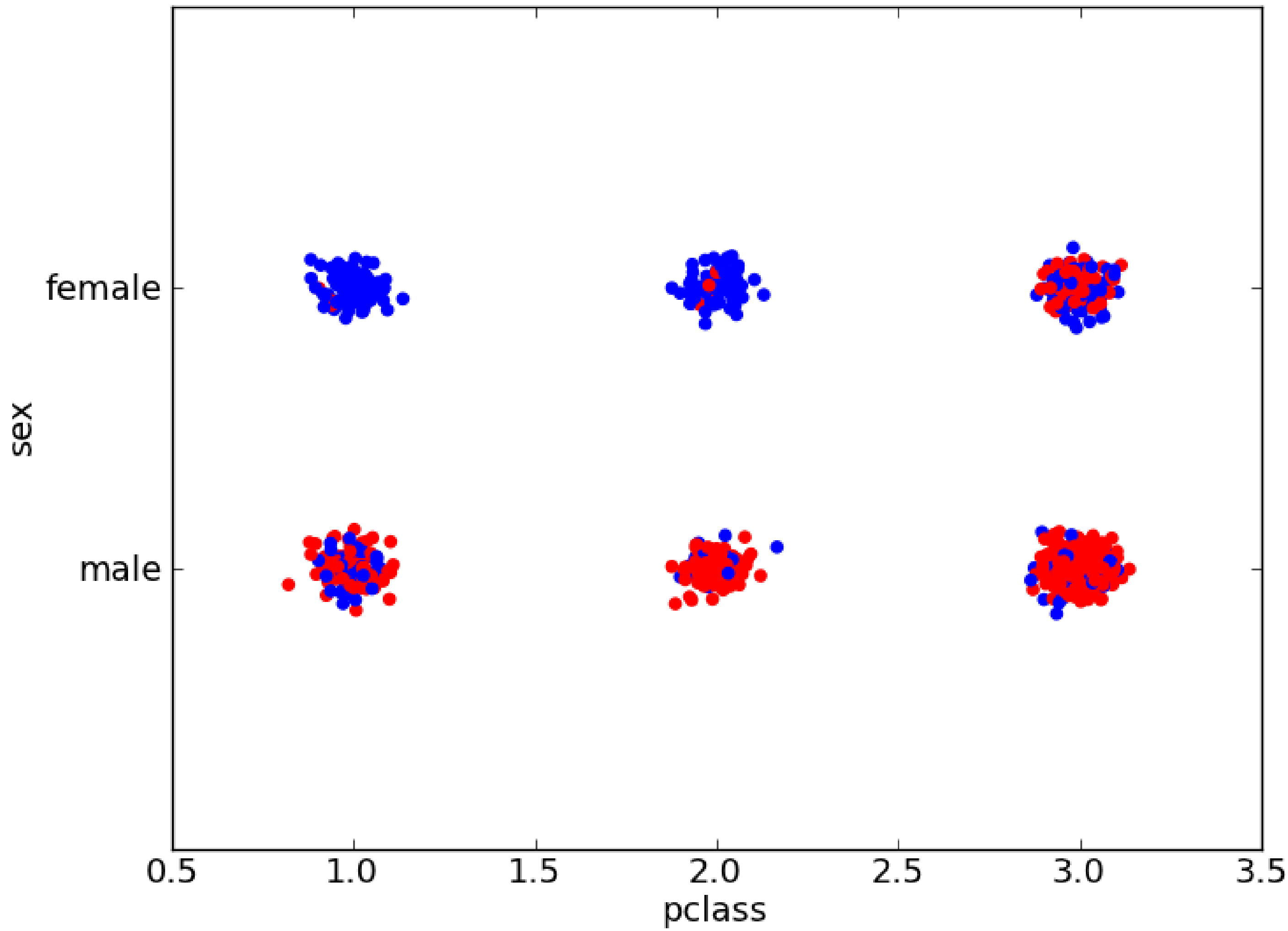
female
male
sex
0.5
1.0
1.5
2.0
2.5
3.0
3.5
pclass

How far can we go?

IF pclass='1' AND sex='female' THEN survive=yes
IF pclass='2' AND sex='female' THEN survive=yes
IF pclass='3' AND sex='female' AND age < 4 THEN survive=yes
IF pclass='3' AND sex='female' AND age >= 4 THEN survive=no
IF pclass='2' AND sex='male' THEN survive=no
IF pclass='3' AND sex='male' THEN survive=no
IF pclass='1' AND sex='male' AND age < 5 THEN survive=yes
…

Sequential Covering

```
Initialize R to the empty set

for each class C {

    while D is nonempty {

        Construct one rule r that correctly classifies
        some instances in D that belong to class C and
        does not incorrectly classify any non-C instances

        Add rule r to ruleset R

        Remove from D all instances correctly classified by r
    }

}

return R
```


src: *Alvarez*

Sequential Covering: Finding next rule for class C

```
Initialize A as the set of all attributes over D

while r incorrectly classifies some non-C instances of D {

    write r as ant(r) => C

    for each attribute-value pair (a=v),
    where a belongs to A and v is a value of a,
    compute the accuracy of the rule

        ant(r) and (a=v) => C

    let (a*=v*) be the attribute-value pair of
    maximum accuracy over D; in case of a tie,
    choose the pair that covers the greatest
    number of instances of D

    update r by adding (a*=v*) to its antecedent:
        r = ( ant(r) and (a*=v*) ) => C
    remove the attribute a* from the set A:
        A = A - {a*}
}
```


src: Alvarez

Strategies for Learning Each Rule

- General-to-Specific
  - Start with an empty rule
  - Add constraints to eliminate negative examples
  - Stop when only positives are covered
- Specific-to-General (not shown)
  - Start with a rule that identifies a single random instance
  - Remove constraints in order to cover more positives
  - Stop when further generalization results in covering negatives

Conflicts

- If more than one rule is triggered
  - Choose the “most specific” rule
  - Use domain knowledge to order rules by priority

Recap

- Representation
  - A set of rules: IF…THEN conditions
- Evaluation
  - coverage: # of data points that satisfy conditions
  - accuracy = # of correct predictions / coverage
- Optimization
  - Build rules by finding conditions that maximize accuracy

One rule is easy to interpret, but a complex set of rules probably isn't