MLPACK::DecisionTree

ISSUE[884]

NIKOLAY APANASOV

Hello, my name is Nikolay Apanasov: during my undergraduate years I studied Mathematical Economic Analsysis and received a B.A. from Rice University. Now I am a graduate student at University of Colorado Boulder, working toward my Master of Science in Computer Science (and likely a PhD). My interests lie in the intersection of Machine Learning and Computer Systems. I have extensive experience in Computer Systems, from Compiler Engineering down to Digital Design and Computer Architecture on FPGAs, and I am working towards developing my expertise in Machine Learning and Deep Learning. Last semester I completed a graduate survey course on ML: ensemble methods were by far my favorite module in the course, and I spent extra time researching and implementing algorithms such as `RandomForest`. I was looking for a way to begin contributing to your project, and Issue[884] got my attention. Issue[884] is a feature request for more `DecisionTree` split types. As requested, I have completed a (brief) survey of literature related to this question. In what follows, I will describe

- an optimization to the current algorithm used in MLPack for choosing an optimal numeric split
- an optimal binary split for categorical variables and a two-class response
- the general case of binary categorical splits with an M-class response
- locally optimal binary splits using a K-means clustering algorithm
- linear combination splits
- some initial empirical results

Based on my research and experience thus far, my proposal is to implement a minor optimization to the current implementation of `BestBinaryNumericSplit`, and to create a new split type `BestBinaryCategoricalSplit`. The categorical splitter constructs an optimal split in linear time for a two-class response, and in time $O(2^Q)$ in the general setting with M classes and Q categories.

Optimal Numerical Splits

We can improve the current splitter `BestBinaryNumericSplit` for quantitative variables. Fayyad and Irani [FI92] discovered (and proved) that independent of the number of classes and their distribution among the samples, a cut point for a numeric variable v_k always occurs between two sample points X_j and X_k such that the $C(X_j) \neq C(X_k)$, where $C(\cdot)$ denotes the class of a sample. That is they proved that a cut point occurs precisely at a *boundary*. The authors prove this result for a φ that is the information entropy heuristic. An immediate consequence of this discovery is that the criterion function φ need not be evaluated $N-1$ times for v_k, but only at these boundary cut points. The authors describe

their results obtained by running Quinlan's ID3 algorithm on medium-sized datasets from the UC Irvine Machine Learning Repository. Although the performance metrics used by Fayyad and Irani are peculiar, and I refer you to their paper for more details, it certainly is clear that their algorithm increased the performance of ID3 in numerous problem settings. The most dramatic improvements occur for settings with a binary response, because boundary points are less likely to occur. Based on the timing results for the UCI datasets, it looks as if the authors achieved a modest speedup of 1.25-2x.

This optimization is relatively trivial to implement, and I have completed an initial implementation, for which I just submitted a pull request. Moreover I have run the modified `DecisionTree` on different datasets, including MNIST, Iris, Titanic and Covertype. The trees that are produced are slightly different, although not by much: in my tests, the classification performance on a test set differed by at most $\approx .002$. For example on the Covertype dataset, the test error increased by .0004, while on MNIST it decreased by .002. Similarly, these limited trials have not demonstrated any notable differences in performance. As noted before, the authors proved the result for the information entropy heuristic, however I constructed trees using the Gini index. It would not be surprising if the result can be generalized to the Gini index, or maybe more generally to a family of 'well-behaved' convex functions.

Finally, I should mention a related discovery that I made. The paper by Fayyad and Irani needs to be read carefully. In particular, there is a special case to be aware of when discussing boundary points. In the case when there is a sequence of samples with repeated values for variable v_k, those samples can of course be rearranged in an equivalent order. Therefore if there is such a sequence, and the samples in the sequence come from multiple classes, then the next cut point is a boundary point, independent of the classes of the samples on either side of the cut. Initially, I forgot about this special case, and implemented the logic without it. That implementation had noticeably different performance. The execution time on several datasets was $10-20\%$ faster. Using the same example as before, in this case the test-error on Covertype increased by .0012, whereas on MNIST it decreased by .0013. It would be worthwhile to perform a systematic analysis, and see if this change, while maybe not theoretically sound, might be useful in practice. In this direction, is there a good way to visualize the MLPack *DecisionTree*? For my Machine Learning course, I implemented Decision Trees and Random Forests from scratch, and to help with debugging, I wrote a little tree-walk routine that produces a DOT format representation that can be processed by a graphviz engine. However I haven't yet found this facility in the MLPack library.

Categorical Splits

MLPack has an optimal binary splitter for numeric types. It would seem reasonable to add an optimal splitter for categorical types as well, say `BestBinaryCategoricalSplit`. At

first glance, knowing that Q categories implies 2^{Q-1} possible binary partitions, it seems that this variant would be infeasible, even for moderately-sized Q. When the problem is binary classification, there is a surprising solution: the idea is to order the categories $C_1, \ldots, C_Q$ of variable v_j by the proportions $\pi_1, \ldots, \pi_Q$ respectively, where π_k is the proportion of samples from category C_k that are in class 1. Then we can proceed to split categorical v_j as if it was a numeric type. It is possible to show (see Proposition 7.1 in Ripley [Rip96] or Theorem 4.5 in CART by Breiman et al. [BFOS84]) that this split is optimal in relation to a cross-entropy or Gini index heuristic. Theorem 4.5 is a generalization of the result of W.D. Fisher [Fis58] which applies when the response Y is quantitative. Thus in the important special case of a binary classification problem, it is obvious what `BestBinaryCategoricalSplit` needs to do.

What about in the more general case for an M-class response? Unfortunately, to the best of my knowledge, nobody has yet discovered an elegant simplification here. The CART algorithm uses the simple answer: for general M, CART chooses the best categorical subset by enumerating all 2^{Q-1} possibilities. There are some interesting alternatives, though.

Chou [Cho91] proved an 'optimal partitioning theorem', which in a sense generalizes Theorem 4.5 of Breiman et al. to settings where the number of classes $M > 2$. The theorem provides necessary conditions on the optimal partition, and the proof is by construction. The construction immediately leads to an algorithm, an iterative descent clustering algorithm formally equivalent to Lloyd's algorithm, using a generalization of KL-divergence as a distance metric instead of Euclidean distance. In the case of a binary split, Chou's algorithm produces a *locally*-optimal split, and it runs in time $O(M \cdot N^{M+1})$: the complexity is $O(M \cdot N)$ per iteration, with an upperbound of $O(N^M)$ iterations, where M is the number of classes and N is the number of samples involved in the split. The algorithm and the paper are quite interesting, but for many multi-class settings involving a large number N of samples, Chou's algorithm may not be practical. In sum, it would be quite nice to have this variant available, but for the moment, my proposal is the brute-force algorithm chosen by CART. Not that this would be the goal of the design: it will be made explicit in the documentation for `BestBinaryCategoricalSplit` that this splitter is meant to be used for the two-class setting, or in the general setting when the number of categories is few.

Currently, MLPack has only one variant for splitting categorical variables, namely `AllCategoricalSplit`, which creates a multiway split, one branch for each category. A number of sources claim that binary tree structures result in more effective classifiers. Ripley [Rip96] highlights a result from Bratko and Konenko [BK87] which shows that in general, binary trees are smaller and have improved accuracy in comparsion with multiway trees. Breiman et al. likewise conveyed preference for binary splits, and the CART algorithm [BFOS84] constructs binary trees. The superiority of binary trees is also claimed by Hastie et al. in ESL [TTF01]:

> ...we might consider multiway splits into more than two groups. While this can sometimes be useful, it is not a good general strategy. The problem is that

multiway splits fragment the data too quickly, leaving insufficient data at the next level down. Hence we would want to use such splits only when needed. Since multiway splits can be achieved by a series of binary splits, the latter are preferred.

Linear Combinations

Lastly, let us briefly consider a very different splitting criterion based on linear combinations. Loh and Vanichsetakul [LV88] proposed a tree algorithm that used a modifed version of LDA to construct splits. Linear combinations were also explored in section 5.2 of CART:

> Suppose there are m_1 ordered variables... At every node t, take a set of coefficients $a = \left(a_1, \ldots, a_{m_1}\right)$ such that $\|a\|^2 = 1$, and search for the best split of the form
>
> $$\sum_m a_m x_m \leq c$$
>
> as c ranges over all possible values. Denote this split by $s^*(a)$ and the corresponding decrease in impurity by $\Delta i(s^*(a), t)$. The best set of coefficients a^* is that a which maximizes $\Delta i(s^*(a), t)$. That is
>
> $$\Delta i(s^*(a^*), t) = \max_a \ \Delta i(s^*(a), t)$$
>
> This produces a linear split of the form
>
> $$\sum_m a_m^* x_m \leq c^*$$
>
> Although the concept is easily stated, the implementation in terms of an effective search algorithm for maximizing $\Delta i(s^*(a), t)$ over the large set of possible values of a is complex. The details of the algorithm we use are given in the appendix to this chapter. It is not guaranteed to produce a^*. Like other search algorithms, it may get trapped in local maxima.

Although linear combinations are described in CART, they were not employed. The authors argue that the resulting trees are no longer invariant under strictly monotone transformations of a variable. Furthermore, you lose interpretability, a significant advantage of Decision Trees. Nevertheless, this could be an interesting avenue to explore for future work. Ripley [Rip96] provides a survey of related literature in chapter 7 of his book.

Conclusion

The proposal is to implement the cut-point optimization for `BestBinaryNumericSplit`, and to create a new split type `BestBinaryCategoricalSplit`. The categorical splitter

constructs an optimal split in linear time for a two-class response, and in time $O(2^Q)$ in the general setting with M classes and Q categories. For future work, it may be worthwhile to consider implementing `KMeansBinaryCategoricalSplit`, a splitter that finds locally optimal binary categorical cut points using a K-means clustering algorithm.

References

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.

[BK87] I. Bratko and I. Kononenko. Learning diagnostic rules from incomplete and noisy data. *Interactions in Artificial Intelligence and Statistical Methods*, pages 142– 153, 1987.

[Cho91] P. A. Chou. Optimal partitioning for classification and regression trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:340–354, 1991.

[FI92] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[Fis58] W. Fisher. On grouping for maximum homogeniety. *Journal of the American Statistical Association*, 53:789–798, 1958.

[LV88] W. Loh and N. Vanichsetakul. Tree structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83:715–725, 1988.

[Rip96] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[TTF01] Hastie T., R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.