Statistical Learning with Extreme Values
Reading report on
Gradient Boosting for extreme quantile regression

Naël Farhan, Matthieu Carreau

December 3, 2023

1 Context and Motivation

One of the main topics in extreme values learning is high-level quantile estimation, as it allows to give an idea of the likelihood of rare events, such as extreme weather events, and then helps to manage risks associated with these events.

For a variable of interest $Y \in \mathbb{R}$, extreme value theory (EVT) offers us a great setting to estimate $Q(\tau)$ at a high-level, that is the General Pareto Distribution (GPD) model. Called the Peak-Over-Threshold, the inference method is based on the property that, for a threshold u high enough, $Y-u|Y > u$ follows a GPD distribution, meaning :

$$\mathbb{P}(Y - u > y|Y > u) \approx 1 - H_{\gamma,\sigma}(y) = (1 + \frac{\gamma y}{\sigma})_+^{-1/\gamma}, \sigma > 0, \gamma \in \mathbb{R} \tag{1}$$

The method then consists first on estimating γ, σ via methods such as maximum likelihood or moments, then finding a convenient threshold u (one possible method was implemented in homework n°2), estimate the associated quantile τ_0 and finally estimate a higher quantile τ using this approximation (which was proven in class) :

$$Q(\tau) \approx Q(\tau_0) + \sigma \frac{\left(\frac{1-\tau}{1-\tau_0}\right)^{-\gamma} - 1}{\gamma} \tag{2}$$

This setup is great when we focus only on one variable of interest Y. However, in the regression setup where we have access to covariates $X \in \mathbb{R}^d$ (also called predictors) which Y depends on, quantile estimation is a bit more tricky and still a field of ongoing research. The quantity of interest is the conditional quantile $Q_X(\tau) = F_Y^{-1}(\tau|X = x)$ which can be hard to estimate especially for large values of d and when Y allows some complex non-linear effects. In the literature, some methods for quantile regression exist based on gradient forest exist but don't include extreme value theory such as equations 3.1 and 2. For extreme quantile regression, both parametric (e.g. linear conditional quantile function) and non-parametric approaches taking into account EVT were made, including additive modelling and kernel smoothing. Such approaches either lack in modelling flexibility or suffer the curse of dimensionality, hence the use of tree-based methods. This was first implemented with a single tree, and our paper extends this method using Gradient Boosting. In a classical regression setting, gradient boosting is a predictive model that takes into account several weak learners (decision trees) to create a stronger one. Its known robustness and versatility, especially to handle complex data structures, motivates its use combined with EVT, which is the method described in the following section.

2 Gradient Boosting method

2.1 Setup

First, let's describe the setup of the method. We have some observations (X_i, Y_i) which are independent copies of some (X, Y). Here, our three main quantities of interest $Q(\tau_0), \gamma, \sigma$ may depend

on covariates, meaning we assume that the approximation 3.1 holds for any $x \in \mathbb{R}^d$ with parameters $Q_x(\tau_0), \gamma(x), \sigma(x)$. We then retrieve our estimation with :

$$Q_x(\tau) \approx Q_x(\tau_0) + \sigma(X)\frac{\left(\frac{1-\tau}{1-\tau_0}\right)^{-1/\gamma(x)} - 1}{\gamma(x)} \tag{3}$$

2.2 Algorithm

Initial quantile $Q_x(\tau_0)$

For the estimation of $Q_x(\tau_0)$, it is not necessary to use EVT, as a simpler quantile regression algorithm can be used such as quantile random forest which is chosen in the paper. Also, for the choice of τ_0, because we will use the excedances for gradient boosting, we need enough data points so a not-so-high τ_0, but also a high enough value to justify our use of the approximation 3. In the paper's simulations τ_0 was set at 0.8.

Estimation of $\gamma(x), \sigma(x)$

After retrieving $\hat{Q}_x(\tau_0)$, the goal is to estimate $\theta(x) = (\sigma(x), \gamma(x))$ only for samples over the initial quantiles, meaning : $Z_i = Y_i - \hat{Q}_{X_i}(\tau_0)$ if $Y_i > \hat{Q}_{X_i}(\tau_0)$, otherwise the sample is dropped.

Before going in details about Boosting, let's introduce our loss function, based on the negative log-likelihood (also called deviance) of a GPD distribution with parameters $\theta(X_i)$:

$$l_{Z_i}(\theta(X_i)) = (1 + \frac{1}{\gamma(X_i)})log(1 + \gamma(X_i)\frac{Z_i}{\sigma(X_i)}) + log(\sigma(X_i)) \tag{4}$$

This choice is justified by its standard use to estimate θ in absence of covariates when $\gamma > -1/2$, as it gives asymptotically normal estimators.

To initialize the algorithm, a first estimation θ_0 is made with a MLE estimator on Z_i's without looking at the X_i's. Then we update our parameters iteratively with B pairs of trees. At each step $b = 1, ..., B$, we create for each parameter a tree of depth D^σ, D^γ, with minimum leaf sizes $L^\sigma_{min}, L^\gamma_{min}$ to update $\theta_{b-1}(x)$ given this methodology :

- A random subsample of fraction $s \in (0, 1]$ is drawn from exceedances
- For this subsample, two regression trees are fitted on gradients $r^\gamma_{b,i}, r^\sigma_{b,i}$ which are deviance derivatives over each parameter (model residuals), taken at the previous parameter
- Parameters are updated via : $\theta_b(x) = (\sigma_{b-1}(x) + \lambda^\sigma T^\sigma_b(x), \gamma_{b-1}(x) + \lambda^\gamma T^\gamma_b(x))$, where λ are the learning rates for each parameter and $T_b(x)$ the outputs of each tree.

Quantile estimation

Finally, with the parameters updated after B iterations, the algorithm outputs conditional quantiles according to the formula :

$$\hat{Q}_x(\tau) = \hat{Q}_x(\tau_0) + \sigma_B(x)\frac{\left(\frac{1-\tau}{1-\tau_0}\right)^{-1/\gamma_B(x)} - 1}{\gamma_B(x)} \tag{5}$$

3 Implementation and Experiments

In order to understand the performances and limitations of the algorithm, we first conducted several qualitative experiments on different synthetic datasets. We chose to take $X \in \mathbb{R}^2$ to make visualizations easier, and applied the methodology for different types of distributions of Y and different forms of dependence of the conditional distribution of Y depending on X.

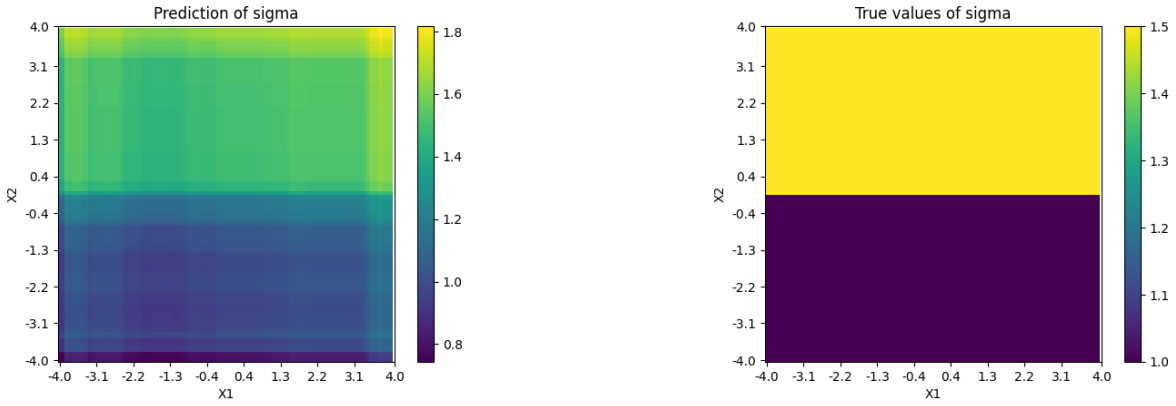

Figure 1: Estimation of scale parameter by gradient boosting

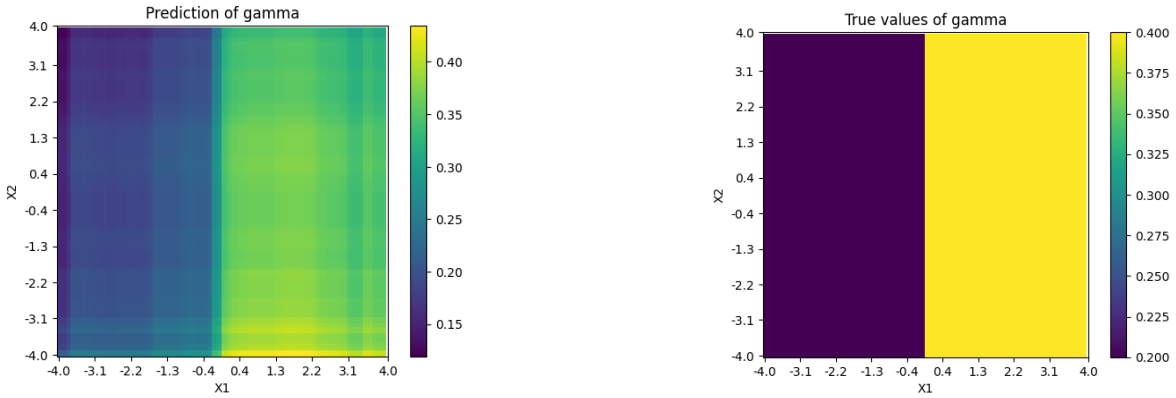

Figure 2: Estimation of shape parameter by gradient boosting

3.1 Gradient boosting for GPD modelling

We first implemented the estimation of the parameters $\sigma(x)$ and $\gamma(x)$ with gradient boosting (algorithm 1 of [1]) on a dataset generated from a GPD distributions.

10000 points (X_i) were sampled uniformly on $[-4, 4]^2$ and then each Y_i was sampled according to $GPD(\sigma(X_i), \gamma(X_i))$, where σ and γ were defined as two step functions along different axes.

The result of the estimators produced by the algorithm are displayed in figure 1 and 2, respectively for $\hat{\sigma}$ and $\hat{\gamma}$.

We observe that the estimators represent quite well the true step functions but they are less stable close to the boundaries of the support of X, where some leaves of the trees might have been trained with less values.

Moreover, we note that $\hat{\sigma}$ is more constant than $\hat{\gamma}$ in the desired regions, indeed, $\hat{\gamma}$ tends to be overestimated where σ is lower and underestimated in the inverse case. This illustrates the fact that the shape parameter is in general harder to predict and that its estimator is also sensitive to changes of other parameters.

3.2 Distributions in different domains of attraction

To test the whole extreme quantile regression algorithm we generated three different datasets, in the three domains of attraction.

We choose the Cauchy, exponential and beta families of distribution, and for each of them, we compared the quantile estimator obtained with the true conditional quantile function of X, at a fixed quantile level of 0.995.

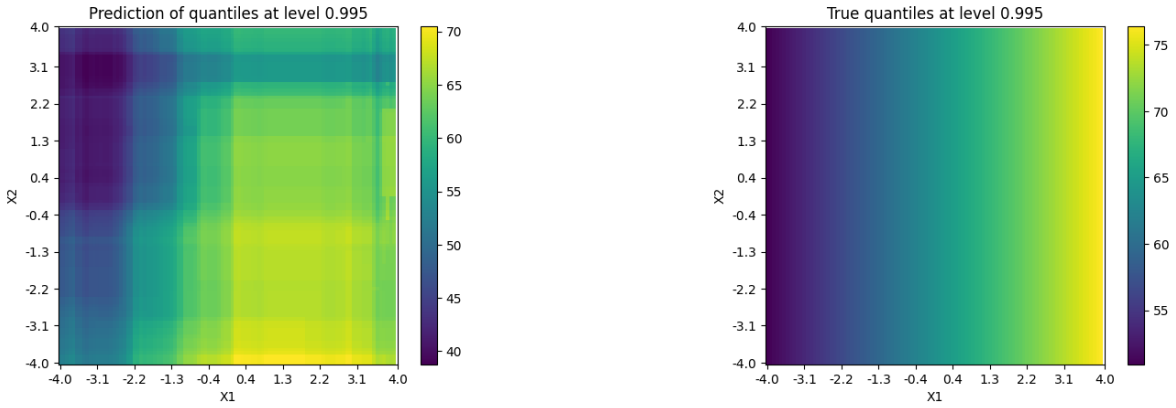

Figure 3: Extreme quantile prediction on the Cauchy distribution

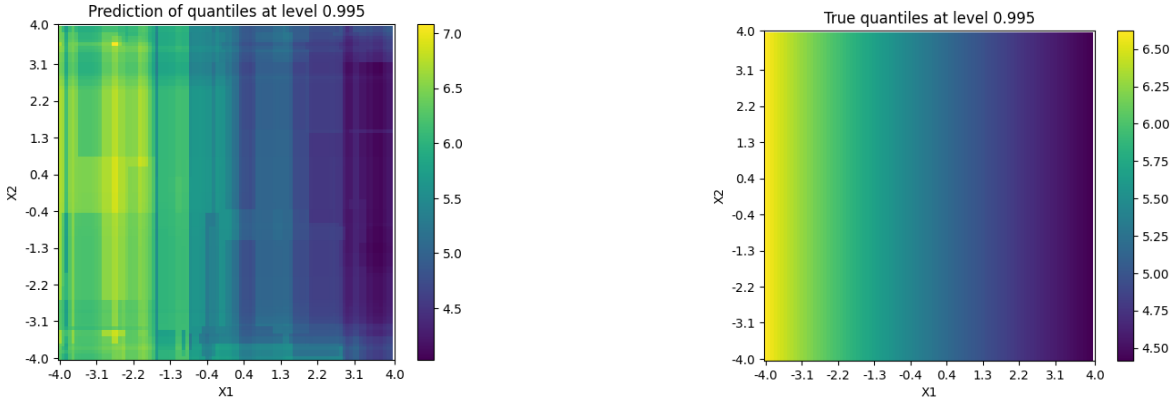

Figure 4: Extreme quantile prediction on the exponential distribution

Fréchet domain

For the Cauchy distribution, the scale parameter of the distribution was defined as an affine function of the first coordinate of X. The quantile estimator obtained is shown in 3. Even if the correct order of values is predicted, the quantiles are underestimated and we observe that they are not constant with respect to the second component of X, which indicates overfitting on the training data.

Gumbel domain

We generated exponential distributions, where the scale parameter is an affine function of the first covariate.

Figure 4 presents the estimator learned from this dataset. This experiment gives more satisfying results than the previous one as the quantiles predicted are closer to the true ones, and we observe less overfitting on the training data.

Weibull domain

For this last domain, we genarated a dataset from beta distributions with parameters $(\alpha(X), \beta(X))$, where α is fixed to 1 and $\beta(x) = 2 + \frac{x_1}{8}$.

We note that the shape parameter γ of the corresponding theoritical GPD model for extremes is equal to $\frac{-1}{\beta}$, and therefore, for $x_1 > 0$, we have $\gamma < \frac{-1}{2}$ which is outside the domain where the estimation with equation 4 is associated to theoritical guarantees.

We observe the results in figure 5, which shows important variations of the predictions at fixed x_1. The estimated quantiles sometimes go beyond the upper bound 1 of Y, the precision of these

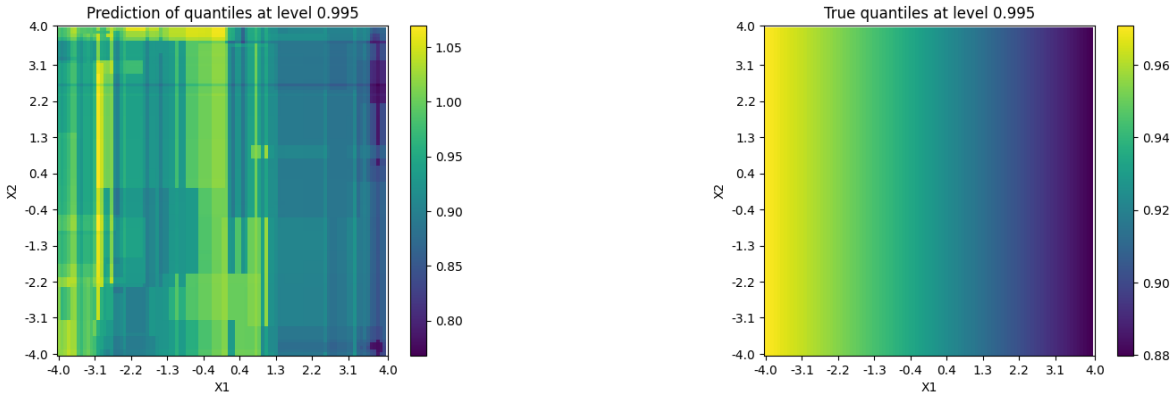

Figure 5: Extreme quantile prediction on the beta distribution

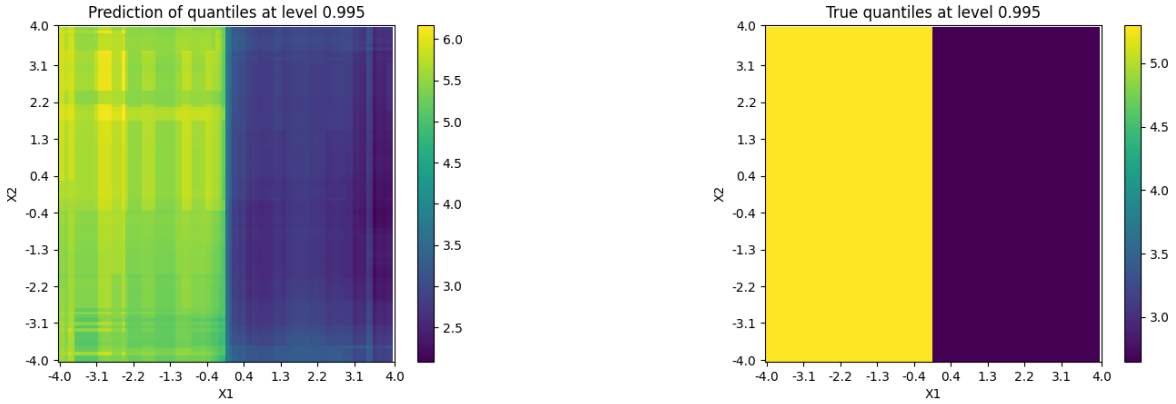

Figure 6: Simple split along an axis

estimations is not satisfying for this type of bounded distributions, where the quantiles are expected to be close to each other.

These qualitative experiments show that the method can give good estimations for distributions with shape parameter $\gamma \geq 0$, but can be more difficult to apply for some distributions in the Weibull domain, especially when $\gamma < -\frac{1}{2}$.

3.3 Types of dependence in the covariates

The use of tree-based regressors leads to a method that is not invariant to rotations, because the input space is divided in regions by splits along the axes. Therefore, we can expect the algorithm to have better performances when the dependence of the true conditional distribution on the covariates is easier to interpret in the canonical basis.

This section illustrates this idea with experiments that compare the predicted quantile on a plane between datasets that are rotations of each other, using exponential distributions.

The first dataset defines the scale parameter of the distribution as a step function of x_1. As we can see in 6, the corresponding quantile prediction shows a clear frontier at $x_1 = 0$ between two nearly uniform areas. We compare this prediction to the one obtained in figure 7, where the distribution parameter is now a step function of $x_1 + x_2$, so that the true frontier is not along an axis anymore. The prediction results in regions that are less uniform with a blurred boundary.

We can increase the complexity of this first example by splitting the input space into four different regions, as shown in figure 8. Here, even if the boundaries are in the directions of the axes, we can see in the predicted quantile map that the four regions are not homogeneous, but the boundaries are still

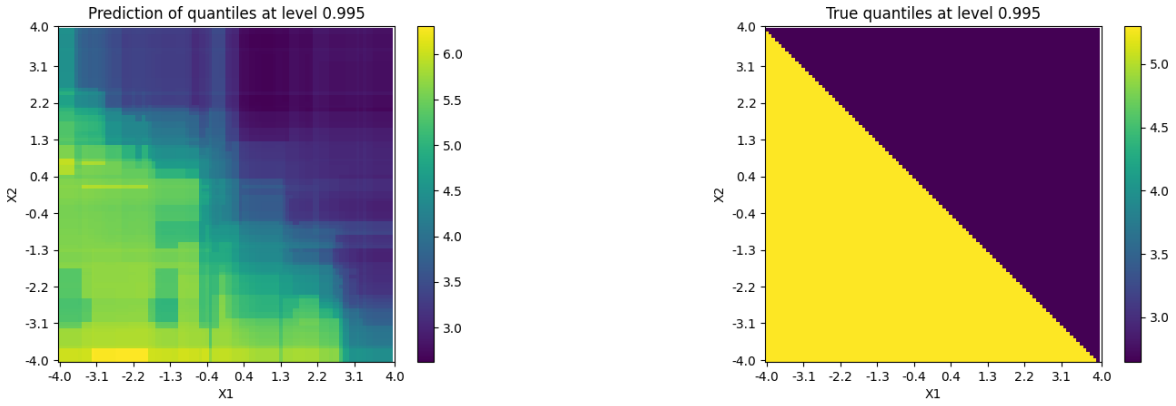

Figure 7: Simple split tilted from the axes

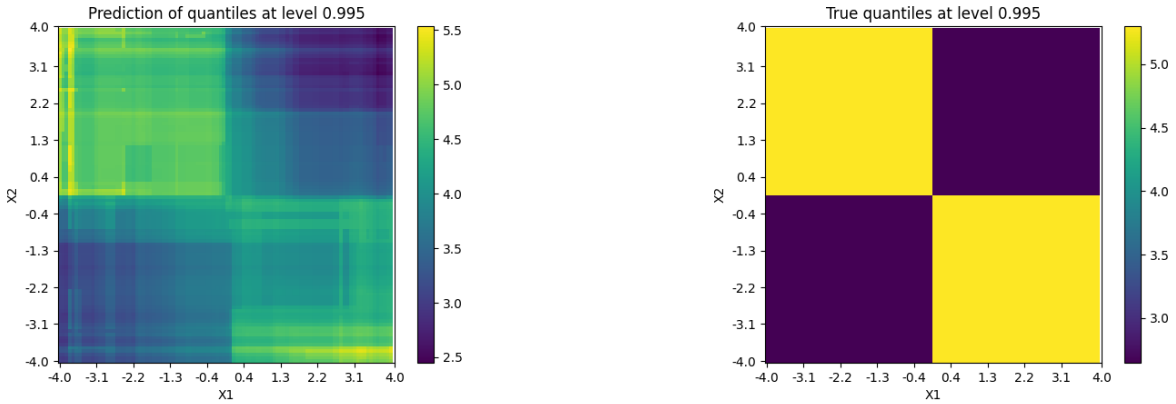

Figure 8: 4-region split

well visible. When we rotate these regions by an eighth of a turn, such as in figure 9, the prediction performances decrease again, and we do not observe the four desired regions on the prediction map.

These examples illustrate that even for quite simple functions, the predictive performances of the model drop when they involve several components at the same time, because the tree-based model often fails to capture patterns that depend on linear combinations of the covariates.

3.4 Parameters tuning and performance comparison

In this section, we will now see how performance behaves when modifying some key parameters, and we will compare it to some other quantile regression models. But first, let's introduce the main metric in quantile regression presented in the article, which is the Integrated Squared Error (ISE) defined by :

$$ISE_\tau = \int_{[-4,4]^d} (\hat{Q}_x(\tau) - Q_x(\tau))^2 dx \tag{6}$$

In practice, and because we considered covariates which are independent uniform variables on $[-4, 4]$ in our examples, we used the empirical version of this error in a Monte Carlo approach via :

$$\hat{ISE}_\tau = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (\hat{Q}_{X_i}(\tau) - Q_{X_i}(\tau))^2 \tag{7}$$

Where the X_i's are n_{test} independent draws from the distribution of training. This metric will enable us to compute performance in the case where we can know the true value of the conditional quantile.

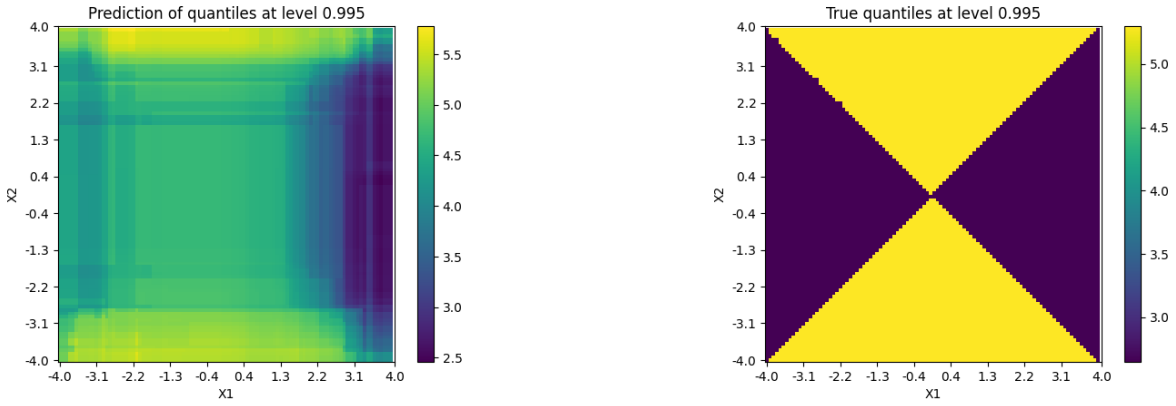

Figure 9: 4-region split, tilted from the axes

Model considered

In this section, we will consider an adaptation the GPD model of section 3.1 : we consider 4 independent covariates, each uniform on $[-4, 4]$, and a GPD model with γ a step function of X_1 (0.2 if negative, 0.4 else) and σ a step function of X_2 (1 if negative, 1.5 else). X_3 and X_4 here play the role of noise, which is important because we want to test if the model is robust to the addition of noisy covariates.

Parameters of interest : Number of trees and depths

In gradient boosting in general, the number of trees B is the most important parameter that can produce underfitting or overfitting. The article recommends a K-fold cross-validation (typically $K = 5$ or 10) and B the minimizer of the deviance defined as :

$$DEV_{CV}(B) = \sum_{k=1}^{K} \sum_{i \in D_k} l_{Z_i}(\hat{\theta}_B^{-D_k}(\mathbf{X}_i)) \tag{8}$$

Each $\hat{\theta}_B^{-D_k}$ is the model trained on data with D_k left out with B trees : this cross-validation deviance adds up all deviances for each fold of our data by evaluating the likelihood (same as 4) of the left-out data under the trained model. This doesn't involve true quantiles but focuses on the validity of the GPD approximation made by the model. We also prefer to use deviance instead of ISE because it doesn't depend on quantiles of interest (only τ_0).

This parameter needs to be studied with tree depths D^σ, D^γ. These parameters, also called interaction depth, allow complexity in the relations between covariates and our parameter $\theta(X)$. For instance, trees of depth 0 produce a constant variable for the parameter. This is important especially for the shape parameter γ which is difficult to estimate in practice; therefore it might happen to assume $\gamma(X) = \gamma$, meaning depth 0. Recommendations are to consider depth of 3 at most, for the sake of interpretation, and to use cross-validation just like B. To test these statements, we kept 5 values of the couple D^σ, D^γ, and looked at how $ISE_{0.995}$ varies with B as well as compared it to the choice made with cross-validated deviance. In the figure 10, for each set of parameters, ISE was computed as a mean of 100 simulations with 100 data points drawn each.

First of all, because of computation times, we did not make computations for every value of B, but only for some (step of 10 between each). What's striking here is a high instability contrary to the paper which presents smooth curves. Because our original dataset only contains 10 000 observations and that, with $\tau_0 = 0.8$, we only keep around 2 000 observations for training, such instability was expected. Though we can observe a decrease and then some kind of stability in performance for each depths. Another thing is that the performance achieved using cross validation doesn't always come close to the best performance overall. Finally, we see that when one or both depths are equal to 2, we obtain higher errors overall. Because of the simplicity of our modelling for γ and σ, decision trees with only depth one catch the simple splits better.

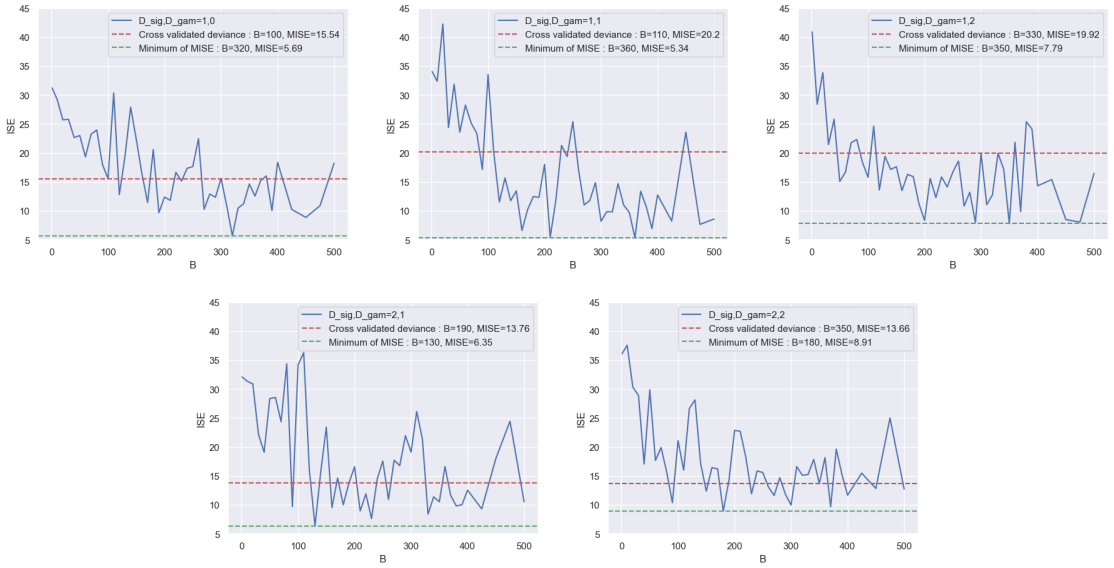

Figure 10: For some sets of tree depths, evolution of performance as the number of trees B varies

The final decision, following cross-validated deviance, was to chose $B = 190$ with depths $D^\sigma, D^\gamma = 1, 1$.

Comparison with other models

After selecting parameters B, D^σ, D^γ for our algorithm, we compared its performance to other quantile regression algorithms. The first one, called *constant*, is a simple method which estimates γ, σ considering them constants, therefore only making regression on the initial conditional quantile $Q_x(\tau_0)$: we actually recover this method with our algorithm and 0 trees. The other model considered is the quantile random forest regressor (*qrf*), a non-parametric, tree-based method that doesn't take into consideration Extreme Value Theory. Other models, such as generalized random forest for quantiles or Generative additive models for exceedances of high thresholds were considered in the original paper but not implemented in Python, so these are out of the scope of our project. With parameters chosen above, our algorithm had an ISE of 21, whereas the constant method was at 44, and the quantile random forest was at more than 200. When we know that the 4 values possible for our conditional quantiles are around $\{9, 14, 18, 27\}$, we see that 21 (meaning the squared error is around 4.5) is a very good score and that for high quantiles, a more classic regression method that don't take Extreme Value Theory into account perform badly.

Variable importance

Another field of interest which is important in model interpretability is variable importance, to know which covariate impacts model deviance. Therefore is introduced a permutation score :

$$I(X_j) = \sum_{i=1}^{n} l_{Z_i}(\hat{\theta}(X_i^{(j)})) - l_{Z_i}(\hat{\theta}(X_i)) \tag{9}$$

With $X_i^{(j)}$ the same data as X_i except the j-th variable is randomly shuffled. Large $I(X_j)$ implies that X_j has a strong effect in the model. In our model with noise, when making several random shuffles to compute permutation score, we expect the importance to be zero for noise covariates and high for the other. That's what we see in figure 11.

Note that it's interesting to see higher values for X_1 than X_2. This is probably the case because X_1 controls γ which is the most impactful parameter in GPD, so shuffling values in the variable that controls it might cause more chaos to our system than shuffling what controls σ.

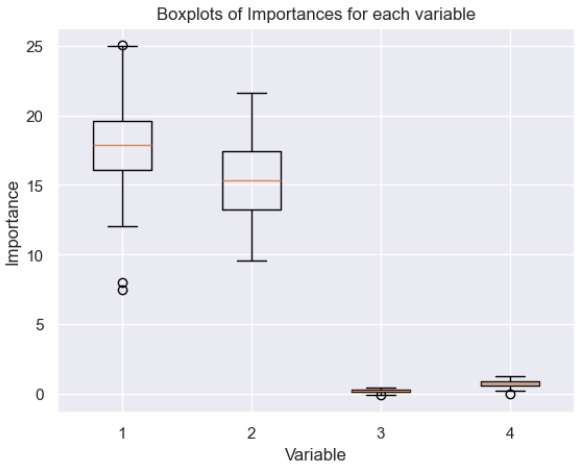

Figure 11: 100 simulations to compute permutation scores for each covariate

3.5 Sensibility to dimension

Another subject that we wanted to touch on was about dimension complexity. The curse of dimensionality is a major challenge in Machine Learning in general, and as the paper claims, we saw that adding noise to the data did not unable the algorithm to achieve high performance. However, we wanted to see, for a fixed set of data points and covariates, when quantitiesof interest depend on more covariates, if at some point the performance would shrink.

For that, we kept a GPD model, but this time with 30 covariates. For some integer j, $X_1, ..., X_j$ control γ, $X_{j+1}, ..., X_{2j}$ control σ and each parameter can take the same two values as section 3.1. The regions are splitted symmetrically at the sign of the product of all considered covariates. In the table 1 down below are performances for some values of j, with model parameters taken from the previous section. We also included results for the constant method.

j	1	2	3	4	5	6	7	8	9	10
$ISE_{0.995}$(gbex)	4.9	23.2	48.1	44.9	46.7	47.2	46.2	48.3	45.0	46.9
$ISE_{0.995}$(cst)	10.3	42.8	44.1	46.8	44.4	49.9	47.3	48.0	49.9	44.4

Table 1: Model performance for several values of j

We see that very quickly performance becomes really bad compared to the simple case $j = 1$. When running the *constant* method on these splitted regions, we saw similar results in ISE for values of $j > 2$, which means that gradient boosting becomes totally ineffective in such cases as it fails to recover simple regions splits from covariates. This might be due to the fact that not enough data points above exceedances are retrieved for the algorithm to find more diverse regions. It might also have to do with the depths of trees, which stayed at $1, 1$ here, or the number of trees, but *gbex* seems not to perform well quickly as j rises.

4 About the other parameters

There are many parameters in the model that we did not discuss in our implementations, and that could be the focus of further analysis, but let's mention them briefly.

First, we did not specify learning rates. This is mainly because there is a known balance between learning rates and number of trees : often, we fix learning rates (typically 0.01 or 0.001) and adjust B. The article parametrizes these learning rates such that : $\lambda_{scale}, \lambda_{ratio} = \lambda^\sigma, \lambda^\sigma/\lambda^\gamma$. It is natural to assume $\lambda_{ratio} > 1$ as γ often requires more regularisation and ranges on smaller scales : in all our simulations, we took $\lambda^\sigma = 0.01$ and $\lambda_{ratio} = 10$.

For the remaining parameters, here are some information :

- Minimum leaf sizes $L^{\sigma}_{min}, L^{\gamma}_{min}$: allows leaves to have enough observations to produce a smoother gradient, recommended in $[10, n/50]$ (10 in our simulations)
- Subsample fraction s : makes sure trees are fitted on different subsamples, recommended in $[0.4, 0.8]$ (5 for us)
- Threshold quantile τ_0 : a higher value gives better approximation for GPD distribution but fewer data points over the quantile ; there's not much theoretical knowledge on how to chose τ_0, so we went along the paper and chose 0.8 .

Moreover, we focused on the parameters of the GPD estimation because this was the main contribution of the paper, but the first step of non-extreme quantile regression also relies on parameters. The tuning of these parameters should also deserve a cross-validation study, because good intermediate quantile estimates are crucial for the rest of the procedure.

5 Conclusion

The experiments we conducted with our implementation of the algorithm allowed to reproduce and confirm some points mentioned in the paper, but also highlighted some limitations of the proposed algorithm.

We confirmed that the gradient boosting with GPD tail modelling gives better extreme quantile estimations than standard quantile regressions or GPD modelling without considering covariates.

We found are that the predictions are very unstable for bounded distributions, i.e when $\gamma < 0$.

Moreover, we showed that the algorithm performs best when the distribution parameters have a simple expression in the canonical basis of the input space, but that these performances drop significantly when these expressions involve linear combinations of the covariates.

Finally, the experiments on the number of covariates show that the gradient boosting algorithm is not as suited for high dimensions as we expected, and it is subject to the curse of dimensionality as other algorithms.

References

[1] Velthoen, J., Dombry, C., Cai, JJ. et al. *Gradient boosting for extreme quantile regression. Extremes (2023).*