ShapeletsBug

May 2019

Example

Consider the case where you have 3 points from 2 classes, X and Y. Let's say x1 = 1 is class X, x2 = 0 is class X, x3 = 1 is class Y. If you had to divide this data for maximum information gain, the best way is somewhere between 0 and 1 (non-inclusive). This achieves a split of [1, 0] and [1, 1].

The way that the existing code does it is to sort the points: x2, x1, x3 Then, for each adjacent pair of observations in this list, it makes a candidate split, making everything to the left one node and the right another.

For instance, it would first look at a split halfway between x2 and x1 (at 0.5), which would give the aforementioned split of [1, 0] and [1, 1]. HOWEVER, it would then create a candidate split between x1 and x3 (which are both at 2), putting x1 and x2 on the left and x3 on the right. This artificially gives 100

Impact

As a result, the information gain reported for each node in training may be inflated, and sub-optimal thresholds and shapelets may be chosen. This degrades test performance. I further tested this by testing it on the training data, and it does not achieve the same accuracy it reported during training.

Luckily, the fix is simple - only perform the check to update the shapelet if the two points have different distances.

Code

```
for (int i = 0; i < (int)Dist.size() - 1; i++) {
        dist_th = (Dist[i].second + Dist[i + 1].second) / 2;
        // gap = Dist[i+1].second - dist_th;
        gap = double(Dist[i + 1].second - dist_th) / sqrt(subseq_len);
        label = Label[Dist[i].first];
        c_in[label]++;
        c_out[label]--;
        total_c_in++;
        num_diff = abs(num_obj - 2 * total_c_in);
        // gain = CalInfoGain1(c_in, c_out);
        gain = CalInfoGain2(c_in, c_out, total_c_in, num_obj - total_c_in);

        sh.setValueFew(gain, gap, dist_th, norm);
        if (bsf_sh < sh) {
          bsf_sh.setValueAll(gain, gap, dist_th, q_obj, q_pos, subseq_len,
                             num_diff, c_in, c_out, norm);
        }
}
```