Computing and the Environment

Douglas H. Fisher

Machine Learning Week 3

Look up these terms from "Tackling Climate Change with Machine Learning" and understand them at a high level of abstraction. The first paragraph or two of Wikipedia articles are decent descriptions in most cases (but deep learning and interpretable machine learning, not so much, so poke around). If you want insights into the algorithmic approaches used, reflect on how you "implement" the various process abstractions.

Supervised learning
Unsupervised learning
Clustering
Semi-supervised learning
Reinforcement learning
Transfer learning
Interpretable learning
Deep learning

Post questions, observations, and insights to Piazza. Know other sustainability terminology as well.

Computing and the Environment

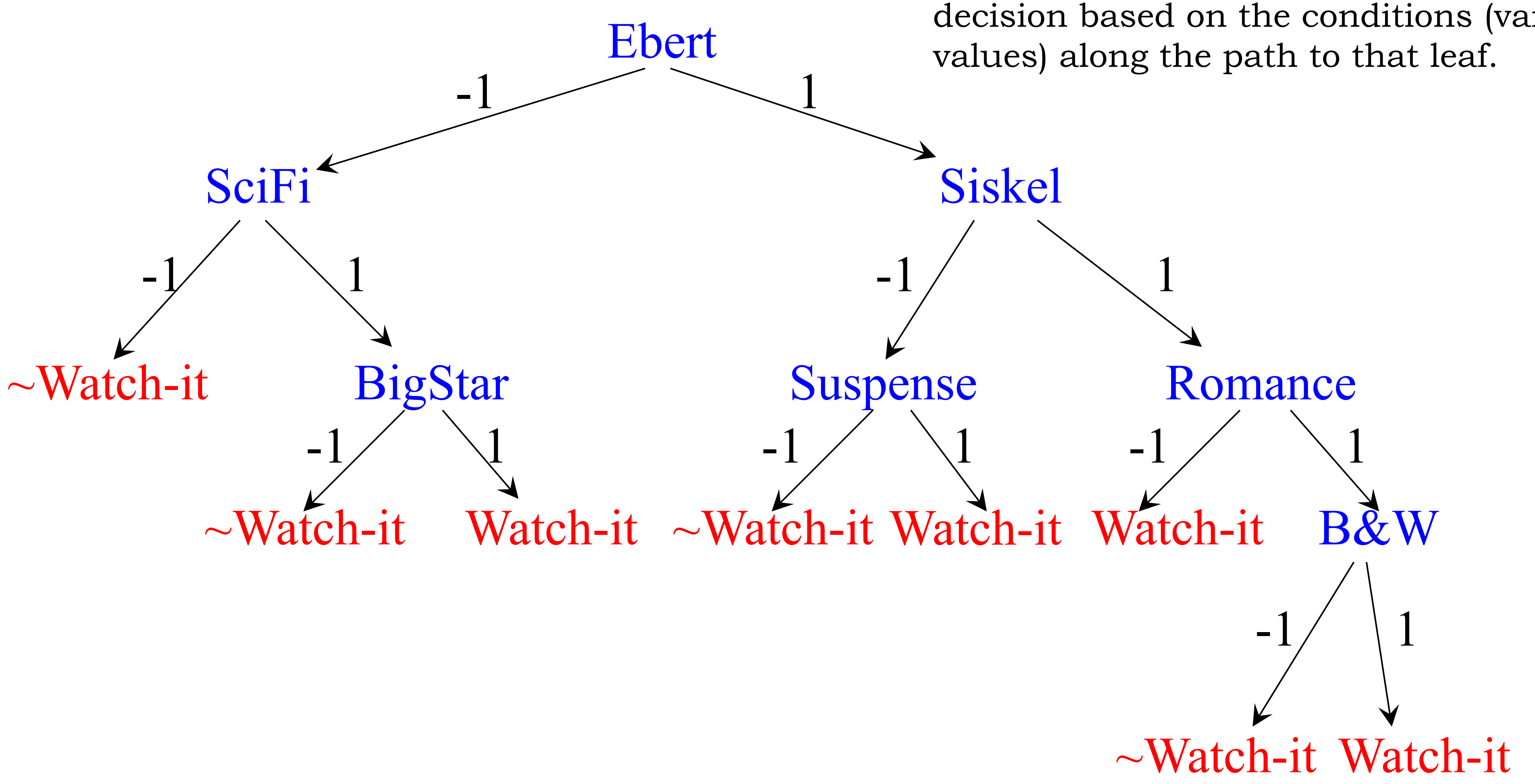
Douglas H. Fisher

Machine Learning Week 3

Decision Trees

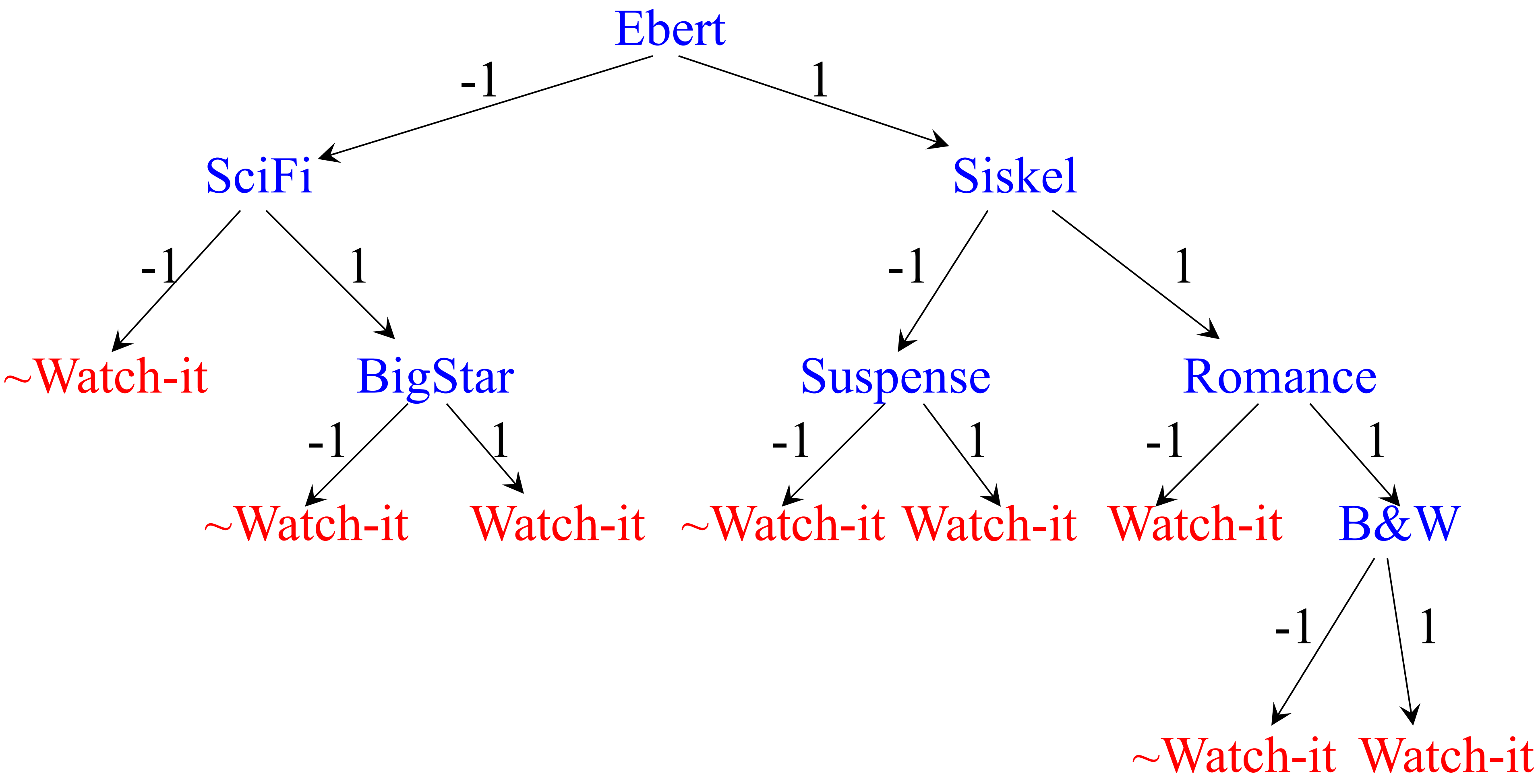
Decision tree classifiers

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.



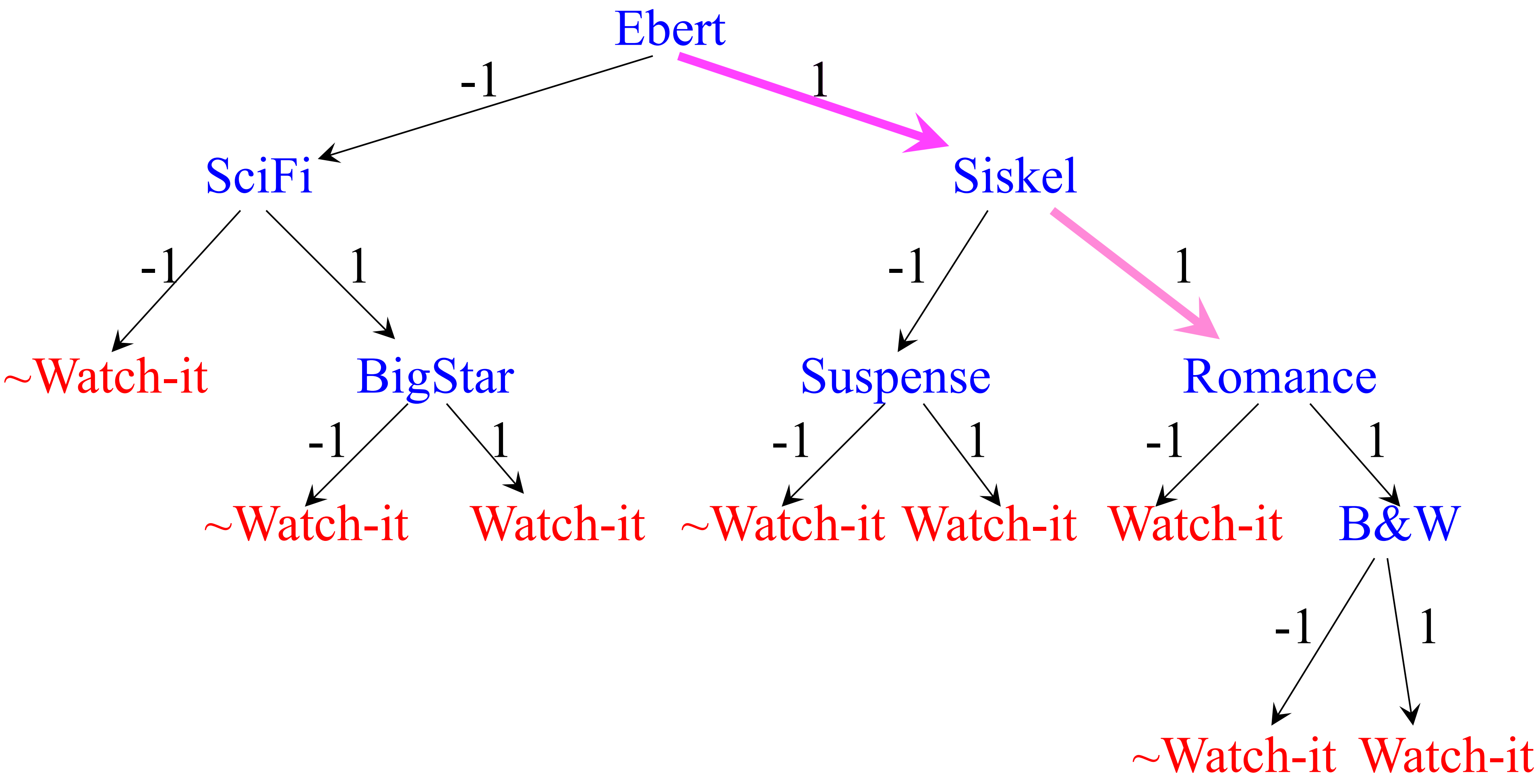
Decision tree classifiers

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



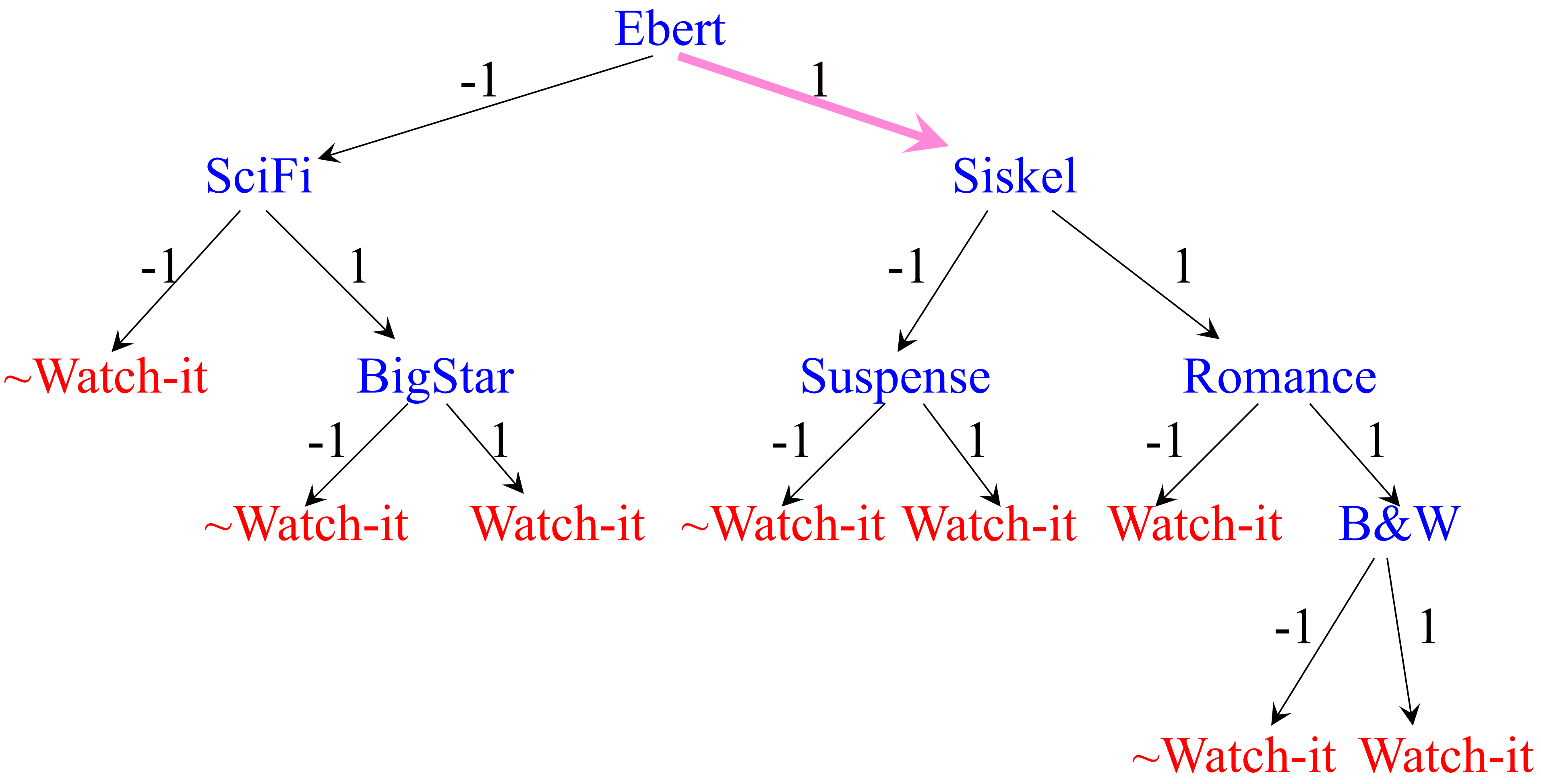
Decision tree classifiers

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



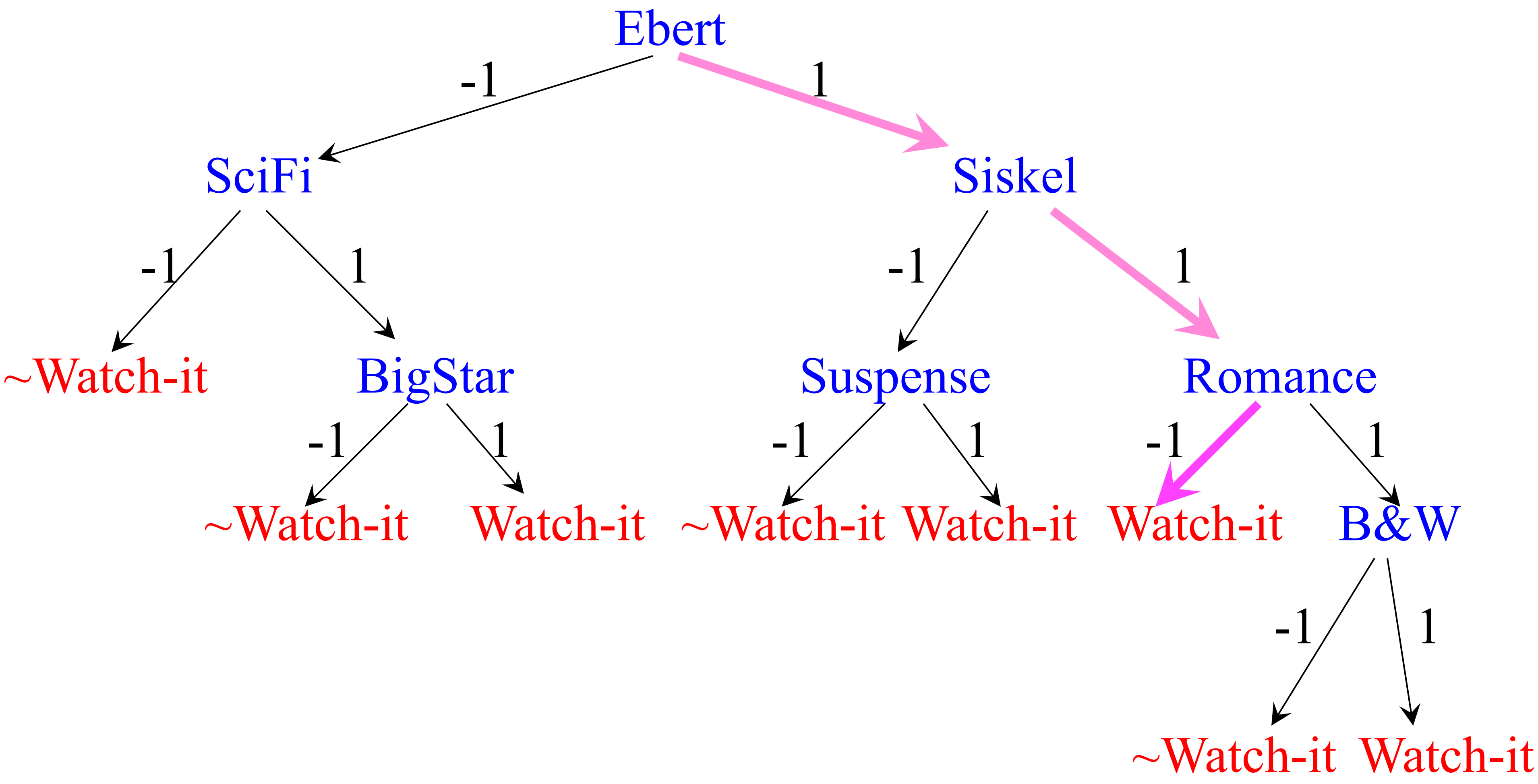
Decision tree classifiers

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



Decision tree classifiers

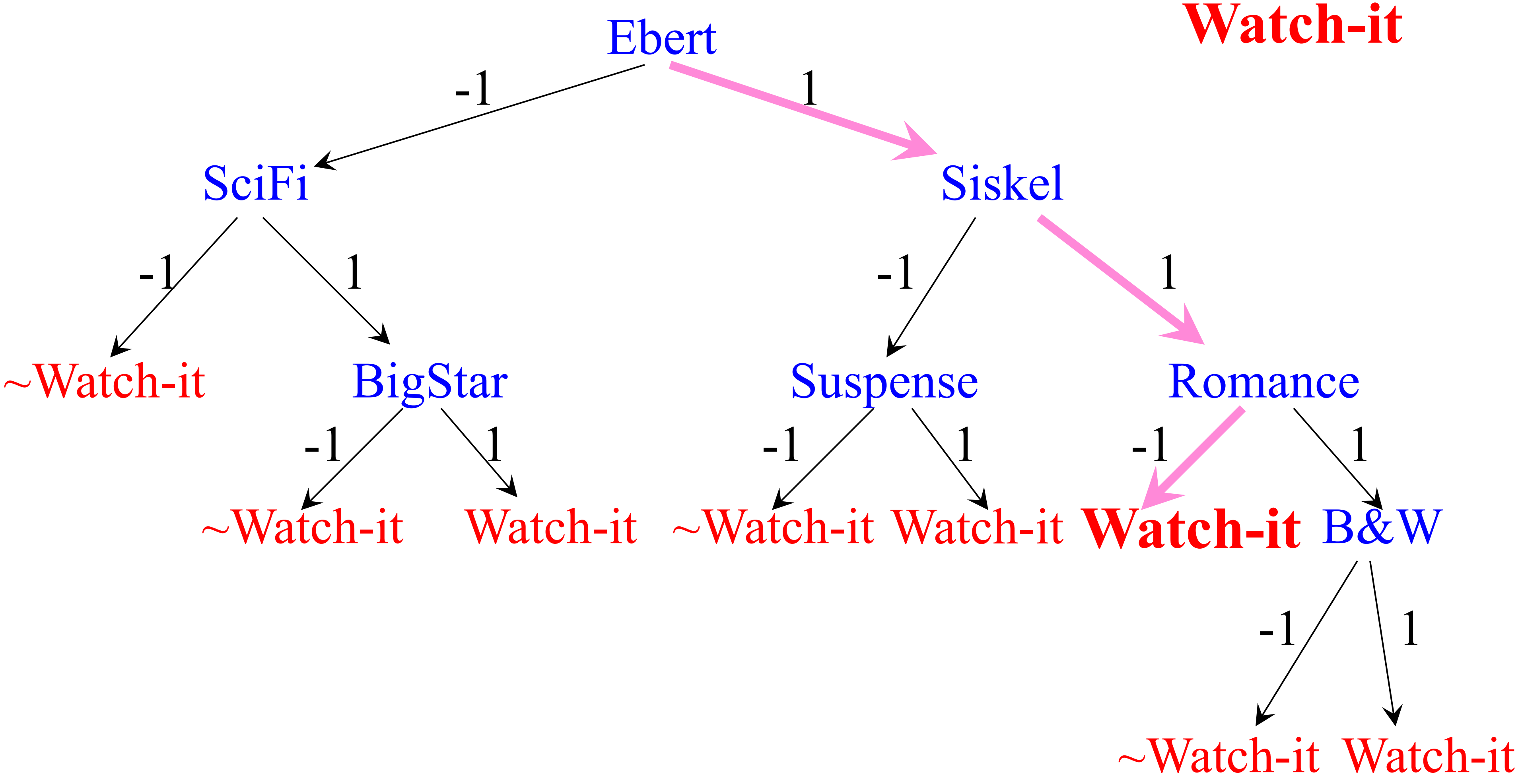
[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, Watch-it???]



Decision tree classifiers

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

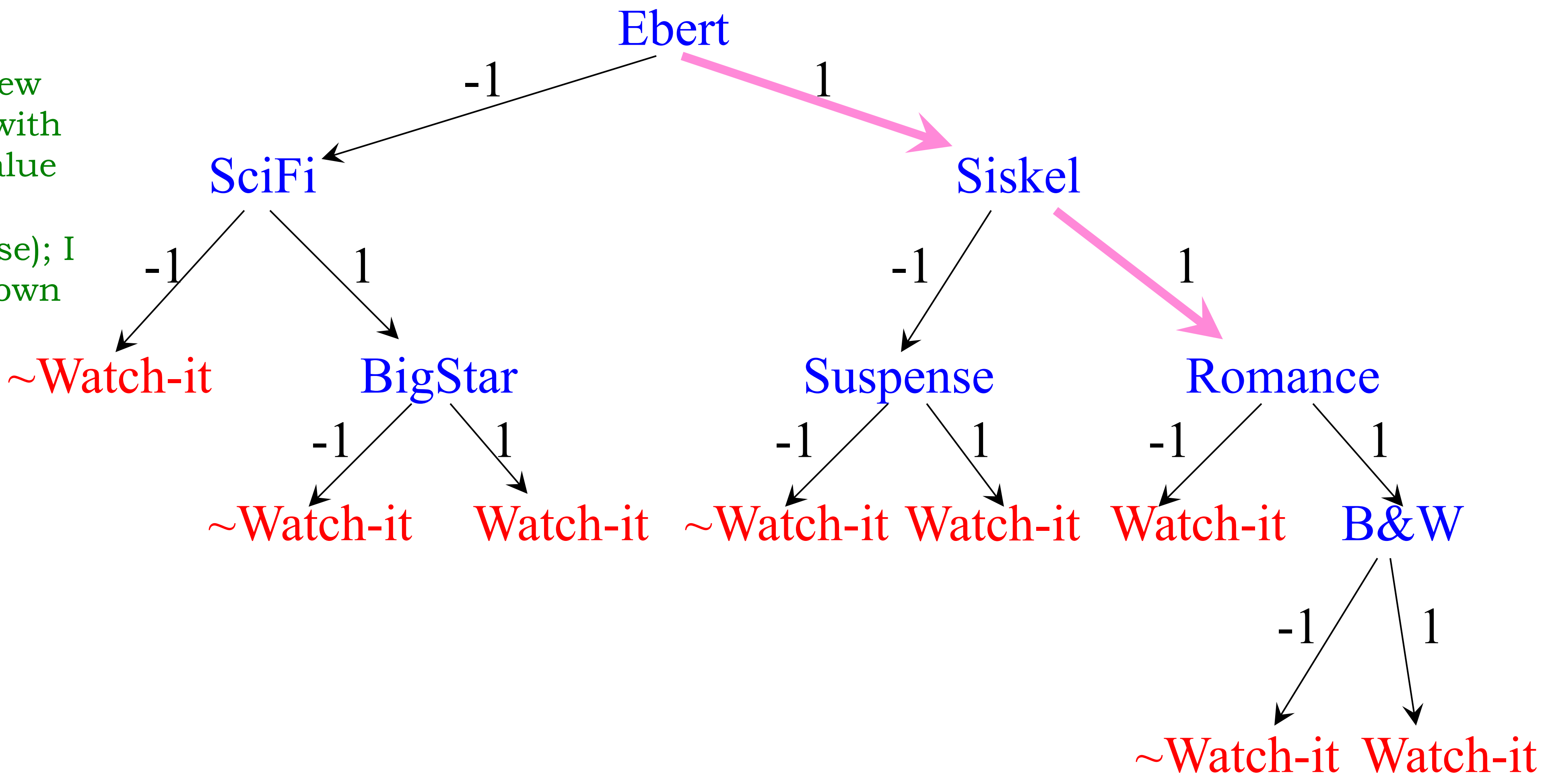
Watch-it



Decision tree classifiers

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

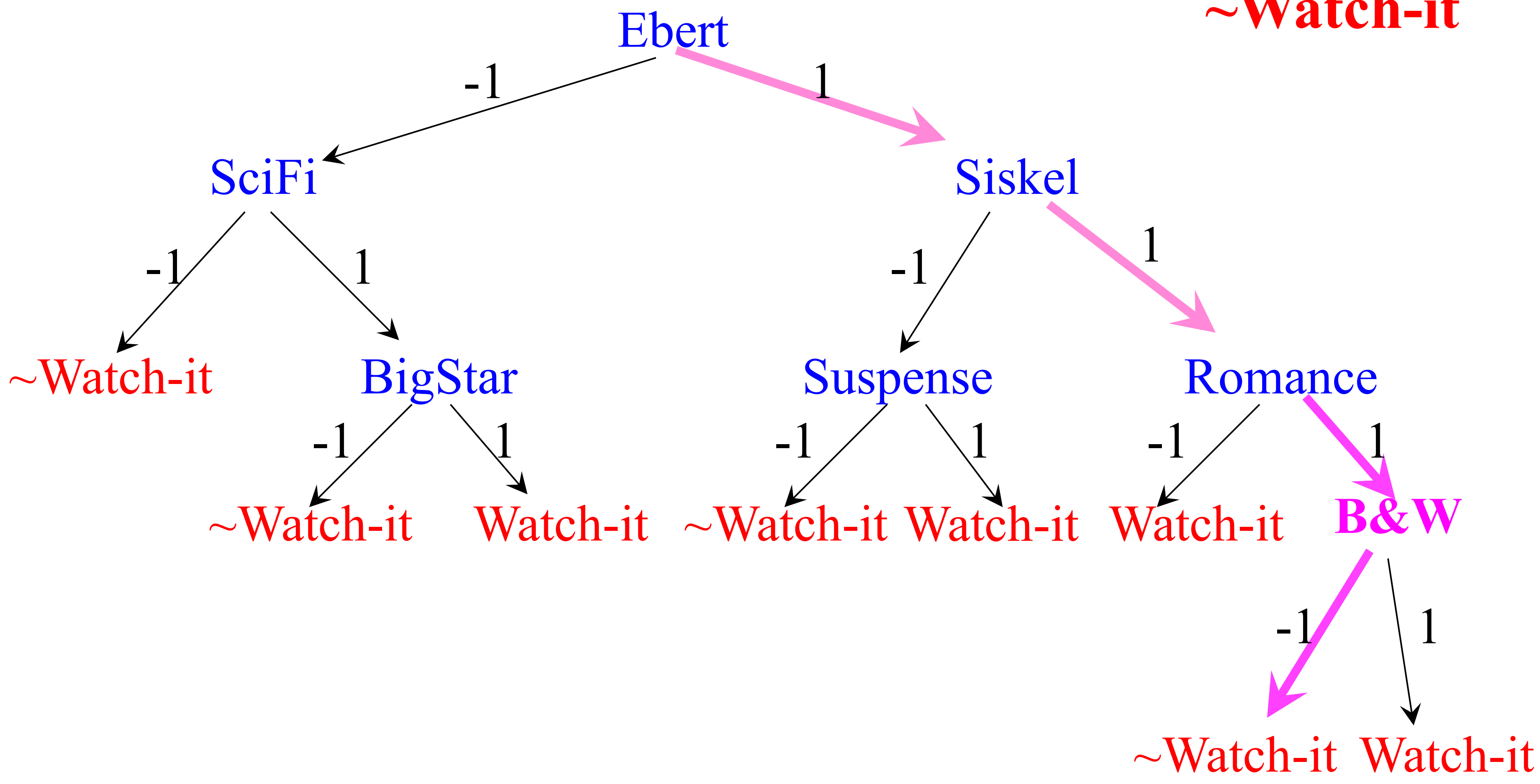
Consider a completely new test datum, with a different value for Romance (and Suspense); I have also shown the value for B&W



Decision tree classifiers

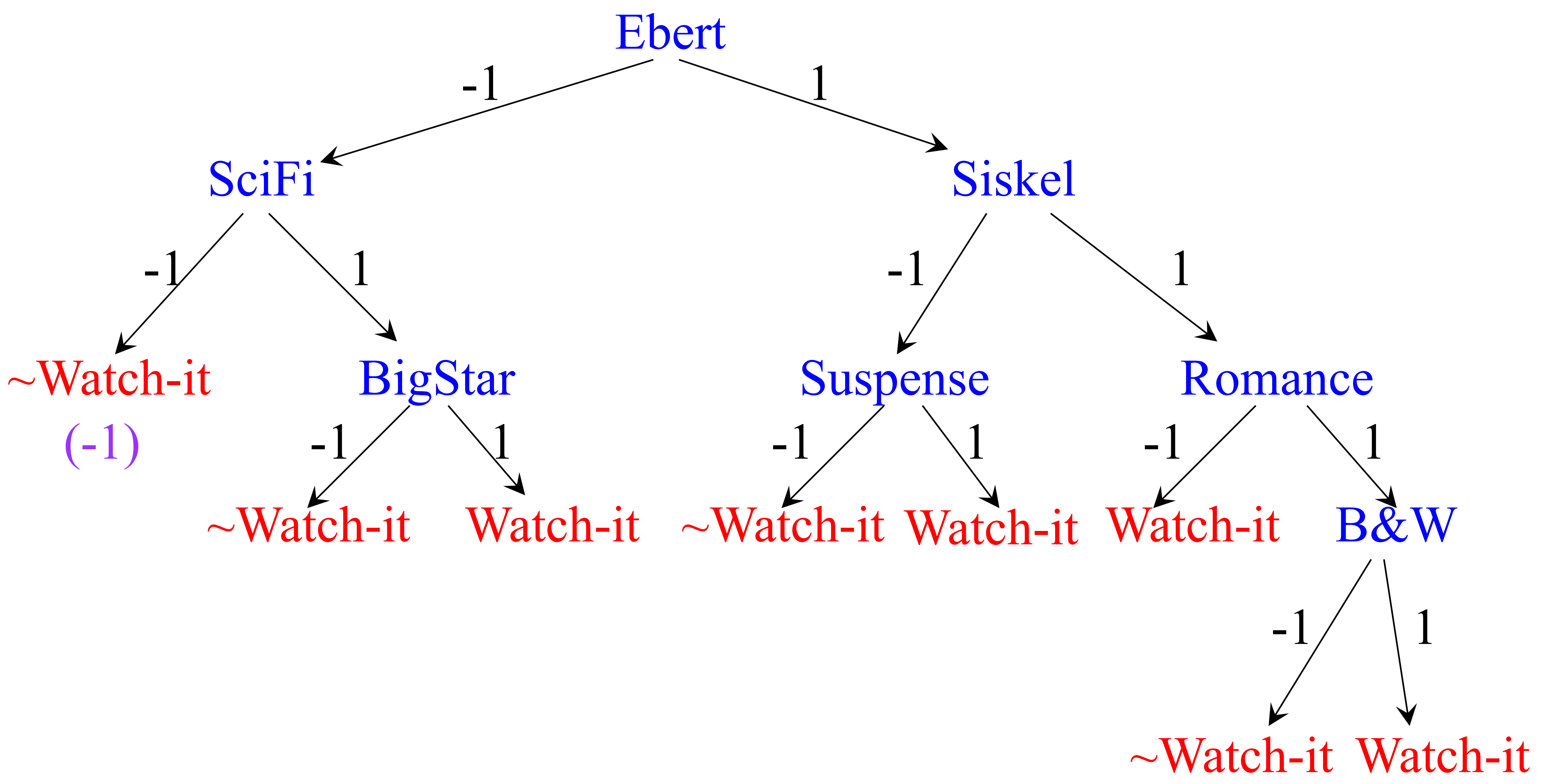
[SciFi = -1, Suspense = -1, Romance = ~~-1~~ 1, Ebert = 1, Siskel = 1, B&W = -1, …, ~~Watch-it???~~]

~Watch-it



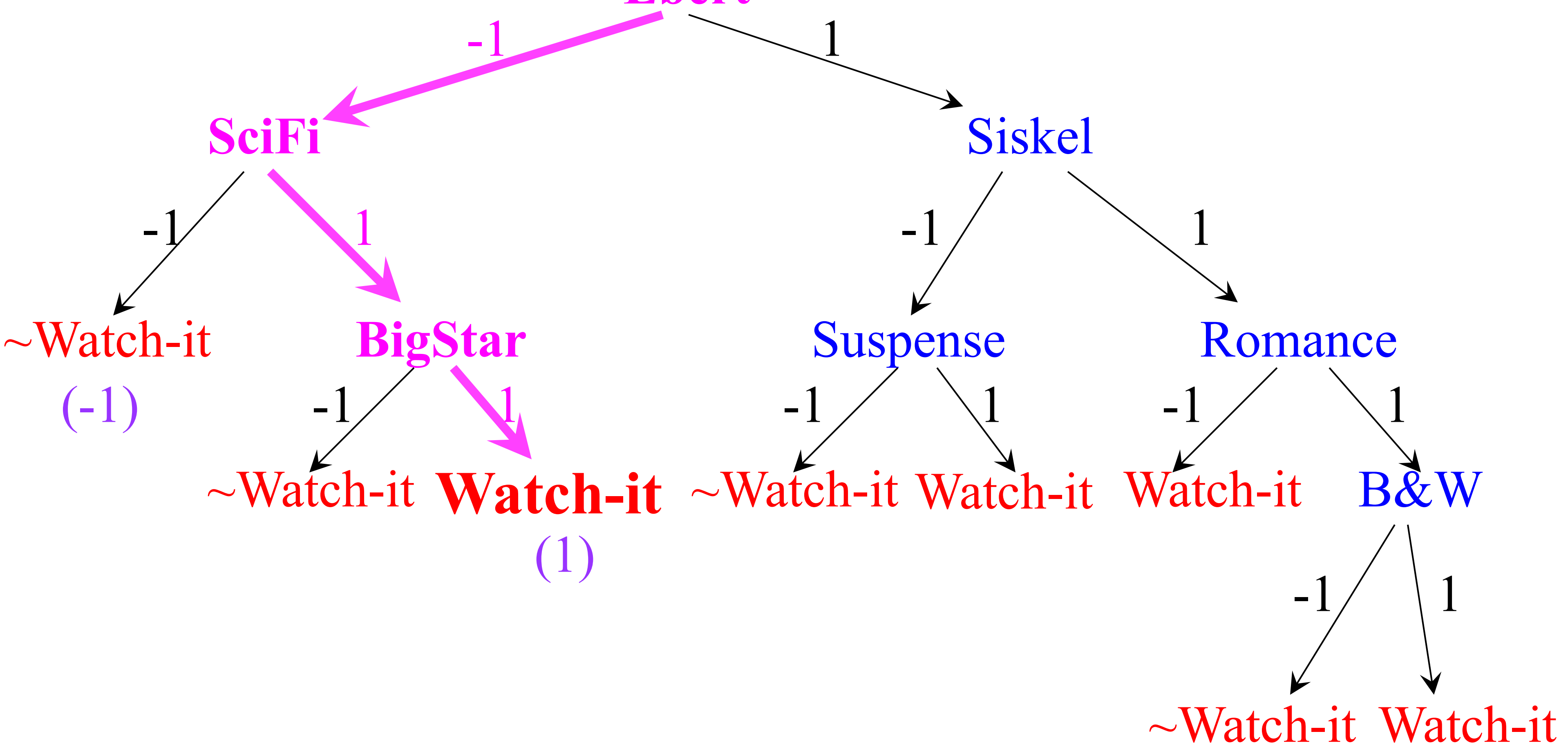
What decision would be made for the following datum, Watch-it or ~Watch-it ?

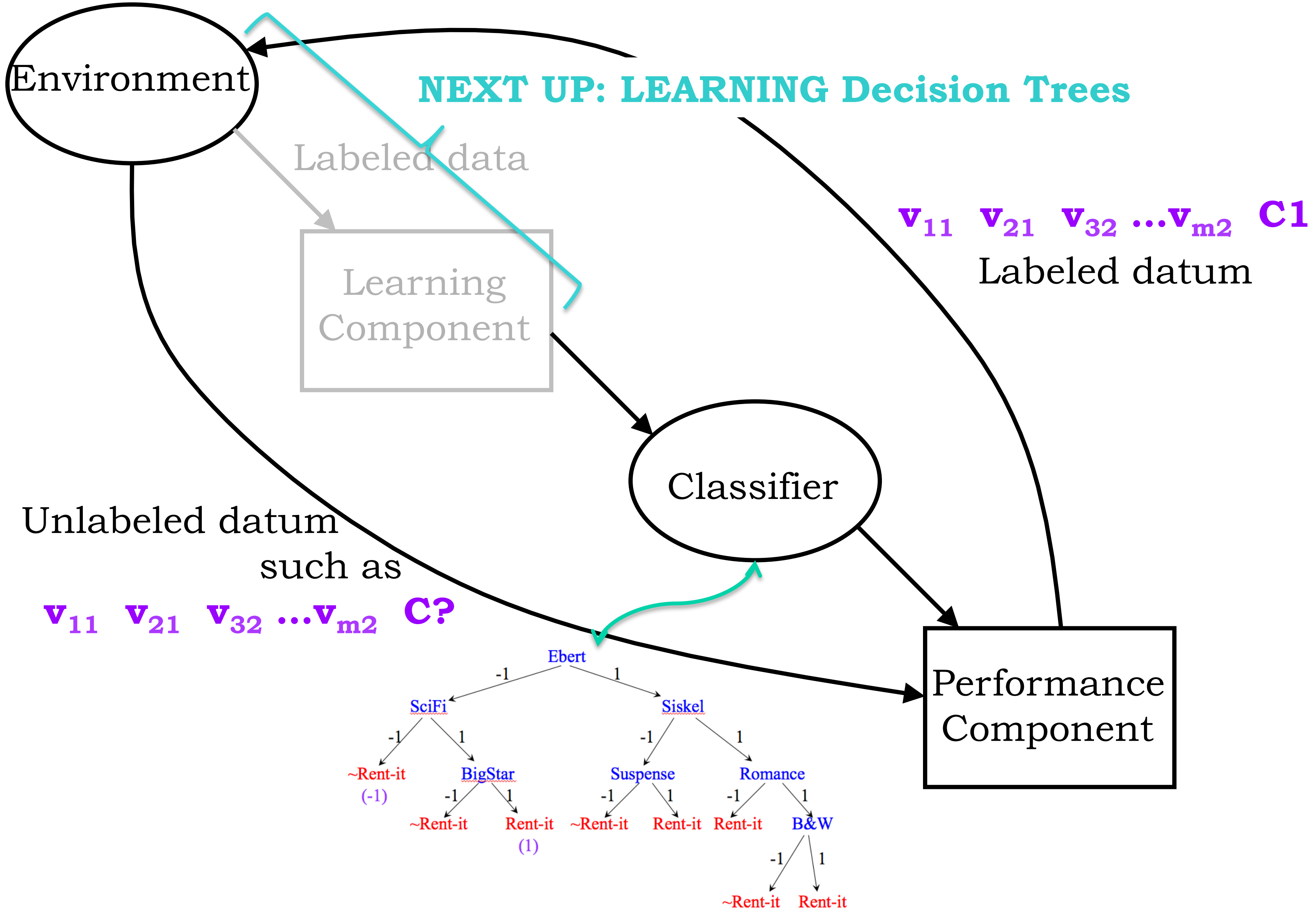
[SciFi = 1, Suspense = 1, Romance = -1, Ebert = -1, Siskel = 1, BigStar = 1, …, Watch-it???]



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

Watch-it





Environment
NEXT UP: LEARNING Decision Trees
Labeled data
Learning Component
v_{11} v_{21} v_{32} ...v_{m2} C1
Labeled datum
Classifier
Unlabeled datum such as
v_{11} v_{21} v_{32} ...v_{m2} C?
Performance Component
Ebert
SciFi
Siskel
~Rent-it (-1)
BigStar
Suspense
Romance
~Rent-it
Rent-it (1)
~Rent-it
Rent-it
Rent-it
B&W
~Rent-it
Rent-it

Decision tree learning (or induction) from data is an instance of *supervised machine learning*.

Decision tree learning is typically regarded as an example of *explainable machine learning*.

Where might decision tree learning be used in addressing climate change and other environmental challenges?

Where might supervised learning (e.g., decision tree learning) be used in addressing climate change (and other environmental challenges)?

From "Tackling Climate Change with Machine Learning" (through page 33)

- forecasting electricity supply and demand (p. 7)
- predicting the properties of different materials (p. 8) and chemicals (p. 28)
- predicting disruptions in nuclear reactors (p. 10)
- predicting remaining battery lifetime (p. 17)
- predicting travel modalities to inform transportation planning (p. 18)

Post page 33

- predicting food shortages and health trending based on pervasive data like cell phone data (p. 44)
- predicting carbon prices with changing tax rates, quotas, tariffs, predicted energy demands (p. 54-55)
- Predicting pipe corrosion to prevent oil and gas leakage (p. 70)

As I go through these slides think about how the following issues, raised in the "Tackling climate Change with Machine Learning"

How might the learning algorithm be adapted to low-data settings?

How might the algorithm effectively use domain-specific knowledge?

Computing and the Environment

Douglas H. Fisher

Machine Learning Week 3

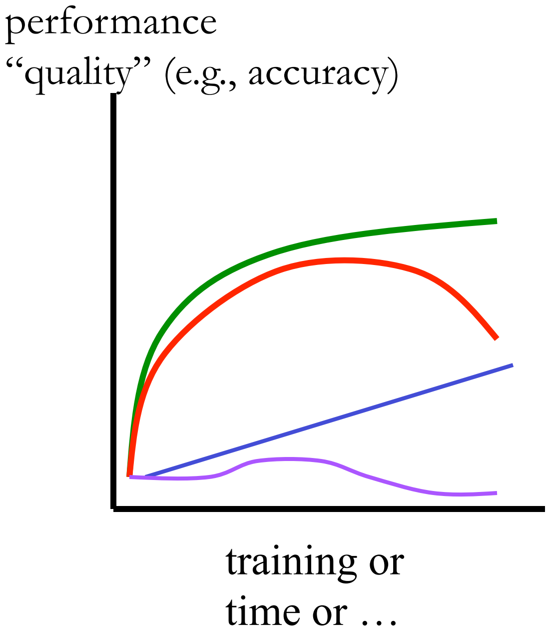
Learning Decision Trees

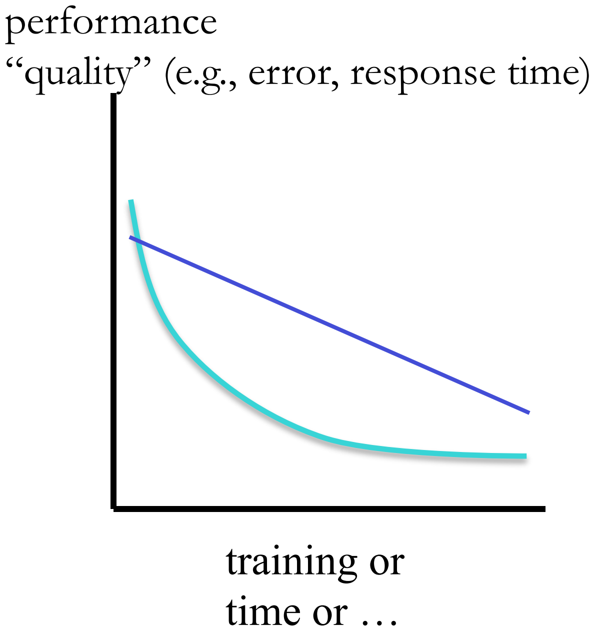
Learning Decision Trees

Two perspectives of Machine Learning:

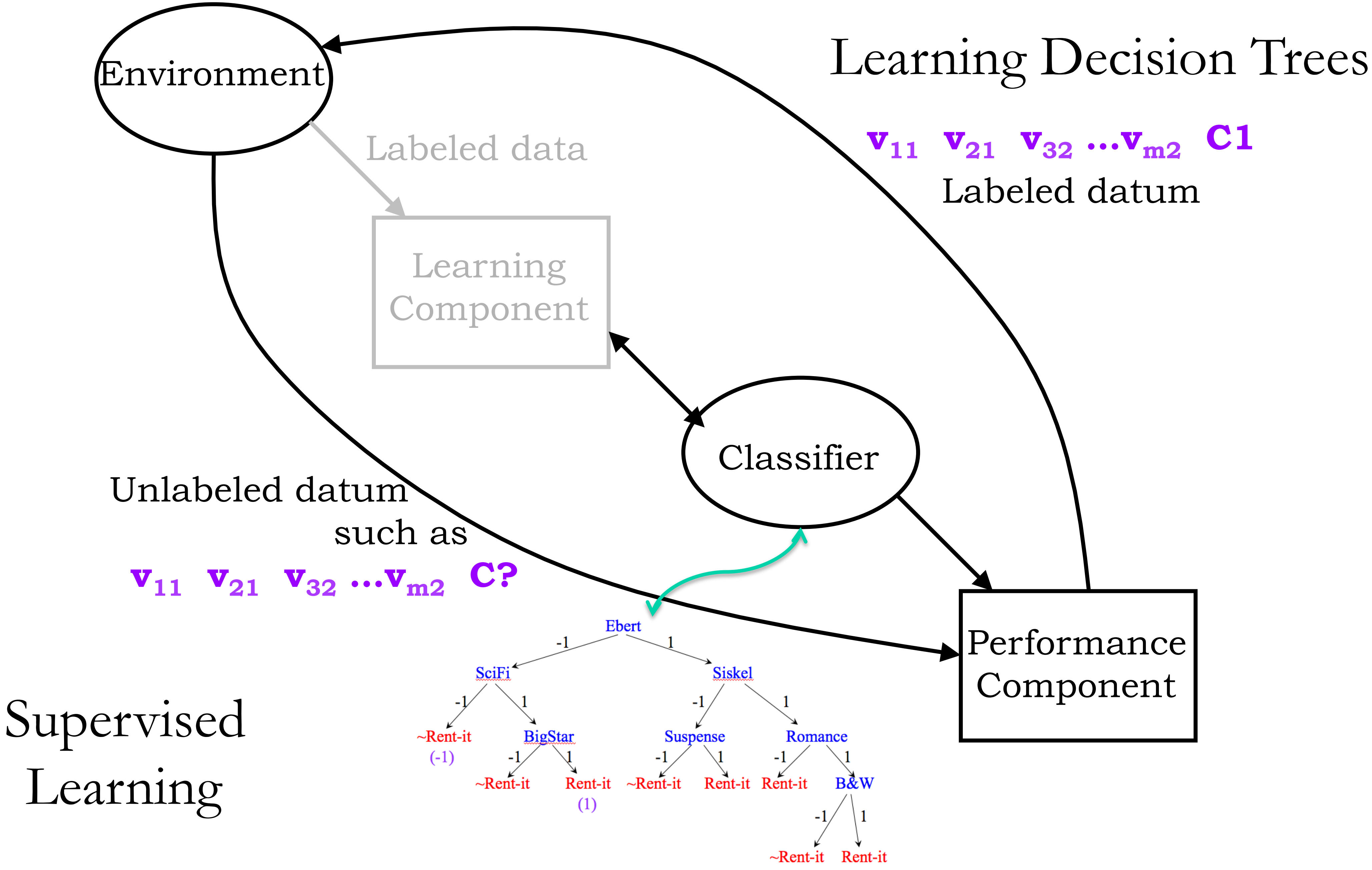
Machine Learning for advanced *data analysis*

Machine Learning for robust artificial *agents*





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



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
Supervised Learning

Top-Down Induction of Decision Trees

The standard greedy (hill-climbing) approach

```
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 v_2

```
TDIDT({d | d in Data and                    TDIDT({d | d in Data and
        Value(BestAttribute, d)                     Value(BestAttribute, d)
              = v1})                                              = v2})
```

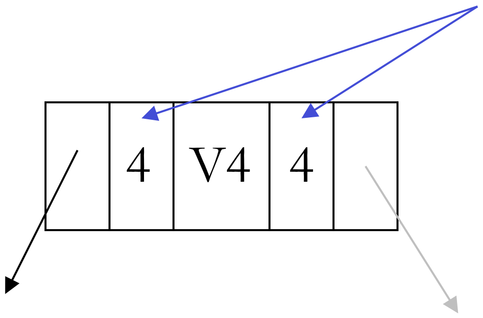
Top-Down Induction of Decision Trees

	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

Training Data Set

Best-attribute: V4

Assume left branch always corresponds to -1



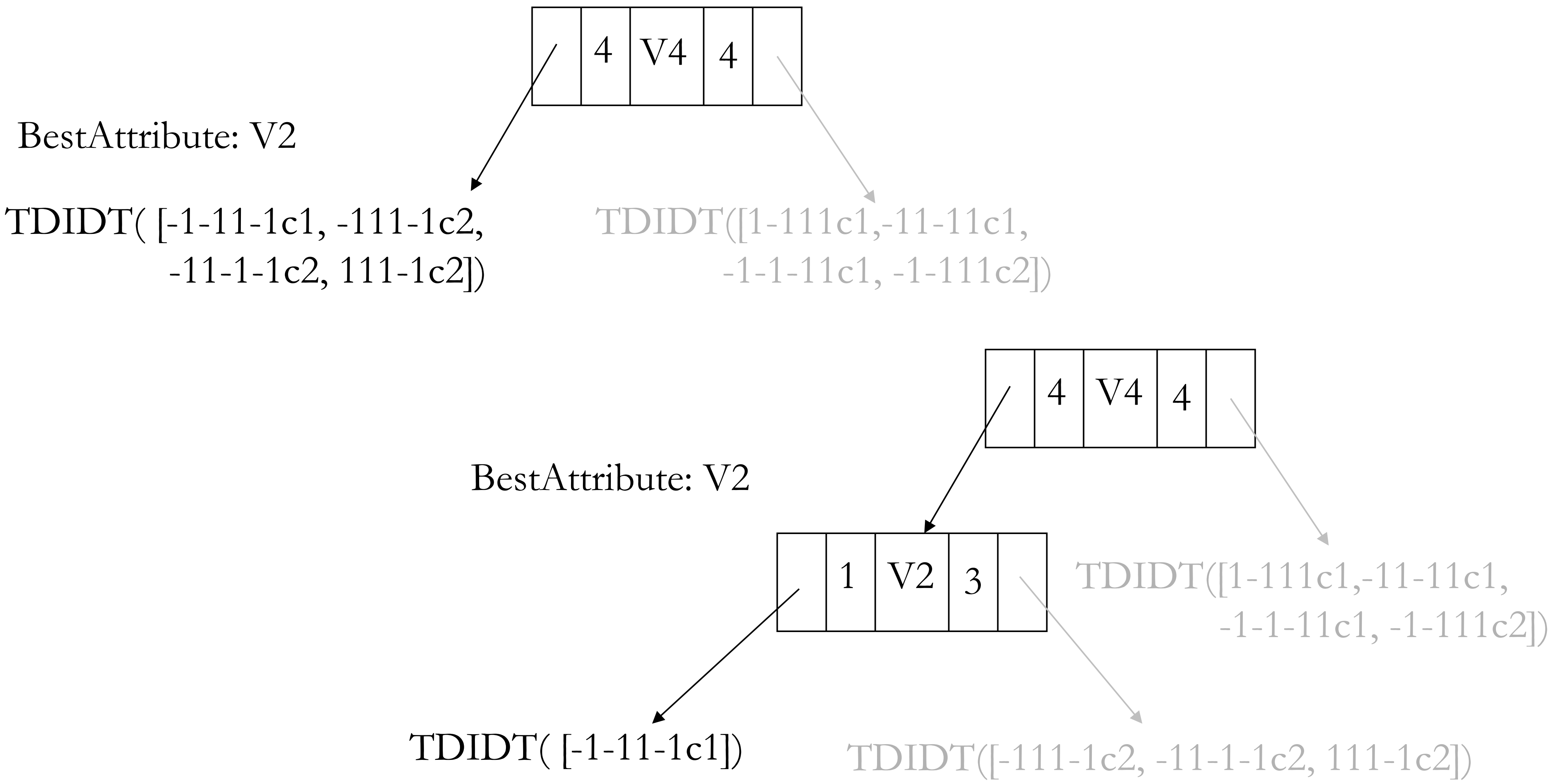
Number of data sent down left and right branches, respectively.

Assume right branch always corresponds to 1

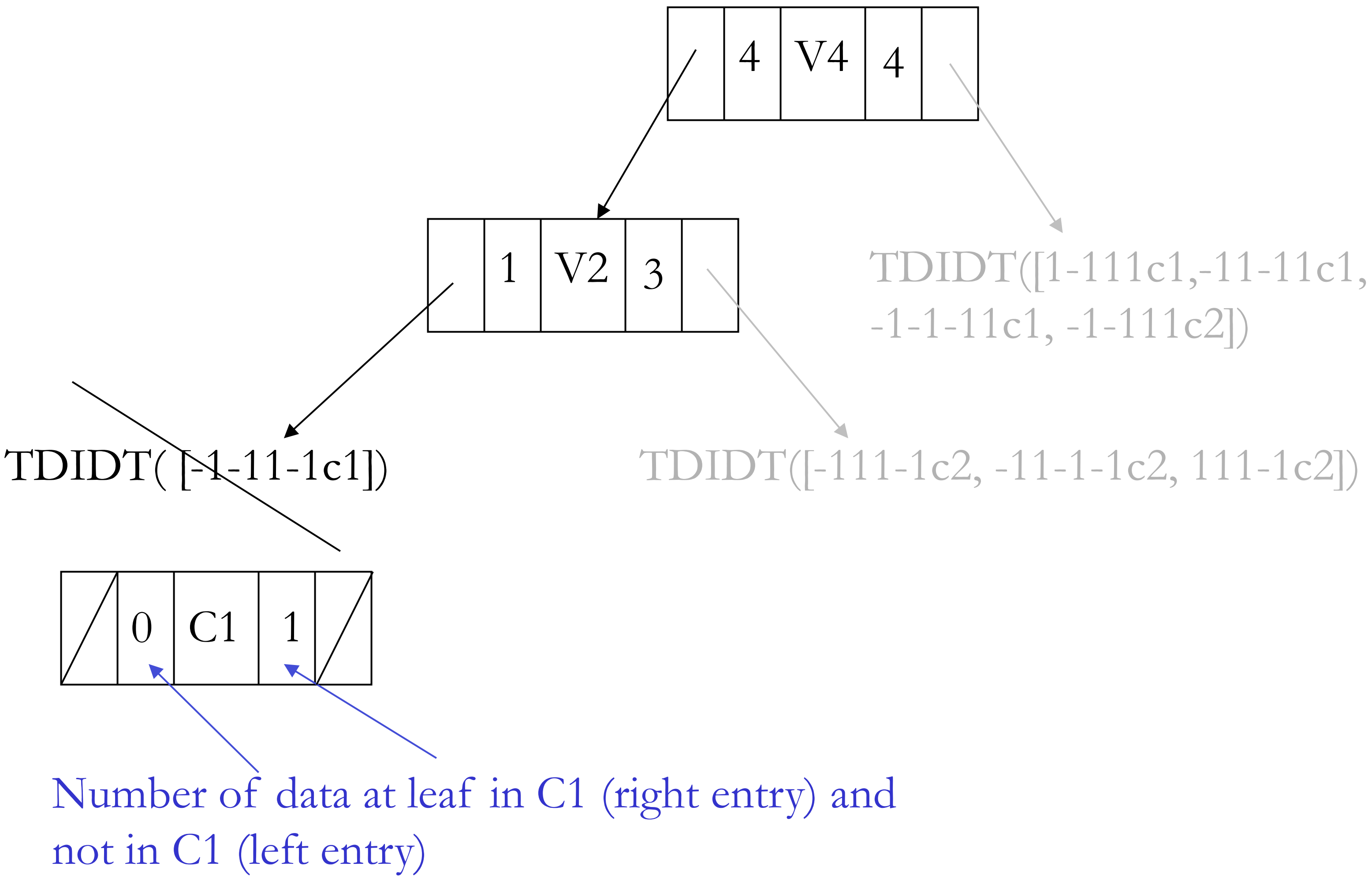
TDIDT([-1-11-1c1, -111-1c2, -11-1-1c2, 111-1c2])

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

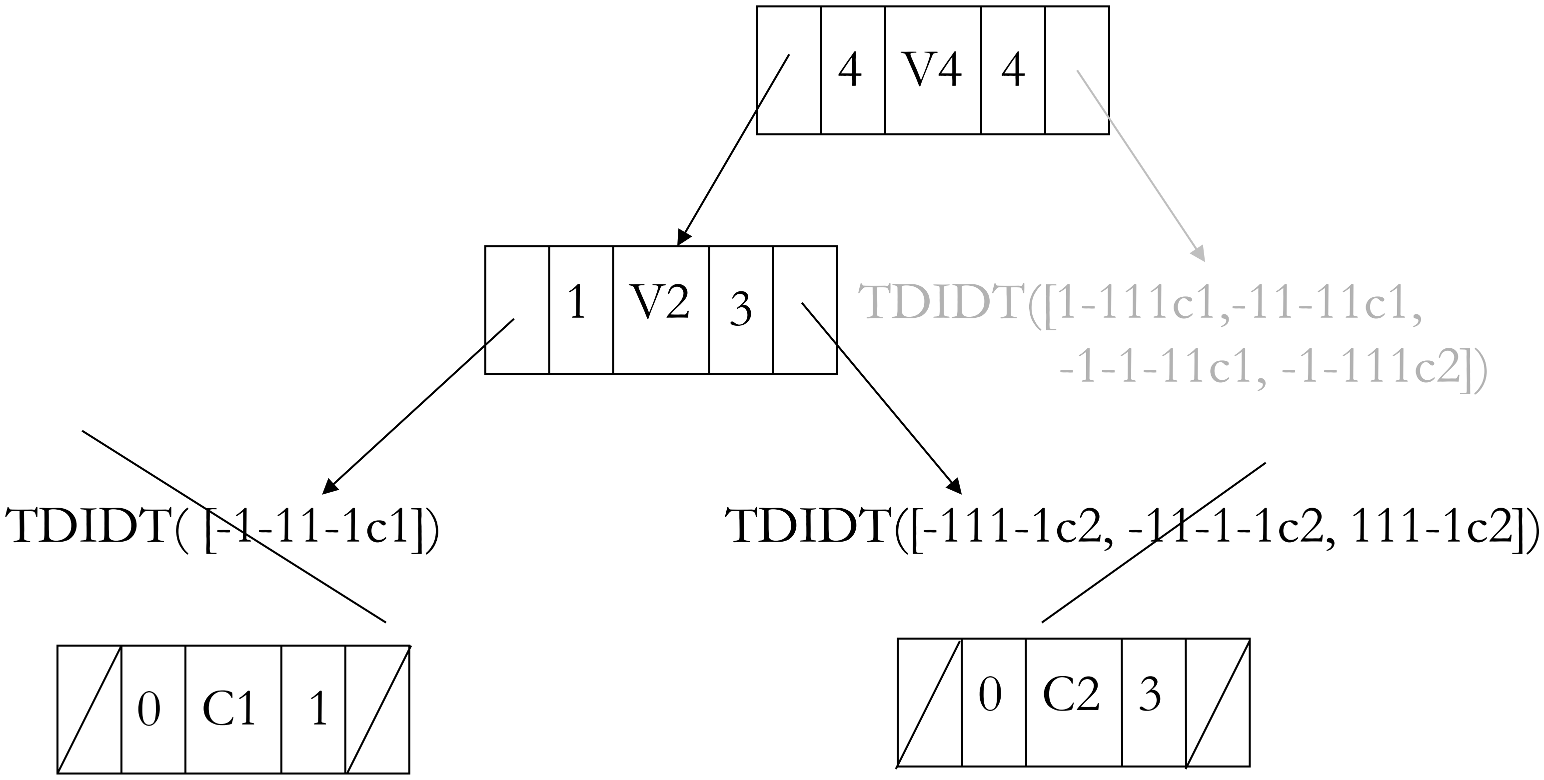
Top-Down Induction of Decision Trees



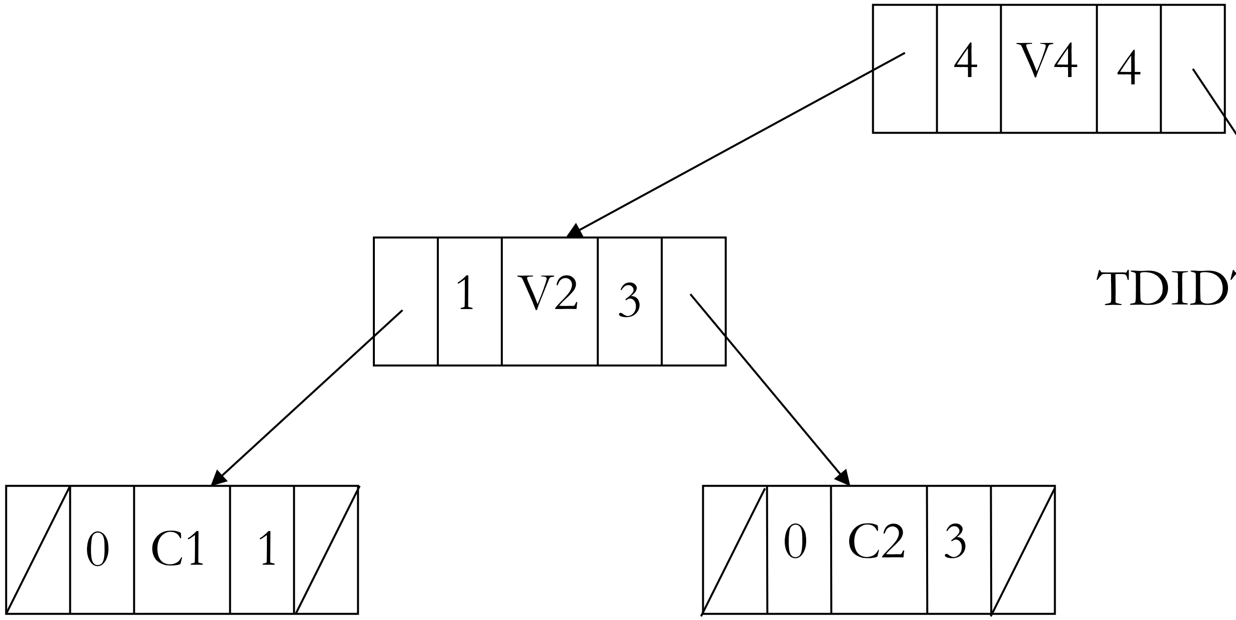
Top-Down Induction of Decision Trees



Top-Down Induction of Decision Trees



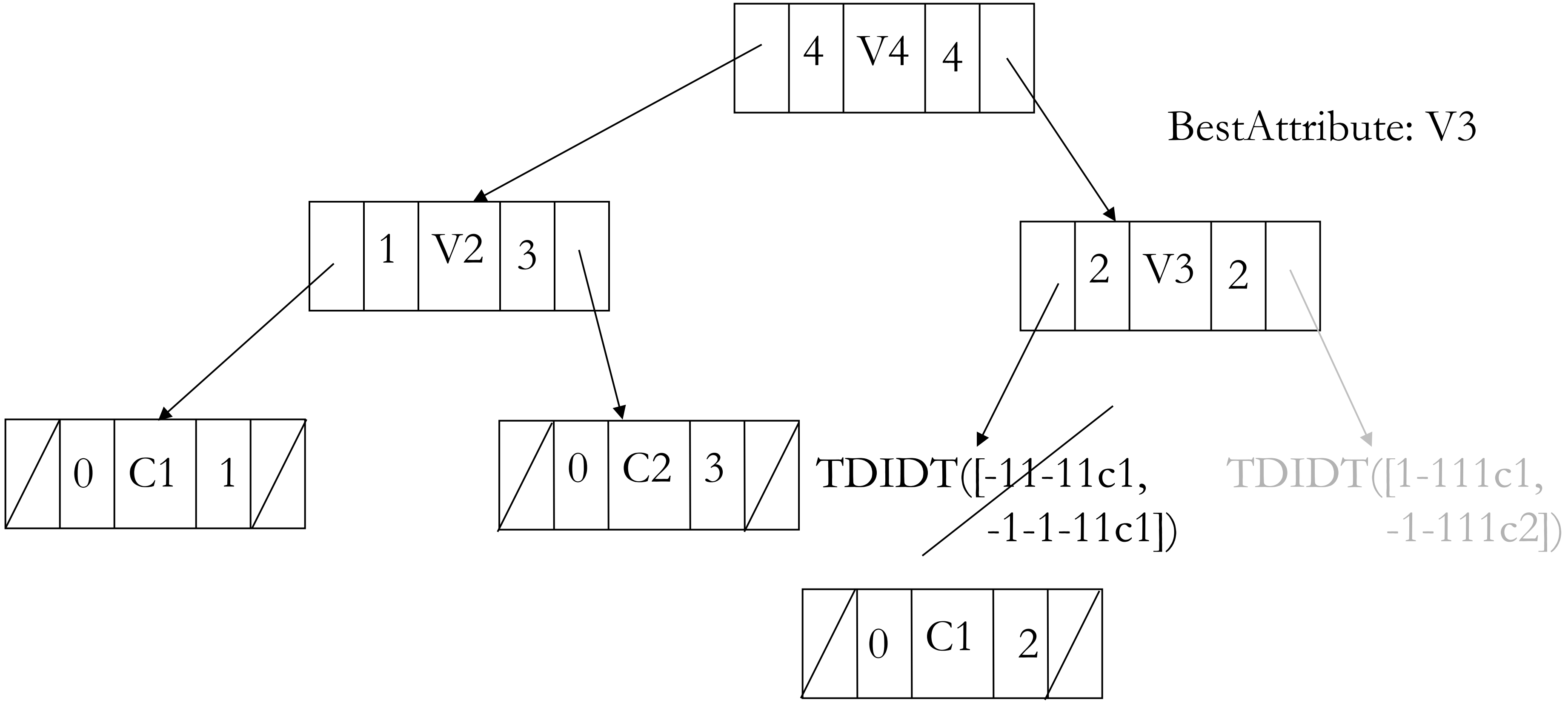
Top-Down Induction of Decision Trees



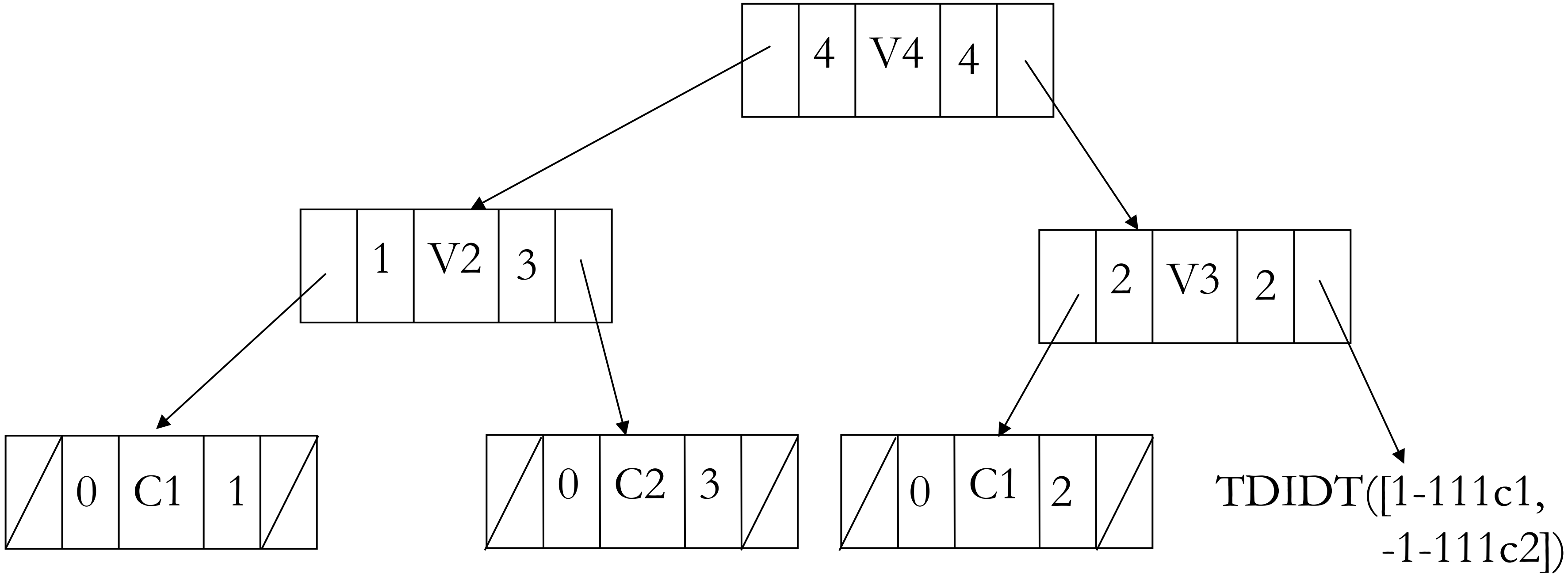
BestAttribute: V3

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

Top-Down Induction of Decision Trees

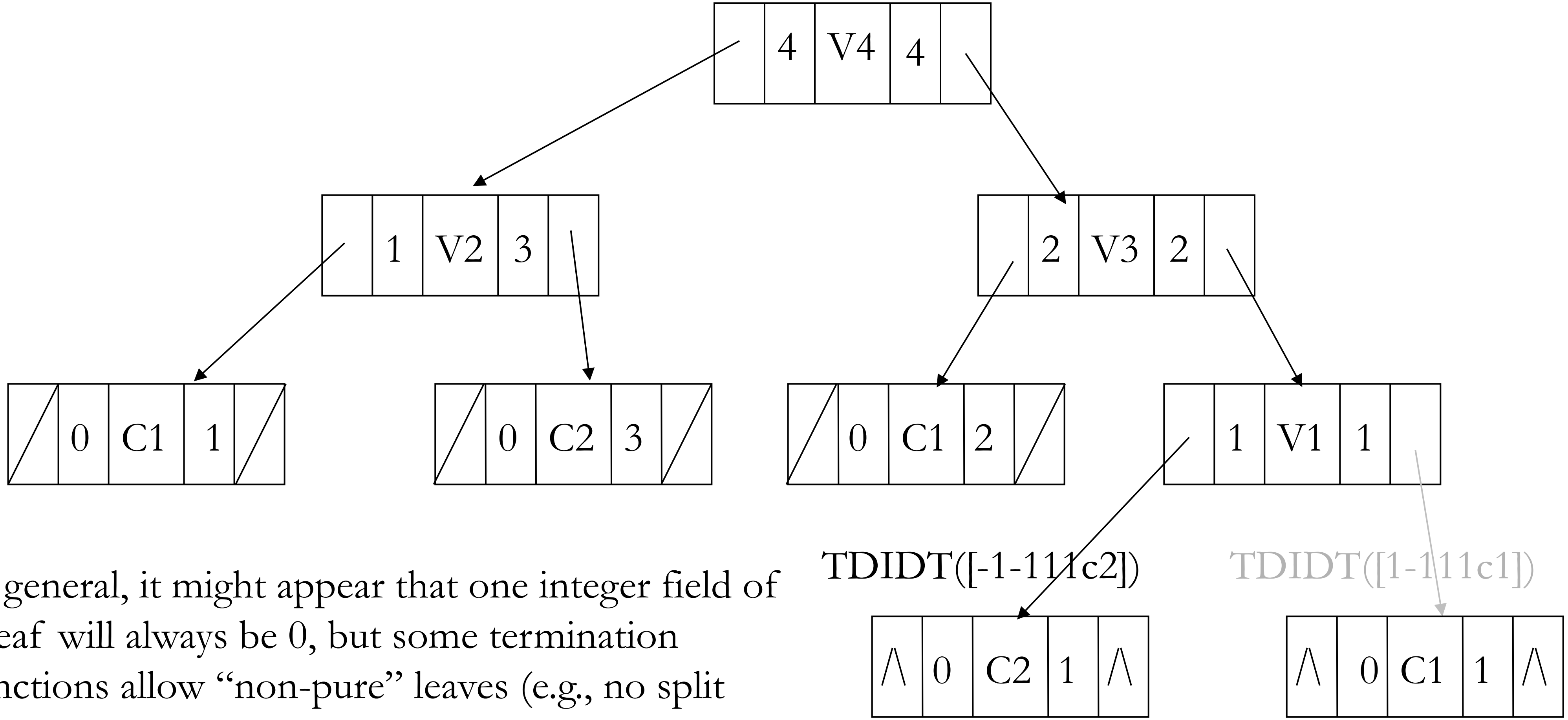


Top-Down Induction of Decision Trees



BestAttribute: V1

Top-Down Induction of Decision Trees



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*).

Computing and the Environment

Douglas H. Fisher

Machine Learning Week 3

Decision Trees as Search

Selecting the best divisive attribute (SelectFN)

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

Selecting the best divisive attribute (SelectFN):

Attribute V_i that maximizes:

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

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

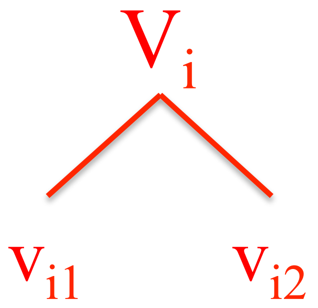
Selecting the best divisive attribute (SelectFN):

Attribute V_i that minimizes:

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

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

How might domain-specific knowledge be used in this process of selecting a best attribute?



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

How might domain-specific knowledge be used in this process of making a decision?

In the movie recommendation domain it might be something like:

- IF person p is considering movie m which is playing at an art house h THEN bump its weight upwards (there may be data on another town that p frequents, but domain knowledge includes a category of art houses that may allow “sharing” of data across theaters.

In sustainability domains (e.g., p. 7 of “Tackling Climate Change with Machine Learning)

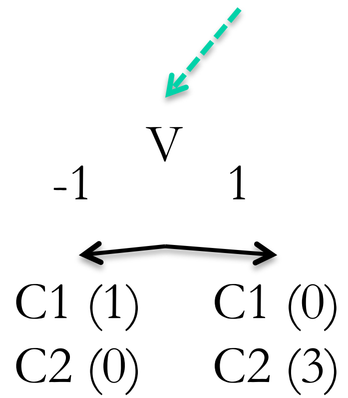
- For purposes of electricity demand forecasting, use a diffy Q based weather model (domain knowledge)to project weather well in advance, and use historical data on the how far apart the weather model (5 days in advance) was from reality to adjust the weather projection.

In a weather domain (e.g., p. 7 of “Tackling Climate Change with Machine Learning)

- When building a decision tree for projecting electricity demand 10 years from now, use a regional climate model (domain knowledge) to adjust historic yearly data to revise relevance of selected attributes

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

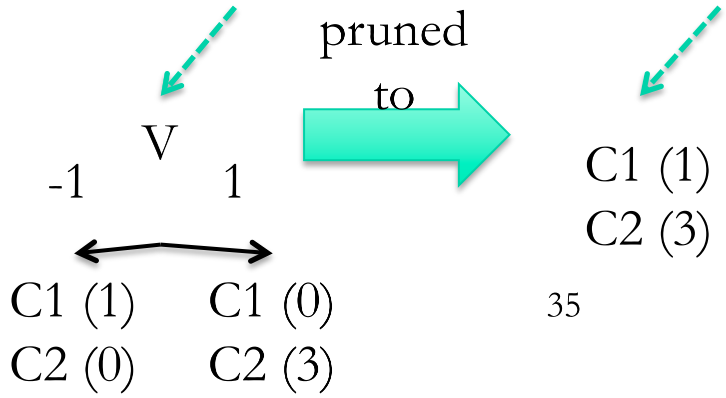
iv) of the 0.5 * N items that went to the left leaf during usage, 25% were C1 and 75% were C2

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

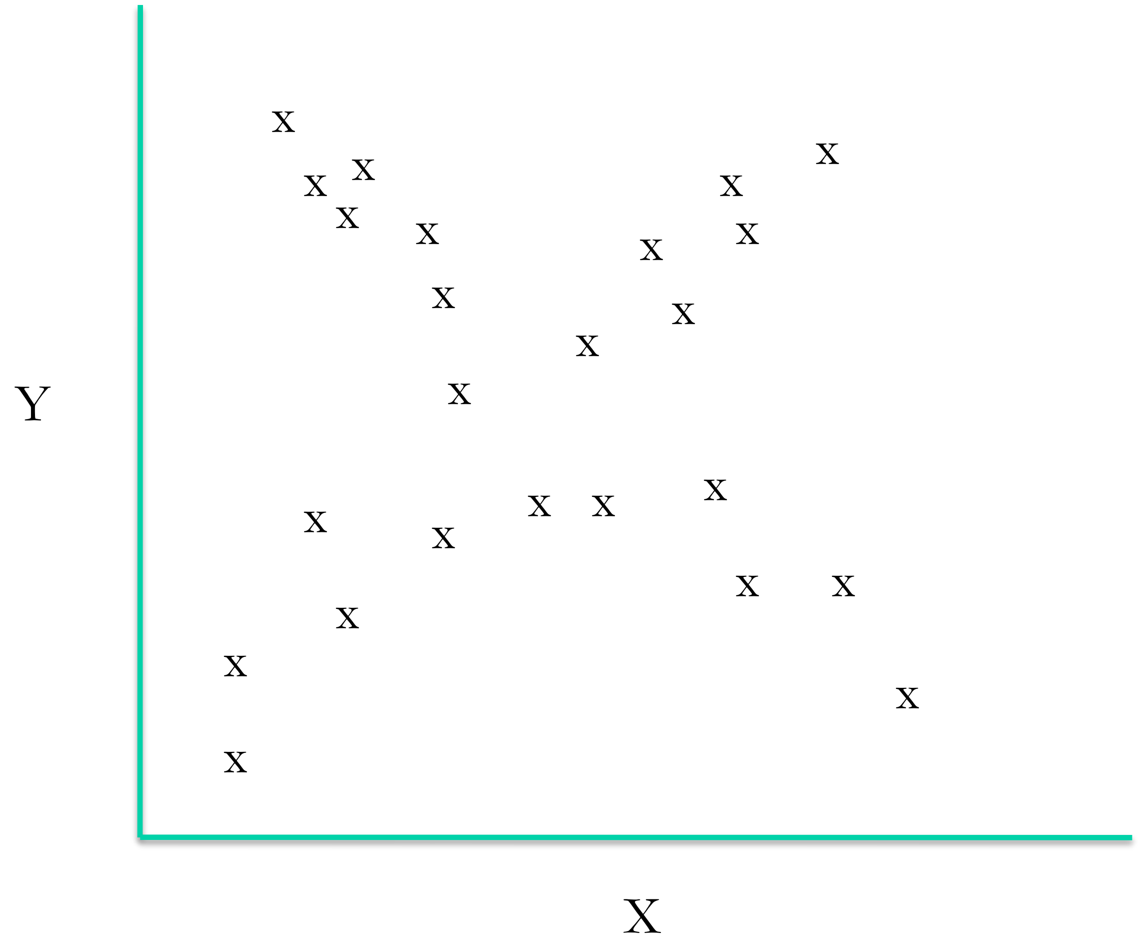
Predicted Counts (if Immigration and Party independent)

	Yes	No
Republican	92	134
Democrat	85	124

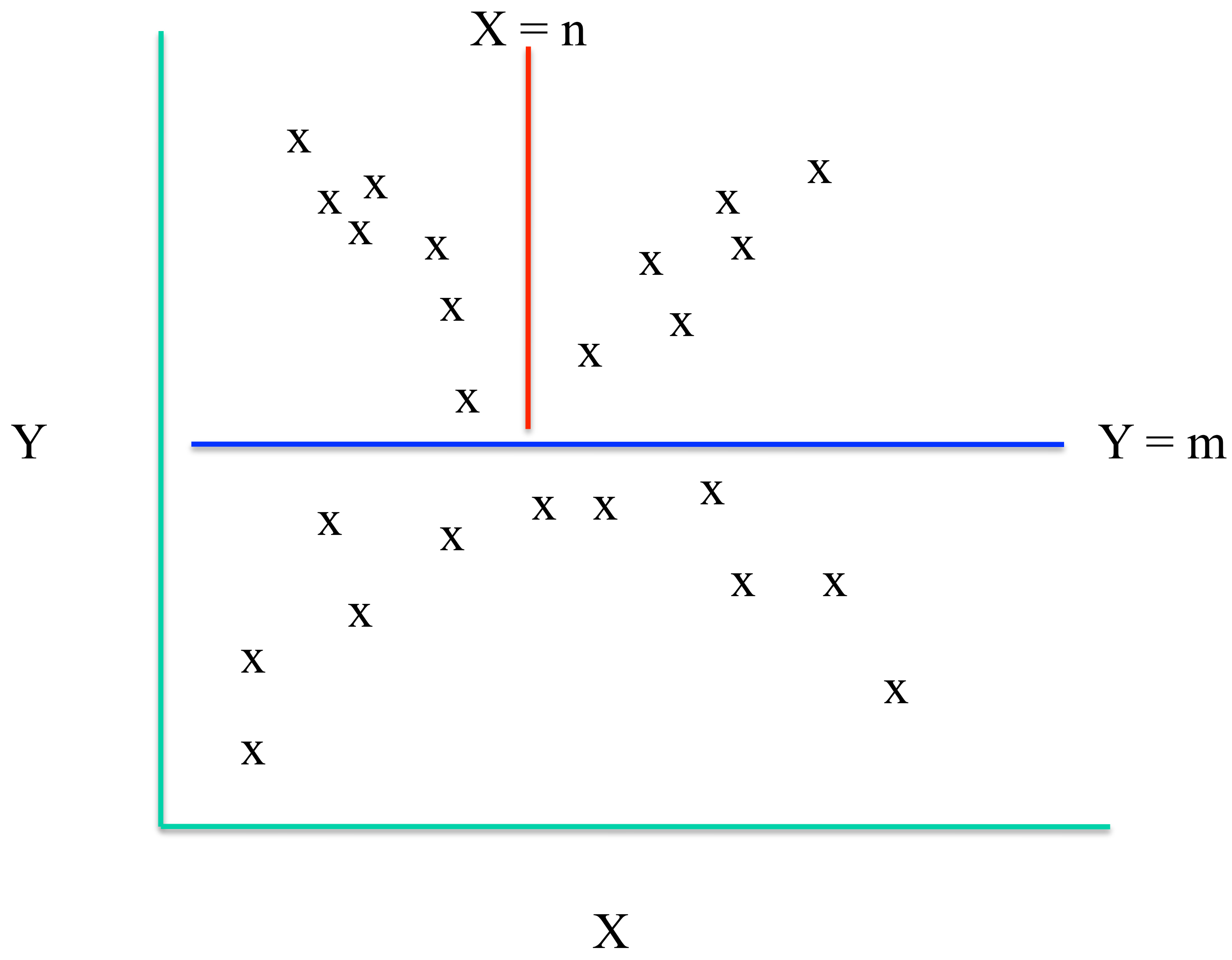
Very different distributions – conclude **dependent**

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

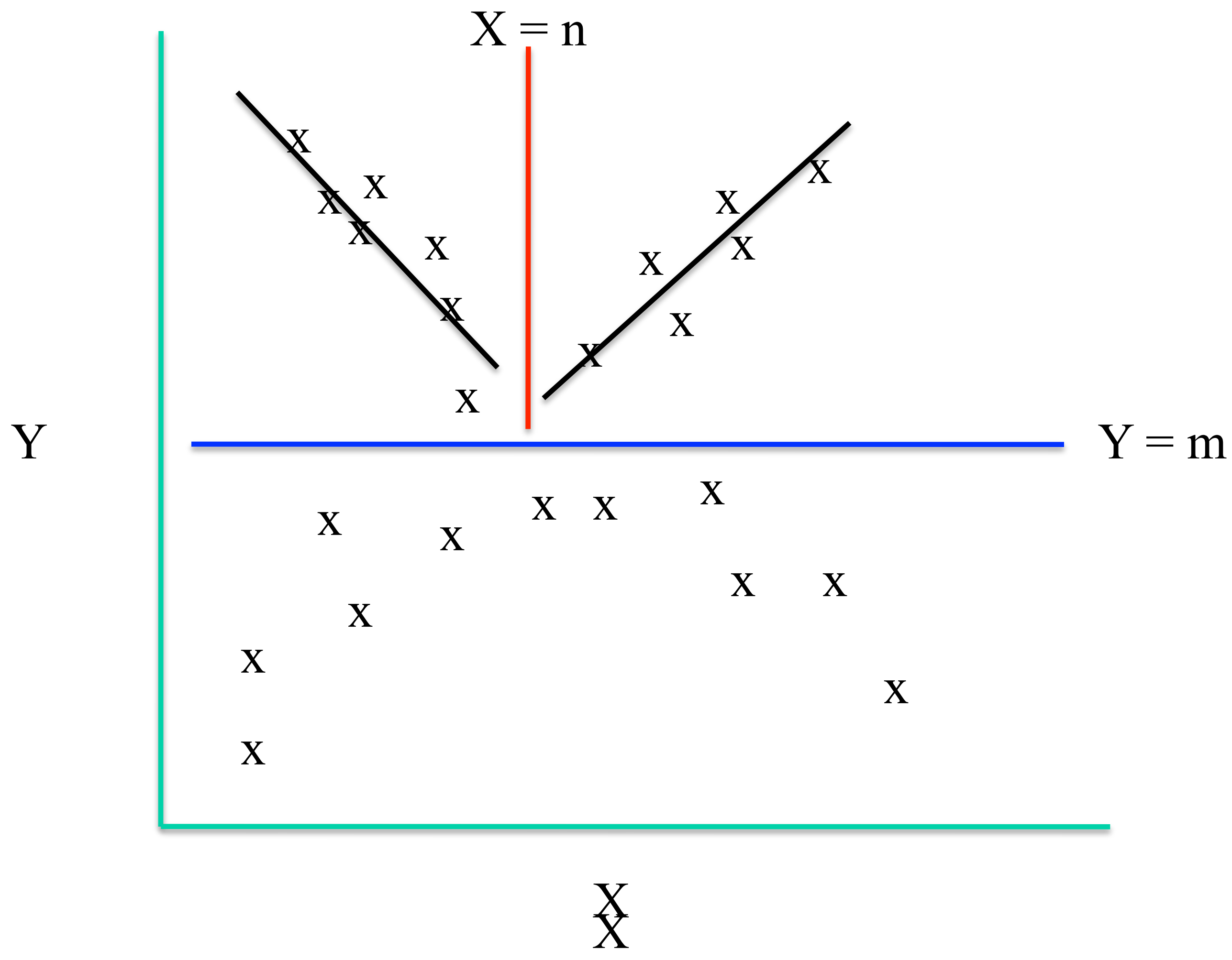
Decomposition and search are important principles in machine learning



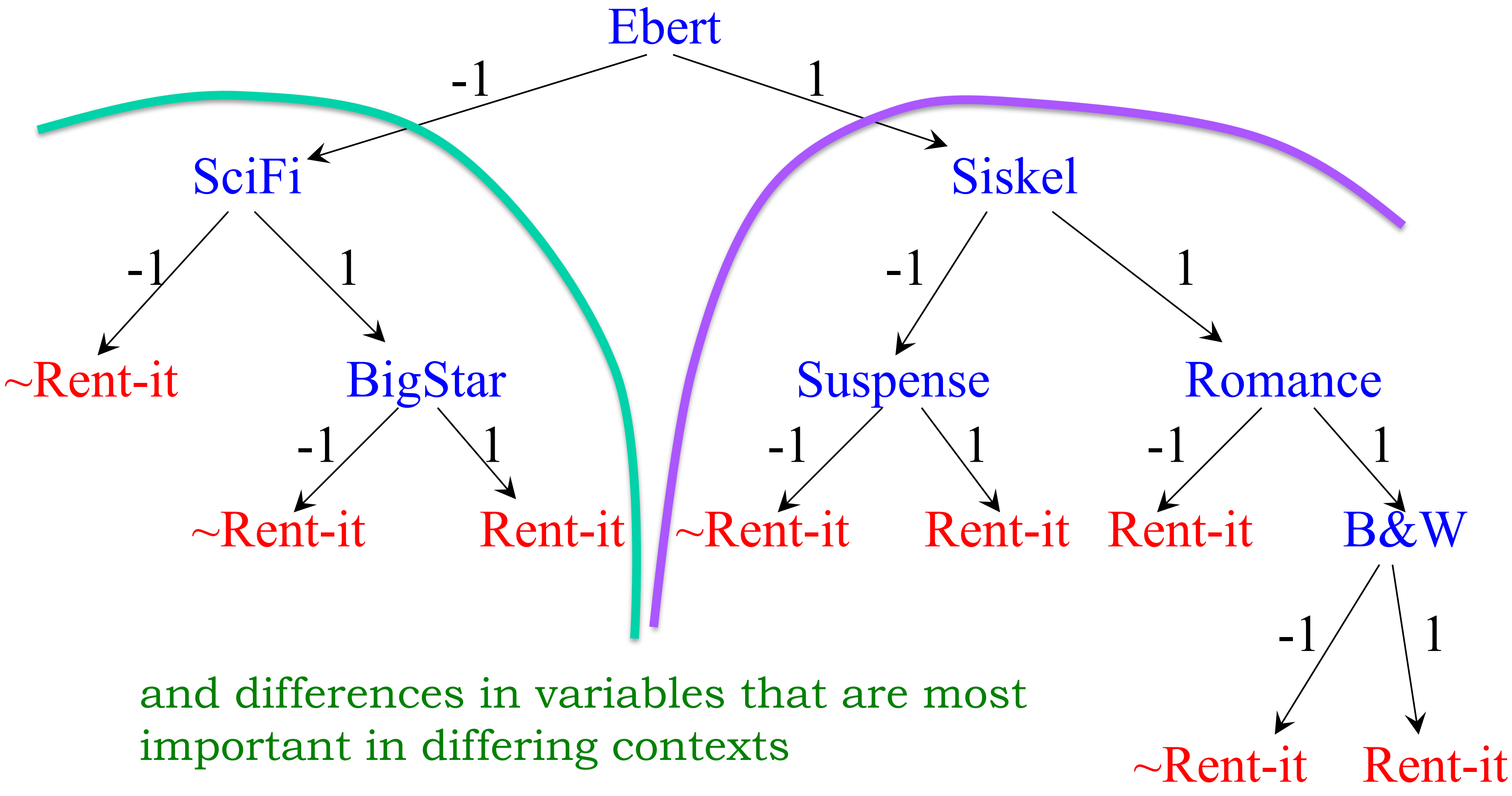
Decomposition and search are important principles in machine learning



Decomposition and search are important principles in machine learning



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

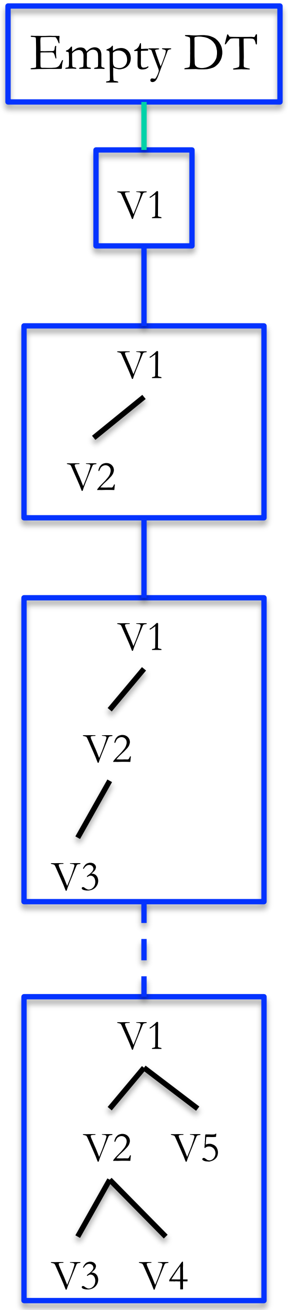
Issues, variations, optimizations, etc:

- 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)**

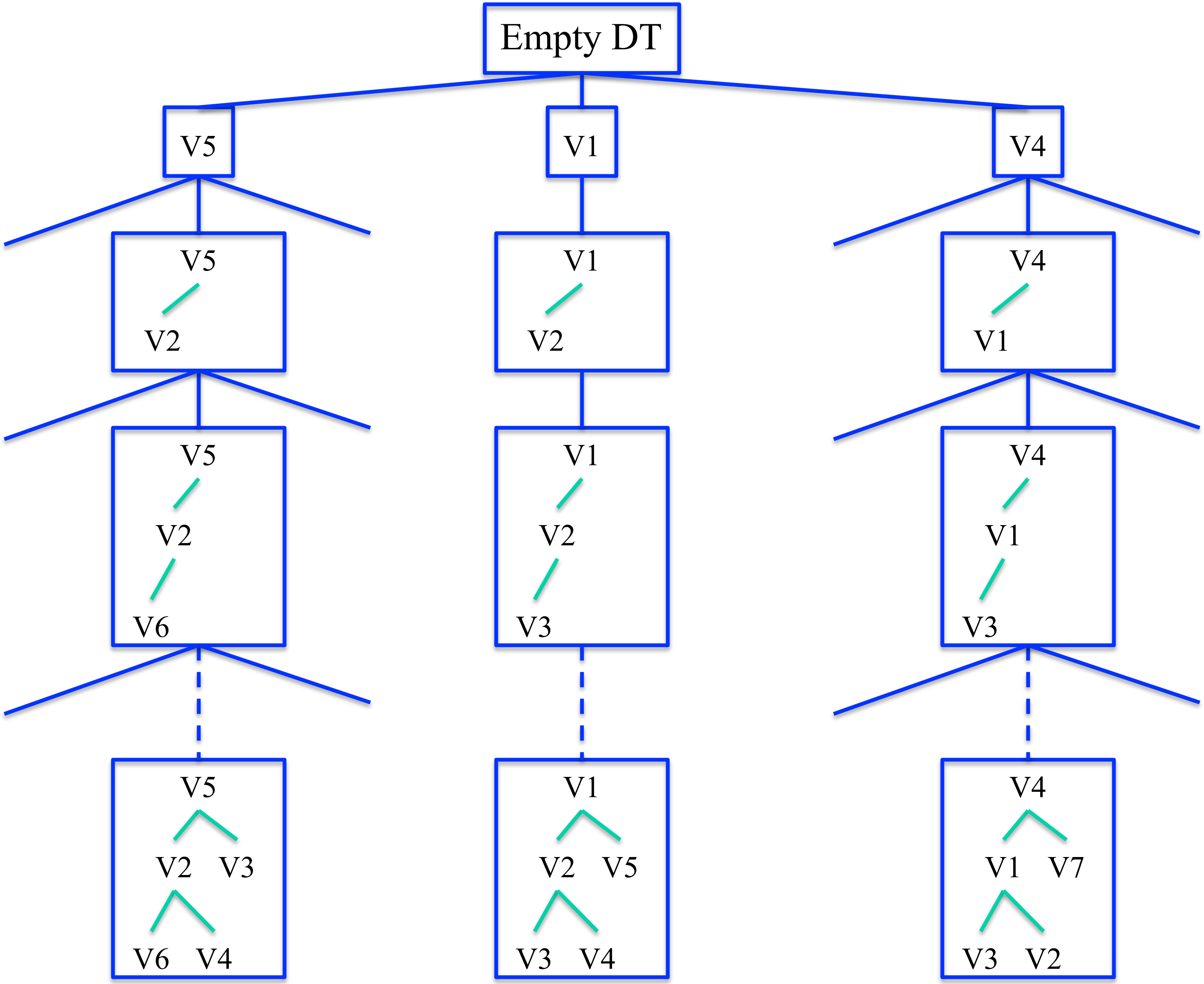
Carlisle, Falcone, Wolock, Meador, Norris (2010) "Predicting the Natural Flow Regime: Models for Assessing Hydrological Alteration in Streams" River Research and Applications, 26, 118-136 (https://my.vanderbilt.edu/csx892/files/2016/06/PredictingLowFlow.pdf)

random forests

The top-down greedy method is essentially a “hill climb”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


Empty DT
V5
V1
V4
V5
V2
V1
V2
V4
V1
V5
V2
V6
V1
V2
V3
V4
V1
V3
V5
V2 V3
V6 V4
V1
V2 V5
V3 V4
V4
V1 V7
V3 V2

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

}

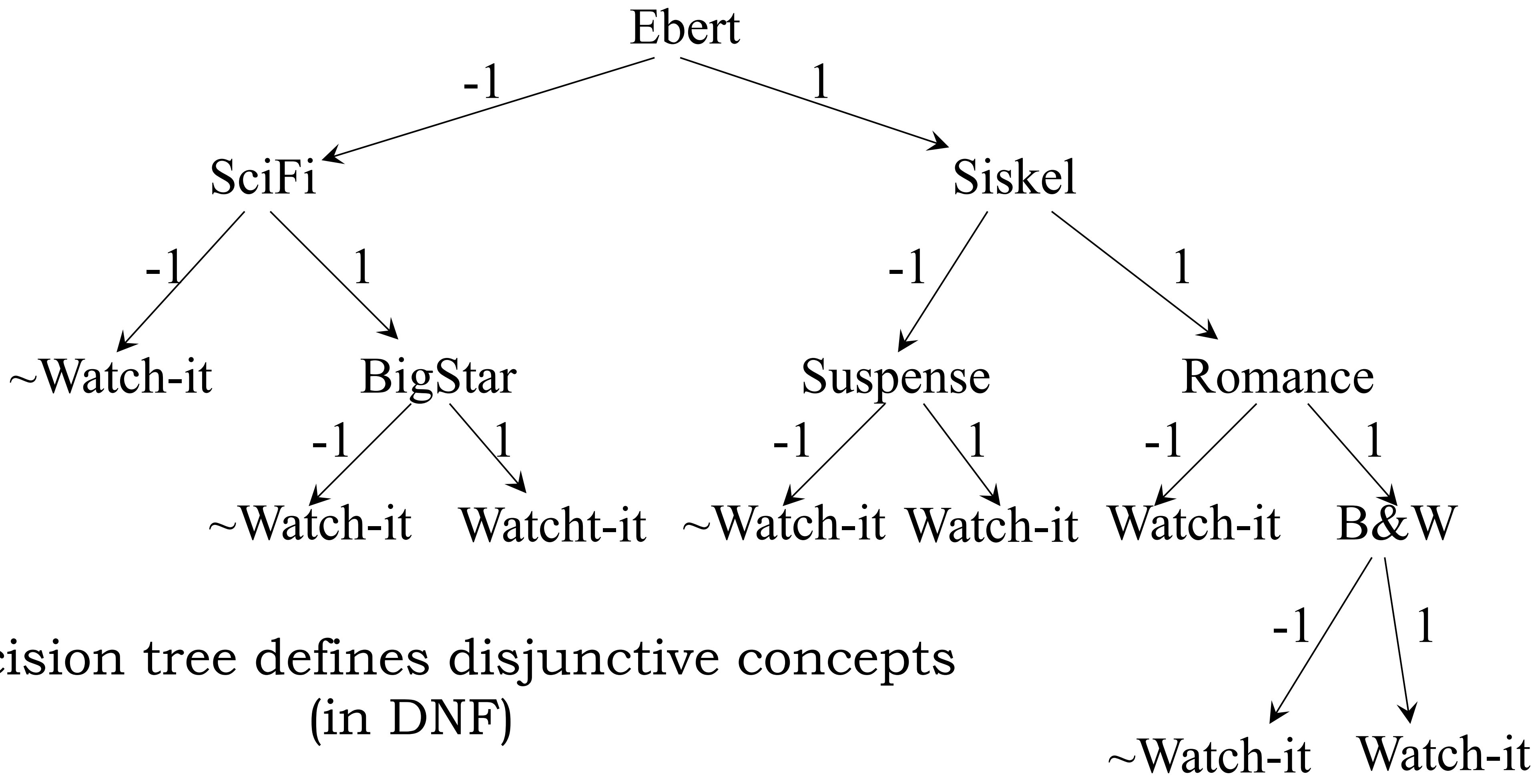
To use the forest classifier, run a test datum through each tree of the forest and take a vote on its classification

CS 5260
Introduction to Artificial Intelligence
Vanderbilt University

Douglas H. Fisher

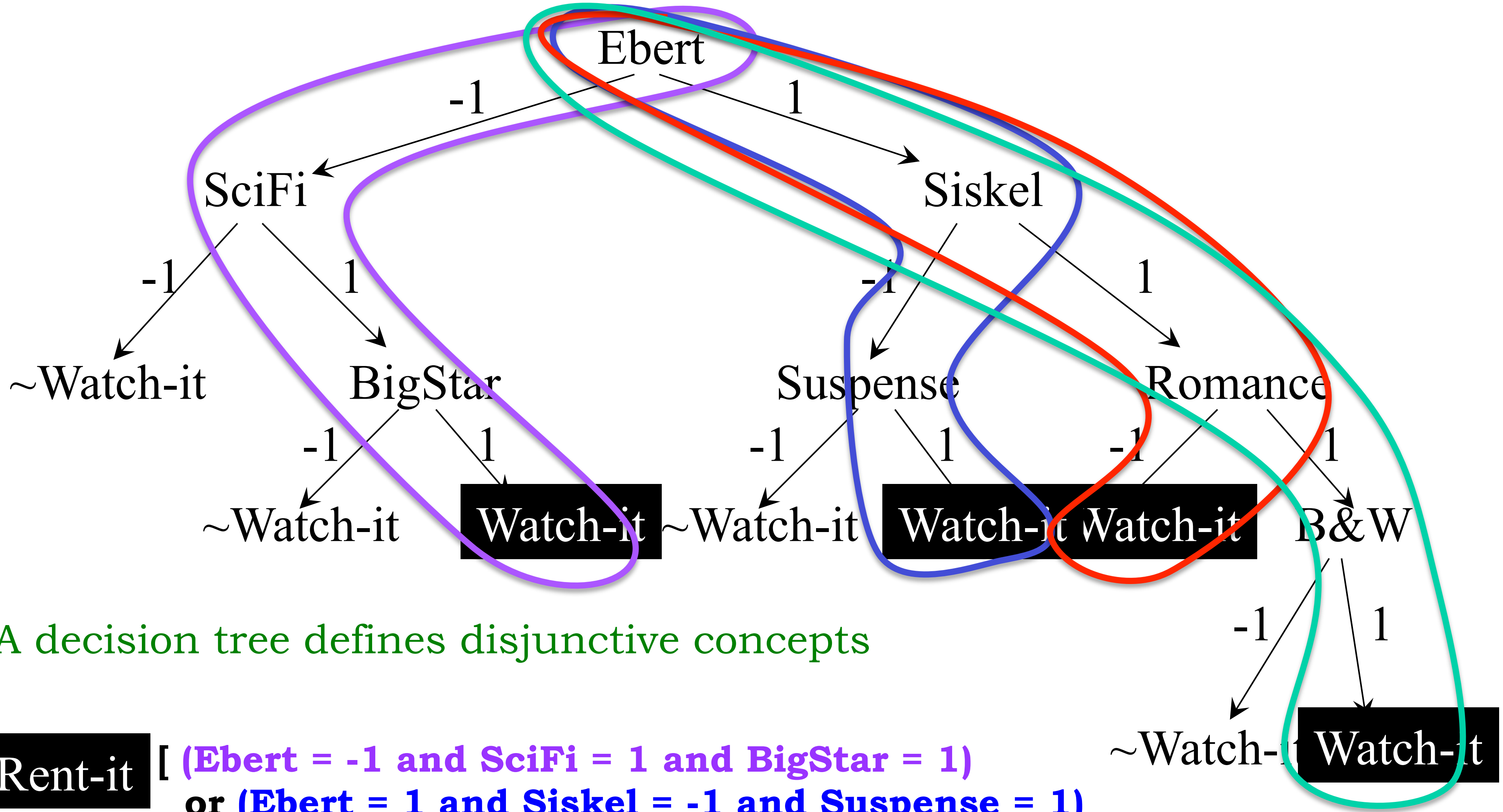
Week 7
Machine Learning from Examples

Properties of Decision Trees



A decision tree defines disjunctive concepts
(in DNF)

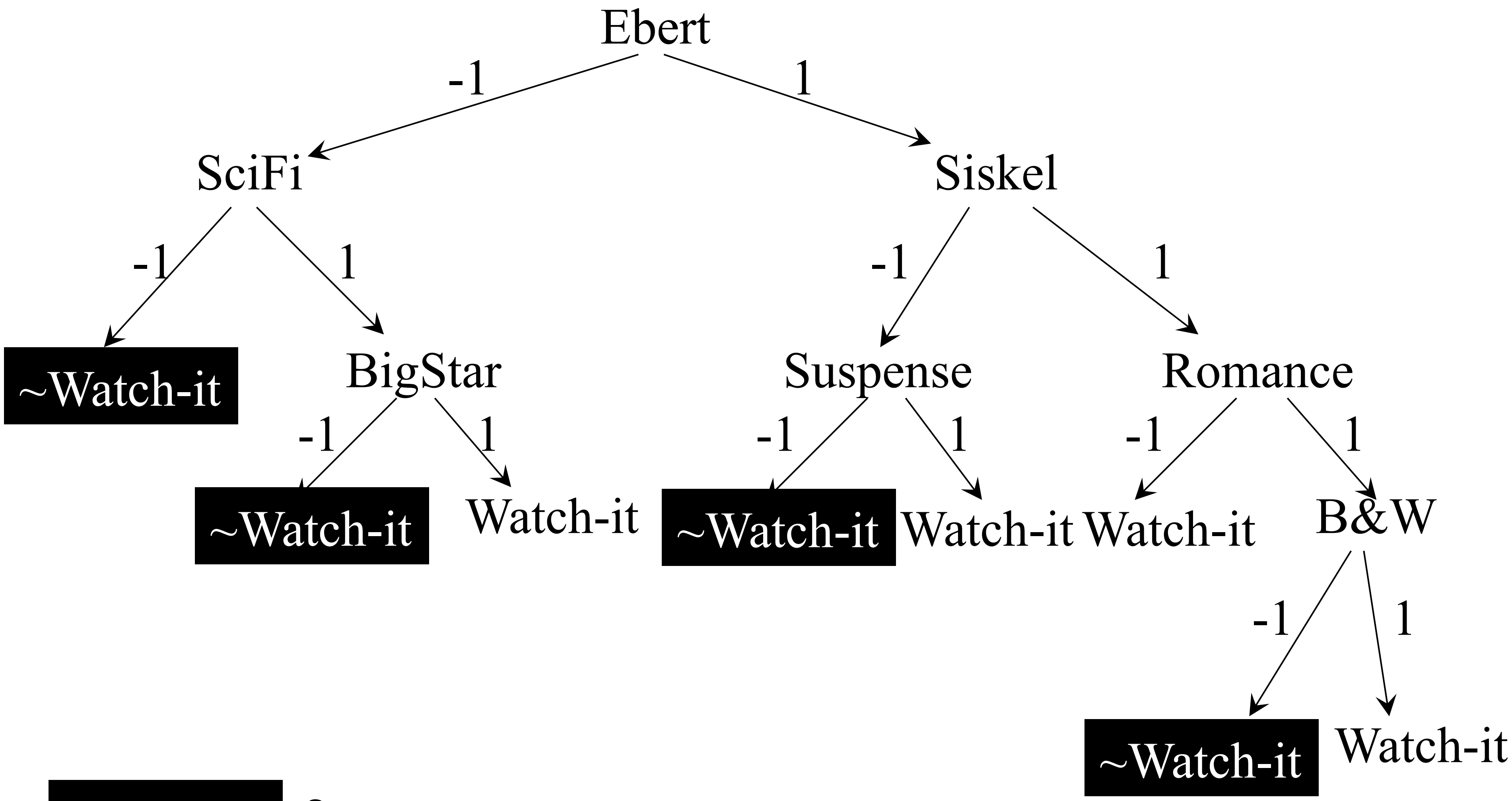
Each path of a decision tree represents a conjunction of values



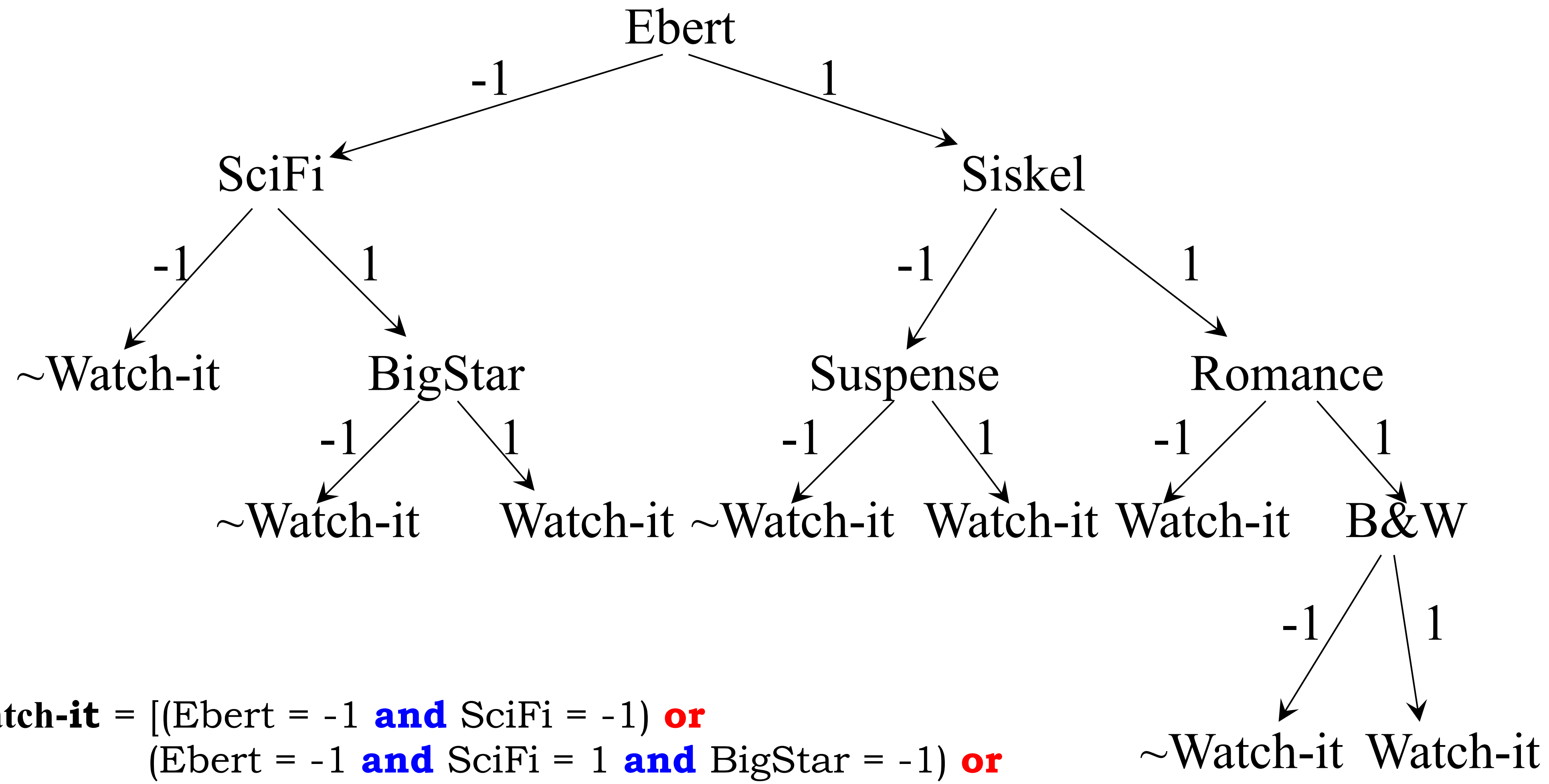
A decision tree defines disjunctive concepts

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)]

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



~Watch-it = ?



~Watch-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)
]

Watch-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 $\wedge$ and 'or' as $\vee$

Watch-it = [(~ebert $\wedge$ scifi $\wedge$ bigstar)
$\vee$ (ebert $\wedge$ ~siskel $\wedge$ suspense)
$\vee$ (ebert $\wedge$ siskel $\wedge$ ~romance)
$\vee$ (ebert $\wedge$ siskel $\wedge$ romance $\wedge$ b&w)]

~Watch-it = [(~ebert $\wedge$ ~scifi)
$\vee$ (~ebert $\wedge$ sciFi $\wedge$ ~bigstar)
$\vee$ (ebert $\wedge$ ~siskel $\wedge$ ~suspense)
$\vee$ (ebert $\wedge$ siskel $\wedge$ romance $\wedge$ ~b&w)]

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

Show that

$$\sim[\,(\sim ebert \wedge scifi \wedge bigstar) \vee (ebert \wedge \sim siskel \wedge suspense) \vee (ebert \wedge siskel \wedge \sim romance) \vee (ebert \wedge siskel \wedge romance \wedge b\&w)\,]$$

$=$

$$[\,(\sim ebert \wedge \sim scifi) \vee (\sim ebert \wedge scifi \wedge \sim bigstar) \vee (ebert \wedge \sim siskel \wedge \sim suspense) \vee (ebert \wedge siskel \wedge romance \wedge \sim b\&w)\,]$$

Computing and the Environment

Douglas H. Fisher

Machine Learning Week 3

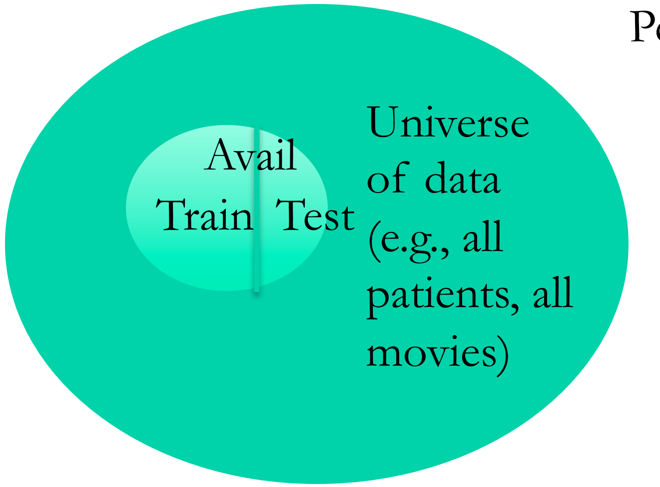
Model Selection and Evaluation

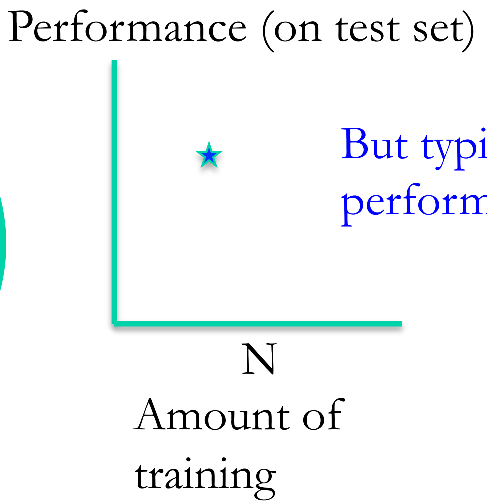
Model Selection and Evaluation

- Testing a classifier/predictor on data that was used for training is overly optimistic,
  - even if the method doesn't memorize each data per se

- More realistic, in most cases, is to test on previously unseen data

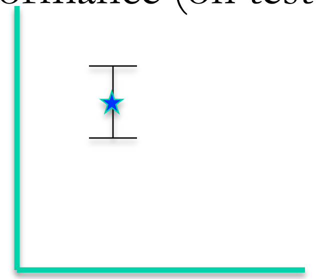
- If there are N training data, then test set accuracy (or error) approximates (to an unknown extent) the performance of classifiers constructed by the learning method on N training data





But typically, most interested in performance on universe

Model Selection and Evaluation



Given: M data available, Avail
Learning Trials, L
Training set size, N
Test set size, M-N

Local: Training Set, Train
Test Set, Test
Classifier
AggregatedPerformance (e.g., Mean, Median, Mode)

Initialize AggregatedPerformance
Repeat L times {
    Train ← Randomly draw N data from Avail, "without replacement"
    Test ← Avail – Train
    Classifier ← Learn(Train)
    AggregatedPerformance ← Performance(Classifier, Test) + AggregatedPerformance
}
Return AggregatedPerformance

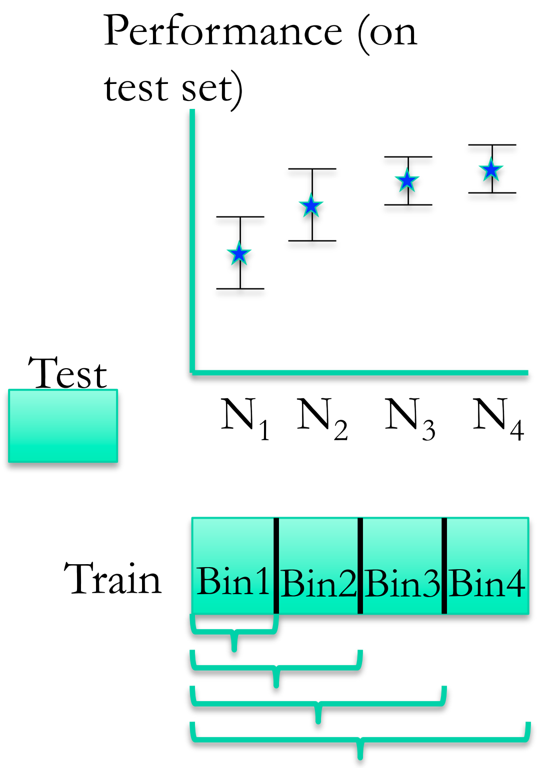
This provides approximation of performance (on Universe) of learning method at training size of N

Generating Learning Curves through repeated Train and Test splits

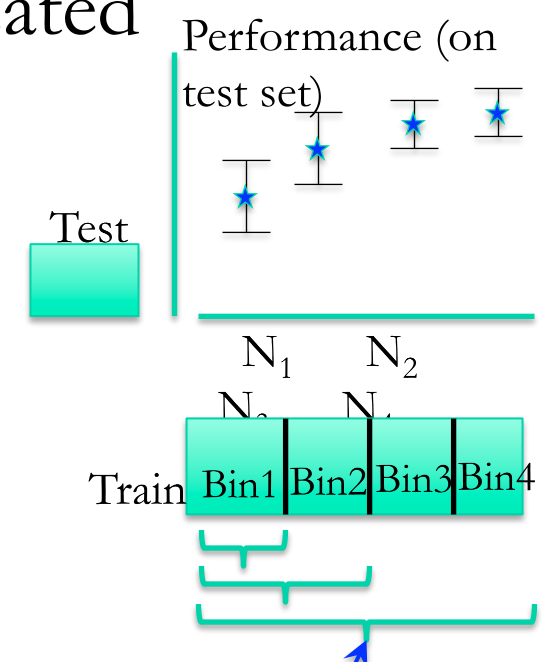
Generating Learning Curves through repeated Train and Test splits

Given: M data available, Avail
Learning Trials, L
Training set sizes, $N_1 \ldots N_{max}$
Test set size, $M-N_{max}$

Local: Training Set, Train
Test Set, Test
Classifier
AggregatedPerformanceVector

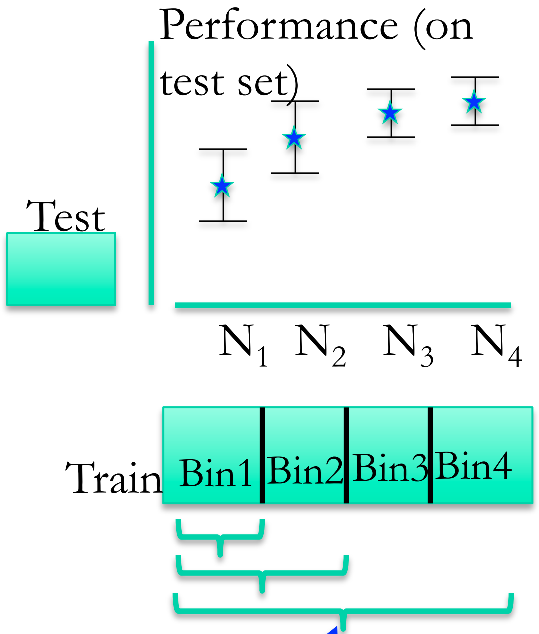


Generating Learning Curves through repeated Train and Test splits



Initialize AggregatedPerformanceVector
Repeat L times {
    Train ← Randomly draw N_{max} data from Avail, "without replacement"
    Test ← Avail – Train
    Partition Train into 1 to max bins, $TrainBin_1$ through $TrainBin_{max}$
    For k = 1 to max {
        Classifier ← Learn(Union of $TrainBin_1$ through $TrainBin_k$)
        AggregatedPerformanceVector[k]
            ← Performance(Classifier, Test) + AggregatedPerformanceVector[k]
}
Return AggregatedPerformanceVector

Generating Learning Curves through repeated Train and Test splits

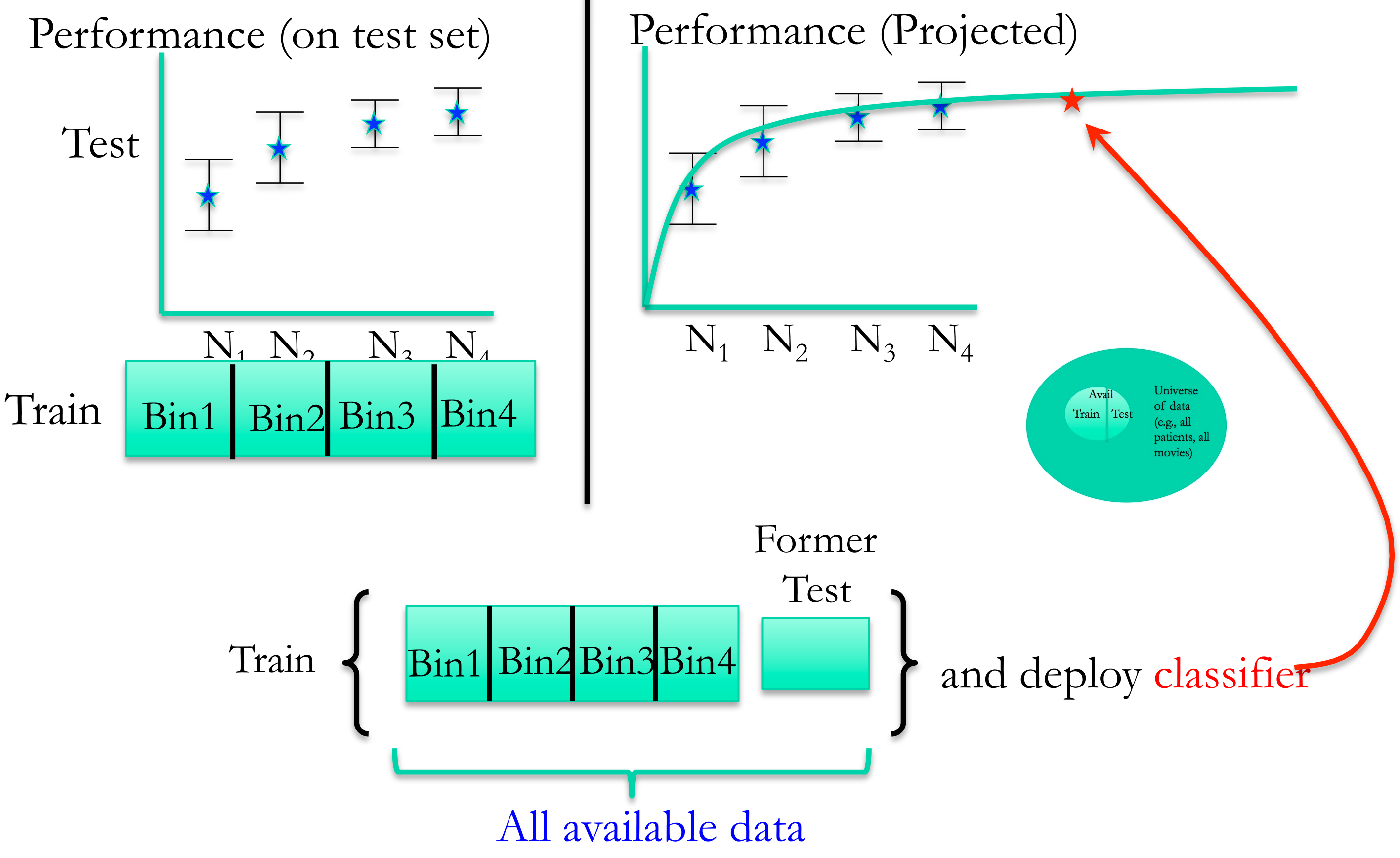


Initialize AggregatedPerformanceVector
Repeat L times {
    Train ← Randomly draw N_{max} data from Avail,
            “without replacement”
    Test ← Avail – Train
    Partition Train into 1 to max bins, TrainBin_1 through TrainBin_{max}
    For k = 1 to max {
        Classifier ← Learn(Union of TrainBin_1 through TrainBin_k)
        AggregatedPerformanceVector[k]
            ← Performance(Classifier, Test) + AggregatedPerformanceVector[k]
}
Return AggregatedPerformanceVector

How Might we use in Real Setting

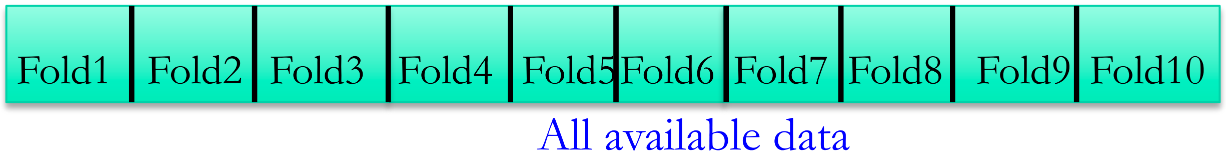
1. Use Training/Test splits to plot performance

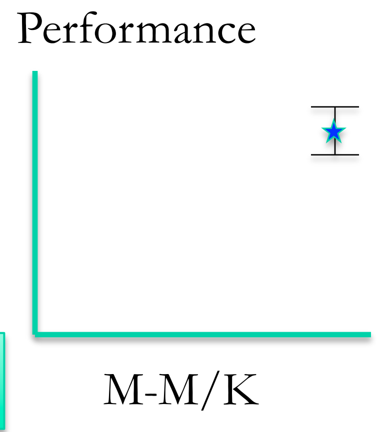
2. Curve fit learning behavior to project performance (on universe) at larger N



K-Fold Cross Validation

1. Randomize order of available M data
2. divide available data into K (e.g., 10) equal size bins or folds
3. For I = 1 to K {
   - Train on union of all folds, except $fold_I$
   - Test on $fold_I$
4. Average results





M-Fold Cross Validation
(or leave-one-out cross validation)

Divide a data set of size M into M singleton folds, and follow algorithm above (e.g., for each datum, train on M-1 other data and test on the datum)

This is often regarded as the best way to leverage the existing data and get as close as one can to estimating performance on a final deployed classifier trained on all data