Investigating the Incorporation of Machine Learning Concepts in Data Structure Education

Bo Liu[1,2], Fengying Xie[1,2]
School of Astronautics[1], Beijing Advanced Innovation Center for Biomedical Engineering[2]
Beihang University
Beijing, China
{bo.liu, xfy_73}@buaa.edu.cn

***Abstract*—This Research to Practice Work-In-Progress paper discussed the incorporation of machine learning (ML) concepts in data structure education. The thriving of the ML especially deep learning techniques has led to an increased demand for trained professionals with ML skills to solve challenging engineering problems in many fields. Getting students familiar with ML as early as from CS2 (the data structure course) could benefit them in many aspects, but this direction has not been explored yet. In this paper, we discussed possible ways to integrate the ML concepts into data structure (DS) course. First, after introducing the concept of tensor in DS classroom teaching, we propose a practical experiment to implement the forward propagation of a pretrained convolutional neural network (CNN) aiming at classifying handwritten digits. Second, an experiment of decision tree based classification is set to give students an illuminating context via practicing the usage of tree structure. Finally, we design the experiment of computing graph to help the understanding of Directed Acyclic Graph (DAG), in which the students are required to implement the calculation of a multiple-variable function and its gradient based on DAG. Practicing DS knowledge in interesting ML-related problem contexts would intrigue the study enthusiasm of students and give them a general understanding of the application of DS knowledge in frontier technology, which could benefit the education of both DS and ML-related courses.**

Keywords—data structure, machine learning, convolutional neural network, decision tree, computing graph

I. INTRODUCTION

This WIP practice paper discussed the incorporation of machine learning concepts in data structure (DS) education. Data structure is the basis for programming and the CS2 (data structure) is the core course for computer and information related majors. The student performance of CS2 has a large impact on their following coding-related courses and may have influence on their career. Many works have been conducted to improve the pedagogy of data structures [1]–[4] or analyze the factors influencing the student performance [5], [6]. The content evolution of the CS2 could be split into two generations [4]. In the first generation, the focus of CS2 was on describing and implementing the basic data structures, implementing simple demo examples, etc. In the second generation, the focus was switched to solving meaningful problems using basic data structures. Nowadays, the incorporation of professional techniques into the teaching of data structure has been extensively discussed. One example is the utilization of the image processing in CS2 teaching [1] [7]. At University of Illinois at Urbana-Champaign, image data was employed as a mechanism to explore course topics [7]. Adams from Calvin College believed the advantage of introducing image process into CS2 teaching was the motivation of student interest as the students could get an immediate, visual feedback of executed code [1]. Beside of image processing, the integration of parallel computing concepts into CS2 has also been considered [7]–[9]. With the rapid advances in parallel system and its widely usage in practice, it was thought that every CS student should learn and gain experience with parallelism [8] and the earlier the students get in touch of parallel computing concepts the more they will benefit [10]. There is no doubt that the incorporation of these concepts in return has a promoting effect on the education of data structure itself. By putting data structure in more meaningful problem contexts, the interest of the students could be motivated and they will have a deeper understand of the data structure.

In recent years, machine learning (ML), especially the deep learning (DL), has become the hottest topic not only in research but also in industry fields. Many countries including the U.S., China, have considered the ML as the impeller for the next-generation industry. The thriving of the machine learning (ML) has led to an increased demand for trained professionals with ML skills to solve challenging engineering problems in many fields. Therefore, it is urgent to introduce the concept of ML to CS students as early as possible, just as the case for parallelism. As data structure is the basic for coding and algorithm and many ML algorithms rely on dedicated data structure, we argue that incorporation of ML concepts in data structure education is a possible way, even maybe a good way, to introduce ML. We also believe the integration not only leads to a more solid foundation for the advanced ML-related core courses but also could improve the teaching performance of DS, as the students have a chance to flexibly practice DS knowledge in interesting and relatively sophisticated problem contexts. However, to the best knowledge of the authors, there is no discussion regarding this topic yet.

In this paper, we presented plausible ways to integrate ML into DS education. We hope this work could arouse more discussion in this field and further improve the education of the students.

II. CONTENT AND MOTIVATION

A. Tensor and Convolution Neural Network

Tensor is the cornerstone data structure for deep learning and is basically an extension to multiple-dimensional array. As Ian Goodfellow put in the book 'deep learning', "in the general case, an array of numbers arranged on a regular grid with a variable number of axes is known as a tensor". Therefore, it is natural and necessary to discuss tensor after array, matrix and generalized list. These important concepts would be covered, including:

1) Rank and shape: While the rank of a tensor is the number of dimensions of that tensor, the shape of a tensor represents the length of each dimension.

2) Data type: Common data types of the tensor and some rules of thumb to choose the data type in practice.

3) Storage and index order: The C-like index order and the Fortran-like index order. In C-like index order, the last axis index changes fastest and back to the first axis index changs slowest. In the contrary, the last axis index changes slowest and the first axis index changs fastest for Fortran-like index order.

4) Tensor Operations: Some basic while useful and intuitive operations help students further understand tensor, including shape related operations (reshape, flatten, transpose, swapaxes, etc), joining and splitting operations (concatenate, stack, split, array_split, etc), adding and removing element operations (delete, insert, append, etc).

To practice tensor manipulation, after introducing these concepts, the students will be instructed to implement the forward propagation of a convolutional neural network (CNN). The classic LeNet5 [11] (as shown in Fig.1) with pre-trained weights will be given to students. The network was trained on MNIST dataset to classify the handwritten digits. After implementation, the students could test the network on digits written by themselves.

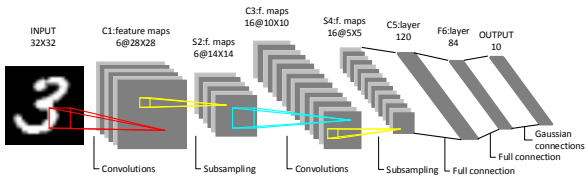

Fig. 1 The structure of LeNet5

The students are first instructed to define the class interface (class Tensor) for 3D tensor. The class includes a linear float array storing the tensor data and an array with three elements storing the shape. The input image, features and the convolution filters will be instances of this class.

Three fundamental functions are involved in this experiment. The first is the convolution function used to perform the convolution filter between an input feature tensor and a convolution filter, the interface of which could be defined as

```
std::shared_pointer<Tensor>
conv2d( std::shared_pointer<Tensor> input_feature,
        std::shared_pointer<Tensor> conv_filter);
```

The second is the max pooling function which calculates the maximum, or largest, value in each patch of each feature map. Its interface is defined as

```
std::shared_pointer<Tensor>
maxpooling( std::shared_pointer<Tensor> input_feature,
int patchsize[2]);
```

The third is the relative simple Rectified Linear Unit (ReLU) function which takes the maximum between zero and the feature value element-wise.

```
std::shared_pointer<Tensor>
relu( std::shared_pointer<Tensor> input_feature);
```

Other operations like fully connected layers may be provided to students to balance the workload of the experiment.

B. Decision Tree for Classification

Decision tree is a common machine learning algorithm used for classification and regression [12]. Based on a tree structure, it makes decisions by selecting current optimal partition attribute so that the samples contained in the branch nodes belong to the same category as possible, as shown in Fig.2. It aims at generating a decision tree with strong generalization. As the decision tree algorithm structurally depends on tree structure, it should be a good topic to teach student about tree manipulation.

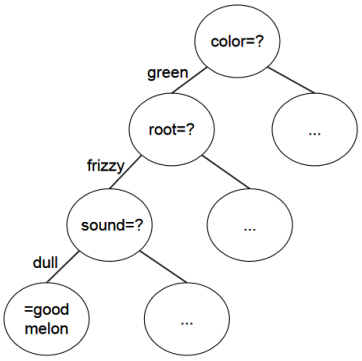

Fig. 2 For a decision tree, while leaf nodes correspond to decision results, root node and other internal nodes correspond to attribute tests

The watermelon dataset, which involves 2 classes of watermelon (good or bad) and 6 kinds of watermelon features (color, root condition, knock sound, texture, touching feel, umbilical region condition), is provided to students. They are expected to build a decision tree model which can predict a new watermelon good or bad according to features. Through this experiment, the programming and the logical reasoning ability of students could be improved, and their comprehension of common data structures including array, linked list and tree are also enhanced.

The dataset is provided as a CSV file. The students are firstly instructed to define the data structure used to store the dataset and implement the I/O function to read the CSV file. To exercise the ability of manipulating nest array, the dataset could be stored into a customized structure as follows:

```
using uchar = unsigned char;
using Feauretype = std::vector<std::vector<uchar>>;
using Labeltype = std::vector<uchar>;
struct DataSet{
   Feauretype features;
   Labeltype labels;
};
```

Then, students are guided to design a dedicated data structure we call TreeNode. The TreeNode structure should contain the chosen attribute, its associated samples and its children nodes. Through this, their ability of designing customized data structure to cater for problem-related data access need will be exercised. Besides, students are able to deepen their understanding of linked list and tree as they choose the design.

After the students determine the basic units and finish the data import, we guide them to the specific recursive steps of building a decision tree. In order to emphasize our focus on data structure, we choose the basic ID3 algorithm to select

feature which is relatively simple in principle. We first consider the end condition of recursion in the order of logical reasoning. Then, in the recursion process, we find the best attribute with the largest information gain as the root node, then divide the examples into several subsets according to the characteristic value of the selected attribute, and recursively invoke the function of building tree for each subset.

Before students recursively create decision trees, they need to implement some auxiliary functions. For example, one is the function which selects the attribute used to further split the associated samples and create branches. In this function, it is needed to transverse from current node to the root to collect attributes that have been used. Besides, the students need to implement the ID3 algorithm to calculate the information gain for each candidate attribute.

```
int select_attribute(TreeNode *node);
```

After this, the students are instructed to implement the recursive function to build the decision tree. The input of this function is the imported training samples and the output is the root node of the decision tree. With a proper design of the data structure and pre-implementation of sub-functions, the implementation of this function would be a simple recursive calling of the same function. However, if the data structure is improper, it would be much harder to correctly implement. Through this, the students would have a deeper apprehension the importance of the data structure design.

```
TreeNode * build_decisiontree(const DataSet &ds_train);
```

With the trained decision tree model, the students are finally instructed to implement the prediction function to predict a watermelon good or bad according to its attributes. In this function, the students should transverse from the node to one leaf node to determine the label of the input sample.

```
uchar predict(
const TreeNode *model, std::vector<uchar>feature);
```

C. Computational Graph

Computational graph is an important and fundamental concept in machine learning. Neural networks are a special form of it, deep learning toolkits like Theano and Tensorflow rely on it to perform forward-propagation and back-propagation. In the language of data structure, computational graph is a directed acyclic graph (DAG), with nodes corresponding to operations or variables. Variables or operations can feed their value or output into operations. Every node in the graph defines a function of the variables with variable nodes as special identity functions.

Considering the importance of the computational graph and its close relation to data structure, we propose to introduce it into the teaching of DAG and related algorithms such as topological sorting. Through a carefully designed experiment, students can learn the construction of computational graph and how to use it to implement more advanced algorithms. Moreover, it is beneficial for them to understand the core idea behind machine learning algorithms more clearly, which lays a foundation for further learning of ML.

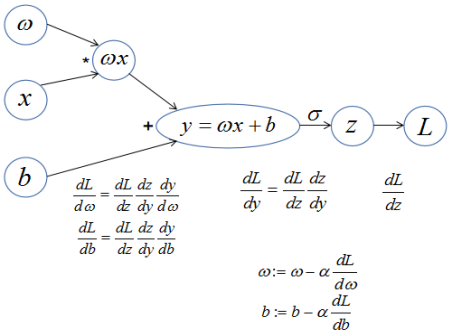

Fig.3 The computational graph and its application in gradient descent algorithm

In this experiment, students are guided to implement the forward and backward propagation of a simple logistic model, as shown in Fig.3. Specifically, we combine the input feature x with the parameters ω, b linearly, and then pass the linear combination through a logistic function σ to get the output. Given a set of features $\{x_i\}$ and their corresponding label $\{\widetilde{y}_i\}$, the students are instructed to minimize the following cross entropy function to obtain optimal parameters:

$$L = -\sum \widetilde{y}_i \log y_i = -\sum \widetilde{y}_i \log\left(\frac{1}{1+e^{-(\omega x_i + b)}}\right)$$

where $\widetilde{y}_i$ is the true label of the i-th sample. We need to obtain the derivative $d\omega$ and db of the cost function L with respect to each parameter through the chain derivation. The derivatives represent the influence of each parameter's change on the cost function.

After giving the students a brief description of the theory of computational graph, they are instructed to realize the computing graph step by step. The implementation of the experiment can be divided into several parts. The first part is the definition of graph structure, including various operations such as topological sorting, information query, etc. This step can deepen the understanding of DAG, enabling them to learn the knowledge of topological ordering in this process.

The second part is to implement several calculating functions, such as finding inner product of vectors, sigmoid function and differential operation. The implementation of these functions can consolidate the basic knowledge of mathematics and strengthen the concept of calculation of the students.

The third part involves the process of iterative optimization through gradient descent, in which the calculating functions of the second parts are called, and the information of the graph is updated. Logistic regression is a classical ML method, and gradient descent algorithm is used in many machine learning theories. Therefore, this experiment is not only an introduction to learning ML-related knowledge, but also an in-depth study of DS, which intrigues students and help them think deeply about the relationship between ML and DS.

III. DISCUSSION AND CONCLUSION

There is an increased demand for trained professionals with ML skills to solve challenging engineering problems in many fields. Meanwhile, the knowledge of data structure is one of the core bases for the information engineering and lies the foundation of many ML algorithms. While some professional techniques such as image processing have been successfully applied to the education of DS, the integration of ML with DS education still needs to be explored.

In this paper, we discuss a plausible way to integrate ML with the teaching of DS. The first suggested topic is the concept of tensor which is the basic data structure for deep learning and an extension of multiple-dimensional array. After introducing its basic concept, the students will be instructed to implement an interesting CNN network to classify hand-written digits. On basis of this, an experiment of decision tree based classification is utilized to practice the usage of tree structure. Given a set of samples containing features and labels, the students are asked to construct the decision tree, and finally test the constructed decision tree on some test samples. In this experiment, the students will focus on DS-related programming, such as tree manipulation, file I/O and string processing. In addition, we utilize the experiment of computing graph to help the teaching of Directed Acyclic Graph (DAG), in which the students are required to implement the calculation of a multiple-variable function and its gradient based on DAG. This mimics the forward and backward propagation of neural network, covering several important knowledge points including the design of the computing node structure, graph construction and topological sorting, etc.

In conclusion, this paper proposes to introduce ML concepts into data structure education and discusses several possible ways of integrated practical experiments. Practicing DS knowledge in interesting ML-related problem contexts would intrigue the study enthusiasm of students and give them an overview of how DS knowledge is applied in frontier technology, which could benefit the education of both DS and ML-related courses. Further works will be conducted to investigate and verify the advantage of incorporating ML in the DS education.

ACKNOWLEDGMENT

This work was supported by education reform project of Beihang University.

REFERENCES

[1] K. Hunt, “Using image processing to teach CS1 and CS2,” *ACM SIGCSE Bull.*, vol. 35, no. 4, pp. 86–89, 2003.

[2] R. R. Murphy, “Teaching computer algorithms & data structures to students in engineering & the physical sciences,” *Proc. - Front. Educ. Conf. FIE*, pp. 397–401, 1993.

[3] C. Latulipe, S. Macneil, and B. Thompson, “Evolving a Data Structures Class Toward Inclusive Success,” *Proc. - Front. Educ. Conf. FIE*, vol. 2018-Octob, pp. 1–9, 2019.

[4] D. J. Ernst, D. E. Stevenson, and P. J. Wagner, “Hybrid and custom data structures: Evolution of the data structures course,” *Proc. Conf. Integr. Technol. into Comput. Sci. Educ. ITiCSE*, pp. 213–217, 2009.

[5] L. Layman, Y. Song, and C. Guinn, “Toward Predicting Success and Failure in CS2: A Mixed-Method Analysis,” pp. 218–225, 2020.

[6] H. Bisgin, M. Mani, and S. Uludag, “Delineating Factors that Influence Student Performance in a Data Structures Course,” *Proc. - Front. Educ. Conf. FIE*, vol. 2018-October, pp. 1–9, 2019.

[7] S. Massung and C. Heeren, “Visualizing parallelism in CS 2,” *Proc. - IEEE 27th Int. Parallel Distrib. Process. Symp. Work. PhD Forum, IPDPSW 2013*, pp. 1252–1256, 2013.

[8] J. C. Adams, “Injecting parallel computing into CS2,” *SIGCSE 2014 - Proc. 45th ACM Tech. Symp. Comput. Sci. Educ.*, pp. 277–282, 2014.

[9] D. Grossman and R. E. Anderson, “Introducing parallelism and concurrency in the data structures course,” *SIGCSE’12 - Proc. 43rd ACM Tech. Symp. Comput. Sci. Educ.*, pp. 505–510, 2012.

[10] H. Wan, X. Gao, X. Long, and B. Jiang, “Introducing parallel computing concepts in computer system related courses,” *Proc. - Front. Educ. Conf. FIE*, vol. 2017-Octob, pp. 1–7, 2017.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[12] L. Rokach and O. Maimon, Data Mining with Decision Trees, vol. 69. WORLD SCIENTIFIC, 2007.