Technical updates in Supervised clustering with Support Vector machine

Vishnu Kumar
B19BB066

❖ **The learnt weight 'w': -**

For our item-pairs there is a pairwise feature vector **φij**. And when we train this item pair on a SVM then we learn some weight. And this weight 'w' gives us the similarity measure when we take its inner product with φij. We use this similarity measure to learn functions from x to y;

f(x; w) = argmax$_{y∈Y}$<w, φij>. This function is Ψ(x, y) which is a representation used for how similar two items were.

But some tasks may have exact argmax$_y$ f(x, y). Now if we approximate our argmax using this learnt 'w' then we may face new challenges. Lets, look at some more detail to understand why w may cause us a challenge.

Before applying our algorithm, we had set some constraints. Let's first understand the constraint which is a linear constraint and it states that **for all training examples** and **any possibly wrong output y**. The **discriminant function for the correct output** should be greater than the **discriminant value of incorrect output** by at least the **loss between the correct clustering and incorrect clustering**. And we also used a **small slack value as an upper bound**. There is also a reason behind this slack as we want not to cut short the running of our algorithm much before reaching its optimum state and we give it a small relaxation here.

∀i, ∀y ∈ Y : <w,Ψ(xi, yi)> + <w,Ψ(xi, y)> ≥ Δ(yi, y) − ξi

And our algorithm is: -

1: Input: (x1, y1), . . . ,(xn, yn), C, ξ
2: Si ← Ø for all i = 1, . . . , n
3: repeat
4: for i = 1, . . . , n do
5: H(y) ≡ Δ(yi , y) + wT Ψ(xi , y)− wT Ψ(xi , yi)
6: compute ŷ = argmax$_{y∈Y}$ H(y)
7: compute ξi = max{0, max$_{y∈Si}$ H(y)}

8: if H(ŷ) > ξi + ξ then
9: Si ← Si ∪ {ŷ}
10: w ← optimize primal over S = $\bigcup_i$ SI
11: end if
12: end for
13: until no Si has changed during iteration

What we have done here is we started with no constraints for all our examples. Then we repeatedly went through all our examples and found out output ŷ associated with most violated constraint. Now we check if the constraint is getting violated by more than ϵ then we introduce our constraints and reoptimize. And we stop finally when there come no constraints throughout the way.
But if the iteration stops before stopping its optimal value because having constraints declared previously.

❖ **Proposal: -**

To counter this problem, we need to do something with our previously defined constraints so what we can do is along with the function calculated by using w, we can use one more function say a quadratic polynomial. So, what we can do is we can formulate a quadratic program,

$\min_{w,\xi}$ ½ $||w||^2$ + C/n $\sum_{k=1}^{n} \xi i;$ s.t. ∀i : ξi ≥ 0.

If we would rely on this polynomial program then it has its advantages such as it will run for a polynomial number of iteration and return the solution as desired to ξ. Now we do not depend only on w so we might end up getting a better result as we have also evolved to get more iterations in our model.
So, what we will be doing is that when we will check for the previous or the linear constraint getting violated then we will introduce our new constraint of the quadratic model and then we will again calculate our w with this and then we will calculate the similarity measure and the function without any constraint and put it back to the algorithm. Then this will be checked if it violated the new constraint or not. If yes then it will get optimized further for several times and this number of iterations also got increased due to using a polynomial program.

If talk about disadvantages of this proposed way then since we are getting relied on two different constraints now so checking for these constraints one after the another and the increased number of iterations can increase the computational cost.

❖ Relaxation Approximation

In the paper relaxed approximation has been use but if we see that relaxed approximation gives us actually a relaxation in obtaining our result. And from the final observations we were able to compare under-constrained greedy and over constrained relaxed approximation. And the results were not significantly different but still the over constrained greedy approximation is bit better for achieving a better result so it is better if we just ignore the relaxations here and stick to greedy only. But still we should not remove our slack value which we have defined earlier in our constraints because that provided initially a greater opportunity to optimise. However, this relaxation and that relaxation are different terms but one may refer it same if the context is ignored.

❖ Conclusion

I proposed that for the learnt weight ‘w’ used for calculating the similarity measure. We should not be dependent only on ‘w’ to find our argmax function. So, we may use other formulations also such as the quadratic program which also enabled us to reach to the optimum values better. And the relaxation given as the slack value should be decided properly as it should be small but still should be able to enable us a few with more iterations if need, and we could ignore the relaxation approximation and just go with under-constrained greedy approximation.