RIPPER

William Cohen, Fast Effective Rule Induction, Proceedings of the 12^{th} International Conference on Machine Learning

IREP-Based

- Based on incremental reduced error pruning (IREP).
- Grow rules one at a time.
- Have a growing set of 2/3 of the examples for building the rule and a pruning set of 1/3.
- Build rules for 2 class problems. Order classes by size from smallest to largest.
- Build rules for smallest class vs. all other examples first.

Using a pruning set

- For statistical validity, must evaluate measure on data not used for training:
  - This requires a *growing set* and a *pruning set*
- *Reduced-error pruning* :
  build full rule set and then prune it
- *Incremental reduced-error pruning* : simplify each rule as soon as it is built
  - Can re-split data after rule has been pruned
- Stratification advantageous

Incremental reduced-error pruning

```
Initialize E to the instance set
Until E is empty do
   Split E into Grow and Prune in the ratio 2:1
   For each class C for which Grow contains an instance
      Use basic covering algorithm to create best perfect rule
         for C
      Calculate w(R):  worth of rule on Prune
            and w(R-): worth of rule with final condition
                       omitted
      If w(R) < w(R-), prune rule and repeat previous step
   From the rules for the different classes, select the one
      that's worth most (i.e. with largest w(R))
  Print the rule
  Remove the instances covered by rule from E
Continue
```

Incremental reduced-error pruning Modified for RIPPER

- Order classes according to increasing prevalence

  $(C_1, \ldots, C_k)$

  find rule set to separate C_1 from other classes

  `IREP(Pos=C1,Neg=C2,...,Ck)`

  remove all instances learned by rule set

  find rule set to separate C_2 from $C_3, \ldots, C_k$

  ...

  C_k remains as default class

Question

- The requirement in RIPPER of a pruning set
  - a) reflects the belief that learning on all training data may overfit
  - b) is done to minimize accuracy
  - c) will work better for large training sets, avoiding starving the learning system for data
  - d) uses the idea of just pruning a test when it does not improve performance on the test data.

Incremental reduced-error pruning Modified for RIPPER

```
procedure IREP(Pos,Neg)
begin
    Ruleset := ∅
    while Pos≠ ∅ do
        /* grow and prune a new rule */
        split (Pos,Neg) into (GrowPos,GrowNeg)
          and (PrunePos,PruneNeg)
        Rule := GrowRule(GrowPos,GrowNeg)
        Rule := PruneRule(Rule,PrunePos,PruneNeg)
        if the error rate of Rule on
          (PrunePos,PruneNeg) exceeds 50% then
            return Ruleset
        else
            add Rule to Ruleset
            remove examples covered by Rule
              from (Pos,Neg)
        endif
    endwhile
    return Ruleset
end
```

Growing a Rule

- To grow a rule, we have a training set of positive and negative examples.

- We add a test to a rule of the form
  - atttribute$_i$= v for a valid nominal value or atttribute$_i$ < x or atttribute$_i$ >= x for a continuous attribute with x in the range of values (usually x is an observed value)

Choosing a test to Grow a Rule

- Foil gain is used:

$$Foil\_Gain(Test, R) = t(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0})$$

- Where p_0 is the number of positive examples covered by R and n_0 is the number of negative examples covered by R
- p_1 is the number of positive examples covered by the R+ Test and n_1 is the number of negative examples covered.
- t is the number of positive bindings of R also covered by R+ Test.

Measures used in IREP

- $[p + (N - n)] / T$
  - (N is total number of negatives, p (n) positive (negative) examples covered, T total number of examples)
  - Counterintuitive:
    - $p = 2000$ and $n = 1000$ vs. $p = 1000$ and $n = 1$
- Success rate p / t
  - Problem: $p = 1$ and $t = 1$

  vs. $p = 1000$ and $t = 1001$
- $(p - n) / t$
  - Same effect as success rate because it equals $2p/t - 1$
- Seems hard to find a simple measure of a rule's worth that corresponds with intuition

Improvements to get RIPPER

$$v(Rule, \Pr uneP os, \Pr uneNeg) \equiv \frac{p-n}{p+n},$$

Where P (N) is the total number of examples in PrunePos (PruneNeg) and p (n) is the number of examples in PrunePos (PruneNeg) covered by Rule.

Improvements to get RIPPER

- Find total description length of rule set and examples computed.
- Stop adding rules when this description length is more that d bits larger than the smallest description length found thus far. (d=64).
- For a rule set R_i, …, R_k consider each rule in turn in order learned. Create replacement and revision rules.

Replacement and Revision Rules

- Replacement for R_i, R_i' is formed by growing and then pruning a rule with pruning guided to minimize error of entire rule set as measured on the pruning set.

$$R_1, \ldots, R_i', \ldots, R_k$$

- The revision is created by greedily adding conditions to R_i, rather than the empty rule.
- The final theory can contain only one of the original, replacement or revision rules based on MDL.

Question

- Ripper growing a replacement rule is based on the idea that
  - a) searching too much is bad
  - b) there are no good rules unless you use all data
  - c) all train/prune splits are equal
  - d) the random split into a training and pruning set may affect the quality of the rules obtained
  - e) a different rule may be built when looking at a full rule sets accuracy

Optimization

- Can add more rules from IREP* to get RIPPER2 and in general can get RIPPERk for k optimizations.
- Let a rule have k conditions of n possible conditions, pr be known by the message recipient (pr=k/n here) and ||k|| be the number of bits needed to send integer k. Equation for bits for rule is below.

$$S(n,k,pr) \equiv (k\log_2 \frac{1}{pr} + (n-k)\log_2 \frac{1}{1-pr} + \| k \|) \times 0.5 = bits$$

Optimization

- Rule accuracy can be encoded by exceptions (false positives and false negatives).
- Let a rule cover p of P cases with fp – false positives and fn - false negatives, the bits required to encode exceptions are:

$$bits = \log_2\left(\binom{p}{fp}\right) + \log_2\left(\binom{P-p}{fn}\right)$$

- To get the MDL you must sum all rules and exceptions for them.

Results

- RIPPER is much better than IREP* (28-7-2) for won, loss and tie on 37 data sets.
- Faster and better than C4.5 rules (20-15-2)