Context-Aware Ensemble Learning for Time Series

Arda Fazla, Mustafa E. Aydin, Orhun Tamyigit and Suleyman S. Kozat, Senior Member, IEEE

Abstract—We investigate ensemble methods for prediction in
an online setting. Unlike all the literature in ensembling, for the
first time, we introduce a new approach using a meta learner
that effectively combines the base model predictions via using
a superset of the features that is the union of the base models’
feature vectors instead of the predictions themselves. Here, our
model does not use the predictions of the base models as inputs
to a machine learning algorithm, but choose the best possible
combination at each time step based on the state of the problem.
We explore three different constraint spaces for the ensembling
of the base learners that linearly combines the base predictions,
which are convex combinations where the components of the
ensembling vector are all nonnegative and sum up to 1; affine
combinations where the weight vector components are required
to sum up to 1; and the unconstrained combinations where the
components are free to take any real value. The constraints are
both theoretically analyzed under known statistics and integrated
into the learning procedure of the meta learner as a part of the
optimization in an automated manner. To show the practical
efficiency of the proposed method, we employ a gradient-boosted
decision tree and a multi-layer perceptron separately as the
meta learners. Our framework is generic so that one can use
other machine learning architectures as the ensemble as long as
they allow for a custom differentiable loss for minimization. We
demonstrate the learning behavior of our algorithm on synthetic
data and the significant performance improvements over the
conventional methods over various real life datasets, extensively
used in the well-known data competitions. Furthermore, we
openly share the source code of the proposed method to facilitate
further research and comparison.

Index Terms—ensemble learning, regression, time series, mul-
tilayer perceptron (MLP), LightGBM

I. INTRODUCTION

We study regression (prediction, learning and similar) in
time series settings where we receive a data sequence
related to a target signal and estimate the signal’s next values in
an online manner. This problem is extensively studied in the
machine learning, computational learning theory and signal
processing literatures under different names [1, 2, 3]. An
effective way in such problems as shown in many recent
competitions [4, 5] is using ensemble models for improved
performance [6], where many base predictor models are first
trained separately or successively and then inputted to a
merging strategy to get the final prediction. Since there seldom
exists a true data generating process in practical time series
applications, the ensembling of individual predictors tend to
produce better performance than the individual models using
diversity, as well known in the machine learning literature [7].

Common strategies for ensembling include boosting [8] and
bootstrap aggregation [9] where the base models are intention-
ally kept weak either by constraining them by design to under-
fit or by restricting the amount of data samples and/or features
used each base model. The resultant ensembled prediction,
however, is a direct average or sum of the predictions of the
base models, e.g., taking the mean of the weak predictions
in regression, or taking the majority vote in classification.
Furthermore, the base models in these “built-in” ensemblers
are often chosen to be the same predictor in practice, e.g.,
in random forests [10], gradient-boosted decision trees [8]
and extra-trees [11], hard decision trees [12] are the base
predictors. This not only prevents them from using diverse
base regressors but also from providing online prediction as the
hard decision trees are not differentiable and therefore require
periodical re-fitting against a real-time stream of data, which
might be time consuming and/or computationally infeasible.

Apart from these built-in ensembling mechanisms, a “meta
learner” approach is also used for combining predictions [13].
In that approach, a separate machine learning model, the meta
learner, is trained either over the predictions of the base models
or the features extracted from the target sequence, where the
output of the meta learner is commonly tailored to mimic
a weighted average, i.e., the output vector has its values all
nonnegative and sum up to 1; the resultant prediction is then
a linear combination of the base predictions. When using the
predictions of the base models as the features to the meta
machine learning model, however, the diversity of the base
predictions should be satisfied as otherwise the correlated
features might hinder (as always hinder as we show in our
simulations) the learning process of the meta learner [14].
In practice, it might be hard to find diverse-enough base
predictors, which renders this approach less practical [15].
Moreover, even when using other features as input to the
ensemble-learning algorithm in addition to the predictions of
the base learners, e.g., those extracted from the time series,
the base predictions themselves tend to dominate the other
features and the correlation issue re-arises [16]. Additionally,
the ensembling models that do not consider the features the
base predictors use tend to produce nonoptimal predictions
as they lack the proper context in which these predictions
are produced, i.e., some base models can work in certain
states better than the others; however, this information is lost
in the ensembling process, since it is oblivious to this state
information. The weights in such combinations therefore tend
to be biased towards the prediction of the “best” base predictor,
which in turn causes meta learner to overfit to the said base
model.

Another important aspect in linear combinations of the
predictions is the constraint space of the ensembling weight
vector. In the general case, the weight vector is unconstrained,

A. Background

...
i.e., its values can take any real valued number. When the
statistics of the combined predictions as well as the target
signal are known, naturally, the unconstrained weights achieve
the lowest possible error [17]. In practice, however, those
statistics are rarely known, and the unsurpassability of the
unconstrained combinations is subject to the learning pro-
cedures, i.e., whether all the parameters will be correctly
learnt or not, and therefore not guaranteed. In fact, since
the parameter space is the entire Euclidean space of ap-
propriate dimension, unconstrained combinations might lead
to overfitting in ensembling. To this end, we also consider
two more combining strategies over the weight vector in
order to generate a “regularization” effect. These strategies
are affine combinations where the weight vector is required to
have its components sum up to 1 and convex combinations
which builds on the affine combination by also requiring
nonnegativity of the ensembling weights. We emphasize that
nondifferentiable ensembler models, e.g., a direct average or
boosted trees, are not flexible enough to satisfy the latter two
constraints in an automated manner [18].

Here, we effectively combine the base predictions in a
context-dependent and base model-agnostic manner. To this
end, we propose a machine learning-based meta learner that
uses a superset of features in training, which equals or extends
the concatenated feature sets of the base models and does
not include the predictions themselves as the features to
the ensemble-learner. For the linear combination of the base
predictions, the ensembler outputs a weight vector which is
either unconstrained or amenable to satisfy the affine or convex
constraints, all in an online manner. Our motivation is to use
the superset feature vector in order to make the combining
model to be “context aware” and less prone to overfitting to
a prominent base model, while exploring various constraint
spaces to allow for a regularization effect. In particular, we
employ a LightGBM (a gradient boosting machine [19])
and a neural network (a multilayer perceptron [20]) as the meta
learners to show the efficacy of the proposed ensembling
approach while emphasizing that the framework is generic
enough such that any machine learning model capable of
minimizing a custom differentiable loss can be used.

B. Related Work

Ensemble models are heavily investigated in machine learn-
ing, computational learning theory, statistics and signal pro-
cessing [21], [22], since they provide superior performance
due to using diversity. There are several methods for ensem-
bles, for example, one can train base models independently
from each other in parallel and then linearly combine their
predictions by a deterministic process, which is called bagging
[9]. In boosting [8], base models are trained sequentially,
with each model solely focusing on correcting the errors
made by the previous ones. On the contrary, the stacking
method [22] relies on a separate machine learning model to
estimate the linear combination weights of the base model
predictions. The main issue with conventional ensembling
methods is the process of combining predictions linearly or
with some other final meta machine learning algorithm. In
most cases, predictions are blended using a simple averaging
process, and even if a separate machine learning model is
used for learning the combination weights, these weights are
learned according to the errors of the base models in the
training dataset. Thus, ensemble models are prone to suffer
from the underfitting and overfitting issues present in the base
models [21]. As a remedy, several adjustments to the ensemble
methods have been proposed, such as employing different
preprocessing techniques for each base model, mixing the
training data with independent noise for each model, using
a different partition of the training data for each model, etc.
[23]. All of these approaches are sub-optimal solutions, as the
ensemble learner still only considers the errors of the base
models, which as we show in our simulations, is inadequate
for the combination weight vector to converge to the optimal
weights. The linear combination weights should be produced
with respect to the data specific side information vector at
each sample to exploit the diversity and avoid the learning
difficulties due to high correlations, which we support with
our simulations. This is natural since real life time series
data often contain different patterns depending on the sample
time or some data specific side information vector [24]. For
example, a daily sales data, such as the M5 Forecasting
Dataset [5], may show different patterns for weekdays and
weekends. Thus, the linear combination of the base model
predictions should be learned according to the day of the
week information. A study of the short term forecasting of
gas demand in Italy [25] illustrates that, the daily industrial
gas demand consumption shows different patterns depending
on the weather temperature, thus, weather temperature should
also be fed to the ensemble learner. Therefore, we introduce
a scheme where the ensemble model learns to combine base
model predictions at each time sample, by learning the relation
between the errors of the base models and the data specific
side information vector.

There have been previous attempts to linearly combine base
model predictions based on a certain side information vector.
The second place winners of the M4 Forecasting Competi-
tion [26] use the LightGBM model [19] as their ensemble model,
which learns to produce weights at each sample time, by
solely relying on the given side information vector. However,
even though the ensemble model predictions rely on the data
specific side information vector, the model is still trained
based on the errors of the base models in the training dataset.
Thus, possible training problems of the base models, such
as overfitting, can mislead the ensemble learner to produce
nonoptimal combination weights during the test dataset.

In order to eliminate any training related issues, we intro-
duce a novel training scheme, where the training dataset is
partitioned into two distinct sets. The base models are trained
on the first partition, and then the ensemble learner is trained
by their features, state of the problem and their errors on the
second partition of the training dataset. Thus, we alleviate
training specific issues by the base models, as these issues are
not leaked to the ensemble model. Another problem with the
given model is that, each weight assigned to any base model is
bound to be nonnegative, and the summation of all of the base
model weights for each sample is bound to sum up to 1, which
is the convex constraint. The motive behind this approach is reducing the learning complexity of the ensemble learner. However, as we illustrate in our simulations, the convex constraint is not always the best solution for every time series data. Hence, we solve the linear weight combination problem under three different constraints, where the weights can be either affine constrained, convex constrained or unconstrained. Therefore, we introduce a more comprehensive solution, where the weight constraint employed can be selected according to the given dataset. As a result, with our novel ensemble approach, we significantly improve the prediction performance compared to the base models and conventional ensembling methods for various datasets, as illustrated in our simulations including both well-known datasets in various competitions and artificially generated datasets.

C. Contributions

Our contributions are as follows:

- For the first time in the literature, we tackle the problem of finding the optimal combination weight vectors to ensemble base predictors based on a sequential data specific side information vector, under three different weight constraints. To learn the optimal time dependent and side information dependent weights, we present two novel and generic ensembling algorithms, based on boosted decision trees [8] and neural networks [20], where we derive all the related equations. Note that our approach is generic as any such universal learner can be used accordingly.
- We introduce a novel ensemble training scheme, where we alleviate any training related issues by the base models, such as co-linearity, high correlation and overfitting by the base algorithms.
- With various experiments containing synthetic and real-life sequential data from the well-known competitions, we illustrate the superiority of our approach over the base models used in the experiments and the conventional ensembling methods.
- We publicly share our code for both model design, comparisons and experimental reproducibility.

D. Organization

The rest of the paper is organized as follows. In Section II, we introduce the problem of finding the optimal base model combination weights based on the side information vector, under different constraints. In Section III, we first analyze the associated learning costs of the optimal combination weight vectors, under all three constraints. Next, we introduce two novel and generic machine learning algorithms to find the optimal weights, along with a new ensemble training scheme. We then illustrate the performance of our proposed algorithms via extensive experiments involving real life and synthetic data in Section IV. We then conclude our paper with remarks in Section V.

1https://github.com/ardafazla/context-aware-ensemble

II. PRELIMINARIES

A. Problem Statement

In this paper, all vectors are real column vectors and are presented by boldface lowercase letters. Matrices are denoted by boldface uppercase letters. \( x^{(k)} \) and \( x_t^{(k)} \) denotes the \( k \)-th element of the vectors \( x \) and \( x_t \), respectively. \( x^T \) represents the ordinary transpose of \( x \). \( X_{i,j} \) represents the entry at the \( i \)-th row and \( j \)-th column of the matrix \( X \).

We study the online prediction of sequential data using a linear mixture of different learning models. The main sequence we aim to predict is \{\( y_t \); to this end, we employ \( M \) (online) learning models, called base models, each of which uses a side information sequence \{\( s_k^{(i)} \)\} for \( k \leq t \) and \( i = 1, \ldots, M \). We emphasize that each base model is free to use a different side information vector, e.g., \{\( s_k^{(i)} \)\} could include the observed past information \{\( y_k \)\} along with model-dependent features. At each time \( t \), all the base models produce predictions \( \tilde{y}_t^{(i)} \). We combine these \( M \) predictions using a linear scheme such that the ensemble prediction \( \tilde{y}_t^{E} \) of \( y_t \) is given as

\[
\tilde{y}_t^{E} = w_t^T \tilde{y}_t,
\]

where \( \tilde{y}_t = [\tilde{y}_t^{(1)}, \ldots, \tilde{y}_t^{(M)}]^T \) is the base prediction vector of size \( M \) consisting of the individual scalar predictions of the base models, and \( w_t \in \mathbb{R}^M \) is the ensemble weight vector at time \( t \). We note that even though the ensembling scheme is linear, the adaptive learning procedure that produces \( w_t \) could be highly nonlinear as we show in Section III-B. Note that other highly nonlinear combination methods including MLPs and decision trees are shown to be highly inadequate in learning the correlated relations as shown in our experiments. When we observe \( y_t \), we suffer the loss

\[
L_t = \ell(y_t, \tilde{y}_t^{E}) = \ell(y_t, w_t^T \tilde{y}_t) = \ell(y_t, \sum_{i=1}^{M} w_t^{(i)} \tilde{y}_t^{(i)}),
\]

where \( \ell \), for example, can be the squared error loss.

Conventionally, the learning of the weight vector \( w_t \) depends only on the base models’ predictions, i.e., \( w_t = f(\tilde{y}_t) \) for some function \( f \) representing a learning model [27], [28]. However, a disadvantage with this approach is that when the base predictions are highly correlated (e.g., due to similar algorithms or outputs), the learning procedure is hindered [14]. Therefore, highly diverse and independent base models are a prerequisite for such ensembling. Furthermore, this approach does not use any side information in the ensembling model either, which could result in nonoptimal combinations, as, naturally, even with any learning issues,

\[
E[\ell(y_t, \tilde{y}_t^{E}) | s_t^{E}] \leq E[\ell(y_t, \tilde{y}_t^{E})],
\]

where \( s_t^{E} \) is the side information vector the ensembling model uses at time step \( t \). We note that the loss function \( \ell \) in (3) is generic; therefore, (3) implies that the expected loss at time \( t \) cannot go higher when using the extra side information vector. To this end, we propose to use \textit{not} the base predictions themselves as inputs to the ensembling model but instead...
We also emphasize that, as shown in Fig. 1, we do compared to the conventional method is illustrated in Fig. 1. where the union of vectors $\bigcup_{i=1}^{M} s_{t}^{(i)}$, i.e., it is at least

$$s_{t}^{E} = \bigcup_{i=1}^{M} s_{t}^{(i)}, \quad (4)$$

where the union of vectors $\{s_{t}^{(i)}\}$ corresponds to concatenating of them while not allowing for duplicate features. We note that $s_{t}^{E}$ may have more features than this union. Our approach compared to the conventional method is illustrated in Fig. 1. We also emphasize that, as shown in Fig. 1, we do not use the base predictions next to $s_{t}^{E}$ as inputs to the ensembling model. The rationale behind this is two folds. Firstly, the base predictions might be highly correlated and as a result, the combining model might struggle in learning the weights. Secondly, when used along with other features, i.e., $s_{t}^{E}$ in this case, the base predictions tend to dominate the other features and the ensemble model would mostly ignore the contributions from possibly valuable side information vectors for ensembling. In order to emphasize that the learned ensembling weight vector in our framework is side information dependent, we denote it via $w_{s_{t}^{E}}$.

As an example of the process, in hourly wind power now-casting, one could employ two parallel-working base models. The first model could use the past 7 hours of wind power, current wind speed and wind direction, i.e., 3 features in total, while the second model could use past 24 hours of wind power and current wind direction, i.e., 2 features in total, as side information. The ensembling model, then, would use 4 features in total, which are the past 7 and 24 hours of the target signal, wind speed and wind direction. Each base model as well as the ensembling model updates itself as soon as new observations of the next hour become available, i.e., they work in an online manner.

We investigate three different mixture approaches which are identified by how they constrain the combining weight vector $w_{s_{t}^{E}}$: these are the **unconstrained** linear combination where each $w_{s_{t}^{E}}^{(i)}$ is free to take any real value, the **affine** linear combination where the components of the weight vector sum up to 1, i.e., $w_{s_{t}^{E}}^T \mathbf{1} = 1$ and the **convex** linear combination where the weight vector not only has its components sum to 1 but also all nonnegative, i.e., $w_{s_{t}^{E}}^T \mathbf{1} = 1$ and $w_{s_{t}^{E}}^{(i)} \geq 0 \forall i$; all of these constraints apply to all time steps.

Hence, in a purely online setting, we aim to adaptively find the “best” weights that linearly combine the predictions of $M$ parallel running base models to minimize a given loss $\ell$ at each time step $t$ using the super feature set $s_{t}^{E}$ under three different constraints imposed on the weight vector. In the next section, we present an analytical overview on the constraints as well as various ensemble algorithms to find the corresponding ensembling weights.

### III. Ensemble Learning

In this section, we first analyze the optimal ensembling vectors in the expectation sense as well as the associated costs for all three constraints on the weight vector; namely, unconstrained, affine and convex combinations. Next, we introduce novel ensembling machine learning algorithms including the mathematical derivations based on boosted decision trees and neural networks in a generic framework to find the optimal weights.

#### A. Analysis on Constraints

We first analyze the optimal weight vectors for ensembling as well as the associated costs under known statistics for each of the three constraints imposed on the ensembling weight vector $w_{s_{t}^{E}} \in \mathbb{R}^{M}$ where $M$ is the number of base models in the ensemble. Let $C(s_{t})$ be the conditional auto correlation matrix at time $t$ of the base prediction vector $\hat{y}_{t}^{(M)}$, given the super set of the side information vectors used by all the base algorithms and specific to the ensemble, and let $\alpha(s_{t})$ be the conditional cross correlation vector at time $t$ between the target signal $y_{t}$ and the base prediction vector given the super set side information, i.e.,

$$C(s_{t}) = E[\hat{y}_{t} \hat{y}_{t}^T | s_{t}^{E}] \quad (5)$$

$$\alpha(s_{t}) = E[y_{t} \hat{y}_{t} | s_{t}^{E}] \quad (6)$$

In the unconstrained case, the components of $w_{s_{t}^{E}}$ are free to take any real number; therefore, the corresponding optimization problem given $s_{t}^{E}$ for a given differentiable loss function $\ell$ is

$$\min_{w_{s_{t}^{E}} \in \mathbb{R}^{M}} \ell(y_{t}, \sum_{i=1}^{M} w_{s_{t}^{E}}^{(i)} \hat{y}_{t}^{(i)}) \quad (7)$$

where the feasible region is the entire $M$ dimensional Euclidean space. If the statistics $C(s_{t})$ and $\alpha(s_{t})$ are known at all times, the optimal unconstrained ensembling vector at time $t$ under the expected squared error loss, i.e., when $\ell(y_{t}, \hat{y}_{t}) = E[(y_{t} - \hat{y}_{t})^2 | s_{t}^{E}] = E[(y_{t} - \sum_{i=1}^{M} w_{s_{t}^{E}}^{(i)} \hat{y}_{t}^{(i)})^2 | s_{t}^{E}]$, is the solution to the normal equations and given as

$$w_{s_{t}^{E}}^{*\text{unc}} = C(s_{t})^{-1} \alpha(s_{t}). \quad (8)$$

Furthermore, assuming the target signal has the conditional variance $\sigma(s_{t})^2$ at time $t$, the conditional mean squared error of the unconstrained ensembling is given by

$$L_{t,\text{unc}}^* := \sigma(s_{t})^2 - \alpha(s_{t})^T C(s_{t})^{-1} \alpha(s_{t}). \quad (9)$$
In the affine constrained case, the components of the weight vector are required to sum up to 1. This translates to the optimization problem given as

$$\min_{w_{st}^F \in \mathbb{R}^M} \ell(y_t, \sum_{i=1}^M w_{st}^{(i)} \hat{y}_{t}^{(i)})$$

subject to

$$w_{st}^F 1 = 1,$$  \hspace{1cm} \text{(10)}

where the feasible region is now an $M - 1$ dimensional hyperplane in $\mathbb{R}^M$. In fact, this problem could be cast as an $M - 1$-dimensional unconstrained optimization over $\tilde{w}_{st} \in \mathbb{R}^{M-1}$ such that the $M^{th}$ component of the weight vector complements the sum of the entire vector to be 1, i.e.,

$$\min_{\tilde{w}_{st} \in \mathbb{R}^{M-1}} \ell(y_t, \sum_{i=1}^{M-1} \tilde{w}_{st}^{(i)} \hat{y}_{t}^{(i)})$$

subject to

$$\tilde{w}_{st}^{(M)} = 1 - \tilde{w}_{st}^T 1.$$  \hspace{1cm} \text{(11)}

With the transformation of the constrained problem from (10), which is a linearly constrained quadratic optimization problem under the squared loss, i.e., $\ell(y_t, \hat{y}_{t}) = \mathbb{E}(\|y_t - \hat{y}_{t}\|^2 | s_t^F)$, to (11), which is an unconstrained problem, the optimal affine ensembling weight vector admits a closed form solution given as

$$w_{st}^{*F, aff} = (C(s_t)^{-1} a(s_t) - \frac{T^T C(s_t)^{-1} a(s_t) - 1}{1 T^T C(s_t)^{-1}} C(s_t)^{-1} 1).$$

We note that the affine combination preserves the unbiasedness of the base predictors. If each base predictor provides unbiased predictions of $y_t$, then we have

$$\mathbb{E}[w_{st}^T \hat{y}_t] = \sum_{i=1}^M w_{st}^{(i)} \mathbb{E}[\hat{y}_{t}^{(i)}] = \sum_{i=1}^M w_{st}^{(i)} \mathbb{E}[y_t] = \mathbb{E}[y_t].$$

For the expected squared error loss, the optimal affine-constrained weights is given as

$$L_t^{*F, aff} := \sigma(s_t)^2 - a(s_t)^T C(s_t)^{-1} a(s_t) - \frac{(1^T C(s_t)^{-1} a(s_t) - 1)^2}{1 T \sigma(s_t)^{-1}}.$$  \hspace{1cm} \text{(12)}

As for the convex constraint case, we not only require that the components of the weight vector sum up to 1 but also they should be nonnegative, i.e., the optimization problem becomes

$$\min_{w_{st}^F \in \mathbb{R}^M} \ell(y_t, \sum_{i=1}^M w_{st}^{(i)} \hat{y}_{t}^{(i)})$$

subject to

$$w_{st}^F 1 = 1$$

$$w_{st}^{(i)} \geq 0, \hspace{0.5cm} i = 1, \ldots, M.$$  \hspace{1cm} \text{(14)}

Under a convex loss $\ell$, problem (14) is a convex quadratic minimization problem, and the feasible region is the $M$-dimensional unit simplex. Unlike the previous two constraints, (14) does not admit a closed form solution. We can, however, project (unconstrained) weights to the unit simplex iteratively to find the optimal weight vector. For brevity, we let $M = 2$, which is a common case in practice, and derive the procedure under squared error loss. To this end, we begin by noting that the squared loss at time $t$ as a function of the ensembling weight vector $w_{st}^{F}$ is

$$L_t(w_{st}^{F}) = \sigma(s_t)^2 - a(s_t)^T C(s_t)^{-1} a(s_t) + (w_{st}^{F} - C(s_t)^{-1} a(s_t))^T C(s_t)(w_{st}^{F} - C(s_t)^{-1} a(s_t)) = L_t, unc + (w_{st}^{F} - w_{st}^{F, unc})^T C(s_t)(w_{st}^{F} - w_{st}^{F, unc}).$$

We can rewrite (15) instead of the optimal affine weights

$$L_t(w_{st}^{F}) = L_t, unc + (w_{st}^{F} - w_{st}^{F, aff} + (w_{st}^{F} - w_{st}^{F, unc}))^T C(s_t)(w_{st}^{F} - w_{st}^{F, unc})$$

$$= L_t, unc + (w_{st}^{F, aff} - w_{st}^{F, unc})^T C(s_t)(w_{st}^{F, aff} - w_{st}^{F, unc}) - 2(w_{st}^{F} - w_{st}^{F, aff})^T C(s_t)(w_{st}^{F, aff} - w_{st}^{F, unc}) = L_t, aff + (w_{st}^{F} - w_{st}^{F, aff})^T C(s_t)(w_{st}^{F} - w_{st}^{F, aff})$$

$$\frac{(1 - 2)^{T^T} w_{st}^{F, unc} - 1}{1 T \sigma(s_t)^{-1}} (w_{st}^{F} - w_{st}^{F, aff})^T 1.$$  \hspace{1cm} \text{(16)}

where the last line follows from $(w_{st}^{F} - w_{st}^{F, aff})^T 1 = 0$ due to the unit simplex constraint. Further ignoring the constant $L_t, aff$, the optimization problem reduces to minimizing the $C(s_t)$-weighted $\ell_2$-norm of $w_{st}^{F} - w_{st}^{F, aff}$ subject to $w_{st}^{F}$ being on the unit simplex. For the assumed $M = 2$, we let $\Delta_2$ be the unit simplex in $\mathbb{R}^2$. There are 2 cases to consider for this problem:

- $w_{st}^{F, aff} \in \Delta_2$: Optimal affine combination is within the unit simplex set and hence a closed form solution can be written for the optimal weight vector: $w_{st}^{F, con} = w_{st}^{F, aff}$. Accordingly, $L_t, con := L_t, aff$.

- $w_{st}^{F, aff} \notin \Delta_2$: This case is when combination weights are of opposite sign. For simplicity, we assume the weight that is assigned to the first base predictor is negative and the other one is positive. Then, we can write $w_{st}^{F, con} = w_{st}^{F, aff} + \alpha [1 - 1]^T$ for some $\alpha \in \mathbb{R}$. Consequently, the cost function to minimize becomes the $C(s_t)$-weighted $\ell_2$-norm of $\alpha [-1 1]^T$. This implies that the cost is proportional to the magnitude of $\alpha$. The optimal choice would be the smallest value that keeps $w_{st}^{F, con}$ within $\Delta_2$, which is $-w_{st}^{F, aff}$. In this case, $w_{st}^{F, con}$ is the corner point of $\Delta_2$: $[0 1]^T$. This leads to $L_t, con := L_t, aff + [-1 1]^T C(s_t) (-1 1)^T (w_{st}^{11}_F)^2$.  

**Remark.** From the inclusion order of the feasible regions of the constrained problems as well as the derived optimal squared error losses as the special cases, we naturally have

$$L_t, unc \leq L_t, aff \leq L_t, con.$$
for all times $t$. This inequality holds in theory when the correlation statistics of the target signal and the base models are known at all times. In practice, however, these statistics are rarely known and therefore there is a learning cost associated with each constraint, which renders the relation in (10) useless. Hence, even though the unconstrained case seems to promise the least error, the associated learning procedure might lead to overfitting, i.e., in essence, affine and convex constrained cases perform as “regulators” in that they offer a tradeoff between bias and variance of the ensembling model. In fact, as shown in our simulations in Section IV, the unconstrained ensembling scheme does not always achieve the lowest error.

Since $C(s_t)$ and $\alpha(s_t)$ are rarely known in practice, next, we present novel and generic ensembling algorithms to find the optimal ensembling weight vectors under three constraint schemes.

B. Introduced Approach

In order to linearly combine the predictions of different base models by assigning weights to each model, we use ensembling algorithms. Our ensemble algorithms learn to combine the predictions of different base algorithms, by minimizing the loss function given in (2), where $y_t$ is the observed sequence at time $t$, $\ell$ is a given differentiable loss function such as the squared error loss, $\hat{y}_s^{E}$ is the prediction of the ensemble algorithm, $M$ is the number of base models, $\hat{y}_t^{(i)}$ are the predictions of the base models and $w_{s_t^{E}}^{(i)}$ are the weights assigned to the base models, by utilizing the side information sequence $s_t^{E}$.

In order to train the ensemble algorithm, we introduce a novel ensemble training scheme, illustrated in Fig. 2 and explained thoroughly in Algorithm 1. Our training scheme consists of two phases: the offline phase and the online phase. In the offline phase, first, the base models are independently trained on a certain partition of the training dataset, given as $1 \leq t \leq T_1$. Then, the predictions of the base models are gathered on the remaining partition of the training data, given as $T_1 + 1 \leq t \leq T$, where these predictions are then used to train the ensemble algorithm. Hence, as the predictions are gathered on unseen data, we overcome the possible overfitting issue of the base models, which is a well-known issue for the conventional ensembling approaches (11). Each base model $i$ makes predictions based on a side information vector $s_t^{(i)}$, which may consist of the past observations of the sequence $y_t$ along with some additional related information specific to the observed sequence or the base model. Our ensemble model is trained with the errors of the base models, while employing the side information sequence $s_t^{E}$, which may consist of the past observations of the sequence $y_t$, side information related to base models, e.g., errors of the base models, and some additional information specific to the observed sequence. The side information vector of the ensemble model is created as explained in (4).

In the online phase, we use our pre-trained ensemble algorithm from the offline phase to produce combination weights for the base algorithms on the new data, which in our case, is the new observations following the training samples of the sequence $y_t$. First, we retrain the base models on the whole training dataset, given as $1 \leq t \leq T$. Then, we produce the base model predictions for the test dataset, $T + 1 \leq t \leq T_2$. We then update the ensemble side information vector $s_t^{E}$ for the test duration. Finally, we produce the ensemble prediction $\hat{y}_t^{E} = \sum_{i=1}^{M} w_{s_t^{E}}^{i} \hat{y}_t^{(i)}$, for $T + 1 \leq t \leq T_2$.

Algorithm 1 Ensemble Learning Procedure

- **Offline Phase:** Training the Base and Ensemble Algorithms
  
  **inputs:**
  
  - $\{y_t\}_{t=1}^{T}$: the training data that consists of $T$ sequential samples from the time series sequence $y_t$.
  - $M$ base models.
  - $\{s_t^{(i)}\}_{t=1}^{M}$: time series features for the $i^{th}$ base predictor at time $T$.
  
  **outputs:**
  
  - ensemble learner, which produces the weight vector $w_{s_t^{E}}^{i}$ for combining base predictions.
  - $\{s_t^{E}\}_{t=1}^{T}$: time series features for the ensemble learner.
  
  **prepare ensemble data:**
  
  - split $\{y_t\}_{t=1}^{T} + 1$ into training and test periods for the base algorithms, where $1 \leq t \leq T_1$ is the training period and $T_1 + 1 \leq t \leq T$ is the test period.
  - train all of the base algorithms over the training period.
  - generate forecasts for the test period for all of the base models, i.e., form $\{\hat{y}_t^{(i)}\}_{t=T_1+1}^{T_2}$.
  - construct the ensemble side information vector $s_t^{E}$ by following the procedure in (4).

  **train the ensemble learner:**
  
  - train the ensemble learner using $s_t^{E}$ to minimize the loss
  
  \[
  \arg \min_{w_{s_t^{E}}} \sum_{t=T_1+1}^{T} \ell(y_t, \hat{y}_t^{E})
  \]  

  \[
  \hat{y}_t^{E} = \sum_{i=1}^{M} w_{s_t^{E}}^{i} \hat{y}_t^{(i)}, \quad \ell \text{ is a given differentiable loss function such as the squared error loss, } w_{s_t^{E}}^{i} = [w_{s_t^{E}}^{(1)}, w_{s_t^{E}}^{(2)}, \ldots, w_{s_t^{E}}^{(M)}]^T \text{ is the weight vector for the linear combination of the base learners, and } w_{s_t^{E}}^{i} \text{ is the combination weight assigned to the } i^{th} \text{ base learner at time } t.
  \]

  - train all of the base algorithms over the entire training data $1 \leq t \leq T$.

- **Online Phase:** Predicting the Test Data
  
  **inputs:**
  
  - $\{y_t\}_{T+1}^{T_2}$: the test data that consists of sequential data samples of the time series sequence $y_t$, from $T + 1$ to $T_2$.
  - the trained ensemble learner from the offline phase.
  - $M$ base models, trained on the training data.
Fig. 2. Flowchart of the proposed ensemble training scheme.

- \( \{s_{i}^{(t)}\}_{t=T+1}^{T_2} \): time series features for the \( i \)th base predictor at time \( T \).
- \( \{s_{E}^{(t)}\}_{t=T+1}^{T_2} \): time series features for the ensemble learner.

**output:**
- combination weights \( w_{s_{E}^{(t)}}^{(i)} \) for each base learner \( 1 \leq i \leq M \), for \( T + 1 \leq t \leq T_2 \).
- ensemble predictions \( \hat{y}_{E}^{(t)} \) for \( T + 1 \leq t \leq T_2 \).

**generate combination weights:**
- generate forecasts for the test data from all of the base models, i.e., generate \( \{\hat{y}_{i}^{(t)}\}_{t=T+1}^{T_2}, 1 \leq i \leq M \).
- construct the ensemble side information vector \( s_{E}^{(t)} \) for the test dataset by following the procedure in [4].
- feed the side information vector \( s_{E}^{(t)} \) to the ensemble learner to generate the weight vector \( w_{s_{E}^{(t)}}^{T} = [w_{s_{E}^{(1)}}, w_{s_{E}^{(2)}}, \ldots, w_{s_{E}^{(M)}}]^T \), for \( T + 1 \leq t \leq T_2 \).
- linearly combine the weights given to each base learner with corresponding predictions to predict \( \hat{y}_{E}^{(t)} = \sum_{i=1}^{M} w_{s_{E}^{(i)}}^{(t)} \hat{y}_{i}^{(t)} \), for \( T + 1 \leq t \leq T_2 \).

We next introduce two novel and generic ensemble algorithms which will achieve the optimal weights under different constraints given in [8], [9] and [10].

**C. LightGBM Ensemble**

Gradient boosting decision trees (GBDT) are sequentially trained decision tree ensembles. In each iteration, GBDT minimizes the residual error made by the previous trees in the sequence. LightGBM is a state-of-the-art GBDT model that deviates in finding split points to minimize a certain loss function for the whole data during training, and is efficient in terms of memory consumption and training speed compared to its counterparts [19].

We employ LightGBM to search for the optimal ensemble combination weights, as described in Algorithm 1. LightGBM requires the gradient and hessian of the objective function (2) in order to minimize the loss with respect to the combination weight vector \( w_{s_{E}^{(t)}} \). We study the linear combination of the base predictions under unconstrained, convex constrained and affine constrained conditions, which were studied in Section III. For each base learner \( i \), LightGBM produces an output value \( p_{s_{E}^{(i)}}^{(t)} \), where we then apply the necessary transformation to achieve \( w_{s_{E}^{(i)}}^{(t)} \) under the given weight constraint. Therefore, for all three cases, we study and provide the gradient and hessian of (2) with respect to \( p_{s_{E}^{(i)}}^{(t)} \) for each base learner \( i \). For all constraints, we set the loss function \( \ell \) as the squared error loss, although any differentiable loss function can be selected, and assume we have the base predictions for \( 1 \leq t \leq T \), as explained in Fig. 2 and Algorithm 1. Therefore, at any given time \( t \) from the training dataset, the loss to minimized by the LightGBM ensemble is

\[
L_{t} = \ell(y_{t}, \hat{y}_{E}^{(t)}) = (y_{t} - \hat{y}_{E}^{(t)})^{2} = (y_{t} - \sum_{i=1}^{M} w_{s_{E}^{(i)}}^{(t)} \hat{y}_{i}^{(t)})^{2},
\]

where \( y_{t} \) is the observed sequence, and \( \hat{y}_{E}^{(t)} \) is the prediction of the ensemble algorithm, at time \( t \). The transformation from
apply the softmax transformation given as $w_{sE}^{(i)} = \tau(p_{sE}^{(i)})$, where $\tau$ indicates the transformation operation and is specific to the given weight constraint. Hence, the gradient and hessian to be calculated at any time $t$, for the base learner $i$, is given as
\[
G_i = \frac{\partial L_t}{\partial p_{sE}^{(i)}},
\]
\[
H_i = \frac{\partial G_i}{\partial p_{sE}^{(i)}}.
\]

Next, we explain the necessary transformations, and provide the gradient and hessian calculations for all three constraints.

1) Unconstrained Case: For the unconstrained case, there is no restriction on the value of $w_{sE}^{(i)}$. Therefore, we can simply write the transformation relation as $w_{sE}^{(i)} = p_{sE}^{(i)}$. The gradient and hessian equations in (18) for the base learner $i$, at time $t$ become
\[
G_i = 2(y_t - y^E_t)(-\frac{\partial \hat{y}^E_t}{\partial p_{sE}^{(i)}})
\]
\[
= 2(y_t - y^E_t)(-\sum_{m=1}^{M} \frac{\partial w_{sE}^{(m)}}{\partial p_{sE}^{(i)}} \hat{y}_t^{(m)})
\]
\[
= 2\hat{y}_t^{(i)}(y^E_t - y_t)
\]
\[
H_i = 2\hat{y}_t^{(i)}(\frac{\partial \hat{y}^E_t}{\partial p_{sE}^{(i)}})
\]
\[
= 2(\hat{y}_t^{(i)})^2.
\]

2) Affine Constrained Case: For the affine constrained case, the restriction is the summation of the combination weights should be equal to one, i.e., $\sum_{i=1}^{M} w_{sE}^{(i)} = 1$. In order to satisfy this relation, we apply the given normalization operation $w_{sE}^{(i)} = \frac{p_{sE}^{(i)}}{\sum_{m=1}^{M} p_{sE}^{(m)}}$. Hence, the gradient and hessian equations in (18) for the base learner $i$, at time $t$ become
\[
G_i = 2(y_t - y^E_t)(-\frac{\partial \hat{y}^E_t}{\partial p_{sE}^{(i)}})
\]
\[
= 2(y_t - y^E_t)(-\sum_{m=1}^{M} \frac{\partial w_{sE}^{(m)}}{\partial p_{sE}^{(i)}} \hat{y}_t^{(m)})
\]
\[
= 2\hat{y}_t^{(i)}(y^E_t - y_t)
\]
\[
H_i = 2\hat{y}_t^{(i)}(\frac{\partial \hat{y}^E_t}{\partial p_{sE}^{(i)}})
\]
\[
= 2\hat{y}_t^{(i)}\sum_{m=1}^{M} \frac{\partial w_{sE}^{(m)}}{\partial p_{sE}^{(i)}} \hat{y}_t^{(m)}
\]
\[
= 2\hat{y}_t^{(i)}\sum_{m=1}^{M} \frac{\partial w_{sE}^{(m)}}{\partial p_{sE}^{(i)}} \hat{y}_t^{(m)}
\]
\[
= 2(\hat{y}_t^{(i)})^2.
\]

3) Convex Constrained Case: For the convex constrained case, the restriction is the summation of the combination weights should be equal to one, such that $\sum_{i=1}^{M} w_{sE}^{(i)} = 1$. In addition, $0 \leq w_{sE}^{(i)} \leq 1$ should also be satisfied. Therefore, we apply the softmax transformation given as $w_{sE}^{(i)} = \frac{e^{p_{sE}^{(i)}}}{\sum_{m=1}^{M} e^{p_{sE}^{(m)}}}$. Therefore, the gradient and hessian equations in (18) for the base learner $i$, at time $t$ become
\[
G_i = 2(y_t - y^E_t)(-\frac{\partial \hat{y}^E_t}{\partial p_{sE}^{(i)}})
\]
\[
= 2(y_t - y^E_t)(-\sum_{m=1}^{M} \frac{\partial w_{sE}^{(m)}}{\partial p_{sE}^{(i)}} \hat{y}_t^{(m)})
\]
\[
= 2(y_t - y^E_t)w_{sE}^{(i)}(\hat{y}_t^{(i)} - \hat{z}_t^{(i)})
\]
\[
H_i = G_i(1 - 2w_{sE}^{(i)}) + 2(w_{sE}^{(i)})^2(\hat{y}_t^{(i)} - \hat{z}_t^{(i)})^2.
\]

D. MLP Ensemble

Artificial Neural Networks(ANN) are models that are based on the processing structure of the brain cells, and are used to model complex patterns and problems [29]. As our ensemble algorithm, we employ a certain class of ANN called Multilayer Perceptron (MLP), which is an ANN with more than two layers. MLPs are heavily used in machine learning due to their ability of learning complex and non-linear relations with high training speed compared to their counterparts [30].

As a specific example, we provide the equations for a two layered network architecture, which consists of an input layer, a hidden layer and an output layer. For more layers, naturally, our derivations can be straightforwardly extended. At the end of the output layer, we apply a transformation operation which maps the weights of the ensemble algorithm according to the given weight constraint. We study the linear combination of the base predictions under affine unconstrained, affine constrained and convex constrained conditions, which were studied in Section III. The feed-forward equations of our model is given as
\[
v = U^{(1)}x
\]
\[
z = f(v)
\]
\[
p = U^{(2)}z
\]
\[
w = \tau(p),
\]
where $x \in \mathbb{R}^K$ is the input vector, $U^{(1)} \in \mathbb{R}^{L \times K}$ are the layer weights connecting the input layer to the hidden layer, $z \in \mathbb{R}^L$ is the input of the hidden layer, which is the rectified linear activation function (ReLU) $f$ applied to $v \in \mathbb{R}^L, U^{(2)} \in \mathbb{R}^{K \times M}$ are the layer weights connecting the hidden layer to the output layer, $p \in \mathbb{R}^M$ is the output vector of the model, and $w \in \mathbb{R}^M$ is the weight vector for the linear combination of the base models, obtained by applying the transformation operation $\tau$ to $p, K, L, M$ are the lengths of the input, hidden and output layers, respectively.

Our ensemble algorithm learns to produce linear combination weights, by minimizing the loss function (2). For all constraints, we set $\ell$ as the squared error loss, although any differentiable loss function can be selected, and assume we have the base predictions for $1 \leq t \leq T$, as explained in Fig. 2 and Algorithm 1. Therefore, at any given time $t$ from the
training data, the loss to be minimized by the MLP ensemble is given as

\[ L_t = \ell(y_t, \hat{y}_t^E) = (y_t - \hat{y}_t^E)^2 = (y_t - \sum_{i=1}^M w_{s_t}^{(i)} \hat{y}_t^{(i)})^2, \]

where \( y_t \) is the observed sequence, and \( \hat{y}_t^E \) is the prediction of the ensemble algorithm, at time \( t \). Here, we employ the notations \( x_i \equiv x_i^{(E)} \), \( w_i \equiv w_i^{(E)} \) and \( p_i \equiv p_i^{(E)} \), for simplicity. The loss is minimized by iteratively updating the layer weights \( U^{(1)} \) and \( U^{(2)} \) with backpropagation. The backpropagation equations are given as

\[
\frac{\partial L_t}{\partial U^{(1)}_{l,k}} = \sum_{i=1}^M \sum_{j=1}^M \frac{\partial L_t}{\partial w_i} \frac{\partial w_i}{\partial \hat{y}_t^{(j)}} \frac{\partial \hat{y}_t^{(j)}}{\partial U^{(1)}_{l,k}},
\]

\[
\frac{\partial L_t}{\partial U^{(2)}_{m,l}} = \sum_{i=1}^M \frac{\partial L_t}{\partial v_i} \frac{\partial v_i}{\partial \hat{y}_t^{(m)}} \frac{\partial \hat{y}_t^{(m)}}{\partial U^{(2)}_{m,l}},
\]

where \( 1 \leq l \leq L, 1 \leq k \leq K, 1 \leq m \leq M \). We then update \( U^{(1)} \) and \( U^{(2)} \) using stochastic gradient descent (SGD). The update equations are given as

\[ U^{(1)}_{l,k} = U^{(1)}_{l,k} - \alpha \frac{\partial L_t}{\partial U^{(1)}_{l,k}}, \quad U^{(2)}_{m,l} = U^{(2)}_{m,l} - \alpha \frac{\partial L_t}{\partial U^{(2)}_{m,l}}, \]

where \( \alpha \) is the learning rate hyperparameter of SGD. Next, we explain the necessary transformations, and provide the closed-form backpropagation equations under the given constraints.

1) Unconstrained Case: For the unconstrained case, there is no restriction on the value of \( w_{s_t}^{(i)} \). Therefore, we can simply write the transformation relation as \( w_{s_t}^{(i)} = p_{s_t}^{(i)} \). Hence, the backpropagation equations become

\[
\frac{\partial L_t}{\partial U^{(1)}_{l,k}} = \sum_{i=1}^M 2(\hat{y}_t^{(i)} - y_t^{(i)}) U^{(2)}_{m,l} f'(v_i) x_k,
\]

\[
\frac{\partial L_t}{\partial U^{(2)}_{m,l}} = 2(\hat{y}_t^{(m)} - y_t^{(m)}) z_l,
\]

where \( f'(v_i) \) is the derivative of ReLU, which is the piece-wise function

\[ f'(v_i) = \begin{cases} 0, & \text{if } v_i \leq 0, \\ v_i, & \text{if } v_i > 0. \end{cases} \]

2) Affine Constrained Case: For the affine constrained case, the restriction is the summation of the combination weights should be equal to one, such that \( \sum_{i=1}^M w_{s_t}^{(i)} = 1 \). In order to satisfy this relation, we apply the given normalization operation \( w_{s_t}^{(i)} = \frac{p_{s_t}^{(i)}}{\sum_{m=1}^M p_{s_t}^{(m)}} \). Hence, the backpropagation equations become

\[
\frac{\partial L_t}{\partial U^{(1)}_{l,k}} = \sum_{i=1}^M 2c(\hat{y}_t^{(i)} - y_t^{(i)}) U^{(2)}_{m,l} f'(v_i) x_k,
\]

\[
\frac{\partial L_t}{\partial U^{(2)}_{m,l}} = 2c(\hat{y}_t^{(m)} - y_t^{(m)}) z_l,
\]

where \( c \) is the constant term given as \( c = \frac{1}{\sum_{m=1}^M p_{s_t}^{(m)}} \).

3) Convex Constrained Case: In the convex constrained case, the restriction is the summation of the combination weights should be equal to one, such that \( \sum_{i=1}^M w_{s_t}^{(i)} = 1 \). In addition, \( 0 \leq w_{s_t}^{(i)} \leq 1 \) should also be satisfied. Therefore, we apply the given softmax transformation \( w_{s_t}^{(i)} = \frac{e^{p_{s_t}^{(i)}}}{\sum_{m=1}^M e^{p_{s_t}^{(m)}}} \).

Hence, the backpropagation equations become

\[
\frac{\partial L_t}{\partial U^{(1)}_{l,k}} = \sum_{i=1}^M 2(\hat{y}_t^{(i)} - y_t^{(i)}) U^{(2)}_{m,l} f'(v_i) x_k,
\]

\[
\frac{\partial L_t}{\partial U^{(2)}_{m,l}} = 2(\hat{y}_t^{(m)} - y_t^{(m)}) z_l.
\]

We now have the update equations to train both the MLP ensemble and LightGBM ensemble algorithms according to Fig. 2 and Algorithm 1.

Remark. Note that, although we provide two different models to use as our ensemble learners, any machine learning model such as random forests \([10]\), linear regression models \([37]\) etc. can be employed.

The following section illustrates the simulations of our ensemble models under different constraints, and also shows their superiority over the base models that we employ.

IV. SIMULATIONS

In this section, we first verify our model by using synthetically generated data to show the learning procedure of our algorithms with respect to the conventional classical ensemble methods widely used in the literature \([22]\). Then, we showcase the performance of our ensemble models under different constraints using real-life data, which are the well-known prediction competition M5 \([5]\), and the daily energy consumption production in Turkey. As the conventional ensemble models, we employ two different models. First, we use a conventional linear ensemble model where linear regression is used to learn the relation between the base model predictions and the observed sequence \( y_t \). Next, we use an MLP ensemble with two layers to learn the relation between the base model predictions and the observed sequence \( y_t \). Thus, both of the conventional models do not employ any side information related to the sequence, and make use of only the base model predictions while learning the combination weights.

For both our MLP ensemble introduced in Section III-D and LightGBM ensemble introduced in Section III-C, we use two base models, and search for the optimal linear combination weights \( w_{s_t}^{(1)} \) and \( w_{s_t}^{(2)} \), given the side information vector \( s_t \), under unconstrained, affine constrained and convex constrained conditions. However, our results can be straightforwardly extended to cases with more than two base models. During all our simulations, the data is divided into two partitions as the training and test datasets according to Fig. 2 and Algorithm 1. We only consider one-step-ahead forecasting and compare models in terms of the total loss

\[
\sum_{t=T+1} T_2 L_t = \sum_{t=T+1} T_2 \ell(y_t, \hat{y}_t),
\]
where \( y_t \) is the observed sequence at time \( t \), \( \ell \) is selected as the squared error loss, \( T + 1 \leq t \leq T_2 \) is the duration of the test dataset, \( 1 \leq t \leq T \) is the duration of the training dataset, and \( \hat{y}_t \) is the prediction of the corresponding (base or ensemble) model. We also illustrate the cumulative normalized total error for all of the models used in the experiments, given as

\[
\sum_{k=T+1}^{t} \left( \frac{(y_k - \hat{y}_k)^2}{t}, t = T + 1, \ldots, T_2, \right.
\]

where \( y_k \) is the current sample of the signal to be predicted and \( \hat{y}_k \) is the prediction of the corresponding model.

A. Synthetic Data

The synthetic data consists of two manually generated and independent data components. First data is generated using the autoregressive integrated moving average (ARIMA) model given by

\[
y_t^{(1)} = 0.2y_{t-1}^{(1)} - 0.1y_{t-1}^{(1)} + 0.3e_t^{(1)} - 0.1e_{t-1}^{(1)} + v_t^{(1)},
\]

where \( v_t^{(1)} \) is a sample function from a stationary white Gaussian process with unit variance and \( e_t^{(1)} \) is the error term given by \( (y_t^{(1)} - 0.2y_{t-1}^{(1)} - 0.1y_{t-2}^{(1)}) \).

The second data is generated by a highly complex and hard to model piecewise-linear structure, and is given as

\[
y_t^{(2)} = 30 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} > 50, y_{t-1}^{(2)} > 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} > 50
\]

\[
y_t^{(2)} = 35 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} > 50, y_{t-1}^{(2)} > 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} < 50
\]

\[
y_t^{(2)} = 40 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} > 50, y_{t-1}^{(2)} < 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} > 50
\]

\[
y_t^{(2)} = 45 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} > 50, y_{t-1}^{(2)} < 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} < 50
\]

\[
y_t^{(2)} = 56 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} < 50, y_{t-1}^{(2)} > 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} > 50
\]

\[
y_t^{(2)} = 61 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} < 50, y_{t-1}^{(2)} < 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} < 50
\]

\[
y_t^{(2)} = 66 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} < 50, y_{t-1}^{(2)} < 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} > 50
\]

\[
y_t^{(2)} = 71 + v_t^{(2)}, \text{ if } y_{t-1}^{(2)} < 50, y_{t-1}^{(2)} < 50, \frac{1}{7} \sum_{k=1}^{7} y_{t-k}^{(2)} < 50
\]

where \( v_t^{(2)} \) is a sample function from a stationary white Gaussian process with unit variance.

For our synthetic data experiments, our data samples are of length 730, where the last 100 samples are taken as the test data, and the remaining are taken as the training data. We evaluate the model accuracies on the test dataset. We train our ensemble models according to Fig. 2 and Algorithm 1 and for the base models, we directly use \( y_t^{(1)} \) and \( y_t^{(2)} \). Therefore, we expect the ensemble models to learn the combination weights \( w_s^{(1)} \) and \( w_s^{(2)} \) used in forming all three ensemble datasets.

As the side information vector \( \mathbf{s}_t^E \), we provide the modulo information that splits the data weights into different regions. Note that the importance of the side information vector to the data as well as the data complexity varies among the three datasets, as there are different number of distinct weight combinations for each dataset. For each of the three datasets, we fit our MLP ensemble model introduced in Section III-D under all three constraints (unconstrained, affine constrained and convex constrained). Note that, as \( w_{s_k}^{(1)} + w_{s_k}^{(2)} = 1 \) is satisfied for all weight combinations, our ensemble algorithm should be theoretically able to capture all of the combination weights under any constraint.

Table I shows the results in terms of the final cumulative error for all three ensemble datasets, for all three constraints. We also illustrate the cumulative normalized total error for the predictions of \( y_t^{(b)} \) and \( y_t^{(c)} \) in Figs. 3 and 4. When the data does not contain many distinct weight combinations related to the side information vector \( (y_t^{(a)}) \), all three ensembles are able to perfectly capture the weight combinations. However, as
we increase the number of distinct weight combinations, the ensemble model under unconstrained and affine constrained conditions tend to make more errors compared to the convex constrained model. This is due to the fact that, these models have more parameters to learn compared to the convex constrained model, as the search space for the optimal combination weights is larger and therefore more complex. Therefore, the models may not converge to the optimal weight conditions, as in our example. Hence, even though the unconstrained ensemble model has the least expected error among all three constraints as explained in Section III, convex and affine models can be preferable in complex problems.

In addition, the MLP ensemble model under all three constraints outperform both the conventional linear ensemble and conventional MLP ensemble. In Table I, it is illustrated that both of the conventional methods are unable to capture the regional change in the structure of the observed sequence $y_t$, hence, producing large errors opposite to our MLP ensemble, which directly employs a side information vector containing the regional switching information. Thus, our results showcase the need of employing a data specific side information vector while learning the associated combination weights.

### B. Total Residential Natural Gas Demand in Turkey

The data used in this section consists of the total residential natural gas demand in Turkey for 1000 days during the years 2018-2020, and each day is considered as a single sample. We train both our LightGBM ensemble introduced in Section III-C and MLP ensemble introduced in Section III-D under all weight constraints, according to Fig. 2 and Algorithm 1. For the base algorithms, we use the Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors (SARIMAX), which is a linear model commonly used in time series forecasting [33], and the LightGBM, which is a gradient boosting framework that uses tree based learning algorithms [19]. We take the last 300 samples of the data as the test dataset, and the remaining as the training dataset. We evaluate the model accuracies on the test dataset.

Table II illustrates our results in terms of the final cumulative error for both the base algorithms and the ensemble algorithms under all three constraints. We also illustrate the cumulative normalized total error for all models in Fig. 5. Note that, the results for the LightGBM ensemble model under unconstrained weights condition is not shown, as the model could not converge to produce reasonable weights. In addition, both our results in Table II and Fig. 5 illustrate that both our LightGBM ensemble and MLP ensemble algorithms outperform the two base algorithms under both affine and convex weight constraints. For both ensemble models, when the weights produced by the models are unconstrained, the models could not improve the predictions produced by the base models. Therefore, as also illustrated in synthetic data experiments, even though the expected error for the unconstrained ensemble models are less than the ensemble models with constrained weights, convex and affine models can be preferable as they have less parameters to learn compared to the unconstrained case. In addition, our ensemble models under affine and convex constraints outperform the conventional ensemble methods, as illustrated in II and Fig. 5.

### C. M5 Forecasting Dataset

The M5 Forecasting dataset [5] involves the unit sales of the products sold by the retail company Walmart in USA. The products are classified in 3 different categories in 7 different departments, and are sold in 10 different stores in 3 different states. For our experiments, we examine the total number of unit sales in store CA_3 department HOUSEHOLD_1,

| Model               | $y_t^{(a)}$ | $y_t^{(b)}$ | $y_t^{(c)}$ |
|---------------------|-------------|-------------|-------------|
| MLP Ensemble        |             |             |             |
| Unconstrained       | 0.0         | 0.10248     | 6.64695     |
| Affine Constrained  | 0.0         | 0.06670     | 2.72202     |
| Convex Constrained  | 0.00603     | 0.21027     |
| Conventional Linear | 136.69053   | 256.78693   | 367.59445   |
| MLP Ensemble        | 124.90278   | 262.02210   | 140.68292   |

**TABLE I**

Final cumulative error on the test set for all ensemble data, under all constraints for the MLP ensemble.
Fig. 5. Comparison of the cumulative error for the prediction of the total residential natural gas demand in Turkey. The performance of our base and ensemble models under different constraints are shown. Note that the error plot for the unconstrained lightgbm ensemble is not provided here, as the errors produced by the model are high compared to other models.

| Model Name                  | Final Cumulative Error (1e10) |
|-----------------------------|-------------------------------|
| SARIMAX Base                | 50.16                         |
| LightGBM Base               | 28.67                         |
| LightGBM Ensemble           |                               |
| Affine Constrained          | 27.51                         |
| Convex Constrained          | 24.54                         |
| MLP Ensemble Unconstrained  | 35.25                         |
| MLP Ensemble Affine Constrained | 25.92   |
| MLP Ensemble Convex Constrained | 28.47   |
| Conventional MLP Ensemble   | 29.43                         |
| Conventional Linear Ensemble| 35.86                         |

TABLE II
The final cumulative error for both the base and ensemble models used in predicting total residential natural gas demand in Turkey, under all constraints.

therefore reducing the possibility of error in the data. The data length is 1941 days, where each day is considered a single sample. We consider the first 1841 days as the training dataset, and the last 100 days as the test dataset.

We train both our LightGBM ensemble introduced in Section III-C and MLP ensemble introduced in Section III-D under all weight constraints, according to Fig. 2 and Algorithm 1. For the base algorithms, we use the SARIMAX and LightGBM models, as in the case in Section IV-B. We evaluate both the base and ensemble model accuracies on the test dataset.

Table III illustrates our results in terms of the final cumulative error for both the base algorithms and the ensemble algorithms under all three constraints. We also give the cumulative normalized total error for all models in Fig. 6. We do not show the results of the ensemble algorithms under unconstrained case, as the models could not converge to produce reasonable weights. Our results in Table III and Fig. 6 illustrate that both our MLP ensemble and LightGBM ensemble models outperform our base models and also the conventional ensemble models, under affine and convex constrained weights.

| Model Name                  | Final Cumulative Error |
|-----------------------------|------------------------|
| SARIMAX Base                | 10377.02               |
| LightGBM Base               | 7811.49                |
| LightGBM Ensemble           |                        |
| Affine Constrained          | 7489.49                |
| Convex Constrained          | 7509.36                |
| MLP Ensemble Affine Constrained | 7582.25   |
| MLP Ensemble Convex Constrained | 7688.94   |
| Conventional MLP Ensemble   | 8485.79                |
| Conventional Linear Ensemble| 9130.51                |

TABLE III
The final cumulative error for both the base and ensemble models used in predicting the total unit sales in store CA_3 DEPARTMENT HOUSEHOLD_1 of Walmart, USA, under affine and convex constraints.

V. Conclusion
We studied the problem of predicting sequential time series data where we combine the predictions of multiple machine learning models using a novel ensembling approach. For the first time in the literature, we tackle the problem of finding the optimal combination weight vectors under unconstrained, affine constrained and convex constrained conditions, while considering a data specific side information vector. We analyzed the associated costs for learning all three constraints given the side information vector, and then introduced two novel and generic ensembling algorithms to find the optimal combination weight vectors under given constraints. In addition, we have presented a novel training scheme for ensemble models, where we have diminished any possible issues related to the training of the base models, such as co-linearity, high correlation and overfitting by the base algorithms. With various experiments containing synthetic and well-known real-life sequential data, we have illustrated superiority of our ensemble models over both the base models used in the experiments and the conventional ensembling methods in the literature.

REFERENCES
[1] A. Sayeed, Y. Choi, J. Jung, Y. Lops, E. Eslami, and A. K. Salman, “A deep convolutional neural network model for improving wrf simu-
Fig. 6. Comparison of the cumulative error for the prediction of the total unit sales in store CA_3 department HOUSEHOLD_1 of Walmart, USA. The performance of our base and ensemble models under different constraints are shown. Note that the error plots for the unconstrained LightGBM ensemble and unconstrained MLP ensemble are not provided here, as the errors produced by the model are high compared to other models.