Descriptive rule learning

As we have seen, the rule format lends itself naturally to predictive models, built from rules with the target variable in the head. It is not hard to come up with ways to extend rule models to regression and clustering tasks

Descriptive models can be learned in either a supervised or an unsupervised way.

As an example of the supervised setting we will discuss how to adapt the given rule learning algorithms to subgroup discovery.

For unsupervised learning of descriptive rule models we will take a look at frequent item sets and association rule discovery.

Rule learning for subgroup discovery

When learning classification models it is natural to look for rules that identify pure subsets of the training examples: i.e., sets of examples that are all of the same class and that all satisfy the same conjunctive concept

We defined subgroups as mappings $\hat{g} : \mathcal{X} \rightarrow \{\text{true}, \text{false}\}$ – or alternatively, subsets of the instance space – that are learned from a set of labelled examples $(x_i, l(x_i))$, where $l : \mathcal{X} \rightarrow \mathcal{C}$ is the true labelling function. A good subgroup is one whose class distribution is significantly different from the overall population.

in our dolphin example, the concept Gills = yes, which covers four negatives and no positives, could be considered as interesting as its complement Gills = no, which covers one negative and all positives. This means that we need to move away from impurity-based evaluation measures.

Like concepts, subgroups can be plotted as points in coverage space, with the positives in the subgroup on the *y*-axis and the negatives on the *x*-axis.

Subgroups above (below) the diagonal have a larger (smaller) proportion of positives than the population. So one way to measure the quality of subgroups is to take one of the heuristics used for rule learning and measure the absolute deviation from the default value on the diagonal.

For example, the precision of any subgroup on the diagonal is equal to the proportion of positives, so this leads to $|prec - pos|$ as one possible quality measure.

As can be seen in Figure 6.15 (left), the introduction of pseudo-counts means that [5+,1−] is evaluated as [6+,2−] and is thus as interesting as the pure concept [2+,0−] which is evaluated as [3+,1−].

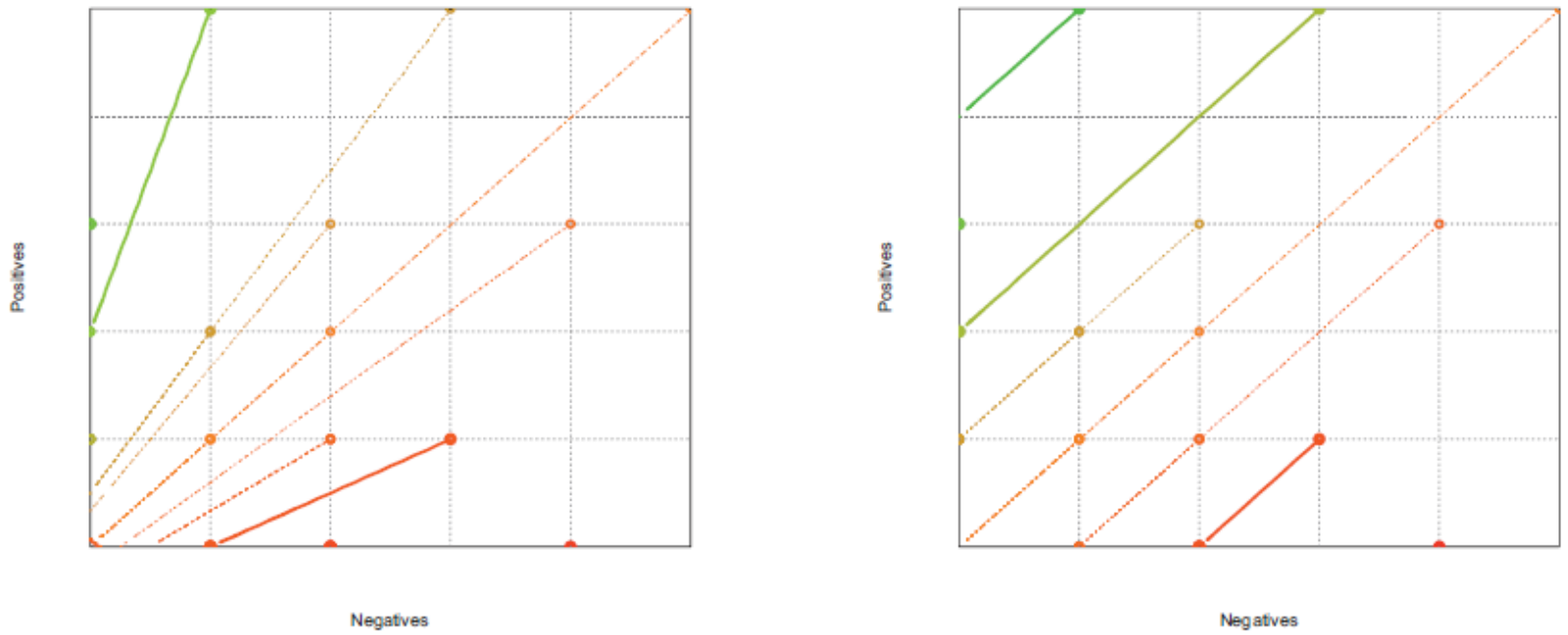

Figure 6.15. (left) Subgroups and their isometrics according to Laplace-corrected precision. The solid, outermost isometrics indicate the best subgroups. **(right)** The ranking changes if we order the subgroups on average recall. For example, [5+, 1−] is now better than [3+, 0−] and as good as [0+, 4−].

Average recall

Notice that subgroups on the diagonal always have average recall 0.5, regardless of the class distribution. So, a good subgroup evaluation measure is $|avg\text{-}rec - 0.5|$. Average recall can be written as $(1 + tpr - fpr)/2$, and thus $|avg\text{-}rec - 0.5| = |tpr - fpr|/2$.

Subgroup	*Coverage*	$prec^L$	*Rank*	*avg-rec*	*Rank*
Gills = yes	[0+, 4−]	0.17	1	0.10	1–2
Gills = no ∧ Teeth = many	[3+, 0−]	0.80	2	0.80	3
Gills = no	[5+, 1−]	0.75	3–9	0.90	1–2
Beak = no	[0+, 2−]	0.25	3–9	0.30	4–11
Gills = yes ∧ Beak = yes	[0+, 2−]	0.25	3–9	0.30	4–11
Length = 3	[2+, 0−]	0.75	3–9	0.70	4–11
Length = 4 ∧ Gills = yes	[0+, 2−]	0.25	3–9	0.30	4–11
Length = 5 ∧ Gills = no	[2+, 0−]	0.75	3–9	0.70	4–11
Length = 5 ∧ Gills = yes	[0+, 2−]	0.25	3–9	0.30	4–11
Length = 4	[1+, 3−]	0.33	10	0.30	4–11
Beak = yes	[5+, 3−]	0.60	11	0.70	4–11

Table 6.1. Detailed evaluation of the top subgroups. Using Laplace-corrected precision we can evaluate the quality of a subgroup as $|prec^L - pos|$. Alternatively, we can use average recall to define the quality of a subgroup as $|avg\text{-}rec - 0.5|$. These two quality measures result in slightly different rankings.

weighted relative accuracy,

which can be written as $pos \cdot neg(tpr - fpr)$. As can be seen by comparing the two isometrics plots in Figure 6.15, using average recall rather than Laplace-corrected precision has an effect on the ranking of some of the subgroups. Detailed calculations are given in Table 6.1.

Subgroup	*Coverage*	*avg-rec*	*Wgtd coverage*	*W-avg-rec*	*Rank*
Gills = yes	[0+, 4−]	0.10	[0+, **3**−]	0.07	1–2
Gills = no	[5+, 1−]	0.90	[**4.5**+, **0.5**−]	0.93	1–2
Gills = no ∧ Teeth = many	[3+, 0−]	0.80	[**2.5**+, 0−]	0.78	3
Length = 5 ∧ Gills = yes	[0+, 2−]	0.30	[0+, 2−]	0.21	4
Length = 3	[2+, 0−]	0.70	[2+, 0−]	0.72	5–6
Length = 5 ∧ Gills = no	[2+, 0−]	0.70	[2+, 0−]	0.72	5–6
Beak = no	[0+, 2−]	0.30	[0+, **1.5**−]	0.29	7–9
Gills = yes ∧ Beak = yes	[0+, 2−]	0.30	[0+, **1.5**−]	0.29	7–9
Beak = yes	[5+, 3−]	0.70	[**4.5**+, **2**−]	0.71	7–9
Length = 4	[1+, 3−]	0.30	[**0.5**+, **1.5**−]	0.34	10
Length = 4 ∧ Gills = yes	[0+, 2−]	0.30	[0+, **1**−]	0.36	11

Table 6.2. The 'Wgtd coverage' column shows how the weighted coverage of the subgroups is affected if the weights of the examples covered by Length = 4 are reduced to 1/2. 'W-*avg-rec*' shows how how the *avg-rec* numbers as calculated in Table 6.1 are affected by the weighting, leading to further differentiation between subgroups that were previously considered equivalent.

The second difference between classification rule learning and subgroup discovery is that in the latter case we are naturally interested in overlapping rules, whereas the standard covering algorithm doesn't encourage this as examples already covered are removed from the training set.

One way of dealing with this is by assigning weights to examples that are decreased every time an example is covered by a newly learned rule.

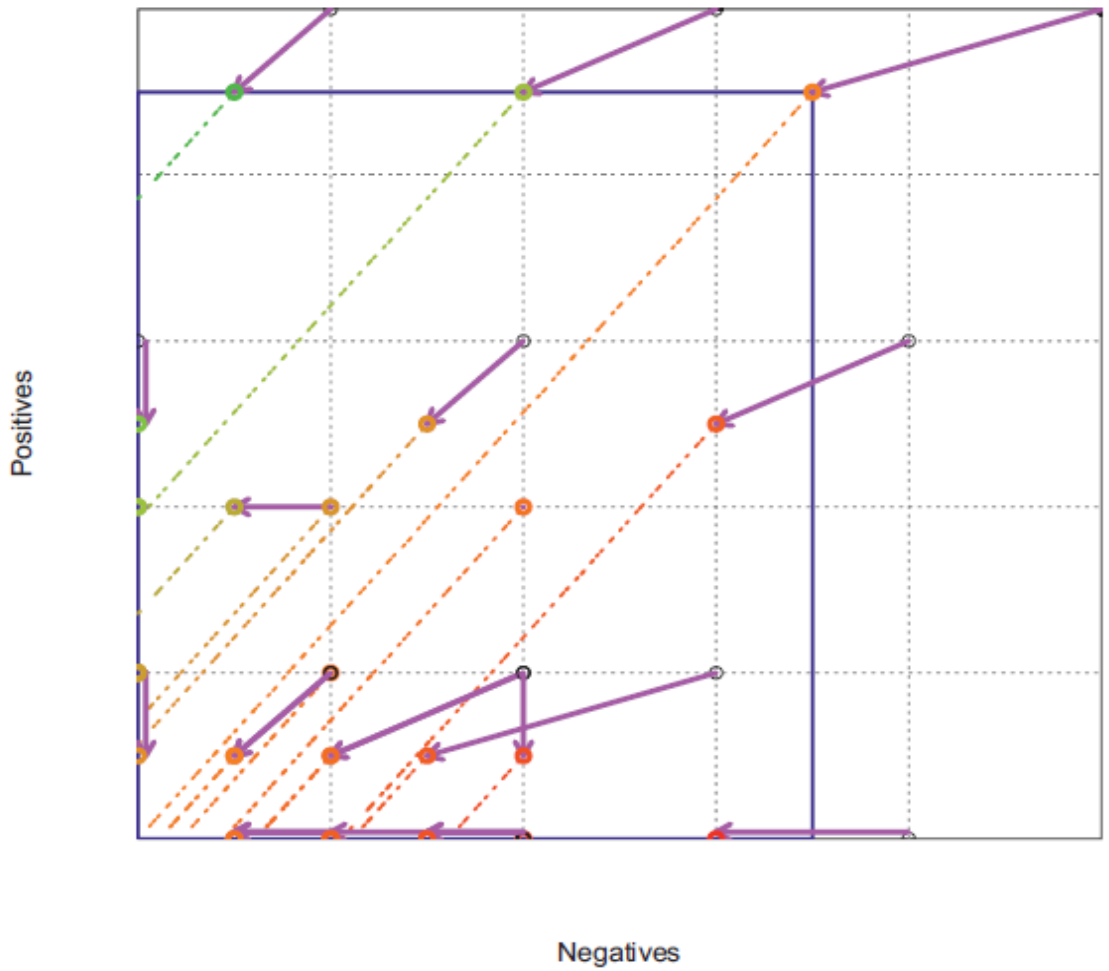

Figure 6.16. Visualisation of the effect of weighted covering. If the first subgroup found is Length = 4, then this halves the weight of one positive and three negatives, shrinking the coverage space to the blue box. The arrows indicate how this affects the weighted coverage of other subgroups, depending on which of the reduced-weight examples they cover.

The *weighted covering* algorithm is given in Algorithm 6.5. Notice that this algorithmcan be applied to discover subgroups over $k > 2$ classes, as long as the evaluationbmeasure used to learn single rules can handle more than two classes.

Algorithm 6.5: WeightedCovering(D) – learn overlapping rules by weighting examples.

```
Input  : labelled training data D with instance weights initialised to 1.
Output : rule list R.
R ← ∅;
while some examples in D have weight 1 do
    r ← LearnRule(D) ;                    // LearnRule: see Algorithm 6.2
    append r to the end of R;
    decrease the weights of examples covered by r;
end
return R
```