Test-Time Collective Prediction

Celestine Mendler-Dünner[*]
MPI for Intelligent Systems, Tübingen
cmendler@tuebingen.mpg.de

Wenshuo Guo
University of California, Berkeley
wguo@cs.berkeley.edu

Stephen Bates
University of California, Berkeley
stephenbates@cs.berkeley.edu

Michael I. Jordan
University of California, Berkeley
jordan@cs.berkeley.edu

Abstract

An increasingly common setting in machine learning involves multiple parties, each with their own data, who want to jointly make predictions on future test points. Agents wish to benefit from the collective expertise of the full set of agents to make better predictions than they would individually, but may not be willing to release labeled data or model parameters. In this work, we explore a decentralized mechanism to make collective predictions at test time, that is inspired by the literature in social science on human consensus-making. Building on a query model to facilitate information exchange among agents, our approach leverages each agent's pre-trained model without relying on external validation, model retraining, or data pooling. A theoretical analysis shows that our approach recovers inverse mean-squared-error (MSE) weighting in the large-sample limit which is known to be the optimal way to combine independent, unbiased estimators. Empirically, we demonstrate that our scheme effectively combines models with differing quality across the input space: the proposed consensus prediction achieves significant gains over classical model averaging, and even outperforms weighted averaging schemes that have access to additional validation data. Finally, we propose a decentralized Jackknife procedure as a tool to evaluate the sensitivity of the collective predictions with respect to a single agent's opinion.

1 Introduction

Large-scale datasets are often collected from diverse sources, by multiple parties, and stored across different machines. In many scenarios centralized pooling of data is not possible due to privacy concerns, data ownership, or storage constraints. The challenge of doing machine learning in such distributed and decentralized settings has motivated research in areas of federated learning [Konečný et al., 2015, McMahan et al., 2017], distributed learning [Dean et al., 2012, Gupta and Raskar, 2018], as well as hardware and software design [Zaharia et al., 2016, Moritz et al., 2018].

While the predominant paradigm in distributed machine learning is *collaborative learning* of one centralized model, this level of coordination across machines at the training stage is sometimes not feasible. In this work, we instead aim for *collective prediction* at test time without posing any specific requirement at the training stage. Combining the predictions of pre-trained machine learning models has been considered in statistics and machine learning in the context of ensemble methods, including bagging [Breiman, 1996a] and stacking [Wolpert, 1992]. In this work, we explore a new perspective on this aggregation problem and investigate whether insights from the social sciences on how humans reach a consensus can help us design more effective aggregation schemes that fully take advantage of

[*] most of the work conducted while at UC Berkeley.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

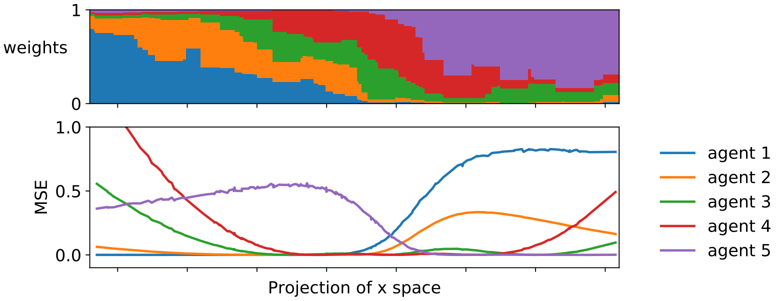


Figure 1: *(Test-time aggregation of heterogeneous models).* The proposed method upweights more accurate models (lower MSE) adaptively in different regions of the input space. Constructed from a regression task described in Section 4.1.

each model's individual strength. At a high level, when a panel of human experts come together to make collective decisions, they exchange information and share expertise through a discourse to then weigh their subjective beliefs accordingly. Experts have a conforming influence on each other which gives rise to a dynamic process of consensus finding that is decentralized and does not rely on any external judgement. We map this paradigm to a machine learning setting where experts correspond to pre-trained machine learning models, and show that it leads to an appealing mechanism for test-time collective prediction.

1.1 Our work

We build on a model for human consensus finding that defines an iterative process of opinion pooling based on *mutual trust scores* among agents [DeGroot, 1974]. Informally speaking, a mutual trust score reflects an agent's willingness to adapt another agent's opinion. Thus, each individual's impact on the final judgment depends on how much it is trusted by the other agents.

Mapping the DeGroot model to our learning scenario, we develop a scheme where each agent uses its own local training data to assess the predictive capabilities of other agents' models and to determine how much they should be trusted. This assessment of mutual trust is designed to be adaptive to the prediction task at hand, by using only a subset of the local data in the area around the current test point. Then, building on these trust scores, a final judgment is reached through an iterative procedure, where agents repeatedly share their updated predictions.

This aggregation scheme is conceptually appealing as it mimics human consensus finding, and it exhibits several practical advantages over existing methods. First, it is decentralized and does not require agents to release their labeled data or model parameters. This can be beneficial in terms of communication costs and attractive from a privacy perspective, as it allows agents to control access to their data and model at any time. Second, it does not require hold-out validation data to construct or train the aggregation function. Thus, there is no need to collect additional labeled data and coordinate shared access to a common database. Third, it does not require any synchronization across agents during the training stage, so agents are free to choose their preferred model architecture, and train their models independently.

A crucial algorithmic feature of our procedure is its *adaptivity* to the test point at hand, which allows it to deal with inherent variations in the predictive quality of models in different regions of the input space. Figure 1 illustrates how our mechanism adaptively upweights models with lower error depending on the test point's location in the input space. In fact, we prove theoretically that our aggregation procedure can recover inverse-MSE weighting in the large-sample limit, which is known to be optimal for variance reduction of unbiased estimators [Bates and Granger, 1969].

To assess our procedure's sensitivity to the predictions of individual agents we propose a decentralized Jackknife algorithm to compute error bars for the consensus predictions with respect to the collection of observed agents. These error bars offer an attractive target of inference since they are agnostic to how individual models have been trained, can be evaluated without additional model evaluations, and allow one to diagnose the vulnerability of the algorithm to malicious agents.

On the empirical side, we demonstrate the efficacy of our mechanism through extensive numerical experiments across different learning scenarios. In particular, we illustrate the mechanism's advantages over model averaging as well as model selection, and demonstrate that it consistently outperforms alternative non-uniform combination schemes that have access to additional validation data across a wide variety of models and datasets.

1.2 Related work

The most common approach for combining expert judgment is to aggregate and average individual expert opinions [Hammitt and Zhang, 2013]. In machine learning this statistical approach of equally-weighted averaging is also known as the "mean-rule" and underlies the popular *bagging* technique [Breiman, 1996a, Buja and Stuetzle, 2006, Efron, 2014]. It corresponds to an optimal aggregation technique for dealing with uncertainty in settings where individual learners are homogeneous and the local datasets are drawn independently from the same underlying distribution. While principled, and powerful due to its simplicity, an unweighted approach cannot account for heterogeneity in the quality, or the expertise, of the base learners.

To address this issue, performance-weighted averaging schemes have been proposed. For determining an appropriate set of weights, these approaches typically rely on calibrating variables [Aspinall, 2010] to detect differences in predictive quality among agents. In machine learning these calibrating variables correspond to labeled cross-validation samples, while in areas such as expert forecasting and risk analysis these variables take the form of a set of agreed-upon seed questions [Cooke, 1991]. These weights can be updated over time using statistical information about the relative predictive performance [Rogova, 2008] or the posterior model probability [Wasserman et al., 2000]. More recently there have been several applications to classification that focus on dynamic selection techniques [Cruz et al., 2018b], where the goal is to select the locally best classifier from a pool of classifiers on the fly for every test point. At their core, these approaches all rely on constructing a good proxy for local accuracy in a region of the feature space around the test point and to use this proxy to weight or rank the candidate models. Various versions of distance and accuracy metrics have been proposed in this context [eg. Woods et al., 1997, Didaci et al., 2005, Cruz et al., 2018a]. However, all these methods rely crucially on shared access to labeled validation data to determine the weights. Similarly, aggregation rules such as *stacking* [Breiman, 1996b, Coscrato et al., 2020] or *meta-models* [Albardan et al., 2020], both of which have proved their value empirically, require a central trusted agency with additional data to train an aggregation function on top of the predictor outputs. A related model-based aggregation method, the mixture-of-experts model [Jacobs et al., 1991], requires joint training of all models as well as data pooling.

To the best of our knowledge there is no approach in the literature that can perform adaptive performance weighting without relying on a central agent to validate the individual models. Notably, our approach differs from general frameworks such as model parameter mixing [Zhang et al., 2013], model fusion [Leontev et al., 2018, Singh and Jaggi, 2020] in that agents are not required to release their models.

Outside machine learning, the question of how to combine opinions from different experts to arrive at a better overall outcome has been studied for decades in risk analysis [Hanea et al., 2018, Clemen and Winkler, 1999] and management science [Morris, 1977]. Simplifying the complex mechanism of opinion dynamics, the celebrated DeGroot model [DeGroot, 1974] offers an appealing abstraction to reason mathematically about human consensus finding. It has been popular in the analysis of convergent beliefs in social networks and it can be regarded as a formalization of the Delphi technique, which has been used to forecast group judgments related to the economy, education, healthcare, and public policy [Dalkey, 1972]. Extensions of the DeGroot model have been proposed to incorporate more sophisticated socio-psychological features that may be involved when individuals interact, such as homophily [Hegselmann and Krause, 2002], competitive interactions [Altafini, 2013] and stubbornness [Friedkin and Bullo, 2017].

2 Preliminaries

We are given K independently trained models and aim to aggregate their predictions at test time. Specifically, we assume there are K agents, each of which holds a local training dataset, $\mathcal{D}_k$, of size

n_k, and a model $f_k : \mathcal{X} \to \mathcal{Y}$. We do not make any prior assumption about the quality or structure of the individual models, other than that they aim to solve the same prediction task.

At test time, let $x' \in \mathcal{X}$ denote the data point that we would like to predict. Each agent $k \in [K]$ has her own prediction for the test point corresponding to $f_k(x')$. Our goal is to design a mechanism $\mathcal{M}$ that combines these predictions into an accurate single prediction, $p^*(x') = \mathcal{M}(f_1(x'), ..., f_K(x'))$. In the following we focus on regression and use the squared loss as a measure of predictive quality.

The requirements we pose on the aggregation mechanism $\mathcal{M}$ are: (i) agents should not be required to share their labeled data or model parameters with other agents, or any external party; (ii) the procedure should not assume access to additional training data for an independent performance assessment or training of a meta-model.

2.1 The DeGroot consensus model

The consensus model proposed by DeGroot [1974] specifies how agents in a group influence each other's opinion on the basis of mutual trust. Formally, we denote the trust of agent i towards the belief of agent j as $\tau_{ij} \in (0, 1]$, with $\sum_{j\in[K]} \tau_{ij} = 1$ for every i. Then, DeGroot defines an iterative procedure whereby each agent i repeatedly updates her belief p_i as

$$p_i^{(t+1)} = \sum_{j=1}^{K} \tau_{ij}\, p_j^{(t)}. \tag{1}$$

The procedure is initialized with initial beliefs $p_i^{(0)}$ and run until $p_i^{(t)} = p_j^{(t)}$ for every pair $i, j \in [K]$, at which point a consensus is deemed to have been reached. We denote the consensus belief by p^*.

The DeGroot model has a family resemblance to the influential PageRank algorithm [Page et al., 1999] that is designed to aggregate information across webpages in a decentralized manner. Similar to the consensus in DeGroot, the page rank corresponds to a steady state of equation (2), where the weights τ_{ij} are computed based on the fraction of outgoing links pointing from page i to page j.

2.2 Computing a consensus

The DeGroot model can equivalently be thought of as specifying a weight for each agent's prediction in a linear *opinion pool* [Stone, 1961]. This suggests another way to compute the consensus prediction. Let us represent the mutual trust scores in the form of a stochastic matrix, $T = \{\tau_{ij}\}_{i,j=1}^{K}$, which defines the state transition probability matrix of a Markov chain with K states. The stationary state w of this Markov chain, satisfying $wT = w$, defines the relative weight of each agent's initial belief in the final consensus

$$p^* = \sum_{j=1}^{K} w_j\, p_j^{(0)}. \tag{2}$$

In our method the Markov chain is guaranteed to be irreducible and aperiodic since all entries of T are positive. Therefore, a unique stationary distribution w exists, and consensus will be reached from any initial state [Chatterjee and Seneta, 1977]. We refer to the weights w as *consensus weights*.

To compute the consensus prediction p^* one can solve the eigenvalue problem numerically for w via power iteration. Letting $T^* = \lim_{m\to\infty} T^m$, the rows of T^* identify, and correspond to the weights w that DeGroot assigns to the individual agents [Durrett, 2019, Theorem 5.6.6]. The consensus (2) is computed as a weighted average over the initial beliefs.

3 DeGroot aggregation for collective test-time prediction

In this section we discuss the details of how we implement the DeGroot model outlined in Section 2.1 for defining a mechanism $\mathcal{M}$ to aggregate predictions across machine learning agents. We then analyze the resulting algorithm theoretically, proving its optimality in the large-sample limit, and we outline how the sensitivity of the procedure with respect to individual agents can be evaluated in a decentralized fashion using a Jackknife method.

Algorithm 1 DeGroot Aggregation

```
Input: $K$ agents with pre-trained models $f_1, \cdots, f_K$ and local data $\mathcal{D}_k, k \in [K]$;
    neighborhood size $N$; test point $x'$.
construct trust scores (based on local accuracy):
for $i = 1, 2, \ldots, K$ do
    Construct local validation dataset $\mathcal{D}_i(x')$ using $N$-nearest neighbors of $x'$ in $\mathcal{D}_i$.
    Compute accuracy $\text{MSE}_{ij}(x')$ of other agents $j = 1 \ldots, K$ on $\mathcal{D}_i(x')$ according to (3).
    Evaluate local trust scores $\{\tau_{ij}\}_{j \in [K]}$ by normalizing $\text{MSE}_{ij}$ as in (4).
end for
find consensus:
Run pooling iterations (2), initialized at $p_j^{(0)} = f_j(x')\ \forall j$, until a consensus $p^*$ is reached.
Return: Collective prediction $p^*(x') = p^*$
```

3.1 Local cross-validation and mutual trust score

A key component of DeGroot's model for consensus finding is the notion of mutual trust between agents, built by information exchange in form of a discourse. Given our machine learning setup with K agents each of which holds a local dataset and a model f_k, we simulate this discourse by allowing query access to predictions from other models. This enables each agent to validate another agent's predictive performance on its own local data. This operationalizes a proxy for trustworthiness.

An important requirement for our approach is that it should account for model heterogeneity, and define trust in light of the test sample x' we want to predict. Therefore, we design an *adaptive* method, where the mutual trust τ_{ij} is reevaluated for every query via a local cross-validation procedure. Namely, agent i evaluates all other agents' predictive accuracy using a subset $\mathcal{D}_i(x') \subseteq \mathcal{D}_i$ of the local data points that are most similar to the test point x' [cf. Woods et al., 1997]. More precisely, agent i evaluates

$$\text{MSE}_{ij}(x') = \frac{1}{|\mathcal{D}_i(x')|} \sum_{(x,y) \in \mathcal{D}_i(x')} (f_j(x) - y)^2, \tag{3}$$

locally for every agent $j \in [K]$, and then performs normalization to obtain the trust scores:

$$\tau_{ij} = \frac{1/\text{MSE}_{ij}}{\sum_{j \in [K]} 1/\text{MSE}_{ij}}. \tag{4}$$

There are various ways one can define the subset $\mathcal{D}_i(x')$; see, e.g., Cruz et al. [2018b] for a discussion of relevant distance metrics in the context of dynamic classifier selection. Since we focus on tabular data in our experiments we use Euclidean distance and assume $\mathcal{D}_i(x')$ has fixed size. Other distance metrics, or alternatively a kernel-based approach, could readily be accommodated within the DeGroot aggregation procedure.

3.2 Algorithm procedure and basic properties

We now describe our overall procedure. We take the trust scores from the previous section, and use them to aggregate the agents' predictions via the DeGroot consensus mechanism. See Algorithm 1 for a full description of the DeGroot aggregation procedure. In words, after each agent evaluates her trust in other agents, she repeatedly pools their updated predictions using the trust scores as relative weights. The consensus to which this procedure converges is returned as the final collective prediction, denoted $p^*(x')$. In the following we discuss some basic properties of the consensus prediction found by DeGroot aggregation.

In the context of combining expert judgement, the seminal work by Lehrer and Wagner [1981] on social choice theory has defined *unanimity* as a general property any such aggregation function should satisfy. Unanimity requires that when all agents have the same subjective opinion, the combined prediction should be no different. Algorithms 1 naturally satisfies this condition.

Proposition 3.1 (Unanimity). *If all agents agree on the prediction, then the consensus prediction from Algorithm 1 agrees with the prediction of each agent:* $p^*(x') = f_i(x')$ *for every* $i \in [K]$.

In addition, our algorithm naturally preserves a global ordering of the models, and the consensus weight of each agent is bounded from below and above by the minimal and maximal trust she receives from any of the other agents.

Proposition 3.2 (Ranking-preserving)**.** *Suppose all agent rankings have the same order: for all $j_1, j_2 \in [K]$, if $\tau_{ij_1} \geqslant \tau_{ij_2}$ for some $i \in [K]$, then $\tau_{i'j_1} \geqslant \tau_{i'j_2}$ for all $i' \in [K]$. Then, Algorithm 1 finds a consensus $p^* = \sum_i w_i f_i(x')$ where for pairs j_1, j_2 such that $\tau_{ij_1} \geqslant \tau_{ij_2}$, we have*

$$w_{j_1} \geqslant w_{j_2}.$$

Proposition 3.3 (Bounded by min and max trust)**.** *The final weight assigned to agent i is between the minimal and maximal trust assigned to it by the set of agents:*

$$\max_j \tau_{ji} \geqslant w_i \geqslant \min_j \tau_{ji}.$$

Finally, Algorithm 1 recovers equally-weighted averaging whenever each agent receives a constant amount of combined total trust from the other agents. This is satisfied if all agents perform equally well on the different validation sets. However, the trust not necessarily needs to be allocated uniformly, there may be multiple T that result in a uniform stationary distribution.

Proposition 3.4 (Averaging as a special case)**.** *If the columns of T sum to 1, then Algorithm 1 returns an equal weighting of the agents: $w_i = 1/K$ for $i = 1, \dots, K$.*

Together these properties serve as a basic sanity check that Algorithm 1 implements a reasonable consensus-finding procedure, from a decision-theoretic as well as algorithmic perspective.

3.3 Optimality in a large-sample limit

In this section we analyze the large-sample behavior of the DeGroot consensus mechanism, showing that it recovers the optimal weighting scheme in the scenario where agents are independent.

For our large-sample analysis, we suppose that the agents' local datasets $\mathcal{D}_k$ are drawn independently from (different) distributions $\mathcal{P}_k$ over $\mathcal{X} \times \mathcal{Y}$. In order for our prediction task to be well-defined, we assume that the conditional distribution of Y given X is the same for all $\mathcal{P}_k$. In other words, the distributions $\mathcal{P}_k$ are all covariate shifts of each other. For simplicity, we assume the distributions $\mathcal{P}_k$ are all continuous and have the same support. We will consider the large-sample regime in which each agent has a growing amount of data, growing at the same rate. That is, if $n = |\mathcal{D}_1| + \cdots + |\mathcal{D}_k|$ is the number of total training points, then $|\mathcal{D}_k|/n \to c_k > 0$, as $n \to \infty$ for each agent $k = 1, \dots, K$. Lastly, we require a basic consistency condition: for any compact set $\mathcal{A} \subset \mathcal{X}$, $f_k \to f_k^*$ uniformly over $x \in \mathcal{A}$ as $n \to \infty$, for some continuous function f_k^*.

Theorem 3.5 (DeGroot converges to inverse-MSE weighting)**.** *Let $x' \in \mathcal{X}$ be some test point. Assume that $\mathcal{P}_k$ is supported in some ball of radius δ_0 centered at x', and that the first four conditional moments of Y given $X = x$ are continuous functions of x in this neighborhood. Next, suppose we run Algorithm 1 choosing $N = \lceil n^c \rceil$ nearest neighbors for some $c \in (0, 1)$. Let*

$$\mathrm{MSE}_k^* = \mathbb{E}\left[(Y - f_k^*(x'))^2\right]$$

denote the asymptotic MSE of model k at the test point x', where the expectation is over a draw of Y from the distribution of $Y \mid X = x'$. Then, the DeGroot consensus mechanism yields weights

$$w_k \to w_k^* = \frac{1/\mathrm{MSE}_k^*}{1/\mathrm{MSE}_1^* + \cdots + 1/\mathrm{MSE}_K^*}.$$

In other words, in the large-sample limit the DeGroot aggregation yields inverse-MSE weighting and thus recovers inverse-variance weighting for unbiased estimators—a canonical weighting scheme from classical statistics [Hartung et al., 2008] that is known to be the optimal way to linearly combine independent, unbiased estimates. This can be extended to our setting to show that DeGroot averaging leads to the optimal aggregation when we have independent[1] agents, as stated next.

[1] While independence is a condition that we can't readily check in practice, it may hold approximately if the agents use different models constructed from independent data. In any case, the primary takeaway is that DeGroot is acting reasonably in this tractable case.

Theorem 3.6 (DeGroot is optimal for independent, unbiased agents)**.** *Assume that the conditional mean and variance of Y given $X = x$ are continuous functions of x. For some $\delta > 0$, consider drawing $\tilde{X}$ uniformly from a ball of radius δ centered at x' and $\tilde{Y}$ from the distribution of $Y \mid X = \tilde{X}$. Under this sampling distribution, suppose the residuals $\tilde{Y} - f_k^*(\tilde{X})$ from the agents' predictions have mean zero and each pair has correlation zero. Then the optimal weights,*

$$\tilde{w} := \underset{w \in \mathbb{R}^k : \|w\|_1 = 1}{\arg\min} \; \mathbb{E}_{(\tilde{X},\tilde{Y})}\left[\left(\tilde{Y} - \sum_{k \in [K]} w_k f_k^*(\tilde{X})\right)^2\right],$$

approach the DeGroot weights: $\tilde{w} \to w^*$ as $\delta \to 0$.

In summary, Theorem 3.5 shows that the DeGroot weights are asymptotically converging to a meaningful (sometimes provably optimal) aggregation of the models – *locally for each test point x'*. This is a stronger notion of adaptivity than that usually considered in the model-averaging literature.

3.4 Error bars for predictions

Finally, we develop a decentralized Jackknife algorithm [Quenouille, 1949, Efron and Tibshirani, 1993] to estimate the standard error of the consensus prediction $p^*(x')$. Our proposed procedure measures the impact of excluding a random agent from the ensemble and returns error bars that measure how stable the consensus prediction is to the observed collection of agents. Formally, let $p^*_{-i}(x')$ denote the consensus reached by the ensemble after removing agent i. Then, the Jackknife estimate of standard error at x' corresponds to

$$\widehat{\mathrm{SE}}(x') = \sqrt{\frac{K-1}{K} \sum_{i \in [K]} \left(p^*_{-i}(x') - \bar{p}^*(x')\right)^2}, \tag{5}$$

where $\bar{p}^*(x') = \frac{1}{K} \sum_{i=1}^{K} p^*_{-i}(x')$ is the average delete-one prediction.

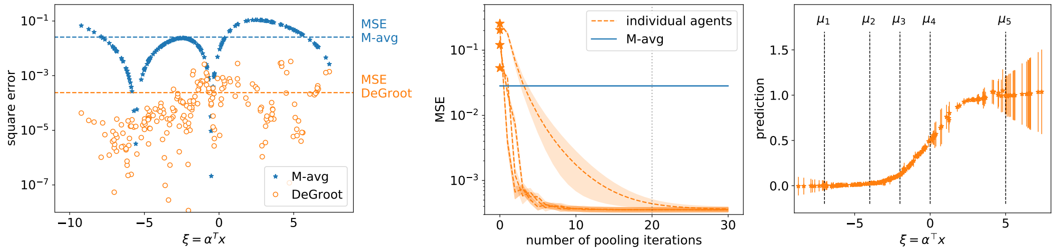
In the collective prediction setting, this is an attractive target of inference because it can be computed in a decentralized manner, is entirely agnostic to the data collection or training mechanism employed by the agents, and requires no additional model evaluations above those already carried out in Algorithm 1. Furthermore, it allows one to diagnose the impact of a single agent on the collective prediction and thus assess the vulnerability of the algorithm to malicious agents. A detailed description of the DeGroot Jackknife procedure can be found in Algorithm 2 in Appendix B. Finally, we want to note that this measure is not specific to how the consensus is reached via the DeGroot method and can readily be applied as a diagnosis tool to other collective prediction algorithms.

3.5 Privacy-preserving properties

It is an appealing property of the DeGroot algorithm that it does not require agents to share their raw training data or model parameters upfront. Information is only exchanged as needed through requesting and answering prediction queries of other agents. This implies that agents can do privacy accounting and remain in control of their data at any time. Sensitive information about the training data can only by leaked in Step 5 of Algorithm 1 by revealing unlabeled data points and prediction queries. For privacy-critical applications this information could further be protected by augmenting DeGroot aggregation with techniques from differential privacy [Dwork and Roth, 2014], or tools from cryptography [Bonawitz et al., 2017]. While we are not providing a rigorous privacy analysis of our method in this work, it could be an interesting extension for future work.

4 Experiments

We investigate the efficacy of the DeGroot aggregation mechanism empirically for various datasets, partitioning schemes and model configurations. We start with a synthetic setup to illustrate the strengths and limitations of our method. Then, we focus on real datasets to see how these gains surface in more practical applications with natural challenges, including data scarcity, heterogeneity of different kinds, and scalability. We compare our Algorithm 1 (DeGroot) to the natural baseline of equally-weighted model averaging (M-avg) and also compare to the performance of individual



(a) individual test points (b) consensus finding (c) DeGroot jackknife

Figure 2: *(Synthetic data experiment)*. Comparison of DeGroot to M-avg. (a) Predictive accuracy on individual test points. (b) Performance for varying number of pooling iterations during DeGroot consensus finding. (c) Confidence intervals for individual test points returned by DeGroot Jackknife.

models composing the ensemble. In Section 4.2 we include an additional comparison with two reference schemes that have access to shared validation data and a central agent that determines an (adaptive) weighting scheme—information DeGroot does not have.

4.1 Synthetic data

First, we investigate a synthetic two-dimensional setting where each agent's features are drawn from a multivariate Gaussian distribution, $x \sim \mathcal{N}(\mu_k, \Sigma_k)$. The true labeling function is given as $y = [1 + \exp(\alpha^\top x)]^{-1}$ and we add white noise of variance σ_Y^2 to the labels in the training data. Unless stated otherwise, we use $K = 5$ agents and let each agent fit a linear model to her local data. See Figure 4 in the Appendix for a visualization of the models learned by the individual agents, and how they accurately approximate the true labeling function in different regions of the input space.[2]

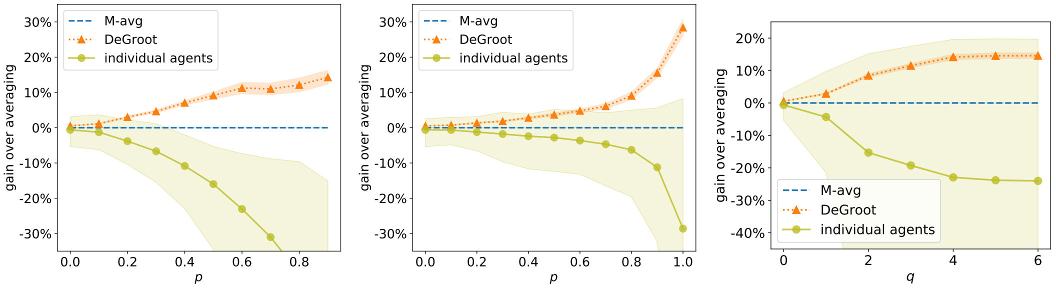
To evaluate the DeGroot aggregation procedure we randomly sample 200 test points from the uniform mixture distribution spanned by the K training data distributions, and compare the prediction of DeGroot to M-avg. We visualize the squared error for the collective prediction on the individual test points in Figure 2a where we use $\xi := \alpha^\top x$ as a summary for the location of the test point on the logistic curve. We observe that DeGroot consistently outperforms M-avg across the entire range of the feature space. Exceptions are three singularities, where the errors of the individual agents happen to cancel out by averaging. Overall DeGroot achieves an MSE of $4.6e^{-4}$ which is an impressive reduction of over $50\times$ compared to M-avg. Not shown in the figure is the performance of the best individual model which achieves an MSE of $5e^{-2}$ and performs worse than the M-avg baseline. Thus, DeGroot outperforms an oracle model selection algorithm and every agent strictly benefits from participating in the ensemble. The power of adaptive weighting such as used in DeGroot is that it can trace out a nonlinear function, given only linear models, whereas any static weighting (or model selection) scheme will always be bound to the best linear fit.

In Figure 2b we show how individual agent's predictions improve with the number of pooling iterations performed in Step 9 of Algorithm 1. The influence of each local model on the consensus prediction of x' and how it changes depending on the location of the test point is illustrated in Figure 1 in the introduction.

An interesting finding that we expand on in Appendix D.1 is that for the given setup the iterative approach of DeGroot is superior to a more naive approach of combining the individual trust scores or local MSE values into a single weight vector. In particular, DeGroot reduces the MSE by $20\times$ compared to using the average trust scores as weights. A theoretical understanding of this remarkable phenomenon remains to be developed.

In Figure 2c we visualize the error bars returned by our DeGroot Jackknife procedure. As expected, the intervals are large in regions where only one a small number of agents possess representative training data, and small in regions where there is more redundancy across agents.

[2] If not stated otherwise we spread the means as $\mu = \{[\text{-}3, \text{-}4], [\text{-}2, \text{-}2], [\text{-}1, \text{-}1], [0, 0], [3, 2]\}$ and let $\Sigma_k = \mathrm{I}$. We use 200 training samples on each agent, for the labeling function we use $\alpha = [1, 1]$, and for the label noise in the training data we set $\sigma_Y = 0.1$. We use $N = 5$ for local cross-validation in DeGroot.



(a) label heterogeneity (b) feature heterogeneity (c) regularizer heterogeneity

Figure 3: *(Data and model heterogeneity for the abalone data).* Relative gain over M-avg. Confidence intervals of DeGroot are over randomness in data partitioning and for the individual agents the shaded area spans the performance from the worst to the best model, the line indicating average performance.

Finally, in Appendix D.1 we conduct further qualitative investigations of the DeGroot algorithm. First, we vary the covariance in the local data feature distribution. We find that the gain of DeGroot over M-avg becomes smaller as the covariance is increased; models become less specialized and there is little for DeGroot to exploit. On the other extreme, if the variance in the data is extremely small, the adaptivity mechanism of DeGroot becomes less effective, as the quality of individual models can no longer be reliably assessed. However, we need to go to extreme values in order to observe these phenomena. In a second experiment we explore DeGroot's sensitivity to choices of N. We find that when N is chosen too large adaptivity can suffer, and for very small values of N performance can degrade if the labels are highly noisy. However, there is a broad spectrum of values for which the algorithm performs well, between 1% and 10% of the local data usually a good choice.

4.2 Real datasets

Turning to real data sets, we first investigate how DeGroot deals with three different sources of heterogeneity. We work with the abalone dataset [Nash et al., 1994] and train a lasso model on each agent[3] with regularization parameter $\lambda_k = \lambda'$ that achieves a sparsity of $\sim$0.8. For our first experiment, shown in Figure 3a, we follow a popular setup from federated learning to control heterogeneity in the data partitioning [see, e.g., Shen et al., 2021]. We partition the training data in the following manner: a fraction $1 - p$ of the training data is partitioned randomly, and a fraction p is first sorted by the outcome variable y and then partitioned sequentially. For the second experiment, shown in Figure 3b, we use a similar partitioning scheme, but sort the data along an important feature dimension, instead of y. This directly leads to heterogeneity in the input domain. Finally, we investigate model heterogeneity arising from different hyper-parameter choices of the regularization parameter λ, given a random partitioning. To control the level of heterogeneity, we start with $\lambda = \lambda'$ on each agent, and let the parameters diverge increasingly, such that $\lambda_k = \lambda' \left[1 + \frac{k - k_0}{K}\right]^q$ with $k_0 = 3$. The results are depicted in Figure 3c. In all three experiments the gain of DeGroot over M-avg is more pronounced with increasing degree of heterogeneity. When there is heterogeneity in the partitioning, DeGroot clearly outperforms the best individual model, by taking advantage of models' local areas of expertise. For the third experiment where the difference is in model quality, DeGroot manages to almost recover the performance of the best single model.

Next, we conduct a broad comparison of DeGroot to additional reference schemes across various datasets and models, reporting the results in Table 1. To the best of our knowledge there is no existing generic adaptive aggregation scheme that does not rely on external validation data to either build the aggregation function or determining the weights w. Thus, we decided to include a comparison with an approach that uses a validation dataset to compute the weighting, where the external validation data has the same size as the local partitions. Our two references are static inverse-MSE weighting (CV-static) where the weights are determined based on the model's average performance on the validation data, and adaptive inverse-MSE weighting (CV-adaptive), where the performance is evaluated only in the region around the current test point x'. We achieve heterogeneity across local datasets by

[3] Similar phenomena can be observed for other datasets and models (see Appendix D.2)

Table 1: *(Benchmark comparison).* Performance for different dataset/model combinations. Datasets have been downloaded from [Fan, 2011]. We use ridge and lasso as a baseline and train a decision tree regressor (DTR) for discrete outcome variables, and a neural net (NN) with two hidden layers for continuous outcomes. A positive relative gain means a reduction in MSE compared to DeGroot.

		MSE DeGroot	gain relative to DeGroot [%]			hyper-parameters
			M-avg	CV-static	CV-adaptive	
Boston	Ridge	25.23	-12.45±2.29	-10.24±1.90	-2.80±1.72	$\lambda = 1e^{-5}$
	Lasso	24.17	-13.70±1.72	-10.35±1.16	-3.46±1.44	$\lambda = 5e^{-3}$
	NN	14.18	-15.18±3.43	-11.01±3.08	-5.81±3.20	layer= $(7, 7)$
E2006	Ridge	0.15	-0.12±0.06	-0.21±0.06	0.13±0.11	$\lambda = 5e^{-2}$
	Lasso	0.097	-0.92±0.15	-0.90±0.13	-0.08± 0.07	$\lambda = 5e^{-5}$
	NN	0.11	-14.58±3.48	-5.69±0.97	-0.16±0.53	layer= $(9, 9)$
Abalone	Ridge	3.18	-3.67 ± 0.55	-3.52±0.47	-0.49±0.46	$\lambda = 5e^{-2}$
	Lasso	4.93	-10.09± 0.83	-10.05±0.80	-0.39±0.75	$\lambda = 5e^{-2}$
	DTR	4.85	-2.53± 0.69	-2.55±0.70	-0.79±0.88	max_depth = 4
cpusmall	Ridge	59.00	-96.23±23.46	-89.81±15.58	-0.78±0.63	$\lambda = 1e^{-5}$
	Lasso	53.82	-76.08±14.45	-73.81±10.51	-2.04±1.82	$\lambda = 1e^{-3}$
	DTR	11.16	-4.05±1.88	-3.90±1.81	1.65±2.20	max_depth=7
YearPrediction	Ridge	95.02	-0.63±0.24	-0.94±0.11	0.21±0.12	$\lambda = 1$
	Lasso	91.21	-1.73±0.47	-1.98±0.35	-0.22±0.29	$\lambda = 1$
	DTR	62.11	-3.13±1.83	-2.26±1.02	-0.54±0.53	max_depth = 4

partially sorting a fraction $p = 0.5$ of the labels, and we choose N to be 1% of the data partition for all schemes (with a had lower bound at 2). Each scheme is evaluated for the same set of local models and confidence intervals are over the randomness of the data splitting into partitions, test and validation data across evaluation runs. The results demonstrate that DeGroot can effectively take advantage of the proprietary data to find a good aggregate prediction, significantly outperforming M-avg and CV-static, and achieving better or comparable performance to CV-adaptive that works with additional data.

To complement our study, we verify in Appendix D.2 that our aggregation mechanism scales robustly with the number of agents in the ensemble and the simultaneously decreasing partitioning size: DeGroot consistently outperforms model averaging on two different datasets on the full range from 2 up to 128 agents. Finally, for the reader who is interested in applying the DeGroot procedure to classification tasks we have outlined a classification example in Appendix D.3.

5 Discussion

We have shown that insights from the literature on humans consensus via discourse suggest an effective aggregation scheme for machine learning. In settings where there is no additional validation data available for assessing individual models' quality, our approach offers an appealing mechanism to take advantage of individual agents' proprietary data to arrive at an accurate collective prediction. Our mechanism relies on information exchange between agents formalized through a query model. This emulation of a decentralized discussion phase that allows agents to influence each other's opinion offers an appealing new perspective on decentralized learning. We are convinced that building on this idea, developing richer models of discourse, and refining mutual trust through experience could offer a promising avenue for future research in federated learning.

Acknowledgments and funding statement

The authors would like to thank Jacob Steinhardt and Alex Wei for feedback on this manuscript. In terms of funding; CM was supported by the Swiss National Science Foundation Early Postdoc Mobility fellowship program, WG was supported by a Google PhD fellowship, and MJ acknowledges support from the Vannevar Bush Faculty fellowship program under grant number N00014-21-1-2941.

References

Mahmoud Albardan, J. Klein, and O. Colot. Spocc: Scalable possibilistic classifier combination - toward robust aggregation of classifiers. *Expert Syst. Appl.*, 150:113332, 2020.

Claudio Altafini. Consensus problems on networks with antagonistic interactions. *IEEE Transactions on Automatic Control*, 58(4):935–946, 2013.

W. Aspinall. A route to more tractable expert advice. *Nature*, 463(7279):294–295, 2010.

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

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, 2017. ISBN 9781450349468. doi: 10.1145/3133956.3133982.

Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996a.

Leo Breiman. Stacked regressions. *Mach. Learn.*, 24(1):49–64, July 1996b.

Andreas Buja and Werner Stuetzle. Observations on bagging. *Statistica Sinica*, 16(2):323–351, 2006.

S. Chatterjee and E. Seneta. Towards consensus: Some convergence theorems on repeated averaging. *Journal of Applied Probability*, 14(1):89–97, 1977.

Robert T. Clemen and Robert L. Winkler. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19(2):187–203, 1999.

Roger M. Cooke. *Experts in Uncertainty: Opinion and Subjective Probability in Science*. Oxford University Press, 1991.

Victor Coscrato, Marco Henrique de Almeida Inacio, and Rafael Izbicki. The NN-Stacking: Feature weighted linear stacking through neural networks. *Neurocomputing*, 399:141–152, 2020.

Rafael M. Cruz, Robert Sabourin, and George D. Cavalcanti. Prototype selection for dynamic classifier and ensemble selection. *Neural Comput. Appl.*, 29(2):447–457, 2018a.

Rafael M. Cruz, Robert Sabourin, and George D.C. Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018b.

N. Dalkey. Studies in the quality of life; DELPHI and decision-making. 1972.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Morris H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345): 118–121, 1974.

Luca Didaci, Giorgio Giacinto, Fabio Roli, and Gian Luca Marcialis. A study on the performances of dynamic classifier selection based on local accuracy estimation. *Pattern Recognition*, 38(11): 2188–2191, 2005.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. 9(3–4):211–407, 2014. ISSN 1551-305X.

Bradley Efron. Estimation and accuracy after model selection. *Journal of the American Statistical Association*, 109(507):991–1007, 2014.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, 1993.

Rong-En Fan. Libsvm data: Classification, regression, and multi-label, 2011. Datasets available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Noah E. Friedkin and Francesco Bullo. How truth wins in opinion dynamics along issue sequences. *Proceedings of the National Academy of Sciences*, 114(43):11380–11385, 2017.

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

James K. Hammitt and Yifan Zhang. Combining experts' judgments: Comparison of algorithmic methods using synthetic data. *Risk Analysis*, 33(1):109–120, 2013.

Anca Hanea, Marissa McBride, Mark Burgman, and Bonnie Wintle. The value of performance weights and discussion in aggregated expert judgments. *Risk Analysis*, 38, 03 2018.

J. Hartung, G. Knapp, and B.K. Sinha. *Statistical Meta-Analysis with Applications*. Wiley Series in Probability and Statistics. Wiley, 2008. ISBN 9780470290897.

Rainer Hegselmann and Ulrich Krause. Opinion dynamics and bounded confidence: Models, analysis and simulation. *Journal of Artificial Societies and Social Simulation*, 5:1–24, 2002.

Olle Häggström. *Finite Markov Chains and Algorithmic Applications*. London Mathematical Society Student Texts. Cambridge University Press, 2002.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. Predicting risk from financial reports with regression. NAACL '09, page 272–280. Association for Computational Linguistics, 2009.

Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *Arxiv*, arxiv:1511.03575, 2015.

Keith Lehrer and Carl Wagner. *Rational Consensus in Science and Society*. Boston: D. Reidel, 1981.

Mikhail Iu. Leontev, Viktoriia Islenteva, and Sergey V. Sukhov. Non-iterative knowledge fusion in deep convolutional neural networks. *Arxiv*, arxiv:1809.09399, 2018.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282. PMLR, 2017.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018. Software available at https://ray.io/.

Peter A. Morris. Combining expert judgments: A bayesian approach. *Management Science*, 23(7): 679–693, 1977.

Warwick Nash, T.L. Sellers, S.R. Talbot, A.J. Cawthorn, and W.B. Ford. 7he population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait. *Sea Fisheries Division, Technical Report No*, 48, 01 1994.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

M. H. Quenouille. Approximate tests of correlation in time-series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 11(1):68–84, 1949.

Galina Rogova. *Combining the Results of Several Neural Network Classifiers*, pages 683–692. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-44792-4.

Zebang Shen, Hamed Hassani, Satyen Kale, and Amin Karbasi. Federated functional gradient boosting. *Arxiv*, arxiv:2103.06972, 2021.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22045–22055, 2020.

M. Stone. The Opinion Pool. *The Annals of Mathematical Statistics*, 32(4):1339 – 1342, 1961.

Larry Wasserman et al. Bayesian model selection and model averaging. *Journal of mathematical psychology*, 44(1):92–107, 2000.

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

K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM*, 59(11):56–65, 2016. Software available at https://spark.apache.org/.

Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 14(68):3321–3363, 2013.

A Proofs

For some of our proofs we will use a Markov chain interpretation of the DeGroot mechanism, where the matrix T defines the transition probabilities of the Markov chain. By definition of the trust scores, the rows of T sum to one. Moreover, the entries are all positive, so this is an aperiodic, irreducible Markov chain and hence has a unique stationary distribution, given by the solution w to the linear system $w = Tw$ [e.g., Häggström, 2002]. Furthermore, T has a unique largest left-eigenvector with eigenvalue 1. It will be convenient for us to consider a Markov chain $Z_0, Z_1, \dots$ with transition matrix T, and Z_0 drawn from the stationary distribution w.

Proof of Proposition 3.1. We prove the claim by induction on the number of consensus steps t. First, we have that $p_i^{(0)} = f_1(x')$ for all $i \in [K]$ by assumption. Suppose $p_i^{(t)} = f_1(x')$ for all $i \in [K]$. Then,

$$p_i^{(t+1)} = \sum_{j=1}^{K} \tau_{i,j} p_j^{(t)} = f_1(x')$$

since the weights sum to 1. ■

Proof of Proposition 3.2. Using our Markov chain notation above, it suffices to show that

$$P(Z_1 = j_1) \geqslant P(Z_1 = j_2).$$

It is easy to see that this is the case since

$$P(Z_1 = j_1 \mid Z_0 = i) \geqslant P(Z_1 = j_2 \mid Z_0 = i)$$

for all $i \in [K]$. ■

Proof of Proposition 3.3. Using our Markov chain notation above, note that

$$\max_j \{P(Z_1 = i \mid Z_0 = j)\} \geqslant P(Z_1 = i) \geqslant \max_j \{P(Z_1 = i \mid Z_0 = j)\}.$$

The result follows. ■

Proof of Proposition 3.4. Note that in this case, $1^\top T = 1$ (where 1 is the K-vector of all ones), so $w = 1/n$ is the stationary state. ■

Proof of Theorem 3.5. Choose any $i \in [K]$. We will show that $\tau_{ij} \to 1/\mathrm{MSE}_j^*$ in probability as $n \to \infty$.

First, let $x_1, \dots, x_N \in \mathcal{D}_i$ be the N nearest neighbors to the test point x' in agent i's data. We claim that

$$\max_{m \in \{1,\dots,N\}} \|x_m - x'\| \to 0 \tag{6}$$

in probability as $n \to \infty$. To see this, for $\delta_0 > \delta > 0$, note that $P(\|x_m - x'\| < \delta) = \varepsilon$ for some $\varepsilon > 0$, since the points in $\mathcal{D}_i$ are an i.i.d. sample from a distribution supported in this ball. But then,

$$P\left(\max_{m \in \{1,\dots,N\}} \|x_m - x'\| \geqslant \delta\right) = P(B < N)$$

where B is distributed as a binomial with $|\mathcal{D}_i|$ draws and success probability ε. But

$$P\left(B < N\right) = P\left(\frac{B - \varepsilon|\mathcal{D}_i|}{\sqrt{|\mathcal{D}_i|\varepsilon(1-\varepsilon)}} < \frac{N - \varepsilon|\mathcal{D}_i|}{\sqrt{|\mathcal{D}_i|\varepsilon(1-\varepsilon)}}\right)$$

By assumption the term $N - \varepsilon|D_i|/\sqrt{|\mathcal{D}_i|}\varepsilon(1-\varepsilon) \to -\infty$ as $n \to \infty$ since $N/|\mathcal{D}_i| \to 0$. Then, by the central limit theorem we conclude that

$$P\left(B < N\right) \to 0$$

as $n \to \infty$, and hence (6) holds.

Next, let $\mu(x) = \mathbb{E}[Y \mid X = x]$ and $\sigma(x) = \text{Var}(Y \mid X = x)$ be the true mean and variance functions, respectively. For convenience, let $B_\delta(x')$ be the ball of radius δ centered at x'. We have

$$\inf_{x \in B_\delta(x')} \{\mu(x) - f_j(x))^2 + \sigma(x)^2\} \leqslant \mathbb{E}\left[1/\tau_{i,j} \middle| \max_{m \in \{1,\ldots,N\}} \|x_m - x'\| < \delta\right] \leqslant \sup_{x \in B_\delta(x')} \{\mu(x) - f_j(x))^2 + \sigma(x)^2\}.$$

Note that the lower and upper bound both converge to MSE_j^* as $\delta \to 0$, by the continuity of $\mu(x), f_j(x)$ and $\sigma(x)$. Similarly, the conditional variance of $1/\tau_{i,j}$ is bounded by C/N for some constant C. Thus, by Chebyshev's inequality, there exist sequences $c_1(n) \to 0$ and $c_2(n) \to 0$ such that

$$P\left(|\tau_{ij} - \text{MSE}_j^*| \geqslant c_1(n) \mid \max_{m \in \{1,\ldots,N\}} \|x_m - x'\| < \delta\right) \leqslant c_2(n).$$

Since the event $\{\max_{m \in \{1,\ldots,N\}} \|x_m - x'\| < \delta\}$ has probability tending to 1 as n grows, this implies that $1/\tau_{i,j} \to \text{MSE}_j^*$ in probability, as desired. ■

Proof of Theorem 3.6. For this distribution, we wish to find weights $\tilde{w}$ solving the following:

$$\tilde{w} := \mathop{\arg\min}_{w \in \mathbb{R}^k, 1^\top w = 1} \mathbb{E}_{(\tilde{X},\tilde{Y})}\left[\left(\tilde{Y} - \sum_{k=1}^K w_k f_k^*(\tilde{X})\right)^2\right].$$

Expanding the right side, we have

$$\begin{aligned}
\mathbb{E}_{(\tilde{X},\tilde{Y})}\left[\left(\tilde{Y} - \sum_{k=1}^K w_k f_k^*(\tilde{X})\right)^2\right] &= \mathbb{E}_{(\tilde{X},\tilde{Y})}\left[\left(\sum_{k=1}^K w_k(\tilde{Y} - f_k^*(\tilde{X}))\right)^2\right] \\
&= \text{Var}\left(\sum_{k=1}^K w_k(\tilde{Y} - f_k^*(\tilde{X}))\right) \\
&= \sum_{k=1}^K \text{Var}\left(w_k(\tilde{Y} - f_k^*(\tilde{X}))\right) \\
&\quad + 2 \sum_{1 \leqslant k_1 < k_2 \leqslant K} \text{Cov}\left(w_{k_1}(\tilde{Y} - f_{k_1}^*(\tilde{X}), w_{k_2}(\tilde{Y} - f_{k_2}^*(\tilde{X}))\right) \\
&= \sum_{k=1}^K w_k^2 \text{Var}\left(\tilde{Y} - f_k^*(\tilde{X})\right)
\end{aligned}$$

With this expression, one can easily verify that the optimal w is then

$$\tilde{w}_k = 1/\text{Var}\left(\tilde{Y} - f_k^*(\tilde{X})\right).$$

Lastly, we claim that

$$\text{Var}\left(\tilde{Y} - f_k^*(\tilde{X})\right) \to \text{MSE}_k^*$$

as $\delta \to 0$. This follows from the decomposition

$$\text{Var}\left(\tilde{Y} - f_k^*(\tilde{X})\right) = \mathbb{E}\left[\text{Var}\left(\tilde{Y} - f_k^*(\tilde{X}) \mid \tilde{X}\right)\right] + \text{Var}\left(\mathbb{E}[\tilde{Y} - f_k^*(\tilde{X}) \mid \tilde{X}]\right).$$

By assumption, the conditional mean and variance are continuous functions of x, so the first term in the sum converges to MSE_k^* as $\delta \to 0$. Likewise, the second term in the sum converges to 0. This completes the proof.

■

B Estimating the Standard Error

In this section, we develop a standard errors estimator for the collective prediction $p^*(x')$ resulting from Algorithm 1. Our proposal is a special case of jackknife standard error estimates, where we consider the agents to be the independent samples from a super-population of agents (this probabilistic interpretation is useful to precisely discuss standard errors but is not needed elsewhere in our work). Our error bars will thus measure how stable the consensus prediction is to the observed collection of agents. In the collective prediction setting, this is an attractive target of inference because it can be estimated without requiring any assumptions about how the agents gather data, their models and training procedure. As a result, our standard error estimate is entirely agnostic to the behavior of the agents, is decentralized, and requires minimal communication—it requires no additional model queries above what was already done in Algorithm 1.

Algorithm 2 DeGroot Jackknife

```
1: Input: Pre-trained $K$ agents $f_1, \cdots, f_K$ with local training dataset $\mathcal{D}_k, k \in [K]$;
   neighborhood size $m$; test point $x'$.
2: Construct trust matrix $T$ as in Algorithm 1.
3: for $i = 1, 2, \dots, K$ do
4:     Create submatrix $T^{(i)}$
5:     Find $v$ such that $vT^{(i)} = v$ by power iteration.
6:     Form collective prediction $p^*_{-i}(x') = \sum_{j \neq i} v_j f_j(x')$
7: end for
8: Return: Jackknife estimate of standard error at $x'$:
       $$\widehat{\mathrm{SE}}(x') = \sqrt{\frac{K-1}{K} \sum_{i=1}^{K} \left(p^*_{-i}(x') - \bar{p}^*(x')\right)^2},$$
   where $\bar{p}^*(x') = \frac{1}{K} \sum_{i=1}^{K} p^*_{-i}(x')$ is the average delete-one prediction at $x'$.
```

Turning to the details, recall that T is the matrix of trust scores, and let $T^{(i)}$ be the principal submatrix of T formed by deleting row i and column i, renormalized so that each row sums to one. That is, $T^{(i)}$ is the trust matrix if agent i is removed from our collection of agents. The idea is that we run DeGroot aggregation with agent i deleted; i.e., we find the collective prediction $p^*_{-i}(x')$ of the remaining $K-1$ agents. Then, by looking at the variability of these predictions with different agents removed, we can quantify the stability of the procedure. Specifically, we take the sample standard deviation of these quantities, scaled by $\sqrt{(K-1)^2/K}$: the scaling prescribed by the theory of the jackknife estimator [Efron and Tibshirani, 1993]. We state this procedure formally in Algorithm 2.

Importantly, this calculation does not require any additional model evaluations above what is required to perform the consensus prediction; this algorithm only requires constructing the matrix T and knowing the predictions $f_j(x')$, which were already needed in Algorithm 1. Thus, estimating the standard error has no additional information-sharing requirement; it requires only a modest extra amount of computation.

See Figure 8 for an empirical demonstration of these error bars on our synthetic example from Section 4.1.

C Details on algorithm implementation

In this section we discuss some practical considerations when implementing Algorithm 1.

Step 4. For our experiments we have chosen the Euclidean distance metric, and a fixed number N of datapoints to determine $\mathcal{D}_i(x')$. To find the N nearest neighbors we have used the `sklearn` unsupervised `NearestNeighbors` classifier from `sklearn.neighbors`. Note that this step could be adjusted to use other distance metrics specific to the modality of the data. In particular, for higher dimensional data such as images a careful choice of distance metric will become crucial.

Step 9. In this step the agents iteratively refine their predictions by pooling other agents predictions to reach a consensus, as described in Section 2.1. These pooling iterations can be executed in a fully decentralized manner, or the consensus weights can be evaluated via the power method at one of the nodes that gets access to all the trust scores in order to construct the matrix T from $\{\tau_{i,j}\}_{i,j\in[K]}$. The best implementation will depend on the requirements of your application. For our experiments we have chosen the centralized evaluation for convenience. Around $t_p = 30$ power iterations are usually far sufficient to reach a consensus. Given that the number of agents is usually small (< 1000) this operation comes with little computational cost and we recommend to pick the number of iterations t_p with sufficient margin to take advantage of the full potential of DeGroot aggregation. Alternatively, one might also implement a stopping criteria in the form of $|p_j^{(t)} - p_j^{(t-1)}| \leqslant \text{tol}$ that is checked independently by each agent $j \in [K]$ after every round.

Step 10. For every test point the algorithm returns the consensus prediction $p^*(x')$. If Step 9 has fully converged we have $p_j = p^*$ for all j and we can return any of the individual agents predictions. However, to get a robust procedure, even if the individual beliefs have not fully converged, we return $p^* = \frac{1}{K}\sum_j p_j$ instead.

D Additional Experimental Details and Evaluations

The code is written in Python and for model training, as well as the nearest neighbor computation we use the built-in functionalities of Sklearn [Pedregosa et al., 2011].

D.1 Evaluation on Synthetic Data

For the synthetic setup the true label is determined by a logistic function, as described in Section 4.1, where we choose $\alpha = [1, 1]$. The features of agent k are distributed according to a multi-variate Gaussian distribution with mean μ_k and each model then fits a linear function to it's training data. We use the following default experimental configuration: we choose the means as $\mu_1 =$ [-3, -4], $\mu_2 =$ [-2, -2], $\mu_3 =$ [-1, -1], $\mu_4 = [0, 0]$ $\mu_5 = [3, 2]$, the covariance of the local data as $\Sigma_k = \Sigma = \mathrm{I}$ and the variance in the label noise as $\sigma_Y = 0.1$. The resulting local linear fits are illustrated in Figure 4 where we see that individual models approximate the true labeling function in different regions of the input space. The accuracy as a function of ξ is illustrated in Figure 1 in the introduction.

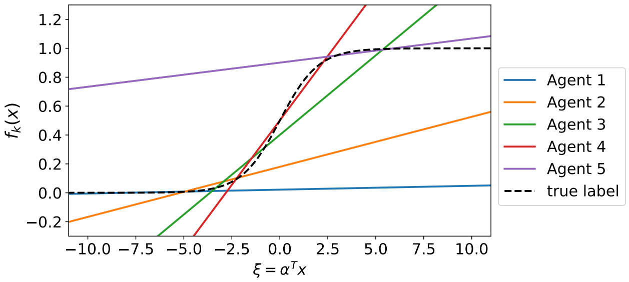
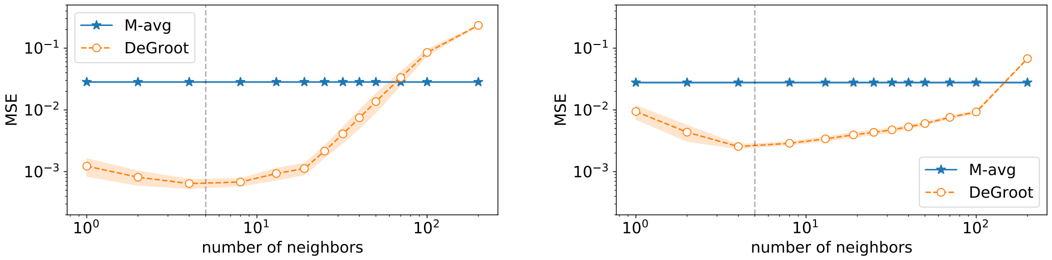


Figure 4: *(agents' local models).* Local linear models fit by individual agents for the default experimental configuration.

Additional Evaluations:

A) Sensitivity of DeGroot to hyperparameter N. In Figure 5 we demonstrate how the performance of Algorithm 1 in this synthetic setting changes with the number of neighbors N used for local validation for two different levels of label noise. Overall we find that the performance of DeGroot is not very sensitive to the choice of N, whereas the optimal regime of values for N increases with the noise in the data. In general, we recommend choosing N to corresponds to approximately $1 - 10\%$ of the available data, with a hard lower-bound on N, bounding it away from 1. For later investigations on real data we pick N to be 1% of the partition size.

B) Varying overlap in local data partitions. In Figure 6 we investigate how the performance of DeGroot changes in comparison to M-avg for varying overlap in the local data partitions. Therefore



(a) $\sigma_Y = 0.1$

(b) $\sigma_Y = 0.3$

Figure 5: *(Performance vs. number of neighbors)*. Synthetic data experiment comparing DeGroot to model averaging (M-avg) for two different noise levels (σ_Y) in the training data labels. Gray dashed line indicate the default value in our experiments.

we vary the covariance $\Sigma_k = \mathrm{I}\sigma^2$ in the feature distribution of the individual agents, while keeping the means μ_k fixed. We find that for larger values of σ^2 there is less to gain for DeGroot since models are less specialized, although the gains remains significant up to $\sigma^2 = 10$. We further see that if the variance in the data is very small and the local datasets are not sufficiently covering the feature space, the local cross-validation procedure and the adaptivity of the DeGroot procedure start to suffer.

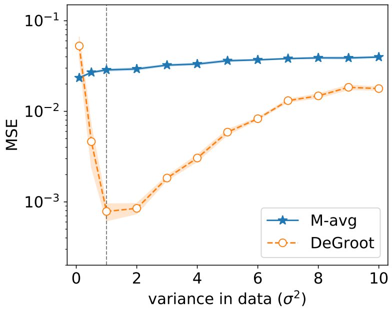


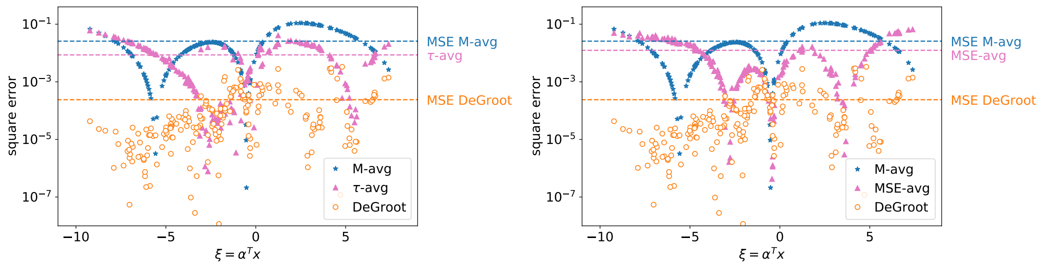
Figure 6: *(Varying overlap in data partitions)*. Comparing DeGroot to classical model averaging (M-avg) for varying covariance $\Sigma_k = \mathrm{I}\sigma^2$ in the feature distribution, keeping the means μ_k fixed. Gray dashed line indicate the default value in our experiments.

C) Alternative pooling operations. In Figure 7 we aim to provide additional insights into the effect of DeGroot's iterative procedure to find consensus and aggregate the individual predictions. In Figure 2b we have shown how the accuracy of individual agents improves with the number of pooling iterations in the DeGroot procedure. After around 20 iterations the agents have reached consensus. In Figure 7 we compare the accuracy of the predictions obtained through DeGroot to alternative ways of aggregating the local MSE values and trust scores into a single weight vector. These baselines are constructed for diagnostic purpose to demonstrate the value of using power iterations instead of an alternative procedure.

The first method, denoted τ-avg takes the mean of the trust scores τ_{ij} across all agents i to obtain the weights w_j of agent j on the final prediction. As shown in Figure 7a τ-avg performs significantly worse than DeGroot on our synthetic example, and achieves an overall MSE of $9e^{-3}$ which is $20\times$ larger than DeGroot. This shows that the DeGroot agents find a better collective prediction through iterative pooling than they would by averaging their beliefs after only a single round of pooling.

The second method first averages the local MSE values of each agent across the different datasets as $\bar{\mathrm{MSE}}_j = \sum_i \mathrm{MSE}_{i,j}$ and then obtains the weights through inverse-MSE weighting as $w_j = 1/\bar{\mathrm{MSE}}_j$. Similar to τ-avg, this alternative procedure performs significantly worse than DeGroot, and acheives an MSE of $1e^{-2}$.

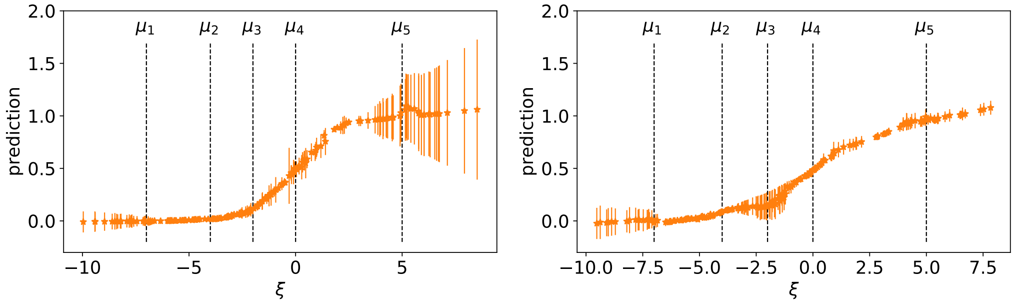
D) DeGroot Jackknife. Finally, in Figure 8, we evaluate the error bars for our predictions using the Jackknife procedure proposed in Algorithm 2 for the test points evaluated in Figure 2a. Here, we



(a) τ-avg aggregation

(b) MSE-avg aggregation

Figure 7: *(Alternative aggregation schemes).* (a) Comparison of DeGroot to τ-avg that average the trust scores across all agents to obtain the weights of the individual agents. (b) Comparison of DeGroot to MSE-avg that performs inverse-MSE weighting based on the aggregated MSE values cross the local datasets. Experiments serve for diagnosis purpose and are conducted on the synthetic setup outlined in Section 4.1.



(a) $\sigma^2 = 1$

(b) $\sigma^2 = 5$

Figure 8: *(DeGroot Jackknife).* Evaluations of error bars using the decentralized Jackknife procedure proposed in Algorithm 2. The test points correspond to the same 200 test points visualized in Figure 2a. For (a) we use the standard configuration, and for (b) we increase the covariance of the local training datasets by a factor of 5, i.e. $\Sigma_k = 5$ I.

report the predictions as a function of ξ, as well as the standard error (5). We find that the standard errors are relatively small in the center, but increase at the edges of the space. This makes sense: on the edges of the space there is only one agent with good prediction accuracy, so the final consensus prediction is not stable to the deletion of any one agent. To validate this explanation further, in the right panel we increase the spread of the training points for each agent, so that the agents' training data overlap more. As expected, the standard errors are now much smaller—because there are multiple agents with training data in most regions, the procedure is more stable.

D.2 Evaluation on Real Data

Datasets. The specifications of the datasets used for our experiments in Section 4.2 can be found in Table 2. The datasets have been downloaded from libsvm [Fan, 2011] and are used without any additional preprocessing, apart from the boston dataset that is normalized for neural network training. The boston housing price data [Dua and Graff, 2017] and the E2006 datasets [Kogan et al., 2009] have continuous outcome variables. Abalone [Dua and Graff, 2017] has integer values outcomes from $1 - 29$, YearPrediction [Dua and Graff, 2017] has integer valued year numbers from $1922 - 2011$, and the labels of cpusmall correspond to integers from $0 - 99$.

Details on Benchmark Study in Table 1:

Baselines. We compare DeGroot to three baselines: M-avg, CV-static, CV-dynamic. The three methods describe different approaches to determine the weight w_j of each model on the final

Table 2: Regression datasets used for evaluation, downloaded from [Fan, 2011].

	# samples	# featues
Boston	506	13
E2006	16087	3308
Abalone	4077	8
cpusmall	8192	12
YearPrediction	463715	90

prediction $p(x') = \sum_j w_j f_j(x')$. (M-avg) corresponds to the most natural baseline of equally weighted averaging with $w_j = \frac{1}{K}$ for all $j \in [K]$. This baseline that is optimal if models are unbiased and have equal variance. (CV-static) and (CV-adaptive) are two methods that have access to additional hold-out data to evaluate the predictive accuracy MSE$_j$ of each model $j \in [K]$ and determine the weights w_j. As the name says (CV-static) uses static weights, and (CV-adaptive) uses an adaptive weighting scheme. Similar to DeGroot both methods use weights that are inversely proportional to the MSE of the models:

$$w_j = \frac{1/\text{MSE}_j}{\sum_j 1/\text{MSE}_j}$$

In CV-static MSE$_j$ is evaluated on the hold-out data, and in CV-adaptive MSE$_j$ is evaluated on a subset of the hold-out data, composing of the N closest points to x'. We use the same distance measure, and number of neighbors N as we use to perform cross-validation on the individual agents in DeGroot. Our goal is to minimize potential confounding for a most meaningful comparative investigation.

Evaluation. In all experiments the data is first randomly partitioned into train,test and validation data. The validation data is completely ignored by DeGroot and M-avg and only used for CV-static and CV-adaptive. The size of the hold-out data is equal to the size of one data partition. We use $n_{test} = \max(0.15n, 500)$ test samples. For every random split all methods are evaluated on the same set of local models. We tune the hyparparameter of each model using a rough hyperparameter search upfront, but our main goal is not to get optimal accuracy with each individual model, but rather to investigate how the different aggregation procedures can deal with (resonable) given models. Hyperparameters are reported in Table 1.

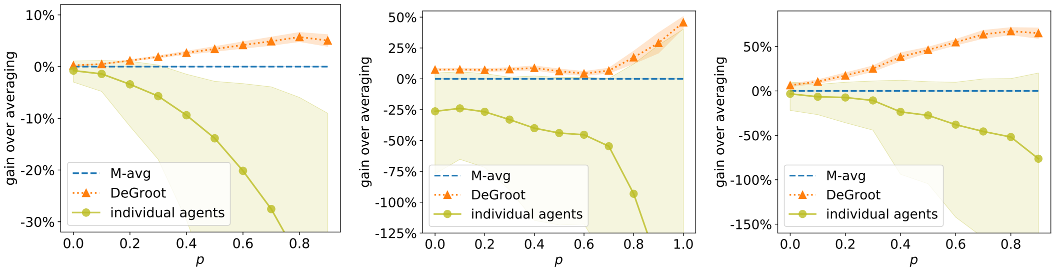
Models. For training of the models we use the built in model classes for scikit-learn [Pedregosa et al., 2011]. Ridge and Lasso are imported from `sklearn.linear_model` and the hyperparameter λ corresponds to the regularizer strength. For the DTR we use the `DecisionTreeRegressor` from `sklearn.tree` where the hyperparameter max_depth corresponds to the maximum depth of the regression tree. For the neural network (NN) we use the `MLPRegressor` class from `sklearn.neural_network`, it uses relu activation by default, and we choose two hidden layers with 8 neurons each, and α corresponds to the regularization strength. All other parameters are set to their default values. Whenever a NN model has not converged within the given number of iterations (500 for E2006 and 200 for boston) the model is nevertheless included in the ensemble since the purpose of this study is the evaluate how the different schemes deal with given models that might be heterogeneous in terms of quality.

Confidence Intervals. The reported confidence intervals are over the randomness in the data splitting and partitioning, they show one standard deviation from the mean.

Additional Evaluations:

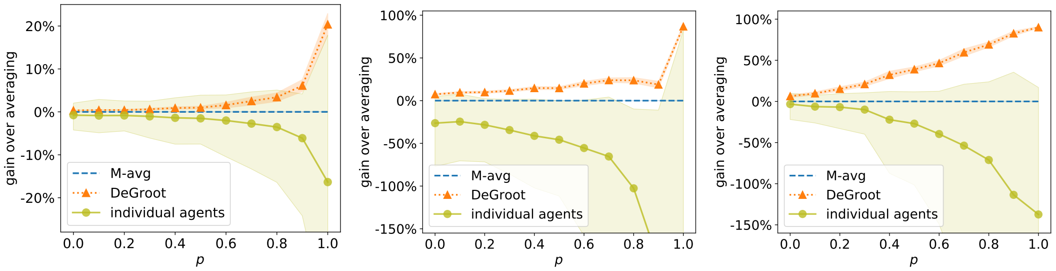
Heterogeneity. We perform additional evaluations with the different approaches of achieving heterogeneity described in Section 4.2. In Figure 9 and Figure 10 we show results for ridge regression on the abalone, boston and cpusmall dataset. The gain of DeGroot over M-avg consistently grows with the degree of heterogeneity.

Scalability. In Figure 11 we investigate the scaling of the DeGroot method on two different datasets. We show results for 1 up to 128 agents. The training data (after test set hold-out) is partitioned across the agents by partially sorting the label with $p = 0.5$. This means with increasing number of agents K the partitions get smaller. We choose the number of neighbors in DeGroot as $N =$



(a) Abalone ($\lambda' = 1$) (b) Boston ($\lambda' = 10^{-5}$) (c) Cpu-small ($\lambda' = 1e^{-5}$)

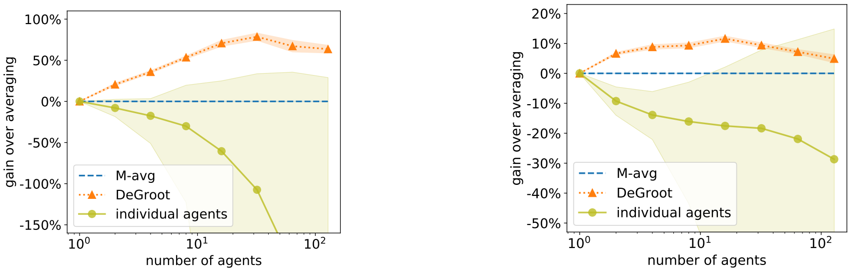
Figure 9: *(Label heterogeneity)*. Relative gain of DeGroot over M-avg. Same setting as Figure 3a for ridge regression on different datasets. We depict the performance of the individual models in the ensemble using the shaded green area, the upper edge corresponds to the best and the lower edge to the worst performing model, the green line indicating the average performance across the models.



(a) Abalone ($\lambda' = 1$) (b) Boston ($\lambda' = 10^{-5}$) (c) Cpu-small ($\lambda' = 1e^{-5}$)

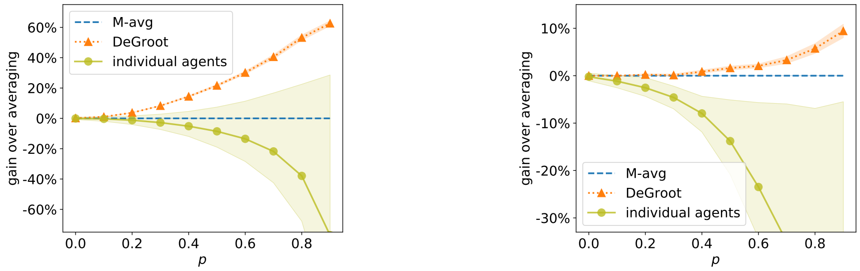
Figure 10: *(Feature heterogeneity)*. Relative gain of DeGroot over M-avg. Same setting as Figure 3b for ridge regression on different datasets. We partially sort the training data along feature 8 (Abalone) feature 9 (Boston) and feature 11 (cpusmall). The performance of the individual models in the ensemble is depicted using the shaded green area, the upper edge corresponds to the best and the lower edge to the worst model, the green line indicating the average performance across the models.

$\max(2, 0.01 * n_{\text{local}})$, where n_{local} denotes the size of a local partition. We find that DeGroot scales robustly with the number of agents in the ensemble and outperforms averaging by a large margin. Remarkably, the adaptive weighting of DeGroot is still effective for 128 agents on the abalone datasets, where each agent only has access to 30 samples.



(a) Cpusmall – ridge ($\lambda' = 1e^{-5}$) (b) Abalone – lasso ($\lambda' = 1e^{-5}$)

Figure 11: *(Scaling behavior of DeGroot)*. Relative gain of DeGroot aggregation over model averaging (M-avg) for different number of agents. Experiments performed for two different configurations.



(a) Cross Entropy

(b) Accuracy

Figure 12: *(Binary Classification)*. Relative gain of DeGroot aggregation over model averaging (M-avg). The value of p is a proxi for the level of heterogeneity in the data partitioning.

D.3 Classification example

In this section we outline how the DeGroot aggregation procedure could be used in combination with classification tasks. The reason why we have focused on regression in the main body of the paper is because it gives us a better handle on optimality which is appealing from an analysis point of view. However there is nothing that prevents us from using the DeGroot consensus procedure for also solving classification tasks. It remains to define an appropriate score for determining mutual trust.

For simplicity, let us consider a binary classification task with $y \in \{0, 1\}$ and let the pre-trained models be L_2-regularized logistic regression models. The individual predictions $f_i(x')$ correspond to the predicted probability $\widehat{p}_i(x') = p_i(Y = 1|X = x')$ of model i. Then, to evaluate the trustworthiness of other models in Step 5 of Algorithm 1 we use the cross entropy loss instead of mean squared error. This is a natural score to use for classification, but there might also be other sensible choices. Formally, this means we replace (3) by:

$$\text{Loss}_{i,j}(x') = \frac{1}{|\mathcal{D}_i(x')|} \sum_{(x,y)\in\mathcal{D}_i(x')} \mathrm{I}[y = 1] \log(\widehat{p}_j(x)) + \mathrm{I}[y = 0] \log(1 - \widehat{p}_j(x))$$

and we compute the trust scores as

$$\tau_{ij} = \frac{1/\text{Loss}_{ij}}{\sum_{j\in[K]} 1/\text{Loss}_{ij}}. \tag{7}$$

Then, following the procedure outlined in Algorithm 1 we apply DeGroot to iteratively pool the individual predictions, using the trust scores as weights.

To evaluate the performance of DeGroot aggregation for this classification task empirically, we reproduce the heterogeneity experiment from Section 4.2 on the Phishing datasets for binary classification downloaded form the LIBSVM website [Fan, 2011]. Results are depicted in Figure 12. Here sorting the labels along the y-axis implies that the label imbalance in agents' partitions increases with p. We plot the relative gain in cross entropy as well as predictive accuracy of DeGroot over model averaging and compare it to individual agents as in Section 4.2. These results suggest that the consistent gains we have observed for regression also translate to classification.