IS-ZC444: Artificial Intelligence

Lecture-13: Machine Learning

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems,
BITS Pilani, Pilani, Jhunjhunu-333031, Rajasthan, INDIA

Nov 15, 2020 FLIPPED (WILP @ BITS-Pilani Jul-Nov 2020)

Classification

Finding the right label

Classification

Finding the right label

Classification

Finding the right label

What feature (attributes) would you choose?

Color, texture, weight, density, hardness .....

K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

K Nearest Neighbor (KNN)

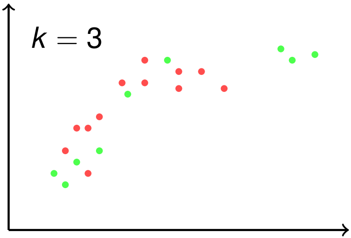
You are most likely as your friends (Bias)

- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting

K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

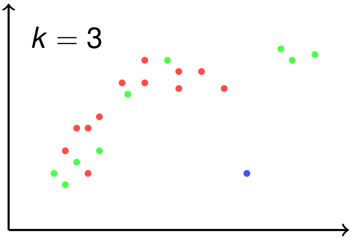
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

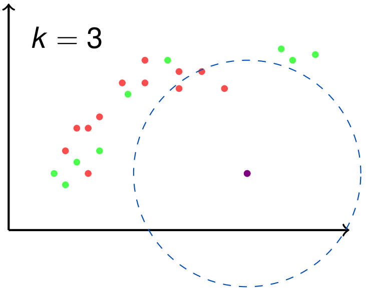
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

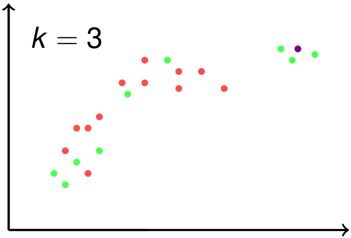
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

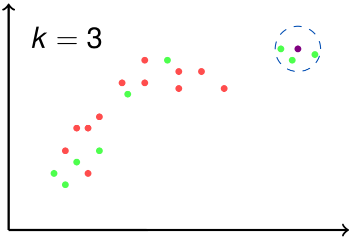
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

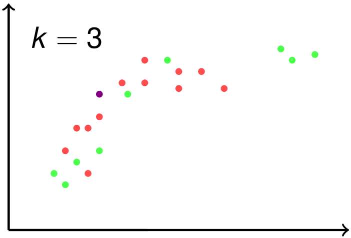
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

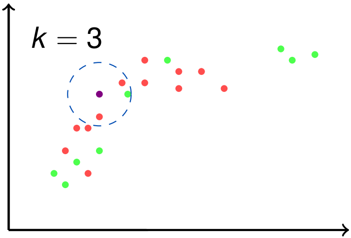
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

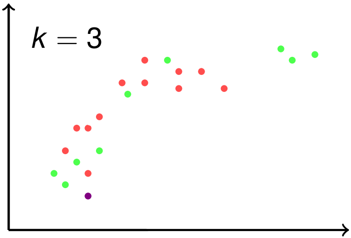
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

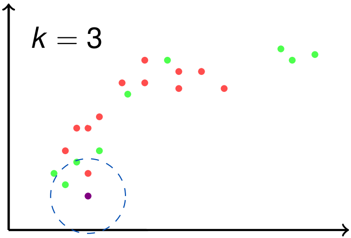
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



Decision Tree

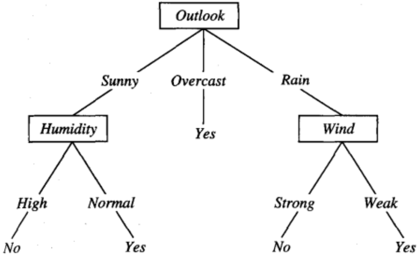
Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

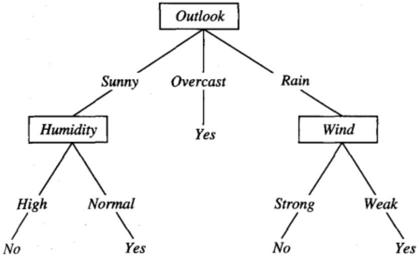


- Each node in the tree specifies a test for some attribute

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

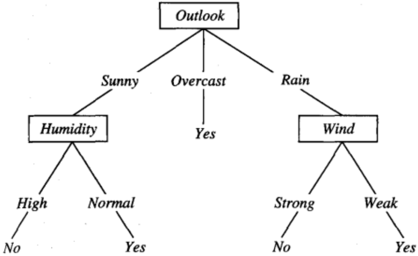


- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

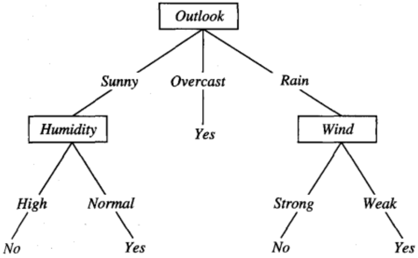


- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value
- Decision trees represent a disjunction of conjunctions

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.



- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value
- Decision trees represent a disjunction of conjunctions

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee \quad (Outlook = Overcast) \vee (Outlook = Rain \wedge Wind = Weak)$$

Entropy

Characterizes the impurity of an arbitrary collection of examples

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

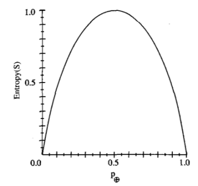
$\boxed{\mathbf{0}}$ – when all members are of same class.

$\boxed{\mathbf{1}}$ – if equal number of positive and negative

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

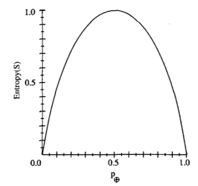
$\boxed{\mathbf{0}}$ – when all members are of same class.

$\boxed{\mathbf{1}}$ – if equal number of positive and negative

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

$\boxed{\mathbf{0}}$ – when all members are of same class.

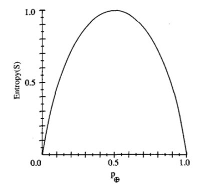
$\boxed{\mathbf{1}}$ – if equal number of positive and negative

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

$\boxed{\mathbf{0}}$ – when all members are of same class.

$\boxed{\mathbf{1}}$ – if equal number of positive and negative

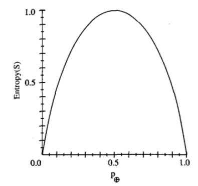
Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

$$Entropy([9+, 5-])$$

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

$\boxed{\mathbf{0}}$ – when all members are of same class.

$\boxed{\mathbf{1}}$ – if equal number of positive and negative

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

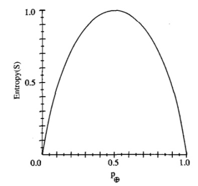
$$Entropy([9+, 5-])$$

$$= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$

Entropy

Characterizes the impurity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

$\boxed{\mathbf{0}}$ – when all members are of same class.

$\boxed{\mathbf{1}}$ – if equal number of positive and negative

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

$$\begin{aligned} Entropy([9+, 5-]) & \\ &= -(9/14)\log_2(9/14) \\ &\quad -(5/14)\log_2(5/14) \\ &= 0.94 \end{aligned}$$

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

[1] Outlook, Temperature, Humidity, Wind

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

here S_v contains that data items of S where the value of attribute A is v

[1] Outlook, Temperature, Humidity, Wind

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

here S_v contains that data items of S where the value of attribute A is v

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

[1] Outlook, Temperature, Humidity, Wind

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

here S_v contains that data items of S where the value of attribute A is v

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

For example

$$\begin{aligned}
S_{Sunny} &= \{D1, D2, D8, D9, D11\} \\
S_{Overcast} &= \{D3, D7, D12, D13\} \\
S_{Cool} &= \{D5, D6, D7, D9\} \\
S_{Hot} &= \{D1, D2, D3, D13\} \\
S_{Normal} &= \{D5, D6, D7, D9, D10, D11, D13\} \\
S_{High} &= \{D1, D2, D3, D4, D8, D12, D14\}
\end{aligned}$$

And so on....

[1] Outlook, Temperature, Humidity, Wind

Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$$
$$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$$

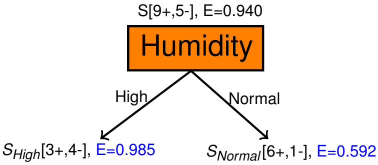
$$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$$
$$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$$

Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$$
$$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$$

$$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$$
$$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$$

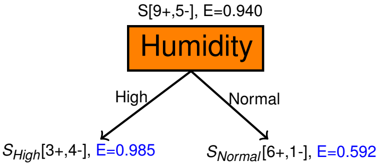


Information Gain

$$Gain(S,A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\} \qquad S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$$

$$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\} \qquad S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$$



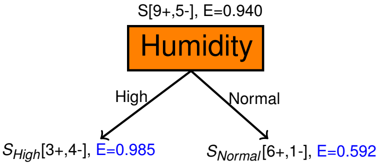
$$\text{Gain(S,Humidity)} = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$$

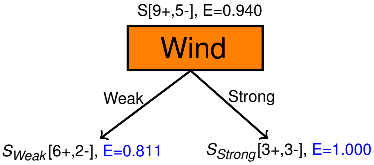
Information Gain

$$Gain(S,A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$$
$$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$$

$$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$$
$$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$$





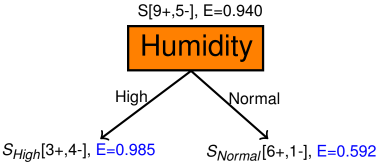
$$\text{Gain(S,Humidity)} = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$$

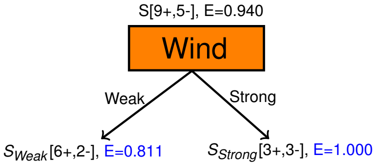
Information Gain

$$Gain(S,A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$$
$$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$$

$$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$$
$$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$$





Gain(S,Humidity)$= 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$

Gain(S,Wind) $= 0.940 - (8/14)0.811 - (6/14)1.000 = 0.048$

Information Gain and Decision Tree

$$\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}$$

Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$

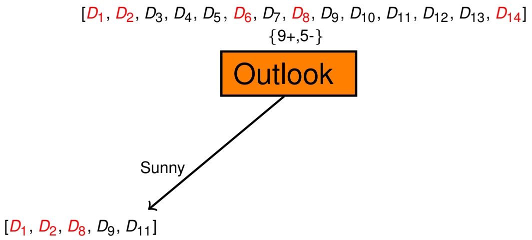
$[D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, D_{11}, D_{12}, D_{13}, D_{14}]$

{9+,5-}

$\boxed{\text{Outlook}}$

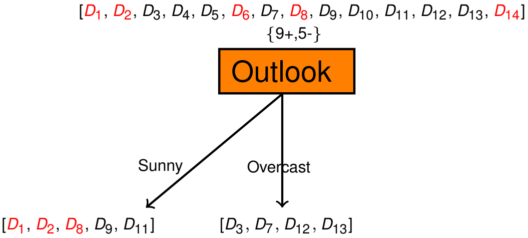
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



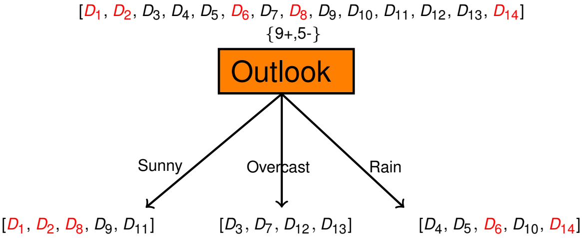
Information Gain and Decision Tree

$$\begin{aligned} Gain(S, Humidity) &= 0.151 \\ Gain(S, Wind) &= 0.048 \\ Gain(S, Outlook) &= 0.246 \\ Gain(S, Temperature) &= 0.029 \end{aligned}$$



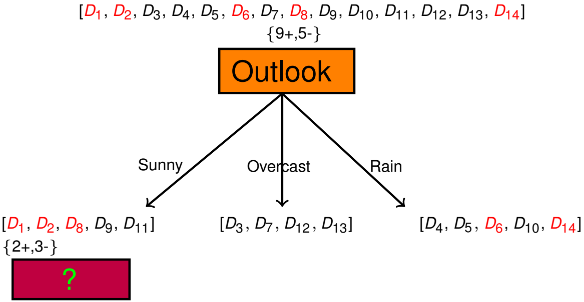
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



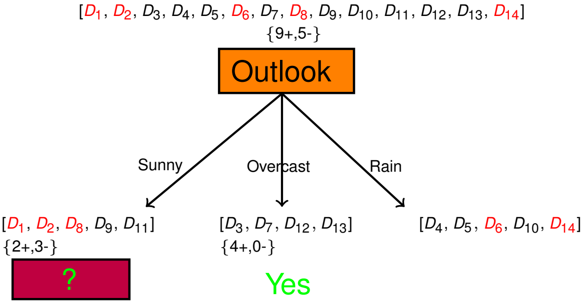
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



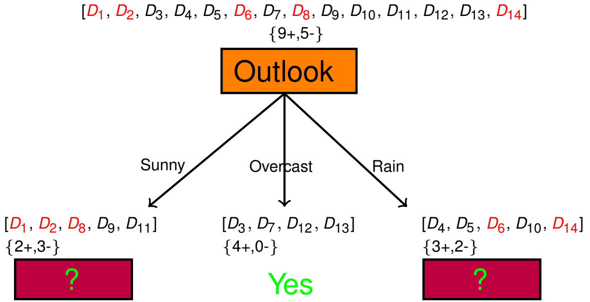
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$

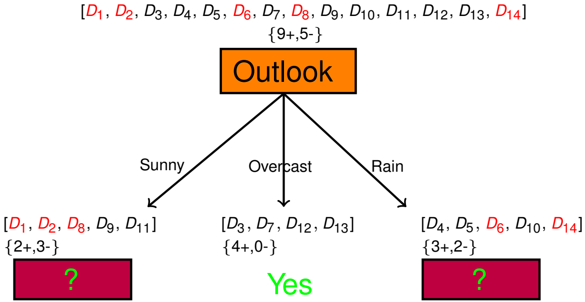


Information Gain and Decision Tree

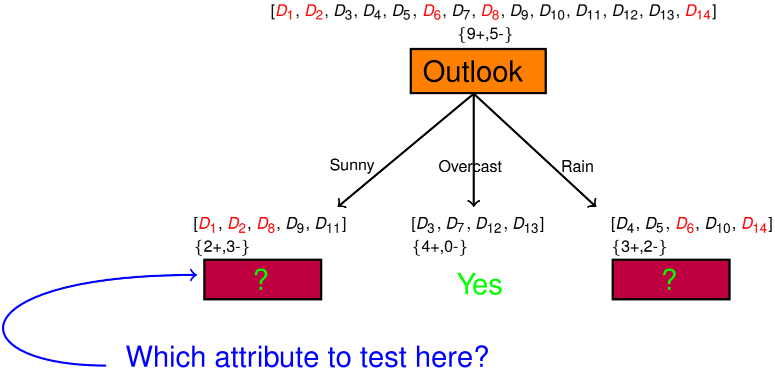
$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



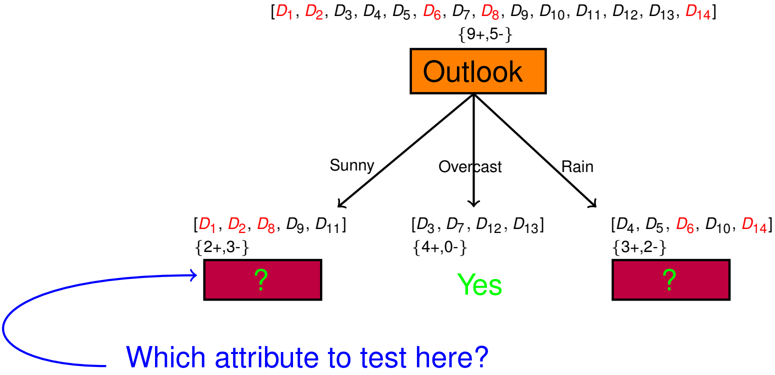
Recursively apply the same



Recursively apply the same

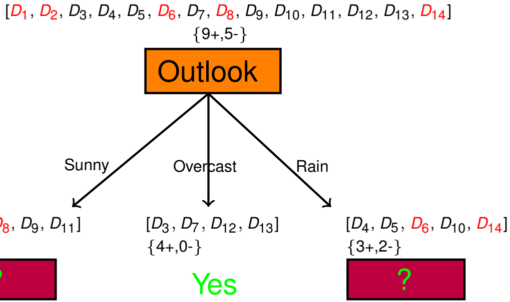


Recursively apply the same



S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Recursively apply the same

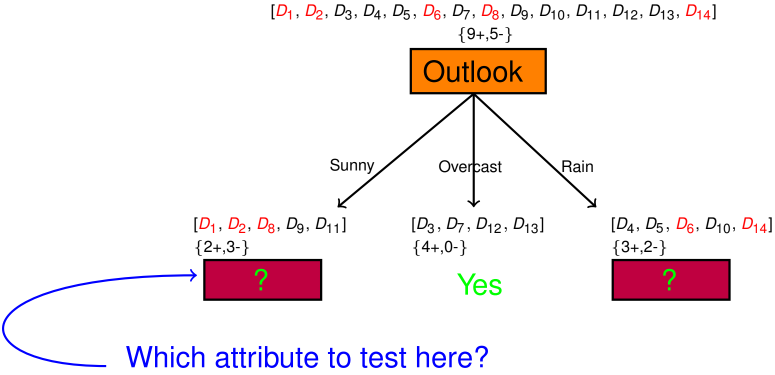


Which attribute to test here?

S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

Recursively apply the same

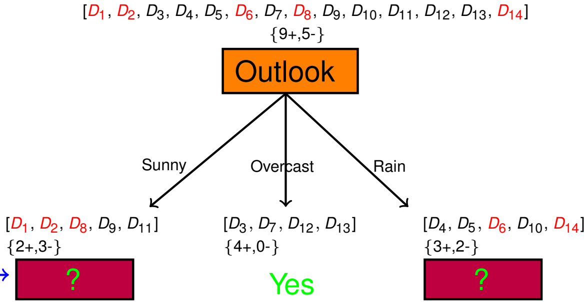


S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

Gain(S_{sunny},Temperature)$= 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.57$

Recursively apply the same



Which attribute to test here?

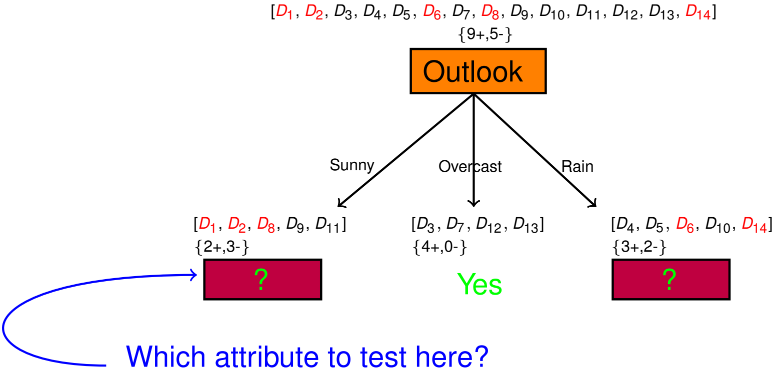
S_{sunny}=$[D_1, D_2, D_8, D_9, D_{11}]$

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

Gain(S_{sunny},Temperature)$= 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.57$

Gain(S_{sunny},Wind)$= 0.970 - (2/5)1.0 - (3/5)1.0 = 0.019$

Recursively apply the same



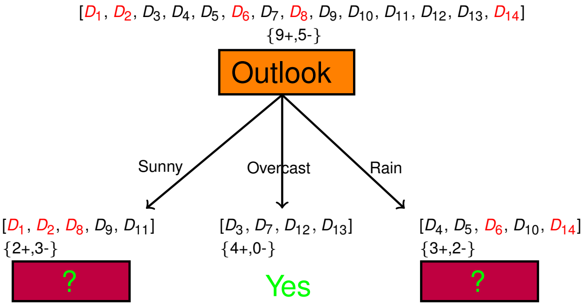
S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

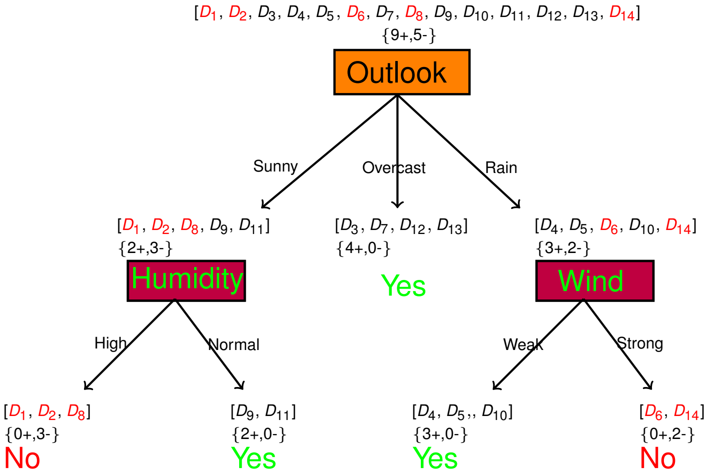
Gain(S_{sunny},Temperature)$= 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.57$

Gain(S_{sunny},Wind)$= 0.970 - (2/5)1.0 - (3/5)1.0 = 0.019$

Recursively apply the same



Recursively apply the same



Decision Tree

A method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

Decision Tree

A method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

What is classification for

$$(Outlook = Rain, Humidity = High, Wind = Weak)$$

Decision Tree

A method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions

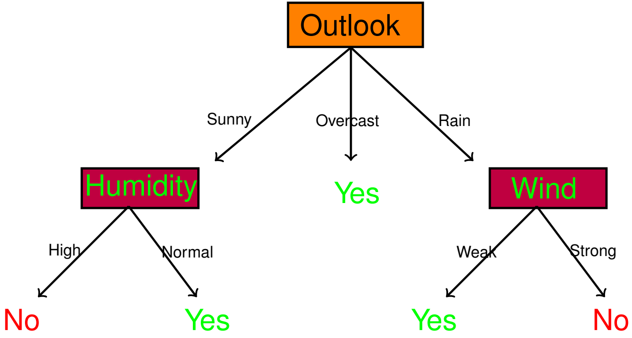
Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

What is classification for

$(Outlook = Rain, Humidity = High, Wind = Weak)$

ALERT: (missing value) what is Temperature?

Example



Classification for ($Outlook = Rain, Humidity = High, Wind = Weak$) is

$\boxed{\text{YES}}$

Iterative-Dichotomiser-3 (ID3) Algorithm By: John Ross Quinlan

Algorithm 1: ID3(Examples,Target_attribute,Attributes)

Examples are the training data, *Target_attribute* is the attribute whose value is to be predicted by the tree. *Attributes* is a list of other attributes that may be tested by the learned decision tree. Algorithm returns a decision tree that correctly classify the given example.

Create a single-node tree *Root*
IF *Examples* are all +ve **THEN return** *Root* with label +ve
IF *Examples* are all -ve **THEN return** *Root* with label -ve
IF *Attributes* $= \phi$ **THEN return** *Root* with most common *Target_attribute*
A $\leftarrow$ attribute from *Attributes* that best classifies *Examples*
Decision attribute for *Root* $\leftarrow$ A
foreach *value* v_i *of A* **do**
    Add a new tree branch below Root, to test A=v_i
    Examples$_{v_i}$ $\leftarrow$ subset of *Examples* having value v_i for A
    IF Examples$_{v_i}$ $= \phi$ **THEN** below this branch add a leaf with label = most common value of *Target_attribute* in *Examples*
    ELSE below this branch add subtree ID3(Examples$_{v_i}$,*Target_attribute*,*Attributes*-$\{A\}$)
return *Root*

Issues Decision Tree

Given a collection of training examples, there could be many decision trees consistent with the examples

- ID3 search strategy
  - selects in favor of shorter trees over longer ones, and
  - selects trees that place the attributes with highest information gain closest to the root

Issues Decision Tree

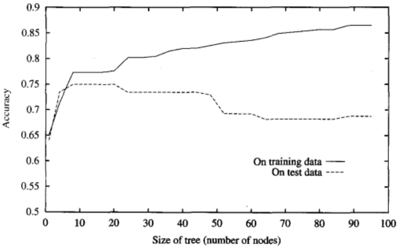
Given a collection of training examples, there could be many decision trees consistent with the examples

- ID3 search strategy
  - selects in favor of shorter trees over longer ones, and
  - selects trees that place the attributes with highest information gain closest to the root
- Issues in decision trees include
  1. how deeply to grow
  2. handling continuous attributes
  3. choosing an appropriate attribute selection measure
  4. missing attribute values
  5. attributes with differing costs, and
  6. improving computational efficiency

Issues in Decision Tree

Overfitting

Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.



- This can occur when training examples contain random errors or noise.

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree
- Use all the available data for training, but apply a statistical test (such as chi-square test) to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree
- Use all the available data for training, but apply a statistical test (such as chi-square test) to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.
- Use an explicit measure of the complexity for encoding the training examples (such as Minimum Description Length) and the decision tree, halting growth of the tree when this encoding size is minimized.

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)

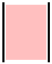
[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



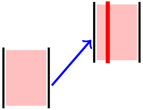
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



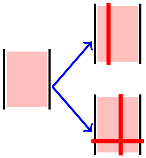
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



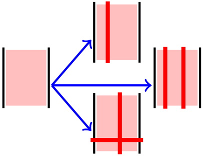
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



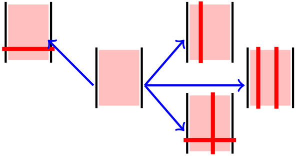
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



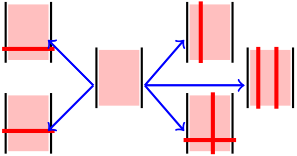
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



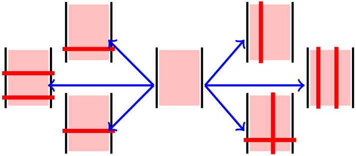
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



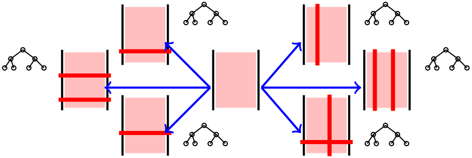
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



- For every subset, built a decision tree

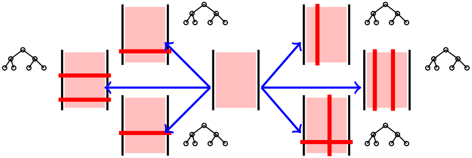
[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



- For every subset, built a decision tree
- To classify new element: **apply voting**

[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Regression

Regression predicts value of continuous a target variable

x_1	x_2	x_3	y
10	50	20	10
11	31	22	12
11	12	15	4
20	55	20	22
23	41	27	1
31	12	35	9
13	18	12	23
21	55	16	16
32	56	27	22
8	22	35	??

What should come at the place of ??

Regression

Regression predicts value of continuous a target variable

Regression

Regression predicts value of continuous a target variable

- A simplest model for regression can be a linear combination of the input variables

$$y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$$

where x is a n dimensional vector $(x_1, x_2, ..., x_n)$ representing some feature

Regression

Regression predicts value of continuous a target variable

- A simplest model for regression can be a linear combination of the input variables

$$y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$$

  where x is a n dimensional vector $(x_1, x_2, ..., x_n)$ representing some feature

- It can be extended by considering linear combination of fixed nonlinear functions ϕ (called basis functions)

$$y(x, w) = w_0 + \sum_{i=1}^{n} w_i \phi_i(x)$$

Regression

Regression predicts value of continuous a target variable

- A simplest model for regression can be a linear combination of the input variables

$$y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$$

where x is a n dimensional vector $(x_1, x_2, ..., x_n)$ representing some feature

- It can be extended by considering linear combination of fixed nonlinear functions ϕ (called basis functions)

$$y(x, w) = w_0 + \sum_{i=1}^{n} w_i \phi_i(x)$$

- In short $y(x, w) = w^T \phi(x)$

Regression

Regression predicts value of continuous a target variable

- A simplest model for regression can be a linear combination of the input variables

$$y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$$

  where x is a n dimensional vector $(x_1, x_2, ..., x_n)$ representing some feature

- It can be extended by considering linear combination of fixed nonlinear functions ϕ (called basis functions)

$$y(x, w) = w_0 + \sum_{i=1}^{n} w_i \phi_i(x)$$

- In short $y(x, w) = w^T \phi(x)$
- Objective is to choose w such that it makes $y(x^{(i)}, w)$ as close to $y^{(i)}$ as possible

Our Regression Example

- If we could correct estimate the values of w's we could determine $y(x^{(i)}, w)$ for all values

x_1	x_2	x_3	y	$y(x^{(i)}, w)$
10	50	20	10	8
11	31	22	12	9
11	12	15	4	3
20	55	20	22	26
23	41	27	1	1
31	12	35	9	4
13	18	12	23	30
21	55	16	16	13
32	56	27	22	21
8	22	35	??	6

Now the question is that how good this w is?

Regression

- Determining w, is similar to solving a minimization problem. Let us define a **squared error cost function** as

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y(x^{(i)}, w) - y^{(i)})^2$$

where m is number of training examples

Regression

- Determining w, is similar to solving a minimization problem. Let us define a **squared error cost function** as

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y(x^{(i)}, w) - y^{(i)})^2$$

where m is number of training examples

- Then one have to minimize the value of $J(w)$

$$\underset{w}{\operatorname{argmin}}\, J(w)$$

Regression

- Determining w, is similar to solving a minimization problem. Let us define a **squared error cost function** as

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)})^2$$

where m is number of training examples

- Then one have to minimize the value of $J(w)$

$$\operatorname*{argmin}_{w} J(w)$$

- Basic idea: Push w_i a bit against the direction of its gradient

Linear Regression

x_1	x_2	x_3	y	$y(x^{(i)}, w)$	$(y(x^{(i)}, w) - y)^2$
10	50	20	10	8	4
11	31	22	12	9	9
11	12	15	4	3	1
20	55	20	22	26	16
23	41	27	1	1	0
31	12	35	9	4	25
13	18	12	23	30	49
21	55	16	16	13	9
32	56	27	22	21	1

Assume for some w we computed $y(x^{(i)}, w)$ then

$$\begin{aligned} J(w) &= \frac{1}{2 \times 9} \times 114 \\ &= 6.33 \end{aligned}$$

Gradient Descent

Algorithm 2: Gradient Descent

Initialize w randomly
repeat
  Simultaneously update all w_j with
  $w_j - \alpha \frac{\partial}{\partial w_j} J(w)$
until *converge*;
return w

Gradient Descent

Algorithm 3: Gradient Descent

Initialize w randomly
repeat
    Simultaneously update all w_j with
    $w_j - \alpha \frac{\partial}{\partial w_j} J(w)$
until *converge*;
return w

- Here α is a learning rate. If α is small enough then $J(w)$ would decrease in every iteration

Gradient Descent

Algorithm 4: Gradient Descent

Initialize w randomly
repeat
    Simultaneously update all w_j with
    $w_j - \alpha \frac{\partial}{\partial w_j} J(w)$
until *converge*;
return w

- Here α is a learning rate. If α is small enough then $J(w)$ would decrease in every iteration
  (large α can overshoot the minimum and may fail to converge)

Gradient Descent

Algorithm 5: Gradient Descent

```
Initialize w randomly
repeat
    Simultaneously update all w_j with
    w_j − α ∂/∂w_j J(w)
until converge;
return w
```

- Here α is a learning rate. If α is small enough then $J(w)$ would decrease in every iteration
  (large α can overshoot the minimum and may fail to converge)
- Susceptible to local minimum

Gradient Descent

Algorithm 6: Gradient Descent

```
Initialize w randomly
repeat
    Simultaneously update all w_j with
    w_j − α ∂/∂w_j J(w)
until converge;
return w
```

- Here α is a learning rate. If α is small enough then $J(w)$ would decrease in every iteration
  (large α can overshoot the minimum and may fail to converge)
- Susceptible to local minimum
- As it moves closer to local minimum, it automatically takes smaller steps as gradient decreases

Batch-Gradient Descent

Algorithm 7: Batch-Gradient Descent

1. Initialize w randomly
2. **repeat**
3. Simultaneously update all w_j with $w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$
4. **until** *converge*;
5. **return** w

Batch-Gradient Descent

Algorithm 8: Batch-Gradient Descent

Initialize w randomly
repeat
    Simultaneously update all w_j with
    $w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$
until *converge*;
return w

- At every step it evaluate all training examples

Batch-Gradient Descent

Algorithm 9: Batch-Gradient Descent

```
Initialize w randomly
repeat
    Simultaneously update all w_j with
    w_j − α (1/m) Σ_{i=1}^{m} (y(x^(i), w) − y^(i)) x_j^(i)
until converge;
return w
```

Initialize w randomly

repeat

  Simultaneously update all w_j with

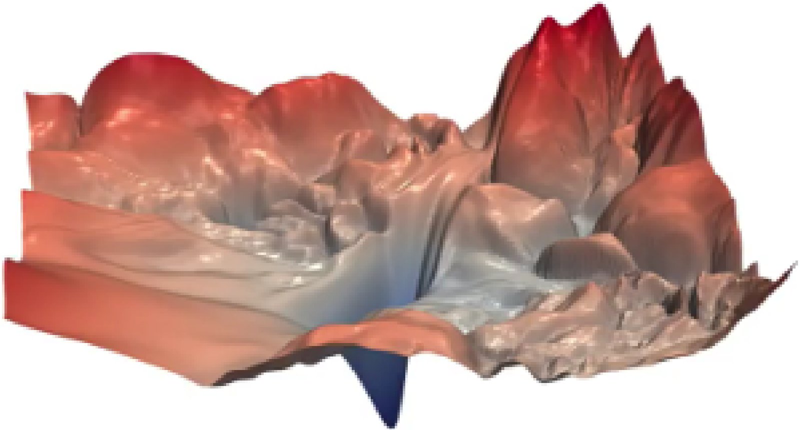
  $w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$

until *converge*;

return w

- At every step it evaluate all training examples
- Some time it is also called multi-variate linear regression

Real Landscape



Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Learning rate $\alpha = 0.1$

Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Learning rate $\alpha = 0.1$

J=396.662506
w=(0.500 0.500 0.500 0.500)

J=19454472.000000
w=(-2.055 -51.070 -100.970 -62.640)

J=1036526813184.000000
w=(590.236 11518.771 23902.906 13778.349)

J=55230041021218816.000000
w=(-135891.922 -2653678.250 -5525792.000 -3170425.000)

J=2942865354556228763648.000000
w=(31365378.000 612476928.000 1275658624.000 731686912.000)

J=156806972273681738831495168.000000
w=(-7240111104.000 -141378551808.000 -294465732608.000 -168895037440.000)

J=8355266546526971027269827428352.000000
w=(1671254376448.000 32634791002112.000 67972370530304.000 38986479304704.000)

J=445200079222591879770706068887306240.000000

w=(-385780270759936.000 -7533178826784768.000

-15690251045437440.000 -8999357718200320.000)

Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Learning rate $\alpha = 0.001$

Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Learning rate $\alpha = 0.001$

J	w
396.663	(0.500 0.500 0.500 0.500)
664.137	(0.474 -0.016 -0.515 -0.131)
1131.021	(0.508 0.631 0.881 0.628)
1943.882	(0.464 -0.249 -0.910 -0.435)
3357.625	(0.523 0.888 1.492 0.914)
5815.401	(0.446 -0.630 -1.641 -0.908)
10087.491	(0.549 1.356 2.518 1.456)
17512.684	(0.415 -1.274 -2.941 -1.693)
30417.834	(0.592 2.183 4.276 2.432)
52847.020	(0.359 -2.383 -5.221 -3.028)
91828.805	(0.668 3.630 7.314 4.151)
159578.781	(0.263 -4.302 -9.200 -5.330)
277327.562	(0.799 6.152 12.580 7.155)
481973.594	(0.093 -7.633 -16.125 -9.316)
837646.250	(1.025 10.537 21.725 12.387)
1455801.375	(-0.201 -13.418 -28.168 -16.234)
2530147.500	(1.417 18.162 37.611 21.491)
4397349.000	(-0.715 -23.472 -49.103 -28.249)
7642525.500	(2.097 31.415 65.218 37.319)
13282603.000	(-1.608 -40.944 -85.492 -49.126)
23084998.000	(3.278 54.449 113.196 64.832)
40121436.000	(-3.162 -71.310 -148.738 -85.405)
69730584.000	(5.329 94.483 196.578 112.653)
121190936.000	(-5.863 -124.085 -258.660 -148.456)
210628448.000	(8.894 164.060 341.494 195.769)
366069856.000	(-10.559 -215.809 -449.705 -258.035)
636225152.000	(15.088 284.983 593.355 340.226)
1105751936.000	(-18.721 -375.224 -781.739 -448.479)
1921783808.000	(25.852 495.147 1031.086 591.291)
3340036608.000	(-32.908 -652.287 -1358.811 -779.468)

Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Learning rate $\alpha = 0.0001$

Example: Gradient Descent (learning rate α)

Consider following data

	x_1	x_2	x_3	y
1	10	50	20	10
2	11	31	22	12
3	11	12	15	4
4	20	55	20	22
5	23	41	27	1
6	31	12	35	9
7	13	18	12	23
8	21	55	16	16
9	32	56	27	22
10	8	22	35	11

Learning rate $\alpha = 0.0001$

J	w
396.663	(0.500 0.500 0.500 0.500)
246.798	(0.497 0.448 0.399 0.437)
158.286	(0.495 0.408 0.321 0.388)
105.980	(0.494 0.377 0.262 0.349)
75.041	(0.493 0.353 0.218 0.319)
56.711	(0.492 0.334 0.184 0.295)
45.826	(0.491 0.320 0.159 0.276)
39.335	(0.491 0.308 0.140 0.260)
35.439	(0.490 0.299 0.126 0.248)
33.077	(0.490 0.291 0.115 0.238)
31.621	(0.490 0.285 0.108 0.229)
30.703	(0.490 0.280 0.103 0.222)
30.104	(0.490 0.276 0.099 0.216)
29.694	(0.489 0.273 0.097 0.210)
29.399	(0.489 0.270 0.096 0.206)
29.172	(0.489 0.268 0.095 0.202)
28.987	(0.489 0.266 0.096 0.198)
28.830	(0.489 0.264 0.096 0.194)
28.689	(0.489 0.262 0.097 0.191)
28.560	(0.489 0.260 0.098 0.188)
28.439	(0.489 0.259 0.099 0.185)
28.325	(0.489 0.258 0.101 0.182)
28.216	(0.489 0.256 0.102 0.179)
28.111	(0.489 0.255 0.104 0.177)
28.011	(0.489 0.254 0.105 0.174)
27.913	(0.489 0.253 0.107 0.172)
27.819	(0.489 0.252 0.109 0.170)
27.728	(0.489 0.251 0.110 0.167)
27.555	(0.490 0.249 0.114 0.163)
24.926	(0.507 0.207 0.215 0.020) **Iteration 300**
24.768	(0.710 0.219 0.213 0.005) **Iteration 3000**

Example: Gradient Descent (Feature scaling)

Feature scaling

	x_1	x_2	x_3	y
1	0.08	0.86	0.35	10
2	0.12	0.43	0.43	12
3	0.12	0.00	0.13	4
4	0.50	0.98	0.35	22
5	0.62	0.66	0.65	1
6	0.96	0.00	1.00	9
7	0.21	0.14	0.00	23
8	0.54	0.98	0.17	16
9	1.00	1.00	0.65	22
10	0.00	0.23	1.00	11

Learning rate $\alpha = 0.1$

Example: Gradient Descent (Feature scaling)

Feature scaling

	x_1	x_2	x_3	y
1	0.08	0.86	0.35	10
2	0.12	0.43	0.43	12
3	0.12	0.00	0.13	4
4	0.50	0.98	0.35	22
5	0.62	0.66	0.65	1
6	0.96	0.00	1.00	9
7	0.21	0.14	0.00	23
8	0.54	0.98	0.17	16
9	1.00	1.00	0.65	22
10	0.00	0.23	1.00	11

Learning rate $\alpha = 0.1$

J	w
95.472	(0.500 0.500 0.500 0.500)
73.399	(1.679 1.025 1.220 0.983)
58.326	(2.658 1.455 1.822 1.364)
48.020	(3.470 1.808 2.326 1.663)
40.961	(4.147 2.096 2.749 1.893)
36.116	(4.710 2.331 3.106 2.066)
32.778	(5.180 2.522 3.407 2.193)
30.468	(5.574 2.677 3.662 2.283)
28.859	(5.903 2.803 3.880 2.341)
27.729	(6.181 2.904 4.066 2.373)
26.925	(6.415 2.985 4.226 2.385)
26.344	(6.613 3.049 4.364 2.379)
25.916	(6.782 3.100 4.485 2.360)
25.593	(6.926 3.140 4.590 2.329)
25.342	(7.050 3.170 4.683 2.289)
25.141	(7.158 3.193 4.766 2.241)
24.974	(7.252 3.210 4.839 2.188)
24.833	(7.334 3.222 4.906 2.129)
24.708	(7.407 3.230 4.966 2.067)**Iteration 18**
24.596	(7.472 3.234 5.021 2.003)
24.493	(7.530 3.236 5.071 1.935)
24.397	(7.583 3.235 5.118 1.866)
24.306	(7.632 3.233 5.161 1.796)
24.219	(7.677 3.229 5.202 1.725)
24.136	(7.718 3.225 5.241 1.653)
24.056	(7.757 3.219 5.277 1.581)
23.979	(7.794 3.213 5.311 1.509)
23.903	(7.830 3.206 5.344 1.436)
23.830	(7.863 3.198 5.375 1.364)
23.759	(7.896 3.191 5.405 1.292) **Iteration 30**
20.174	(12.021 4.618 4.794 -7.329) **Iteration 3000**

Similar Mechanism for Classification

Classification have predefined fixed number of labels (0 and 1 in this case)

x_1	x_2	x_3	*Class*
10	50	20	1
11	31	22	1
11	12	15	0
20	55	20	0
23	41	27	0
31	12	35	1
13	18	12	0
21	55	16	1
32	56	27	0
8	22	35	??

What should come at the place of ??

Logistic Regression

Moving from linear regression $y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$ to **logistic regression**

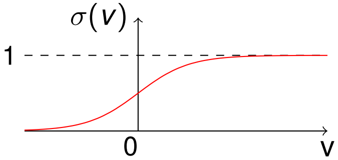
$$y(x, w) = \sigma(w_0 + w_1 x_1 + ... + w_n x_n)$$

Logistic Regression

Moving from linear regression $y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$ to **logistic regression**

$$y(x, w) = \sigma(w_0 + w_1 x_1 + ... + w_n x_n)$$

- Enables "classification" apart from the regression. Where σ is called as sigmoid function that produces values in range $[0, 1]$ and is defined as $\sigma(v) = \frac{1}{1+e^{-v}}$



Decision on classification

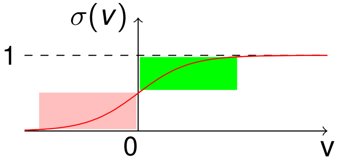
$$classification = \begin{cases} 1 & \text{if } y(x, w) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Logistic Regression

Moving from linear regression $y(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$ to **logistic regression**

$$y(x, w) = \sigma(w_0 + w_1 x_1 + ... + w_n x_n)$$

- Enables "classification" apart from the regression. Where σ is called as sigmoid function that produces values in range $[0, 1]$ and is defined as $\sigma(v) = \frac{1}{1+e^{-v}}$



Decision on classification

$$classification = \begin{cases} 1 & \text{if } y(x, w) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Decision Boundary in Logistic Regression

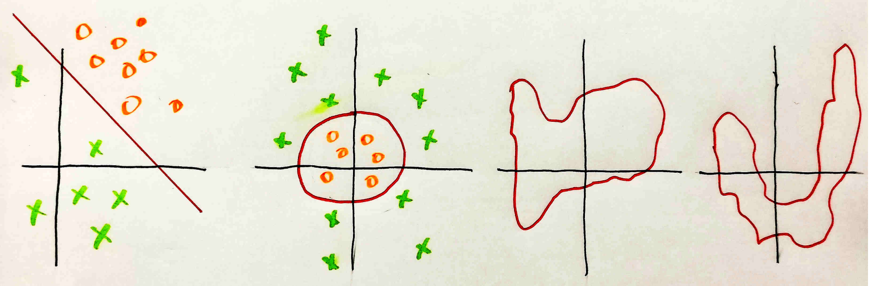
$$classification = \begin{cases} 1 & \text{if } y(x, w) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- This choice of w partitions the space into two sections and the hyper-plane separating them is called **decision boundary**

Decision Boundary in Logistic Regression

$$classification = \begin{cases} 1 & \text{if } y(x, w) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- This choice of w partitions the space into two sections and the hyper-plane separating them is called **decision boundary**
- By adding more complex or polynomial terms one can get more complex decision boundary



Cost Function

- Cost function used for the liner regression

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y(x^{(i)}, w) - y^{(i)})^2$$

becomes a non convex function in case of logistic regression

Cost Function

- Cost function used for the liner regression

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y(x^{(i)}, w) - y^{(i)})^2$$

becomes a non convex function in case of logistic regression

Therefore, a different cost function is chosen

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} Cost(y(x^{(i)}, w), y^{(i)})$$

where

$$Cost(y(x^{(i)}, w), y^{(i)}) = \begin{cases} -\log(y(x^{(i)}, w)) & \text{if } y^{(i)} = 1 \\ -\log(1 - y(x^{(i)}, w)) & \text{otherwise} \end{cases}$$

A simplified version of this cost function is

$Cost(y(x^{(i)}, w), y^{(i)}) = -y^{(i)} \log(y(x^{(i)}, w)) - (1 - y^{(i)}) \log(1 - y(x^{(i)}, w))$

Learning With This Cost Function

- Learning corresponds to the minimization of $J(w)$ by changing w

$$\underset{w}{\operatorname{argmin}} J(w) = \frac{1}{m} \sum_{i=1}^{m} Cost(y(x^{(i)}, w), y^{(i)})$$

Learning With This Cost Function

- Learning corresponds to the minimization of $J(w)$ by changing w

$$\operatorname*{argmin}_{w} J(w) = \frac{1}{m}\sum_{i=1}^{m} Cost(y(x^{(i)}, w), y^{(i)})$$

$$\operatorname*{argmin}_{w} J(w) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log(y(x^{(i)}, w)) - (1 - y^{(i)})\log(1 - y(x^{(i)}, w))]$$

Learning With This Cost Function

- Learning corresponds to the minimization of $J(w)$ by changing w

$$\underset{w}{\operatorname{argmin}}\, J(w) = \frac{1}{m}\sum_{i=1}^{m} Cost(y(x^{(i)}, w), y^{(i)})$$

$$\underset{w}{\operatorname{argmin}}\, J(w) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log(y(x^{(i)}, w)) - (1 - y^{(i)})\log(1 - y(x^{(i)}, w))]$$

- Gradient Descent can be used for this purpose

Algorithm 12: Logistic Regression

1. Initialize w randomly
2. **repeat**
3. Simultaneously update all w_j with $w_j - \alpha\frac{\partial}{\partial w_j}J(w)$
4. **until** *converge*;
5. **return** w

The Partial Derivative

Partial derivative term

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(y(x^{(i)}, w)) - (1 - y^{(i)}) \log(1 - y(x^{(i)}, w))]$$

The Partial Derivative

Partial derivative term

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(y(x^{(i)}, w)) - (1 - y^{(i)}) \log(1 - y(x^{(i)}, w))]$$

comes out to be

$$\frac{\partial}{\partial w_j} J(w) = \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$$

The Partial Derivative

Partial derivative term

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(y(x^{(i)}, w)) - (1 - y^{(i)}) \log(1 - y(x^{(i)}, w))]$$

comes out to be

$$\frac{\partial}{\partial w_j} J(w) = \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$$

Algorithm 15: Logistic Regression

1 Initialize w randomly
2 **repeat**
3 Simultaneously update all w_j with $w_j - \alpha \times \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$
4 **until** *converge*;
5 **return** w

The Partial Derivative

Partial derivative term

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(y(x^{(i)}, w)) - (1 - y^{(i)}) \log(1 - y(x^{(i)}, w))]$$

comes out to be

$$\frac{\partial}{\partial w_j} J(w) = \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$$

Algorithm 16: Logistic Regression

Initialize w randomly
repeat
    Simultaneously update all w_j with
    $w_j - \alpha \times \frac{1}{m} \sum_{i=1}^{m} (y(x^{(i)}, w) - y^{(i)}) x_j^{(i)}$
until *converge*;
return w

It looks identical to liner regression but, $y(x^{(i)}, w)$ is different here $y(x^{(i)}, w) = \frac{1}{1 + e^{-(w_0 + w_1 x_1^{(i)} + \ldots + w_n x_n^{(i)})}}$

Example: Logistic Regression

Consider following data

	x_1	x_2	x_3	*Class*
1	2	2	2	1
2	3	2	2	1
3	2	3	2	1
4	2	2	3	1
5	7	6	9	0
6	9	7	6	0
7	9	6	7	0
8	6	8	9	0
9	8	9	6	0
10	8	9	9	0

Learning rate $\alpha = 0.01$

Example: Logistic Regression

Consider following data

	x_1	x_2	x_3	*Class*
1	2	2	2	1
2	3	2	2	1
3	2	3	2	1
4	2	2	3	1
5	7	6	9	0
6	9	7	6	0
7	9	6	7	0
8	6	8	9	0
9	8	9	6	0
10	8	9	9	0

Learning rate $\alpha = 0.01$

J	w
6.912	(0.500 0.500 0.500 0.500)
6.496	(0.494 0.453 0.455 0.454)
5.944	(0.488 0.406 0.410 0.408)
5.316	(0.482 0.360 0.366 0.363)
4.692	(0.477 0.313 0.321 0.317)
4.072	(0.471 0.267 0.277 0.272)
3.460	(0.465 0.221 0.233 0.227)
2.860	(0.460 0.175 0.189 0.182)
2.279	(0.454 0.130 0.146 0.138)
1.735	(0.449 0.086 0.104 0.095)
1.262	(0.445 0.044 0.064 0.054)
0.906	(0.441 0.008 0.029 0.018)
0.685	(0.438 -0.022 0.000 -0.011)
0.566	(0.437 -0.044 -0.020 -0.032)
0.504	(0.436 -0.060 -0.035 -0.048)
0.470	(0.436 -0.072 -0.046 -0.059)
0.451	(0.436 -0.081 -0.055 -0.068)
0.438	(0.436 -0.088 -0.061 -0.074)
0.431	(0.437 -0.093 -0.066 -0.080)
0.425	(0.438 -0.098 -0.070 -0.084)
0.422	(0.439 -0.101 -0.074 -0.088)
0.419	(0.440 -0.105 -0.077 -0.091)
0.417	(0.441 -0.107 -0.079 -0.093)
0.416	(0.443 -0.110 -0.081 -0.095)
0.415	(0.444 -0.112 -0.082 -0.097) **Iteration 25**
0.412	(0.451 -0.119 -0.088 -0.103) **Iteration 30**
0.348	(0.857 -0.179 -0.084 -0.132) **Iteration 300**
0.116	(3.256 -0.409 -0.135 -0.291) **Iteration 3000**
0.012	(7.596 -0.748 -0.361 -0.588) **Iteration 30000**
0.001	(11.975 -1.091 -0.599 -0.896) **Iteration 300000**

Example: Find J

As (w_0, w_1, w_2, w_3)=$(0.5, 0.5, 0.5, 0.5)$, $v = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$
$y(x^{(i)}, w) = \sigma(v)$
And log term is $-y^{(i)} \log(y(x^{(i)}, w)) - (1 - y^{(i)}) \log(1 - y(x^{(i)}, w))$

i	x_1	x_2	x_3	$y^{(i)}$	v	$y(x^{(i)}, w)$	log term
1	2	2	2	1	3.5	0.970	0.029
2	3	2	2	1	4.0	0.982	0.018
3	2	3	2	1	4.0	0.982	0.018
4	2	2	3	1	4.0	0.982	0.018
5	7	6	9	0	11.5	0.999	11.49
6	9	7	6	0	11.5	0.999	11.49
7	9	6	7	0	11.5	0.999	11.49
8	6	8	9	0	12	0.999	11.51
9	8	9	6	0	12	0.999	11.51
10	8	9	9	0	13	0.999	11.51
						Total/10:	6.9118

Example: Find next W

Let (w_0, w_1, w_2, w_3)=$(0.5, 0.5, 0.5, 0.5)$ and $t_i = (y(x^{(i)}, w) - y^{(i)})x_j^{(i)}$

Then $\frac{1}{m}\sum_{i=1}^{m}(y(x^{(i)}, w) - y^{(i)})x_j^{(i)} = \frac{1}{m}\sum_{i=1}^{m} t_i$ let $\hat{y}^{(i)} = y(x^{(i)}, w)$

Then update w_j with $w_j - \alpha \times \frac{1}{m}\sum_{i=1}^{m} t_i$ we have set $\alpha = 0.01$

i	x_0	x_1	x_2	x_3	$y^{(i)}$	$\hat{y}^{(i)}$	t_0	t_1	t_2	t_3
1	1	2	2	2	1	0.970	-0.029	-0.058	-0.058	-0.058
2	1	3	2	2	1	0.982	-0.017	-0.053	-0.035	-0.035
3	1	2	3	2	1	0.982	-0.017	-0.035	-0.053	-0.035
4	1	2	2	3	1	0.982	-0.017	-0.035	-0.035	-0.053
5	1	7	6	9	0	0.999	0.999	6.999	5.999	8.999
6	1	9	7	6	0	0.999	0.999	8.999	6.999	5.999
7	1	9	6	7	0	0.999	0.999	8.999	5.999	6.999
8	1	6	8	9	0	0.999	0.999	5.999	7.999	8.999
9	1	8	9	6	0	0.999	0.999	7.999	8.999	5.999
10	1	8	9	9	0	0.999	0.999	7.999	8.999	8.999
						Total	5.916	46.815	44.815	45.815
						$w_i - \alpha \times (total/m)$	0.494	0.453	0.455	0.454

Example: Classification across Iterations

Following table shows classification as the weights get modified along 1^{st}, 100^{th}, 300^{th} and 500^{th} iteration

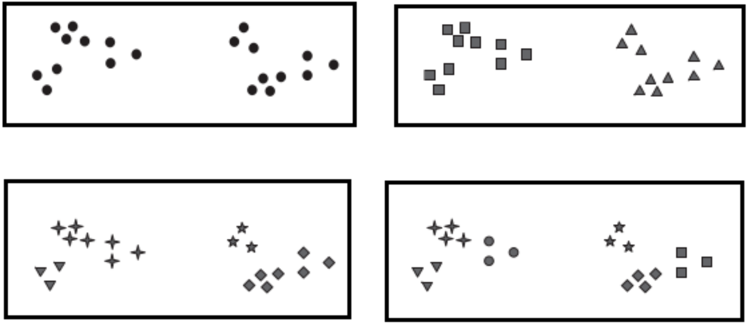
i	x_1	x_2	x_3	$y^{(i)}$	1	100	300	500
1	2	2	2	1	1	0	1	1
2	3	2	2	1	1	0	0	1
3	2	3	2	1	1	0	1	1
4	2	2	3	1	1	0	1	1
5	7	6	9	0	1	0	0	0
6	9	7	6	0	1	0	0	0
7	9	6	7	0	1	0	0	0
8	6	8	9	0	1	0	0	0
9	8	9	6	0	1	0	0	0
10	8	9	9	0	1	0	0	0

Clustering

Grouping data based on their homogeneity (similarity or closeness).

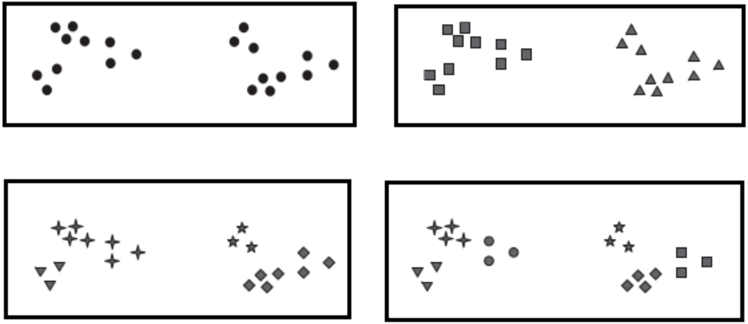
Clustering

Grouping data based on their homogeneity (similarity or closeness).



Clustering

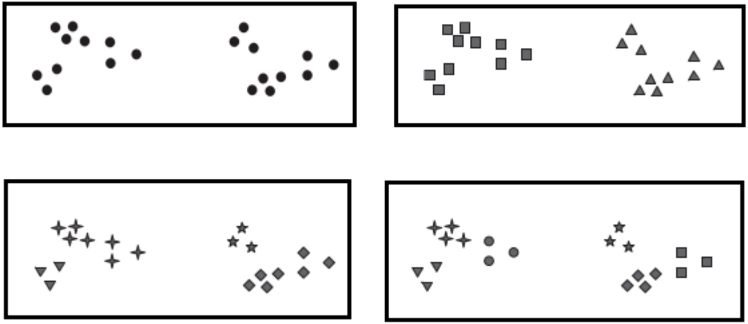
Grouping data based on their homogeneity (similarity or closeness).



Objects within a group are similar (or related) and are different from the objects in other groups.

Clustering

Grouping data based on their homogeneity (similarity or closeness).



Objects within a group are similar (or related) and are different from the objects in other groups. When it is better?

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)

[3] object is closer (more similar) to a prototype

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization,

[3] object is closer (more similar) to a prototype

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression,

[3] object is closer (more similar) to a prototype

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression, and 3) Efficiently Finding Nearest Neighbors

[3] object is closer (more similar) to a prototype

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression, and 3) Efficiently Finding Nearest Neighbors
- **Type:**
  - Hierarchical (nested) versus Partitional

[3] object is closer (more similar) to a prototype

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression, and 3) Efficiently Finding Nearest Neighbors
- **Type:**
  - Hierarchical (nested) versus Partitional
  - Exclusive versus Overlapping versus Fuzzy

[3] object is closer (more similar) to a prototype

Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression, and 3) Efficiently Finding Nearest Neighbors
- **Type:**
  - Hierarchical (nested) versus Partitional
  - Exclusive versus Overlapping versus Fuzzy
  - Complete versus Partial

[3] object is closer (more similar) to a prototype

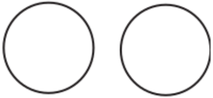
Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression, and 3) Efficiently Finding Nearest Neighbors
- **Type:**
  - Hierarchical (nested) versus Partitional
  - Exclusive versus Overlapping versus Fuzzy
  - Complete versus Partial
- **K-means:** This is a prototype-based[3], partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids.

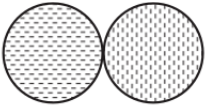
---

[3] object is closer (more similar) to a prototype

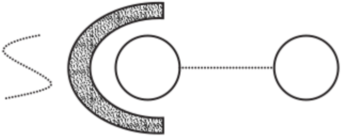
Clustering Approaches



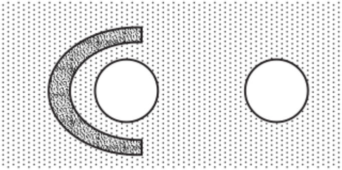
Well-separated clusters.



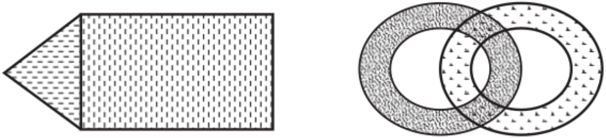
Center-based clusters.



Contiguity-based clusters.



Density-based clusters.



Conceptual clusters.

K-means Algorithm

Number of clusters *i.e.* the value of K is provided by the user

K-means Algorithm

Number of clusters *i.e.* the value of K is provided by the user

Algorithm 18: K-means

```
Randomly select K points as centroids
repeat
    foreach datum point d_i do
        Assign d_i to one of the closest centroids
        (thereby forming K clusters)
    Recompute centroid (mean) for each cluster
until The centroids converge;
```

K-means Algorithm

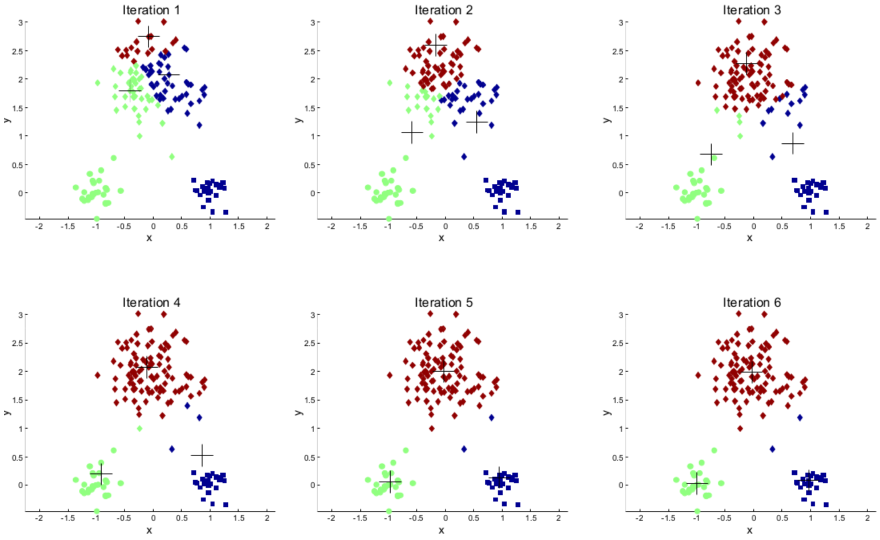
Number of clusters *i.e.* the value of K is provided by the user

Algorithm 19: K-means

Randomly select K points as centroids
repeat
    foreach *datum point* d_i **do**
        Assign d_i to one of the closest centroids (thereby forming K clusters)
    Recompute centroid (mean) for each cluster
until *The centroids converge*;

Closeness is measured by **Euclidean distance**, cosine similarity, correlation, Bregman divergence *etc*

K-means in Action



Evaluation of K-means[4]

For a given data set $\{x_1, x_2, ..., x_n\}$, let K-means partitions it in $\{S_1, S_2, ..., S_K\}$ then the objective is

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} dist^2(x, \mu_i)$$

[4] Hamerly, Greg and Elkan, Charles, "Learning the k in k-means", pp 281–288, NIPS-2003

Evaluation of K-means[4]

For a given data set $\{x_1, x_2, ..., x_n\}$, let K-means partitions it in $\{S_1, S_2, ..., S_K\}$ then the objective is

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} dist^2(x, \mu_i)$$

where μ_i corresponds to i^{th} centroid. $\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$

- Typical choice for *dist* function is Euclidean Distance

[4] Hamerly, Greg and Elkan, Charles, “Learning the k in k-means”, pp 281–288, NIPS-2003

Evaluation of K-means[4]

For a given data set $\{x_1, x_2, ..., x_n\}$, let K-means partitions it in $\{S_1, S_2, ..., S_K\}$ then the objective is

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} dist^2(x, \mu_i)$$

where μ_i corresponds to i^{th} centroid. $\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$

- Typical choice for *dist* function is Euclidean Distance

How to proceed?

- Choose a K (How?)
  - Run K-means algorithm multiple times
  - Choose clusters corresponding to the one that minimized sum of squared error (SSE)

[4] Hamerly, Greg and Elkan, Charles, "Learning the k in k-means", pp 281–288, NIPS-2003

Evaluation of K-means[4]

For a given data set $\{x_1, x_2, ..., x_n\}$, let K-means partitions it in $\{S_1, S_2, ..., S_K\}$ then the objective is

$$\underset{S}{\text{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} dist^2(x, \mu_i)$$

where μ_i corresponds to i^{th} centroid. $\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$

- Typical choice for *dist* function is Euclidean Distance

How to proceed?

- Choose a K (How?)
  - Run K-means algorithm multiple times
  - Choose clusters corresponding to the one that minimized sum of squared error (SSE)
- If $K == n$, no error.

[4] Hamerly, Greg and Elkan, Charles, "Learning the k in k-means", pp 281–288, NIPS-2003

Evaluation of K-means[4]

For a given data set $\{x_1, x_2, ..., x_n\}$, let K-means partitions it in $\{S_1, S_2, ..., S_K\}$ then the objective is

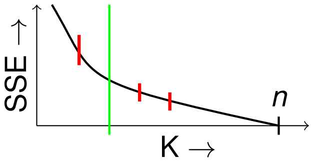
$$\underset{S}{\text{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} dist^2(x, \mu_i)$$

where μ_i corresponds to i^{th} centroid. $\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$

- Typical choice for *dist* function is Euclidean Distance

How to proceed?

- Choose a K (How?)
  - Run K-means algorithm multiple times
  - Choose clusters corresponding to the one that minimized sum of squared error (SSE)
- If $K == n$, no error.
- Good clustering has smaller K



[4] Hamerly, Greg and Elkan, Charles, “Learning the k in k-means”, pp 281–288, NIPS-2003

Evaluation of K-means

- **Choosing K:**

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge,

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm,

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:**

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space,

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space, 2) Random point of data,

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space, 2) Random point of data, 3) look for dense region, 4) Space uniformly in feature space
- **Cluster Quality:**

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space, 2) Random point of data, 3) look for dense region, 4) Space uniformly in feature space
- **Cluster Quality:** 1) Diameter of cluster verses Inter-cluster distance,

Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space, 2) Random point of data, 3) look for dense region, 4) Space uniformly in feature space
- **Cluster Quality:** 1) Diameter of cluster verses Inter-cluster distance, 2) Distance between members of a cluster and the cluster center,

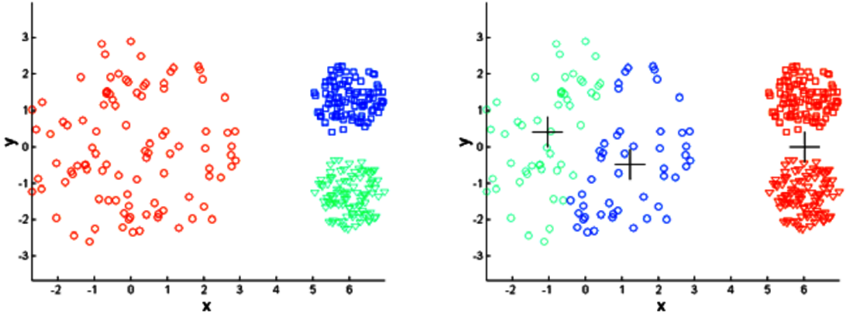
Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space, 2) Random point of data, 3) look for dense region, 4) Space uniformly in feature space
- **Cluster Quality:** 1) Diameter of cluster verses Inter-cluster distance, 2) Distance between members of a cluster and the cluster center, 3) Diameter of smallest sphere,

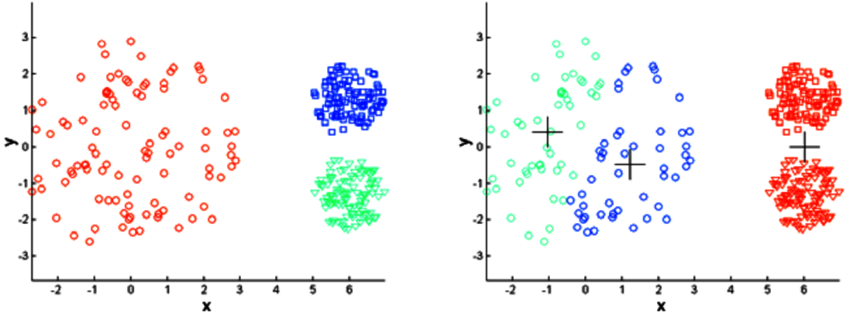
Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on K
- **Initialization of Centers:** 1) Random point in space, 2) Random point of data, 3) look for dense region, 4) Space uniformly in feature space
- **Cluster Quality:** 1) Diameter of cluster verses Inter-cluster distance, 2) Distance between members of a cluster and the cluster center, 3) Diameter of smallest sphere, 4) Ability to discover hidden patterns

Limitations of K-means

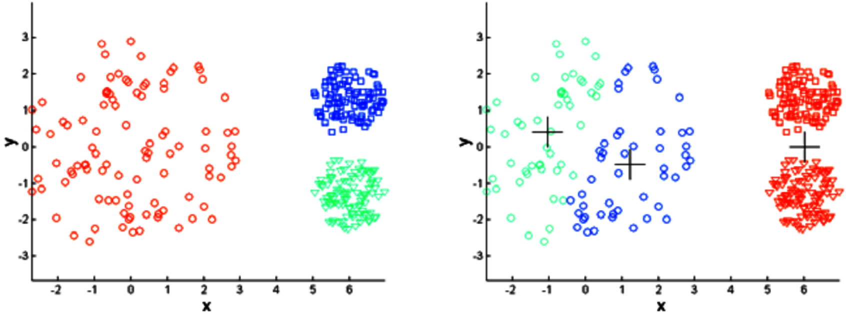


Limitations of K-means



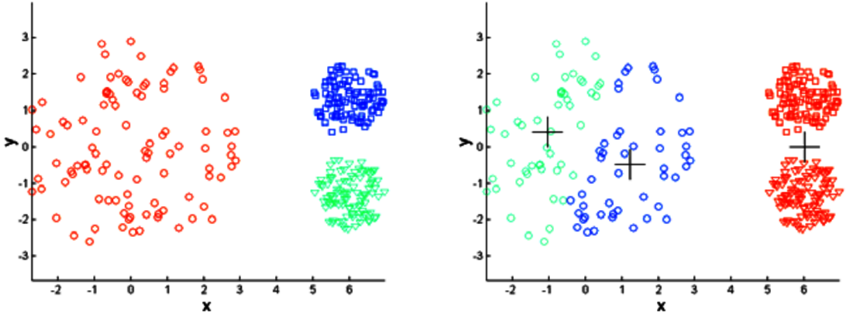
- Has problem when data has
  - Different size clusters

Limitations of K-means



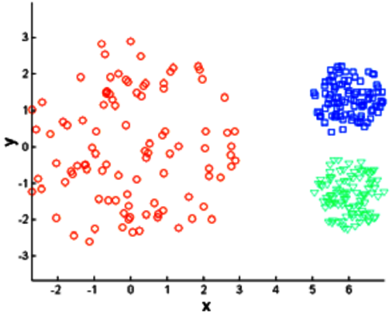
- Has problem when data has
  - Different size clusters
  - Different densities

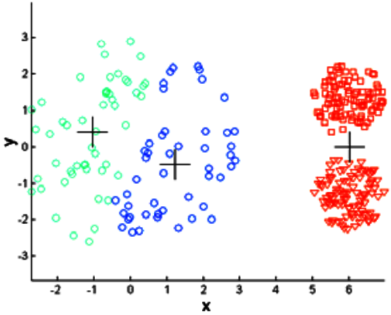
Limitations of K-means



- Has problem when data has
  - Different size clusters
  - Different densities
  - Non-globular shape

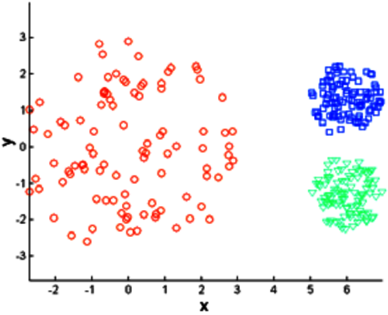
Limitations of K-means

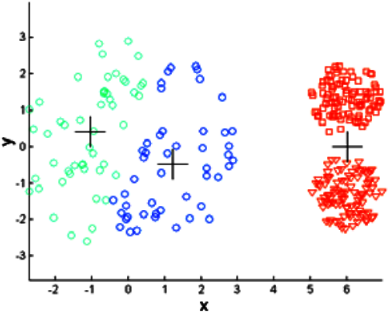




- Has problem when data has
  - Different size clusters
  - Different densities
  - Non-globular shape
- Handling Empty Clusters

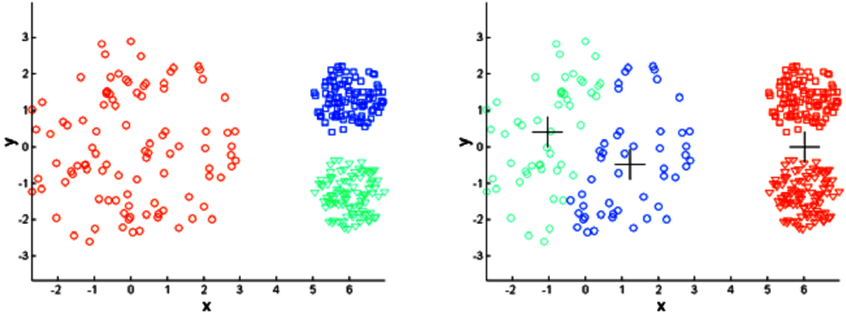
Limitations of K-means





- Has problem when data has
  - Different size clusters
  - Different densities
  - Non-globular shape
- Handling Empty Clusters
- When there are outliers

Limitations of K-means



- Has problem when data has
  - Different size clusters
  - Different densities
  - Non-globular shape
- Handling Empty Clusters
- When there are outliers
- Updating Centroids Incrementally

Important Note:

- K-Means and K-NN are different (K nearest neighbors)

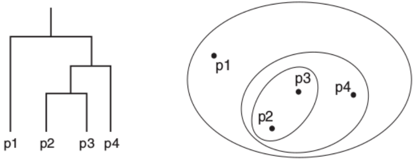
K-NN is a **supervised** approach for **classification**

Other Clustering Approaches

- **K-Medoids:** chooses data point as center and minimizes a sum of pairwise dissimilarities. Resistance to noise and/or outliers

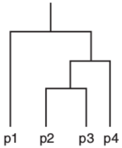
Other Clustering Approaches

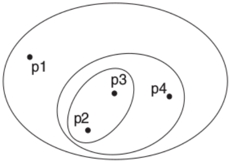
- **K-Medoids:** chooses data point as center and minimizes a sum of pairwise dissimilarities. Resistance to noise and/or outliers
- **Agglomerative Hierarchical Clustering:** repeatedly merging the two closest clusters until a single (Single Link)



Other Clustering Approaches

- **K-Medoids:** chooses data point as center and minimizes a sum of pairwise dissimilarities. Resistance to noise and/or outliers
- **Agglomerative Hierarchical Clustering:** repeatedly merging the two closest clusters until a single (Single Link)





- **DBSCAN:** density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm.