# Supervised Learning via Ensemble Tensor Completion

Nikos Kargas
*Dept. of ECE, Univ. of Minnesota*
Minneapolis, USA
karga005@umn.edu

Nicholas D. Sidiropoulos
*Dept. of ECE, Univ. of Virginia*
Charlottesville, USA
nikos@virginia.edu

***Abstract*—Learning nonlinear functions from input-output data pairs is one of the most fundamental problems in machine learning. Recent work has formulated the problem of learning a general nonlinear multivariate function of discrete inputs, as a tensor completion problem with smooth latent factors. We build upon this idea and utilize two ensemble learning techniques to enhance its prediction accuracy. We showcase the effectiveness of the proposed ensemble models on several regression tasks and report significant improvements compared to the single model.**

***Index Terms*—Supervised Learning, Tensor Completion, Ensemble Learning, Canonical Polyadic Decomposition**

## I. Introduction

Function approximation is one of the core tasks in machine learning. Every supervised learning problem can be viewed as a function approximation problem where we are given input and output data pairs and we seek to learn the true data generating function. Commonly used algorithms for approximating nonlinear functions are based on deep neural networks, kernel methods and decision trees [1].

Tensor decomposition is another class of methods which have been proposed for learning nonlinear functions. For example, Polynomial Networks (PN) and Factorization Machines (FM) [2], [3] use the Canonical Polyadic Decomposition (CPD) to model low-rank polynomial functions. Other tensor models such as the Tensor Train (TT) [4] and the Tucker model [5], have also been used for parameterizing polynomial functions [6], [7]. Most of these methods are limited to low-order polynomials which may be very restrictive in practice. Apart from polynomials, TT has been used for learning more general functions using a multidimensional Fourier series expansion [8]. Recently, a single high-order CPD model was used for modeling a multivariate function with discrete inputs [9]. The problem was cast as a tensor completion problem with smooth latent factors to account for ordinal input. Under certain conditions, CPD is unique and identification of the true underlying mapping is possible, leading to accurate predictions for new unseen data. A drawback of the Canonical System Identification (CSID) algorithm proposed in [9] is that it cannot be directly applied to continuous data as it requires discretization of the input. Using a coarse discretization makes the algorithm computationally efficient but can lead to models with high bias that underfit the data and have poor generalization. Fine discretization on the other hand, improves expressiveness but the algorithm exhibits higher computational cost and can lead to overfitting.

In this work, we investigate the use of ensemble learning techniques to enhance the prediction accuracy of the CSID model. Ensemble learning is the process of combining multiple learning algorithms into one predictive model to reduce the variance and/or bias of the predictions and obtain better performance [10]. Ensemble methods can be divided into two main groups, parallel and sequential. Bagging also known as bootstrap aggregation is a parallel ensemble method where multiple base models are trained in parallel on different subsets of the data that have been chosen randomly with replacement from the original training data. The output of these models is usually combined and a single prediction is computed using averaging. One of the most popular bagging techniques is random forests [11]. Boosting is a sequential ensemble method where a sequence of base models are fit sequentially to modified versions of the data. Popular boosting algorithms include AdaBoost [12] and Gradient Boosting [13].

We develop two approaches based on these ensemble learning techniques for learning multivariate functions using the Canonical Polyadic Decomposition. We use the CSID algorithm as our base model. We show that ensemble learning can enhance the prediction accuracy of the CSID model and also counter the performance degradation resulting from the discretization step when applied to functions with continuous inputs. We train our models using Stochastic Gradient Descent (SGD) and evaluate their performance on several regression tasks.

**Notation**. We use the symbols $x$, $\mathbf{x}$, $\mathbf{X}$, $\mathcal{X}$ for scalars, vectors, matrices, and tensors respectively. We use the notation $\mathbf{x}(i)$, $\mathbf{X}(i,j)$, $\mathcal{X}(i,j,k)$ to refer to a particular element of a vector, matrix and a tensor. Symbol $\circ$ denote the outer product.

## II. Background

### A. Canonical Polyadic Decomposition

The Canonical Polyadic Decomposition expresses an $N$-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as a sum of rank-1 tensors $\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r^1 \circ \mathbf{a}_r^2 \circ \cdots \circ \mathbf{a}_r^N$, where $\mathbf{a}_r^n \in \mathbb{R}^{I_n}$, and $\circ$ denotes

The work of the authors was supported in part by ECCS-1852831, IIS-1704074.

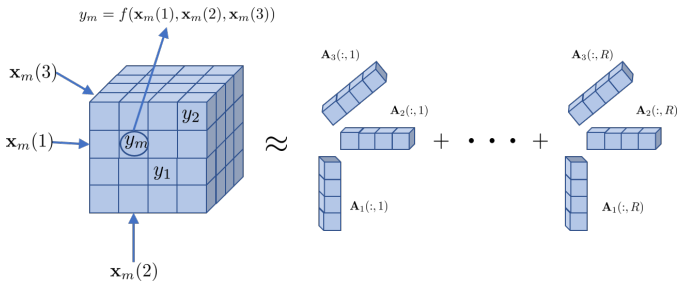


Fig. 1. Canonical System Identification.

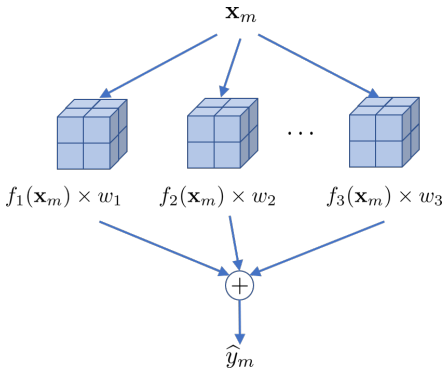


Fig. 2. Ensemble Tensor Completion.

the vector outer product[1]. By defining factor matrices $\mathbf{A}_n = [\mathbf{a}_1^n \cdots \mathbf{a}_R^n] \in \mathbb{R}^{I_n \times R}$, the individual elements of the tensor $\mathcal{X}$ can be expressed as

$$\mathcal{X}(i_1, \ldots, i_N) = \sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{A}_n(i_n, r). \tag{1}$$

The CPD model allows us to approximate a high-dimensional tensor of size $\prod_{n=1}^{N} I_n$ using only order of $(\sum_{n=1}^{N} I_n)R$ parameters. CPD is one of the most popular tensor decomposition models mainly due to its identifiability properties [14].

### B. Canonical System Identification

Canonical System Identification (CSID) is a CPD based method which casts the problem of nonlinear function approximation as a tensor completion problem [9]. Let $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$ be a dataset of $M$ input-output pairs where $\mathbf{x}_m \in \mathbb{R}^N$ and $y_m \in \mathbb{R}$. CSID requires that all predictors are discrete and take values from a predictor-specific finite alphabet $\mathcal{I}_n = \{1, \ldots, I_n\}$, $n = 1, \ldots, N$. The scalar output $y_m$ is a nonlinear function of the input $\mathbf{x}_m$, i.e.,

$$y_m = f\left(\mathbf{x}_m(1), \ldots, \mathbf{x}_m(N)\right). \tag{2}$$

The nonlinear function $f(\cdot)$ can be viewed as an $N$-way tensor $\mathcal{X}$ where each input vector $[\mathbf{x}_m(1), \ldots, \mathbf{x}_m(N)]$ is a cell multi-index and the cell content is the estimated response of the system $\widehat{y}_m$. Additional assumptions need to be placed on $\mathcal{X}$ such that the model is able to generalize to new data points. Specifically, CSID casts the problem as a tensor completion problem where we seek the principal components of the nonlinear function while penalizing higher tensor ranks. The problem of finding the rank-$R$ approximation which best fits the data is formulated as

$$\begin{aligned} \min_{\{\mathbf{A}_n\}_{n=1}^N} \frac{1}{M} \sum_{m=1}^{M} \left(y_m - f\left(\mathbf{x}_m; \{\mathbf{A}_n\}_{n=1}^N\right)\right)^2 \\ + \sum_{n=1}^{N} \rho \|\mathbf{A}_n\|_F^2 + \sum_{n=1}^{N} \mu \|\mathbf{T}_n \mathbf{A}_n\|_F^2, \end{aligned} \tag{3}$$

[1]Note that in our notation superscripts are indices, not exponents.

where $f\left(\mathbf{x}_m; \{\mathbf{A}_n\}_{n=1}^N\right) = \sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{A}_n(\mathbf{x}_m(n), r)$. We use Frobenius norm regularization, and $\mathbf{T}_n$ is a smoothness promoting matrix defined as $\mathbf{T}_n \in \mathbb{R}^{(I_n-1)\times I_n}$ with $\mathbf{T}_n(i,i) = 1$ and $\mathbf{T}_n(i, i+1) = -1$ or $\mathbf{T}_n \in \mathbb{R}^{(I_n-2)\times I_n}$ with $\mathbf{T}_n(i,i) = -1$, $\mathbf{T}_n(i, i+1) = 2$ and $\mathbf{T}_n(i, i+2) = -1$. The model is visualized in Figure 1.

Optimization problem (3) is a tensor completion problem with smooth latent factors and it is naturally suited for discrete ordinal ($\mu > 0$) or categorical ($\mu = 0$) predictors. When the inputs of the system are continuous, the performance of the model highly depends on the discretization method used, as well as the number of discretization intervals. Coarse discretization is preferable for building computationally efficient models but can have poor generalization if the discretization intervals are not carefully selected. In the following section we show how ensemble learning can boost the performance of the baseline model, while mitigating discretization effects at the same time.

## III. Proposed Approach

### A. Bagging

Our first approach is a Bagging method. They key idea of bagging is that different training data sets are created by sampling the original dataset usually with replacement and a set of models are trained in parallel [1]. During testing, Bagging estimates the output of the system for a new data point by averaging the predictions of the individual models.

We propose training a group of $K$ different CSID models each one on datasets that have been obtained by sampling the original dataset with replacement after applying a specific discretization method. The different models need to be sufficiently diverse so we choose between 3 discretization mechanisms, namely, uniform in which all intervals for each input have identical widths, quantile in which all intervals for each feature have the same number of points, and K-means.

**Algorithm 1** Bagging CSID Ensemble

**Input:** $\mathbf{X}$, $\mathbf{X}_{\text{test}}$, $\mathbf{y}$, $\mathbf{y}_{\text{test}}$, $R$, $K$
**for** $k = 1$ **to** $K$ **do** in parallel:
  $\mathbf{X}^k \leftarrow$ sample points from $\mathbf{X}$ with replacement
  $\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}} \leftarrow$ Split dataset $\mathbf{X}^k$
  $\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}} \leftarrow$ Discretize $\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}}$
  Solve optimization problem (3) using SGD
**end for**
Compute $w_k$ via (5)
Combine predictions via (4)

All models are trained by solving the Optimization problem (3) using SGD. We compute the final estimate as

$$f_{\text{CSID}-\text{Bag}}(\mathbf{x}) = \sum_{k=1}^{K} w_k f_k\left(\mathbf{x}; \{\mathbf{A}_n^k\}_{n=1}^N\right), \tag{4}$$

where $w_k$ are positive weights that measure the importance of each model for the final prediction (Fig. 2). Each CSID model is parameterized by different factor matrices $\{\mathbf{A}_n^k\}_{n=1}^N$ and is trained in parallel. We propose combining the individual models by weighting each one based on the error obtained from the validation set. In this way, models that do not perform well on the validation set do not affect the predictions as it is likely that they will not perform well on the training set. Specifically, we define $w_k$ to be

$$w_k = \frac{1/\text{Err}_\text{k}}{\sum_{k=1}^{K} 1/\text{Err}_\text{k}}. \tag{5}$$

where $\text{Err}_\text{k}$ is the Root Mean Square Error (RMSE) of the $k$-th model on the validation set. A high level description of the procedure is provided in algorithm 1.

The advantage of Bagging is that it is computational efficient as each model can be trained independently. On the other hand, when training a model it may be beneficial to take into account information from the other models. Our second approach is a sequential ensemble method which is computationally less efficient but each model uses knowledge obtained from the previous ones.

### *B. Boosting*

Forward Stage-wise Additive Modeling (FSAM) is a boosting algorithm where the individual models are built sequentially [1]. Similarly with Bagging, the output of the ensemble model is given by combining the predictions of the individual models

$$f_{\text{CSID}-\text{Boost}}(\mathbf{x}) = \sum_{k=1}^{K} f_k\left(\mathbf{x}; \{\mathbf{A}_n^k\}_{n=1}^N\right), \tag{6}$$

The major difference is that the models are fit on the prediction errors. Specifically, at each iteration $k$, we compute $\{\mathbf{A}_n^k\}_{n=1}^N$

**Algorithm 2** Boosting CSID Ensemble

**Input:** $\mathbf{X}$, $\mathbf{X}_{\text{test}}$, $\mathbf{y}$, $\mathbf{y}_{\text{test}}$, $R$, $K$
$\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}} \leftarrow$ Split dataset $\mathbf{X}$
$\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}} \leftarrow$ discretize$\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}}$
Set $k = 1$ and solve optimization problem (7) using SGD
Compute prediction errors
**for** $k = 2$ **to** $K$ **do**:
  $\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}} \leftarrow$ discretize $\mathbf{X}^k_{\text{train}}, \mathbf{X}^k_{\text{val}}$
  Solve optimization problem (7) using SGD
  Add model $k$ to the expansion
  Compute prediction errors
**end for**
Compute predictions via (6)

by solving the following optimization problem

$$\begin{aligned}\min_{\{\mathbf{A}_n^k\}_{n=1}^N} \quad & \frac{1}{M}\sum_{m=1}^{M}\left(y_m - \widehat{y}_m^{k-1} - f_k\left(\mathbf{x}_m; \{\mathbf{A}_n^k\}_{n=1}^N\right)\right)^2 \\ & + \sum_{n=1}^{N}\rho\|\mathbf{A}_n^k\|_F^2 + \sum_{n=1}^{N}\mu\|\mathbf{T}_n\mathbf{A}_n^k\|_F^2\end{aligned} \tag{7}$$

where

$$\widehat{y}_m^{k-1} = \sum_{k'=1}^{k-1} f_k\left(\mathbf{x}_m; \{\mathbf{A}_n^{k'}\}_{n=1}^N\right). \tag{8}$$

At each iteration the model that best fits the current errors is added to the expansion. The key idea is that each subsequent model focuses on data points that are difficult to predict. A high level description of the procedure is provided in algorithm 2.

## IV. Numerical Tests

We evaluate the proposed methods in regression tasks using several datasets obtained from the UCI machine learning repository [15]. Our proposed approach is implemented in Python using PyTorch [16]. For each experiment we split the dataset into two sets, $85\%$ used for training and $15\%$ for testing and we tune the hyper-parameters using $5$-fold cross-validation. We compare the performance of the different algorithms in terms of the RMSE.

We compare the ensemble models against a single CSID model. We combine $10$ CSID models to build two ensembles based on Bagging and FSAM. All the methods are trained using SGD and Adam optimizer with a learning rate $10^{-2}$ for a maximum of $50$ epochs. We fix the alphabet size to be $I = 20$ and discretize all contintuous inputs. Dataset information is shown in table II.

Table I shows the RMSE performance of the different methods. We observe that significant gains are obtained in $3$ out of $4$ datasets when combining multiple base models compared to the single one. Figure 3 and Figure 4 show the RMSE performance of the two methods while the number of the models increases for two of the datasets.

TABLE I
COMPARISON OF RMSE PERFORMANCE OF DIFFERENT MODELS ON MULTI-OUTPUT REGRESSION.

| Dataset | CSID | CSID-Bag (10) | CSID-Boost (10) |
|---|---|---|---|
| QSAR | 1.51 | 1.37 | 1.49 |
| CCS | 6.25 | 5.69 | 5.46 |
| CPP | 4.22 | 3.89 | 3.97 |
| PP | 4.29 | 3.95 | 3.98 |

TABLE II
COMPARISON OF RMSE PERFORMANCE.

| Dataset | N | M |
|---|---|---|
| QSAR AQUATIC TOXICITY (QSAR) | 8 | 546 |
| CONCRETE COMPRESSIVE STRENGTH (CCS) | 9 | 1030 |
| CYCLE POWER PLANT (CPP) | 4 | 9568 |
| PHYSICOCHEMICAL PROPERTIES (PP) | 9 | 45730 |

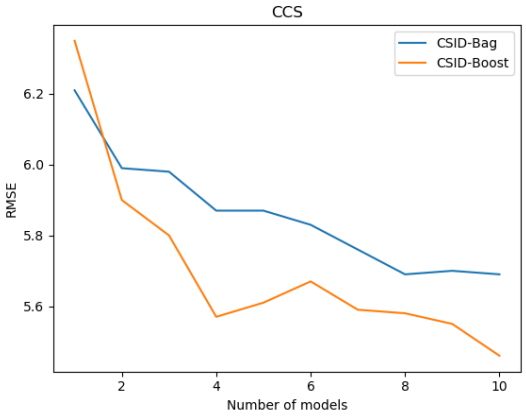


Fig. 3. RMSE performance in CCS dataset.

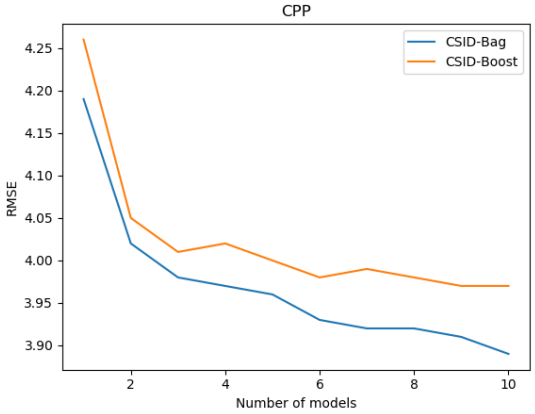


Fig. 4. RMSE performance in CPP dataset.

## V. CONCLUSION AND FUTURE WORK

In this paper, we proposed two ensemble techniques for improving the prediction performance of a CPD based function learning method, particularly for functions with continuous inputs. We presented two approaches based on Bagging and Boosting and reported results showing good performance improvements on several real data regression tasks.

## REFERENCES


[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics, 2009.

[2] S. Rendle, "Factorization machines," in *Proceedings of the IEEE International Conference on Data Mining*, Dec 2010, pp. 995–1000.

[3] M. Blondel, M. Ishihata, A. Fujino, and N. Ueda, "Polynomial networks and factorization machines: New insights and efficient training algorithms," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 850–858.

[4] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[5] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[6] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines," in *International Conference on Learning Representations Workshop*, 2016.

[7] I. Perros, F. Wang, P. Zhang, P. Walker, R. Vuduc, J. Pathak, and J. Sun, "Polyadic regression and its application to chemogenomics," in *Proceedings of the SIAM International Conference on Data Mining*, 2017, pp. 72–80.

[8] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen, "Learning multidimensional fourier series with tensor trains," in *Proceedings of the IEEE Global Conference on Signal and Information Processing*, 2014, pp. 394–398.

[9] N. Kargas and N. D. Sidiropoulos, "Nonlinear system identification via tensor completion," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

[10] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm." Citeseer, 1996.

[13] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[14] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[15] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf