Theoretical Guarantees of Learning Ensembling Strategies with Applications to Time Series Forecasting

Hilaf Hasson [1] **Danielle C. Maddix** [1] **Yuyang Wang** [1] **Gaurav Gupta** [1] **Youngsuk Park** [1]

Abstract

Ensembling is among the most popular tools in machine learning (ML) due to its effectiveness in minimizing variance and thus improving generalization. Most ensembling methods for black-box base learners fall under the umbrella of "stacked generalization," namely training an ML algorithm that takes the inferences from the base learners as input. While stacking has been widely applied in practice, its theoretical properties are poorly understood. In this paper, we prove a novel result, showing that choosing the best stacked generalization from a (finite or finite-dimensional) family of stacked generalizations based on cross-validated performance does not perform "much worse" than the oracle best. Our result strengthens and significantly extends the results in Van der Laan et al. (2007). Inspired by the theoretical analysis, we further propose a particular family of stacked generalizations in the context of probabilistic forecasting, each one with a different sensitivity for how much the ensemble weights are allowed to vary across items, timestamps in the forecast horizon, and quantiles. Experimental results demonstrate the performance gain of the proposed method.

1. Introduction

Ensemble methods have been a staple in machine learning with ensemble-based methods such as Random Forests (Breiman, 2001) and XGBoost (Chen & Guestrin, 2016) being among the most popular choices for tabular data by practitioners (KS2, 2020; 2021). Both classical (mean, median, weights based on cross-validated performance) and relatively modern ensembling strategies, e.g., Erickson et al. (2020) (which in two popular Kaggle competitions has beat 99% of the participating data scientists after training on the raw data) are examples of "stacked generalizations", or "stacking" (Wolpert, 1992); see (Ting & Witten, 1997b). Stacking is an umbrella term referring to the process of training one model on top of the predictions of the base learners. In the simple mean and median cases, the stacking model is constant rather than learned. Despite being heavily used in practice, the theoretical properties of stacked generalizations are underexplored, and stacking has notoriously been referred to as "black art" (Wolpert, 1992; Ting & Witten, 1997a).

One early attempt to quantifiably understand stacked generalization is due to Van der Laan et al. (2007). Built on top of Van der Vaart et al. (2006), the authors prove a theoretical guarantee (Theorem 2 of Van der Laan et al. (2007)) that choosing the best stacked generalization out of a *finite* (rather than *finite-dimensional*) family of *constant* (as opposed to *learned*) stacked generalizations based on cross-validation performance does not do much worse than the oracle best. They then apply their result to the case of a single learned stacked generalization by discretizing the space of the functions that the stacked generalization may become after the training, resulting in a guarantee that degrades with the cardinality of the chosen discretization.

In this paper, we make a major step towards a better theoretical understanding of the stacking mechanism. More precisely, we show new results that extend Van der Laan et al. (2007) in two ways: 1. We remove the assumption that the stacked generalizations are constant, and permit them to be learned, without changing the conclusion of the result. This allows us to get much better guarantees even for the case of a single stacked generalization, as it removes the need for a discretization. (See the discussion following Theorem 4.1.) 2. We extend the result from the case of finite families to the case of finite-dimensional families.

We further push the boundary beyond the tabular case to study the concrete use case of time series forecasting. Ensembling of time series models is a notoriously challenging task, coined as the "Forecast Combination Puzzle" by (Stock & Watson, 2004), with many follow-up works in the literature; see Section 2. We concern ourselves with datasets that involve multiple time series, forecasting a fixed number of timestamps into the future, and predicting multiple quantiles

[1] AWS AI Labs, Santa Clara, CA, USA. Correspondence to: Hilaf Hasson <hashilaf@amazon.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

for each time series and timestamp. Motivated by the theoretical insights, we propose a particular finite-dimensional family of stacked generalizations, in which how much the ensemble weights are allowed to vary across time series, timestamps, and quantiles is each controlled by a dimension of the family.

In summary, our contributions are two-fold:

1. In the tabular case we show (Theorem 4.1) that the process of letting cross validation determine a stacked generalization from an arbitrary finite or finite-dimensional family of stacked generalizations cannot perform much worse than the oracle best, and under an additional assumption, from each of the base learners. This extends a result from Van der Laan et al. (2007) by allowing the stacked generalizations to be learned rather than constant, and by extending from the case of a finite to finite dimensional family. In the case of a single stacked generalization, our result gives tighter bounds than Van der Laan et al. (2007), as the latter requires a discretization of the family of potential models of the stacked generalization once trained.
2. We propose a family of stacked generalizations in the time series use case, as well as a setup for choosing the best performing one, in a manner that learns the ensemble elasticity (how much the weights are allowed to vary) across time series/timestamps/quantiles. Despite Theorem 4.1 not applying directly to the non-tabular case, we show experiments that demonstrate that our method inspired by this theory is effective, supported by strong evidence that it is able to adjust to changes in performance of the base learners across time series/timestamps/quantiles.

We organize the rest of the paper as follows. Related work is discussed in Section 2, followed by preliminaries and notation (Section 3) to set the stage. The subsequent sections (4 and 5) cover the main results with its proof. In Section 6, we present the case study of time series forecasting, and use the next two sections to provide empirical evidences for the effectiveness of the proposed algorithm before concluding the paper.

2. Related Work

In this section, we give an overview of theoretical and applied research on stacked generalizations, as well as their use in time series forecasting.

Stacked Generalizations Stacked generalization has become an exceedingly popular choice for ensembling in practice, and particularly impressive in its results, as illustrated in Erickson et al. (2020). In a recent work (Kim et al., 2021), it has also been used for ensembling probabilistic predictions in the tabular setup. We refer to Zhou (2012) for a general discussion on the benefits of ensembling, and to Cruz et al. (2018) for a thorough benchmarking of ensembling methods.

We remark that while our paper concerns black box base learners, there has also been an active research front on creating top performing algorithms by choosing the base learners as well as the ensembling method, e.g., Breiman (2001); Chen & Guestrin (2016); Fort et al. (2019); Lakshminarayanan et al. (2017), etc.

Theoretical Results for Stacked Generalizations As mentioned in the introduction, in Van der Laan et al. (2007) the authors provide theoretical guarantees for stacked generalizations. They first discretize the set of possible functions that the stacked generalization may become once trained, and in this way reduce the problem to giving guarantees (Theorem 2 of Van der Laan et al. (2007)) that choosing the best performing (in cross validation) stacked generalization from a finite set of stacked generalizations, each of which is a constant function (e.g., simple mean, or some other constant weighted combination), does not do much worse than the oracle best.

Time Series Ensembling In the time series use case, it has been empirically observed that often simple averaging of the forecasts of the base learners is superior to more sophisticated ensemble methods; a problem dubbed the "forecast combination puzzle" in Stock & Watson (2004). This dates back to Bates & Granger (1969), where the authors observed that attempting to learn covariance terms between the errors of the individual learners introduces too much variance. Similar concerns over the amount of variance introduced when learning weights for time series ensembling are voiced in Smith & Wallis (2009) and Claeskens et al. (2016), where both works explored squared loss. Elliott (2011) focuses on understanding the potential gain from learning the optimal weights compared to simple averaging, and provides the following two results: first, that if the maximal eigenvalue of the covariance matrix of errors of the base learners is bounded, then as the number of base learners goes to infinity, the performance of using the optimal weights converges to that of using simple average; second, he gives an upper bound for the potential gain for the case of 3 and 4 base learners. We remark that the papers referenced here regarding the forecast combination puzzle are restricted to the situation of learning weights that are global (unchanging across time or any other dimension), and based on an analysis of mean squared loss (so as to simplify the theory), and they do not apply more broadly.

Stacked generalizations have been applied for time series forecast ensembling as far back as in Donaldson & Kamstra

(1996); Moon et al. (2020); Massaoudi et al. (2021). In Gastinger et al. (2021) we see the success of these single stacked generalizations in a large empirical study. In all of these cases a single stacked generalization was used, and not a family of stacked generalizations.

3. Preliminaries and Notation

In both the main theorem and in its proof we use language and notation commonly used in oracle inequalities. In order to be self contained we will now introduce the relevant notation and terminology.

Let $\{\theta_\alpha\}_{\alpha\in\mathcal{J}}$ be a finite set of algorithms for tabular data. Assume that feature space is d dimensional, the target values are r dimensional, and the output of predictions is r' dimensional. (An example when $r \neq r'$ would be: $r = 1$, and the predictions are for r' many conditional quantiles.) We write $\theta_\alpha(D_0) : \mathbb{R}^d \to \mathbb{R}^{r'}$ for the model that results from letting θ_α train on a training dataset D_0. (For the sake of simplicity, assume that $\theta_\alpha(D_0)$ is completely determined by D_0, though this assumption can easily be removed.) Let $L : \mathbb{R}^{r'} \times \mathbb{R}^r \to \mathbb{R}$ be a fixed loss function, and let $L_\alpha(D_0) : \mathbb{R}^d \times \mathbb{R}^r \to \mathbb{R}$ be defined by $L_\alpha(D_0)((x,y)) \coloneqq L(\theta_\alpha(D_0)(x), y)$.

Let X be a random sample of $\mathbb{R}^d \times \mathbb{R}^r$ w.r.t. the same probability distribution P as the one being sampled by the data.[1] The $\tilde{\alpha}$ that minimizes $\mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X)dP)$ (where the expected value runs over D_0) is called the "oracle". We never have direct access to the oracle, but given a validation set D_1 we can choose the $\hat{\alpha}$ for which $\theta_{\hat{\alpha}}$ has the lowest validation loss. In order to be more precise, write $D_i = \{X^i_1, ..., X^i_{n_i}\}$ for $i = 0, 1$ (where the $\{X^i_l\}^{n_i}_{l=1}$ are IID random variables taking values in $\mathbb{R}^d \times \mathbb{R}^r$ for some fixed $n_i \in \mathbb{N}$), representing training ($i = 0$) and validation ($i = 1$) sets; and let $\mathbb{P}_i$ for $i = 0, 1$ be the empirical distribution $\frac{1}{n_i}\sum^{n_i}_{l=1}\delta_{X^i_l}$. Then $\hat{\alpha}$ is the index for which $\mathbb{E}(\int L_{\hat{k}}(D_0)(X)d\mathbb{P}_1)$ is minimized.

Oracle inequalities are about giving upper bounds of $\mathbb{E}(\int L_{\hat{\alpha}}(D_0)(X)dP)$ in terms of $\mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X)dP)$, quantifying how much worse $\hat{\alpha}$ is compared with the oracle. (While much of the relevant literature accommodates a scheme where the data is split multiple times to train and validation, we have chosen to focus on the case of a single split for the sake of simplified notation. The interested reader should be able to easily generalize.) In Section 5 we introduce a new oracle inequality that allows us to prove our main theorem (Theorem 4.1).

In the oracle inequalities we discuss we will use the following notion.

Definition 3.1. Let $\mathcal{X}$ be a sample space with probability measure P. Then a we say that a function $f : \mathcal{X} \to \mathbb{R}$ has Bernstein numbers (or Bernstein pair) $(M(f), v(f)) \in \mathbb{R}^2$ if:

$$M(f)^2 \int \left(e^{\frac{|f|}{M(f)}} - 1 - \frac{|f|}{M(f)}\right) dP \le \frac{1}{2}v(f). \qquad (1)$$

A set of functions $\mathcal{F}$ from $\mathcal{X}$ to $\mathbb{R}$ is said to have Bernstein numbers $(M(\mathcal{F}), v(\mathcal{F})) \in \mathbb{R}^2$ if they are Bernstein numbers for all $f \in \mathcal{F}$.

The existence of Bernstein numbers can be viewed as a weak moment condition. (It is a much weaker condition than the functions in $\mathcal{F}$ having a uniformly bounded range; we refer to (Wellner et al., 2013) for further intuition.)

Finally, for any subset $\mathcal{J}$ of Euclidean space and value $\varepsilon > 0$ the notation $N^{\text{int}}(\mathcal{J}, \varepsilon)$ denotes the (internal) covering number of the space $\mathcal{J}$ with respect to its ambient Euclidean space.

4. Main Theorem

Let D_0 and D_1 be datasets of, respectively, n_0 and n_1 IID random variables, each single variable taking values in $\mathbb{R}^d \times \mathbb{R}^r$ (i.e, with feature dimension d and target dimension r).

Let $\eta_1, ..., \eta_m$ be a finite set of tabular algorithms (henceforth "the base learners") with feature dimension d, prediction dimension r', and target dimension r. Let $\{A_\alpha\}_{\alpha\in\mathcal{J}}$ be a set of stacked generalizations (tabular algorithms with feature dimension m, prediction dimension r', and target dimension r). Let D_0 be further split as $D_0 = D_{00} \uplus D_{01}$. The base learners will train on D_{00}, whereas the stacked generalizations will train on the predictions of the base learners on D_{01}. (We remark that having D_{00} and D_{01} be disjoint is the method recommended in (Zhou, 2012); though in fact what follows will continue to hold if you set $D_0 = D_{00} = D_{01}$.)

Let

$$Z \coloneqq \{((\eta_1(D_{00})(x), ..., \eta_m(D_{00})(x)), y) | (x, y) \in D_{01}\}$$

be the predictions on D_{01} of the base learners that were trained on D_{00}. Let $L : \mathbb{R}^{r'} \times \mathbb{R}^r \to \mathbb{R}$ be some loss function, and let $L_\alpha(D_{00}, D_{01})((x,y)) \coloneqq L(A_\alpha(Z)(\eta_1(D_{00})(x), ..., \eta_m(D_{00})(x)), y)$, where $A_\alpha(Z)$ stands for A_α trained on Z.

Consider $\mathcal{F} \coloneqq \{(x,y) \mapsto L_\alpha(D_{00}, D_{01})((x,y)) | \alpha \in \mathcal{J}\}$, and let $(M(\mathcal{F}), v(\mathcal{F}))$ be its Bernstein numbers. Finally, let X be a random variable in $\mathbb{R}^d \times \mathbb{R}^r$ following the same distribution that D_0 and D_1 are sampling from. Then the following holds, using the notation in Section 3.

Theorem 4.1. *With the notation above, for every $\hat{\alpha} \in \mathcal{J}$ let*

$$W_{\hat{\alpha}} \coloneqq \Big\{\tilde{\alpha} \in \mathcal{J} \Big| \mathbb{E}(\int L_{\hat{\alpha}}(D_{00}, D_{01})(X)d\mathbb{P}_1) \le \mathbb{E}(\int L_{\tilde{\alpha}}(D_{00}, D_{01})(X)d\mathbb{P}_1)\Big\},$$

[1] This notation is common in literature, but note that it is confusing: X is both the feature vector and the target.

be the set of indices $\tilde{\alpha}$ for which $\theta_{\hat{\alpha}}$ outperforms $\theta_{\tilde{\alpha}}$ on validation. Further assume that $\mathcal{J}$ is a bounded subset of a finite dimensional Euclidean space, and that for every D_0 the function that takes $\alpha \in \mathcal{J}$ to $L_\alpha(D_{00}, D_{01}) \in \mathcal{F}$ is Lipschitz with constant ℓ w.r.t. the infinity norm in $\mathcal{F}$. Then we have that for every $\delta > 0$, $1 \leq p \leq 2$, and sequence $\varepsilon_{n_1} > 0$,

$$\begin{aligned}&\mathbb{E}(\int L_{\hat{\alpha}}(D_{00}, D_{01})(X)dP) \leq \\ &(1+2\delta) \inf_{\tilde{\alpha} \in W_{\hat{\alpha}}} \left(\mathbb{E}(\int L_{\tilde{\alpha}}(D_{00}, D_{01})(X)dP) \right) \\ &\quad + \sup_{f \in \mathcal{F}}(B_f) + 2((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1},\end{aligned} \tag{2}$$

where

$$B_f := \frac{16\left(\frac{M(\mathcal{F})}{n_1^{1-\frac{1}{p}}} + \left(\frac{v(\mathcal{F})}{(\delta \int f dP)^{2-p}}\right)^{\frac{1}{p}}\right)\log(1 + N^{int}(\mathcal{J}, \frac{\varepsilon_{n_1}}{\ell}))}{n_1^{\frac{1}{p}-\frac{1}{2}}}.$$

In addition, the following hold:

1. *If we choose $\varepsilon_{n_1} := n_1^{-\frac{1}{2}-\epsilon}$ for any $\epsilon > 0$ then $\sup_{f \in \mathcal{F}}(B_f)$ and $2((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1}$ are $o(n_1)$.*
2. *For the guarantee to hold not only against the best stacked generalizations in the family, but also against the original base learners, one simply adds the summand m to the inside of the* log *term.*
3. *If $\mathcal{J}$ is finite rather than finite dimensional, then the log term can be replaced with $\log(1 + |\mathcal{J}|)$, and we may remove the last summand $2((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1}$.*

In the case that $|\mathcal{J}|$ is finite, and that, in addition, the stacked generalizations are constant functions (i.e., do not depend on Z at all), then the theorem above specializes to Theorem 2 of Van der Laan et al. (2007). The two main differences are the following:

1. There the authors make additional assumptions (such as that $r = r' = 1$ and that the loss function is MSE); and they write in the language of multiple cross validations, which we have chosen not to use so as not to overload notation (though generalization is straightforward).
2. In Van der Laan et al. (2007), the authors argue that their result applies to the case that $|\mathcal{J}| = 1$ and the stacked generalization is learned (does depend on Z) by first discretizing the set of functions that the stacked generalization may become, thus reducing to the case of a finite number of constant stacked generalizations $\tilde{\mathcal{J}}$ as in Theorem 2 therein; and that the error term from this step is asymptotically negligible. Note, however, that by following this procedure the log term becomes $\log(1 + |\tilde{\mathcal{J}}| + m)$ (which explodes with the size of the discretization); whereas if you use the theorem above you get $\log(2+m)$ and does depend on a discretization. Therefore the theorem above is a significant improvement even in the case $|\mathcal{J}| = 1$.

5. Reduction to Oracle Inequalities

One of the main insights that led to Theorem 4.1 is the recognition that a sufficiently general and strong version of an oracle inequality (Theorem 5.1 below) can lead to significant gains in obtaining better and more general theoretical guarantees for stacked generalizations, as we proceed to illustrate.

We deviate from common notation by now letting D_0 and D_1 be *ordered* datasets of IID feature and target pairs; taking values in $(\mathbb{R}^d \times \mathbb{R}^r)^{n_0}$ and $(\mathbb{R}^d \times \mathbb{R}^r)^{n_1}$ respectively. Let X be a random variable in $\mathbb{R}^d \times \mathbb{R}^r$ following the same distribution. Let $\{\theta_\alpha\}_{\alpha \in \mathcal{J}}$ be a set of tabular algorithms that may take into account the order of the training dataset. Let $L : \mathbb{R}^{r'} \times \mathbb{R}^r \to \mathbb{R}$ be some loss function, and let $L_\alpha(D_0)((x,y)) := L(\theta_\alpha(D_0)(x), y)$. Finally let $\mathcal{F} := \{(x,y) \mapsto L_\alpha(D_0)((x,y)) | \alpha \in \mathcal{J}\}$, and let $(M(\mathcal{F}), v(\mathcal{F}))$ be its Bernstein numbers. Then the following holds.

Theorem 5.1. *With the notation above, for every $\hat{\alpha} \in \mathcal{J}$ let*

$$\begin{aligned}W_{\hat{\alpha}} := \Big\{ \tilde{\alpha} \in \mathcal{J} \Big| \\ \mathbb{E}(\int L_{\hat{\alpha}}(D_0)(X)d\mathbb{P}_1) \leq \mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X)d\mathbb{P}_1) \Big\},\end{aligned}$$

be the set of indices $\tilde{\alpha}$ for which $\theta_{\hat{\alpha}}$ outperforms $\theta_{\tilde{\alpha}}$ on validation. Further assume that $\mathcal{J}$ is a bounded subset of a finite dimensional Euclidean space, and that for every $D_0 \in (\mathbb{R}^d \times \mathbb{R}^r)^{n_0}$ the function that takes $\alpha \in \mathcal{J}$ to $L_\alpha(D_0) \in \mathcal{F}$ is Lipschitz with constant ℓ w.r.t. the infinity norm in $\mathcal{F}$. Then we have that for every $\delta > 0$, $1 \leq p \leq 2$, and sequence $\varepsilon_{n_1} > 0$,

$$\begin{aligned}&\mathbb{E}(\int L_{\hat{\alpha}}(D_0)(X)dP) \leq \\ &(1+2\delta) \inf_{\tilde{\alpha} \in W_{\hat{\alpha}}} \left(\mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X)dP) \right) \\ &\quad + \sup_{f \in \mathcal{F}}(B_f) + 2((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1},\end{aligned} \tag{3}$$

where

$$B_f := \frac{16\left(\frac{M(\mathcal{F})}{n_1^{1-\frac{1}{p}}} + \left(\frac{v(\mathcal{F})}{(\delta \int f dP)^{2-p}}\right)^{\frac{1}{p}}\right)\log(1 + N^{int}(\mathcal{J}, \frac{\varepsilon_{n_1}}{\ell}))}{n_1^{\frac{1}{p}-\frac{1}{2}}}.$$

In addition, the following hold:

1. *If we choose $\varepsilon_{n_1} := n_1^{-\frac{1}{2}-\epsilon}$ for any $\epsilon > 0$ then $\sup_{f \in \mathcal{F}}(B_f)$ and $2((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1}$ are $o(n_1)$.*

2. *If $\mathcal{J}$ is finite rather than finite dimensional, then the log term can be replaced with* $\log(1 + |\mathcal{J}|)$, *and we may remove the last summand* $2((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1}$.
3. *If one wants to artificially add to $W_{\hat{\alpha}}$ any finite number p of additional algorithms, then, in both the case that $\mathcal{J}$ is finite and finite-dimensional, one simply adds p to the inside of the* $\log$ *term.*

Remark 5.2. We remark that ℓ can easily be discerned from the Lipschitz constant of L in its first coordinate and the Lipschitz constants of each of the maps $\alpha \mapsto \theta_\alpha$. Note also that if $\mathcal{J}$ is compact then there exists a $\hat{\alpha}$ for which $W_{\hat{\alpha}} = \mathcal{J}$, in which case $\hat{\alpha}$ has the interpretation of being the index of the best performing stacked generalization on cross validation, and $\inf_{\tilde{\alpha}\in\mathcal{J}} (\mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X)dP))$ is achieved by some $\tilde{\alpha}$ that has the interpretation of being the oracle best stacked generalization.

We postpone the proof of this theorem to Appendix A, though it is important to point out that this is both a strengthening and a generalization of Lemma 2.3 (or more accurately of Lemmas 2.1 and 2.2) of (Van der Vaart et al., 2006) to the case where the datasets are ordered and to the case that $\mathcal{J}$ is potentially infinite. This is precisely the version that we require to prove Theorem 4.1 easily.

Proof. (of Theorem 4.1)
Apply Theorem 5.1 to the set of algorithms $\theta_\alpha(D_0) \coloneqq A_\alpha(Z) \circ (\eta_1(D_{00}), ..., \eta_m(D_{00}))$. □

Note that even in the case that $\mathcal{J}$ is finite it would not have been possible to use the oracle inequalities in Van der Vaart et al. (2006) directly to the algorithms $\theta_\alpha(D_0) \coloneqq A_\alpha(Z) \circ (\eta_1(D_{00}), ..., \eta_m(D_{00}))$ since these θ_α's depend on the order of D_0 (to separate out D_{00} and D_{01}); and that this is the reason that Van der Laan et al. (2007) first reduced to the case of constant stacked generalizations (via a discretization of the set of functions that the stacked generalization may become).

6. Case Study: Probabilistic Time Series Forecasting

Consider probabilistic forecasting algorithms that output a prediction for each time series in the dataset (henceforth "item"), for each timestamp in the forecast horizon, and for each of a list of predefined quantiles. In combining these algorithms into a weighted sum, it is a natural question whether it is a good idea to share the weights across items/timestamps/quantiles. As we will see, this translates naturally to choosing the best stacked generalization out of a family. While Theorem 4.1 does not apply directly outside of the tabular use case, we use it as motivation.

6.1. Setup for Stacked Generalizations in Forecasting

As is common in probabilistic time series forecasting, we consider a dataset containing N time series (henceforth "items"). We consider a base learners $\eta_1, ..., \eta_m$, where each one of them makes predictions for a fixed number of timestamps in the future h (henceforth "the forecast horizon"), and the output for each item and timestamp is a prediction of some fixed predicted quantiles q (e.g., $q = 3$ with quantiles $0.1, 0.5, 0.9$). In particular the output for each base learner is in $\mathbb{R}^{N\times h\times q}$. (See Appendix C.1 for details.)

Our interpretation of a stacked generalization in this setup is an algorithm that takes as input the output of all of the base learners trained up to some backtest window together with the true values from this backtest window, and outputs for each tuple (item, timestamp in the forecast horizon, and quantile) a weight for each of the base learners. (One example of such a stacked generalization is one that optimizes mean weighted quantile loss over the backtest window treating all of the weights as independent; another example would be one that optimizes mean weighted quantile loss on the backtest window, but with each base learner having the same weight regardless of the timestamp, item, and quantile. We refer to Appendix C.1 for the definition of mean weighted quantile loss.)

Once the weights are learned in this fashion on a single backtest window, the inference of the stacked generalization is a weighted sum of the base learners, where the weights vary based on item, timestamp, and quantile.

Given a family of stacked generalizations, we interpret choosing the best performing one based on cross validation as specified in Algorithm 1, where the notation is borrowed from Figure 1.

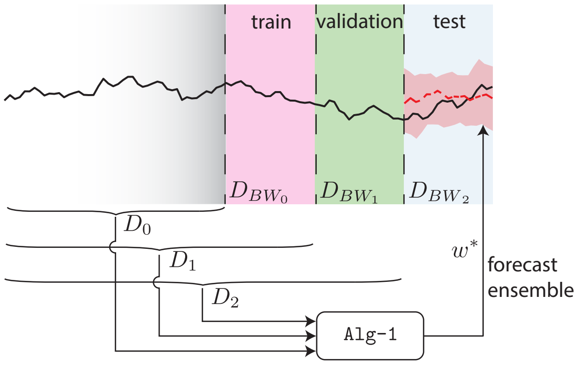

Figure 1. Dataset split for the purpose of choosing the best stacked generalization out of a family based on cross validatation. (See Algorithm 1.) Each D_{BW_i} contains as many timestamps as the forecast horizon length of the base learners. The split is done across all items.

Algorithm 1 Choosing a Stacked Generalization Out of Family in the Time Series Setting

```
Input: $m$ base learners: $\eta_1, \dots, \eta_m$, a time series dataset
    split as in Figure 1; and a family of stacked generalizations $\{A_\alpha\}_{\alpha \in \mathcal{J}}$.
Output: Predictions for $D_{\mathrm{BW}_2}$.
for values of $\alpha$ do
    Train $\eta_1, ..., \eta_m$ on $D_0$ and use their predictions together with $D_{\mathrm{BW}_0}$ to train $A_\alpha$. Store the resulting weights $w$.
    Train $\eta_1, ..., \eta_m$ on $D_1$ and $w$ to make a weighted sum prediction on $D_{\mathrm{BW}_1}$; compute mean weighted quantile loss.
end for
Store the $\hat{\alpha}$ for which the mean weighted quantile loss computed was minimal.
Retrain $A_{\hat{\alpha}}$ on the output of the models $\eta_1, ..., \eta_m$ trained on $D_1$, together with $D_{\mathrm{BW}_1}$. Store the resulting weights as $w^*$.
Train $\eta_1, ..., \eta_m$ on $D_2$ and output their weighted sum using $w^*$.
```

We remark that a uniform notation between tabular and the time series use cases it possible, albeit confusing. We refer to Appendix D for such a treatment, while in the remainder of the paper we make use of simplified notation.

6.2. A Family of Stacked Generalizations Controlling Uniformity of Weights

We propose a family of stacked generalizations indexed by $\alpha \in \mathbb{R}^4_{\geq 0}$; taking the output of the base learners together with true values for the backtest window on which they are making predictions, and finding the weights that optimize mean weighted quantile loss (see Appendix C.1 for the definition) together with the regularizers:

$$\sum_{d=1}^{3} \alpha_d \mathbb{H}(\sigma^{(d)}(w)) + \alpha_4 \sum_{i,j,k,l} |w^{(l)}_{i,j,k}|,$$

where $\mathbb{H}(\sigma^{(d)}(w))$ is an entropy term defined in detail in Appendix C.2 that controls whether the weights are forced to be uniform across items, timestamps, and quantiles. Thus, for example, the algorithm $A_{(0,0,0,0)}$ optimizes mean weighted quantile loss treated all the weights as independent; whereas $A_{(\alpha_1,0,0,0)}$ as α_1 goes to infinity would optimize mean weighted quantile loss under the constraint that for a specific timestamp and quantile the base learner weights don't vary across items.

6.3. Experiments

In this subsection, we train base learners $\{\eta_l(D_2)\}_l$ on the full historical data D_2, and compare our optimal stacked generalization ensemble $A_{\hat{\alpha}}(D_2)$ in Eqn. (12) to the predictions from the base learners and simple ensembling methods. We report the mean weighted quantile loss (Eqn. (8)) over the quantiles $\tau = [0.1, 0.5, 0.9]$ on the test set $L(A_{\hat{\alpha}}(D_2), D_{\mathrm{BW}_2})$ on the corresponding most recent in time backtest window D_{BW_2} (See Eqn. (9) to recall the definition of (D_2, D_{BW_2})). For each α, we use autodiff in PyTorch (Paszke et al., 2017) to find the optimal weights for each A_α in Algorithm 1 (using softmax to satisfy the constraint), and use scipy's implementation of COBYLA (Powell, 1994) to find the optimal $\hat{\alpha}$ since looping over an infinite set is impossible. (It is possible to do a grid search to find $\hat{\alpha}$, but we have empirically observed that using an optimization method is faster and leads to equivalent results.)

We test on real-world open-source datasets from the UCI data repository (Dheeru & Karra Taniskidou, 2017), Kaggle (Lai, 2017), and `M4` competition datasets (Makridakis et al., 2018) (see Table 5 in Appendix E).

Base Learners. We call the following $m = 6$ base learners from GluonTS (Alexandrov et al., 2020) with the default hyper-parameter settings: the local state space models ARIMA and ETS [2] (Hyndman et al., 2008), Non-Parametric Time Series (NPTS) (Gasthaus, 2016), the global deep learning models DeepAR (Flunkert et al., 2019) and MQ-CNN (Wen et al., 2017), and Rotbaum (XGBoost-based forecaster, a GluonTS implementation of TTSW from Hasson et al. (2021). We compare to the simple ensemble baselines in Appendix 6.3. The experiments are performed 5 times to compute confidence intervals.

Baselines. We compare to the following simple ensemble baselines:

- Mean: For each item, timestamp, and quantile, take a simple mean of all of the base learners.
- Median: For each item, timestamp, and quantile, take a simple median of all of the base learners.
- Global Best (GB) (D_1, D_{BW_1}): Of all of the possible combinations of algorithms to choose, choose the one for which the simple mean of the predictions of the chosen algorithms lead to the best performance on D_{BW_1}. The same algorithms are chosen across timestamps, quantiles, and items. (Greedy ensemble, (Caruna et al., 2004), is an approximation of this method in the case that there are many base learners.)
- Best (D_1, D_{BW_1}): Choose the single best base learner on D_{BW_1}.
- Unregularized: Use $\alpha = (0, 0, 0, 0)$ (in the sense of the family given in Subsection 6.2); i.e., always use the unregularized stacked generalization, rather than choosing the best stacked generalization via cross validation.

[2] The error from the ETS predictions on `Wiki` is too large to include, and so we omit ETS in the corresponding ensembles.

Table 1. Mean weighted quantile losses over quantiles $\tau = [0.1, 0.5, 0.9]$ for various base learners and ensembling strategies on real-world open-source time series datasets over 5 runs.

Base Learner / Ensemble Strategy	`Elec`	`Kaggle`	`M4-daily`	`Traf`	`Wiki`
ARIMA	0.105 ± 0.0	0.1201 ± 0.0	0.0341 ± 0.0	0.2219 ± 0.0	0.6008 ± 0.0
DeepAR	0.0585 ± 0.0011	0.2478 ± 0.0056	0.035 ± 0.0009	0.0987 ± 0.0014	0.3311 ± 0.0095
ETS	0.0796 ± 0.0203	**0.1193** ± **0.0004**	0.033 ± 0.0005	0.2999 ± 0.0	N/A
MQ-CNN	0.0698 ± 0.0018	0.2452 ± 0.0018	0.0408 ± 0.0025	0.2491 ± 0.0423	0.3639 ± 0.0038
NPTS	0.0536 ± 0.0005	0.1393 ± 0.0001	0.1218 ± 0.0001	0.1172 ± 0.0002	0.4798 ± 0.0001
Rotbaum	0.0603 ± 0.001	0.2098 ± 0.0	0.0342 ± 0.0003	0.1289 ± 0.0004	0.3983 ± 0.0022
mean	0.0616 ± 0.0034	0.1556 ± 0.0006	0.0431 ± 0.0002	0.1458 ± 0.004	0.3866 ± 0.0029
median	0.0549 ± 0.0012	0.1323 ± 0.0005	0.0351 ± 0.0001	0.123 ± 0.0038	0.3489 ± 0.0027
GB (D_1, D_{BW_1})	0.0531 ± 0.0004	0.16 ± 0.0012	0.0327 ± 0.0001	0.0987 ± 0.0014	0.3412 ± 0.0061
Best (D_1, D_{BW_1})	0.056 ± 0.0024	0.1201 ± 0.0	0.0332 ± 0.0005	0.0987 ± 0.0013	0.3339 ± 0.0148
Unregularized	**0.0475** ± **0.0005**	0.1979 ± 0.0009	0.028 ± 0.0001	0.0986 ± 0.0017	0.3366 ± 0.0008
Ours	0.0494 ± 0.0006	0.1663 ± 0.0003	**0.0266** ± **0.0003**	**0.094** ± **0.001**	**0.3187** ± **0.002**

Table 2. Mean weighted quantile losses over quantiles $\tau = [0.1, 0.5, 0.9]$ for various base learners and ensembling strategies on real-world open-source time series datasets, where Gaussian noise is added to the base learner predictions across time over 5 runs.

Base Learner / Ensemble Strategy	`Elec`	`Kaggle`	`M4-daily`	`Traf`	`Wiki`
ARIMA	1.9146 ± 0.1236	1.102 ± 0.0038	0.0367 ± 0.0001	2.9926 ± 0.0089	0.7672 ± 0.0018
DeepAR	1.3185 ± 0.0826	3.0007 ± 0.0359	0.0355 ± 0.0008	2.1192 ± 0.0161	0.464 ± 0.0058
ETS	0.5137 ± 0.0758	0.2298 ± 0.0042	0.0332 ± 0.0004	1.2715 ± 0.0071	N/A
MQ-CNN	2.375 ± 0.1399	4.5208 ± 0.035	0.0481 ± 0.0062	3.4619 ± 0.4748	0.5667 ± 0.0096
NPTS	1.2909 ± 0.0622	2.5102 ± 0.0106	0.1235 ± 0.0002	1.7341 ± 0.0117	0.5871 ± 0.001
Rotbaum	0.3656 ± 0.0211	4.5134 ± 0.0157	0.0343 ± 0.0003	0.5403 ± 0.0032	0.438 ± 0.0015
mean	0.5672 ± 0.0424	1.2186 ± 0.0046	0.0436 ± 0.0003	0.8564 ± 0.0438	0.4283 ± 0.0027
median	0.521 ± 0.0406	0.8816 ± 0.0057	0.0356 ± 0.0001	0.8325 ± 0.0159	0.374 ± 0.0023
GB (D_1, D_{BW_1})	0.3044 ± 0.0346	0.2298 ± 0.0042	0.0329 ± 0.0001	0.5403 ± 0.0032	0.4054 ± 0.0051
Best (D_1, D_{BW_1})	0.3656 ± 0.0211	0.2298 ± 0.0042	0.0332 ± 0.0004	0.5403 ± 0.0032	0.438 ± 0.0015
Unregularized	0.0589 ± 0.0009	0.2385 ± 0.0002	0.0274 ± 0.0001	0.134 ± 0.0007	0.343 ± 0.0007
Ours	**0.0587** ± **0.0007**	**0.1965** ± **0.0007**	**0.0268** ± **0.0003**	**0.1334** ± **0.0007**	**0.3294** ± **0.0016**

We remark that if we had used only one backtest window, namely if we learned α on the same backtest window on which the weights are optimized, then α would have to equal $(0, 0, 0, 0)$, and so this setup is equivalent to this baseline.

Benchmarking with Raw Predictions. We first compare the predictions from our method to the raw predictions from the base learners, and the baseline ensembling methods.

We remark that unlike in the literature about the "forecast combination puzzle" (Section 2), our method offers significant performance boost compared with simple average. We attribute this discrepancy to the idealized assumptions in the literature, most prominent among them being that they assume that the weights remain the same across timestamps or any other dimension. Table 1 shows the results.

Synthetic Experiments Adding Noise to the Predictions Across Various Dimensions. We have seen in Table 3 that our algorithm works well in a natural setting. But given that the family we have chosen (in Subsection 6.2) is meant to find the ideal degree to which the weights are forced to be similar across item/timestamps/quantiles, in this section we create synthetic experiments the stress-test this ability, and give strong evidence that it is able to adjust to these disruptions.

To be specific, for each set of base learner predictions $\eta_l(D_n) = \{\hat{z}^{(l)}_{i,j,k}(D_n)\}^{i,l}_{j,k}$, where $n = 0, 1, 2$, i indexes the items, j the forecast horizon, and k the quantiles, we introduce Gaussian noise. Since we are applying the noise for each data training and validation splits, in the following we simply use the notation $\{\hat{z}^{(l)}_{i,j,k}\}^{i,l}_{j,k}$. Let $s^{(l)}_{i,k} = \text{std}_j(\{\hat{z}^{(l)}_{i,j,k}\})$ be the empirical standard deviation computed over the time dimension j for fixed i, k, l. For each simulation, we let the selected noise be the same for all the backtest windows. For each i, j, k and l, we then add Gaussian noise $\epsilon^{(l)}_{i,j,k}$ to the base learner prediction, i.e.

$$\hat{z}^{(l)}_{i,j,k} \mapsto \hat{z}^{(l)}_{i,j,k} + N(0, \epsilon^{(l)}_{i,j,k}),$$

where the noise $\epsilon^{(l)}_{i,j,k}$ varies in the following three ways:

1. Across time: For each l, let $r_l \sim U(0, 0.5)$ and $\epsilon^{(l)}_{i,j,k} = 2jr_l s^{(l)}_{i,k}/h$ for forecast horizon h. In this experiment, we design the noise to be increasing with the time

Table 3. Mean weighted quantile losses over quantiles $\tau = [0.1, 0.5, 0.9]$ for various base learners and ensembling strategies on real-world open-source time series datasets, where Gaussian noise is added to the base learner predictions across items over 5 runs.

Base Learner / Ensemble Strategy	`Elec`	`Kaggle`	`M4-daily`	`Traf`	`Wiki`
ARIMA	0.2026 ± 0.0019	0.1223 ± 0.0001	0.0348 ± 0.0001	0.3707 ± 0.0012	0.6081 ± 0.0004
DeepAR	0.283 ± 0.011	0.2758 ± 0.0051	0.0353 ± 0.0008	0.4442 ± 0.0053	0.3564 ± 0.0085
ETS	0.3534 ± 0.031	**0.1212** ± **0.0005**	0.0337 ± 0.0005	0.6846 ± 0.0027	N/A
MQ-CNN	0.3297 ± 0.0237	0.2752 ± 0.0009	0.0425 ± 0.0033	0.4601 ± 0.0603	0.3796 ± 0.004
NPTS	0.314 ± 0.0151	0.1636 ± 0.0001	0.1231 ± 0.0002	0.4158 ± 0.0023	0.4981 ± 0.0006
Rotbaum	0.437 ± 0.0269	0.2381 ± 0.0004	0.0355 ± 0.0003	0.4265 ± 0.0016	0.4649 ± 0.0039
mean	0.1338 ± 0.0043	0.1593 ± 0.0005	0.0434 ± 0.0002	0.2218 ± 0.0034	0.3956 ± 0.0033
median	0.1526 ± 0.0065	0.1354 ± 0.0001	0.0357 ± 0.0002	0.2179 ± 0.0066	0.3539 ± 0.0031
GB (D_1, D_{BW_1})	0.1333 ± 0.0039	0.1618 ± 0.001	0.0331 ± 0.0002	0.2099 ± 0.0053	0.3519 ± 0.0058
Best (D_1, D_{BW_1})	0.2026 ± 0.0019	0.1223 ± 0.0001	0.0337 ± 0.0005	0.3707 ± 0.0012	0.3732 ± 0.0107
Unregularized	0.0521 ± 0.002	0.2209 ± 0.0006	0.0276 ± 0.0001	0.1125 ± 0.0003	0.3384 ± 0.0007
Ours	**0.0505** ± **0.0017**	0.156 ± 0.0008	**0.0268** ± **0.0003**	**0.1094** ± **0.0004**	**0.3235** ± **0.0016**

Table 4. Mean weighted quantile losses over quantiles $\tau = [0.1, 0.5, 0.9]$ for various base learners and ensembling strategies on real-world open-source time series datasets, where Gaussian noise is added to the base learner predictions across quantiles over 5 runs.

Base Learner / Ensemble Strategy	`Elec`	`Kaggle`	`M4-daily`	`Traf`	`Wiki`
ARIMA	0.2347 ± 0.0067	0.1269 ± 0.0001	0.0344 ± 0.0	0.3877 ± 0.0008	0.606 ± 0.0005
DeepAR	0.789 ± 0.0573	0.3571 ± 0.0069	0.0369 ± 0.0007	1.2155 ± 0.013	0.4525 ± 0.004
ETS	0.4443 ± 0.0687	**0.1201** ± **0.0003**	0.0359 ± 0.0004	0.9363 ± 0.0111	N/A
MQ-CNN	0.9111 ± 0.0167	0.2904 ± 0.0013	0.0531 ± 0.0096	1.2778 ± 0.2005	0.4692 ± 0.0076
NPTS	0.6306 ± 0.024	0.2942 ± 0.0011	0.1269 ± 0.0002	0.9207 ± 0.0064	0.5888 ± 0.0017
Rotbaum	0.5671 ± 0.0525	0.2497 ± 0.0004	0.0374 ± 0.0005	0.71 ± 0.0024	0.5356 ± 0.0072
mean	0.2511 ± 0.0173	0.1743 ± 0.0007	0.0441 ± 0.0004	0.3715 ± 0.0156	0.4107 ± 0.0032
median	0.2444 ± 0.0222	0.1394 ± 0.0003	0.036 ± 0.0002	0.3749 ± 0.0106	0.3664 ± 0.0027
GB (D_1, D_{BW_1})	0.2268 ± 0.0103	0.1455 ± 0.0014	0.0335 ± 0.0002	0.3634 ± 0.0036	0.4057 ± 0.0046
Best (D_1, D_{BW_1})	0.2347 ± 0.0067	0.1256 ± 0.0022	0.0344 ± 0.0	0.3877 ± 0.0008	0.4692 ± 0.0076
Unregularized	0.0555 ± 0.0006	0.235 ± 0.0003	**0.0268** ± **0.0001**	0.1233 ± 0.0005	0.3431 ± 0.0005
Ours	**0.0541** ± **0.0015**	0.1637 ± 0.0028	0.0269 ± 0.0002	**0.1228** ± **0.0004**	**0.3285** ± **0.001**

index j to mimic the natural order of time, where the predictions become worse over time. (See Table 2).

2. Across items: For each l and $i = 1, ..., N$ let $r_{i,l} \sim U(0, 2)$ and $\epsilon_{i,j,k}^{(l)} = \epsilon_{i,k}^{(l)} = r_{i,l} s_{i,k}^{(l)}$ (See Table 3).
3. Across quantiles: For each l and $k = 1, 2, 3$, let $r_{k,l} \sim U(0, 2)$ and $\epsilon_{i,j,k}^{(l)} = \epsilon_{i,k}^{(l)} = 2 r_{k,l} s_{i,k}^{(l)}$ (See Table 4).

See Tables 2 (adding noise across time) and 3 (adding noise across items) and 4 (adding noise across quantiles) for results. In these three synthetic experiments, we see that our method is successful in adjusting for the noises, and gives similar results in those in Table 1 with no noise added. We also see that using the unregularized loss, which corresponds to choosing $\alpha = (0, 0, 0, 0)$ rather than learning it via cross-validation, is a surprisingly close second. It is not remarkable that the rest of the baselines do not perform well since the other baselines except for median use the same weights for all timestamps/quantiles/items, and median is another one size fits all solution that does not vary across timestamps/quantiles/items.

7. Conclusion

In this paper, we provide theoretical guarantees for families of stacked generalizations, which are commonly used in practice in state-of-the-art ensembling methods. We extend the work in Van der Laan et al. (2007) that shows that choosing the best stacked generalization from a finite family of constant stacked generalizations based on cross-validated performance does not perform much worse than the oracle best, both by allowing the stacked generalizations to be learned rather than constant, and by giving a result for finite-dimensional, rather than finite, families. We then show a particular family of stacked generalizations in the probabilistic time series forecasting use case. We formulate the problem of finding this family as a regularized regression problem, where the choice of regularizers parametrizes the family. Our experiments demonstrate that our intuition that designing a family of stacked generalizations in the time series case, where each member of the family gives different flexibility for the weights to vary across the various timestamps/quantiles/items is effective.

References

Kaggle survey 2020. https://www.kaggle.com/kaggle-survey-2020, 2020.

Kaggle survey 2021. https://www.kaggle.com/kaggle-survey-2021, 2021.

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

Bates, J. M. and Granger, C. W. The combination of forecasts. *Journal of the Operational Research Society*, 20 (4):451–468, 1969.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Caruna, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Claeskens, G., Magnus, J. R., Vasnev, A. L., and Wang, W. The forecast combination puzzle: A simple theoretical explanation. *International Journal of Forecasting*, 32(3): 754–762, 2016.

Cruz, R. M., Sabourin, R., and Cavalcanti, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017.

Donaldson, R. G. and Kamstra, M. Forecast combining with neural networks. *Journal of Forecasting*, 15(1):49–61, 1996.

Elliott, G. Averaging and the optimal combination of forecasts. *University of California, San Diego*, 2011.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Flunkert, V., Salinas, D., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting, arXiv:1704.04110*, 2019.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Gasthaus, J. Non-parametric time series forecaster. Technical Report, Amazon, 2016.

Gastinger, J., Nicolas, S., Stepić, D., Schmidt, M., and Schülke, A. A study on ensemble learning for time series forecasting and the need for meta-learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Hasson, H., Wang, B., Januschowski, T., and Gasthaus, J. Probabilistic forecasting: A level-set approach. *Advances in neural information processing systems*, 34, 2021.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

Kim, T., Fakoor, R., Mueller, J., Tibshirani, R. J., and Smola, A. J. Deep quantile aggregation. *arXiv preprint arXiv:2103.00083*, 2021.

Lai. Dataset of Kaggle Competition Web Traffic Time Series Forecasting, Version 3, August 2017.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Makridakis, S. et al. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802 – 808, 2018.

Massaoudi, M., Refaat, S. S., Chihi, I., Trabelsi, M., Oueslati, F. S., and Abu-Rub, H. A novel stacked generalization ensemble-based hybrid lgbm-xgb-mlp model for short-term load forecasting. *Energy*, 214:118874, 2021.

Moon, J., Jung, S., Rew, J., Rho, S., and Hwang, E. Combination of short-term load forecasting models based on a stacking ensemble approach. *Energy and Buildings*, 216: 109921, 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop*, 2017.

Powell, M. J. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pp. 51–67. Springer, 1994.

Smith, J. and Wallis, K. F. A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics*, 71(3):331–355, 2009.

Stock, J. H. and Watson, M. W. Combination forecasts of output growth in a seven-country data set. *Journal of forecasting*, 23(6):405–430, 2004.

Ting, K. M. and Witten, I. H. Stacked generalization: when does it work? 1997a.

Ting, K. M. and Witten, I. H. Stacking bagged and dagged models. 1997b.

Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

Van der Vaart, A. W., Dudoit, S., and Van der Laan, M. J. Oracle inequalities for multi-fold cross validation. *Statistics & Decisions*, 24(3):351–371, 2006.

Wellner, J. et al. *Weak convergence and empirical processes: with applications to statistics*. Springer Science & Business Media, 2013.

Wen, R., Torkkola, K., and Narayanaswamy, B. A multi-horizon quantile recurrent forecaster. *NIPS Workshop on Time Series, arXiv:1711.11053*, 2017.

Wolpert, D. H. Stacked generalization. *Neural networks*, 5 (2):241–259, 1992.

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

A. Omitted Proofs

In this section we provide the proofs that were omitted from Section 5.

Using the notation of Section 3, let $\mathbb{G}_i := \sqrt{n_i}(\mathbb{P}_i - P)$ be the "empirical process" of $\mathbb{P}_i$ for $i = 1, 2$. The following lemma is a straightforward generalization of Lemma 2.1 in (Van der Vaart et al., 2006) to the case of infinitely many θ_αs and order-dependent algorithms. As the proof is the same *mutatis mutandis* as in (Van der Vaart et al., 2006), we omit it.

Lemma A.1. *For every $\hat{\alpha} \in \mathcal{J}$, let*

$$W_{\hat{\alpha}} := \Big\{\tilde{\alpha} \in J \Big| \mathbb{E}(\int L_{\hat{\alpha}}(D_0)(X) d\mathbb{P}_1) \\ \leq \mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X) d\mathbb{P}_1)\Big\},$$

be the set of indices $\tilde{\alpha}$ for which $\theta_{\hat{\alpha}}$ outperforms $\theta_{\tilde{\alpha}}$ on validation. Then for any $\delta \geq -1$:

$$\begin{aligned}&\mathbb{E}(\int L_{\hat{\alpha}}(D_0)(X)dP) \leq \\ &\quad (1+2\delta) \inf_{\tilde{\alpha} \in W_{\hat{\alpha}}} \left(\mathbb{E}(\int L_{\tilde{\alpha}}(D_0)(X)dP)\right) \\ &\quad + \frac{1}{\sqrt{n_1}} \mathbb{E}(\sup_{\alpha} \int L_\alpha(D_0)(X) d((1+\delta)\mathbb{G}_1 - \delta\sqrt{n_1}P)) \\ &\quad + \frac{1}{\sqrt{n_1}} \mathbb{E}(\sup_{\alpha} - \int L_\alpha(D_0)(X) d((1+\delta)\mathbb{G}_1 + \delta\sqrt{n_1}P)).\end{aligned} \tag{4}$$

Our goal, therefore, is to bound the last two terms. In the case that $|\mathcal{J}|$ is finite this was done in Lemma 2.2 of (Van der Vaart et al., 2006).

Lemma A.2. *(Lemma 2.2 of (Van der Vaart et al., 2006)) Assume that $\mathcal{J}$ is finite, that $\mathcal{F}$ has Bernstein numbers $(M(\mathcal{F}), v(\mathcal{F}))$, and that the θ_α's are invariant of the training dataset's order. Then for every $\delta > 0$ and $1 \leq p \leq 2$:*

$$\begin{aligned}&\mathbb{E}(\max_{f \in \mathcal{F}} \pm \int f d(\mathbb{G}_1 - \delta\sqrt{n_1}P)) \leq \\ &\quad \frac{8}{n_1^{\frac{1}{p}-\frac{1}{2}}} \log(1 + |\mathcal{J}|) \max_{f \in \mathcal{F}} \left(\frac{M(\mathcal{F})}{n_1^{1-\frac{1}{p}}} + \left(\frac{v(\mathcal{F})}{(\delta \int f dP)^{2-p}}\right)^{\frac{1}{p}} \right).\end{aligned} \tag{5}$$

While the lemma assumes the θ_α's are invariant of the training dataset order, this assumption is in fact not used in its proof. Thus it suffices to reduce the case that $\mathcal{J}$ is finite dimensional to the lemma above. This is done via a discretization of $\mathcal{J}$ itself. (Note that this is not the same as discretizing the set of functions that the stacked generalizations may become.)

Theorem A.3. *Under the assumptions of Theorem 5.1, for every $\delta > 0$, $1 \leq p \leq 2$, and sequence $\varepsilon_{n_1} > 0$:*

$$\begin{aligned}&\mathbb{E}(\sup_{f \in \mathcal{F}} \pm \int f d(\mathbb{G}_1 - \delta\sqrt{n_1}P)) \leq \\ &\quad \sup_{f \in \mathcal{F}} \left(\frac{8\left(\frac{M(\mathcal{F})}{n_1^{1-\frac{1}{p}}} + \left(\frac{v(\mathcal{F})}{(\delta \int f dP)^{2-p}}\right)^{\frac{1}{p}}\right) \log(1 + N^{int}(\mathcal{J}, \frac{\varepsilon_{n_1}}{\ell}))}{n_1^{\frac{1}{p}-\frac{1}{2}}} \right) \\ &\quad + ((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1}.\end{aligned} \tag{6}$$

Proof. (of Theorem A.3)

First note that if $\|f - f'\|_\infty < \varepsilon_{n_1}$ then

$$\begin{aligned}
|\mathbb{G}_1 f - \delta\sqrt{n_1}Pf - (\mathbb{G}_1 f' - \delta\sqrt{n_1}Pf')| &\leq |\mathbb{G}_1 f - \mathbb{G}_1 f'| + \delta\sqrt{n_1}|P(f - f')| \\
&\leq n_1^{-\frac{1}{2}} \sum_{l=1}^{n_1} |f(X_l^1) - f'(X_l^1)| + (n_1^{-\frac{1}{2}} + \delta\sqrt{n_1})|P(f - f')| \\
&\leq \sqrt{n_1}\varepsilon_{n_1} + (n_1^{-\frac{1}{2}} + \delta\sqrt{n_1})\varepsilon_{n_1} \\
&= ((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1},
\end{aligned}$$

and therefore, after taking suprema and expected values, we get that for every subset $\mathcal{F}'$ of $\mathcal{F}$ for which every element in $\mathcal{F}$ is ε_{n_1}-close to an element in $\mathcal{F}'$

$$\mathbb{E}(\sup_{f\in\mathcal{F}} \pm \int f d(\mathbb{G}_1 - \delta\sqrt{n_1}P)) \leq \mathbb{E}(\sup_{f'\in\mathcal{F}'} \pm \int f' d(\mathbb{G}_1 - \delta\sqrt{n_1}P)) + ((1+\delta)\sqrt{n_1} + \frac{1}{\sqrt{n_1}})\varepsilon_{n_1}.$$

We now proceed to reduce Theorem A.3 to Lemma A.2 above, which applies to the case of finite $\mathcal{J}$. By our assumptions, we have that $N^{\text{int}}(\mathcal{F}, \ell\varepsilon) \leq N^{\text{int}}(\mathcal{J}, \varepsilon)$, or in other words $N^{\text{int}}(\mathcal{F}, \varepsilon) \leq N^{\text{int}}(\mathcal{J}, \frac{\varepsilon}{\ell})$.

Combining Lemma A.2 applied to the centers $\mathcal{F}'$ of a minimal ε_{n_1}-covering of $\mathcal{F}$ with the inequality above, the result follows. □

Proof. (of Theorem 5.1)
The main portion of the theorem follows directly from Theorem A.3 and Lemma A.1.

The second portion of the theorem follows from the inequality $N^{\text{int}}(\mathcal{J}, \frac{\varepsilon}{\ell}) \leq N^{\text{ext}}(\mathcal{J}, \frac{\varepsilon}{2\ell})$ (with N^{ext} representing the external covering number), as well as the fact that if $\mathcal{J}$ is contained in a K-ball around the origin in an ambient Euclidean space of dimension M, then we have the following inequality for the external covering number:

$$N^{\text{ext}}\left(\mathcal{J}, \frac{\varepsilon_{n_1}}{2\ell}\right) \leq \left(\frac{4\ell K\sqrt{M}}{\varepsilon_{n_1}}\right)^M.$$

Therefore if one chooses $\varepsilon_{n_1} := n_1^{-\frac{1}{2}-\epsilon}$ for some $\epsilon > 0$, then $\log(1 + N^{\text{int}}(\mathcal{J}, \frac{\varepsilon_{n_1}}{2\ell}))$ is order of magnitude of $\log(n_1)$. Further, if one wants to include p additional algorithms as stacked generalizations then that can be done by adding them into both $\mathcal{F}$ and $\mathcal{F}'$ in the proof of Theorem A.3, and therefore results in the log term described. □

B. Tabular Analogue of the Family in Section 6

In this appendix we point out that the construction in Section 6 can be employed in the tabular situation as well, for any case where we have a decomposition of the prediction space $\mathbb{R}^{r'} \cong \mathbb{R}^{r'_1} \bigotimes \cdots \bigotimes \mathbb{R}^{r'_p}$ of the base learners $\eta_1, ..., \eta_m$ into a tensorial factors. For $\alpha := (\alpha_1, ..., \alpha_p)$ we construct a stacked generalization $A_\alpha(D_0)$ as $(x, \eta_1(D_0)(x), ..., \eta_m(D_0)(x)) \mapsto \sum_{\forall j \; 1 \leq d_j \leq r'_i} \left(\sum_l w_{(d_i)_{i=1}^p, l} \eta_l(D_0)(x) \right) e_{d_1} \otimes \cdots \otimes e_{d_p}$. The weights are chosen to minimize

$$f_\alpha(D_0) := \sum_{X \in D_0} L_\alpha(X) + \sum_{i=1}^{p} \alpha_i \mathbb{H}(\sigma^{(i)}(w)) \tag{7}$$

where $\mathbb{H}^{(i)}$ is the entropy term similar to Section 6. It is also possible to add other regularization terms such as l_1 regularization as we did in Section 6.

C. Experimental Detail

C.1. Probabilistic Time Series Forecasting Problem Definition

We first define the time-series data as $D := \{z_{i,j'}\}^{i}_{j'},\ z_{i,j'} \in \mathbb{R}$, such that $i = 1, 2, \ldots, N$ denotes the i^{th} time-series (henceforth, item) with a total of N items, and for each item-i, $j' = 1, 2, \ldots, T_i$ denotes the time-points with the historical length T_i. For simplicity, we assume $\forall i\ T_i = T$. The goal of a probabilistic time-series forecasting algorithm is to output a probabilistic prediction for the future h time-steps for each time-series, where h is pre-determined, for quantiles $0 \le \tau_1, \ldots, \tau_q \le 1$. For the notational purpose and to avoid repetition, we reserve $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, h$, and $k = 1, 2, \ldots, q$ through rest of the paper, and henceforth drop the index range for i, j, k. The output consists of estimates $\{\hat{z}_{i,j,k}\}^{i}_{j,k}$ of the τ_k-quantiles of $z_{i,T+j}$ conditioned on historical data.

Given computed predictions $\hat{Z}(D) := \{\hat{z}_{i,j,k}\}^{i}_{j,k}$ and the true values $D_{\text{BW}} := \{z_{i,T+j}\}^{i}_{j}$, we aim to minimize the mean weighted quantile loss:

$$L(\hat{Z}(D), D_{\text{BW}}) := \frac{2}{q} \frac{\sum_{i,j,k} \max\{\tau_k(z_{i,T+j} - \hat{z}_{i,j,k}), (1 - \tau_k)(\hat{z}_{i,j,k} - z_{i,T+j})\}}{\sum_{i,j} |z_{i,T+j}|}. \tag{8}$$

C.2. Family of Stacked Generalizations for Probabilistic Time Series Forecastings

Let $\eta_1, \ldots, \eta_m$ be m arbitrary probabilistic time-series forecasting algorithms. In order to simplify the notation, if D is time-series data as defined in Section C.1, we let $\eta_l(D)$ be η_l trained on D and then making inferences on D; i.e., $\eta_l(D) = \{\hat{z}^{(l)}_{i,j,k}(D)\}^{i}_{j,k}$, which consists of the predictions for each item in D for the h many unseen timestamps into the forecast horizon after the last seen value for q quantiles; i.e., $\eta_l(D) = \{\hat{z}^{(l)}_{i,j,k}(D)\}^{i}_{j,k}$.

Since cross-validation is more subtle in the time series use case, we introduce some additional notation. In particular, we define training and backtest window datasets pairs, where a backtest window consists of the true values on a subsequent window of length h after the last training time point as:

$$(D_n, D_{\text{BW}_n}) := (\{z_{i,j'}\}^{i}_{j'},\ \{z_{i,T-\mu h+j}\}^{i}_{j}), \tag{9}$$

for $j' = 1, \ldots, T - \mu h$, $\mu = 2 - n$, $n = 0, 1, 2$. Here we use two pairs of training and validation sets, that is (D_0, D_{BW_0}) and (D_1, D_{BW_1}). Note that D and D_{BW} in Section C.1 are equivalent to to D_2 and D_{BW_2}, respectively.

For every choice of ensemble weights $w = \{w^{(l)}_{i,j,k}\}^{i,l}_{j,k}$, $l = 1, 2, \ldots, m$ we denote:

$$\hat{Z}(D, w) := \left\{ \sum_{l} w^{(l)}_{i,j,k}(D) \hat{z}^{(l)}_{i,j,k}(D) \right\}^{i}_{j,k}, \tag{10}$$

as the weighted ensemble combination. The weights are restricted to sum up to 1; namely, for every i, j, k we have that $\sum_l w^{(l)}_{i,j,k}(D) = 1$. We let

$$f_\alpha[(D_n, D_{\text{BW}_n}), w] = L\big(\hat{Z}(D_n, w), D_{\text{BW}_n}\big) + \sum_{d=1}^{3} \alpha_d \mathbb{H}(\sigma^{(d)}(w)) + \alpha_4 \sum_{i,j,k,l} |w^{(l)}_{i,j,k}|, \tag{11}$$

be the regularized mean weighted quantile loss with L given by Eqn. (8) and ensemble estimate $\hat{Z}(D_n, w)$ by Eqn. (10). We use an entropy of softmax regularizer:

$$\mathbb{H}(\sigma^{(d)}(w)) := \sum_{i,j,k,l} \sigma^{(d)}(w^{(l)}_{i,j,k}) \log(\sigma^{(d)}(w^{(l)}_{i,j,k})),$$

where $\sigma^{(d)}(w^{(l)}_{i,j,k}) = \exp(w^{(l)}_{i,j,k}) / \sum_{\mathcal{I}(d)} \exp(w^{(l)}_{i,j,k})$ denotes the softmax with the denominator summed over the double summations specified by d, that is, $\mathcal{I}(1), \mathcal{I}(2), \mathcal{I}(3)$ are $(j,k), (i,k), (i,j)$ respectively for $d = 1, 2, 3$. The role of the entropy regularizing parameters $\{\alpha_d\}_d$ for $d = 1, 2, 3$ in this family is to control how much the ensemble weights are allowed to vary across items, timestamps in the forecast horizon, quantiles, respectively. Lastly, α_4 denotes the l_1 regularizer parameter, which specifies how much to be biased towards preferring a single algorithm versus diversifying.

We follow Algorithm 1 to compute the optimal ensembling weights w^*, and output the optimal $\tilde{\alpha}$ ensembling member from our family of stacked generalizations:

$$A_{\tilde{\alpha}}(D_2) := \hat{Z}(D_2, w^*), \tag{12}$$

where $\hat{Z}$ is the weighted linear combination of base learners in Eqn. (10).

D. Towards a Uniform Treatment of Both Tabular and Time Series Data

A prominent difference in the way that we have treated tabular and time series data is in the meaning of "cross validation". In Section 3, the validation set consists of a different set of datapoints, whereas in Section 6 the validation set is in the same time series but with a different forecast horizon. In this appendix we will attempt to unify the notation for both cases.

The way to bridge this gap is to replace the dimension d of feature space, from Section 3, with the first infinite ordinal. In more colloquial terms, feature space will be the disjoint union $\dot{\bigcup}_{d \in \mathbb{N}} \mathbb{R}^d$. We keep the dimension of prediction space fixed, since in our time series forecasting setup infernces are always fixed size (length of forecast horizon times number of items times number of quantiles). In this way we may phrase cross validation for time series in the terminology of Section 3, by allowing each dataset to consist of a single datapoint. To wit, let D_n, D_{BW_n} for $n = 0, 1$ be as in Section 6, and let $\tilde{D}_n := \{(D_n, D_{\mathrm{BW}_n})\}$ for $n = 0, 1$. Then cross validation in the sense of Section 6 is simply cross-validation in the sense of Section 3 for $\tilde{D}_0$ and $\tilde{D}_1$.

In order to apply any of the theory of the tabular case, we must endow $(\dot{\bigcup}_{d \in \mathbb{N}} \mathbb{R}^d) \times \mathbb{R}^{r'}$, where r' is the dimension of prediction space, with a topology and a probability measure. While this space has a natural topology (which is even metrizable) by letting each $\mathbb{R}^d$ be a separate connected component in $\dot{\bigcup}_{d \in \mathbb{N}} \mathbb{R}^d$, this topology does not jibe with intuition. For example, two time series that are identical except that one has one additional timestamp compared to the other would be very far away. Perhaps a more natural choice would be a topology induced by dynamic time warping. Once a topology is fixed, one has to choose a (Borel) probability measure on $(\dot{\bigcup}_{d \in \mathbb{N}} \mathbb{R}^d) \times \mathbb{R}^{r'}$.

In whatever way that these choices are done, it is unrealistic to assume that the datapoint in $\tilde{D}_0$ is independent of the one in $\tilde{D}_1$. This implies that cross validation in the time series case cannot truly be brought into the fold of the tabular theory. Nevertheless, the tabular theory is directional.

E. Dataset Details

Table 5 lists the details of the datasets that we use in our experiments.

Table 5. Summary of dataset statistics, where `Elec` and `Traf` are from the UCI data repository ((Dheeru & Karra Taniskidou, 2017)), `Kaggle` and `Wiki` from Kaggle ((Lai, 2017)), and 6 different `M4` competition datasets ((Makridakis et al., 2018)).

DOMAIN	NAME	SUPPORT	FREQ	NO. TS	AVG. LEN	PRED. LEN
electrical load	`Elec`	$\mathbb{R}^+$	H	321	21044	24
Rossman	`Kaggle`	$\mathbb{N}$	D	500	1736	90
M4 forecasting competition	`M4-daily`	$\mathbb{R}^+$	D	4227	2357	14
road traffic	`Traf`	$[0, 1]$	H	862	14036	24
visit counts of wikipedia pages	`Wiki`	$\mathbb{N}$	D	9535	762	7