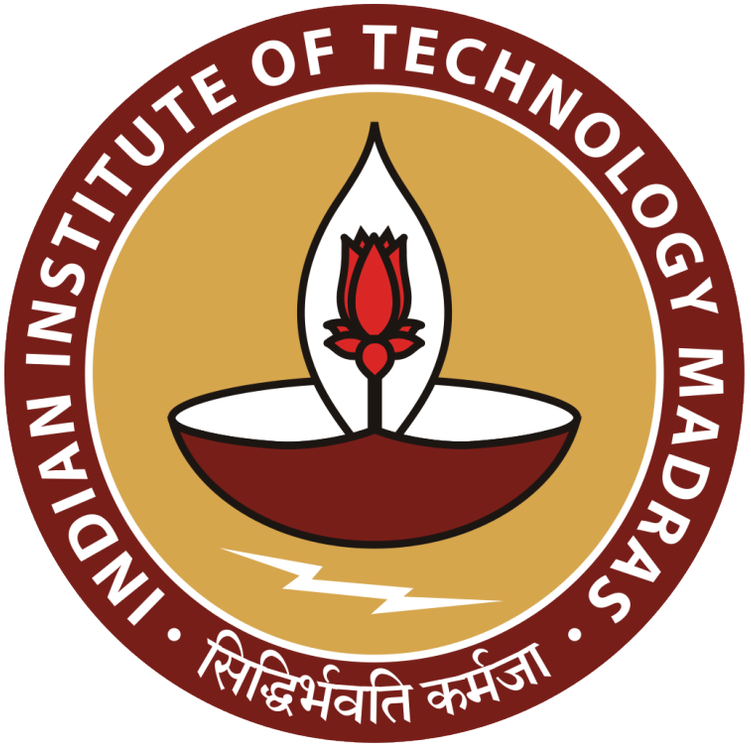

IIT Madras

ONLINE DEGREE

Bottom-Up Computing

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**
- We can consider two kinds of problems that are suitable for bottom-up computing:

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**
- We can consider two kinds of problems that are suitable for bottom-up computing:
  - In **Classification** problems, the task is to label (or identify) a given data element: identify a character from handwriting, a face from a picture, a disease from its symptoms, etc.

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**
- We can consider two kinds of problems that are suitable for bottom-up computing:
  - In **Classification** problems, the task is to label (or identify) a given data element: identify a character from handwriting, a face from a picture, a disease from its symptoms, etc.
  - **Prediction** guesses the (future) value of something based on (past) values of available data: traffic conditions at a junction, sales numbers for the next year, weather patterns in a city, ...

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**
- We can consider two kinds of problems that are suitable for bottom-up computing:
  - In **Classification** problems, the task is to label (or identify) a given data element: identify a character from handwriting, a face from a picture, a disease from its symptoms, etc.
  - **Prediction** guesses the (future) value of something based on (past) values of available data: traffic conditions at a junction, sales numbers for the next year, weather patterns in a city, ...
- We saw some simple examples of classification where we tried to identify a student from his/her marks and profile, or a shopper from his/her buying behaviour.

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**
- We can consider two kinds of problems that are suitable for bottom-up computing:
  - In **Classification** problems, the task is to label (or identify) a given data element: identify a character from handwriting, a face from a picture, a disease from its symptoms, etc.
  - **Prediction** guesses the (future) value of something based on (past) values of available data: traffic conditions at a junction, sales numbers for the next year, weather patterns in a city, ...
- We saw some simple examples of classification where we tried to identify a student from his/her marks and profile, or a shopper from his/her buying behaviour.
- We also tried to predict the total marks given the marks in two subjects

What is bottom-up computing?

- In all the problems that we have tackled so far, the "algorithm" or pseudo-code is fixed and remains the same for any data given to it. In other words, it is top-down.
  - What if we are allowed to change the algorithm or code?
  - Starting from a simple template, we construct the code as the data keeps coming in
  - This is called **bottom-up computing**
- We can consider two kinds of problems that are suitable for bottom-up computing:
  - In **Classification** problems, the task is to label (or identify) a given data element: identify a character from handwriting, a face from a picture, a disease from its symptoms, etc.
  - **Prediction** guesses the (future) value of something based on (past) values of available data: traffic conditions at a junction, sales numbers for the next year, weather patterns in a city, ...
- We saw some simple examples of classification where we tried to identify a student from his/her marks and profile, or a shopper from his/her buying behaviour.
- We also tried to predict the total marks given the marks in two subjects
- In this lecture, we will look at one more example of prediction

Example: Predict train delays

- Trains are often delayed. There could be many reasons for the delay, but often the delay is due to the trains travelling just prior to this train being delayed

Example: Predict train delays

- Trains are often delayed. There could be many reasons for the delay, but often the delay is due to the trains travelling just prior to this train being delayed
  - If two trains share a track, then the previous train has to clear the track before this train can occupy it
  - If the previous train stops at a station platform that this train also has to stop at, then the previous train has to leave the station before this train can enter it
  - Since shared tracks and shared platforms are closely related, we will only look at shared stations in this example

Example: Predict train delays

- Trains are often delayed. There could be many reasons for the delay, but often the delay is due to the trains travelling just prior to this train being delayed
  - If two trains share a track, then the previous train has to clear the track before this train can occupy it
  - If the previous train stops at a station platform that this train also has to stop at, then the previous train has to leave the station before this train can enter it
  - Since shared tracks and shared platforms are closely related, we will only look at shared stations in this example
- Consider three trains T1, T2 and T, where T1 and T2 arrive at a particular fixed station S just prior to the arrival of train T at S.

Example: Predict train delays

- Trains are often delayed. There could be many reasons for the delay, but often the delay is due to the trains travelling just prior to this train being delayed
  - If two trains share a track, then the previous train has to clear the track before this train can occupy it
  - If the previous train stops at a station platform that this train also has to stop at, then the previous train has to leave the station before this train can enter it
  - Since shared tracks and shared platforms are closely related, we will only look at shared stations in this example
- Consider three trains T1, T2 and T, where T1 and T2 arrive at a particular fixed station S just prior to the arrival of train T at S.
- We are provided with a table, whose first, second and third column entries are the running delays of T1, T2 and T as recorded at S. Each row represents one day.

Dataset for the train delay example

- We are provided with a table, whose first, second and third column entries are the running delays of T1, T2 and T as recorded at S (in minutes). Each row represents one day in the recent past. The table is shown below.

T1 delay	T2 delay	T delay
20	25	20
10	50	40
50	15	35
40	80	70
25	15	20

- The running delay of T1 and T2 for today are 75 and 40 minutes respectively. Can we predict what will be the delay of T today?

Determining the prediction function

- This is clearly a prediction problem. We need to guess the weights w_1 and w_2 that we have to assign to the delays Δ_1 and Δ_2 of trains T1 and T2, so that the sum $w_1 \times \Delta_1 + w_2 \times \Delta_2$ is as close to Δ, the delay of train T as possible

Determining the prediction function

- This is clearly a prediction problem. We need to guess the weights w_1 and w_2 that we have to assign to the delays Δ_1 and Δ_2 of trains T1 and T2, so that the sum $w_1 \times \Delta_1 + w_2 \times \Delta_2$ is as close to Δ, the delay of train T as possible
- We can start by assuming that both have equal weight in the delay - i.e. that $w_1 = w_2 = 0.5$

Determining the prediction function

- This is clearly a prediction problem. We need to guess the weights w_1 and w_2 that we have to assign to the delays Δ_1 and Δ_2 of trains T1 and T2, so that the sum $w_1 \times \Delta_1 + w_2 \times \Delta_2$ is as close to Δ, the delay of train T as possible
- We can start by assuming that both have equal weight in the delay - i.e. that $w_1 = w_2 = 0.5$
- Let us calculate the error we get for the first row of the table. Error (fifth column) is obtained by taking the difference of the calculated delay (fourth column) from the actual delay (third column) and squaring it.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25

Determining the prediction function

- This is clearly a prediction problem. We need to guess the weights w_1 and w_2 that we have to assign to the delays Δ_1 and Δ_2 of trains T1 and T2, so that the sum $w_1 \times \Delta_1 + w_2 \times \Delta_2$ is as close to Δ, the delay of train T as possible
- We can start by assuming that both have equal weight in the delay - i.e. that $w_1 = w_2 = 0.5$
- Let us calculate the error we get for the first row of the table. Error (fifth column) is obtained by taking the difference of the calculated delay (fourth column) from the actual delay (third column) and squaring it.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25

- Calculated delay is $0.5 \times 20 + 0.5 \times 25 = 22.5$ and the error is $(22.5 - 20)^2 = 6.25$

Determining the prediction function

- This is clearly a prediction problem. We need to guess the weights w_1 and w_2 that we have to assign to the delays Δ_1 and Δ_2 of trains T1 and T2, so that the sum $w_1 \times \Delta_1 + w_2 \times \Delta_2$ is as close to Δ, the delay of train T as possible
- We can start by assuming that both have equal weight in the delay - i.e. that $w_1 = w_2 = 0.5$
- Let us calculate the error we get for the first row of the table. Error (fifth column) is obtained by taking the difference of the calculated delay (fourth column) from the actual delay (third column) and squaring it.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25

- Calculated delay is $0.5 \times 20 + 0.5 \times 25 = 22.5$ and the error is $(22.5 - 20)^2 = 6.25$
- Since the calculated delay is higher, the function over-estimates the value. But since the error is small, we don't change the weights as yet.

Adjusting the weights of the prediction function

- Let us try the second row now

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25
10	50	40	30	100

Adjusting the weights of the prediction function

- Let us try the second row now

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25
10	50	40	30	100

- Calculated delay is $0.5 \times 10 + 0.5 \times 50 = 30$ and the error is $(30 - 40)^2 = 100$

Adjusting the weights of the prediction function

- Let us try the second row now

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25
10	50	40	30	100

- Calculated delay is $0.5 \times 10 + 0.5 \times 50 = 30$ and the error is $(30 - 40)^2 = 100$
- The calculated delay is lower than the actual, so it is an under-estimate. Aren't we glad that we did not adjust the weights for a higher value in the first row !

Adjusting the weights of the prediction function

- Let us try the second row now

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	22.5	6.25
10	50	40	30	100

- Calculated delay is $0.5 \times 10 + 0.5 \times 50 = 30$ and the error is $(30 - 40)^2 = 100$
- The calculated delay is lower than the actual, so it is an under-estimate. Aren't we glad that we did not adjust the weights for a higher value in the first row !
- Since the error is high, we should adjust the weights upwards. Since the first co-efficient (10) is low, we will not change that weight. To get an increase of 10 from the T2's delay, we can add $10/50 = 0.2$ to w_2 to get $w_2 = 0.7$.

Adjusting the weights of the prediction function

- Let us try the second row now, but with weights $w_1 = 0.5$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	27.5	56.25
10	50	40	40	0

Adjusting the weights of the prediction function

- Let us try the second row now, but with weights $w_1 = 0.5$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	27.5	56.25
10	50	40	40	0

- The second row error is zero (because we made it that way !).

Adjusting the weights of the prediction function

- Let us try the second row now, but with weights $w_1 = 0.5$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	27.5	56.25
10	50	40	40	0

- The second row error is zero (because we made it that way !).
- However, first row error increases: Calculated delay is $0.5 \times 20 + 0.7 \times 25 = 27.5$ and the error is $(27.5 - 20)^2 = 56.25$

Adjusting the weights of the prediction function

- Let us try the second row now, but with weights $w_1 = 0.5$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	27.5	56.25
10	50	40	40	0

- The second row error is zero (because we made it that way !).
- However, first row error increases: Calculated delay is $0.5 \times 20 + 0.7 \times 25 = 27.5$ and the error is $(27.5 - 20)^2 = 56.25$
- The error is high. So we can try to remedy this by reducing w_1. To get a decrease of 7.5 from the T1's delay, we can subtract $7.5/20 = 0.375$ from w_1 to get 0.125. Table is:

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	20	0
10	50	40	36.25	14.06

Adjusting the weights of the prediction function

- Let us try the second row now, but with weights $w_1 = 0.5$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	27.5	56.25
10	50	40	40	0

- The second row error is zero (because we made it that way !).
- However, first row error increases: Calculated delay is $0.5 \times 20 + 0.7 \times 25 = 27.5$ and the error is $(27.5 - 20)^2 = 56.25$
- The error is high. So we can try to remedy this by reducing w_1. To get a decrease of 7.5 from the T1's delay, we can subtract $7.5/20 = 0.375$ from w_1 to get 0.125. Table is:

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	20	0
10	50	40	36.25	14.06

- The second row has an error now, but is small enough.

Adjusting the weights of the prediction function

- Let us try the third row now, with weights $w_1 = 0.125$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	20	0
10	50	40	36.25	14.06
50	15	35	16.75	333.1

Adjusting the weights of the prediction function

- Let us try the third row now, with weights $w_1 = 0.125$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	20	0
10	50	40	36.25	14.06
50	15	35	16.75	333.1

- Calculated delay is $0.125 \times 50 + 0.7 \times 15 = 16.75$ and the error is $(16.75 - 35)^2 = 333.1$

Adjusting the weights of the prediction function

- Let us try the third row now, with weights $w_1 = 0.125$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	20	0
10	50	40	36.25	14.06
50	15	35	16.75	333.1

- Calculated delay is $0.125 \times 50 + 0.7 \times 15 = 16.75$ and the error is $(16.75 - 35)^2 = 333.1$
- Error for the third row is way too high ! The adjustment to w_1 was an overkill. We need to adjust it upwards by approximately $18.25/50 = 0.36$, but this gets us back close to the original level of 0.5. Let us set $w_1 = 0.4$ and see what it gets us. The resulting table is:

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25

Adjusting the weights of the prediction function

- Let us try the third row now, with weights $w_1 = 0.125$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	20	0
10	50	40	36.25	14.06
50	15	35	16.75	333.1

- Calculated delay is $0.125 \times 50 + 0.7 \times 15 = 16.75$ and the error is $(16.75 - 35)^2 = 333.1$
- Error for the third row is way too high ! The adjustment to w_1 was an overkill. We need to adjust it upwards by approximately $18.25/50 = 0.36$, but this gets us back close to the original level of 0.5. Let us set $w_1 = 0.4$ and see what it gets us. The resulting table is:

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25

- First row error is up, but the *average* error $= (30.25 + 1 + 20.25)/3 \approx 17$ is low.

Adjusting the weights of the prediction function

- Let us try the fourth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4

Adjusting the weights of the prediction function

- Let us try the fourth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4

- Calculated delay is $0.4 \times 40 + 0.7 \times 80 = 72$ and the error is $(72 - 70)^2 = 4$, very low !

Adjusting the weights of the prediction function

- Let us try the fourth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4

- Calculated delay is $0.4 \times 40 + 0.7 \times 80 = 72$ and the error is $(72 - 70)^2 = 4$, very low !
- Average error has reduced further to $(30.25 + 1 + 20.25 + 4)/4 \approx 14$!

Adjusting the weights of the prediction function

- Let us try the fourth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4

- Calculated delay is $0.4 \times 40 + 0.7 \times 80 = 72$ and the error is $(72 - 70)^2 = 4$, very low !
- Average error has reduced further to $(30.25 + 1 + 20.25 + 4)/4 \approx 14$!
- So, let us try the fifth row.

Adjusting the weights of the prediction function

- Let us try the fifth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4
25	15	20	20.5	0.25

Adjusting the weights of the prediction function

- Let us try the fifth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4
25	15	20	20.5	0.25

- Calculated delay is $0.4 \times 25 + 0.7 \times 15 = 20.5$ and the error is $(20.5 - 20)^2 = 0.25$, very low !

Adjusting the weights of the prediction function

- Let us try the fifth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4
25	15	20	20.5	0.25

- Calculated delay is $0.4 \times 25 + 0.7 \times 15 = 20.5$ and the error is $(20.5 - 20)^2 = 0.25$, very low !
- Average error has reduced further to $(30.25 + 1 + 20.25 + 4 + 0.25)/5 \approx 11$!

Adjusting the weights of the prediction function

- Let us try the fifth row now, with weights $w_1 = 0.4$ and $w_2 = 0.7$.

T1 delay	T2 delay	T delay	Calc delay	Error
20	25	20	25.5	30.25
10	50	40	39	1
50	15	35	30.5	20.25
40	80	70	72	4
25	15	20	20.5	0.25

- Calculated delay is $0.4 \times 25 + 0.7 \times 15 = 20.5$ and the error is $(20.5 - 20)^2 = 0.25$, very low !
- Average error has reduced further to $(30.25 + 1 + 20.25 + 4 + 0.25)/5 \approx 11$!
- We are done with all the data values now, and the prediction function $0.4 \times \Delta_1 + 0.7 \times \Delta_2$ seems to be a reasonably good fit to the given data.

Making a prediction using the prediction function

- Now that we know the prediction function: $0.4 \times \Delta_1 + 0.7 \times \Delta_2$, let us try applying it to the given delays for today $\Delta_1 = 75$ and $\Delta_2 = 40$ (last row of the table):

T1 delay	T2 delay	T delay	Calc delay
20	25	20	25.5
10	50	40	39
50	15	35	30.5
40	80	70	72
25	15	20	20.5
75	40	??	58

Making a prediction using the prediction function

- Now that we know the prediction function: $0.4 \times \Delta_1 + 0.7 \times \Delta_2$, let us try applying it to the given delays for today $\Delta_1 = 75$ and $\Delta_2 = 40$ (last row of the table):

T1 delay	T2 delay	T delay	Calc delay
20	25	20	25.5
10	50	40	39
50	15	35	30.5
40	80	70	72
25	15	20	20.5
75	40	??	58

- Calculated delay is $0.4 \times 75 + 0.7 \times 40 = 58$

Making a prediction using the prediction function

- Now that we know the prediction function: $0.4 \times \Delta_1 + 0.7 \times \Delta_2$, let us try applying it to the given delays for today $\Delta_1 = 75$ and $\Delta_2 = 40$ (last row of the table):

T1 delay	T2 delay	T delay	Calc delay
20	25	20	25.5
10	50	40	39
50	15	35	30.5
40	80	70	72
25	15	20	20.5
75	40	??	58

- Calculated delay is $0.4 \times 75 + 0.7 \times 40 = 58$
- So our prediction is that train T is going to be 58 minutes late today.

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:
  - At each node, some property of the data element is checked: based on the value, one of the branches from that node is chosen

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:
  - At each node, some property of the data element is checked: based on the value, one of the branches from that node is chosen
  - The leaf nodes contain the classification label to be assigned to the element

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:
  - At each node, some property of the data element is checked: based on the value, one of the branches from that node is chosen
  - The leaf nodes contain the classification label to be assigned to the element
- Prediction tries to find a (linear) numerical function that connects the value to be predicted to the numerical values of the data elements that are available

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:
  - At each node, some property of the data element is checked: based on the value, one of the branches from that node is chosen
  - The leaf nodes contain the classification label to be assigned to the element
- Prediction tries to find a (linear) numerical function that connects the value to be predicted to the numerical values of the data elements that are available
- Classification and Prediction can be combined:

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:
  - At each node, some property of the data element is checked: based on the value, one of the branches from that node is chosen
  - The leaf nodes contain the classification label to be assigned to the element
- Prediction tries to find a (linear) numerical function that connects the value to be predicted to the numerical values of the data elements that are available
- Classification and Prediction can be combined:
  - The branch decision at a node can use prediction - first construct a numerical function and then based on different cut-off values of the result of this function, a different branch is taken.

Summary

- In bottom-up computing, the code is constructed from (a sample of) the data elements
- In classification, a tree like structure (decision tree) is created:
  - At each node, some property of the data element is checked: based on the value, one of the branches from that node is chosen
  - The leaf nodes contain the classification label to be assigned to the element
- Prediction tries to find a (linear) numerical function that connects the value to be predicted to the numerical values of the data elements that are available
- Classification and Prediction can be combined:
  - The branch decision at a node can use prediction - first construct a numerical function and then based on different cut-off values of the result of this function, a different branch is taken.
  - A decision tree is first made. Instead of the leaves containing labels, a prediction function is constructed for each leaf node. For a given data element, we first use the decision tree to find the right leaf node, and then apply its prediction function to find the desired value