CS 3265 and CS 5265
Vanderbilt University

Lecture on Data Mining

- Difference between FD mining and association rule mining
- Unsupervised learning and supervised learning
- Decision trees

A comment on difference between association rules
(https://my.vanderbilt.edu/cs265/data-mining-tasks-1/)
and functional dependencies
(https://my.vanderbilt.edu/cs265/data-mining-programming-assignment/)

Association rules, as presented

{onions, potatoes} —> {burger}, (e.g., with support 0.15)
where 'onions', 'potatoes', 'burger' are values of binary-valued attributes (aka variables) that also have values of ~onions, ~potatoes, ~burger

Attribute Burger has possible values burger or ~burger (or present or absent)
Attribute Onion has possible values onion or ~onion (or present or absent)
Attribute Potato has possible values potato or ~potato (or present or absent)

More generally, we can have many-valued attributes, such as
Color (with values red, green, blue, yellow, …),
Size (with values xxl, xl, l, m, s, xs)
Shape (with values triangle, square, circle, squashed-ellipse, …)

Color=red, Shape=circle → Size=xl (or just red, circle → xl)
Color=blue, Shape=circle → Size = s
Size=m → Shape=Shape=triangle
…

Functional Dependencies

Example with many-valued attributes, such as

Color (with values red, green, blue, yellow)   4 values
Size (with values xxl, xl, l, m, s, xs)   6 values
Shape (with values triangle, square, circle)   3 values

Color, Shape → Size   (e.g., with support 0.96)

Color=red, Shape=circle → Size=xl
Color=blue, Shape=circle → Size=s
Color=green, Shape=circle → Size=s
…
Color=green, Shape=triangle → Size=m

Color=x, Shape=y → Size=z
for all (4*3) x, y pairs,
there exists a z (with HIGH support)

One could think of FD discovery as including association rule discovery as a subroutine, but the uses of association rule discovery (requiring only very modest Support) and FD discovery (requiring high support) are substantially different.

Association rules are a form of unsupervised data mining (or machine learning)

If a data set is described by attributes A, B, C, … then the result of association rule learning can (potentially help predict the value of any attribute X, from the values of one or more of the other attributes (i.e., pattern completion)

So, given

blue ? circle

↓

Knowledge Base

↓

blue small circle

…

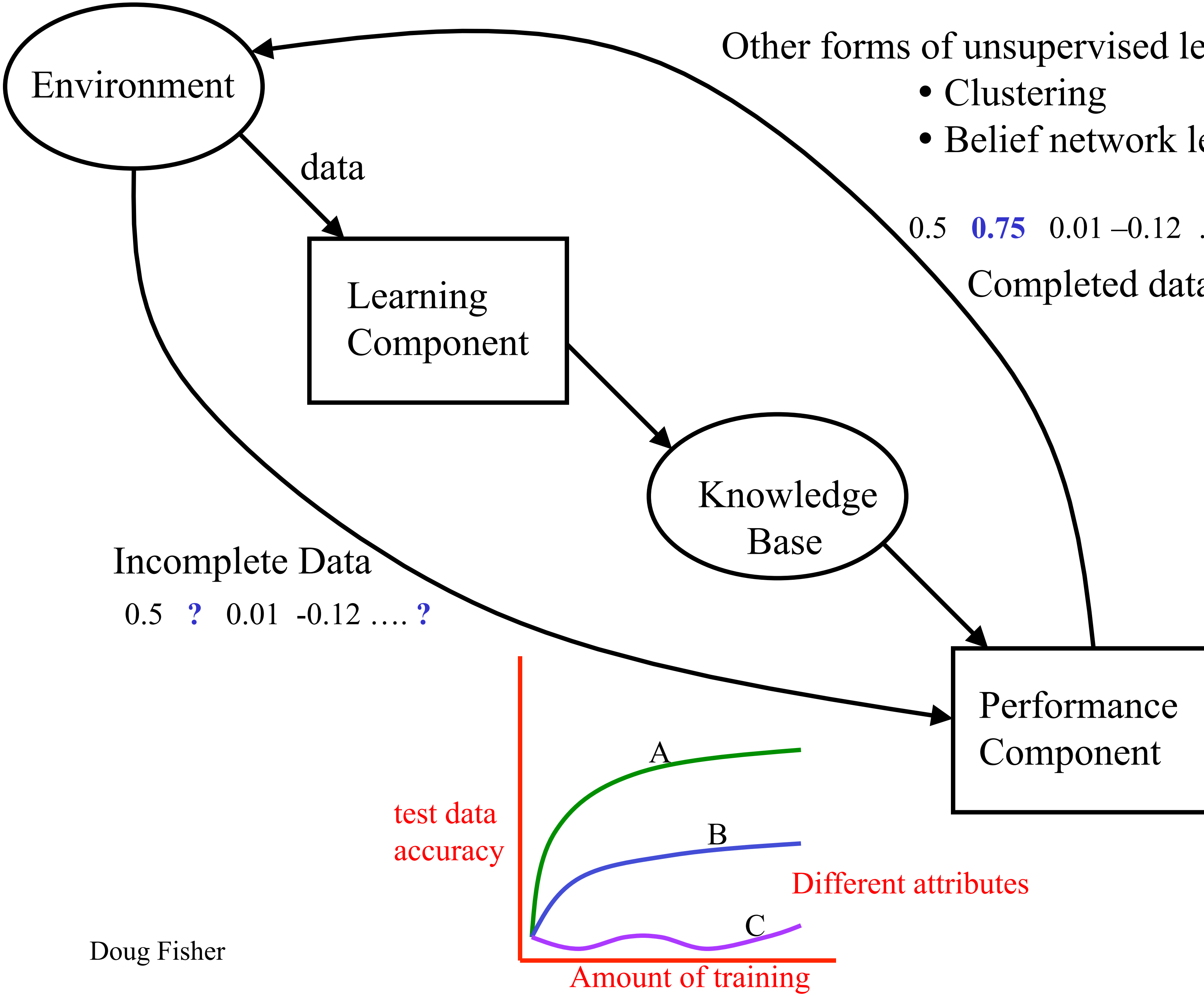
Color=blue, Shape=circle → Size=s

…

Size=m → Shape=Shape=triangle

…

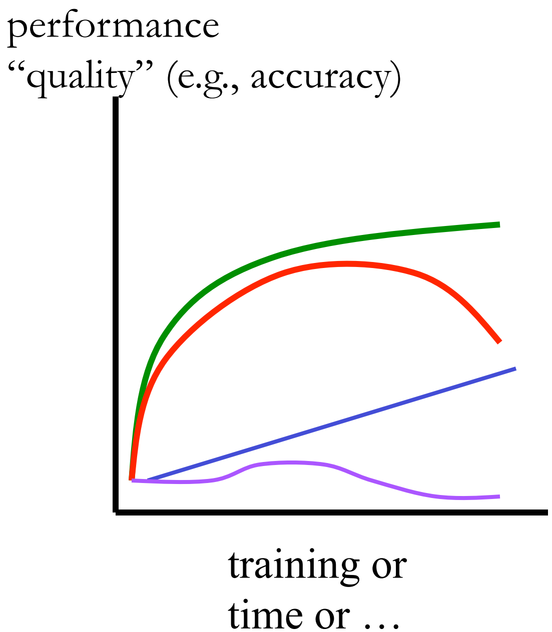
Other forms of unsupervised learning:
- Clustering
- Belief network learning

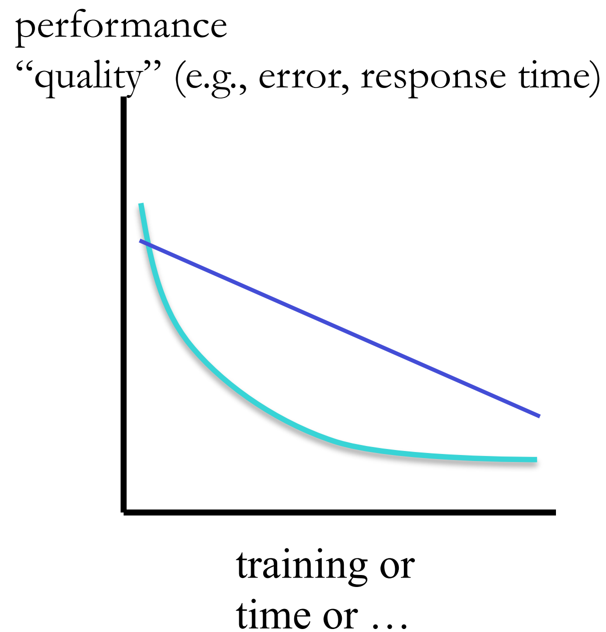


Two perspectives of Machine Learning:

Machine Learning for advanced *data analysis*

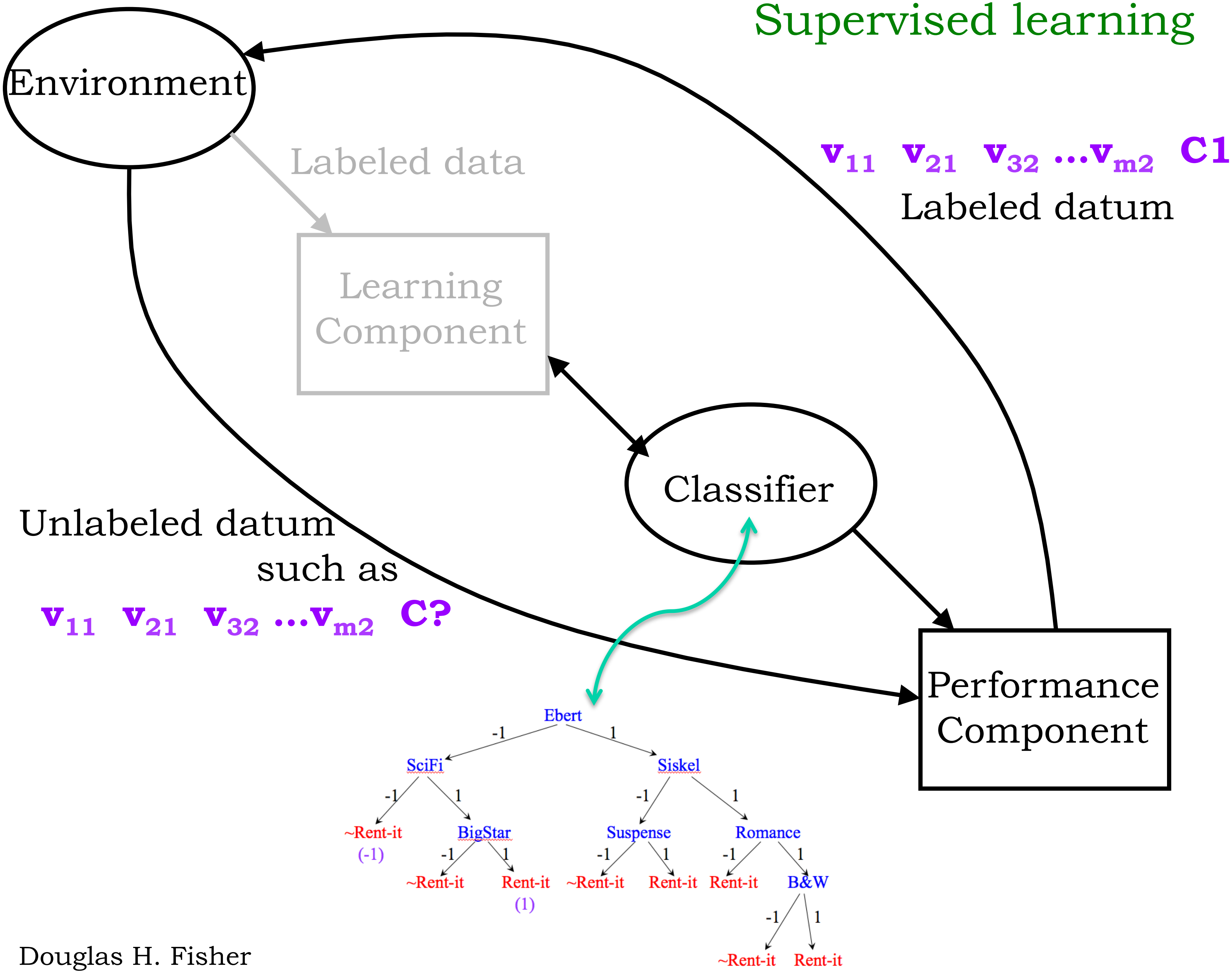
Machine Learning for robust artificial *agents*





pessimism (be cautious) and optimism (jump to conclusions)

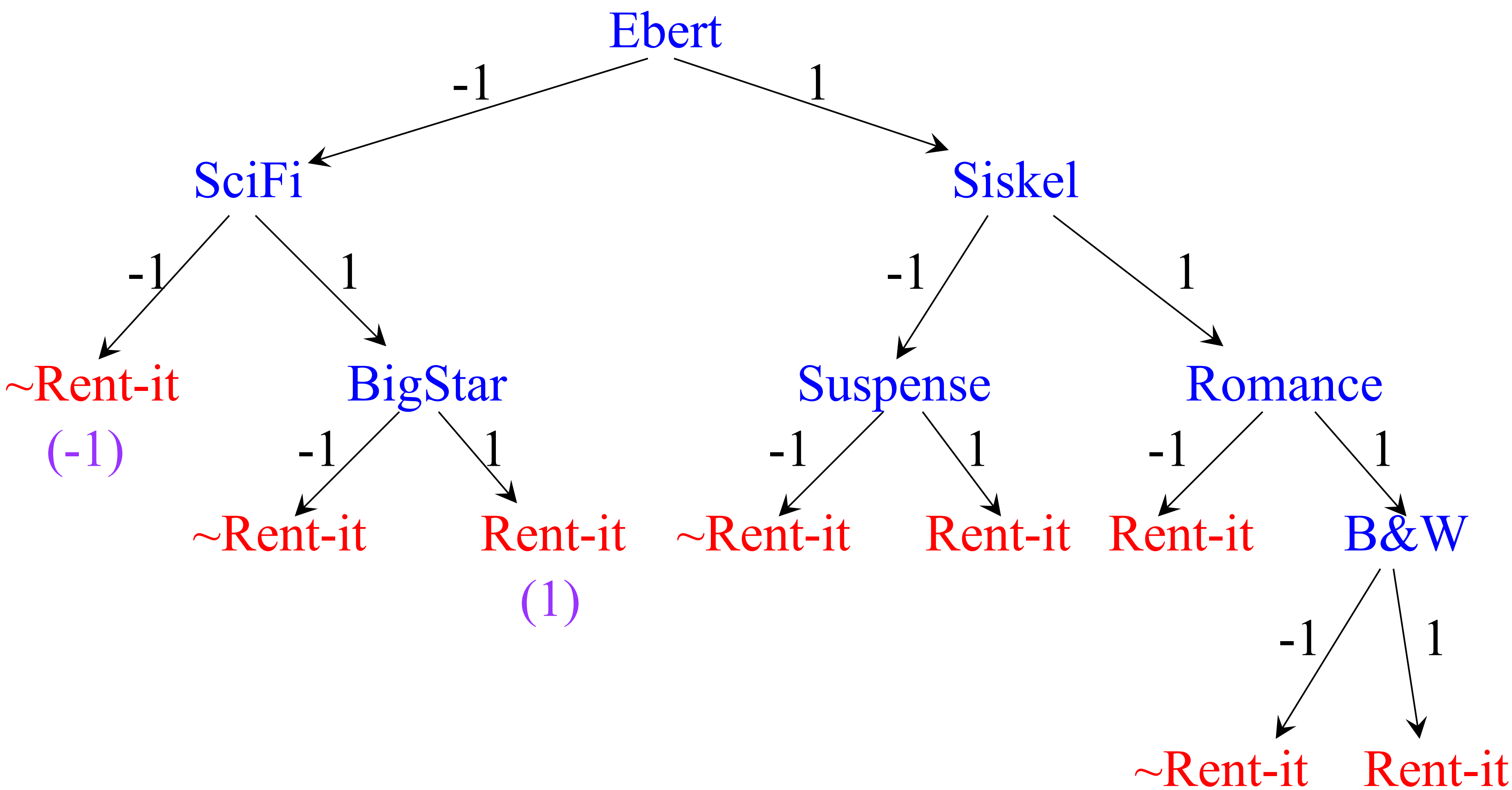
Supervised learning



Douglas H. Fisher

Decision tree classifiers

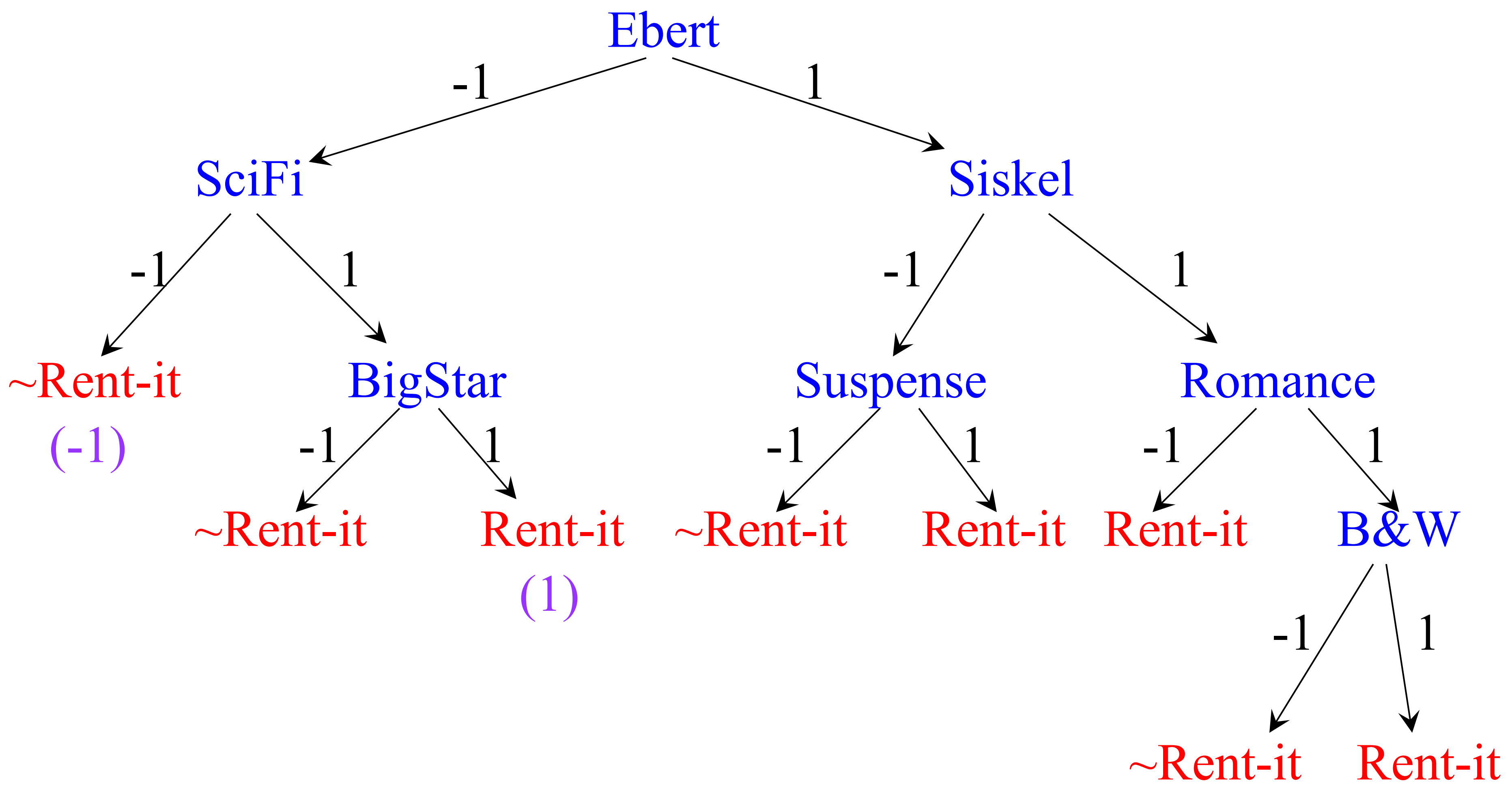
Each internal node represents a test of a variable, and each leaf represents a decision based on the conditions (variable values) along the path to that leaf.



Douglas H. Fisher

Decision tree classifiers

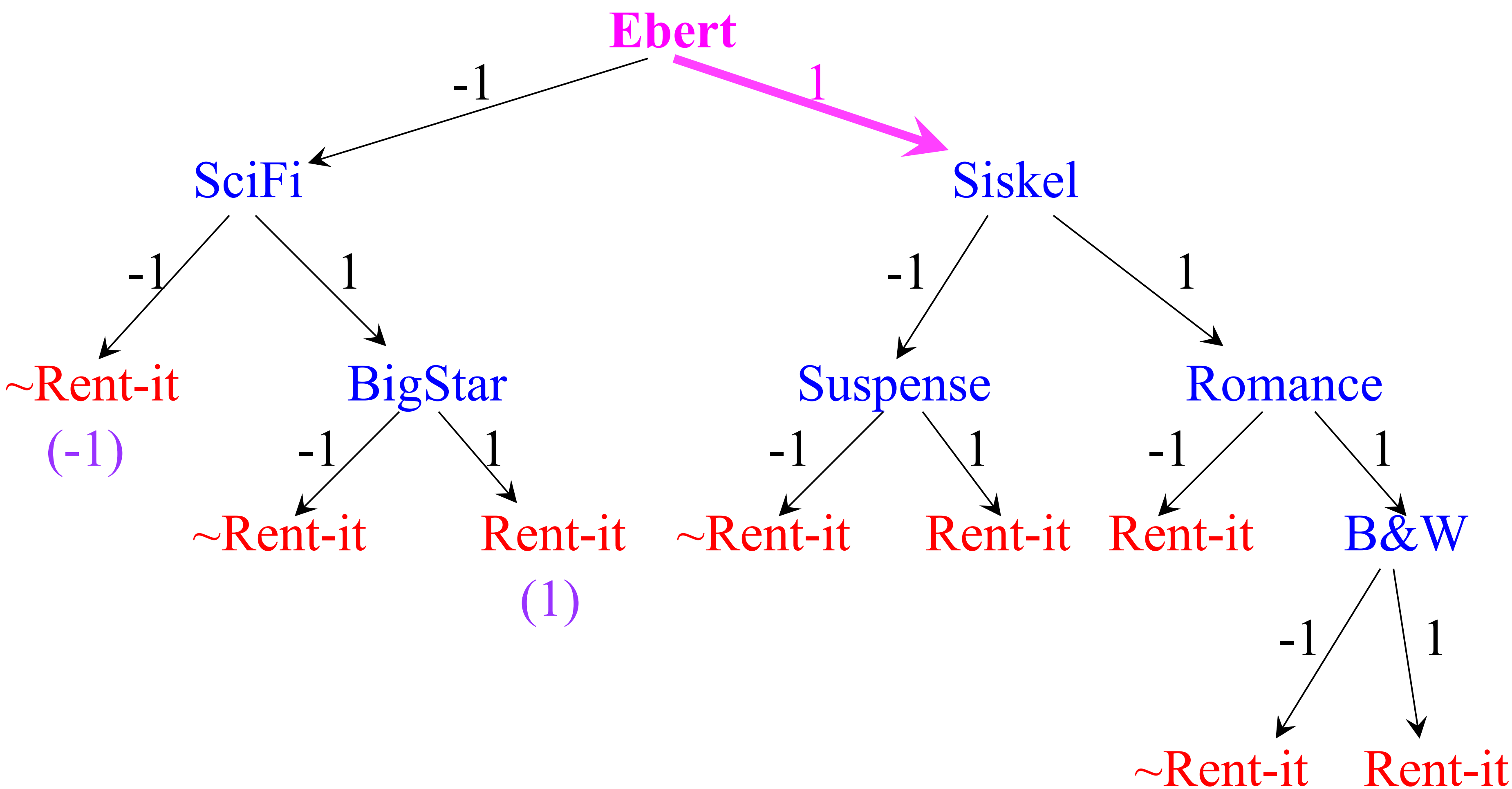
[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, Rent-it???]



Douglas H. Fisher

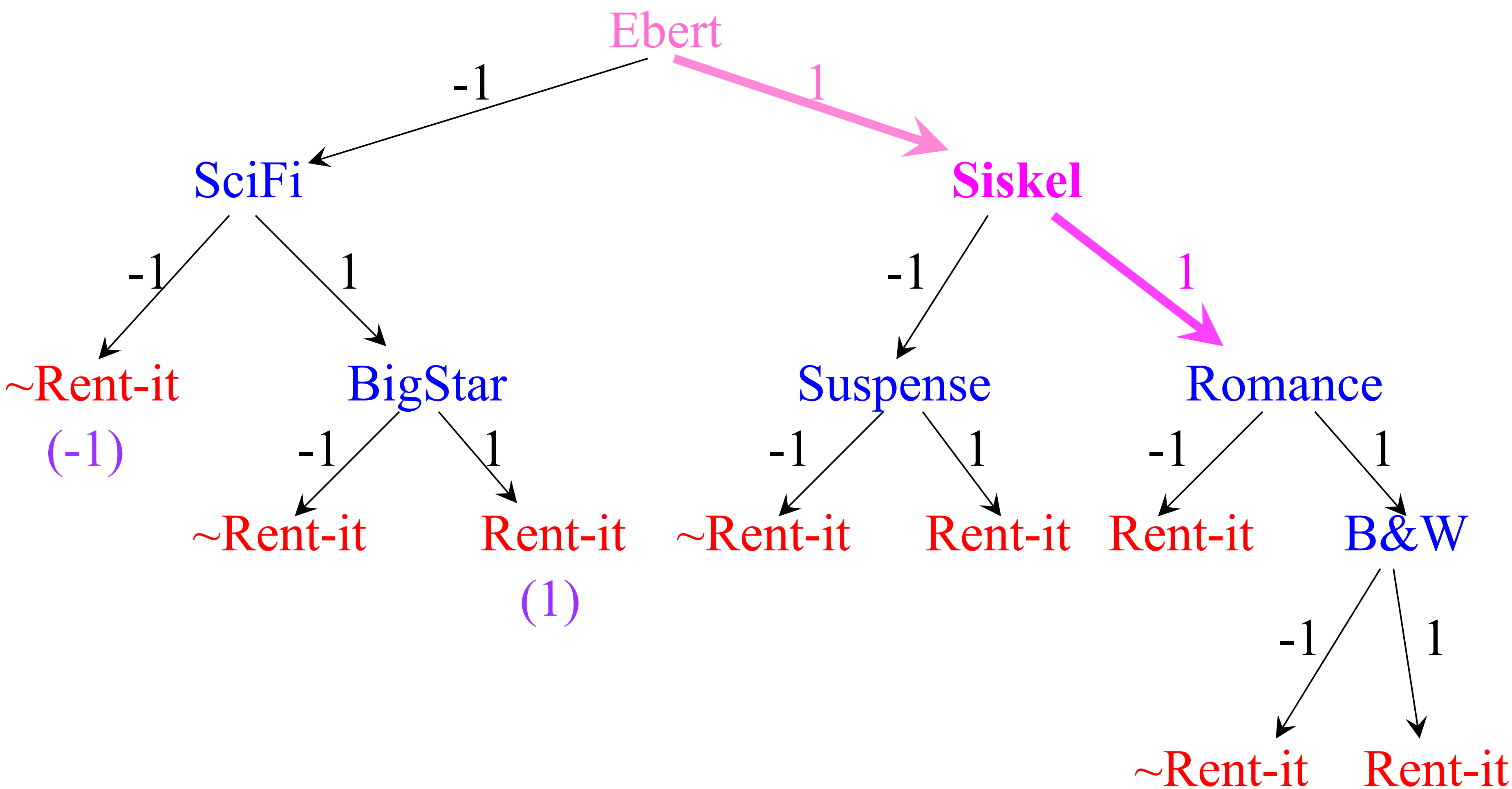
Decision tree classifiers

[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, Rent-it???]



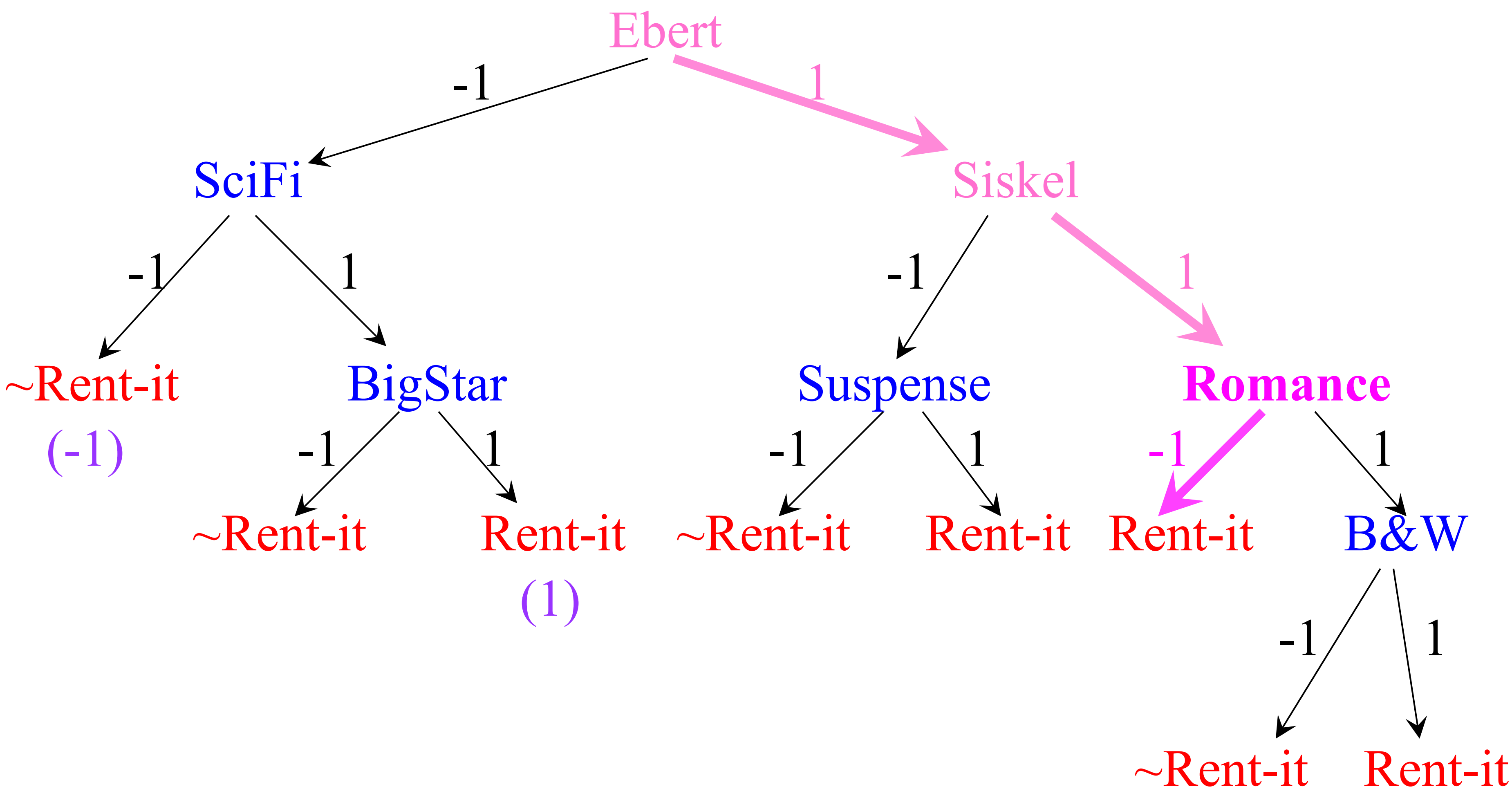
Decision tree classifiers

[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, Rent-it???]



Decision tree classifiers

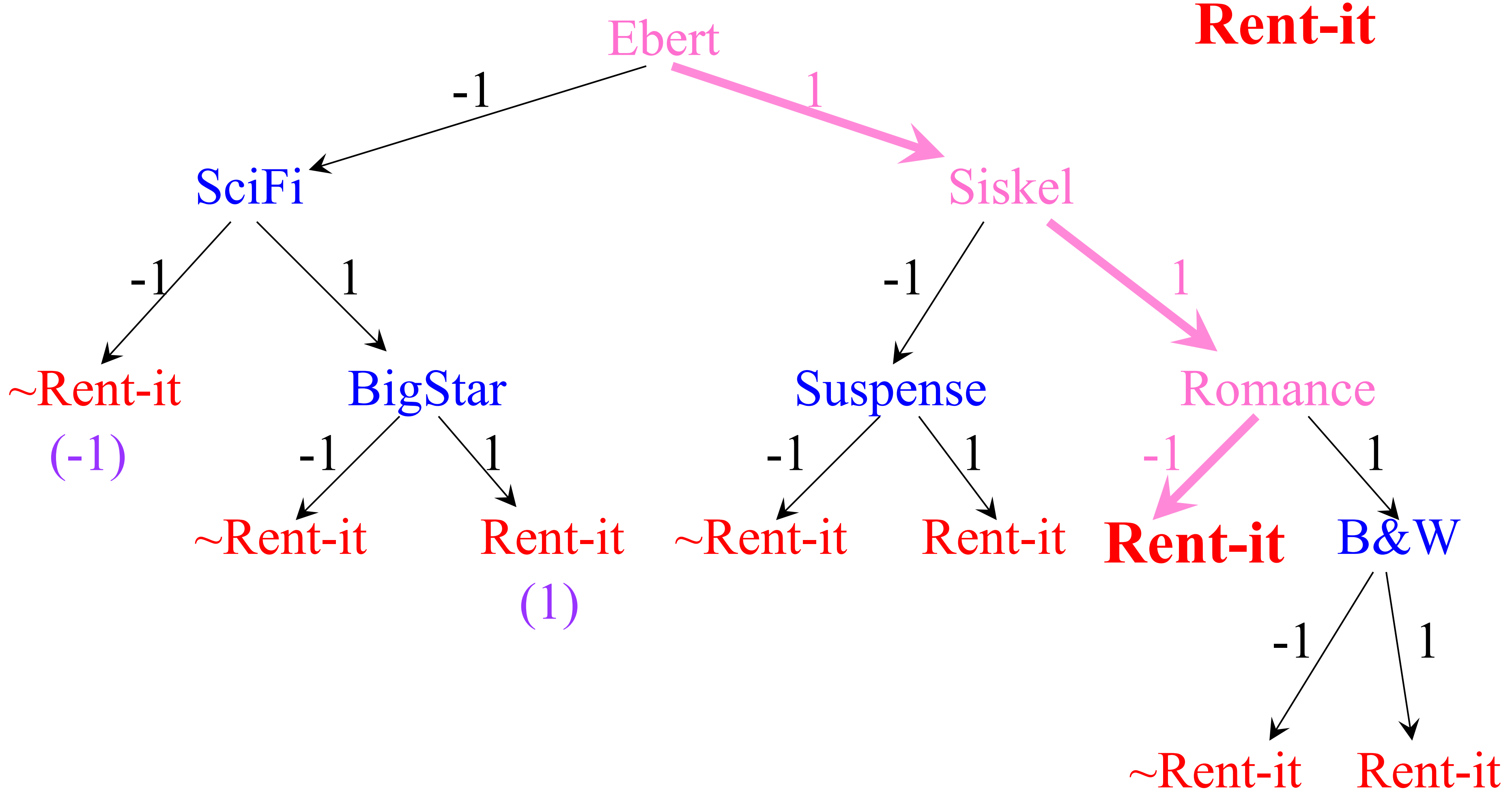
[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, Rent-it???]



Decision tree classifiers

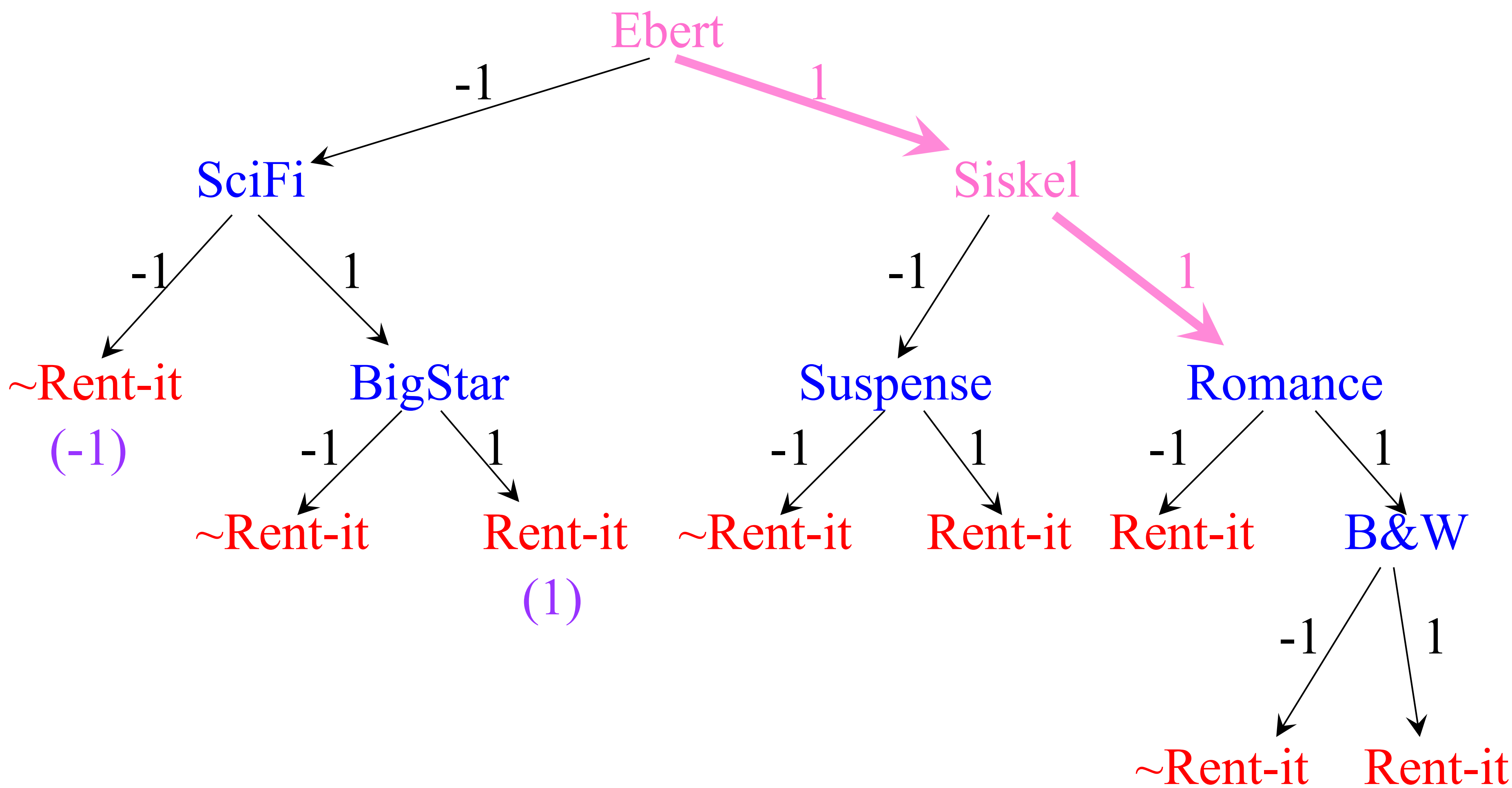
[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, ~~Rent-it???~~]

Rent-it



Consider a completely new test datum, with a different value for Romance (and Suspense); I have also shown the value for B&W

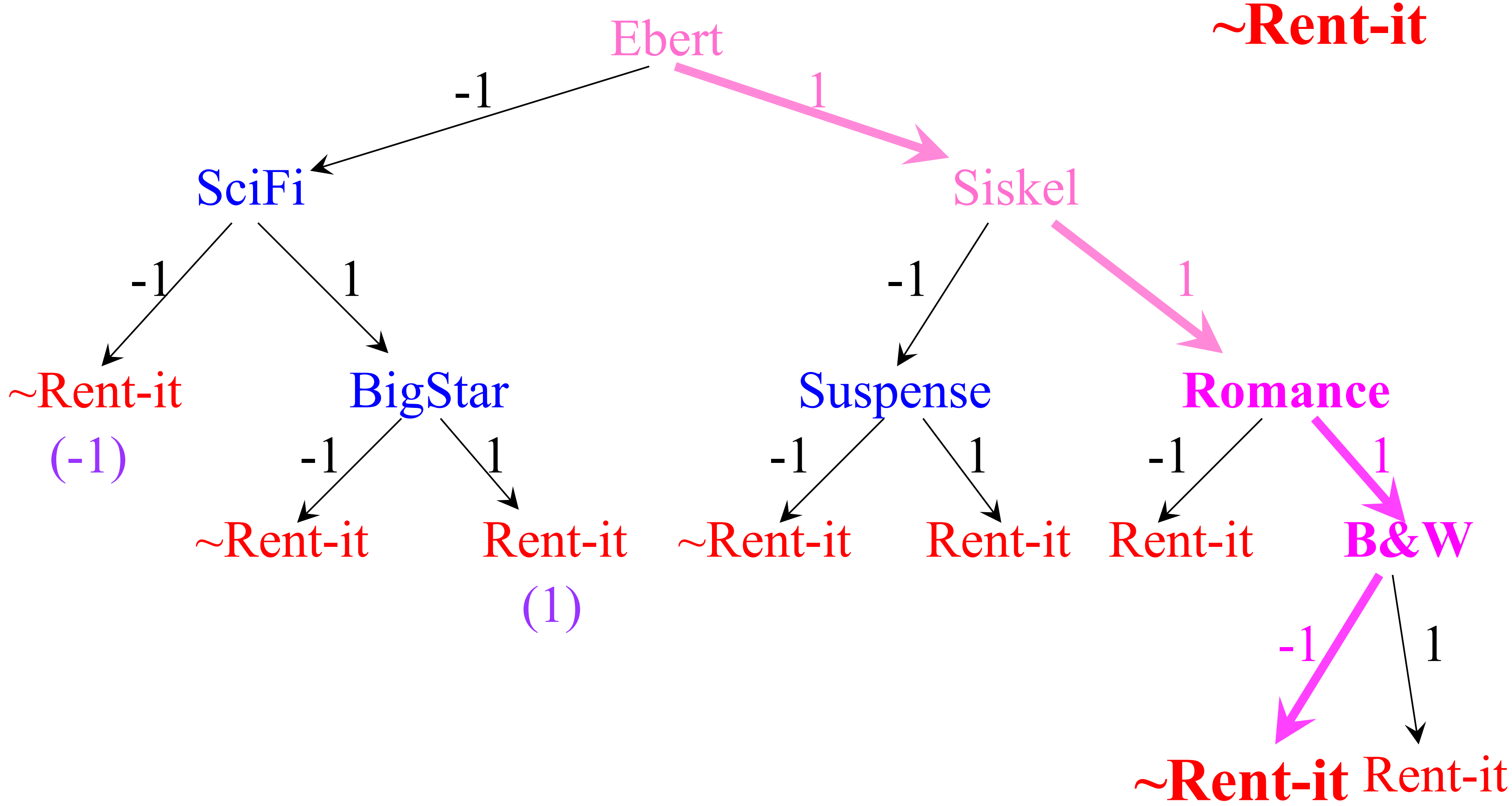
[SciFi = -1, Suspense = -1, Romance = ~~-1~~ **1**, Ebert = 1, Siskel = 1, B&W = -1, …, Rent-it???]



The values for Romance and B&W of this new datum would lead to a different classification than the previous datum

[SciFi = -1, Suspense = -1, Romance = ~~-1~~ 1, Ebert = 1, Siskel = 1, B&W = -1, …, ~~Rent-it???~~]

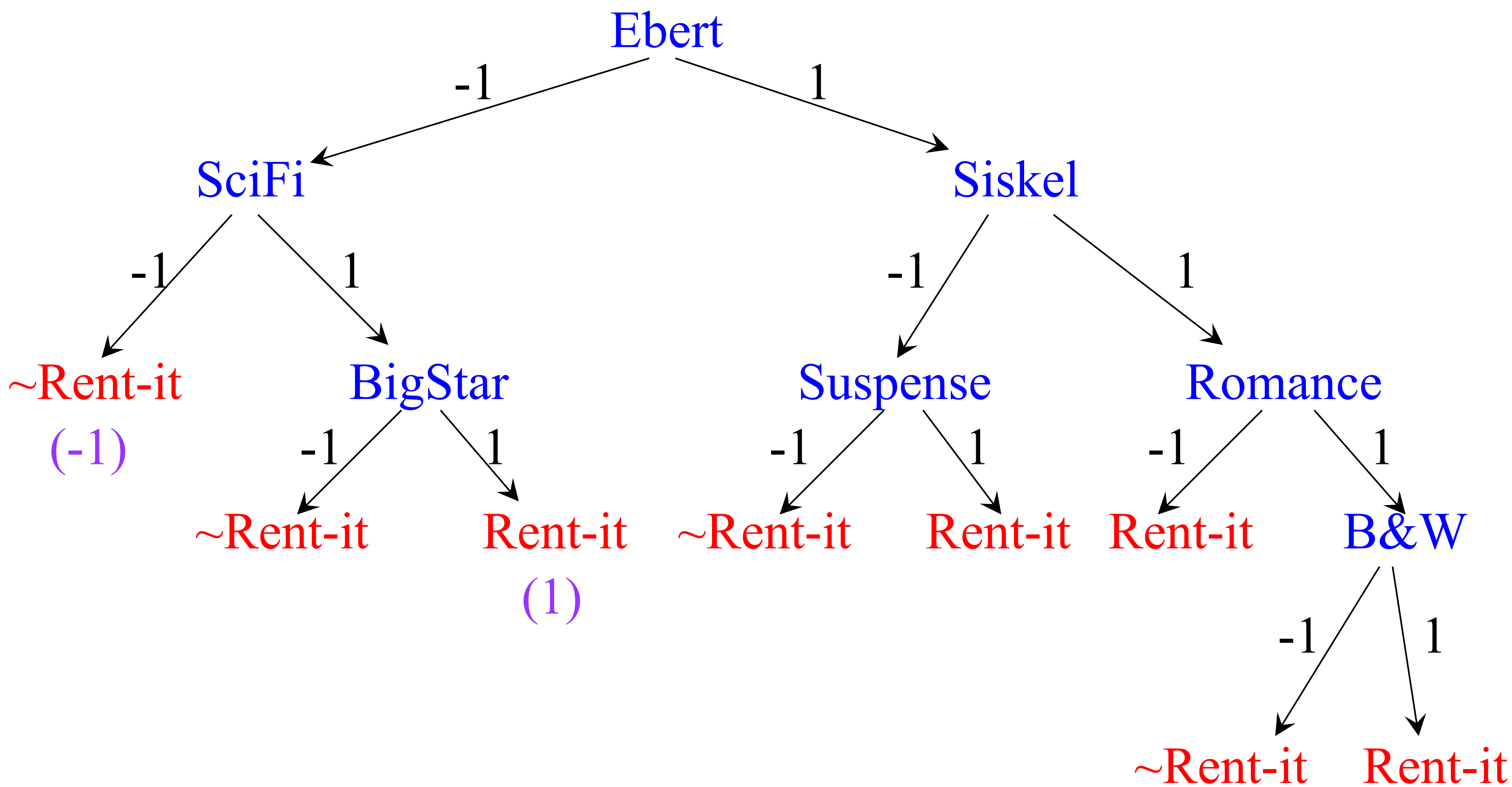
~Rent-it



Douglas H. Fisher

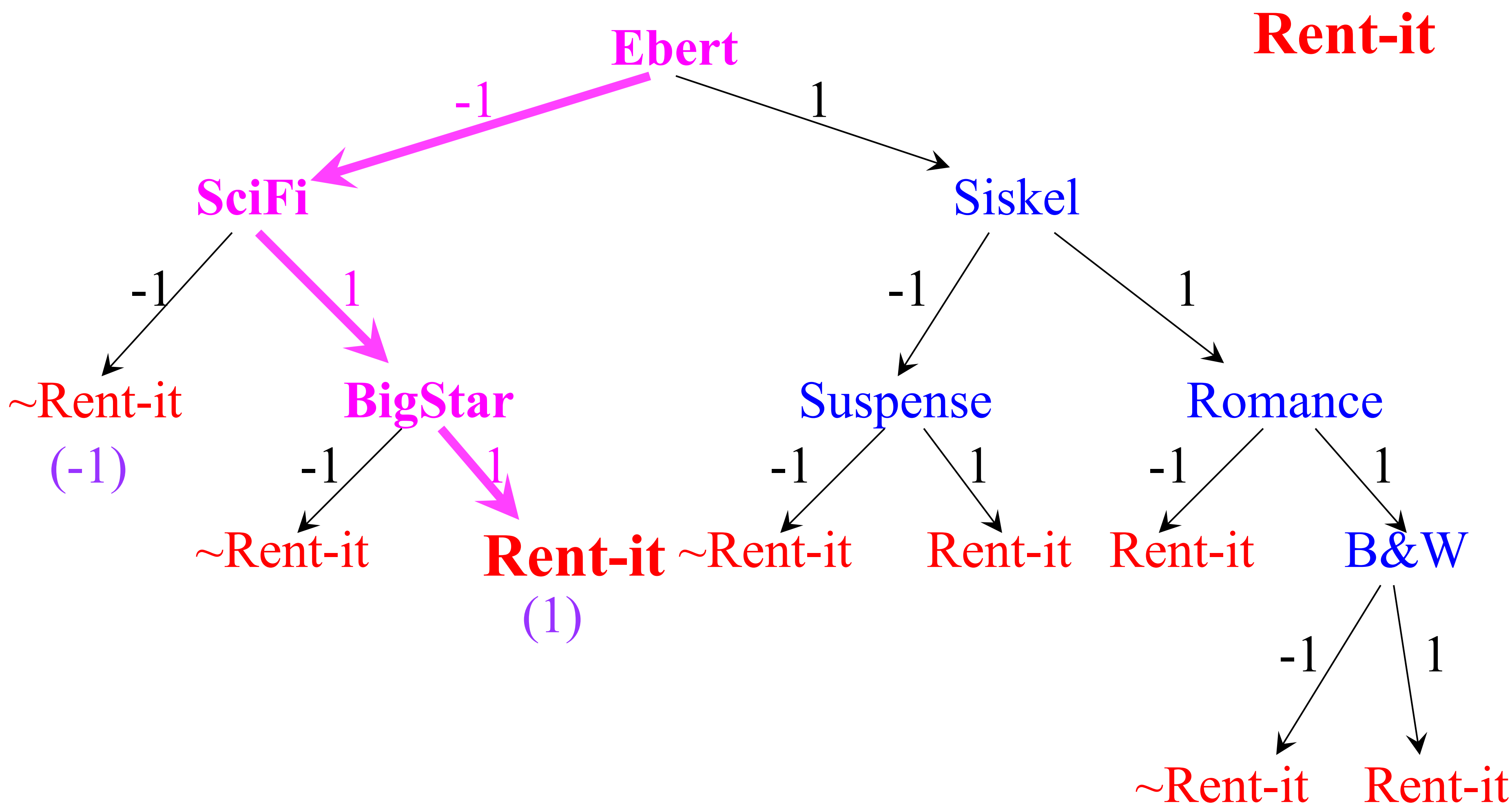
What decision would be made for the following datum, Rent-it or ~Rent-it ?

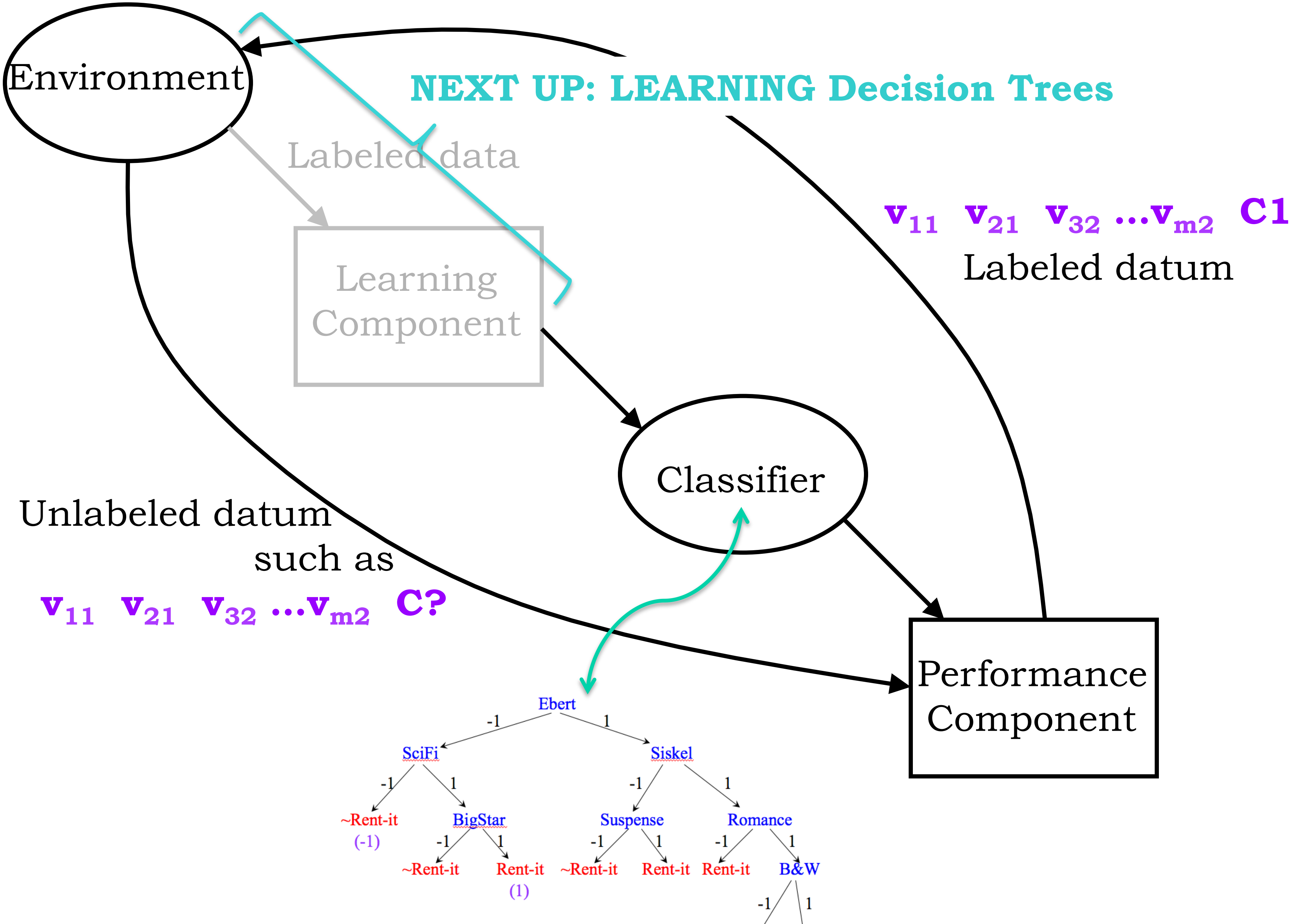
[SciFi = 1, Suspense = 1, Romance = -1, Ebert = -1, Siskel = 1, BigStar = 1, …, Rent-it???]



Douglas H. Fisher

[SciFi = 1, Suspense = 1, Romance = -1, Ebert = -1, Siskel = 1, BigStar = 1, …, ~~Rent-it???~~]




NEXT UP: LEARNING Decision Trees
Environment
Labeled data
Learning Component
Classifier
Performance Component
v11 v21 v32 …vm2 C1
Labeled datum
Unlabeled datum such as
v11 v21 v32 …vm2 C?
Ebert
-1
1
SciFi
Siskel
-1
1
~Rent-it
(-1)
BigStar
-1
1
~Rent-it
Rent-it
(1)
Suspense
-1
1
~Rent-it
Rent-it
Romance
-1
1
Rent-it
B&W
-1
1
~Rent-it
Rent-it

The standard greedy (hill-climbing) approach
(Top-Down Induction of Decision Trees)

Node TDIDT (Set **Data**,
int (* **TerminateFn**) (Set, Set, Set),
Variable (* **SelectFn**) (Set, Set, Set)) {

IF ((* TerminateFn) (Data)) RETURN ClassNode(Data);

BestVariable = (* SelectFn)(Data);

RETURN (TestNode(BestVariable))

v_1 → TDIDT({d | d in Data and Value(BestAttribute, d) = v_1})

v_2 → TDIDT({d | d in Data and Value(BestAttribute, d) = v_2})

This is not the only way to learn a decision tree !!

	V1	V2	V3	V4	C
Data:	1	-1	1	1	c1
	-1	1	-1	1	c1
	-1	-1	-1	1	c1
	-1	-1	1	-1	c1
	-1	1	1	-1	c2
	-1	-1	1	1	c2
	-1	1	-1	-1	c2
	1	1	1	-1	c2

A datum d→Vector

d→Class

Training Data Set

Number of data sent down left and right branches, respectively.

Best-attribute: V4

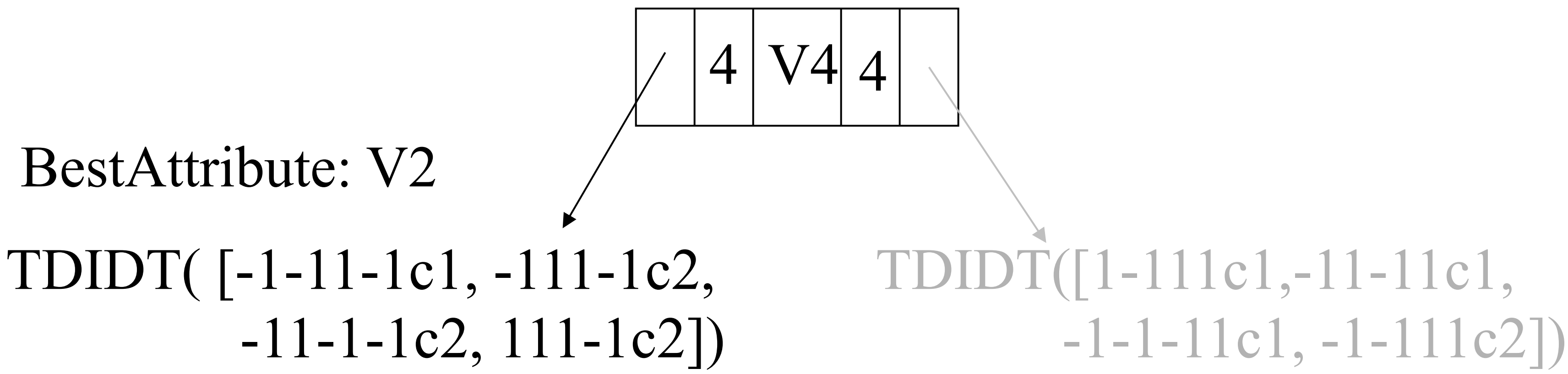
Assume left branch always corresponds to -1

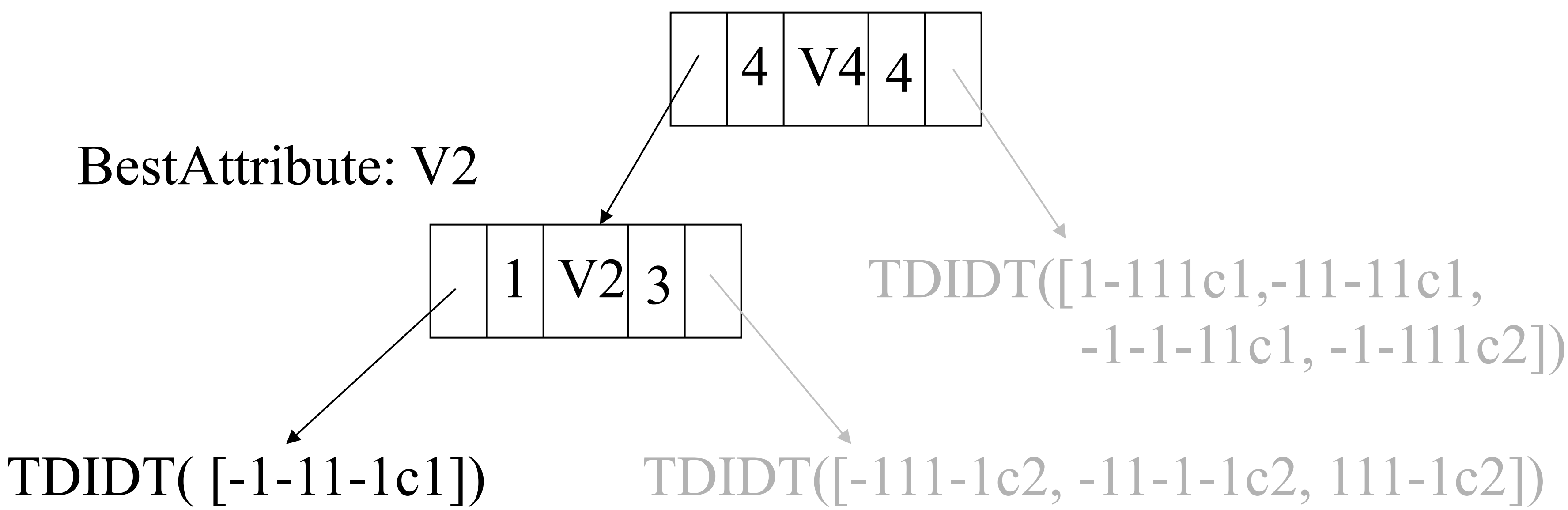
	4	V4	4	

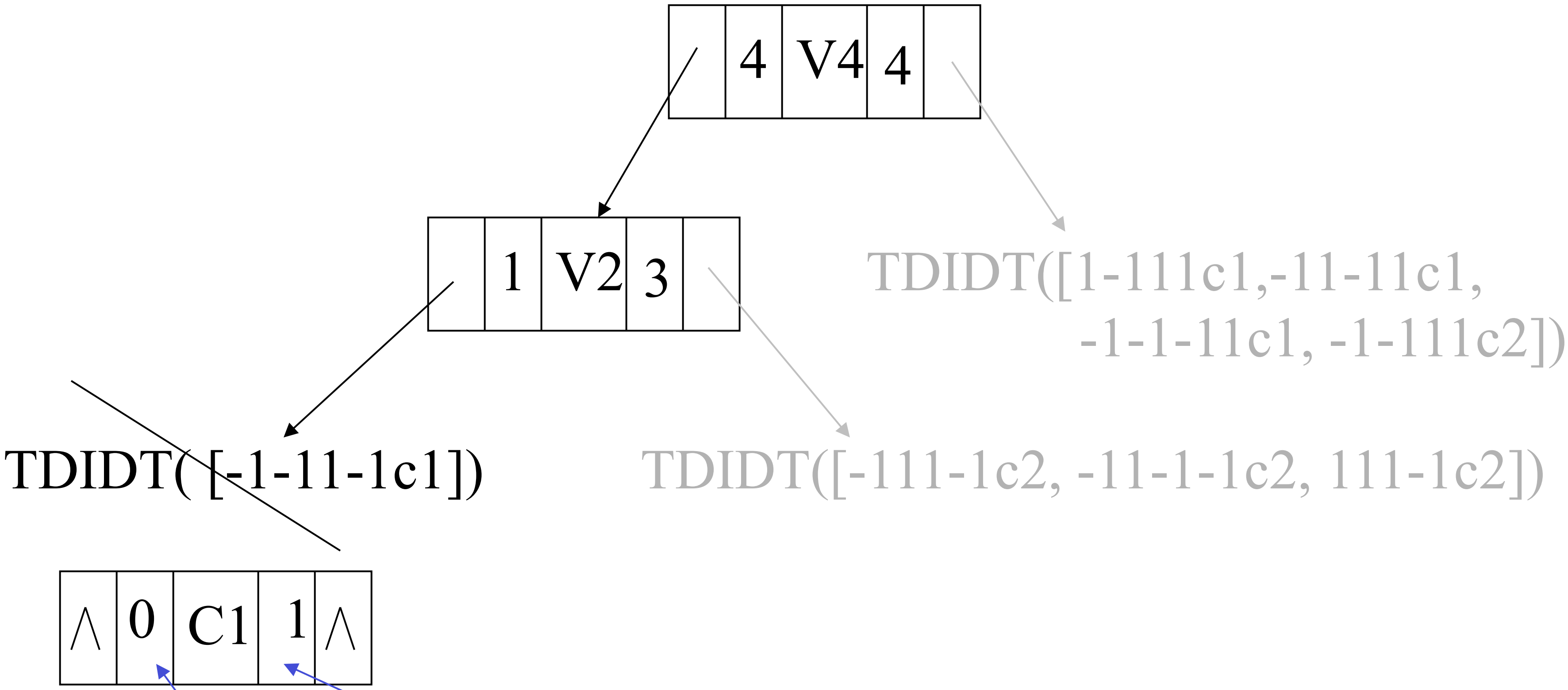
Assume right branch always corresponds to 1

TDIDT([-1-11-1c1, -111-1c2, -11-1-1c2, 111-1c2])

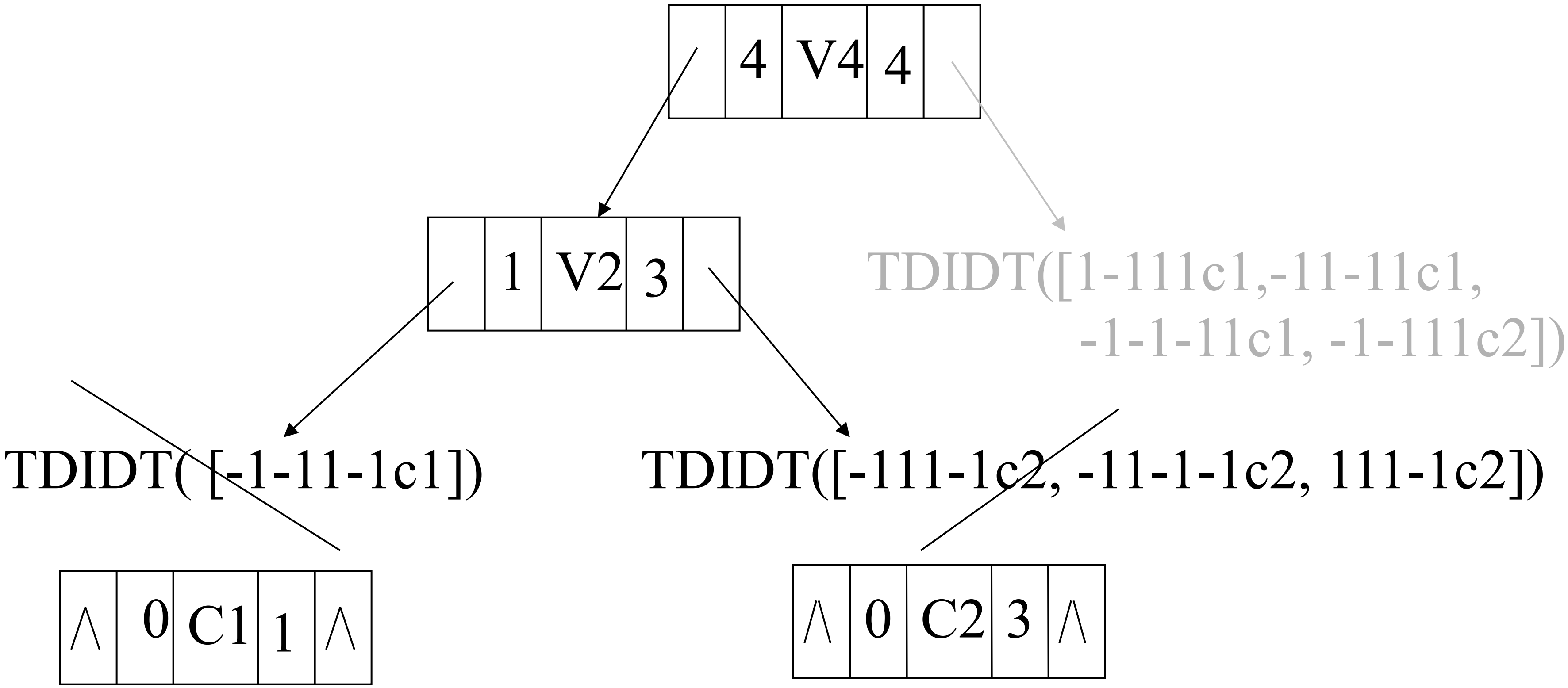
TDIDT([1-111c1, -11-11c1, -1-1-11c1, -1-111c2])


4
V4
4
BestAttribute: V2
TDIDT([-1-11-1c1, -111-1c2,
-11-1-1c2, 111-1c2])
TDIDT([1-111c1,-11-11c1,
-1-1-11c1, -1-111c2])

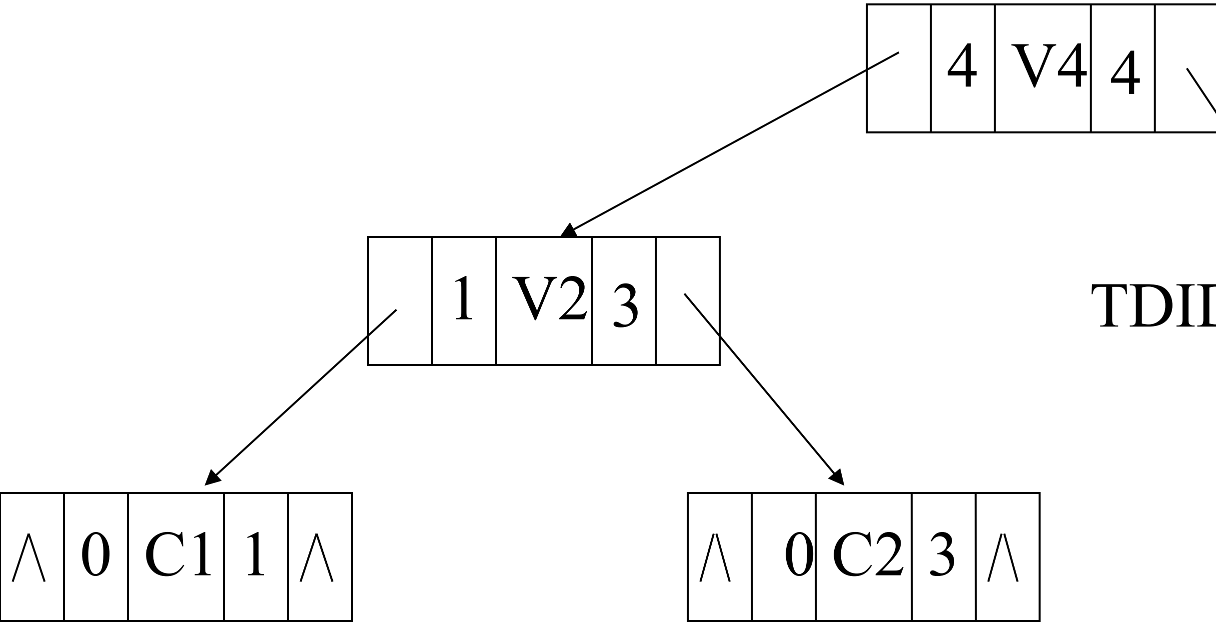

4
V4
4
BestAttribute: V2
1
V2
3
TDIDT([1-111c1,-11-11c1,
-1-1-11c1, -1-111c2])
TDIDT([-1-11-1c1])
TDIDT([-111-1c2, -11-1-1c2, 111-1c2])



Number of data at leaf in C1 (right entry) and
not in C1 (left entry)

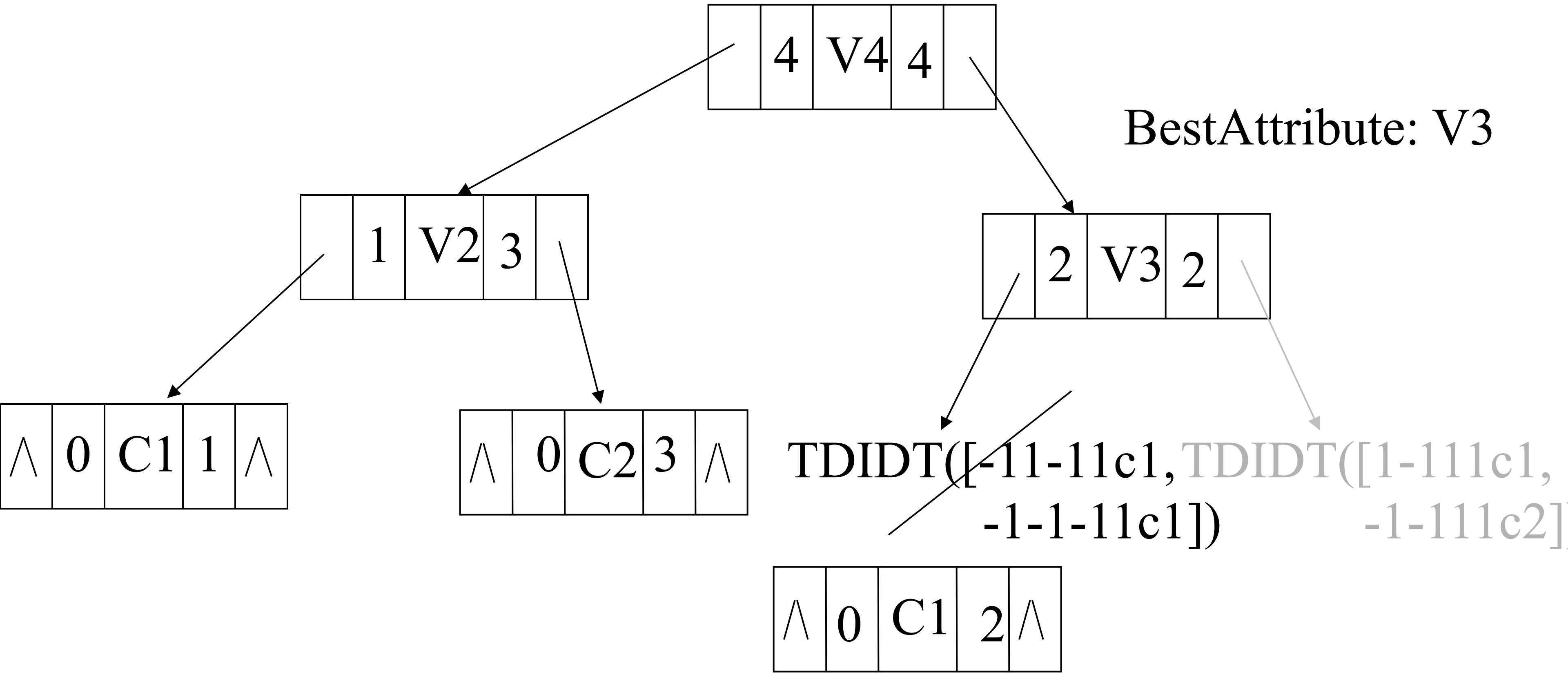


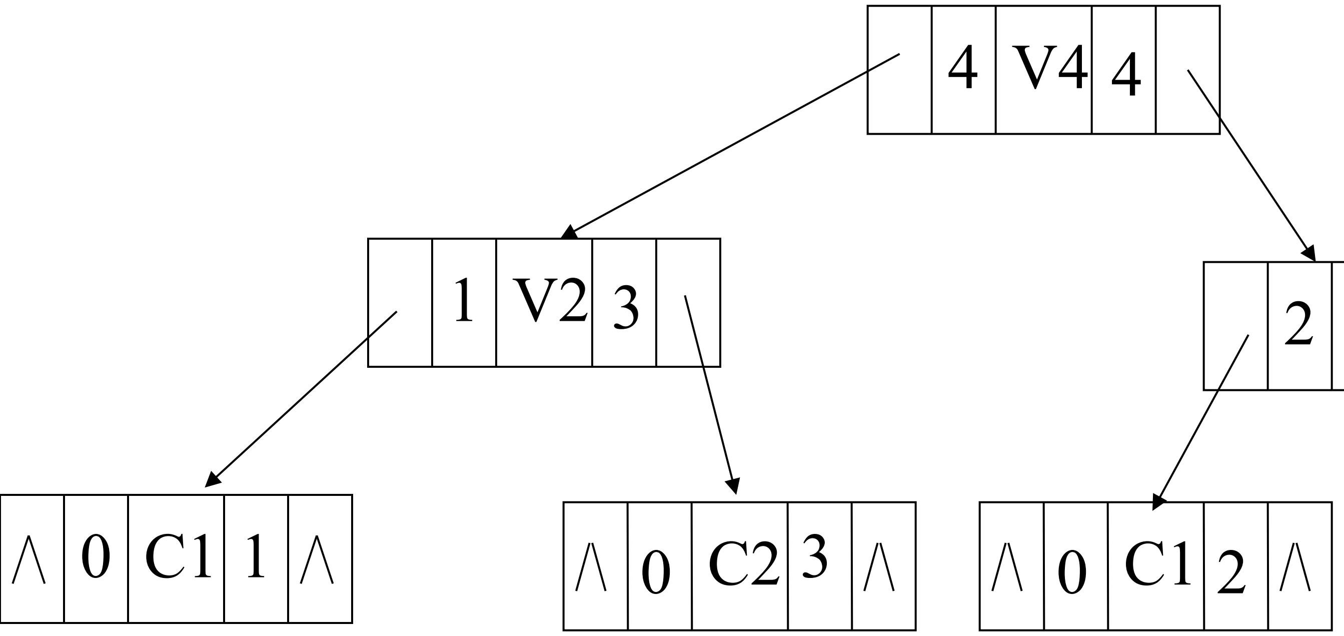
4 V4 4
1 V2 3
TDIDT([1-111c1,-11-11c1,
-1-1-11c1, -1-111c2])
TDIDT([-1-11-1c1])
TDIDT([-111-1c2, -11-1-1c2, 111-1c2])
/\ 0 C1 1 /\
/\ 0 C2 3 /\



BestAttribute: V3

TDIDT([1-111c1,-11-11c1,
-1-1-11c1, -1-111c2])

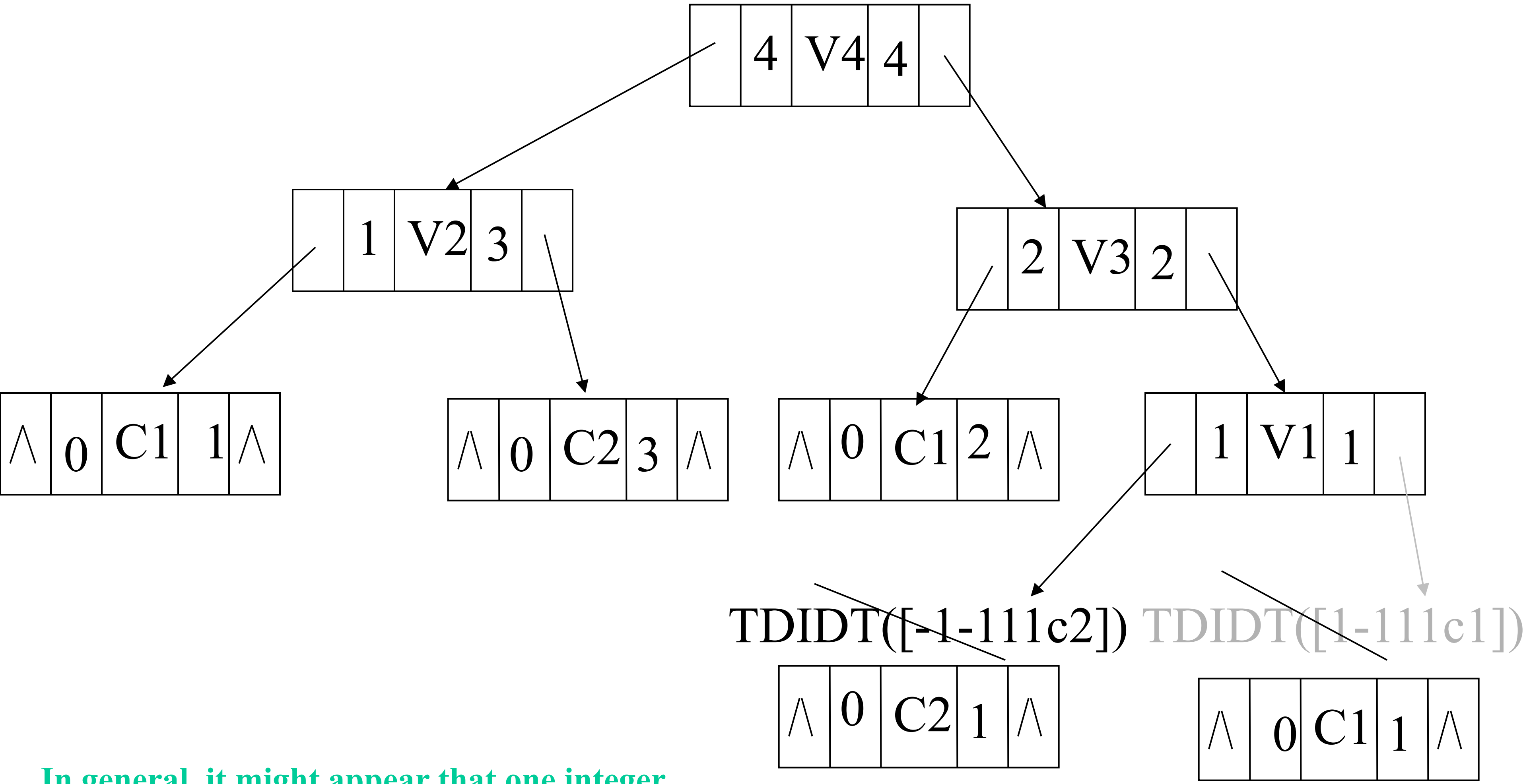

4
V4
4
BestAttribute: V3
1
V2
3
2
V3
2
/\
0
C1
1
/\
/\
0
C2
3
/\
TDIDT([-11-11c1,
-1-1-11c1])
TDIDT([1-111c1,
-1-111c2])
/\
0
C1
2
/\



TDIDT([1-111c1,
-1-111c2])

BestAttribute: V1

Douglas H. Fisher



In general, it might appear that one integer field of a leaf will always be 0, but some termination functions allow "non-pure" leaves (e.g., no split changes the class distribution *significantly*).

Selecting the best divisive attribute (SelectFN):

Attribute V_i that minimizes:

treat 0 * log 0 as 0, else a runtime error will be generated (log 0 is undefined)

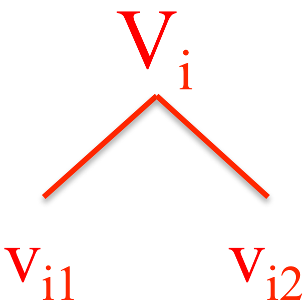
$$\underbrace{\sum_{j} P(V_i = v_{ij}) \underbrace{\sum_{k} P(C_k \mid V_i = v_{ij}) \underbrace{\left| \log P(C_k \mid V_i = v_{ij}) \right|}_{\text{\#bits necessary to encode } C_k \text{ conditioned on } V_i = v_{ij}}}_{\text{Expected number of bits necessary to encode C membership conditioned on } V_i = v_{ij}}}_{\text{Expected number of bits necessary to encode C conditioned on knowledge of } V_i \text{ value}}$$

Selecting the best divisive attribute (SelectFN):

Attribute V_i that minimizes:

treat 0 * log 0 as 0, else a runtime error will be generated (log 0 is undefined)

$$\sum_j P(V_i = v_{ij}) \sum_k P(C_k \mid V_i = v_{ij}) \, |\log P(C_k \mid V_i = v_{ij})|$$



0.5 * [[0.5 * 1] + [0.5 * 1]] +
0.5 * [[0.5 * 1] + [0.5 * 1]] = 1 V_1

0.5 * [[0.75 * 0.42] + [0.25 * 2]] +
0.5 * [[0.5 * 1] + [0.5 * 1]] = 0.9075 V_2

0.5 * [[0.75 * 0.42] + [0.25 * 2]] +
0.5 * [[0.25 * 2] + [0.75 * 0.42]] = 0.815 V_3

0.8 * [[0.9 * 0.152] + [0.1 * 3.32]] +
0.2 * [[0.3 * 1.74] + [0.7 * 0.52]] = 0.5522 V_4

0.5 * [[1.0 * 0.0] + [0.0 * undefined]] +
0.5 * [[0.0 * undefined] + [1.0 * 0.0]] = 0 V_5

Selecting the best divisive attribute (alternate):

Attribute that maximizes:

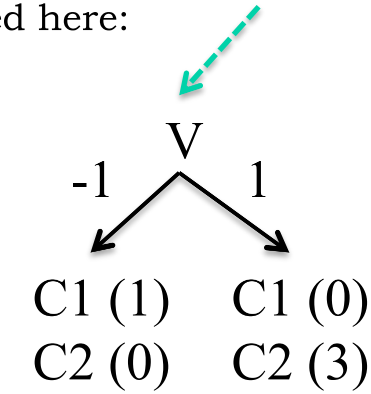
$$\sum_{j} P(Vi = vij) \sum_{k} P(Ck \mid Vi = vij)\text{^}2$$

The big picture on attribute selection:

- **if Vi and C are statistically independent, value Vi least**
- **if each value of Vi associated with exactly one C, value Vi most**
- **most cases somewhere in between**

Overfitting Illustrated

Assume that a decision tree has been constructed from training data, and it includes a node that tests on V at the frontier of the tree, with it left child yielding a prediction of class C1 (because the only training datum there is C1), and the right child predicting C2 (because the only training data there are C2). The situation is illustrated here:



Suppose that during subsequent use, it is found that

i) a large # of items (N > 1000) are classified to the node (with the test on V to the right)

ii) 50% of these have V= -1 and 50% of these have V = 1

iii) post classification analysis shows that of the N items reaching the node during usage, 25% were C1 and 75% were C2

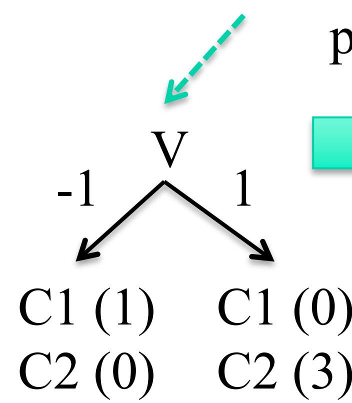
iv) of the 0.5 * N items that went to the left leaf during usage, 25% were C1 and 75% were C2

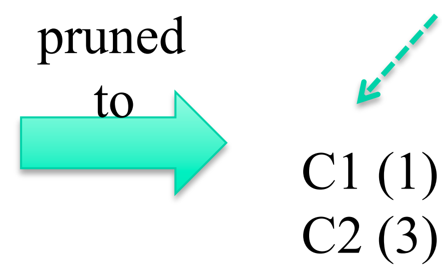
v) of the 0.5 * N items that went to the right leaf during usage, 25% were also C1 and 75% were C2

What was the error rate on the sample of N items that went to the sub-tree shown above?

0.5(0.75) + 0.5(0.25) = 0.5

What would the error rate on the same sample of N items have been if the sub-tree on previous page (and reproduced here) had been pruned to not include the final test on V, but to rather be a leaf that predicted C2?





0.25

Issue: C and V are statistically independent in this context (that is, conditionally independent)

Mitigate overfitting by statistical testing for likely dependence?

From data. Consider congressional voting records. Suppose that we have data on House votes (and political party). Suppose variables are ordered

Party, Immigration, StarWars, ….

Party P(Republican) = 0.52 (226/435 Republicans
209/435 Democrats)

To determine relationship between Party and Immigration, we count

Actual Counts

	Immigration Yes	Immigration No
Republican	17	209
Democrat	160	49

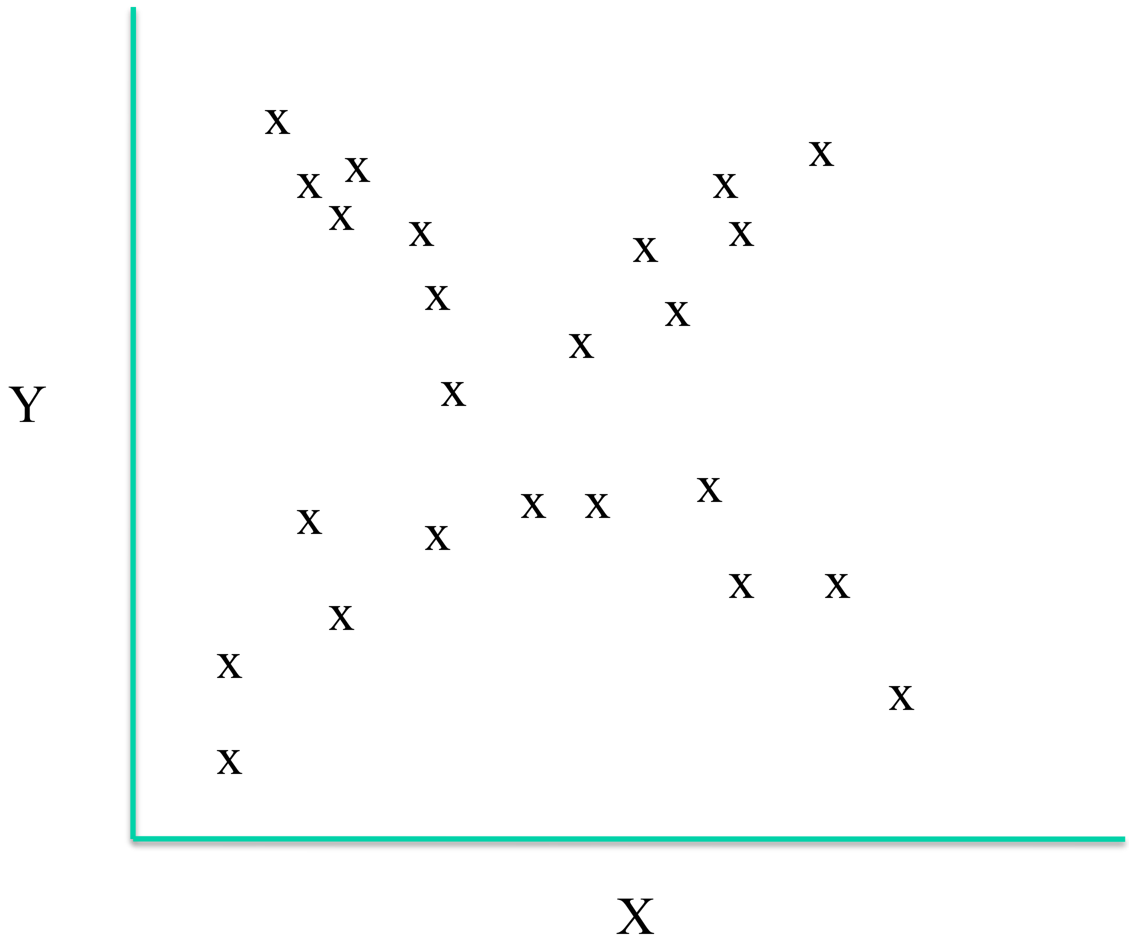
Predicted Counts (if Immigration and Party independent)

	Yes	No
Republican	92	134
Democrat	85	124

Very different distributions – conclude **dependent**

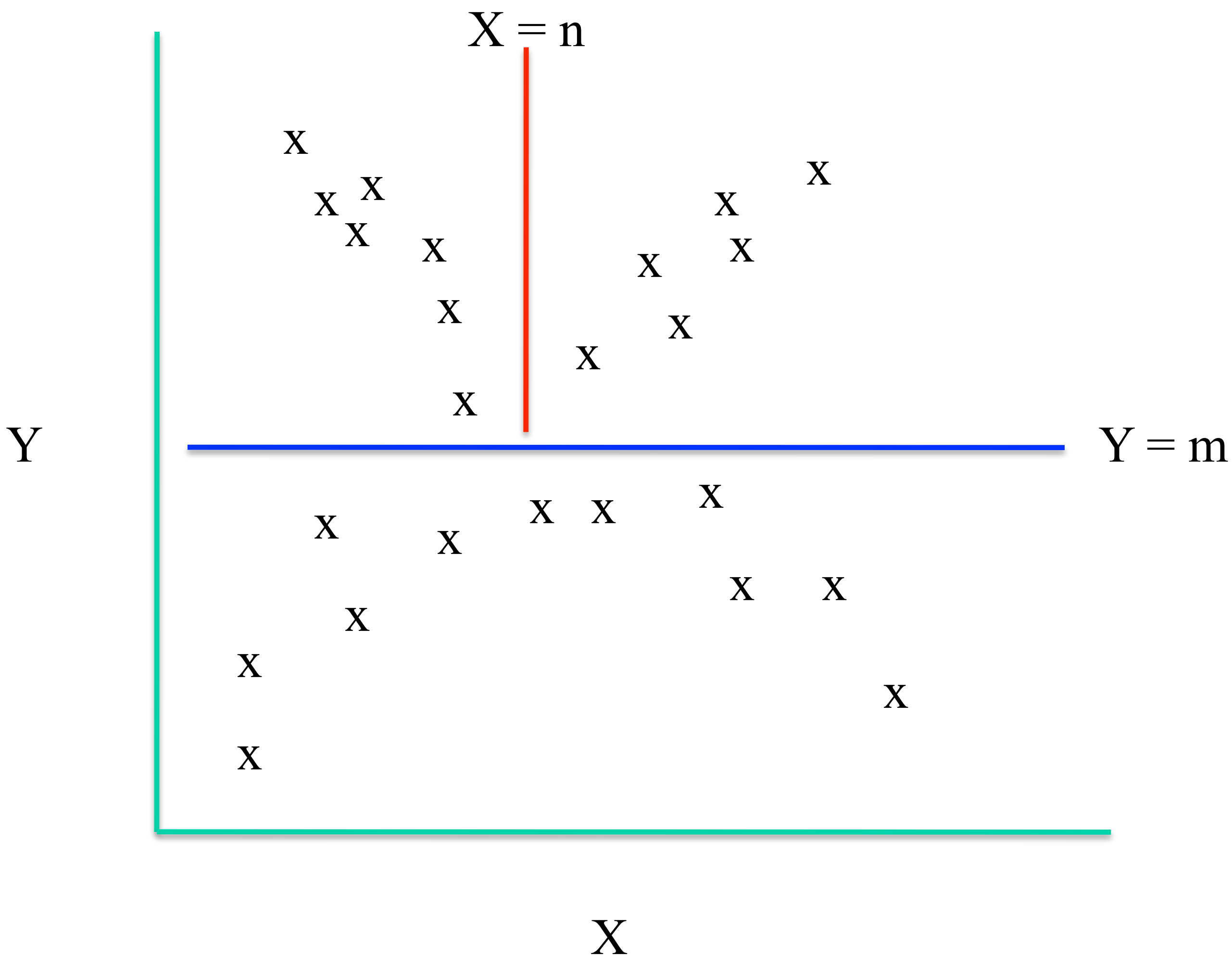
P(Rep)*P(Yes) * 435
= 0.52 * (17+160)/435 * 435

Decomposition and search are important principles in machine learning



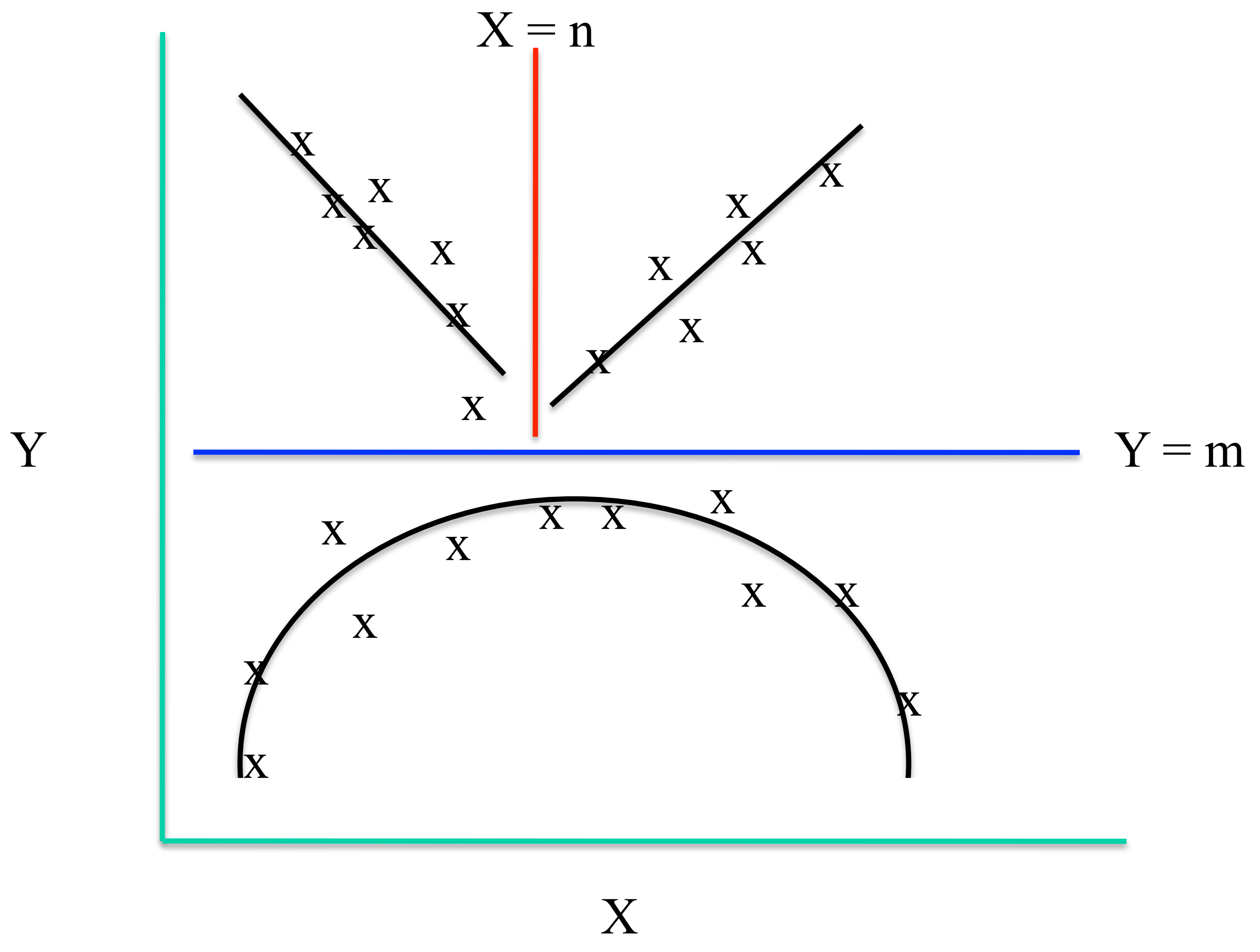
Douglas H. Fisher

Decomposition and search are important principles in machine learning



Douglas H. Fisher

Decomposition and search are important principles in machine learning

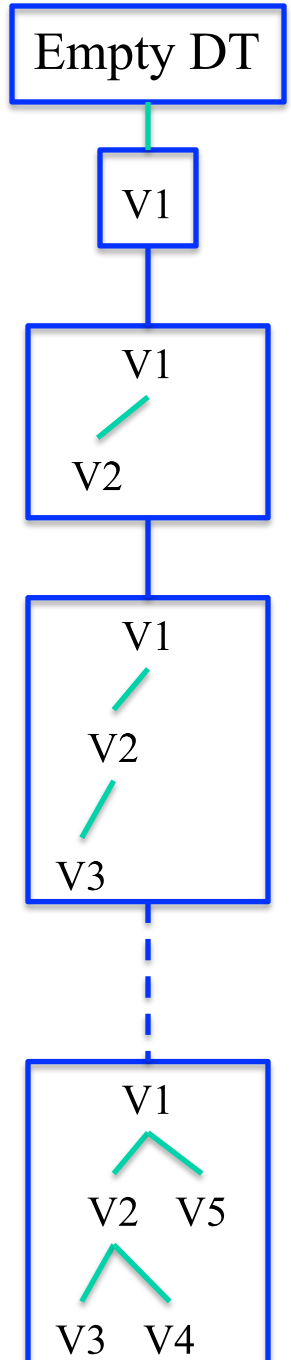


Douglas H. Fisher

Issues, variations, optimizations, etc:

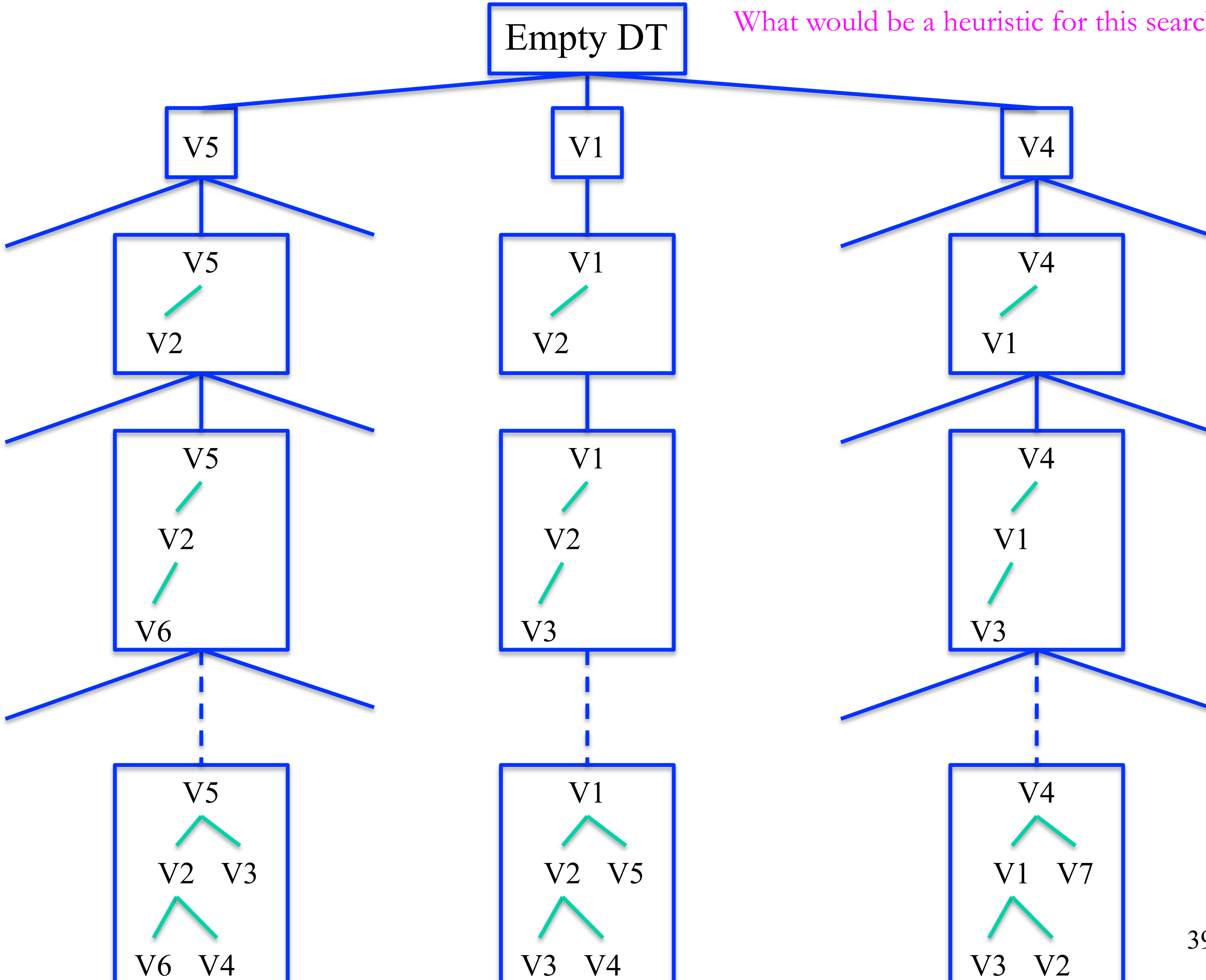
- continuous attributes
  - hard versus soft splits
- other node types (e.g., perceptron trees)
- continuous classes (regression trees)
- **termination conditions (pruning)**
- selection measures (see problem DT1)
- missing values
  - during training
  - during classification (see expansion)
- noise in data
- irrelevant attributes
- **less greedy variants (e.g., lookahead, search)**
- incremental construction
- applications (e.g., Banding)
- cognitive modeling (e.g., Hunt)
- DT based approaches to nearest neighbor search, object recognition
- background **knowledge** to augment feature space
- **ensembles (forests of decision trees)**

The top-down greedy method is essentially a “hill climb” (section 4.7.1) in what could be a much more extensive search



The top-down greedy method tends to result in “small” and accurate trees, but a systematic search could do better

What would be a heuristic for this search?



Ensembles of classifiers
Decision Forests

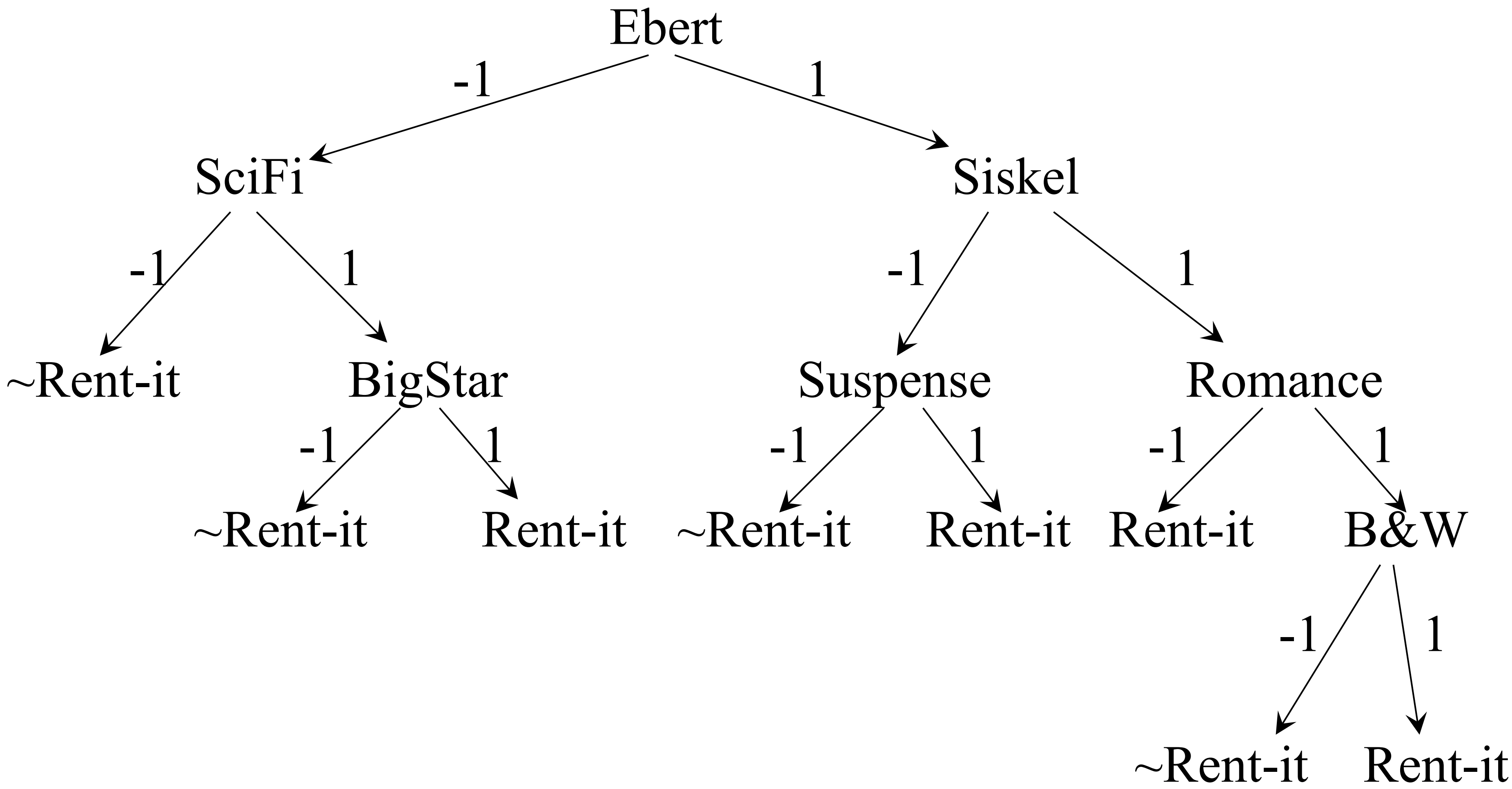
"Bagging" is one (of several) methods for building a forest. Assume that there are N training data D

Embed greedy DT induction into a loop

```
For i = 1 to desired size of forest {

    Training Set, TrS = Randomly sample N times from D, with replacement

    Run greedy DT induction on TrS

    Output resulting tree to forest
}
```

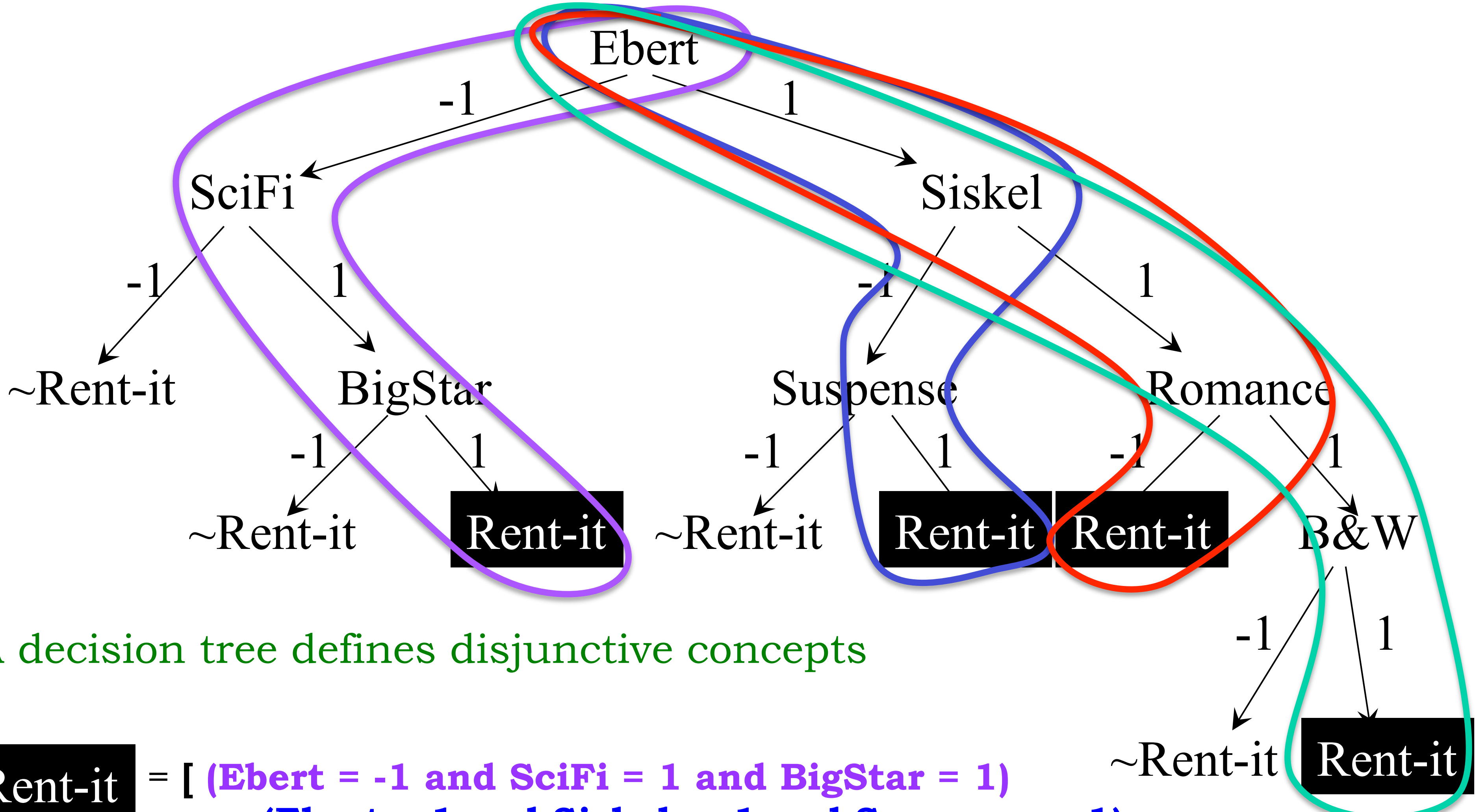
To use the forest classifier, run a test datum through each tree of the forest and take a vote on its classification

More on decision tree classifiers



A decision tree defines disjunctive concepts
(in DNF)

Each path of a decision tree represents a conjunction of values

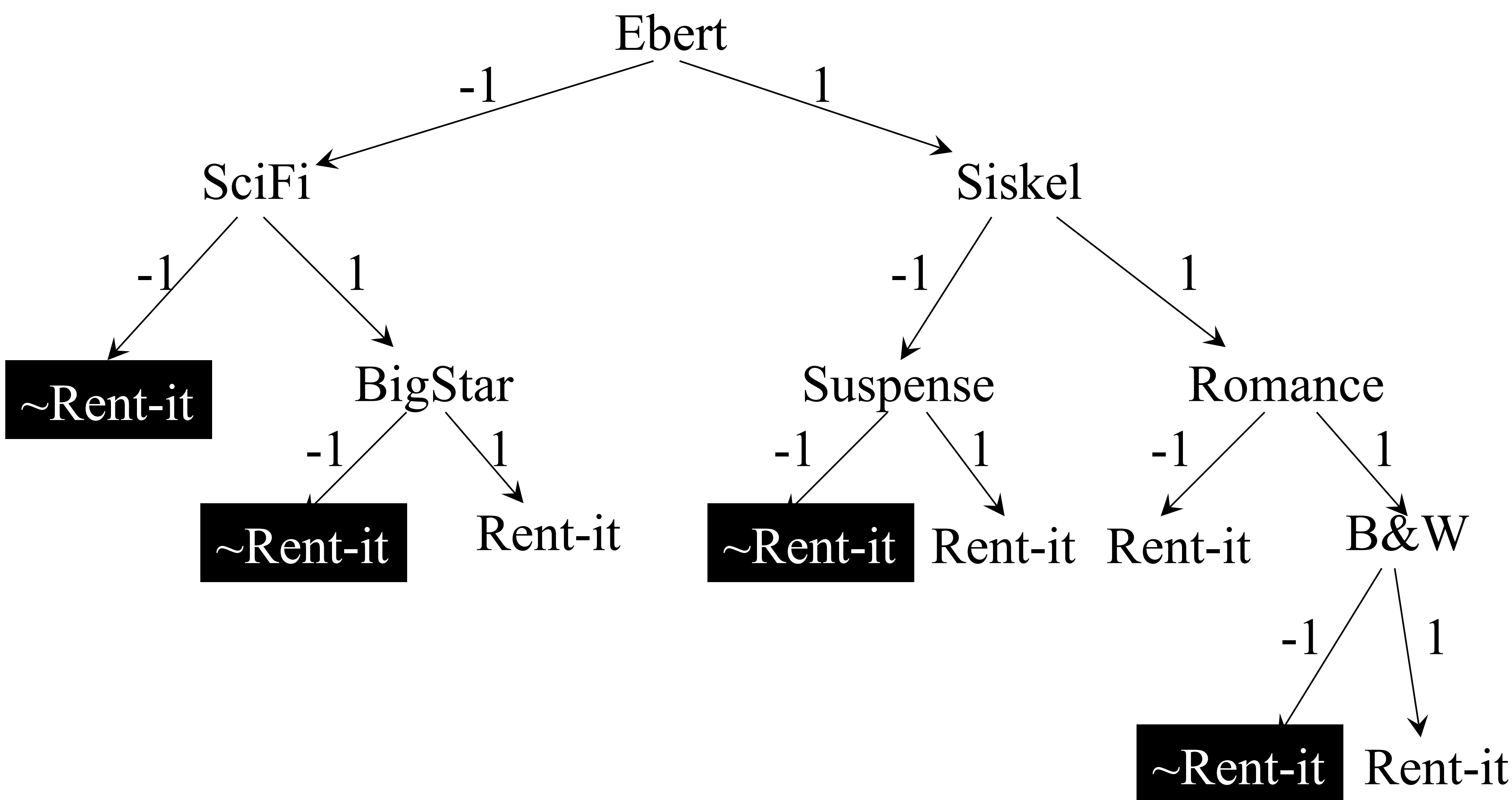


A decision tree defines disjunctive concepts

Rent-it = [**(Ebert = -1 and SciFi = 1 and BigStar = 1)**
or (Ebert = 1 and Siskel = -1 and Suspense = 1)
or (Ebert = 1 and Siskel = 1 and Romance = -1)
or (Ebert = 1 and Siskel = 1 and Romance = 1 and B&W = 1)]

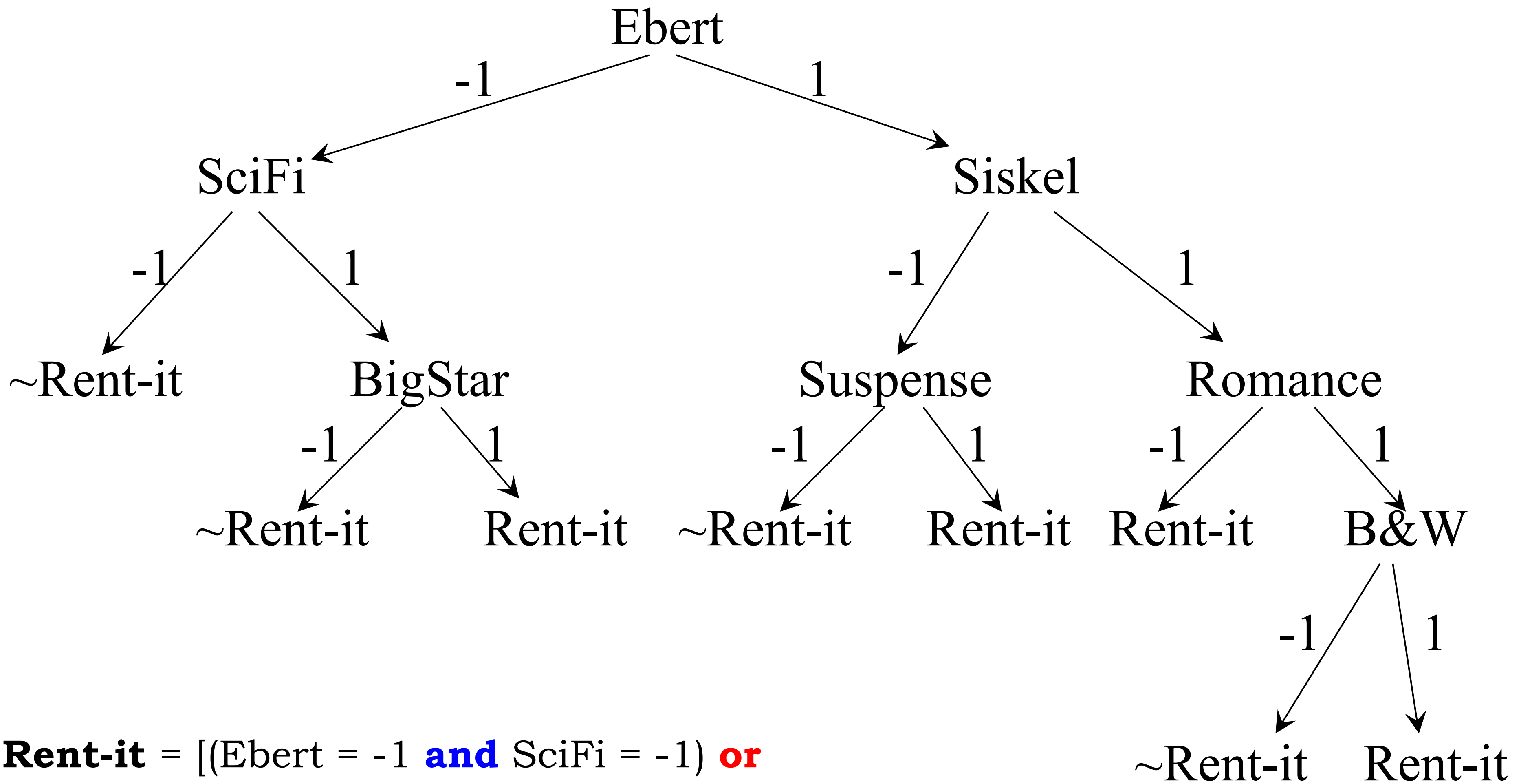
Douglas H. Fisher

What is the DNF representation of **~Rent-it** ?



~Rent-it = ?

~Rent-it definition: each path to a leaf labeled by ~Rent-it is a disjunct in the DNF expression



~Rent-it = [(Ebert = -1 **and** SciFi = -1) **or**
(Ebert = -1 **and** SciFi = 1 **and** BigStar = -1) **or**
(Ebert = 1 **and** Siskel = -1 **and** Suspense = -1) **or**
(Ebert = 1 **and** Siskel = 1 **and** Romance = 1 **and** B&W = -1)
]

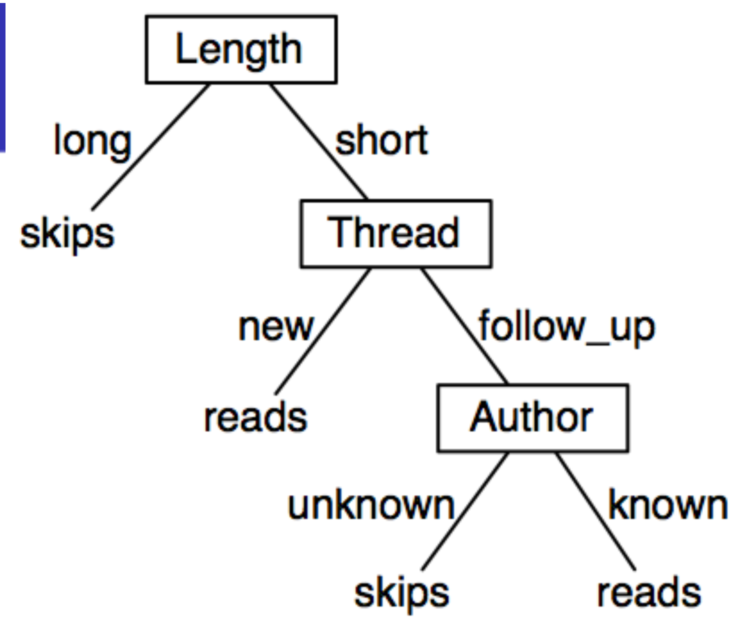
Rent-it = [(Ebert = -1 **and** SciFi = 1 **and** BigStar = 1)
or (Ebert = 1 **and** Siskel = -1 **and** Suspense = 1)
or (Ebert = 1 **and** Siskel = 1 **and** Romance = -1)
or (Ebert = 1 **and** Siskel = 1 **and** Romance = 1 **and** B&W = 1)]

In propositional form, write X=1 as X and X= -1 as ~X,
'and' as ∧ and 'or' as ∨

Rent-it = [(~ebert ∧ scifi ∧ bigstar)
∨ (ebert ∧ ~siskel ∧ suspense)
∨ (ebert ∧ siskel ∧ ~romance)
∨ (ebert ∧ siskel ∧ romance ∧ b&w)]

~Rent-it = [(~ebert ∧ ~scifi)
∨ (~ebert∧ sciFi ∧ ~bigstar)
∨ (ebert ∧ ~siskel ∧ ~suspense)
∨ (ebert ∧ siskel ∧ romance ∧ ~b&w)]

Equivalent Logic Program



$$skips \leftarrow long.$$
$$reads \leftarrow short \land new.$$
$$reads \leftarrow short \land follow\_up \land known.$$
$$skips \leftarrow short \land follow\_up \land unknown.$$

or with negation as failure:

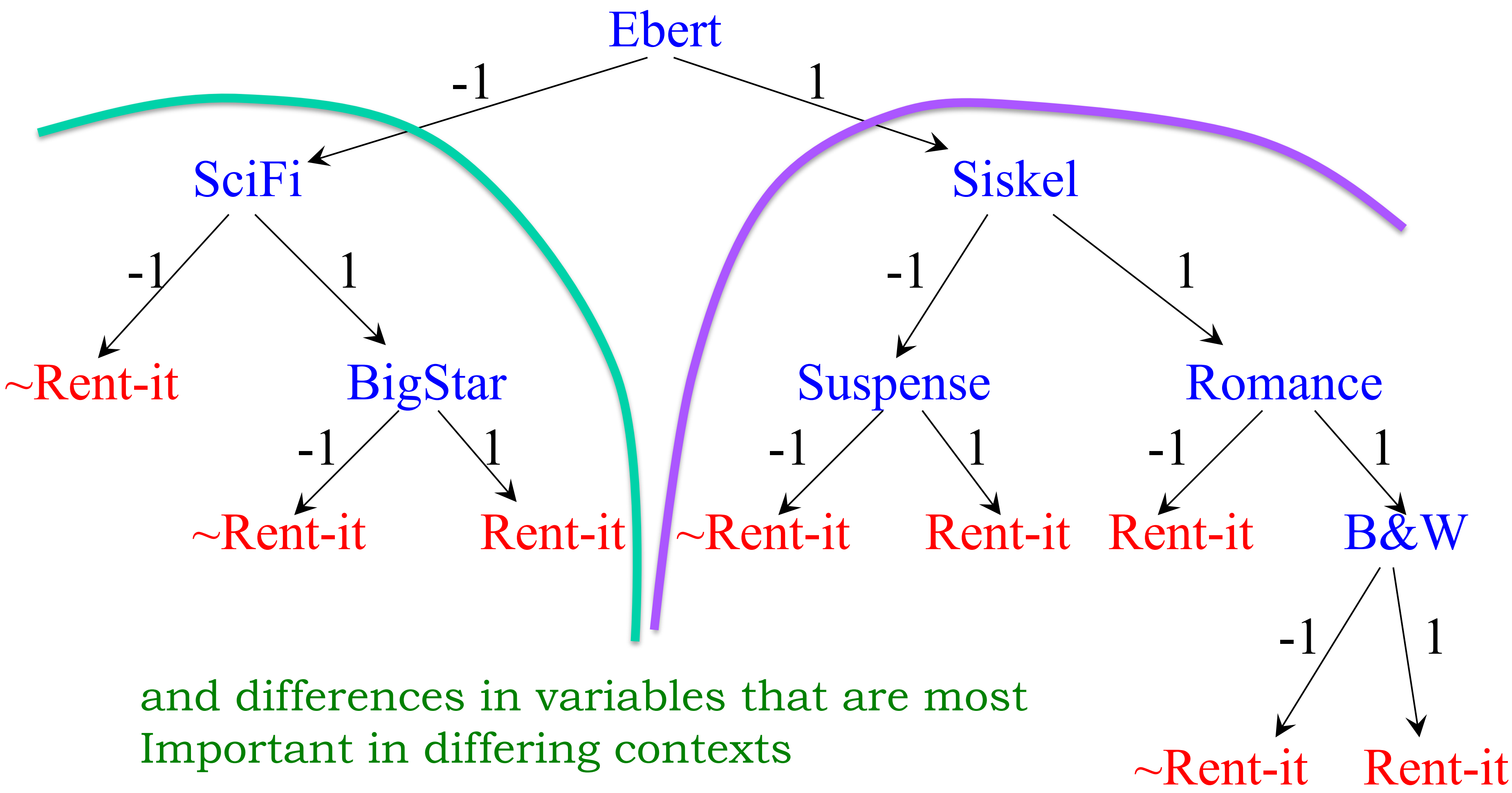
$$reads \leftarrow short \land new.$$
$$reads \leftarrow short \land \sim new \land known.$$

A decision tree covers all possible data defined over the tree's variables:

Show that

$$
\begin{aligned}
\sim[\ & (\sim ebert \wedge scifi \wedge bigstar) \\
& \vee (ebert \wedge \sim siskel \wedge suspense) \\
& \vee (ebert \wedge siskel \wedge \sim romance) \\
& \vee (ebert \wedge siskel \wedge romance \wedge b\&w)\ ] \\
=\ & \\
[\ & (\sim ebert \wedge \sim scifi) \\
& \vee (\sim ebert \wedge scifi \wedge \sim bigstar) \\
& \vee (ebert \wedge \sim siskel \wedge \sim suspense) \\
& \vee (ebert \wedge siskel \wedge romance \wedge \sim b\&w)\ ]
\end{aligned}
$$

Decision trees explicitly encode context



and differences in variables that are most
Important in differing contexts

Different kinds of variables
(though all appear the same to the learning system)

Low level descriptive variables, such as "black-and-white?"
or even continuous variables (e.g., runtime < 90 min or >= 90min)

Variables with values that are values of well-defined functions over "basic" variables (e.g., logical equivalence of two binary variables; the square of a more basic continuous variable)

Variables with values that are complex (and UNKNOWN) functions of other variables:

Genre (human consensus)

Human recommendations (experts, friends, etc)

Other recommender systems (or AIs generally)
like those of Netflix, iTunes, etc