Distributed Double Machine Learning with a Serverless Architecture

Malte S. Kurz
University of Hamburg
Hamburg, Germany
malte.simon.kurz@uni-hamburg.de

ABSTRACT

This paper explores serverless cloud computing for double machine learning. Being based on repeated cross-fitting, double machine learning is particularly well suited to exploit the high level of parallelism achievable with serverless computing. It allows to get fast on-demand estimations without additional cloud maintenance effort. We provide a prototype Python implementation `DoubleML-Serverless` for the estimation of double machine learning models with the serverless computing platform AWS Lambda and demonstrate its utility with a case study analyzing estimation times and costs.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Machine Learning; Serverless Computing; Function-as-a-Service (FaaS); Distributed Computing; AWS Lambda; Causal Machine Learning

1 INTRODUCTION

Double machine learning (DML) models [19] are becoming increasingly popular among statisticians, econometricians and data scientists with numerous methodological extensions [8, 18, 21, 27–29, 33, 34, 37] and applications in areas like finance [22], COVID-19 research [20, 39] or economics [30, 38]. The DML models allow researchers to exploit the excellent prediction power of machine learning algorithms in a valid statistical framework for estimation and inference on causal parameters. Recently, the Python and R packages `DoubleML` with a flexible object-oriented structure for estimating double machine learning models have been published [6, 7].

Serverless cloud computing is predicted to be the dominating and default architecture of cloud computing in the coming decade (Berkley View on Serverless Computing [26]) and is becoming increasingly adopted in the industry and by researchers. Its Function as a Service (FaaS) paradigm lowers the entry bar to cloud computing technologies as the cloud providers are responsible for almost every operational and maintenance task. A key advantage of serverless computing is the high elasticity in terms of an automated on-demand scaling depending on the actual amount of computing requests. A second key advantage of serverless computing is the pricing model: Only actually used resources are charged without provisioning costs.

The management of computing clusters is usually not part of the daily business of econometricians or data scientists using DML for data analysis or in applied research. Nevertheless there is demand for a high level of scalability to speed up the estimation of models like DML in interactive data analysis tasks. In our experience, econometricians or data scientists who consider using cloud computing resources often want to achieve goals like the following:

- A high level of parallelism.
- A "cloud button": Easy deployment and if possible no ongoing maintenance tasks for the user.
- A high level of elasticity: On-demand availability of a high level of parallelism, pay-per-request and ideally no costs when the systems are idle.

The goal of this paper is to explore to what extent such goals are achievable with serverless cloud computing and we put special focus on DML models as an application. Our study is based on AWS Lambda and we made our prototype implementation `DoubleML-Serverless` publicly available.[1] We demonstrate the functionalities of the prototype with an experiment where we analyze estimation times and costs with different settings.

The rest of the paper is organized as follows: Introductions to serverless computing and double machine learning are given in Sections 2 and 3. The prototype implementation `DoubleML-Serverless` is described in Section 4. Section 5 presents our experiment setup and results. In Section 6 we discuss our prototype implementation and give an outlook to potential future extensions. Section 7 concludes the paper.

2 SERVERLESS COMPUTING

A core principle of serverless computing is that the user just writes a cloud function, often in a high-level programming language like Python, and all the server provisioning and administration is done

---

[1]GitHub: https://github.com/DoubleML/doubleml-serverless and AWS Serverless Application Repository: https://serverlessrepo.aws.amazon.com/applications/eu-central-1/839779594349/doubleml-serverless.

by the cloud provider. These serverless cloud function offerings are often called Function as a Service (FaaS), because the user basically only specifies the function code to be executed and declares which events should trigger such function calls. There is especially no need for ex-ante provisioning of computing resources. It is in the hand of the cloud provider to automatically scale up resources depending on the number of requests sent to the FaaS. This is one of the key differences in comparison to a classical cloud server, where the user ex-ante needs to decide which requirements best match the upcoming computing tasks.

General discussions of serverless computing, recent developments and challenges can be found in [9, 23, 26, 32, 41, 42]. Besides that, serverless computing is getting more and more adopted for various machine learning tasks, like for example to serve deep learning models [14, 24, 40] or more generally for ML model training and hyperparameter tuning [15, 16, 43].

Another core principle of serverless computing is the pricing model. The billing is usually done proportionally to the actually used resources and not proportionally to resources provisioned. In case of AWS Lambda it is proportional to the execution time and very fine grained as the duration billing granularity was recently lowered to per millisecond billing [2].

When using AWS Lambda there is one key parameter set by the user, which is the memory available to the function at runtime. AWS Lambda also scales other resources like CPU power proportionally to the allocated memory. In the past the maximum memory allocatable was regularly increased and recently there was an significant extension from a maximum of 3 GB to 10 GB [3]. According to AWS this translates to a maximum of 6 vCPUs accessible in a single FaaS request [3]. By its nature the enormous elasticity of serverless computing platforms comes at the cost of rather strict resource limits for a single request. When using AWS Lambda among others the maximum runtime is 15 minutes. However, the recent updates make serverless computing increasingly attractive for computationally intense tasks like machine learning.

3 A BRIEF INTRODUCTION TO DOUBLE MACHINE LEARNING

Double machine learning (DML) was developed in a series of papers [10–12] and introduced as a general framework in [19]. The application of DML for model classes like the partially linear regression model, the partially linear instrumental variable model, the interactive regression model and the interactive instrumental variable model is discussed in [19]. Recently the DML framework and related techniques have been extended to numerous model classes like for example reinforcement learning [27, 34], transformation models [28], generalized additive models [8], continuous treatment effects [21, 37], dynamic treatment effects [33], Gaussian graphical models [29], difference-in-differences models [18] and many more. In these applications of DML, one is usually interested in statistical inference for a causal parameter θ_0. The DML framework makes it possible to obtain valid statistical inference for θ_0 while exploiting the excellent prediction quality of machine learning methods for estimating nuisance functions denoted as η_0.

As an example, we consider the partially linear regression (PLR) model as studied by [36]

$$Y = D\theta_0 + g_0(X) + U, \qquad \mathbb{E}(U|X, D) = 0, \qquad (1)$$
$$D = m_0(X) + V, \qquad \mathbb{E}(V|X) = 0, \qquad (2)$$

with outcome variable Y, treatment/policy variable D and the potentially high-dimensional vector of controls $X := (X_1, \ldots, X_p)$. The causal parameter of interest is θ_0. It measures the average treatment effect of D on Y, if D is conditionally exogenous. The confounding variables X affect D via the function $m_0(X)$ and Y via the function $g_0(X)$. Figure 1 visualizes the interpretation in a causal diagram. The DML framework allows to obtain valid statistical inference for θ_0 while exploiting the excellent prediction quality of machine learning methods when estimating the nuisance functions $\eta_0 = (g_0, m_0)$. In the DML framework the nuisance functions $\eta_0 = (g_0, m_0)$ can be estimated with different ML-methods, e.g., [19] use random forests, regression trees, boosting, lasso, neural networks and ensembles of these methods. Depending on the structural assumptions on η_0, different ML-methods are appropriate.[2]
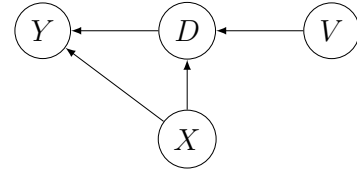
Figure 1: Causal Diagram for the PLR Model (1)–(2).

A key component of the DML framework are so-called Neyman orthogonal score functions $\psi(W; \theta, \eta)$. The score functions identify the causal parameter of interest θ_0 as the unique solution to $\mathbb{E}(\psi(W; \theta_0, \eta_0)) = 0$. Neyman orthogonality of $\psi(W; \theta, \eta)$ with respect to the nuisance functions η guarantees that there are no first-order effects of estimation errors in the nuisance functions on the estimation of the causal parameter θ_0.

A second key component of the DML framework is sample splitting to avoid biases caused by overfitting. The application of repeated cross-fitting is further recommended in [19]. This makes it particularly well suited for a distributed architecture where the computationally intense inference tasks run in parallel. Estimation of typical DML models often requires the estimation and prediction of several hundreds of ML models to approximate nuisance functions in different sample splits. An ambitious goal of a serverless DML implementation would be to achieve that the estimation of the whole DML model with repeated cross-fitting does not take much longer than the estimation of the nuisance functions on a single fold. The enormous elasticity of serverless cloud computing makes such a goal achievable in an on-demand setup with no need to start and maintain a large computing cluster, which is becoming costly if being idle.

The DML algorithm with repeated cross-fitting can be summarized as follows (w.l.o.g. we assume that the number of observations N is divisible by the number of folds K):

---

[2]We refer to [19, Section 3] for a discussion and the formal conditions for the quality of the nuisance estimators.

(1) For each $m \in [M] := \{1, \ldots, M\}$ draw a K-fold random partition $(I_{m,k})_{k \in K}$ of observation indices $[N] := \{1, \ldots, N\}$ of size $n = N/K$. Define $I^c_{m,k} := [N] \setminus I_{m,k}$ and for each k construct a ML estimator

$$\hat{\eta}_{0,k} = \hat{\eta}_0((W_i)_{i \in I^c_{m,k}}).$$

(2) For each sample split, compute an estimate $\tilde{\theta}_{0,m}$ of the causal parameter as the solution to the equation

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{i \in I_{m,k}} \psi(W_i; \tilde{\theta}_{0,m}, \hat{\eta}_{0,k}) = 0.$$

The final estimate for the causal parameter is obtain via aggregation $\tilde{\theta}_0 = \text{Median}((\tilde{\theta}_{0,m})_{m \in [M]})$.

Note that the number of nuisance functions, which need to be estimated with ML methods, depends on the considered model, e.g., for the PLR model we have $L = 2$ nuisance functions $\eta_0 = (g_0, m_0)$. The total number of ML fits is $M \times K \times L$, i.e., one ML estimation in each fold, of each repeated sample splitting and for each nuisance function. For example [19] choose $K = 5$ (or $K = 2$) and $M = 100$, which for the PLR model with $L = 2$ nuisance functions amounts to 1000 (or 400) ML fits or for the partially linear instrumental variable model with $L = 3$ nuisance functions it amounts to 1500 (or 600) ML fits. Note that for the interactive regression models, as considered in [19], even more nuisance functions need to be estimated. As mentioned before, our prototype for serverless DML allows for parallelization of all these 1000 machine learning tasks and therefore potentially speeds up the estimation of DML models by a significant factor. Basically, the estimation time with repeated cross-fitting with five folds and 100 repetitions could be almost reduced to the time needed to estimate a single nuisance function for one fold in a single sample split. Note that we do not require to transfer the estimated ML models for the nuisance functions $\hat{\eta}_{0,k}$, instead it suffices to return the predictions on the test datasets (i.e., for the observations indexed with $i \in I_{m,k}$) to evaluate the score function and solve for the causal parameter $\tilde{\theta}_{0,m}$ in a second step.

Neyman orthogonal score functions for many model classes, like for example the PLR model, can be written as linear functions in the parameter θ, i.e.,

$$\psi(W; \theta; \eta) = \theta \psi_a(W; \eta) + \psi_b(W; \eta).$$

This common property forms the basis for a very general object-oriented implementation of DML models in the Python package `DoubleML` [6], which serves as a basis four our prototype `DoubleML-Serverless`.

4 SERVERLESS DOUBLE MACHINE LEARNING

Similar to `PyWren` [25], our prototype implementation `DoubleML-Serverless` is intended to be used in an interactive fashion: The user runs a Python session on a local machine or server, but at the same time has access to a high level of parallelism with an on-demand and pay-per-request interface for the computationally most

intense tasks during the estimation of DML models.[3] In comparison to `PyWren`, which allows to run more or less arbitrary parallel tasks, like for example map reduce, our implementation is more specialized to the specific use case of DML models. Many cloud providers have serverless FaaS offerings. Our prototype `DoubleML-Serverless` uses AWS Lambda and is developed in Python as an extension of the `DoubleML` package [6].[4]

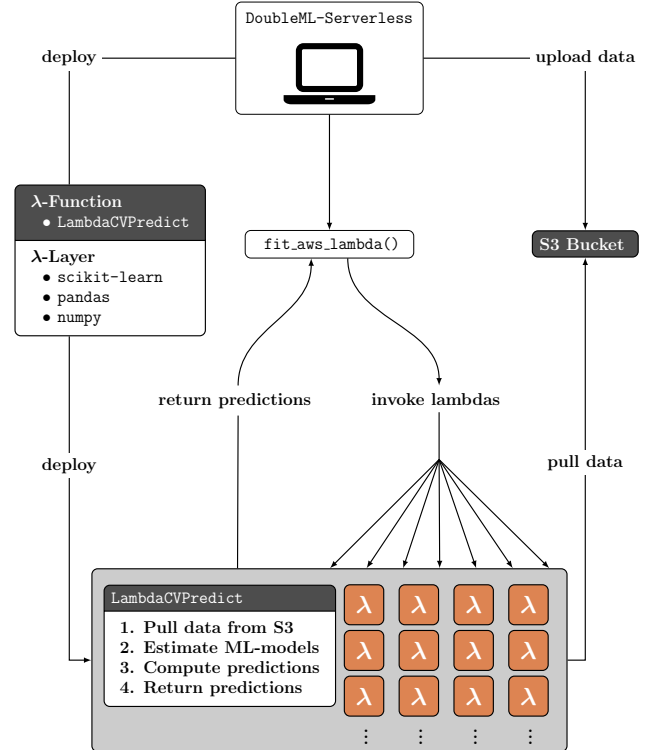### 4.1 The Architecture of DoubleML-Serverless

Figure 2: DoubleML-Serverless: Architecture.

The architecture of `DoubleML-Serverless` is summarized in Figure 2. As data storage we use the AWS S3 object storage. In the `DoubleML-Serverless` package, we implement a `DoubleMLDataS3` class, which serves as a data backend. It is inherited from the `DoubleML` class `DoubleMLData` and primarily extends it by methods to transfer datasets from and to AWS S3. The model classes, like for example `DoubleMLPLRServerless` for the PLR model, extend the corresponding classes from the `DoubleML` package by methods to perform the ML estimation and prediction step on AWS Lambda. In addition to the standard inputs for `DoubleML` model classes, the user needs to provide the name of the deployed lambda function and the AWS region on initialization. Then the DML model can

---

[3]As suggested by an anonymous referee, alternatively a fully serverless version could be implemented using services like AWS Step Functions to organize the serverless workflow.

[4]The prototype is tied to AWS Lambda. Adaptions of the data transfer and the deployment process would be necessary to make it compatible with other serverless platforms.

be estimated with a call to the method `fit_aws_lambda()`. On invocation, each request consists of a reference to the dataset on S3, the nuisance-function-specific names of target variables and confounders and the sample splitting. The lambda function returns the predictions for the corresponding test indices.

4.2 The Level of Scaling

Our prototype implementation `DoubleML-Serverless` offers two different degrees of scaling. Figure 3 visualizes the level of scaling options for the PLR class `DoubleMLPLRServerless`. Per-sample-split scaling is achieved by choosing `scaling = 'n_rep'`. It results in a lambda function invocation for each nuisance function and repeated sample split, i.e., for each blue rectangle in Figure 3. In each such invocation, K machine learning models are estimated and corresponding predictions for the test indices returned. As an alternative one can choose `scaling = 'n_folds * n_rep'` to invoke a separate lambda for each single fold, nuisance function and sample split, i.e., for each orange rectangle in Figure 3.
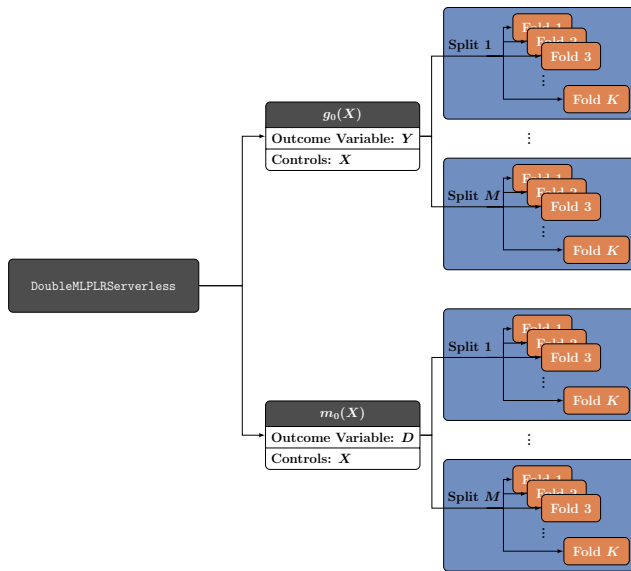

Figure 3: DoubleML-Serverless: Level of Scaling.

If we again consider the above mentioned PLR model with $K = 5$ folds, $M = 100$ splits and $L = 2$ nuisance functions, it means that we either sent $M \times L = 200$ requests or $M \times K \times L = 1000$ requests. Which level of scaling is favorable depends on the individual use case. First of all, the runtime limit of AWS Lambda implies that the per-sample-split scaling cannot be applied if the estimation of K machine learning models takes longer than the maximum runtime, which might be the case, depending on the machine learning approach and the size of the dataset. Furthermore, there is always a cost vs. estimation-time tradeoff which the user controls via the `scaling` parameter and the allocated memory.

4.3 Deployment with AWS SAM

User-friendly deployment of the prototype is achieved with the AWS Serverless Application Model (AWS SAM) for deploying our

FaaS to AWS Lambda. AWS SAM [4] allows for easy deployment of serverless applications to AWS Lambda and is configured via template files. We added an AWS SAM template to our prototype, which deploys the following components (see Figure 2 for a visualization of the architecture):

- A lambda function called `LambdaCVPredict`.
- A layer providing the Python libraries `scikit-learn`, `pandas` and `numpy` together with their dependencies.
- An S3 bucket for the data transfer (can be optionally generated, or an existing bucket is used).
- A role for the execution of the lambda function `LambdaCVPredict` which consists of the AWS-managed `AWSLambdaBasicExecutionRole` policy plus read access to the S3 bucket for data transfer.

`LambdaCVPredict` is the main function being invoked when estimating DML models on AWS Lambda. The main advantage of AWS SAM is that the deployment process is simple with only two calls `sam build` and `sam deploy -guided`. Additionally, based on the same SAM template, even simpler deployment is offered directly from the AWS Serverless Application Repository.[5] The listing in the AWS Serverless Application Repository gives the user almost a "bring me to the cloud"-button for estimating DML models.[6]

5 ESTIMATING DOUBLE MACHINE LEARNING MODELS WITH DOUBLEML-SERVERLESS

To demonstrate our prototype implementation `DoubleML-Serverless` we revisit the Pennsylvania Reemployment Bonus experiment and estimate the effect of provisioning a cash bonus on the unemployment duration as studied in [19].

5.1 Experiment Setup

We consider the previously discussed PLR model (1)–(2). The nuisance functions g_0 and m_0 are estimated using a random forest with 500 regression trees.[7] We choose $K = 5$ folds and $M = 100$ splits.

At invocation, the following information is transferred to `LambdaCVPredict`:

- The name of the outcome variable, e.g., for g_0 the Y column.
- The names of the controls, e.g., for g_0 the X columns.
- The ML model to be estimated, e.g., random forest.
- The set of indices $I_{m,k}$.

In Listing 1 we provide sample code which demonstrates the syntax to estimate the described DML model with `DoubleML-Serverless` for the *bonus* dataset.

Listing 1: Estimation of a Partially Linear Regression (PLR) Model with DoubleML-Serverless.

```
from doubleml.datasets import fetch_bonus
from doubleml_serverless import DoubleMLDataS3, DoubleMLPLRServerless
from sklearn.base import clone
from sklearn.ensemble import RandomForestRegressor
```

---

[5]https://serverlessrepo.aws.amazon.com/applications/eu-central-1/839779594349/doubleml-serverless

[6]From the AWS Serverless Application Repository, the deployment can be done directly in the browser by clicking "Deploy" and following the steps in the AWS Management Console.

[7]In case of a binary treatment variable D, one can also use classifiers to estimate m_0.

```
df_bonus = fetch_bonus('DataFrame')
dml_data_bonus = DoubleMLDataS3(
    'doubleml-serverless-data', 'bonus_data.csv', df_bonus,
    y_col='inuidur1', d_cols='tg',
    x_cols=['female', 'black', 'othrace', 'dep1', 'dep2',
        'q2', 'q3', 'q4', 'q5', 'q6', 'agelt35',
        'agegt54', 'durable', 'lusd', 'husd'])
dml_data_bonus.store_and_upload_to_s3()

ml = RandomForestRegressor(n_estimators = 500, n_jobs=-1)
ml_g = clone(ml)
ml_m = clone(ml)
dml_lambda_plr_bonus = DoubleMLPLRServerless(
    'LambdaCVPredict', 'eu-central-1',
    dml_data_bonus, ml_g, ml_m,
    n_folds=5, n_rep=100)

dml_lambda_plr_bonus.fit_aws_lambda()
```

The FaaS function `LambdaCVPredict` returns predictions which are obtained by estimating the nuisance function based on the training indices $I^c_{m,k}$ and then predictions are computed for all $i \in I_{m,k}$. When all requested predictions have been returned, the score function components for the PLR model at hand are obtained as

$$\psi_a(W_i; \hat{\eta}_0) = -(D_i - \hat{m}_0(X_i))(D_i - \hat{m}_0(X_i)),$$
$$\psi_b(W_i; \hat{\eta}_0) = (Y_i - \hat{g}_0(X_i))(D_i - \hat{m}_0(X_i)).$$

Using the evaluated score function components, we can solve for the parameter estimate

$$\tilde{\theta}_0 = \frac{-\sum_{i=1}^{N} \psi_b(W_i; \hat{\eta}_0)}{\sum_{i=1}^{N} \psi_a(W_i; \hat{\eta}_0)}.$$

Based on the evaluated score function, inference tasks like the computation of standard errors and confidence intervals that build on a multiplier bootstrap approach could be easily done locally using the functionalities of the `DoubleML` package. For further details, we refer to the paper introducing the DML framework [19] and the documentation of the `DoubleML` package [6].[8]

5.2 Timings and Costs

To demonstrate the utility of our prototype `DoubleML-Serverless` we ran a couple of experiments on AWS Lambda with the above stated *bonus* data example. We especially focus on the two different settings for the `scaling` parameter, i.e., `scaling = 'n_rep'` for per-sample-split scaling and `scaling = 'n_folds * n_rep'` for per-fold scaling. With the above mentioned settings ($K = 5$ folds and $M = 100$ splits) this amounts to 200 and 1000 invocations, respectively. Additionally, we also alter the memory available to the function at runtime which also impacts the CPU power, because AWS Lambda scales other resources proportionally to the allocated memory. All experiments are repeated 100 times and the estimation times and costs are visualized with boxplots in Figure 4 and 5.

In Figure 4 we can clearly see that the total estimation times for the DML models decrease if more memory is allocated. However, the marginal improvement in the estimation times is decreasing which is a typical behavior as for example documented in [1, 24]. It is also important to point out that faster estimation does not necessarily come at higher costs. In Figure 5 we see that by allocating more memory, 512 MB or 1024 MB instead of 256 MB, besides lowering the estimation time we could also lower the total costs for the estimation
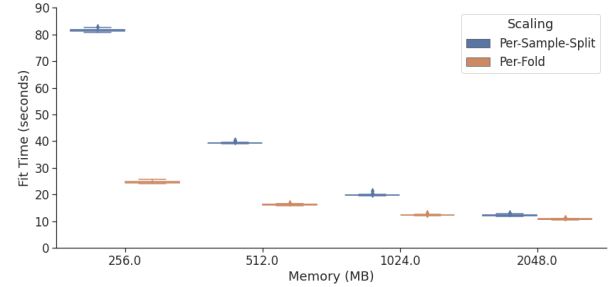

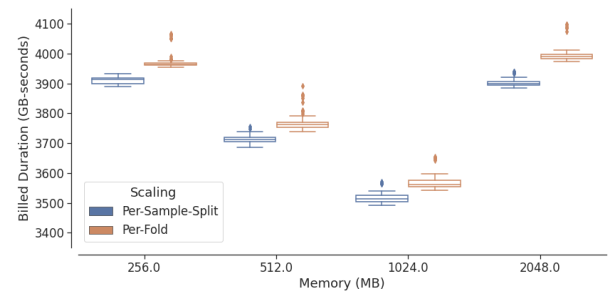**Figure 4: Serverless Fit Times with Different Scaling and Allocated Memory.**

Figure 5: Serverless Costs with Different Scaling and Allocated Memory.

on AWS Lambda. The observation that too low or high memory allocations result in higher costs is also common for serverless computing with AWS Lambda and this observation has been used to propose cost optimization frameworks [1, 17].

When comparing the two different levels of scaling, we can see in Figure 4 that by choosing per-fold scaling the estimation times can be further decreased. It is important to note that the costs are only slightly increasing when going from per-sample-split to per-fold scaling (see Figure 5). This is one of the benefits of serverless computing where one can increase the concurrency dramatically but still the billing is proportional to the actual computing time and therefore is often only slightly increased due to more overhead.

Table 1 provides more detailed results for the cheapest case in our experiment which is the setting with 1024 MB memory allocated and per-sample-split scaling. We can see that in the 100 repetitions of our experiment the estimation time was on average 19.82 seconds. The response time from the invocation of the first lambda until we received the predictions from each of the 200 invocations took on average 19.09 seconds and the average computation time for a single invocation was 17.16 seconds.[9] Therefore, in this setting we are very close to the ambitious goal that using serverless computing the estimation of the DML model with repeated cross-fitting only takes a little bit more time than estimating with only a single sample

---

[8]https://docs.doubleml.org

[9]The maximum total response time of 20.76 seconds for 200 invocations, each with an computation time between 17.16 and 17.44 seconds (see Table 1), also gives some indication that a high level of elasticity seems to be achievable. For an empirical evaluation of the elasticity of different FaaS platforms we refer to [31].

split on a machine with similar CPU power as one lambda. In Table 1 we can further see that the average estimation costs amount to 3515.36 GB-seconds, which translates to roughly 0.05858 USD at the current price of 0.0000166667 USD per GB-second that AWS charges in eu-central-1 [5].[10]

Table 1: Serverless Fit Times and Costs with 1024 MB Memory and Per-Sample-Split Scaling (Mean, Min & Max in 100 Runs).

	Mean	Min	Max
Fit Time (s)	19.82	19.53	21.49
Billed Duration (GB-s)	3515.36	3492.01	3571.42
Avg. Duration per Invocation (s)	17.16	17.05	17.44
Total Response Time (s)	19.09	18.81	20.76

6 DISCUSSIONS

In the following, we discuss features, advantages and limits of the current prototype implementation `DoubleML-Serverless` and give an outlook to potential future extensions.

Reproducibility and seeds: The prototype comes with a basic implementation of seeds to obtain reproducible results. We refer to the `numpy` documentation [35] for a discussion of parallel random number generation.

Launch overhead & cold vs. warm invocations: It is well known that there is a launch overhead when using serverless computing which results in timing differences between so-called cold and warm starts. We report timings for warm starts and refer to [24, 26] for a discussion of the phenomenon.

Transfer of ML models: The ML models are transferred at invocation using their string representation and only a subset of all `scikit-learn` ML-models is supported. To transfer more sophisticated learners, an alternative approach like pickling the learners similar to `PyWren` could be implemented.

Data transfer via payloads: The prototype uses the payloads to transfer the test indices and to return the predictions. This implies some restrictions, which could be overcome by implementation of a data transfer via S3.

Distributed storage: The datasets, which are loaded in every learning task, are stored in the Amazon S3 object storage. An alternative would be the AWS Elastic File System (EFS) which can be mounted directly for AWS Lambda calls [13].

Cost optimization: The main configuration parameter of AWS Lambda is the allocated memory. It is important to know that AWS Lambda allocates CPU power proportional to the amount of memory. Therefore, the memory allocation has an impact on the total execution time and the costs. Discussions and proposal for cost optimization of serverless applications are provided in [24, 43] and

implementations of frameworks for cost optimization in [1, 17]. Similar approaches could also be used to cost-optimize our prototype `DoubleML-Serverless`.

Limits on runtime and memory: Currently on AWS Lambda, there is an upper limit for execution time of 15 minutes. Obviously, our prototype cannot be used if the single fold estimation is not doable within this limit. Considering the previously discussed scenario with $K = 5$ folds, 100 splits and two nuisance functions and assuming that the estimation of each task is of similar effort, this translates to a total estimation time limit of roughly 10.5 days ($5 \times 100 \times 2 \times 15$ minutes). Note that in the past AWS Lambda regularly increased these limits.

Limits on memory: Recently AWS Lambda announced a significant increase of their memory limit from 3 GB to 10 GB [3]. This implies that serverless computing is becoming increasingly suitable and attractive for memory-intense models and big data applications. For standard applications of DML these memory limits are not an issue. However, DML is particularly well suited for causal inference in high-dimensional settings and therefore also used for very big datasets. Realizing such estimations in very high-dimensional and big data sets with our prototype will be challenging.

Parameter tuning for DML models: As usual in machine learning, hyperparameter tuning is also done for DML models. The prototype could be extended to also support hyperparameter tuning with an efficient serverless implementation.

DML models with multiple treatment variables: The prototype implementation only supports a single treatment variable but an extension to multiple treatment variables, as supported by `DoubleML`, would be straightforward.

7 CONCLUSION

For many users like econometricians, statisticians and data scientists existing serverfull frameworks for distributed machine learning have a high entry barrier and are often expensive if being used infrequently or inefficiently. In this paper we explore serverless cloud computing for estimation of double machine learning models. Our prototype `DoubleML-Serverless` using AWS Lambda gives econometricians, statisticians and data scientists access to an enormous level of parallelism, it almost comes with a "cloud button" as it can be easily deployed via AWS SAM and it comes at the advantage of a pay-per-request pricing model.

REFERENCES

[1] N. Akhtar, A. Raza, V. Ishakian, and I. Matta. 2020. COSE: Configuring Serverless Functions using Statistical Learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 129–138. https://doi.org/10.1109/INFOCOM41043.2020.9155363
[2] AWS. 2020. AWS Lambda changes duration billing granularity from 100ms down to 1ms. https://aws.amazon.com/about-aws/whats-new/2020/12/aws-lambda-changes-duration-billing-granularity-from-100ms-to-1ms/, accessed 20-Dec-2020.
[3] AWS. 2020. AWS Lambda now supports up to 10 GB of memory and 6 vCPU cores for Lambda Functions. https://aws.amazon.com/about-aws/whats-new/2020/12/

---

[10]For comparison, the estimation of the same DML model on a virtual machine (AWS EC2 instance of type `m5.2xlarge` with 8 vCPUs) takes much longer with approximately 383.90 seconds and at the same time amounts to slightly lower costs of 0.04905 USD at the current price of 0.46 USD per hour when ignoring the additional costs from setup and teardown of the virtual machine.

aws-lambda-supports-10gb-memory-6-vcpu-cores-lambda-functions/, accessed 20-Dec-2020.

[4] AWS. 2020. AWS Serverless Application Model (SAM). https://aws.amazon.com/serverless/sam/, accessed 04-Jan-2021.

[5] AWS. 2021. AWS Lambda Pricing. https://aws.amazon.com/lambda/pricing/, accessed 04-Jan-2021.

[6] P. Bach, V. Chernozhukov, M. S. Kurz, and M. Spindler. 2020. *DoubleML - Double Machine Learning in Python.* https://github.com/DoubleML/doubleml-for-py, Python-Package version 0.1.2.

[7] P. Bach, V. Chernozhukov, M. S. Kurz, and M. Spindler. 2020. *DoubleML - Double Machine Learning in R.* https://github.com/DoubleML/doubleml-for-r, R-Package version 0.1.1.

[8] Philipp Bach, Sven Klaassen, Jannis Kueck, and Martin Spindler. 2020. Uniform Inference in High-Dimensional Generalized Additive Models. arXiv:2004.01623 [stat.ME]

[9] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, and Philippe Suter. 2017. *Serverless Computing: Current Trends and Open Problems.* Springer Singapore, Singapore, 1–20. https://doi.org/10.1007/978-981-10-5026-8_1

[10] Alexandre Belloni, Victor Chernozhukov, Denis Chetverikov, and Ying Wei. 2018. Uniformly valid post-regularization confidence regions for many functional parameters in z-estimation framework. *Ann. Statist.* 46, 6B (2018), 3643–3675. https://doi.org/10.1214/17-AOS1671

[11] Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. 2013. Inference on Treatment Effects after Selection among High-Dimensional Controls. *The Review of Economic Studies* 81, 2 (2013), 608–650. https://doi.org/10.1093/restud/rdt044

[12] A. Belloni, V. Chernozhukov, and K. Kato. 2014. Uniform post-selection inference for least absolute deviation regression and other Z-estimation problems. *Biometrika* 102, 1 (2014), 77–94. https://doi.org/10.1093/biomet/asu056

[13] James Beswick. 2020. Using Amazon EFS for AWS Lambda in your serverless applications. AWS Compute Blog: https://aws.amazon.com/blogs/compute/using-amazon-efs-for-aws-lambda-in-your-serverless-applications/, accessed 04-Jan-2021.

[14] A. Bhattacharjee, A. D. Chhokra, Z. Kang, H. Sun, A. Gokhale, and G. Karsai. 2019. BARISTA: Efficient and Scalable Serverless Serving System for Deep Learning Prediction Services. In *2019 IEEE International Conference on Cloud Engineering (IC2E).* 23–33. https://doi.org/10.1109/IC2E.2019.00-10

[15] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. 2018. A case for serverless machine learning. In *Workshop on Systems for ML and Open Source Software at NeurIPS.*

[16] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. 2019. Cirrus: A Serverless Framework for End-to-End ML Workflows. In *Proceedings of the ACM Symposium on Cloud Computing* (Santa Cruz, CA, USA) *(SoCC '19).* Association for Computing Machinery, New York, NY, USA, 13–24. https://doi.org/10.1145/3357223.3362711

[17] A. Casalboni. 2020. AWS Lambda Power Tuning. https://github.com/alexcasalboni/aws-lambda-power-tuning.

[18] Neng-Chieh Chang. 2020. Double/debiased machine learning for difference-in-differences models. *The Econometrics Journal* 23, 2 (2020), 177–191. https://doi.org/10.1093/ectj/utaa001

[19] V. Chernozhukov, D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal* 21, 1 (2018), C1–C68. https://doi.org/10.1111/ectj.12097

[20] Victor Chernozhukov, Hiroyuki Kasahara, and Paul Schrimpf. 2021. Causal impact of masks, policies, behavior on early covid-19 pandemic in the U.S. *Journal of Econometrics* 220, 1 (2021), 23–62. https://doi.org/10.1016/j.jeconom.2020.09.003 Pandemic Econometrics.

[21] Kyle Colangelo and Ying-Ying Lee. 2020. Double Debiased Machine Learning Nonparametric Inference with Continuous Treatments. arXiv:2004.03036 [econ.EM]

[22] G. Feng, S. Giglio, and D. Xiu. 2020. Taming the Factor Zoo: A Test of New Factors. *The Journal of Finance* 75, 3 (2020), 1327–1370. https://doi.org/10.1111/jofi.12883

[23] Joseph M. Hellerstein, Jose Faleiro, Joseph E. Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. 2018. Serverless Computing: One Step Forward, Two Steps Back. arXiv:1812.03651 [cs.DC]

[24] V. Ishakian, V. Muthusamy, and A. Slominski. 2018. Serving Deep Learning Models in a Serverless Platform. In *2018 IEEE International Conference on Cloud Engineering (IC2E).* 257–262. https://doi.org/10.1109/IC2E.2018.00052

[25] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. 2017. Occupy the Cloud: Distributed Computing for the 99%. In *Proceedings of the 2017 Symposium on Cloud Computing* (Santa Clara, California) *(SoCC '17).* Association for Computing Machinery, New York, NY, USA, 445–451. https://doi.org/10.1145/3127479.3128601

[26] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A. Patterson. 2019. Cloud Programming Simplified: A Berkeley View on Serverless Computing. arXiv:1902.03383 [cs.OS]

[27] Nathan Kallus and Masatoshi Uehara. 2020. Double Reinforcement Learning for Efficient Off-Policy Evaluation in Markov Decision Processes. *Journal of Machine Learning Research* 21, 167 (2020), 1–63.

[28] Sven Klaassen, Jannis Kueck, and Martin Spindler. 2017. Transformation Models in High-Dimensions. arXiv:1712.07364 [stat.ME]

[29] Sven Klaassen, Jannis Kück, Martin Spindler, and Victor Chernozhukov. 2018. Uniform Inference in High-Dimensional Gaussian Graphical Models. arXiv:1808.10532 [stat.ME]

[30] Michael C. Knaus. 2020. Double Machine Learning based Program Evaluation under Unconfoundedness. arXiv:2003.03191 [econ.EM]

[31] Jörn Kuhlenkamp, Sebastian Werner, Maria C. Borges, Dominik Ernst, and Daniel Wenzel. 2020. Benchmarking Elasticity of FaaS Platforms as a Foundation for Objective-Driven Design of Serverless Applications. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (Brno, Czech Republic) *(SAC '20).* Association for Computing Machinery, New York, NY, USA, 1576–1585. https://doi.org/10.1145/3341105.3373948

[32] J. Kuhlenkamp, S. Werner, and S. Tai. 2020. The Ifs and Buts of Less is More: A Serverless Computing Reality Check. In *2020 IEEE International Conference on Cloud Engineering (IC2E).* 154–161. https://doi.org/10.1109/IC2E48712.2020.00023

[33] Greg Lewis and Vasilis Syrgkanis. 2020. Double/Debiased Machine Learning for Dynamic Treatment Effects. arXiv:2002.07285 [econ.EM]

[34] Yusuke Narita, Shota Yasui, and Kohei Yata. 2020. Off-policy Bandit and Reinforcement Learning. arXiv:2002.08536 [cs.LG]

[35] NumPy. 2021. Parallel Random Number Generation. https://numpy.org/doc/stable/reference/random/parallel.html, accessed 04-Jan-2021.

[36] P. M. Robinson. 1988. Root-N-Consistent Semiparametric Regression. *Econometrica* 56, 4 (1988), 931–954.

[37] Vira Semenova and Victor Chernozhukov. 2020. Debiased machine learning of conditional average treatment effects and other causal functions. *The Econometrics Journal (forthcoming)* (2020). https://doi.org/10.1093/ectj/utaa027

[38] Vira Semenova, Matt Goldman, Victor Chernozhukov, and Matt Taddy. 2020. Estimation and Inference about Heterogeneous Treatment Effects in High-Dimensional Dynamic Panels. arXiv:1712.09988 [stat.ML]

[39] Gerard Torrats-Espinosa. 2021. Using machine learning to estimate the effect of racial segregation on COVID-19 mortality in the United States. *Proceedings of the National Academy of Sciences* 118, 7 (2021). https://doi.org/10.1073/pnas.2015577118

[40] Zhucheng Tu, Mengping Li, and Jimmy Lin. 2018. Pay-Per-Request Deployment of Neural Network Models Using Serverless Architectures. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations.* Association for Computational Linguistics, New Orleans, Louisiana, 6–10. https://doi.org/10.18653/v1/N18-5002

[41] Erwin van Eyk, Alexandru Iosup, Cristina L. Abad, Johannes Grohmann, and Simon Eismann. 2018. A SPEC RG Cloud Group's Vision on the Performance Challenges of FaaS Cloud Architectures. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering* (Berlin, Germany) *(ICPE '18).* Association for Computing Machinery, New York, NY, USA, 21–24. https://doi.org/10.1145/3185768.3186308

[42] Erwin van Eyk, Alexandru Iosup, Simon Seif, and Markus Thömmes. 2017. The SPEC Cloud Group's Research Vision on FaaS and Serverless Architectures. In *Proceedings of the 2nd International Workshop on Serverless Computing* (Las Vegas, Nevada) *(WoSC '17).* Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3154847.3154848

[43] H. Wang, D. Niu, and B. Li. 2019. Distributed Machine Learning with a Serverless Architecture. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications.* 1288–1296. https://doi.org/10.1109/INFOCOM.2019.8737391