ECE594BB Homework 5 Report
Nikita Buzov
03/20/2019

1. **Introduction**

   For this assignment we were given an imbalanced dataset with only few positive samples and had to use 'monomial learning' to obtain a k-term DNF (k=1 for the first part, and k=2 for the second). The dataset for the first part (monomial) consisted of 1500 samples with only one positive sample, each having 100 binary features. For the second part of this assignment (2-term DNF), the dataset had 300 samples with 2 of them positive.

2. **Design Methodology**

   a. **Monomial Learning**

   In order to find the right monomial, we need to encode all of the clauses into one CNF formula, which then can be fed into a SAT solver. First, define 3n variables to encode the constraints:

   - N_i: feature x_i is Negative in the Monomial
   - P_i: feature x_i is Positive in the Monomial
   - Z_i: feature x_i doesn't appear in the Monomial

   Now encode the hypothesis space (one-hot clause):

   $$(N_i+P_i+Z_i)\ (\neg N_i+\neg P_i)\ (\neg N_i+\neg Z_i)\ (\neg Z_i+\neg P_i)$$

   Then, we need to encode the positive and negative samples of the dataset. The encoding for the positive samples follows this algorithm:

```
For each positive sample:
     For i_th feature in sample:
          If i_th feature = 1 → new_literal = ¬N_i
          If i_th feature = 0 → new_literal = ¬P_i
          tmp_clause = And(tmp_clause, new_literal)
     clauses = And(clauses, tmp_clause)
```

   Next, encode the negative samples following the algorithm below:

```
For each negative sample:
     For i_th feature in sample:
          If i_th feature = 1 → new_literal = N_i
          If i_th feature = 0 → new_literal = P_i
          tmp_clause = Or(tmp_clause, new_literal)
     clauses = And(clauses, tmp_clause)
```

Lastly, we need to encode the monomial length, since we know that l=5. This is done by constraining the number of Z_i's that can be true (don't appear in the monomial). The following equations impose that constraint:

$$(Z_0 + \neg S_{0,0})(\neg Z_0 + S_{0,0})$$
$$\prod_{j=1}^{k-1} (\neg S_{0,j})$$
$$\prod_{i=1}^{n-1} (\neg Z_i + \neg S_{i-1,k})$$
$$\prod_{i=1}^{n-2} \Big((Z_i + S_{i-1,0} + \neg S_{i,0})(\neg Z_i + S_{i,0})(\neg S_{i-1,0} + S_{i,0})$$
$$\prod_{j=1}^{k-1} \Big((Z_i + S_{i-1,j} + \neg S_{i,j})(S_{i-1,j-1} + S_{i-1,j} + \neg S_{i,j})(\neg S_{i-1,j} + S_{i,j})(\neg Z_i + \neg S_{i-1,j-1} + S_{i,j}) \Big) \Big)$$
$$(Z_{n-1} + S_{n-2,k-1})(S_{n-2,k-2} + S_{n-2,k-1})$$

In the above equations S is a serial counter, k is a number of true Z_i's (k=95), and n is the number of features (n=100)

Now that we have all of the clauses, we `And()` them together to obtain one single CNF formula and run the SAT solver. The result will return a solution with five Z_i's equal to zero, meaning that those x_i features define the Monomial. Then we can find the Monomial itself by checking whether N_i or P_i is true.

b. 2-term DNF

The process for encoding the 2-term DNF is similar to what was done in part (a), except a few alterations. The main difference is that now we need to encode an additional dimension, so the variables N_i, P_i, Z_i now will have another index that corresponds to the dimension (e.g. $Z_{1,i}$). This doubles the number of variables used. For hypothesis encoding, we will now need two equations per feature like in part (a) from each dimension, and then `And()` them to get a complete clause. Similar idea applies to encoding positive and negative samples. We will define new literals for each dimension in the same way and then combine them:

```
For each positive sample:
      ...
      tmp_clause = Or(tmp_clause_1, tmp_clause_2)
      ...
For each negative sample:
      ...
      tmp_clause = And(tmp_clause_1, tmp_clause_2)
      ...
```

Lastly, the monomial length has to be encoded. We use the same equations for this, but now `k=195` (total length of the monomial is still 5) and `n=200` (100 features in every dimension), and `Z` is a list of both $Z_{1,i}$ and $Z_{2,i}$ appended term-by-term.

Now, once we `And()` the clauses up together, the formula can be plugged into SAT solver. The result will yield a total of five literals from both dimensions. The final 2-term DNF function is then equal to `Or()` of the monomial from 1st dimension and monomial from 2nd dimension: (Monomial1 + Monomial2)

3. Conclusion

When we first approached the problem of learning from highly imbalanced dataset in the previous homework, using the algorithms and tricks we knew at the time, the accuracy of prediction was pretty low. The CART or decision tree models didn't perform well with the lack of positive samples. Now we can see that monomial learning can outperform CART and decision tree algorithms and be very effective at finding the function that predicts whether a sample is positive, even with limited data. This can be very valuable since very often we don't have many positive samples or large datasets available to us in real world problems.