Towards Model Agnostic Federated Learning Using Knowledge Distillation

Anonymous authors
Paper under double-blind review

Abstract

An often unquestioned assumption underlying most current federated learning algorithms is that all the participants use identical model architectures. In this work, we initiate a theoretical study of *model agnostic* communication protocols which would allow data holders (agents) using *different models* to collaborate with each other and perform federated learning. We focus on the setting where the two agents are attempting to perform kernel regression using different kernels (and hence have different models). Our study yields a surprising result—the most natural algorithm of using *alternating knowledge distillation* (AKD) imposes overly strong regularization and may lead to severe under-fitting. Our theory also shows an interesting connection between AKD and the alternating projection algorithm for finding intersection of sets. Leveraging this connection, we propose a new algorithm which improves upon AKD, but at the cost of using multiple models. Our theoretical predictions also closely match real world experiments using neural networks. Thus, our work proposes a rich yet tractable framework for analyzing and developing new practical model agnostic federated learning algorithms.

1 Introduction

Federated learning (and more generally collaborative learning) involves multiple data holders (whom we call agents) collaborating with each other to train their machine learning model over their collective data. Crucially, this is done without directly exchanging any of their raw data (McMahan et al., 2017; Kairouz et al., 2019). Thus, communication is limited to only what is essential for the training process and the data holders (aka agents) retain full ownership over their datasets.

Algorithms for this setting such as FedAvg or its variants all proceed in rounds (Wang et al., 2021). In each such round, the agents first train their models on their local data. Then, the knowledge from these different models is aggregated by *averaging the parameters*. However, exchanging knowledge via averaging the model parameters is only viable if all the agents use the same model architecture. This fundamental assumption is highly restrictive. Different agents may have different computational resources and hence may want to use different model architectures. Further, directly averaging the model parameters can fail even when all clients have the same architecture (Wang et al., 2019b; Singh & Jaggi, 2020; Yu et al., 2021). This is because the loss landscape of neural networks is highly non-convex and has numerous symmetries with different parameter values representing the same function. To overcome such limitations, we would need to take a *functional view* view of neural networks i.e. we need methods which are agnostic to the model architecture and parameters. This motivates the central question investigated in this work:

Can we design *model agnostic* federated learning algorithms which would allow each agent to train their model of choice on the combined dataset?

Specifically, we restrict the algorithms to access the models using only two primitives (an universal model API): train on some dataset i.e. *fit*, and yield predictions on some inputs i.e. *predict*. Our goal is to be able to collaborate with and learn from any agent which provides these two functionalities.

A naive such model agnostic algorithm indeed exists—agents can simply transfer their entire training data to each other and then each agent can train any model of choice on the combined dataset. However, transferring of the dataset is disallowed in federated learning. Instead, we will replace the averaging primitive in federated learning with *knowledge distillation* (KD) (Bucilua et al., 2006;

Hinton et al., 2015). In knowledge distillation (KD), information is transferred from model A to model B by training model B on the predictions of model A on some data. Since we only access model A through its predictions, KD is a functional model-agnostic protocol. The key challenge of KD however is that it is poorly understood and cannot be formulated in the standard stochastic optimization framework like established techniques (Wang et al., 2021). Thus, designing and analyzing algorithms which utilize KD requires developing an entirely new framework and approach.

Our Contributions. The main results in this work are

- We formulate the model agnostic learning problem as two agents with local datasets wanting to perform kernel regression on their combined datasets. Kernel regression is both simple enough to be theoretically tractable, and rich enough to capture non-linear function fitting thereby allowing each agent to have a different kernel (hence different models).
- We analyze alternating knowledge distillation (AKD) and show that it is closely linked to the alternating projection method for finding intersection of convex sets. Our analysis reveals that AKD can impose an overly strong regularization, leading to degradation of performance. This degradation is more severe when the two agents have very different models.
- Using the connection to alternating projection, we analyze other possible variants such as *averaged* knowledge distillation (AvgKD) and attempt to construct an 'optimal' scheme.
- Finally, we evaluate all algorithms on real world deep learning models and datasets, and show that the empirical behavior closely matches our insights from the theoretical analysis. This demonstrates the utility of our framework for analyzing and designing new algorithms.

2 Related work

Federated learning (FL). In FL (Kairouz et al., 2019), training data is distributed over several agents or locations. For instance, these could be several hospitals collaborating on a clinical trial, or billions of mobile phones involved in training a voice recognition application. The purpose of FL is to enable training on the union of all agents' individual data without needing to transmit any of the raw sensitive data. Typically, the training is coordinated by some trusted server. One can also instead use direct peer to peer communications (Nedic, 2020). A large body of work has designed algorithms for FL under the identical model setting where we either learn a single global model (McMahan et al., 2017; Reddi et al., 2020; Karimireddy et al., 2020b;a; Wang et al., 2021), or multiple personalized models (Wang et al., 2019a; Deng et al., 2020; Mansour et al., 2020; Grimberg et al., 2021).

Knowledge distillation (KD). Initially, KD was introduced as way to compress models i.e. as a way to transfer the knowledge of a large model to a smaller model (Bucilua et al., 2006; Hinton et al., 2015). Since then, it has found much broader applications such as improving generalization performance via self-distillation, learning with noisy data, and transfer learning (Yim et al., 2017). We refer to a recent survey (Gou et al., 2021) for progress in this vast area.

KD in FL. Numerous papers propose to use KD to transfer knowledge from the agent models to a centralized server model (Seo et al., 2020; Sattler et al., 2020; Lin et al., 2020; Li et al., 2020; Wu et al., 2021). However, all of these methods rely on access to some common public dataset which may be impractical. In the closely related *codistillation* setting (Zhang et al., 2018; Anil et al., 2018; Sodhani et al., 2020), an ensemble of students learn collaboratively without a central server model. While codistillation does not need additional unlabelled data, it is only suitable for distributed training within a datacenter since it assumes all agents have access to the same dataset (i.i.d). In FL however, there is both model heterogeneity and data heterogeneity. Further, none of the afore-mentioned algorithms have any theoretical analysis.

KD analysis. Despite the empirical success of KD, it is poorly understood with very little theoretical analysis. Phuong & Lampert (2021) explore a generalization bound for a distillation-trained linear model, and Tang et al. (2021) conclude that by using KD one re-weights the training examples for the student, and Menon et al. (2020) consider a Bayesian view showing that the student learns better if the teacher provides the true Bayes probability distribution. Allen-Zhu & Li (2020) show how ensemble distillation can preserve their diversity in the student model. Finally, Mobahi et al. (2020) consider self-distillation in a kernel regression setting i.e. the model is retrained using its own predictions on the training data. They show that iterative self-distillation induces a strong regularization effect. We significantly extend their theoretical framework in our work to analyze KD in federated learning where agents have different models and different datasets.

3 Framework and setup

Notation. We denote a set as $\mathcal{A}$, a matrix as $\boldsymbol{A}$, and a vector as $\boldsymbol{a}$. $\boldsymbol{A}[i,j]$ is the (i, j)-th element of matrix $\boldsymbol{A}$, $\boldsymbol{a}[i]$ denotes the i'th element of vector $\boldsymbol{a}$. $||\boldsymbol{a}||$ denotes the ℓ_2 norm of vector $\boldsymbol{a}$.

Centralized kernel regression (warmup). Consider, as a warm-up, the centralized setting with a training dataset $\mathcal{D} \subseteq \mathbb{R}^d \times \mathbb{R}$. That is, $\mathcal{D} = \cup_i^N \{(\boldsymbol{x}_i, y_i)\}$, where $\boldsymbol{x}_n \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y_n \in \mathcal{Y} \subseteq \mathbb{R}$. Given training set $\mathcal{D}$, our aim is to find best function $f^\star \in \mathcal{F} : \mathcal{X} \to \mathcal{Y}$. To find $f^\star$ we solve the following regularized optimization problem:

$$f^\star := \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_n (f(\boldsymbol{x}_n) - y_n)^2 + cR_u(f), \text{ with} \tag{1}$$

$$R_u(f) := \int_{\mathcal{X}} \int_{\mathcal{X}} u(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}) f(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \,. \tag{2}$$

Here, $\mathcal{F}$ is defined to be the space of all functions such that (2) is finite, c is the regularization parameter, and $u(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function. That is, u is symmetric $u(\boldsymbol{x}, \boldsymbol{x}') = u(\boldsymbol{x}', \boldsymbol{x})$ and positive with $R_u(f) = 0$ only when $f = 0$ and $R_u(f) > 0$ o.w. Further, let $k(\boldsymbol{x}, \boldsymbol{t})$ be the function s.t.

$$\int_{\mathcal{X}} u(\boldsymbol{x}, \boldsymbol{x}') k(\boldsymbol{x}', \boldsymbol{t}) d\boldsymbol{x}' = \delta(\boldsymbol{x} - \boldsymbol{t}) \,, \quad \text{where } \delta(\cdot) \text{ is the Dirac delta function.} \tag{3}$$

Now, we can define the positive definite matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ and vector $\boldsymbol{k}_{\boldsymbol{x}} \in \mathbb{R}^N$ as:

$$\boldsymbol{K}[i,j] := \frac{1}{N} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{and} \quad \boldsymbol{k}_{\boldsymbol{x}}[i] := \frac{1}{N} k(\boldsymbol{x}, \boldsymbol{x}_i), \quad \text{for} \quad \boldsymbol{x}_i \in \mathcal{D}, \forall i \in [N] \,. \tag{4}$$

Note that $\boldsymbol{k}_{\boldsymbol{x}}$ is actually a vector valued function which takes any $\boldsymbol{x} \in \mathcal{X}$ as input, and both $\boldsymbol{k}_{\boldsymbol{x}}$ and $\boldsymbol{K}$ depend on the training data $\mathcal{D}$. We can then derive a closed form solution for $f^\star$.

Proposition I (Schölkopf et al. (2001)). *The $f^\star$ which minimizes (1) is given by*

$$f^\star(\boldsymbol{x}) = \boldsymbol{k}_{\boldsymbol{x}}^\top (c\boldsymbol{I} + \boldsymbol{K})^{-1} \boldsymbol{y} \,, \quad \textit{for} \quad \boldsymbol{y}[i] := y_i, \forall i \in [N] \,.$$

Note that on the training data $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ with $\boldsymbol{X}[i,:] = \boldsymbol{x}_i$, we have $f^\star(\boldsymbol{X}) = \boldsymbol{K}(c\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{y}$. Kernel regression for an input $\boldsymbol{x}$ outputs a weighted average of the training $\{y_n\}$. These weights are computed using a learned measure of distance between the input $\boldsymbol{x}$ and the training $\{\boldsymbol{x}_n\}$. Intuitively, the choice of the kernel $u(\boldsymbol{x}, \boldsymbol{x}')$ creates an inductive bias and corresponds to the choice of a model in deep learning, and the regularization parameter c acts similarly to tricks like early stopping, large learning rate etc. which help in generalization. When $c = 0$, we completely fit the training data and the predictions exactly recover the labels with $f^\star(\boldsymbol{X}) = \boldsymbol{K}(\boldsymbol{K})^{-1}\boldsymbol{y} = \boldsymbol{y}$. When $c > 0$ the predictions $f^\star(\boldsymbol{X}) = \boldsymbol{K}(c\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{y} \neq \boldsymbol{y}$ and they incorporate the inductive bias of the model. In knowledge distillation, this extra information carried by the predictions about the inductive bias of the model is popularly referred to as "dark knowledge" (Hinton et al., 2015).

Federated kernel regression (our setting). We have two agents, with agent 1 having dataset $\mathcal{D}_1 = \cup_i^N \{(\boldsymbol{x}_i^1, y_i^1)\}$ and agent 2 with dataset $\mathcal{D}_2 = \cup_i^N \{(\boldsymbol{x}_i^2, y_i^2)\}$. Agent 1 aims to find the best approximation mapping $g^\star \in \mathcal{F}_1 : \mathcal{X} \to \mathcal{Y}$ using a kernel $u_1(\boldsymbol{x}, \boldsymbol{x}')$ and objective:

$$g^{1\star} := \arg\min_{g \in \mathcal{F}_1} \frac{1}{2N} \sum_n (g(\boldsymbol{x}_n^1) - y_n^1)^2 + \frac{1}{2N} \sum_n (g(\boldsymbol{x}_n^2) - y_n^2)^2 + cR_{u_1}(g), \text{with} \tag{5}$$

$$R_{u_1}(g) := \int_{\mathcal{X}} \int_{\mathcal{X}} u_1(\boldsymbol{x}, \boldsymbol{x}') g(\boldsymbol{x}) g(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \,. \tag{6}$$

Note that the objective of agent 1 is defined using its *individual kernel* $u_1(\boldsymbol{x}, \boldsymbol{x}')$, but over the *joint dataset* $(\mathcal{D}_1, \mathcal{D}_2)$. Correspondingly, agent 2 also uses its own kernel function $u_2(\boldsymbol{x}, \boldsymbol{x}')$ to define the regularizer $R_{u_2}(g^2)$ over the space of functions $g^2 \in \mathcal{F}_2$ and optimum $g^{2\star}$. Thus, our setting has model heterogeneity (different kernel functions u_1 and u_2) and data heterogeneity ($\mathcal{D}_1$ and $\mathcal{D}_2$ are not i.i.d.). Given that the setting is symmetric between agents 1 and 2, we can focus solely on error in terms of agent 1's objective (5) without loss of generality.

Proposition I can be used to derive a closed form form for function $g^{1\star}$ minimizing objective (5). However, computing this requires access to the datasets of both agents. Instead, we ask "can we design an iterative federated learning algorithm which can approximate $g^{1\star}$"?

4 Alternating knowledge distillation

In this section we describe a popular iterative knowledge distillation algorithm and analyze its updates in our framework. Our analysis leads to some surprising connections between KD and projection onto convex sets, and shows some limitations of the current algorithm.

Algorithm. Denote the data on agent 1 as $\mathcal{D}_1 = (\boldsymbol{X}^1, \boldsymbol{y}^1)$ where $\boldsymbol{X}^1[i,:] = \boldsymbol{x}_n^1$ and $\boldsymbol{y}^1[i] = y_i^1$. Correspondingly, we have $\mathcal{D}^2 = (\boldsymbol{X}^2, \boldsymbol{y}^2)$. Now starting from $\hat{\boldsymbol{y}}_0^1 = \boldsymbol{y}^1$, in each round $t \geq 0$:

a. Agent 1 trains their model on dataset $(\boldsymbol{X}^1, \hat{\boldsymbol{y}}_t^1)$ to obtain g_t^1.
b. Agent 2 receives g_t^1 and uses it to predict labels $\hat{\boldsymbol{y}}_t^2 = g_t^1(\boldsymbol{X}^2)$.
c. Agent 2 trains their model on dataset $(\boldsymbol{X}^2, \hat{\boldsymbol{y}}_t^2)$ to obtain h_t
d. Agent 1 receives a model h_t from agent 2 and predicts $\hat{\boldsymbol{y}}_{t+1} = h_t(\boldsymbol{X}_1)$.

Thus the algorithm alternates between training and knowledge distillation on each of the two agents. We also summarize the algorithm in Figure 1a. Importantly, note that there is no exchange of raw data but only of the trained models. Further, each agent trains their choice of model on their data with agent 1 training $\{g_t^1\}$ and agent 2 training $\{h_t\}$.

4.1 Theoretical analysis

Similar to (3), let us define functions $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and $k_2(\boldsymbol{x}, \boldsymbol{x}')$ such that they satisfy

$$\int_{\mathcal{X}} u_a(\boldsymbol{x}, \boldsymbol{x}') k_a(\boldsymbol{x}', \boldsymbol{t}) d\boldsymbol{x}' = \delta(\boldsymbol{x} - \boldsymbol{t}) \quad \text{for } a \in \{1, 2\}\,.$$

For such functions, we can then define the following positive definite matrix $\boldsymbol{L} \in \mathbb{R}^{2N \times 2N}$:

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{L}_{12} \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22} \end{pmatrix}, \quad \boldsymbol{L}_{a,b}[i,j] = \frac{1}{N} k_1(\boldsymbol{x}_i^a, \boldsymbol{x}_j^b) \text{ for } a, b \in \{1,2\} \text{ and } i, j \in [N]\,.$$

Note $\boldsymbol{L}$ is symmetric (with $\boldsymbol{L}_{12}^\top = \boldsymbol{L}_{21}$) and is also positive definite. Further, each component $\boldsymbol{L}_{a,b}$ measures pairwise similarities between inputs of agent a and agent b using the kernel k_1. Correspondingly, we define $\boldsymbol{M} \in \mathbb{R}^{2N \times 2N}$ which uses kernel k_2:

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{21} & \boldsymbol{M}_{22} \end{pmatrix}, \quad \boldsymbol{M}_{a,b}[i,j] = \frac{1}{N} k_2(\boldsymbol{x}_i^a, \boldsymbol{x}_j^b) \text{ for } a, b \in \{1,2\} \text{ and } i, j \in [N]\,.$$

We can now derive the closed form of the AKD algorithm repeatedly using Proposition I.

Proposition II. *The model in round t learned by the alternating knowledge distillation algorithm is*

$$g_t^1(\boldsymbol{x}) = \boldsymbol{l}_{\boldsymbol{x}}^\top (c\boldsymbol{I} + \boldsymbol{M}_{22})^{-1} \boldsymbol{L}_{21} (c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1} \left(\boldsymbol{M}_{12} (c\boldsymbol{I} + \boldsymbol{M}_{22})^{-1} \boldsymbol{L}_{21} (c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1} \right)^{t-1} \boldsymbol{y}^1\,,$$

where $\boldsymbol{l}_{\boldsymbol{x}} \in \mathbb{R}^N$ is defined as $\boldsymbol{l}_{\boldsymbol{x}}[i] = \frac{1}{N} k_1(\boldsymbol{x}, \boldsymbol{x}_i^1)$. Further, for any fixed $\boldsymbol{x}$ we have

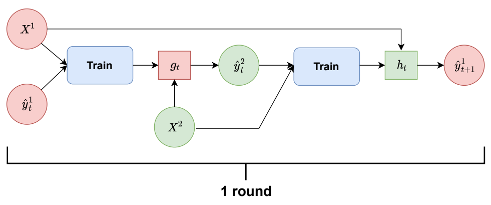
$$\lim_{t \to \infty} g_t^1(\boldsymbol{x}) = 0\,.$$

First, note that if agents 1 and 2 are identical with the same data and same model, we have $\boldsymbol{M}_{12} = \boldsymbol{M}_{22} = \boldsymbol{L}_{11} = \boldsymbol{L}_{12}$. This setting corresponds to *self-distillation* where the model is repeatedly retrained on its own predictions. Proposition II shows that after $2t$ rounds of self-distillation, we obtain a model is of the form $g_t^1(\boldsymbol{x}) = \boldsymbol{l}_{\boldsymbol{x}}^\top (c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1} \left(\boldsymbol{L}_{11} (c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1} \right)^{2t-1} \boldsymbol{y}$. Here, the effect of c is amplified as t increases. Thus, this shows that repeated self-distillation induces a strong regularization effect, recovering the results of (Mobahi et al., 2020).

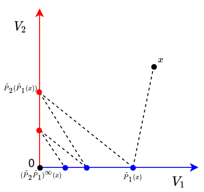
Perhaps more strikingly, Proposition II shows that not only does AKD fail to converge to the actual optimal solution $g^{1^\star}$ as defined in (5), it will also slowly degrade and eventually converges to 0. We next expand upon this and explain this phenomenon.

4.2 Degradation and connection to projections

While mathematically, Proposition II completely describes the AKD algorithm, it does not provide much intuition. In this section, we rephrase the result in terms of projections and contractions which provides a more visual understanding of the method.



(a) Alternating KD starting from agent 1. We predict and train using predictions, alternating between the two agents.


V_2
x
$\tilde{P}_2(\tilde{P}_1(x))$
0
$(\tilde{P}_2\tilde{P}_1)^\infty(x)$
$\tilde{P}_1(x)$
V_1

(b) Alternating oblique projections starting from x converges to 0.

Oblique projections. A projection operator $\boldsymbol{P}$ linearly maps (projects) all inputs onto some linear subspace $\mathcal{A} = \text{Range}(\boldsymbol{A})$. In general, we can always rewrite such a projection operation as

$$\boldsymbol{P}\boldsymbol{x} = \min_{\boldsymbol{y}\in\text{Range}(\boldsymbol{A})} (\boldsymbol{y}-\boldsymbol{x})^\top \boldsymbol{W}(\boldsymbol{y}-\boldsymbol{x}) = \boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{x}\,.$$

Here, $\boldsymbol{A}$ defines an orthonormal basis of the space $\mathcal{A}$ and $\boldsymbol{W}$ is a positive definite weight matrix which defines the geometry $\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\boldsymbol{W}} := \boldsymbol{x}^\top\boldsymbol{W}\boldsymbol{y}$. When $\boldsymbol{W} = \boldsymbol{I}$, we recover the familiar orthogonal projection. Otherwise, projections can happen 'obliquely' following the geometry defined by $\boldsymbol{W}$.

Contractions. A contraction is a linear operator $\boldsymbol{C}$ which contracts all inputs towards the origin:

$$\|\boldsymbol{C}\boldsymbol{x}\| \leq \|\boldsymbol{x}\| \quad \text{for any } \boldsymbol{x}\,.$$

Given these notions, we can rewrite Proposition II as follows.

Proposition III. *There exist oblique projection operators $\boldsymbol{P}_1^\top$ and $\boldsymbol{P}_2^\top$, contraction matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$, and orthonormal matrices $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ such that the model in round t learned by the alternating knowledge distillation algorithm is*

$$g_t^1(\boldsymbol{x}) = \boldsymbol{l}_{\boldsymbol{x}}^\top(c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1}\boldsymbol{V}_1^\top \left(\boldsymbol{C}_1\boldsymbol{P}_2^\top\boldsymbol{C}_2\boldsymbol{P}_1^\top\right)^t \boldsymbol{V}_1\boldsymbol{y}^1\,.$$

In particular, the predictions on agent 2's inputs are

$$g_t^1(\boldsymbol{X}^2) = \boldsymbol{V}_2^\top\boldsymbol{P}_1^\top \left(\boldsymbol{C}_1\boldsymbol{P}_2^\top\boldsymbol{C}_2\boldsymbol{P}_1^\top\right)^t \boldsymbol{V}_1\boldsymbol{y}^1\,.$$

Further, the projection matrices satisfy $\boldsymbol{P}_1^\top\boldsymbol{P}_2^\top\boldsymbol{x} = \boldsymbol{x}$ only if $\boldsymbol{x} = 0$.

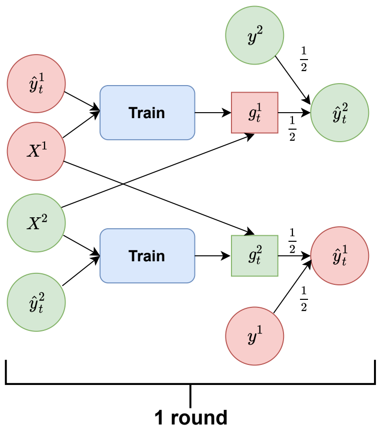
Let us examine the term $\left(\boldsymbol{C}_1\boldsymbol{P}_2^\top\boldsymbol{C}_2\boldsymbol{P}_1^\top\right)^t$ in the above result. This is the only term which depends on the number of rounds t and so captures the dynamics of the algorithm. The result can be interpreted as follows: two rounds of alternating knowledge distillation (first to agent 1 and then back to 2) correspond to a multiplication with $\boldsymbol{C}_1\boldsymbol{P}_2^\top\boldsymbol{C}_2\boldsymbol{P}_1^\top$ i.e. *alternating projections* interspersed by contractions. Thus, the dynamics of alternating knowledge distillation is exactly the same as that of alternating projections interspersed by contractions. The projections $\boldsymbol{P}_1^\top$ and $\boldsymbol{P}_2^\top$ have orthogonal ranges whose intersection only contains the origin 0. As Fig. 1b shows, non-orthogonal alternating projections between orthogonal spaces converge to the origin. Contractions further pull the inputs closer to the origin, speeding up the convergence.

Speed of degradation. The alternating projection algorithm converges to the intersection of the subspaces corresponding to the projection operators (their range) (Boyd & Dattorro, 2003). In our particular case, the fixed point of $\boldsymbol{P}_1^\top$ and $\boldsymbol{P}_2^\top$ is 0, and hence this is the point the algorithm will converge to. The contraction operations $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ only speed up the convergence to the origin 0 (also see Fig. 1b). This explains the degradation process notes in Proposition II. We can go further and examine the rate of degradation using known analyses of alternating projections Aronszajn (1950).

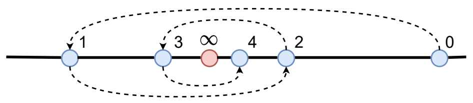
Proposition IV (Informal)**.** *The rate of convergence to $g_t^1(\boldsymbol{x})$ to 0 gets faster if:*

- *a stronger inductive bias is induced via a larger regularization constant c,*
- *the kernels $k_1(\boldsymbol{x}, \boldsymbol{y})$ and $k_2(\boldsymbol{x}, \boldsymbol{y})$ are very different, or*
- *the difference between the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ as measured by $k_1(\boldsymbol{x}, \boldsymbol{y})$ increases.*

In summary, both data and model heterogeneity may speed up the degradation defeating the purpose of model agnostic FL. All formal proofs and theorem statements are moved to the Appendix.



(a) AvgKD scheme



(b) Intuition behind ensemble scheme. In each round, AKD alternates between overfitting the data of agent 1 or of agent 2. We can construct an ensemble out of these models to correct for this bias and quickly converge to the true optimal.

5 Additional Variants

In the previous section, we saw that the alternating knowledge distillation (AKD) algorithm over multiple iterations suffered slow degradation, eventually losing all information about the training data. In this section we explore some alternative approaches which attempt to correct for this. We first analyze a simple way to re-inject the training data after every KD iteration which we call averaged distillation. Then, we show an ensemble algorithm which can recover the optimal model $g^{1\star}$.

5.1 Averaged knowledge distillation

As we saw earlier, each step of knowledge distillation seems to lose some information about the training data, replacing with the inductive bias of the model. One approach to counter this slow loss of information is to recombine it with the original training data labels such as is commonly done in co-distillation (Sodhani et al., 2020).

Algorithm. Recall that agent 1 has data $\mathcal{D}_1 = (\boldsymbol{X}^1, \boldsymbol{y}^1)$ and correspondingly agent 2 has data $\mathcal{D}^2 = (\boldsymbol{X}^2, \boldsymbol{y}^2)$. Now starting from $\hat{\boldsymbol{y}}_0^1 = \boldsymbol{y}^1, \hat{\boldsymbol{y}}_0^2 = \boldsymbol{y}^2$, in each round $t \geq 0$:

a. Agents 1 and 2 train their model on datasets $(\boldsymbol{X}^1, \hat{\boldsymbol{y}}_t^1)$ and $(\boldsymbol{X}^2, \hat{\boldsymbol{y}}_t^2)$ to obtain g_t^1 and g_t^2.
b. Agents exchange their models g_t^1 and g_t^2 between each other.
c. Agents use exchanged models to predict labels $\hat{\boldsymbol{y}}_{t+1}^1 = \frac{\boldsymbol{y}^1 + g_t^2(\boldsymbol{X}^1)}{2}$, $\hat{\boldsymbol{y}}_{t+1}^2 = \frac{\boldsymbol{y}^2 + g_t^1(\boldsymbol{X}^2)}{2}$.

The summary the algorithm is depicted in Figure 2a. Again, notice that there is no exchange of raw data but only of the trained models. The main difference between AKD and AvgKD (averaged knowledge distillation) is that we average the predictions with the original labels. This re-injects information $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ at every iteration, preventing degradation. We theoretically characterize its dynamics next in terms of the afore-mentioned contraction and projection operators.

Proposition V. *There exist oblique projection operators $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, contraction matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$, and orthonormal matrices $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ such that the model of agent 1 in round t learned by the averaged knowledge distillation (AvgKD) algorithm is*

$$g_t^1(\boldsymbol{x}) = \frac{\boldsymbol{F}}{2}\Big(\sum_{i=0}^{t-1}\Big(\frac{\boldsymbol{C}_1\boldsymbol{P}_2^\top\boldsymbol{C}_2\boldsymbol{P}_1^\top}{4}\Big)^i\Big)\boldsymbol{z}_1 + \frac{\boldsymbol{F}\boldsymbol{C}_1\boldsymbol{P}_2^\top}{4}\Big(\sum_{i=0}^{t-2}\Big(\frac{\boldsymbol{C}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\boldsymbol{P}_2^\top}{4}\Big)^i\Big)\boldsymbol{z}_2\,.$$

where $\boldsymbol{F} = \boldsymbol{l}_{\boldsymbol{x}}^\top(c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1}\boldsymbol{V}_1^\top$. Further, in the limit of rounds for any fixed $\boldsymbol{x}$ we have

$$\lim_{t\to\infty} g_t^1(\boldsymbol{x}) = \frac{\boldsymbol{F}}{2}\Big(\boldsymbol{I} - \frac{\boldsymbol{C}_1\boldsymbol{P}_2^\top\boldsymbol{C}_2\boldsymbol{P}_1^\top}{4}\Big)^\dagger\boldsymbol{z}_1 + \frac{\boldsymbol{F}\boldsymbol{C}_1\boldsymbol{P}_2^\top}{4}\Big(\boldsymbol{I} - \frac{\boldsymbol{C}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\boldsymbol{P}_2^\top}{4}\Big)^\dagger\boldsymbol{z}_2\,.$$

This shows that the model learned through AvgKD does not degrade to 0, unlike AKD. Instead, it converges to a limit for which we can derive closed-form expressions. Unfortunately, this limit model is still not the same as our desired optimal model $g^{1\star}$. We next try overcome this using ensembling.

5.2 Ensembled Knowledge Distillation

We first analyze how the limit solution of AvgKD differs from the actual optimum $g^{1^\star}$. We will build upon this understanding to construct an ensemble which approximates $g^{1^\star}$.

Understanding AvgKD. For simplicity, we assume that the regularization coefficient $c = 0$. Then,

Proposition VI. *There exist matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ such that the minimizer $g^{1^\star}$ of objective (5) predicts $g^{1^\star}(\boldsymbol{X}_i) = \boldsymbol{A}_1\boldsymbol{\beta}_1 + \boldsymbol{A}_2\boldsymbol{\beta}_2$ for $i \in \{1, 2\}$, where $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ satisfy*

$$\begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{L}_{21} & \boldsymbol{M}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}. \tag{7}$$

In contrast, for the same matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, the limit models of AvgKD predict $g^1_\infty(\boldsymbol{X}_i) = \frac{1}{2}\boldsymbol{A}_1\boldsymbol{\beta}_1$ and $g^2_\infty(\boldsymbol{X}_i) = -\frac{1}{2}\boldsymbol{A}_2\boldsymbol{\beta}_2$, for $i \in \{1, 2\}$ for $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ satisfying

$$\begin{pmatrix} \boldsymbol{L}_{11} & \frac{\boldsymbol{M}_{12}}{2} \\ \frac{\boldsymbol{L}_{21}}{2} & \boldsymbol{M}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ -\boldsymbol{y}_2 \end{pmatrix}. \tag{8}$$

By comparing the equations (7) and (8), the output $2(g^1_\infty(\boldsymbol{x}) - g^2_\infty(\boldsymbol{x}))$ is close to the output of $g^{1^\star}(\boldsymbol{x})$, except that the off-diagonal matrices are scaled by $\frac{1}{2}$ and we have $-\boldsymbol{y}_2$ on the right hand side. We need two tricks if we want to approximate $g^{1^\star}$: first we need an ensemble using differences of models, and second we need to additionally correct for the bias in the algorithm.

Correcting bias using infinite ensembles. Consider the initial AKD (alternating knowledge distillation) algorithm illustrated in the Fig. 1a. Let us run two simultaneous runs, ones starting from agent 1 and another starting from agent 2, outputting models $\{g^1_t\}$, and $\{g^2_t\}$ respectively. Then, instead of just using the final models, we construct the following infinite ensemble. For an input $\boldsymbol{x}$, we output:

$$f_\infty(\boldsymbol{x}) = \sum_{t=0}^{\infty} (-1)^t (g^1_t(\boldsymbol{x}) + g^2_t(\boldsymbol{x})) \tag{9}$$

That is, we take the models from odd steps t with positive signs and from even ones with negative signs and sum their predictions. We call this scheme Ensembled Knowledge Distillation (EKD). The intuition behind our ensemble method is schematically visualized in 1-dimensional case in the Fig. 2b where numbers denote the t variable in the equation (9). We start from the sum of both agents models obtained after learning from ground truth labels (0-th round). Then we subtract the sum of both agents models obtained after first round of KD. Then we add the sum of both agents models obtained after second round of KD and so on. From the section 4.2 we know that in the AKD process model gradually degrades towards $\boldsymbol{0}$, in other words each next round obtained model adds less values to the whole sum. Hence we gradually converge to the limit model, which is the point ∞ in the Fig. 2b. We formalize this and prove the following.

Proposition VII. *The predictions of f_∞ using (9) satisfies $f_\infty(\boldsymbol{X}_i) = g^{1^\star}(\boldsymbol{X}_i)$ for $i \in \{1, 2\}$.*

Thus, not only did we succeed in preventing degeneracy to 0, we also managed to recover the predictions of the optimal model. However, note that this comes at a cost of an infinite ensemble. While we can approximate this using just a finite set of models (as we explore experimentally next), this still does not recover a *single* model which matches $g^{1^\star}$.

6 Experiments

Setup. We consider three settings corresponding to the cases Proposition IV with the agents having

- the same model architecture and close data distributions (Same model, Same data)
- different model architectures and close data distributions (Different model, Same data)
- the same model architecture and different data distributions (Same model, Different data).

The toy experiments solve a linear regression problem of the form $\boldsymbol{A}\boldsymbol{x}^\star = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{x}^\star \in \mathbb{R}^d$ is randomly generated for $n = 1.5d$ and $d = 100$. Then, the data $\boldsymbol{A}$ and $\boldsymbol{b}$ is split between the two agents at proportion $0.6/0.4$. This is done randomly in the 'same data' case, whereas in the 'different data' case the data is sorted according to b before splitting to maximize heterogeneity.

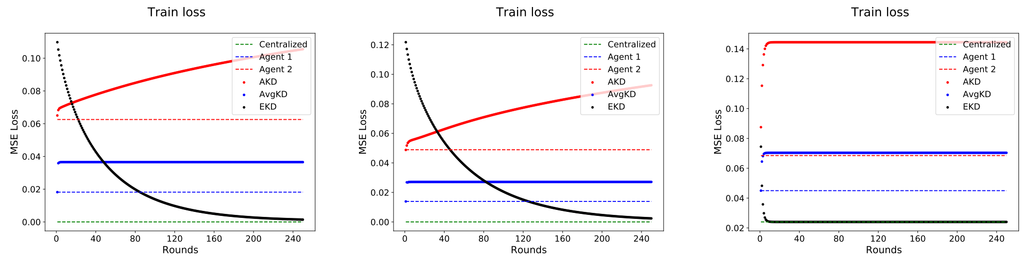


Figure 3: AKD, AvgKD, and EKD methods for linear regression on synthetic data with same data (left), different data (middle), and strong regularization (right). EKD (black) eventually matches centralized performance (dashed green), whereas AvgKD (solid blue) is worse than only local training (dashed blue and red). AKD (in solid red) performs the worst and degrades with increasing rounds.

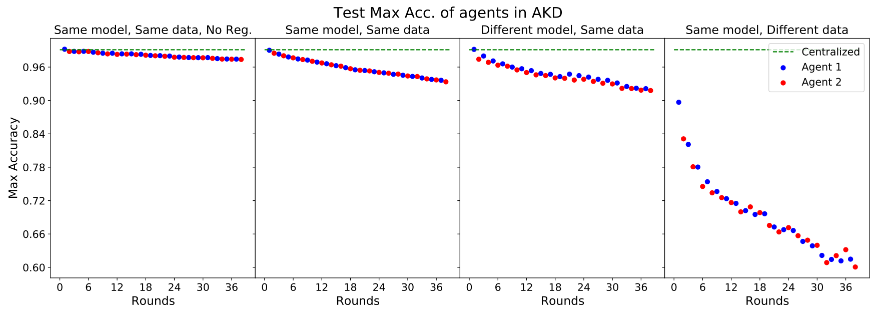


Figure 4: Test accuracy of centralized (dashed green), and AKD on MNIST using model starting from agent 1 (blue) and agent 2 (red) with varying amount of regularization, model heterogeneity, and data heterogeneity. In all cases, performance degrades with increasing rounds with degradation speeding up with increase in regularization, model heterogeneity, or data heterogeneity.

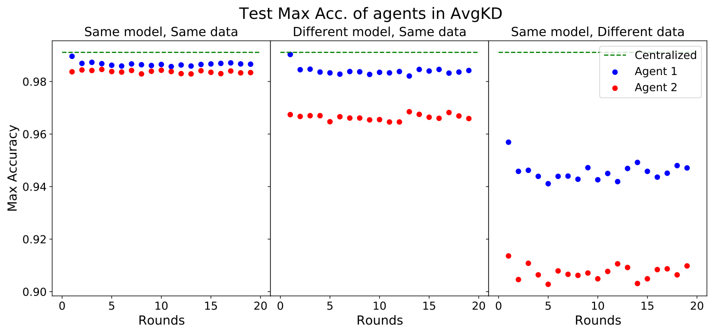


Figure 5: Test accuracy of AvgKD on MNIST using model starting from agent 1 (blue) and agent 2 (red) with varying model heterogeneity, and data heterogeneity. In all cases, there is no degradation of performance, though the best accuracy is obtained by agent 1 in round 1 with only local training.

The real world experiments are conducted using a CNN and MLP network on MNIST, and VGG16 and CNN models on CIFAR10 datasets. We use squared loss since it is closer to the the theoretical setting. In the 'same model' setting both agents use the CNN model, whereas agent 2 instead uses an MLP in the 'different model' setting. Further, we split the training data randomly with a $70\% - 30\%$ split in the same dataset setting. For the different data setting, we randomly subsample a very small percentage of the data—1% for AKD and 4% for AvgKD. By subsampling a very small percentage, the two agents datasets are likely very different from each other. Finally, we use the Adam optimizer in all experiments with a default regularization (weight decay) of 3×10^{-4}, unless in the 'no regularization' case when it is set to 0. We next summarize and discuss our results.

AvgKD > AKD. In all settings (both synthetic and real world), we see that with increasing number of rounds the performance of AKD significantly degrades whereas that of AvgKD stabilizes. However,

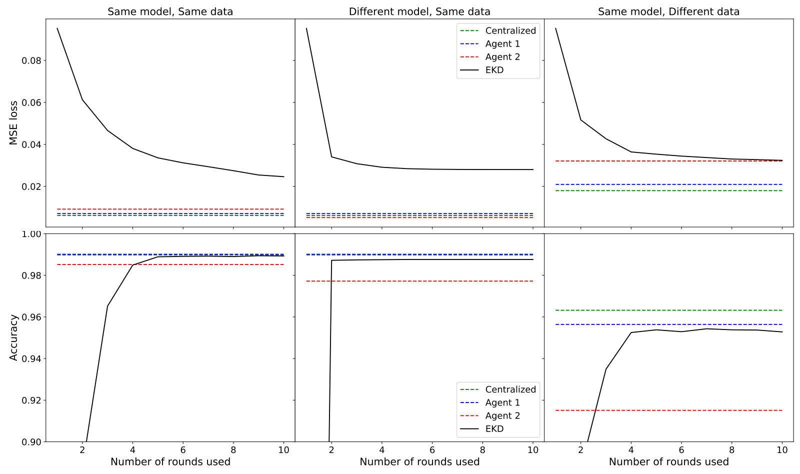


Figure 6: Test loss (top) and test accuracy (bottom) of EKD (black) on MNIST as compared to centralized training (dashed green) and local only training (dashed blue and green). Data and model heterogeneity is varied left to right. While the test accuracy and loss of EKD improves with rounds, we do not recover the centralized performance at the end of 10 rounds (20 models ensemble).

even in AvgKD the best model is the one trained using only local data in the first round, and does not match centralized accuracy. Thus, here too the additional KD steps do not improve performance.

EKD > AvgKD. In all experiments, EKD outperformed AvgKD and AKD. In the synthetic setting (Fig. 3), in 250 rounds (ensemble of 500 models) it even matches the centralized model. However, in the real world setting the improvement is slower and it doe not match centralized performance. This might be due to the small number of rounds run (only 10). EKD is also the only method which improves with subsequent rounds. Finally, we observed that increasing regularization actually sped up the convergence of EKD, while it worsened AKD and AvgKD.

Data heterogeneity > model heterogeneity. In all our experiments, both data and model heterogeneity degraded the performance of AKD and AvgKD. However, model heterogeneity had a much more milder effect, showing that model agnostic protocols are feasible. Replicating our experiments on other federated datasets with realistic heterogeneity is an exciting future question.

In the appendix, we show that similar trends hold for the cross entropy loss (Figs. 8, 9), as well on CIFAR10 in Fig. 10. In the latter, we see higher speeds of degradation which is probably due to CIFAR10 being much more complex (meaning potentially larger data heterogeneity), and also more complex models such as VGG (potentially larger model heterogeneity).

7 Conclusion

While stochastic optimization framework has been very useful in analyzing and developing new algorithms for federated learning so far, it fundamentally cannot capture learning with different models. We instead introduced the federated kernel regression framework where formalize both notions of model heterogeneity and data heterogeneity. Using this, we analyzed different knowledge distillation schemes and came to the surprising conclusion that they are all lacking–either they degenerate quickly or they do not converge to the optimum. Further, these theoretical results are exactly reflected in our deep learning experiments as well. This demonstrates the strong potential of using our framework to analyze and develop new algorithms for model agnostic federated learning.

We also utilize our framework to design a novel ensembling method motivated by correcting for the bias. However, this method could require very large ensembles (up to 500 models) in order to match the centralized performance. Thus, we view our ensembling method not as a practical algorithm, but more of a demo on how our framework can be leveraged to design novel algorithms. Similarly, our experiments are prelimnary and are not on real world datasets. We believe there is great potential in further exploring our results and using our framework to design new algorithms.

References

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv 2012.09816*, 2020.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv 1804.03235*, 2018.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Stephen Boyd and Jon Dattorro. Alternating projections. 01 2003.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv 2003.13461*, 2020.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Felix Grimberg, Mary-Anne Hartley, Sai Praneeth Karimireddy, and Martin Jaggi. Optimal model averaging: Towards personalized collaborative learning. *FL ICML workshop*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv 1503.02531*, 2015.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv 1912.04977*, 2019.

Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv 2008.03606*, 2020a.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. In *37th International Conference on Machine Learning (ICML)*, 2020b.

Adrian Lewis, Russell Luke, and Jerome Malick. Local convergence for alternating and averaged nonconvex projections. *arXiv 0709.0109*, 2007.

Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. *arXiv 2010.01017*, 2020.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *arXiv 2006.07242*, 2020.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv 2002.10619*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, pp. 1273–1282, 2017.

Aditya Krishna Menon, Ankit Singh Rawat, Sashank J. Reddi, Seungyeon Kim, and Sanjiv Kumar. Why distillation helps: a statistical perspective. *arXiv 2005.10419*, 2020.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-distillation amplifies regularization in hilbert space. *arXiv 2002.05715*, 2020.

Angelia Nedic. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020.

Mary Phuong and Christoph H. Lampert. Towards understanding knowledge distillation. *arXiv 2105.13093*, 2021.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv 2003.00295*, 2020.

Felix Sattler, Arturo Marban, Roman Rischke, and Wojciech Samek. Communication-efficient federated distillation. *arXiv 2012.00632*, 2020.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pp. 416–426. Springer, 2001.

Hyowoon Seo, Jihong Park, Seungeun Oh, Mehdi Bennis, and Seong-Lyun Kim. Federated knowledge distillation. *arXiv 2011.02367*, 2020.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020.

Shagun Sodhani, Olivier Delalleau, Mahmoud Assran, Koustuv Sinha, Nicolas Ballas, and Michael Rabbat. A closer look at codistillation for distributed training. *arXiv 2010.02838*, 2020.

Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H. Chi, and Sagar Jain. Understanding and improving knowledge distillation. *arXiv 2002.03532*, 2021.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv 2107.06917*, 2021.

Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv 1910.10252*, 2019a.

Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019b.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

Chuhan Wu, Fangzhao Wu, Ruixuan Liu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Fedkd: Communication efficient federated learning via knowledge distillation. *arXiv 2108.13323*, 2021.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141, 2017.

Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2066–2074, 2021.

Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4320–4328, 2018.

A ALTERNATING KD WITH REGULARIZATION

Different models Keeping the notation of section 4 we can construct the following matrix:

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{L} & 0 \\ 0 & \boldsymbol{M} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{L}_{12} & 0 & 0 \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22} & 0 & 0 \\ 0 & 0 & \boldsymbol{M}_{11} & \boldsymbol{M}_{12} \\ 0 & 0 & \boldsymbol{M}_{21} & \boldsymbol{M}_{22} \end{pmatrix}.$$

Notice that each of the sub-blocks $\boldsymbol{L}$ and $\boldsymbol{M}$ are symmetric positive semi-definite matrices. This makes matrix $\boldsymbol{K}$ symmetric positive semi-definite matrix and it has eigen-decomposition form:

$$\boldsymbol{K} = \boldsymbol{V}^\top \boldsymbol{D} \boldsymbol{V} \quad \text{and} \quad \boldsymbol{V} = \left(\boldsymbol{V}_1 \boldsymbol{V}_2 \tilde{\boldsymbol{V}}_1 \tilde{\boldsymbol{V}}_2\right),$$

where $\boldsymbol{D}$ and $\boldsymbol{V}$ are $\mathbb{R}^{4N\times 4N}$ diagonal and orthogonal matrices correspondingly. This means that the $\mathbb{R}^{4N\times N}$ matrices $\boldsymbol{V}_1, \boldsymbol{V}_2, \tilde{\boldsymbol{V}}_1, \tilde{\boldsymbol{V}}_2$ are also orthogonal to each other. Then one can deduce:

$$\boldsymbol{L}_{ij} = \boldsymbol{V}_i^\top \boldsymbol{D} \boldsymbol{V}_j, \quad \text{and} \quad \boldsymbol{M}_{ij} = \tilde{\boldsymbol{V}}_i^\top \boldsymbol{D} \tilde{\boldsymbol{V}}_j \quad \forall i, j = 1, 2\,.$$

In AKD setting only agent 1 has labeled data. The solution of learning from the dataset $\mathcal{D}_1$ evaluated at set $\mathcal{X}_2$ is the following:

$$g_1^1(\mathcal{X}_2) = \boldsymbol{L}_{21}(c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1}\boldsymbol{y}_1 = \boldsymbol{V}_2^\top((\boldsymbol{V}_1^\top(c\boldsymbol{I}+\boldsymbol{D})\boldsymbol{V}_1)^{-1}\boldsymbol{V}_1^\top\boldsymbol{D})^\top\boldsymbol{y}_1\,. \tag{10}$$

Let us introduce notation:

$$\boldsymbol{P}_1 = \boldsymbol{V}_1(\boldsymbol{V}_1^\top(c\boldsymbol{I}+\boldsymbol{D})\boldsymbol{V}_1)^{-1}\boldsymbol{V}_1^\top(c\boldsymbol{I}+\boldsymbol{D}) \quad \text{and} \quad \tilde{\boldsymbol{P}}_2 = \tilde{\boldsymbol{V}}_2(\tilde{\boldsymbol{V}}_2^\top(c\boldsymbol{I}+\boldsymbol{D})\tilde{\boldsymbol{V}}_2)^{-1}\tilde{\boldsymbol{V}}_2^\top(c\boldsymbol{I}+\boldsymbol{D})\,.$$

These are well known oblique (weighted) projection matrices on the subspaces spanned by the columns of matrices $\boldsymbol{V}_1$ and $\tilde{\boldsymbol{V}}_2$ correspondingly, where the scalar product is defined with Gram matrix $\boldsymbol{G} = c\boldsymbol{I} + \boldsymbol{D}$. Similarly one can define $\boldsymbol{P}_2$ and $\tilde{\boldsymbol{P}}_1$.

Given the introduced notation and using the fact that $\boldsymbol{V}_2^\top\boldsymbol{V}_1 = \boldsymbol{0}$, we can rewrite equation 10 as:

$$g_1^1(\mathcal{X}_2) = \boldsymbol{V}_2^\top \boldsymbol{P}_1^\top \boldsymbol{V}_1 \boldsymbol{y}_1\,. \tag{11}$$

Similarly, given $\mathcal{X}_1 \subseteq \mathcal{D}_1$ and agent 1 learned from $\hat{\mathcal{Y}}_2 = g_1^1(\mathcal{X}_2)$, inferred prediction $\hat{\mathcal{Y}}_1 = h_1(\mathcal{X}_1)$ by agent 2 can be written as:

$$h_1(\mathcal{X}_1) = \tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{P}}_2^\top \tilde{\boldsymbol{V}}_2 \boldsymbol{V}_2^\top \boldsymbol{P}_1^\top \boldsymbol{z}_1, \quad \text{where} \quad \boldsymbol{z}_1 = \boldsymbol{V}_1 \boldsymbol{y}_1\,.$$

At this point we need to introduce additional notation:

$$\boldsymbol{C}_1 = \boldsymbol{V}_1 \tilde{\boldsymbol{V}}_1^\top \quad \text{and} \quad \tilde{\boldsymbol{C}}_2 = \tilde{\boldsymbol{V}}_2 \boldsymbol{V}_2^\top\,.$$

These are matrices of contraction linear mappings.

One can now repeat the whole process again with replacement $\mathcal{Y}_1$ by $\hat{\mathcal{Y}}_1$. The predictions by agent 1 and agent 2 after such t rounds of AKD are:

$$g_t^1(\mathcal{X}_2) = \boldsymbol{V}_2^\top \boldsymbol{P}_1^\top \left(\boldsymbol{C}_1 \tilde{\boldsymbol{P}}_2^\top \tilde{\boldsymbol{C}}_2 \boldsymbol{P}_1^\top\right)^t \boldsymbol{z}_1 \quad \text{and} \quad h_t(\mathcal{X}_1) = \tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{P}}_2^\top \tilde{\boldsymbol{C}}_2 \boldsymbol{P}_1^\top \left(\boldsymbol{C}_1 \tilde{\boldsymbol{P}}_2^\top \tilde{\boldsymbol{C}}_2 \boldsymbol{P}_1^\top\right)^t \boldsymbol{z}_1\,. \tag{12}$$

If one considers the case where agents have the same kernel $u_1 = u_2 = u$ then one should 'remove all tildas' in the above expressions and the obtained expressions are applied. This means $\boldsymbol{M} = \boldsymbol{L}$, $\boldsymbol{V}_1 = \tilde{\boldsymbol{V}}_1$ and $\boldsymbol{V}_2 = \tilde{\boldsymbol{V}}_2$. Crucial changes in this case are $\boldsymbol{C}_1 = \tilde{\boldsymbol{C}}_2 \to \boldsymbol{I}$ and $\left(\boldsymbol{C}_1 \tilde{\boldsymbol{P}}_2^\top \tilde{\boldsymbol{C}}_2 \boldsymbol{P}_1^\top\right)^t \to \left(\boldsymbol{P}_2^\top \boldsymbol{P}_1^\top\right)^t$. The matrix $\left(\boldsymbol{P}_2^\top \boldsymbol{P}_1^\top\right)^t$ is an alternating projection Boyd & Dattorro (2003) operator after t steps. Given 2 closed convex sets, alternating projection algorithm in the limit finds a point in the intersection of these sets, provided they intersect Boyd & Dattorro (2003); Lewis et al. (2007). In our case matrices operators $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ project onto linear spaces spanned by columns of matrices $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ correspondingly. These are orthogonal linear subspaces, hence the unique point of their intersection is the origin point $\boldsymbol{0}$. This means that $\left(\boldsymbol{P}_2^\top \boldsymbol{P}_1^\top\right)^t \xrightarrow[t\to\infty]{} \boldsymbol{0}$ and in the limit of such AKD procedure both agents predict 0 for any data point.

Speed of degradation. The speed of convergence for the alternating projections algorithm is known and defined by the minimal angle ϕ between corresponding sets Aronszajn (1950) (only non-zero elements from sets one has to consider):

$$||\left(\boldsymbol{P}_2\boldsymbol{P}_1\right)^t \boldsymbol{v}|| \leq (\cos(\phi))^{2t-1}||\boldsymbol{v}|| = (\cos(\phi))^{2t-1} \quad \text{as} \quad ||\boldsymbol{v}|| = 1\,,$$

where $\boldsymbol{v}$ is one of the columns of matrix $\boldsymbol{V}_1$. In our case we can write the expression for the cosine:

$$\cos(\phi) = \max_{\boldsymbol{v}_1,\boldsymbol{v}_2}\left(\frac{|\boldsymbol{v}_1^\top(c\boldsymbol{I}+\boldsymbol{D})\boldsymbol{v}_2|}{\sqrt{(\boldsymbol{v}_1^\top(c\boldsymbol{I}+\boldsymbol{D})\boldsymbol{v}_1)\cdot(\boldsymbol{v}_2^\top(c\boldsymbol{I}+\boldsymbol{D})\boldsymbol{v}_2)}}\right) = \max_{\boldsymbol{v}_1,\boldsymbol{v}_2}\left(\frac{|\boldsymbol{v}_1^\top\boldsymbol{D}\boldsymbol{v}_2|}{\sqrt{(c+\boldsymbol{v}_1^\top\boldsymbol{D}\boldsymbol{v}_1)\cdot(c+\boldsymbol{v}_2^\top\boldsymbol{D}\boldsymbol{v}_2)}}\right).$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are the vectors from subspaces spanned by columns of matrices $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ correspondingly. Hence the speed of convergence depends on the elements of the matrix $\boldsymbol{L}_{12}$. Intuitively these elements play the role of the measure of 'closeness' between data points. This means that the 'closer' points of sets $\mathcal{X}_1$ and $\mathcal{X}_2$ the higher absolute values of elements in matrix $\boldsymbol{L}_{12}$. Moreover, we see inversely proportional dependence on the regularization constant c. All these sums up in the following proposition which is the formal version of the proposition IV:

Proposition VIII (Formal)**.** *The rate of convergence of $g_t^1(\boldsymbol{x})$ to 0 gets faster if:*

- *larger regularization constant c is used during the training,*
- *smaller the eigenvalues of the matrix $\mathbf{V}_1\tilde{\boldsymbol{V}}_1^\top$, or*
- *smaller absolute value of non-diagonal block $\boldsymbol{L}_{12}$*

B Alternating KD without regularization

Before we saw that models of both agents degrade if one uses regularization. A natural question to ask if models will degrade in the lack of regularization. Consider the problem (1) with $c \to 0$, so the regularization term cancels out. In the general case, there are many possible solutions as kernel matrix $\boldsymbol{K}$ may have 0 eigenvalues. Motivated by the fact that Stochastic Gradient Descent (SGD) tends to find the solution with minimal norm (Wilson et al., 2017), we propose to analyze the minimal norm solution in the problem of alternating knowledge distillation with 2 agents each with private dataset. Mainly, the agent I solution evaluated at set $\mathcal{X}_2$ with all the above notation can be written as:

$$g_1^\star(\mathcal{X}_2) = \boldsymbol{K}_{21}\boldsymbol{K}_{11}^\dagger\boldsymbol{y}_1 = \boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1(\boldsymbol{V}_1^\top\boldsymbol{D}\boldsymbol{V}_1)^\dagger\boldsymbol{y}_1, \tag{13}$$

where $\dagger$ stands for pseudoinverse.
In this section let us consider 3 possible settings one can have:

1. Self-distillation: The datasets and models of both agents are the same.
2. Distillation with $\boldsymbol{K} > 0$: The datasets of both agents are different and private. Kernel matrix $\boldsymbol{K}$ is positive definite.
3. Distillation with $\boldsymbol{K} \geq 0$: The datasets of both agents are different and private. Kernel matrix $\boldsymbol{K}$ is positive semi-definite.

Self-distillation Given the dataset $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$, the solution of the supervised learning with evaluation at any $\boldsymbol{x} \in \mathbb{R}^d$ is:

$$g_1^\star(\boldsymbol{x}) = \boldsymbol{l}_{\boldsymbol{x}}^\top(\boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V})^\dagger\boldsymbol{y}.$$

Then the expression for the self-distillation step with evaluation at any $\boldsymbol{x} \in \mathbb{R}^d$ is:

$$g_2^\star(\boldsymbol{x}) = \boldsymbol{l}_{\boldsymbol{x}}^\top(\boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V})^\dagger\boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V}(\boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V})^\dagger\boldsymbol{y} = \boldsymbol{l}_{\boldsymbol{x}}^\top(\boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V})^\dagger\boldsymbol{y} = g_1^\star(\boldsymbol{x}),$$

where property of pseudoinverse matrix was used: $\boldsymbol{A}^\dagger\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{A}^\dagger$.
That is, self-distillation round does not change obtained model. One can repeat self-distillation step t times and there is no change in the model:

$$g_{t+1}^\star(\boldsymbol{x}) = \boldsymbol{l}_{\boldsymbol{x}}^\top(\boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V})^\dagger\boldsymbol{y} = g_1^\star(\boldsymbol{x}),$$

Hence in the no regularization setting self-distillation step does not give any change in the obtained model.

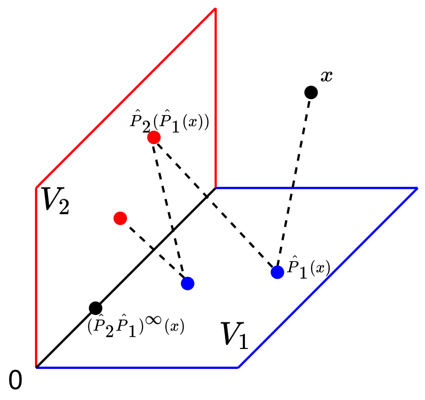


Figure 7: Illustrative projection 3D

Distillation with $\boldsymbol{K} > 0$ In this setting we have the following identity $\boldsymbol{K}^\dagger = \boldsymbol{K}^{-1}$ and results are quite similar to the setting with regularization. But now the Gram matrix of scalar product in linear space is $\boldsymbol{D}$ instead of $c\boldsymbol{I} + \boldsymbol{D}$ in the regularized setting. By assumption on matrix $\boldsymbol{K}$ it follows that $\boldsymbol{D}$ is of full rank and the alternating projection algorithm converges to the origin point $\mathbf{0}$. Therefore in the limit of AKD steps predictions by models of two agents will degrade towards $\mathbf{0}$.

Distillation with $\boldsymbol{K} \geq 0$ In this setting kernel matrix $\boldsymbol{K}$ has at least one 0 eigenvalue and we take for the analysis the minimal norm solution (13). We can rewrite this solution:

$$g_1^\star(\mathcal{X}_2) = \boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger \boldsymbol{y}_1 = \boldsymbol{V}_2^\top \hat{\boldsymbol{P}}_1^\top \boldsymbol{z}_1,$$

where $\hat{\boldsymbol{P}}_1 = \boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger \boldsymbol{V}_1^\top \boldsymbol{D}$ and $\boldsymbol{z}_1 = \boldsymbol{V}_1\boldsymbol{y}_1$.

One can notice the following projection properties of matrix $\hat{\boldsymbol{P}}_1$:

$$\hat{\boldsymbol{P}}_1^2 = \hat{\boldsymbol{P}}_1 \quad \text{and} \quad \hat{\boldsymbol{P}}_1\boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger = \boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger .$$

That is, $\hat{\boldsymbol{P}}_1$ is a projection matrix with eigenspace spanned by columns of the matrix $\boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger$. Similarly, one can define $\hat{\boldsymbol{P}}_2$. Then the solution for the first AKD step evaluated at set $\mathcal{X}_1$ is:

$$g_2^\star(\mathcal{X}_1) = \boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_2(\boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_2)^\dagger \boldsymbol{V}_2^\top \hat{\boldsymbol{P}}_1^\top \boldsymbol{z}_1 = \boldsymbol{V}_1^\top \hat{\boldsymbol{P}}_2^\top \hat{\boldsymbol{P}}_1^\top \boldsymbol{z}_1,$$

and after t rounds we obtain for agent I and agent II correspondingly:

$$g_{2t-1}^\star(\mathcal{X}_2) = \boldsymbol{V}_2^\top \hat{\boldsymbol{P}}_1^\top (\hat{\boldsymbol{P}}_2^\top \hat{\boldsymbol{P}}_1^\top)^{t-1}\boldsymbol{z}_1 \quad \text{and} \quad g_{2t}^\star(\mathcal{X}_1) = \boldsymbol{V}_1^\top (\hat{\boldsymbol{P}}_2^\top \hat{\boldsymbol{P}}_1^\top)^t \boldsymbol{z}_1.$$

In the limit of the distillation rounds operator $(\hat{\boldsymbol{P}}_2\hat{\boldsymbol{P}}_1)^t$ tends to the projection on the intersection of 2 subspaces spanned by the columns of matrices $\boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger$ and $\boldsymbol{V}_2(\boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_2)^\dagger$. We should highlight that in general, the intersection set in this case consists not only from the origin point $\mathbf{0}$ but some rays that lie simultaneously in the eigenspaces of both projectors $\hat{\boldsymbol{P}}_1$ and $\hat{\boldsymbol{P}}_2$. The illustrative example is shown in the Fig. 7. We perform the alternating projection algorithm between orthogonal linear spaces that have a ray in the intersection and we converge to some non-zero point that belongs to this ray. The intersection of eigenspaces of both projectors $\hat{\boldsymbol{P}}_1$ and $\hat{\boldsymbol{P}}_2$ is the set of points $\boldsymbol{x} \in \text{Span}(\boldsymbol{V}_1(\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1)^\dagger)$ s.t. $\hat{\boldsymbol{P}}_1\hat{\boldsymbol{P}}_2\boldsymbol{x} = \boldsymbol{x}$.

C Averaged KD

In this section we present the detailed analysis of the algorithm presented in the section 4.2 keeping the notation of the section A. As a reminder, models of both agents after round t of AvgKD algorithm are as follows:

$$g_t^1(\boldsymbol{X}_2) = \frac{1}{2}\boldsymbol{L}_{21}(c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1}(\boldsymbol{y}_1 + g_{t-1}^2(\boldsymbol{X}_1)) = \frac{1}{2}\boldsymbol{V}_2^\top \boldsymbol{P}_1^\top(\boldsymbol{z}_1 + g_{t-1}^2),$$

$$g_t^2(\boldsymbol{X}_1) = \frac{1}{2}\boldsymbol{M}_{12}(c\boldsymbol{I} + \boldsymbol{M}_{22})^{-1}(g_{t-1}^1(\boldsymbol{X}_2) + \boldsymbol{y}_2) = \frac{1}{2}\tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{P}}_2^\top(g_{t-1}^1 + \boldsymbol{z}_2).$$

where

$$g^1_{t-1} = \frac{1}{2}(c\boldsymbol{I} + \boldsymbol{D})\boldsymbol{V}_1(c\boldsymbol{I} + \boldsymbol{L}_{11})^{-1}(\boldsymbol{y}_1 + g^2_{t-2}(\boldsymbol{X}_1)), \quad \forall t \geq 3 \tag{14}$$

$$g^2_{t-1} = \frac{1}{2}(c\boldsymbol{I} + \boldsymbol{D})\tilde{\boldsymbol{V}}_2(c\boldsymbol{I} + \boldsymbol{M}_{22})^{-1}(g^1_{t-2}(\boldsymbol{X}_2) + \boldsymbol{y}_2), \quad \forall t \geq 3 \tag{15}$$

$$t = 2: \quad g^1_1 = \boldsymbol{P}_1^\top \boldsymbol{z}_1 \quad \text{and} \quad g^2_1 = \tilde{\boldsymbol{P}}_2^\top \boldsymbol{z}_2, \quad \text{where} \quad \boldsymbol{z}_1 = \boldsymbol{V}_1\boldsymbol{y}_1, \quad \boldsymbol{z}_2 = \tilde{\boldsymbol{V}}_2\boldsymbol{y}_2\,. \tag{16}$$

The illustration of this process is presented in the Fig. 2a.

Consider the sequence of solutions for the agent 1 with evaluation at $\boldsymbol{X}_2$:

- Supervised Learning:
$$g^1_1(\boldsymbol{X}_2) = \boldsymbol{V}_2^\top \boldsymbol{P}_1^\top \boldsymbol{z}_1$$
- 1st round of KD:
$$g^1_2(\boldsymbol{X}_2) = \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top}{2}(\boldsymbol{z}_1 + \boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top \boldsymbol{z}_2)$$
- 2nd round of KD:
$$g^1_3(\boldsymbol{X}_2) = \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top}{2}(\boldsymbol{z}_1 + \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{2}\boldsymbol{z}_2 + \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top}{2}\boldsymbol{z}_1)$$
- 3rd round of KD:
$$g^1_4(\boldsymbol{X}_2) = \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top}{2}(\boldsymbol{z}_1 + \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{2}\boldsymbol{z}_2 + \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top}{4}\boldsymbol{z}_1 + \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4}\boldsymbol{z}_2)$$
- t-th round of KD:
$$g^1_{t+1}(\boldsymbol{X}_2) = \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top}{2}(\sum_i^t(\frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top}{4})^i)\boldsymbol{z}_1 + \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4}(\sum_i^{t-1}(\frac{\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4})^i)\boldsymbol{z}_2$$
- The limit of KD steps:
$$g^1_\infty(\boldsymbol{X}_2) = \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top}{2}(\boldsymbol{I} - \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top}{4})^\dagger\boldsymbol{z}_1 + \boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4}(\boldsymbol{I} - \frac{\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4})^\dagger\boldsymbol{z}_2,$$
where $\dagger$ stands for pseudoinverse.

Let us analyze the limit solution and consider the first term of its expression. One can deduce the following identity: [1]

$$\boldsymbol{V}_2^\top \frac{\tilde{\boldsymbol{P}}_1^\top}{2}(\boldsymbol{I} - \frac{\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top}{4})^\dagger\boldsymbol{z}_1 = \tag{17}$$

$$\frac{\boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1}{2}(c\boldsymbol{I} + \boldsymbol{V}_1^\top\boldsymbol{D}\boldsymbol{V}_1 - \frac{\tilde{\boldsymbol{V}}_1^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2}(c\boldsymbol{I} + \tilde{\boldsymbol{V}}_2^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2)^{-1}\frac{\boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1}{2})^\dagger\boldsymbol{y}_1 \tag{18}$$

In a similar manner, we can deal with the second term:

$$\boldsymbol{V}_2^\top \frac{\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4}(\boldsymbol{I} - \frac{\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top}{4})^\dagger\boldsymbol{z}_2 =$$

$$\frac{\boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1}{2}(c\boldsymbol{I} + \boldsymbol{V}_1^\top\boldsymbol{D}\boldsymbol{V}_1 - \frac{\tilde{\boldsymbol{V}}_1^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2}(c\boldsymbol{I} + \tilde{\boldsymbol{V}}_2^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2)^{-1}\frac{\boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1}{2})^\dagger\frac{\tilde{\boldsymbol{V}}_1^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2}(c\boldsymbol{I} + \tilde{\boldsymbol{V}}_2^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2)^{-1}\boldsymbol{y}_2$$

Now, we notice Schur complement expression in the equation (18):

$$(c\boldsymbol{I} + \boldsymbol{V}_1^\top\boldsymbol{D}\boldsymbol{V}_1 - \frac{\tilde{\boldsymbol{V}}_1^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2}(c\boldsymbol{I} + \tilde{\boldsymbol{V}}_2^\top\boldsymbol{D}\tilde{\boldsymbol{V}}_2)^{-1}\frac{\boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1}{2})\,.$$

[1]In case of $c = 0$, the whole analysis can be repeated by replacing inverse sign with $\dagger$ sign and using the following fact for positive semidefinite matrices Zhang (2006):

$$\boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1(\boldsymbol{V}_1^\top\boldsymbol{D}\boldsymbol{V}_1)^\dagger(\boldsymbol{V}_1^\top\boldsymbol{D}\boldsymbol{V}_1) = \boldsymbol{V}_2^\top\boldsymbol{D}\boldsymbol{V}_1\,.$$

This means that we can consider the following problem:

$$\begin{pmatrix} \boldsymbol{L}_{11}+c\boldsymbol{I} & \frac{\boldsymbol{M}_{12}}{2} \\ \frac{\boldsymbol{L}_{21}}{2} & \boldsymbol{M}_{22}+c\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1+c\boldsymbol{I} & \frac{\tilde{\boldsymbol{V}}_1^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2} \\ \frac{\boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_1}{2} & \tilde{\boldsymbol{V}}_2^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2+c\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ -\boldsymbol{y}_2 \end{pmatrix}, \quad (19)$$

and derive that $g^1_\infty(\boldsymbol{X}_2) = \frac{\boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_1}{2}\boldsymbol{\beta}_1$ and $g^2_\infty(\boldsymbol{X}_1) = -\frac{\tilde{\boldsymbol{V}}_1^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2}\boldsymbol{\beta}_2$, where $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2$ are defined as the solution to the problem (19). Given this, we can derive the following:

$$\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1\boldsymbol{\beta}_1 + \frac{\tilde{\boldsymbol{V}}_1^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2}{2}\boldsymbol{\beta}_2 = \boldsymbol{y}_1 - c\boldsymbol{\beta}_1 = 2g^1_\infty(\mathcal{X}_1) - g^2_\infty(\mathcal{X}_1), \quad (20)$$

$$-\frac{\boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_1}{2}\boldsymbol{\beta}_1 - \tilde{\boldsymbol{V}}_2^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2\boldsymbol{\beta}_2 = \boldsymbol{y}_2 + c\boldsymbol{\beta}_2 = 2g^2_\infty(\boldsymbol{X}_2) - g^1_\infty(\boldsymbol{X}_2). \quad (21)$$

As one can see, there is a strong relation between the limit KD solutions and the solution of a linear system of equations with modified matrix $\boldsymbol{K}$ and right-hand side. Mainly, we take the kernel matrix and divide its non-diagonal blocks by 2, which intuitively shows that our final model accounts for the reduction of the 'closeness' between datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. And on the right-hand side, we see $-\boldsymbol{y}_2$ instead of $\boldsymbol{y}_2$ which is quite a 'artificial' effect. Overall these results in the fact that both limit solutions (for each agent) do not give a ground truth prediction for $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ individually, which one can see from equation (20) with $c = 0$. That is, we need combine the predictions of both agents in a specific way to get ground truth labels for datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. Moreover, the way we combine the solutions differs between datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that one can see from comparison of right-hand sides of expressions (20) and (21). Given all the above, to predict optimal labels we need to change we way we combine models of agents in dependence on a dataset, but an usual desire is to have one model that predicts ground truth labels for at least both training datasets.

To obtain the expressions for the case of identical models one should 'remove all tildas' in the above expressions and by setting $\boldsymbol{V}_1 = \tilde{\boldsymbol{V}}_1$, $\boldsymbol{V}_2 = \tilde{\boldsymbol{V}}_2$, $\boldsymbol{C}_1 = \tilde{\boldsymbol{C}}_2 \to \boldsymbol{I}$ and $(\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top)^n \to (\boldsymbol{P}_2^\top\boldsymbol{P}_1^\top)^n$

D ENSEMBLED KD

One can consider the following problem:

$$\begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{L}_{21} & \boldsymbol{M}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1 & \tilde{\boldsymbol{V}}_1^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2 \\ \boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_1 & \tilde{\boldsymbol{V}}_2^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}, \quad (22)$$

with the following identities:

$$\boldsymbol{V}_1^\top \boldsymbol{D}\boldsymbol{V}_1\boldsymbol{\beta}_1 + \tilde{\boldsymbol{V}}_1^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2\boldsymbol{\beta}_2 = \boldsymbol{y}_1 \quad \text{and} \quad \boldsymbol{V}_2^\top \boldsymbol{D}\boldsymbol{V}_1\boldsymbol{\beta}_1 + \tilde{\boldsymbol{V}}_2^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2\boldsymbol{\beta}_2 = \boldsymbol{y}_2\,. \quad (23)$$

One can find $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2$ and deduce the following prediction by the model associated with the system (22) for $i = 1, 2$:

$$\begin{gathered} \boldsymbol{V}_i^\top \boldsymbol{D}\boldsymbol{V}_1\boldsymbol{\beta}_1 + \tilde{\boldsymbol{V}}_i^\top \boldsymbol{D}\tilde{\boldsymbol{V}}_2\boldsymbol{\beta}_2 = \\ \boldsymbol{V}_i^\top\boldsymbol{P}_1^\top(\boldsymbol{I} - \boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top)^\dagger\boldsymbol{z}_1 - \boldsymbol{V}_i^\top\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top(\boldsymbol{I} - \tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top)^\dagger\boldsymbol{z}_2 + \\ \tilde{\boldsymbol{V}}_i^\top\tilde{\boldsymbol{P}}_2^\top(\boldsymbol{I} - \tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top)^\dagger\boldsymbol{z}_2 - \tilde{\boldsymbol{V}}_i^\top\boldsymbol{P}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top(\boldsymbol{I} - \boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top)^\dagger\boldsymbol{z}_1 = \\ \boldsymbol{V}_i^\top\boldsymbol{P}_1^\top\sum_{t=0}^{\infty}(\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top)^t\boldsymbol{z}_1 - \boldsymbol{V}_i^\top\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\sum_{t=0}^{\infty}(\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top)^t\boldsymbol{z}_2 + \\ \tilde{\boldsymbol{V}}_i^\top\boldsymbol{P}_2^\top\sum_{t=0}^{\infty}(\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top)^t\boldsymbol{z}_2 - \tilde{\boldsymbol{V}}_i^\top\boldsymbol{P}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top\sum_{t=0}^{\infty}(\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top)^t\boldsymbol{z}_1. \end{gathered}$$

From this one can easily deduce (23) which means that for datasets of both agents this model predicts ground truth labels. To obtain the expressions for the case of identical models one should 'remove all tildas' in all the above expressions and by setting $\boldsymbol{V}_1 = \tilde{\boldsymbol{V}}_1$, $\boldsymbol{V}_2 = \tilde{\boldsymbol{V}}_2$, $\boldsymbol{C}_1 = \tilde{\boldsymbol{C}}_2 \to \boldsymbol{I}$ and $(\boldsymbol{C}_1\tilde{\boldsymbol{P}}_2^\top\tilde{\boldsymbol{C}}_2\boldsymbol{P}_1^\top)^t \to (\boldsymbol{P}_2^\top\boldsymbol{P}_1^\top)^t$

The last question is how one can construct the scheme of iterative KD rounds to obtain the above expression for the limit model. From the form of the prediction, we conclude that one should use models obtained in the process of AKD. There are many possible schemes how one can combine these models to obtain the desired result. One of the simplest possibilities is presented in the section 5.2.

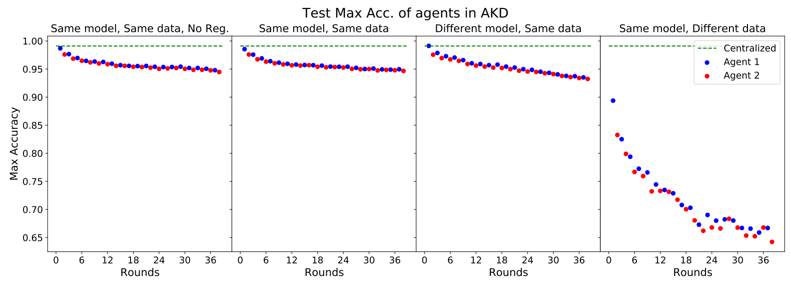


Figure 8: AKD Test CE, MNIST

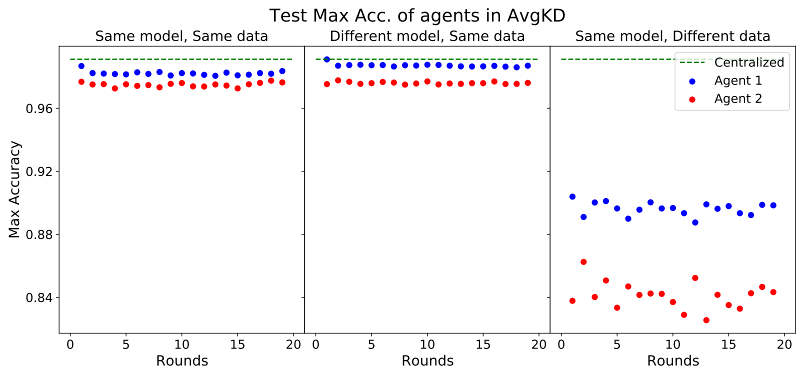


Figure 9: AvgKD Test CE, MNIST

E Additional experiments

E.1 Cross-Entropy

In the Fig. 8 and 9 one can see the results of AKD and AvgKD schemes correspondingly for CE loss.

E.2 CIFAR10

In this section we present the results for CIFAR10 dataset in the Fig. 10 and 11.

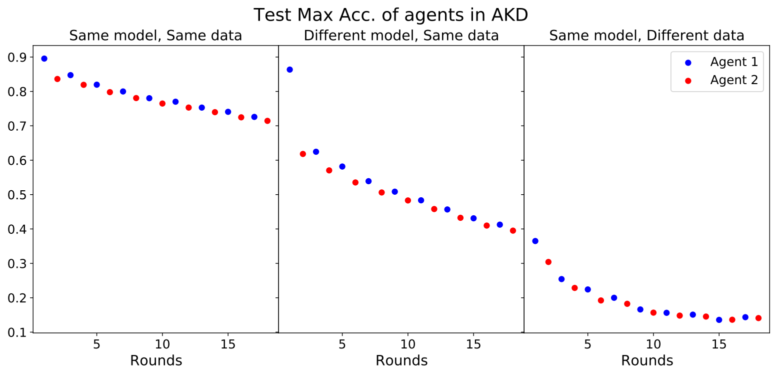


Figure 10: AKD Test MSE, CIFAR10

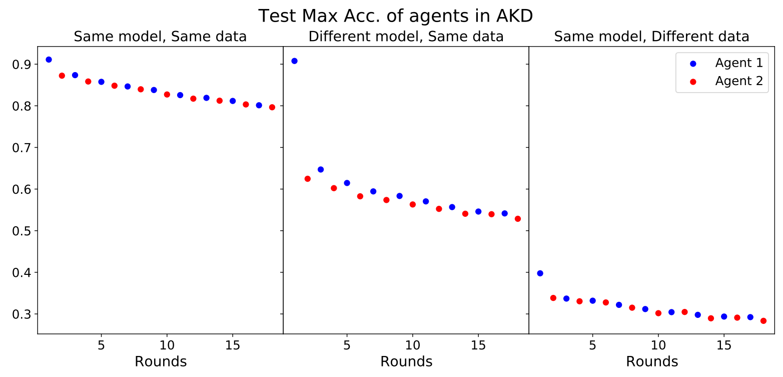


Figure 11: AKD Test CE, CIFAR10