Inferring Simple Rules
Apply Simplicity First!

PROF. JACQUES SAVOY
UNIVERSITY OF NEUCHATEL

I. H. WITTEN, E. FRANK, M.A. HALL: DATA MINING. PRACTICAL MACHINE LEARNING TOOLS AND TECHNIQUES. MORGAN KAUFMANN.

Simplicity First

Simple algorithms often work very well!

There are many kinds of simple structure, e.g.:

- One attribute does all the work.
- All attributes contribute equally & independently.
- A weighted linear combination might do the job.
- Instance-based: use a few (good) examples.
- Use simple logical rules.

Success of method depends on the domain.

Hypothesis: we assume that a *single* attribute is enough to achieve a good classification (the underlying structure of the real world can sometimes be quite simple).

Simplicity First

- Some works are in favor of the simplicity (and not directly with the 1R) (Hand, 2006).
- Past performance is not a guarantee for the future performance (evolution).
- Few predictors better than a lot of them.
- Simpler model perform between 85% to 95% to more advanced complex models.
- Complex models offer small improvements (over-fitting?).
- Poor quality of the training data.
- Try other simpler models

D. J. Hand: Classifier Technology and the Illusion of Progress. *Statistical science*, 21(1), 2006, pp. 1-14.

Inferring rudimentary rules

1R (1-rule): learns a 1 level decision tree.

- i.e., a set of rules that all test *one single* particular attribute

Basic version

- One branch for each value
- Each branch assigns most frequent class
- Error rate: proportion of instances that don’t belong to the majority class of their corresponding branch
- Choose attribute with lowest error rate

(*assumes nominal attributes*).

Example: Weather Problem

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Example: Weather problem

In this sample, we have 14 instances.
Classification: "*Play*": with 5 "*no*" and 9 "*yes*".

First approximation: answer always "*yes*" .
Error rate 5 / 14 = 0.357

Better: we can take account of one single attribute
and build a few rules according to the values of this single attribute.

Formally we can define as:

Pseudo-code for 1R

```
For each attribute,
  For each value of the attribute, make a rule as follows:
    - count how often each class appears
    - find the most frequent class
    - make the rule assign that class to this
        attribute-value
  Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

Note: “missing” is treated as a separate attribute value.

Example: Weather problem

We take into account for the value of the *outlook* attribute

Three possible values

- sunny (2 yes / 3 no),
- overcast (4 yes / 0 no),
- rainy (3 yes / 2 no)

The resulting single 1R is

Attribute	**Rules**	**Errors**	**Total errors**
Outlook	sunny → no overcast → yes rainy → yes	2 / 5 0 / 4 2 / 5	4 / 14

Example: Weather problem

Other attributes are possible:

Attribute	**Rules**	**Errors**	**Total errors**
Outlook	sunny → no	2 / 5	4 / 14
	overcast → yes	0 / 4	
	rainy → yes	2 / 5	
temperature	hot → no (random choice)	2 / 4	5 / 14
	mild → yes	2 / 6	
	cool → yes	1 / 4	
humidity	high → no	3 / 7	4 / 14
	normal → yes	1 / 7	
windy	false → yes	2 / 8	5 / 14
	true → no (random choice)	3 / 6	

Reduce error rate from 5/14 to 4/14.

Dealing with numeric attributes

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

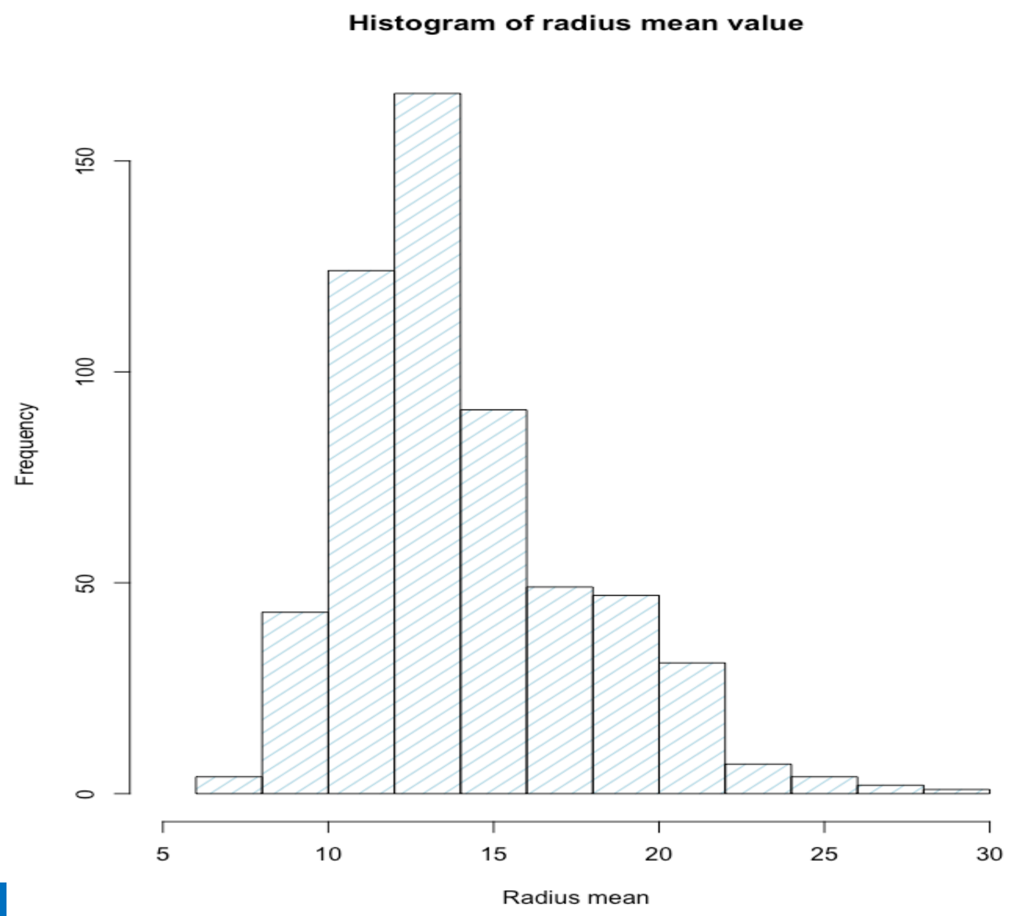
Dealing with numeric attributes

Missing value is viewed as an additional value possible for the corresponding attribute.

Discretize numeric attributes.

Unsupervised approach.

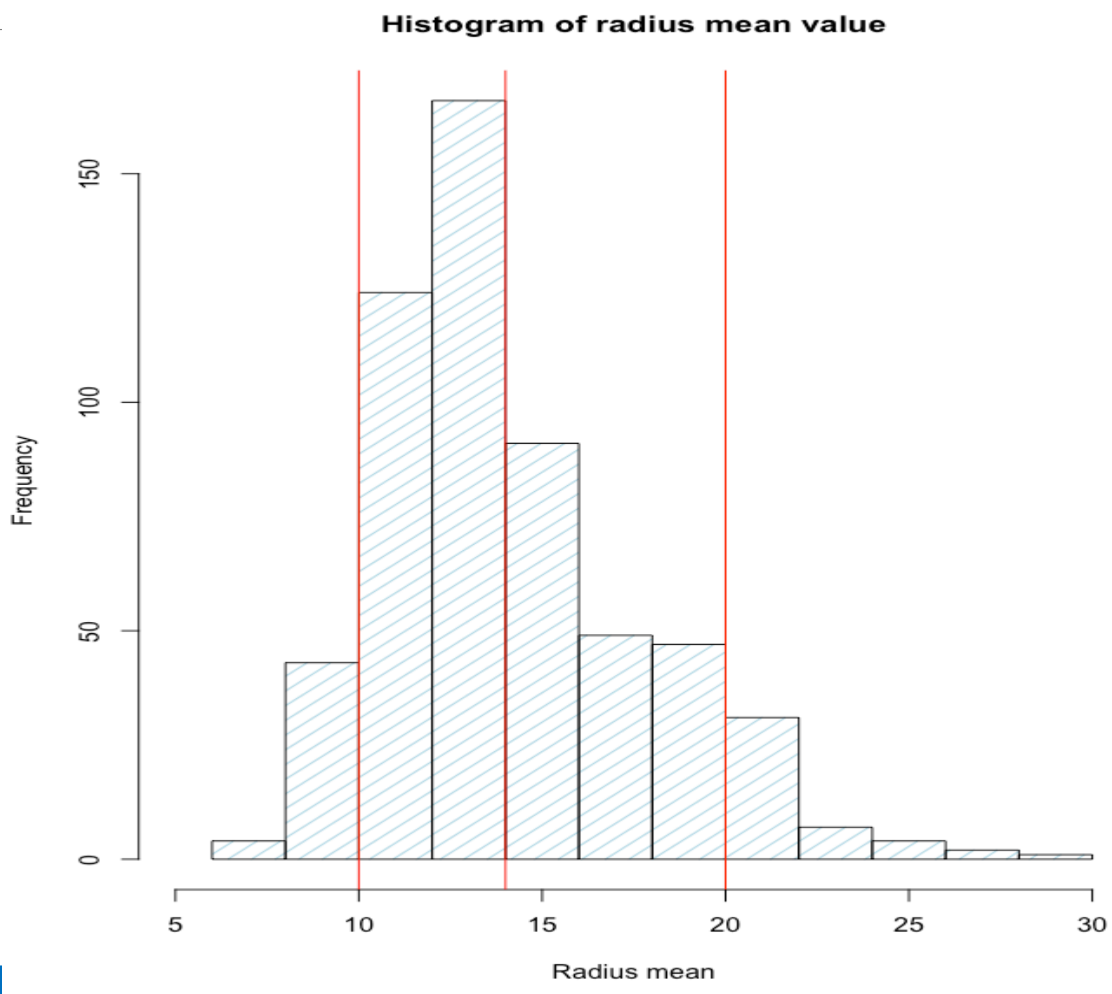
Build an histogram and select a few ranges (but ignoring the corresponding target value).



Dealing with numeric attributes

Unsupervised approach.

Build an histogram and select a few ranges (but ignoring the corresponding target value).



Binary Representation

ID	sun?	over?	rain?	hot?	mild?	cool?	high?	norm?	Windy?	Play?
ID1	1	0	0	1	0	0	1	0	0	0
ID2	1	0	0	1	0	0	1	0	1	0
ID3	0	1	0	1	0	0	1	0	0	1
ID4	0	0	1	0	1	0	1	0	0	1
ID5	0	0	1	0	0	1	0	1	0	1
ID6	0	0	1	0	0	1	0	1	1	0
ID7	0	1	0	0	0	1	0	1	1	1
ID8	1	0	0	0	1	0	1	0	0	0
ID9	1	0	0	0	0	1	0	1	0	1
ID10	0	0	1	0	1	0	0	1	0	1
ID11	1	0	0	0	1	0	0	1	1	1
ID12	0	1	0	0	1	0	1	0	1	1
ID13	0	1	0	1	0	0	0	1	0	1
ID14	0	0	1	0	1	0	1	0	1	0

Dealing with numeric attributes

Supervised methods.

Discretize numeric attributes.

Divide each attribute’s range into intervals:

- Sort instances according to attribute’s values.
- Place breakpoints where class changes (majority class).
- This minimizes the total error.

Example: *temperature* from weather data.

Dealing with numeric attributes

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

Dealing with numeric attributes

Example: *temperature* from weather data

Sort them:

```
64  65  68  69  70  71  72  72  75  75  80  81  83  85
Y   N   Y   Y   Y   N   N   Y   Y   Y   N   Y   Y   N
```

Place a breakpoint at each change:

```
64  |65  |  68  69  70|  71  72  |  72  75  75  |  80  |81  83  |  85
Y   | N  |  Y   Y   Y |  N   N   |  Y   Y   Y   |  N   |  Y   Y  |  N
```

and generate the breakpoint value (64.5, 66.5, 70.5, …)
but we have two instances with 72 values and different classification result. Move the break at 73.5 (and produce one error by considering the majority).

Dealing with numeric attributes

Tend to produce a large number of categories.

This procedure is very sensitive to noise.

- One instance with an incorrect class label will probably produce a separate interval.

Also: *time stamp* attribute (or other identification code) will have zero errors on the training example.

But this is clearly an overfitting schema.

Dealing with numeric attributes

Simple solution: *enforce minimum number of instances in majority class per interval*

Example (with *min* = 3):

```
64  65  68  69  70  |  71  72  72  75  75  |  80  81  83  85
Y   N   Y   Y   Y   |  N   N   Y   Y   Y   |  N   Y   Y   N
```

Producing the rule:

```
temperature ≤ 77.5 → yes
            > 77.5 → no
```

Example: Weather problem

With continuous value:

Attribute	Rules	Errors	Total errors
Outlook	sunny → no overcast → yes rainy → yes	2 / 5 0 / 4 2 / 5	4 / 14
temperature	≤ 77.5 → yes > 77.5 → no	3 / 10 2 / 4	5 / 14
humidity	≤ 82.5 → yes > 82.5 and ≤ 95.5 → no > 95.5 → yes	1 / 7 2 / 6 0 / 1	3 / 14
windy	false → yes true → no (random choice)	2 / 8 3 / 6	5 / 14

Reduce error rate from 5/14 to 3/14.

Conclusion

1R was described in a paper by Holte (1993).

- Contains an experimental evaluation on 16 datasets (using *cross-validation* so that results were representative of performance on future data).
- Minimum number of instances was set to 6 after some experimentation.
- 1R's simple rules performed not much worse than much more complex decision trees.

Simplicity first pays off!

Conclusion: Democracy

- Another simple technique: build one rule for each class.
  - Each rule is a conjunction of tests, one for each attribute.
  - For numeric attributes: test checks whether instance's value is inside an interval.
  - Interval given by minimum and maximum observed in training data.
  - For nominal attributes: test checks whether value is one of a subset of attribute values.
  - D. J. Hand: ClassSubset given by all possible values observed in training data
  - Class with most matching tests is predicted.

D. J. Hand: Classifier Technology and the Illusion of Progress. *Statistical science*, 21(1), 2006, pp. 1-14.

I Need to be More Precise “Simplicity First”

- Recent works are in favor of the simplicity (and not directly with the 1R) (Hand, 2006)
  - Past performance is not a guarantee for the future performance (evolution).
  - Few predictors better than a lot of them.
  - Simpler model perform between 85% to 95% to more advanced complex models.
  - Complex models offer small improvements (over-fitting?).
  - Poor quality of the training data.
- Try other simpler models.

D. J. Hand: Classifier Technology and the Illusion of Progress. *Statistical science*, 21(1), 2006, pp. 1-14.

Simplicity First

Using neural network model, the complexity measures by the number of hidden nodes.

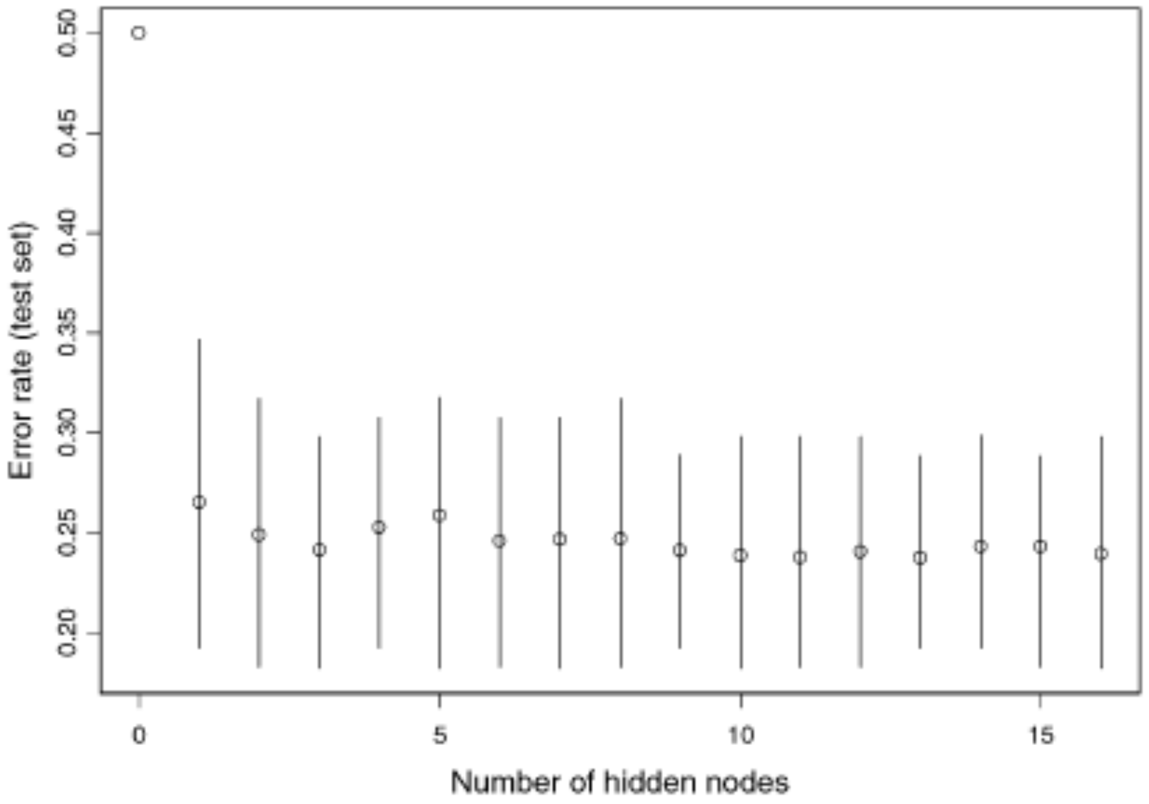


FIG. 2. *Effect on misclassification rate of increasing the number of hidden nodes in a neural network to predict the class of the sonar data.*

Simplicity First

In theoretical ML, we can detect a clear trend towards more complex classifiers.

Published results tend to show an improvement.

- Do we need to ignore simple approach?

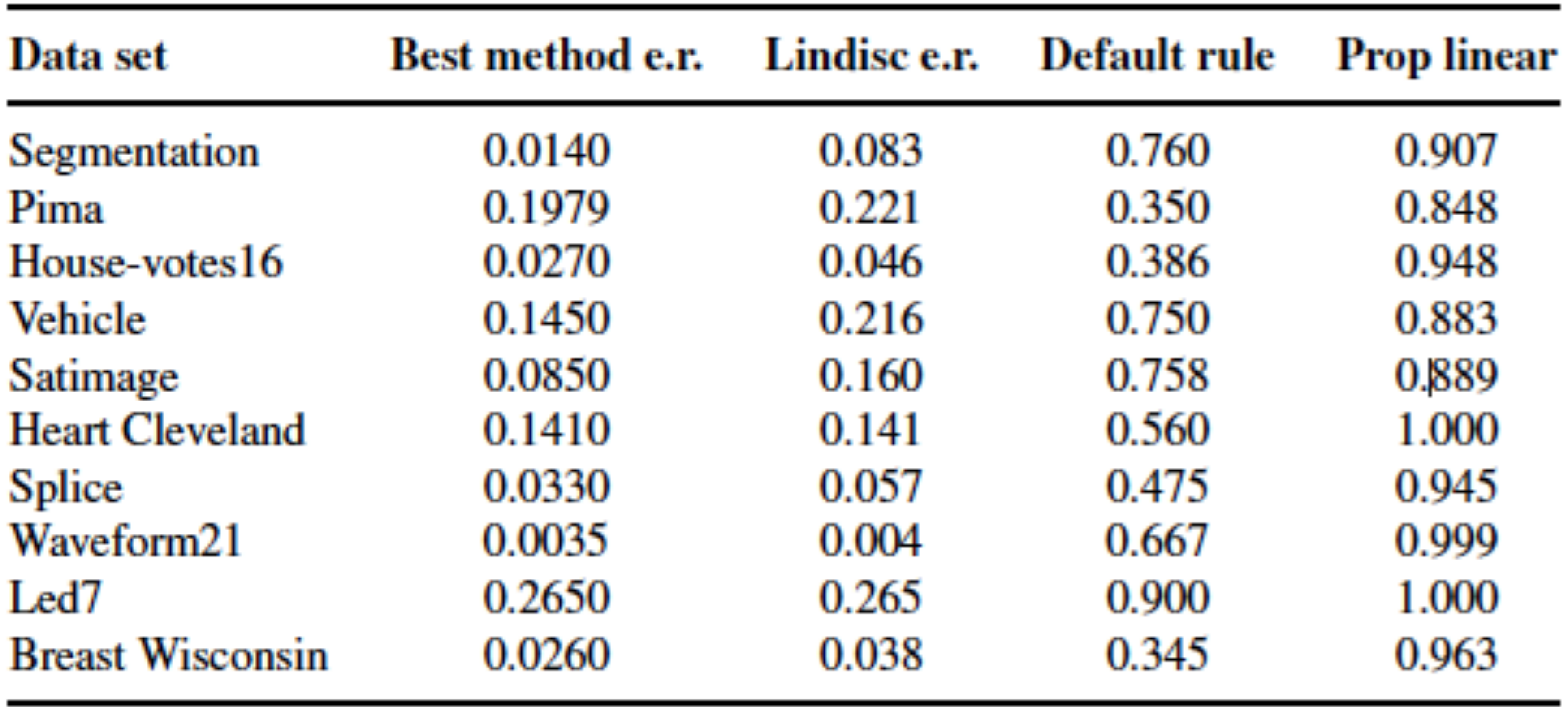
New (and complex) solutions show improvements.

Sure?

- Comparing linear discrimination function (Fisher, 1936).
- Comparing with 1R (Holte, 1993).

Hand, D.J.. (2006). Classifier Technology and the Illusion of Progress. *Statistical Science, 21(1), 1-14*.

Error Rate



Data set	Best method e.r.	Lindisc e.r.	Default rule	Prop linear
Segmentation	0.0140	0.083	0.760	0.907
Pima	0.1979	0.221	0.350	0.848
House-votes16	0.0270	0.046	0.386	0.948
Vehicle	0.1450	0.216	0.750	0.883
Satimage	0.0850	0.160	0.758	0.889
Heart Cleveland	0.1410	0.141	0.560	1.000
Splice	0.0330	0.057	0.475	0.945
Waveform21	0.0035	0.004	0.667	0.999
Led7	0.2650	0.265	0.900	1.000
Breast Wisconsin	0.0260	0.038	0.345	0.963

Hand, D.J.. (2006). Classifier Technology and the Illusion of Progress. *Statistical Science, 21(1), 1-14*.

Simplicity First

In theoretical ML, we can detect a clear trend towards more complex classifiers.

Published results tend to show an improvement.

- Do we need to ignore simple approach?

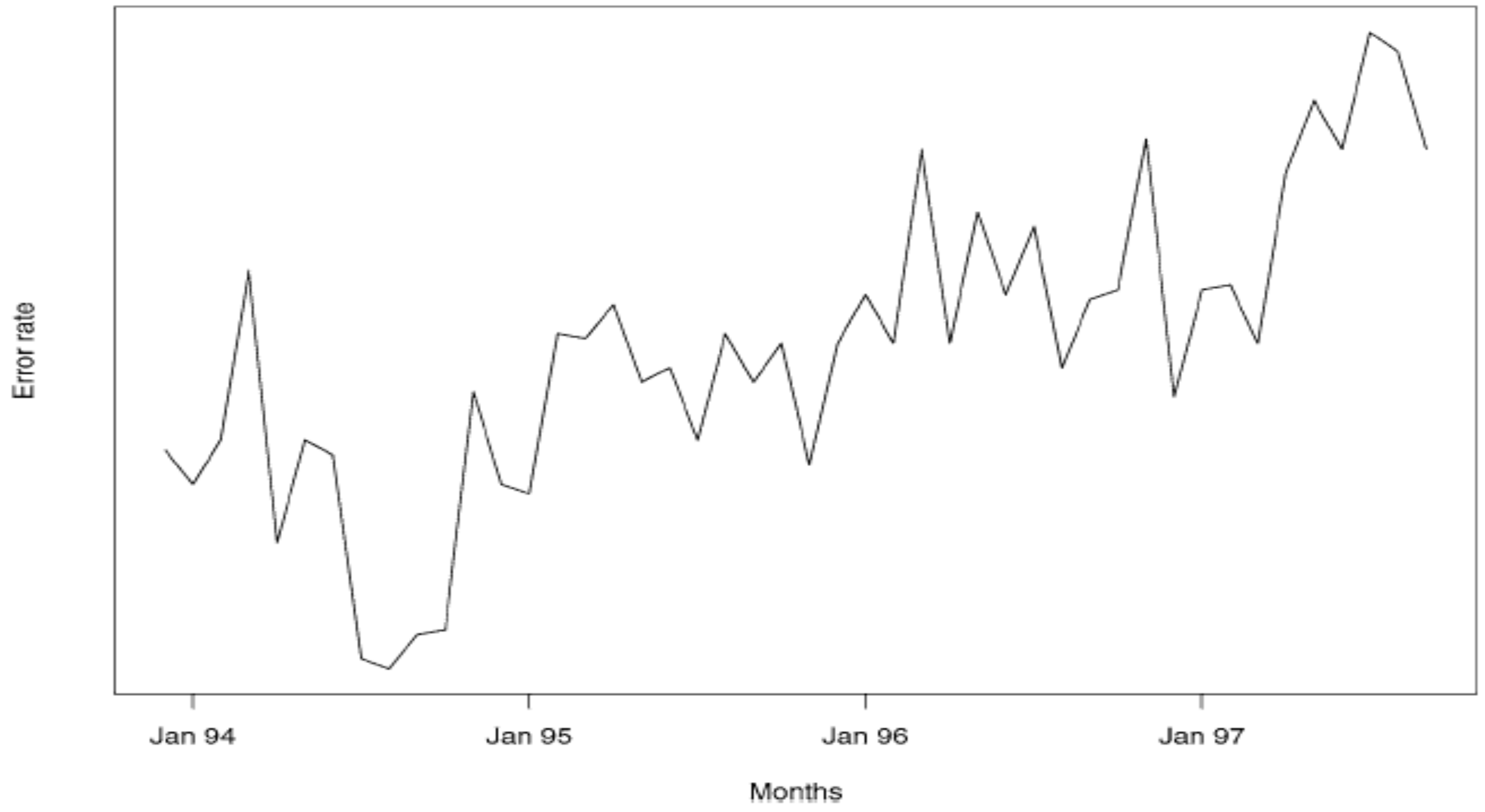
New (and complex) solutions show improvements.

Sure?

- Comparing linear discrimination function (Fisher, 1936).
- Comparing with 1R (Holte, 1993).

Hand, D.J.. (2006). Classifier Technology and the Illusion of Progress. *Statistical Science, 21(1), 1-14*.

What we (largely) ignore
The evolution (of the error rate)



Simplicity First

New solutions show only small improvements (law of diminishing returns).
Simple classifiers tend to represent 85% to 99% of the performance.

Evolution of the underlying distributions.

Complex solution tend to overfit the data.
Unable to produce good result on new unseen data.

Difficult to interpret the result:
why your system reach this target category?   (ensemble learning strategies)

Errors in category labels, errors in the data.

Hand, D.J.. (2006). Classifier Technology and the Illusion of Progress. *Statistical Science, 21(1), 1-14*.