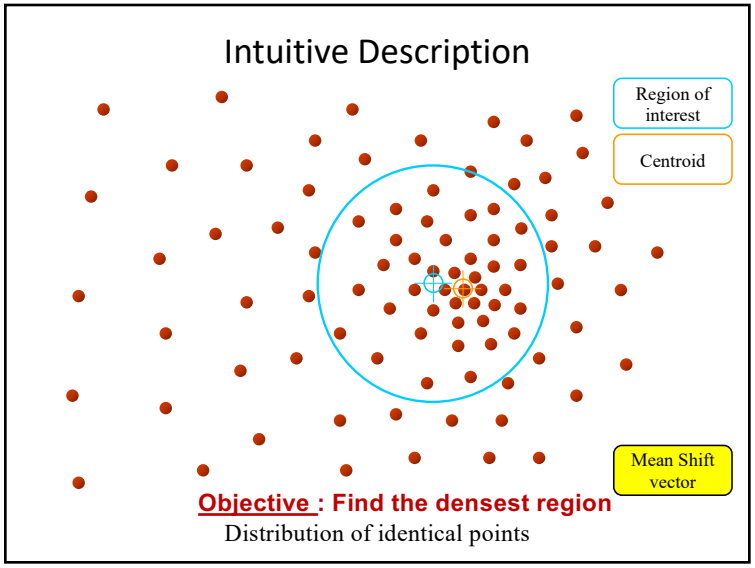
104



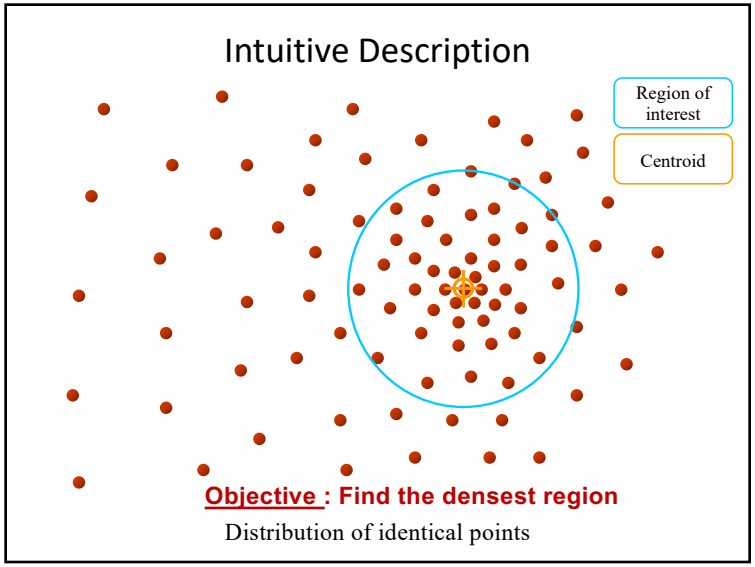
105



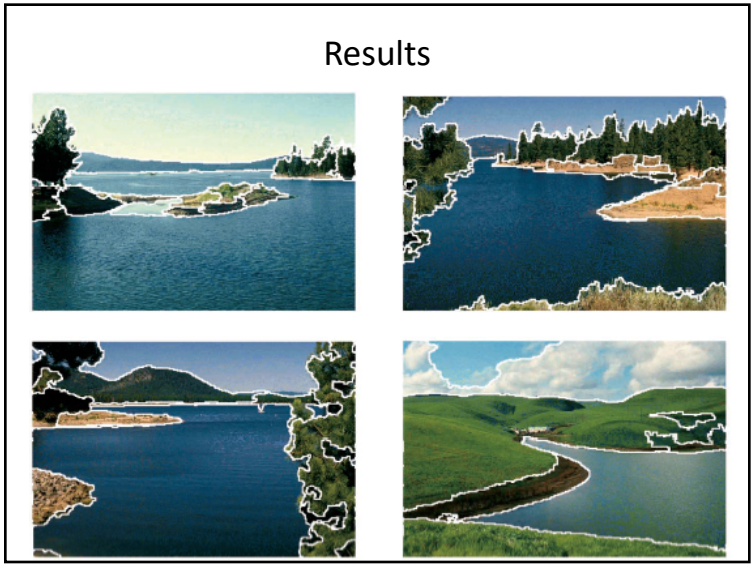
106



107

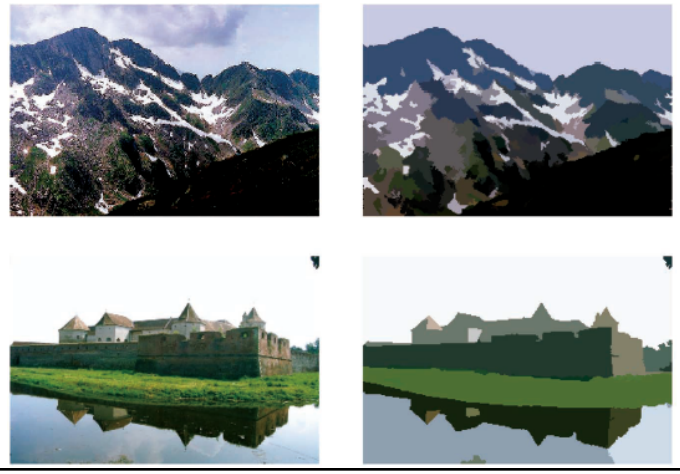


108



111

Results



112

Supervised Learning

- A labeled training sample is a collection of examples (aka instances): $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
- Assume $(\boldsymbol{x}_i, y_i) \overset{\text{i.i.d.}}{\sim} P(\boldsymbol{x}, y)$ and $P(\boldsymbol{x}, y)$ is *unknown*
- Supervised learning learns a function $h: \boldsymbol{x} \rightarrow y$ in some function family, H, such that $h(\boldsymbol{x})$ predicts the true label y on *future data*, $\boldsymbol{x}$, where
  $(\boldsymbol{x}, y) \overset{\text{i.i.d.}}{\sim} P(\boldsymbol{x}, y)$
  - Classification: if y discrete
  - Regression: if y continuous

114

Labels

- Examples
  - Predict gender (M, F) from weight, height
  - Predict adult, juvenile (A, J) from weight, height
- A **label** y is the desired prediction for an instance $\boldsymbol{x}$
- Discrete labels: **classes**
  - M, F; A, J: often encode as 0, 1 or -1, 1 or +, -
  - Multiple classes: 1, 2, 3, …, C. No class order implied.
- Continuous label: e.g., blood pressure

115

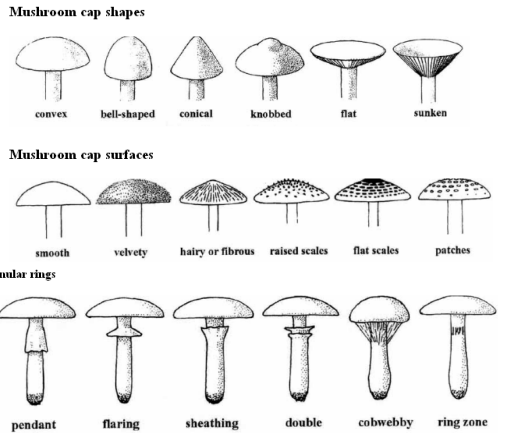
Concept Learning

- Determine if a given example is or is not an instance of the concept/class/category
  - If it is, call it a **positive** example
  - If not, called it a **negative** example

116

Example: Mushroom Classification

Edible or Poisonous?



http://www.usask.ca/biology/fungi/

117

Mushroom Features/Attributes

1. **cap-shape**: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. **cap-surface**: fibrous=f, grooves=g, scaly=y, smooth=s
3. **cap-color**: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. **bruises?**: bruises=t, no=f
5. **odor**: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. **gill-attachment**: attached=a, descending=d, free=f, notched=n
7. …

Classes: edible=e, poisonous=p

118

- Start here

119

Supervised Concept Learning by Induction

- Given a **training set** of positive and negative examples of a concept:
  - $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
    where each y_i is either **+** or **–**
- Construct a description that accurately classifies whether **future examples** are positive or negative:
  - $h(x_{n+1}) = y_{n+1}$
    where y_{n+1} is the **+** or **–** prediction

120

Supervised Learning Methods

- ***k*-nearest-neighbors (*k*-NN)** (Chapter 18.8.1)
- Decision trees
- Neural networks (NN)
- Support vector machines (SVM)
- etc.

121

Inductive Learning by Nearest-Neighbor Classification

A simple approach:

- save each training example as a point in Feature Space
- classify a new example by giving it the same classification as its ***nearest neighbor*** in Feature Space

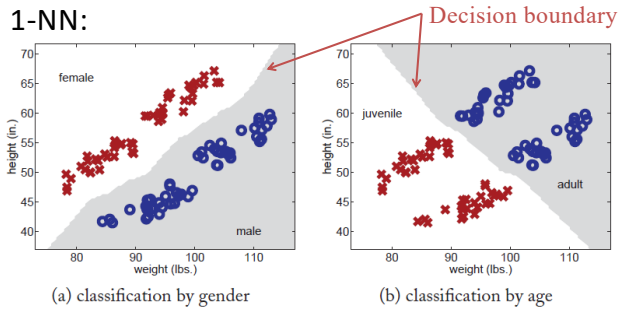
122

k-Nearest-Neighbors (k-NN)

Input: Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$*; distance function* $d()$*; number of neighbors* k*; test instance* $\mathbf{x}^*$

1. *Find the* k *training instances* $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ *closest to* $\mathbf{x}^*$ *under distance* $d()$.
2. *Output* y^* *as the majority class of* $y_{i_1}, \ldots, y_{i_k}$*. Break ties randomly.*

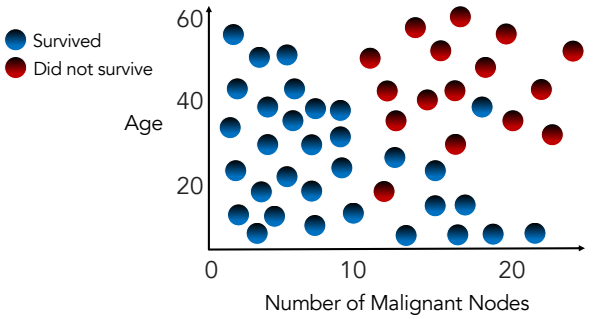
- 1-NN:



(a) classification by gender (b) classification by age

123

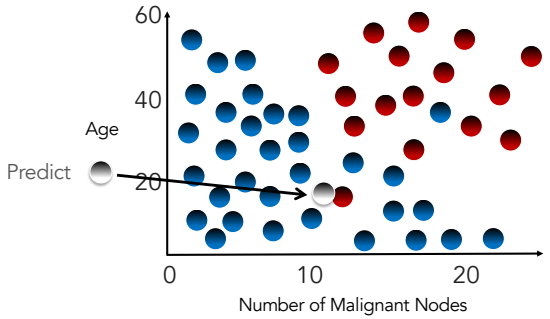
k-Nearest Neighbors Classification



Slide by Intel Software

124

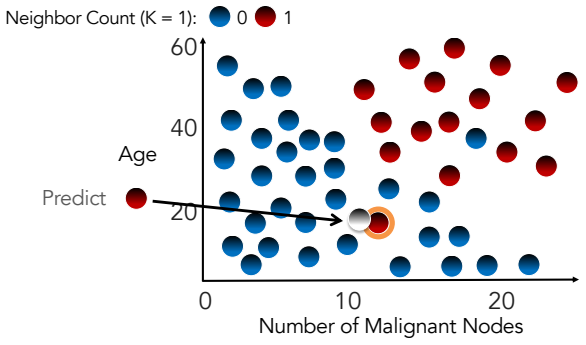
k-Nearest Neighbors Classification



Slide by Intel Software

125

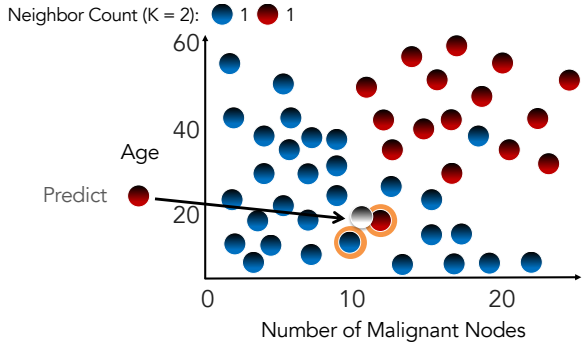
k-Nearest Neighbors Classification



Slide by Intel Software

126

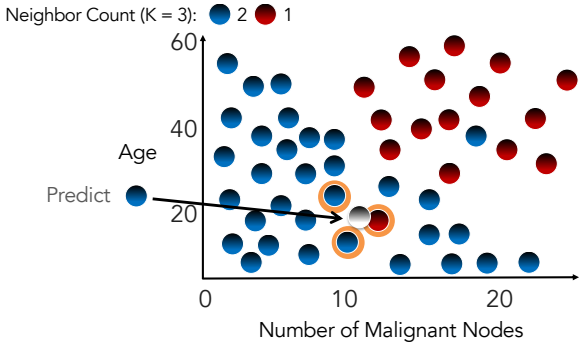
k-Nearest Neighbors Classification



Slide by Intel Software

127

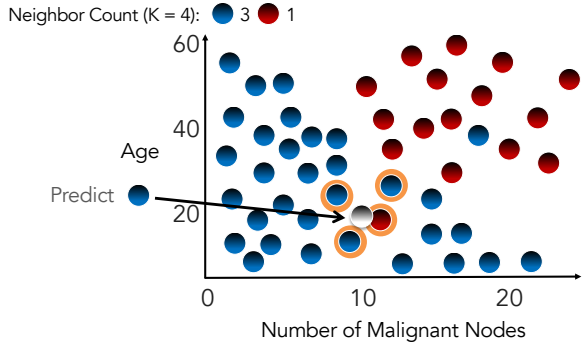
k-Nearest Neighbors Classification



Slide by Intel Software

128

K Nearest Neighbors Classification



Slide by Intel Software

129

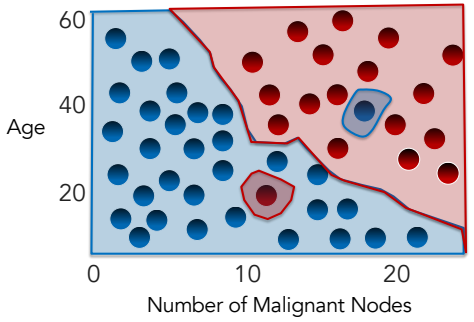
k-NN

- What if we want regression?
  - Instead of majority vote, take ***average*** of neighbors' *y* values
- How to pick *k*?
  - Split data into training and tuning sets
  - Classify tuning set with different values of *k*
  - Pick the *k* that produces the smallest tuning-set error

130

k-Nearest Neighbors Decision Boundary

k = 1

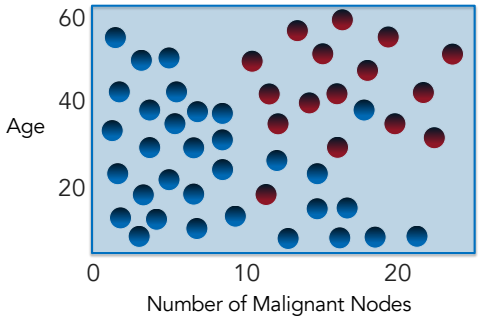


Slide by Intel Software

131

k-Nearest Neighbors Decision Boundary

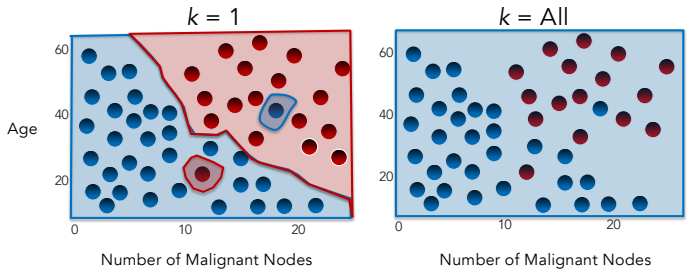
k = All



Slide by Intel Software

132

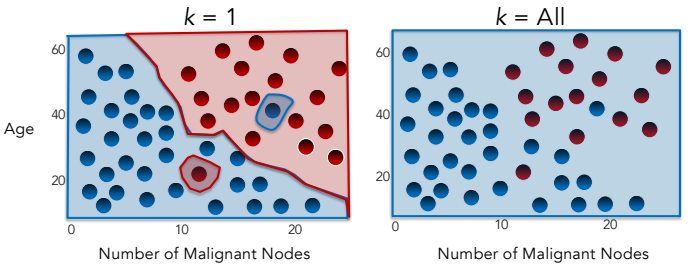
Value of *k* Affects Decision Boundary



Slide by Intel Software

133

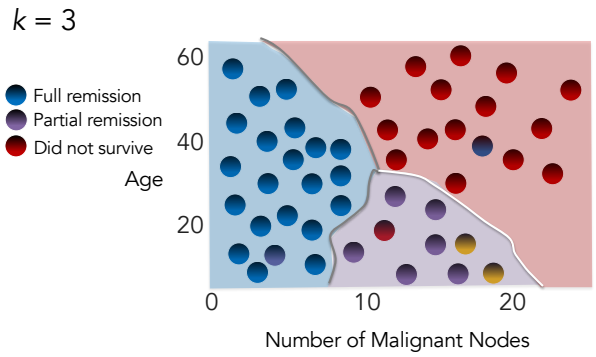
Value of *k* Affects Decision Boundary



Slide by Intel Software

134

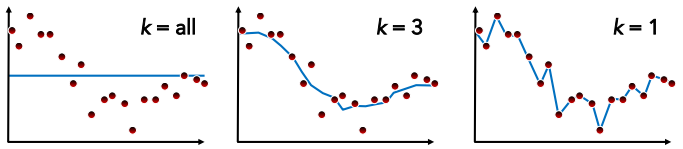
Multiclass *k*-NN Decision Boundary



Slide by Intel Software

135

Regression with *k*-NN



Slide by Intel Software

136

Characteristics of a *k*-NN Model

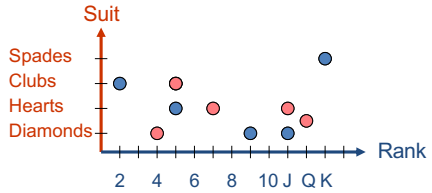
- Fast to create model because it simply stores the data (the training data is the model)
- Slow to classify a test example because many distance calculations are required
- Requires lots of memory if dataset is large

Slide by Intel Software

137

Characteristics of a *k*-NN Model

- Doesn't generalize well if the examples in each class are not well "clustered"



138

k-NN Demo

http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html

139

Inductive Bias

- Inductive learning is an inherently conjectural process. Why?
  - any knowledge created by generalization from specific facts cannot be proven true
  - it can only be proven false
- Hence, inductive inference is "***falsity preserving***," not "truth preserving"

142

Inductive Bias

- Learning can be viewed as searching the Hypothesis Space H of possible h functions
- Inductive Bias
  - is used when one h is chosen over another
  - is needed to generalize beyond the specific training examples
- Completely unbiased inductive algorithm
  - only memorizes training examples
  - can't predict anything about unseen examples

143

Inductive Bias

Biases commonly used in machine learning:

- Restricted Hypothesis Space Bias:
  allow only certain types of h's, not arbitrary ones
- Preference Bias:
  define a metric for comparing h's so as to determine whether one is better than another

144