Computers and Electrical Engineering 000 (2017) 1–15

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

An efficient distributed protein disorder prediction with pasted samples[☆]

Denson Smith[a], Sumanth Yenduri[b], Sumaiya Iqbal[c], P. Venkata Krishna[d,*]

[a] University of New Orleans, USA
[b] Department of Computer Science, Columbus State University, USA
[c] Department of Computer Science, University of New Orleans, USA
[d] Senior Member, IEEE, Dept. of Computer Science, Sri Padmavati Mahila University, Tirupati, India

ARTICLE INFO

Article history:
Received 29 December 2016
Revised 4 August 2017
Accepted 5 August 2017
Available online xxx

Keywords:
Protein disorder classification
Big data
Pasting
Statistical query learning
Machine learning

ABSTRACT

In this paper, we compare prediction performance of a machine learning classifier constructed at once in memory with an ensemble of models constructed with the pasting procedure for protein disorder prediction. The pasting procedure takes sample bites of the training data as input, constructs a classification predictor on each sample and pastes the predictors together. This method has not been previously tested on protein structure data. With a sufficiently large sample size we observed increased performance for the pasting procedure compared with a single model constructed at once in memory for all window sizes. We attribute this increased performance to the robustness of the statistical query learning model. This procedure provides a means to improve classification performance at the protein disorder prediction task as well as construct models too large to be held at once in memory.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

In the field of biomedical research, big data analytics has proven to be an extremely useful tool in addressing many types of challenges. Often, our analysis involves the application of machine learning to model our data in order to gain insights into the underlying processes and make accurate predictions based on our past observations. Often, the datasets we encounter are too large to be analysed on a single computer, especially when the underlying processes are complex. Therefore, we must somehow break the problem into parts that can be solved on a single computer and combine the parts into a single set of result. Ideally, the model constructed from the combined parts would be the same as if we had been able to construct a single large model on a single computer.

The process of distributing a problem over a cluster of computers and then combining the intermediate results from each node into a final solution is usually referred to as the Map Reduce programming model [1]. During the Map phase the problem is distributed to nodes in the cluster and during the Reduce phase the results from each node are aggregated into a final solution or model. While methods to efficiently parallelize machine learning algorithms exist [2], there are others for which no efficient parallelization method has been found. Indeed, whether all tractable problems can be parallelized in such

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. S. A. Aljawarneh.
[*] Corresponding author.
E-mail addresses: dsmith8@uno.edu (D. Smith), yenduri_sumanth@columbusstate.edu (S. Yenduri), siqbal1@uno.edu (S. Iqbal), Dr.krishna@ieee.org, parimalavk@gmail.com (P.V. Krishna).

http://dx.doi.org/10.1016/j.compeleceng.2017.08.002
0045-7906/© 2017 Elsevier Ltd. All rights reserved.

a way that a gain in computational speed is achieved is an open problem in computer science (NC = P?) [3]. For example, it seems that gradient boosting algorithms [4] are inherently sequential and impossible to efficiently parallelize (assuming NC ! = P). Other machine learning algorithms such as support vector machines are only partially parallelizable [5]. Note that we are speaking of parallelizing an algorithm to obtain the exact results as the sequential version of the algorithm and not some approximation.

Many times we encounter datasets that are sufficiently large that they cannot be stored at once in the memory on an available cluster. Indeed, datasets of petabyte size are becoming increasingly common [6,7]. There are well tested and supported methods that enable the application of machine learning methods to large datasets [6,8–13]. Some machine learning methods such as stochastic gradient decent logistic regression classifier [14] do not require that the entire dataset be in memory at once. This type of algorithm is known as “external memory” or “out of core” [15]. Unfortunately, we have a limited selection of available learning algorithms that are scalable in this way. For example, Apache Mahout has only three classification methods: logistic regression trained with stochastic gradient descent, naïve bayes and hidden Markov models. While these methods are useful, they do not perform well on all types of problems, we may encounter. A common practice is to evaluate multiple methods for any given problem and select the best performing method [16,17]. By comparison, scikit-learn [18] which is less scalable, has over a dozen available classifiers.

In order to address this limitation, methods have been developed that seek to approximate the solution that would be obtained if we had available sufficient computational resources to load the entire dataset and construct a sequential model at once in memory [10,12,13]. These techniques are based upon the statistical query learning model of Kearns [19,20]. One such method that has been successfully applied to many types of algorithms and problem domains is “pasting” [13,21,22]. Pasting combines statistical query learning with “stacking” predictions [23–25]. In stacking, we combine the predictions from multiple models to arrive at a final prediction. Often, the different models are the same algorithm with different parameters. With pasting, we train different models on different samples (statistical queries) of the dataset and then combine the individual predictions to make a final prediction on test data. Stacking has been shown to be robust against noise [26], overfitting [27], outliers [28] and often improves prediction performance compared to a single, tuned model. Since pasting employs multiple independent models it is inherently parallelizable. Since the models are constructed on subsets of the training data it inherently requires less memory than a model trained on the entire dataset. As mentioned above, the performance of this approach has been tested and compared to the performance of single models trained at once in memory. Louppe and Geurts [21] showed that for some problems the pasting method performed as well as a single model constructed at once in memory but for other problems pasting resulted in substantially reduced performance. In this work we compare the performance of the pasting method with a single model on a particularly challenging problem domain: the prediction of intrinsically disordered proteins (IDPs) and intrinsically disordered regions (IDRs) in proteins.

IDPs and IDRs are entire proteins or regions in proteins that do not adopt a well-defined, stable three-dimensional structure in their native state [29]. The prediction of protein structures is a very active area of research with competitions organized by the Protein Structure Prediction Center (http://predictioncenter.org/) and sponsored by the National Institute of General Medical Sciences (https://www.nigms.nih.gov/) every two years. Knowledge of protein structures is extremely useful in understanding a variety of both normal biological functions as well as disease conditions. Additionally, predicting protein structures is useful in synthesizing new proteins that may be used as drugs to treat diseases. Since synthesizing proteins is expensive in terms of both time and money, the accurate prediction of protein structures is critical in drug design and disease treatment.

This problem domain presents a number of challenges for constructing a predictive model that performs well:

1. Protein structure classification is a highly non-linear problem.
2. Some learning methods/dataset combinations we wish to evaluate require a large amount of memory.
3. There are many potential datasets to be tried.
4. There is a large parameter space to be searched for model tuning.
5. The input feature dimensionality is fairly high (>1000 features).
6. The available data have missing labels and incorrect labels.
7. Individual rows in the dataset are not independent. If the first protein consists of 100 residues then the first 100 rows of the data are all related to one another.

In order to construct a predictive model for protein structure prediction we must complete the following steps:

1. We must select a number of candidate protein structure datasets for training. Ideally, the proteins included in our final training dataset are representative of all proteins with unknown structure that we wish to predict in the future. Obviously, this step requires a great deal of domain knowledge and is usually accomplished through a community collaboration.
2. In order to evaluate each candidate dataset we must use it as input to construct a machine learning model and measure the predictive performance of that model against a set of known data that is representative of unknown data we wish to predict in the future. The best performing machine learning methods such as artificial neural networks (ANN) and support vector machines (SVM) usually perform very poorly (often worse than chance) without extensive parameter tuning [30]. In order to construct a model with acceptable performance, a large parameter space must be searched in order to tune the model. In order to evaluate each candidate parameter set we must construct a separate machine

Download English Version:

https://daneshyari.com/en/article/6883578

Download Persian Version:

https://daneshyari.com/article/6883578

Daneshyari.com