Gradient Ascent for Best Response Regression

Victoria Racher[1,3] and Christian Borgelt[1,2]

[1] Department of Mathematics, University of Salzburg, Salzburg, Austria
[2] Department of Computer Sciences, University of Salzburg, Salzburg, Austria
[3] Paracelsus Medical University, Salzburg, Austria

Abstract. Although regression is among the oldest areas of statistics, new approaches may still be found. One recent suggestion is Best Response Regression, where one tries to find a regression function that provides, for as many instances as possible, a better prediction than some reference regression function. In this paper we propose a new method for Best Response Regression that is based on gradient ascent rather than mixed integer programming. We evaluate our approach for a variety of noise (or error) distributions, showing that especially for heavy-tailed distributions best response regression outperforms, on unseen data, ordinary least squares regression, both w.r.t. the sum of squared errors as well as the number of instances for which better predictions are provided.

Keywords: best response regression · objective function smoothing · gradient ascent · resilient backpropagation · Newton–Raphson method

1 Introduction

Prediction is central to both Statistics and Machine Learning. Given a set of instances, described by one or more explanatory (or input) variables, along with a response (or output) value for each instance, a predictor has to identify the response for a new (unseen) instance as accurately as possible. Usually, if the response value is categorical, the prediction task is called classification, if it is metric, it is called regression. Many approaches to such problems are well known. For instance, in the case of regression, we commonly aim to minimize the sum of squared errors, often for a simple linear prediction function. Admittedly, linear regression is one of the simplest models to understand in terms of regression. Nevertheless, this does not mean that the classical statistical approaches are the best or even only ones. On the contrary, the combination of different subfields of Mathematics and Computer Science allows new and interesting perspectives. We follow the recent idea by Ben-Porat and Tennenholtz, which is called Best Response Regression [2]. The essential theory behind Best Response Regression is the combination of concepts from game theory and statistical learning.

This paper ist organized as follows: In Section 2 we review Best Response Regression and point out some shortcomings of the original approach. In Section 4 we derive our own gradient ascent based approach drawing on a relaxation (or "smoothing") of the original objective function. In Section 5 we report experimental results, focusing on exploring different noise (or error) distributions. Finally, in Section 6 we draw conclusions from our discussion.

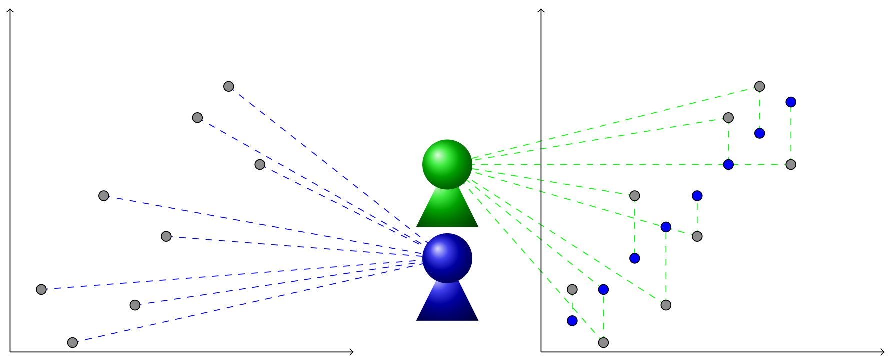


Fig. 1. Illustration of the main idea of Best Response Regression. An expert (blue) sees instances along with their true response values to make predictions, an agent (green) sees also the discrepancies between the expert's predictions and the true values.

2 Best Response Regression

In 2017, Ben-Porat and Tennenholtz [2] proposed a completely new way of interpreting (linear) regression tasks based on machine learning techniques, which they called Best Response Regression. They interpreted the prediction task as a game that two players, an agent and an expert, play against each other. The expert uses a number of historical instances along with their response values to make predictions for a new (unseen) instance. The agent also sees the historical instances and their true response values, but gets as additional information the discrepancies between the expert's predictions and the true values. The main idea is illustrated in Figure 1. The fundamental difference to classical approaches is that Best Response Regression is no longer trying to minimize the discrepancy between predicted and observed values, but focuses on maximing the probability to predict better than an expert (represented, e.g., by a reference prediction function). For more details and a game theoretical justification, we refer to [2].

2.1 Shortcomings of the Approach by Ben-Porat and Tennenholtz

To find a best reponse regression function, Ben-Porat and Tennenholtz formulated a Mixed Integer Linear Problem, where the optimization step relied on the Simplex Algorithm. Unfortunately, as the number of explanatory variables increases, the runtime of the Simplex Algorithm increases exponentially. This problem is handled by setting timeouts, that is, the implementation outputs the best solution that could be found before the timeout. However, one does know neither whether this is the final optimal solution nor whether the solution is unique. Therefore, we considered a new approach using gradient ascent on "smoothed" versions of the objective function. Even though such an approach also cannot guarantee that an optimal solution is found, it is computationally much more efficient and, in principle, applicable to arbitrary regression functions and arbitrary numbers of explanatory (or independent) variables.

3 Notation

Let a data set $(\mathbf{X}, \boldsymbol{y})$ be given, where $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is a tuple of input vectors with $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{im}) \in \mathbb{R}^m$ and $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ is a vector of output values. Let $f_\circ(\boldsymbol{x})$ be a reference regression function and $f_*(\boldsymbol{x}; \boldsymbol{a})$ a best response regression function with parameters $\boldsymbol{a}$. To simplify notation, we define

$$g_\pm(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon) \;=\; f_*(\boldsymbol{x}_i; \boldsymbol{a}) - (y_i \pm (1-\varepsilon) \cdot |f_\circ(\boldsymbol{x}_i) - y_i|),$$

where ε, $0 < \varepsilon \ll 1$, is a required minimum prediction improvement. With this notation, the objective function of best response regression can be written as

$$F_H(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) \;=\; \sum_{i=1}^{n} \Big(H\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - H\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big),$$

where H is the Heaviside function (or unit step function)

$$H: \quad \mathbb{R} \to \{0, 1\}, \qquad z \mapsto \begin{cases} 1, & \text{if } z \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

The objective function F_H is to be maximized by choosing $\boldsymbol{a}$. Note that this optimization problem may not have a unique solution, since the objective function is essentially counting for how many data points the best response regression function yields a better prediction than the reference regression function and the same count may be obtained for different best response regression functions.

4 Gradient Ascent Approach

We propose to relax the optimization problem by using a "smoothed" Heaviside function, for which the (scaled) logistic function is a natural (first) choice:

$$L_\beta(z) \;=\; \frac{1}{1 + e^{-\beta z}} \;=\; \left(1 + e^{-\beta z}\right)^{-1}.$$

Here β is a steepness parameter: the greater β, the steeper (and thus the less "smooth") the function $L_\beta(z)$ is. For $\beta \to \infty$ we get $L_\beta(z) \to H(z)$. This leads to a "smoothed" objective function (on which a gradient ascent becomes possible)

$$F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) \;=\; \sum_{i=1}^{n} \Big(L_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big),$$

which is again to be maximized by choosing $\boldsymbol{a}$. Figure 2 provides an illustration of one term of the original and the smoothed objective function.

Clearly, a best response regression function has to pass through the region, in which the depicted ith term of $F_H(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$ (black line) is one, in order to "win" the ith data point. The core idea of our approach is to relax this condition by asking for a value as high as possible for the ith term of $F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$ (red line). Note that by letting the parameter β grow in the course of the optimization, we can always approach the original function $F_H(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$ as closely as desired.

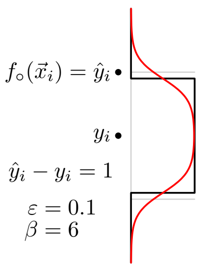


Fig. 2. The value y_i is the true output for input vector $\boldsymbol{x}_i$, while $\hat{y}_i$ is the output produced by the reference regression function, that is, $\hat{y}_i = f_\circ(\boldsymbol{x}_i)$. The black and red lines show the ith term of the original objective function $F_H(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$ and the "smoothed" version $F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$, respectively. The shape of the red curve relative to the black curve depends on the value of β and on the distance between $\hat{y}_i$ and y_i (which is set to 1 here for illustrative purposes).

In order to find a maximum of the objective function $F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$ we consider, in a first approach, a gradient ascent scheme. That is, we start from an initial guess $\boldsymbol{a}^{(0)}$ of the parameters $\boldsymbol{a}$ and update them iteratively according to

$$\boldsymbol{a}^{(i+1)} = \boldsymbol{a}^{(i)} + \eta \cdot \boldsymbol{\nabla}_{\boldsymbol{a}}\, F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})\big|_{\boldsymbol{a}^{(i)}} = \boldsymbol{a}^{(i)} + \eta \cdot \big(\boldsymbol{\nabla}_{\boldsymbol{a}}\, F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})\big)\big(\boldsymbol{a}^{(i)}\big),$$

where η is a step width parameter that has to be chosen by a user. In order to be able to evaluate this expression, we first compute

$$L'_\beta(z) \;=\; \frac{\mathrm{d}}{\mathrm{d}z} L_\beta(z) \;=\; \frac{\mathrm{d}}{\mathrm{d}z} \left(1 + e^{-\beta z}\right)^{-1} \;=\; \beta \cdot L_\beta(z) \cdot (1 - L_\beta(z)),$$

the well-known expression for the derivative of the logistic function, as well as

$$\boldsymbol{\nabla}_{\boldsymbol{a}}\, g_\pm(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \pm\varepsilon) = \boldsymbol{\nabla}_{\boldsymbol{a}}\, (f_*(\boldsymbol{x}_i; \boldsymbol{a}) - (y_i \pm (1-\varepsilon) \cdot |f_\circ(\boldsymbol{x}_i) - y_i|)) = \boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i; \boldsymbol{a}).$$

This leads to the following gradient of the objective function

$$\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{a}} F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) &= \boldsymbol{\nabla}_{\boldsymbol{a}} \sum_{i=1}^{n} \Big(L_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big) \\
&= \sum_{i=1}^{n} \big(\boldsymbol{\nabla}_{\boldsymbol{a}}\, L_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - \boldsymbol{\nabla}_{\boldsymbol{a}}\, L_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \big) \\
&= \sum_{i=1}^{n} \Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big) \cdot \boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i; \boldsymbol{a}).
\end{aligned}$$

If we consider the special case $f_*(\boldsymbol{x}, \boldsymbol{a}) = \boldsymbol{a}^\top \boldsymbol{x}^*$, where it is $\boldsymbol{a} = (a_0, a_1, \ldots, a_m)$ and $\boldsymbol{x}^* = (1, x_1, \ldots, x_m)$ for $\boldsymbol{x} = (x_1, \ldots, x_m)$, that is, a linear function, we get

$$\boldsymbol{\nabla}_{\boldsymbol{a}}\, f_*(\boldsymbol{x}; \boldsymbol{a}) \;=\; \boldsymbol{\nabla}_{\boldsymbol{a}}\, \boldsymbol{a}^\top \boldsymbol{x}^* \;=\; \boldsymbol{x}^* \qquad \text{and therefore}$$

$$\boldsymbol{\nabla}_{\boldsymbol{a}}\, F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) \;=\; \sum_{i=1}^{n} \Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big) \cdot \boldsymbol{x}_i^*.$$

As a second approach we may draw on gradient method improvements as they have been developed, for example, in artificial neural networks, There is a large variety of such approaches like the introduction of a momentum term [11], Nesterov's accelerated gradient [10], self-adaptive error backpropagation [15], resilient error backpropagation [12,13], quick backpropagation [6], AdaGrad [5], RMSProp [14], AdaDelta [16], Adam and AdaMax [9], and NAdam [4]. For simplicity, we choose one of them, namely resilient backpropagation, which uses the signs of the current and the previous gradient to adapt a parameter step width, one for each parameter. That is, the general update scheme is $a_k^{(i+1)} = a_k^{(i)} + \Delta a_k^{(i)}$ and the step width $\Delta a_k^{(i)}$ (which is specific to parameter a_k) is computed as

$$\Delta a_k^{(i)} = \begin{cases} c^- \cdot \Delta a_k^{(i-1)}, & \text{if } (\boldsymbol{\nabla}_{a_k} F_{L_\beta})\big|_{a_k^{(i)}} \cdot (\boldsymbol{\nabla}_{a_k} F_{L_\beta})\big|_{a_k^{(i-1)}} < 0, \\ c^+ \cdot \Delta a_k^{(i-1)}, & \text{if } (\boldsymbol{\nabla}_{a_k} F_{L_\beta})\big|_{a_k^{(i)}} \cdot (\boldsymbol{\nabla}_{a_k} F_{L_\beta})\big|_{a_k^{(i-1)}} < 0 \\ & \wedge\ (\boldsymbol{\nabla}_{a_k} F_{L_\beta})\big|_{a_k^{(i-1)}} \cdot (\boldsymbol{\nabla}_{a_k} F_{L_\beta})\big|_{a_k^{(i-2)}} \geq 0, \\ \Delta a_k^{(i-1)}, & \text{otherwise.} \end{cases}$$

That is, the step width is increased if the current and the previous gradient point in the same direction, and it is decreased if they point in opposite directions, thus indicating that an optimum was leaped over. We chose the growth factor $c^+ = 1.2$ and the shrink factor $c^- = 0.5$, which is a typical choice.

As a third approach one may carry out a root search on the gradient, for example, by applying the Newton–Raphson method to the gradient. In this case the parameter update rule is generally (note: no step width parameter η)

$$\boldsymbol{a}^{(i+1)} = \boldsymbol{a}^{(i)} - \left(\boldsymbol{\nabla}_{\boldsymbol{a}}^2 F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})\big|_{\boldsymbol{a}^{(i)}}\right)^{-1} \cdot \left(\boldsymbol{\nabla}_{\boldsymbol{a}} F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})\big|_{\boldsymbol{a}^{(i)}}\right).$$

For this we first compute the second derivative of the logistic function:

$$\begin{aligned} L''_\beta(z) = \frac{\mathrm{d}}{\mathrm{d}z} L'_\beta(z) &= \frac{\mathrm{d}}{\mathrm{d}z} \left(\beta \cdot L_\beta(z) \cdot (1 - L_\beta(z))\right) \\ &= \beta^2 \cdot L_\beta(z) \cdot (1 - L_\beta(z)) \cdot (1 - 2L_\beta(z)), \end{aligned}$$

With this result we compute the second derivative of the objective function as

$$\begin{aligned} &\boldsymbol{\nabla}_{\boldsymbol{a}}^2 F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) \\ &= \boldsymbol{\nabla}_{\boldsymbol{a}} \left(\sum_{i=1}^{n} \Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big)\Big) \cdot \boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i; \boldsymbol{a}) \right) \\ &= \sum_{i=1}^{n} \Big(\quad \Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big)\Big) \cdot \boldsymbol{\nabla}_{\boldsymbol{a}}^2 f_*(\boldsymbol{x}_i; \boldsymbol{a}) \\ &\qquad + \boldsymbol{\nabla}_{\boldsymbol{a}} \Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big)\Big) \cdot \big(\boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i; \boldsymbol{a})\big)^\top \Big) \\ &= \sum_{i=1}^{n} \Big(\Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big)\Big) \cdot \boldsymbol{\nabla}_{\boldsymbol{a}}^2 f_*(\boldsymbol{x}_i; \boldsymbol{a}) \\ &\qquad + \Big(L''_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L''_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big)\Big) \cdot \boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i; \boldsymbol{a}) \cdot \big(\boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i; \boldsymbol{a})\big)^\top \Big). \end{aligned}$$

If we consider the special case $f_*(\boldsymbol{x}, \boldsymbol{a}) = \boldsymbol{a}^\top \boldsymbol{x}^*$, where it is $\boldsymbol{a} = (a_0, a_1, \ldots, a_m)$ and $\boldsymbol{x}^* = (1, x_1, \ldots, x_m)$ for $\boldsymbol{x} = (x_1, \ldots, x_m)$, that is, a linear function, we get

$$\boldsymbol{\nabla}_{\boldsymbol{a}}^2 F_{L_\beta}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) = \sum_{i=1}^{n} \Big(L''_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L''_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big)\Big) \cdot \boldsymbol{x}_i^* \boldsymbol{x}_i^{*\top},$$

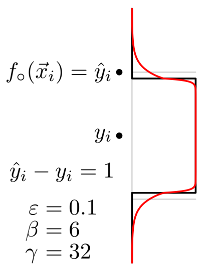


Fig. 3. The value y_i is the true output for input vector $\boldsymbol{x}_i$, while $\hat{y}_i$ is the output produced by the reference regression function, that is, $\hat{y}_i = f_\circ(\boldsymbol{x}_i)$. The black and red lines show the ith term of the original objective function $F_H(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$ and the "smoothed" version $F_{R_\beta^\gamma}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y})$, respectively. The shape of the red curve relative to the black curve depends on the value of β, but *not* on the distance between $\hat{y}_i$ and y_i (which is set to 1 here for illustrative purposes).

since $\boldsymbol{\nabla}_{\boldsymbol{a}}^2\, \boldsymbol{a}^\top \boldsymbol{x}^* = \mathbf{0}$ where $\mathbf{0}$ is the null matrix. That is, the update rule reads

$$\boldsymbol{a}^{(i+1)} = \boldsymbol{a}^{(i)} - \left(\sum_{i=1}^{n} \Big(L''_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L''_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big) \cdot \boldsymbol{x}_i^* \boldsymbol{x}_i^{*\top} \right)^{-1} \cdot \left(\sum_{i=1}^{n} \Big(L'_\beta\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - L'_\beta\big(g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) \Big) \cdot \boldsymbol{x}_i^* \right).$$

An alternative way of "smoothing" the Heaviside function is

$$R_\beta^\gamma(z) = \begin{cases} \frac{1}{2} e^{\beta z}, & \text{if } z < 0, \\ 1 - \frac{1}{2} e^{-\gamma z}, & \text{if } z \geq 0. \end{cases}$$

Here the flank of the Heaviside function is replaced by two exponential functions, where β and γ are steepness parameters. For $\beta, \gamma \to \infty$ we get $R_\beta^\gamma(z) \to H(z)$. The difference to $L_\beta(z)$ is that for $\beta \neq \gamma$ this function is not continuously differentiable at $z = 0$. However, as this is only a single point, this appears to be acceptable and seems to work in practice (see experiments below).

The underlying idea is to have a stronger gradient outside the "win region", and a smaller one inside it (except close to the boundaries), so that an improvement inside the "win region" (pushing the prediction closer to the center) for one point cannot easily compensate another point not being inside this region.

This alternative smoothing approach leads to the objective function

$$F_{R_\beta^\gamma}(\boldsymbol{a}; \mathbf{X}, \boldsymbol{y}) = \sum_{i=1}^{n} \Big(R_\beta^\gamma\big(g_-(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) + R_\beta^\gamma\big(-g_+(\boldsymbol{a}; \boldsymbol{x}_i, y_i, \varepsilon)\big) - 1 \Big),$$

which is again to be maximized by choosing $\boldsymbol{a}$. Figure 3 provides an illustration of one term of the original and the smoothed objective function.

In order to conduct the optimization, we need the derivative of $R_\beta^\gamma(z)$, i.e.

$$R_\beta^{\gamma\prime}(z) = \frac{\mathrm{d}}{\mathrm{d}z} R_\beta^\gamma(z) = \begin{cases} \frac{\mathrm{d}}{\mathrm{d}z} \frac{1}{2} e^{\beta z} & = \beta \frac{1}{2} e^{\beta z} = \beta R_\beta^\gamma(z), & \text{if } z < 0, \\ \frac{\mathrm{d}}{\mathrm{d}z} (1 - \frac{1}{2} e^{-\gamma z}) & = \gamma \frac{1}{2} e^{-\gamma z} = \gamma(1 - R_\beta^\gamma(z)), & \text{if } z \geq 0, \end{cases}$$

if we use the right hand derivative ($\neq$ left hand derivative) at $z = 0$. We obtain

$$\begin{aligned}
&\boldsymbol{\nabla}_{\boldsymbol{a}}\ F_{R_\beta^\gamma}(\boldsymbol{a};\mathbf{X},\boldsymbol{y})\\
&= \boldsymbol{\nabla}_{\boldsymbol{a}} \sum_{i=1}^{n} \Big(R_\beta^\gamma\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) + R_\beta^\gamma\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - 1\Big)\\
&= \sum_{i=1}^{n} \Big(\boldsymbol{\nabla}_{\boldsymbol{a}}\ R_\beta^\gamma\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) + \boldsymbol{\nabla}_{\boldsymbol{a}}\ R_\beta^\gamma\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - \boldsymbol{\nabla}_{\boldsymbol{a}} 1\Big)\\
&= \sum_{i=1}^{n} \Big(R_\beta^{\gamma\prime}\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - R_\beta^{\gamma\prime}\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big) \cdot \boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a}).
\end{aligned}$$

For a Newton-Raphson approach we also need the second derivative of $R_\beta(z)$:

$$R_\beta^{\gamma\prime\prime}(z) = \frac{\mathrm{d}}{\mathrm{d}z} R_\beta^{\gamma\prime}(z) = \begin{cases} \frac{\mathrm{d}}{\mathrm{d}z}\beta\frac{1}{2}e^{\beta z} &= \beta^2\frac{1}{2}e^{\beta z} &= \beta^2 R_\beta(z), & \text{if } z < 0,\\ \frac{\mathrm{d}}{\mathrm{d}z}\gamma\frac{1}{2}e^{-\gamma z} &= -\gamma^2\frac{1}{2}e^{-\gamma z}, &= \gamma^2(R_\beta^\gamma(z) - 1), & \text{if } z \ge 0. \end{cases}$$

Again we use the right hand side derivative at $z = 0$, where the derivative $R'_\beta(z)$ itself is not even continuous. For the objective function this leads to

$$\begin{aligned}
&\boldsymbol{\nabla}^2_{\boldsymbol{a}} F_{R_\beta}(\boldsymbol{a};\mathbf{X},\boldsymbol{y})\\
&= \boldsymbol{\nabla}_{\boldsymbol{a}}\Bigg(\sum_{i=1}^{n}\Big(R'_\beta\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - R'_\beta\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big)\cdot\boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a})\Bigg)\\
&= \sum_{i=1}^{n}\Bigg(\Big(R'_\beta\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - R'_\beta\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big)\cdot\boldsymbol{\nabla}^2_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a})\\
&\qquad + \boldsymbol{\nabla}_{\boldsymbol{a}}\Big(R'_\beta\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - R'_\beta\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big)\cdot\big(\boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a})\big)^\top\Bigg)\\
&= \sum_{i=1}^{n}\Bigg(\Big(R'_\beta\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) - R'_\beta\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big)\cdot\boldsymbol{\nabla}^2_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a})\\
&\qquad + \Big(R''_\beta\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) + R''_\beta\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big)\cdot\boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a})\cdot\big(\boldsymbol{\nabla}_{\boldsymbol{a}} f_*(\boldsymbol{x}_i;\boldsymbol{a})\big)^\top\Bigg).
\end{aligned}$$

If we consider the special case $f_*(\boldsymbol{x},\boldsymbol{a}) = \boldsymbol{a}^\top\boldsymbol{x}^*$, where it is $\boldsymbol{a} = (a_0, a_1, \ldots, a_m)$ and $\boldsymbol{x}^* = (1, x_1, \ldots, x_m)$ for $\boldsymbol{x} = (x_1, \ldots, x_m)$, that is, a linear function, we get

$$\boldsymbol{\nabla}^2_{\boldsymbol{a}}\ F_{R_\beta}(\boldsymbol{a};\mathbf{X},\boldsymbol{y}) \ = \ \sum_{i=1}^{n}\Big(R''_\beta\big(g_-(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big) + R''_\beta\big(-g_+(\boldsymbol{a};\boldsymbol{x}_i,y_i,\varepsilon)\big)\Big)\cdot\boldsymbol{x}_i^*\boldsymbol{x}_i^{*\top},$$

since $\boldsymbol{\nabla}^2_{\boldsymbol{a}}\ \boldsymbol{a}^\top\boldsymbol{x}^* = \mathbf{0}$ where $\mathbf{0}$ is the null matrix.

5 Experiments

We implemented our gradient ascent approach in both Python and C, but used only the C implementation for the experiments (due to its much shorter execution time).[4] We implemented both "smoothed" versions of the objective function

[4] These implementations are publicly available at `www.borgelt.net/brreg.html`.

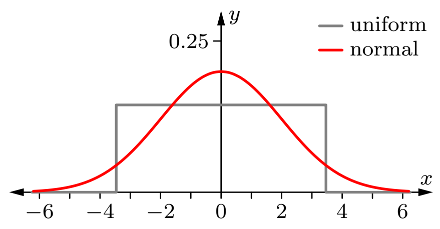
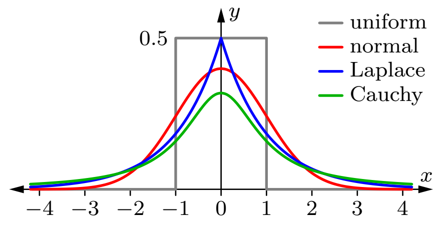
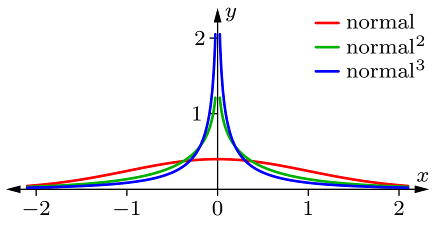


Fig. 4. The distributions from which the x-values were sampled (left diagram) and the distributions from which the error ϵ was sampled (bottom diagrams). The distributions normal^2 and normal^3 are obtained by multiplying two or three samples, resp., from a normal distribution.





that are described in the preceding section, using a minimum prediction improvements $\varepsilon \in \{0.001, 0.01, 0.1\}$ and steepnesses $\beta \in \{1, 3, 6, 10\}$ and $\gamma = 32$.

We confined ourselved to a very simple univariate linear regression with the model $y = x + \epsilon$, where ϵ is a noise (or error) term. Although our approach is, in principle, applicable to multivariate and non-linear regression, our focus was more on understanding how different types of noise (or error) distributions affect the regression performance, for which univariate linear regression appears sufficient. Note that in [2] no systematic investigation is conducted with the help of simulated data. Hence that paper does not provide any information about the influence of different types of noise (or error) distributions.

We drew the x-values of all data samples (with $n \in \{20, 50, 100\}$ data points) from either a uniform or a normal distribution (Figure 4 top left; both have a standard deviation of 2) and the noise ϵ from either a uniform, a normal, a Laplace, or a Cauchy distribution, or computed it as a product of two or three samples from a normal distribution (Figure 4 bottom). All distributions were parameterized with a dispersion parameter of 1. Note that a Cauchy distribution has no finite variance (its dispersion parameter is half the interquartile range), which is why it is often used to model extremal events [3].

Since we use a simple linear model as the ground truth to sample from, it is natural to use ordinary least squares (OLS) linear regression as the reference. As a baseline for comparisons we chose robust linear regression based on M-estimators [7, 8] using Tukey's bisquare (or biweights) function [1] for the error weights, since it is geared towards providing better parameter estimates for heavy-tailed noise distributions, which is what we wanted to investigate.

As a baseline for the optimization, we used a bootstrap sampling scheme, in which 1000 bootstrap samples were drawn from the given data, an OLS regression computed for each, and the one that "won" the most data points compared to OLS regression on all data points chosen as the result. The advantage of such a scheme is that it can also be applied for the original (non-smoothed) objective function F_H. Furthermore we used standard gradient ascent and resilient

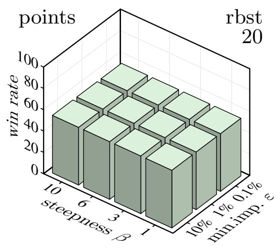
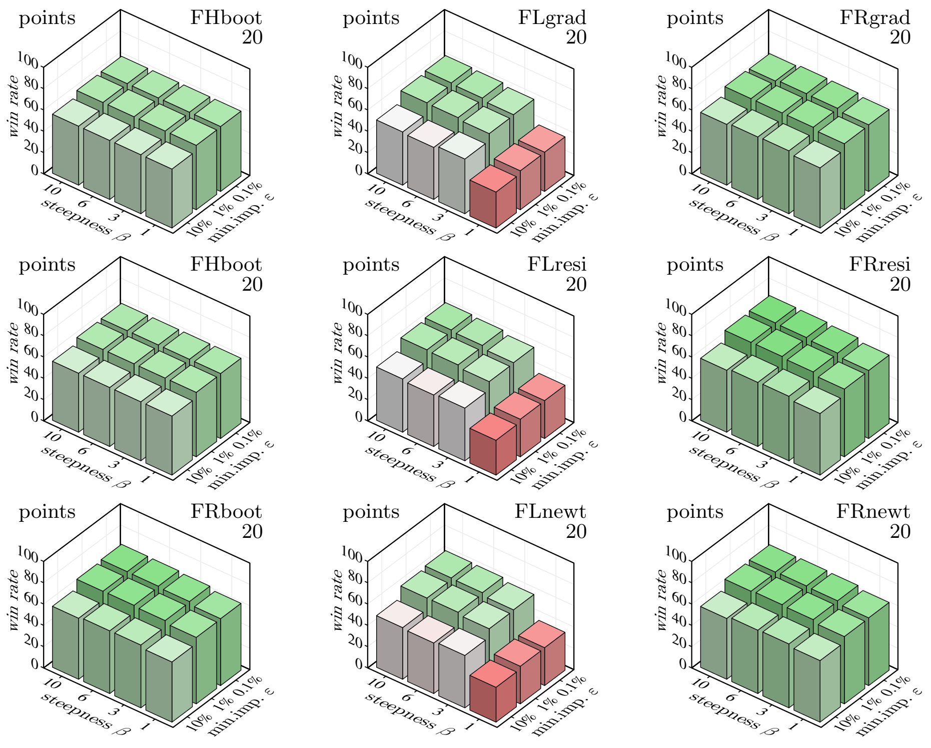


Fig. 5. Better prediction win rate on test data relative to ordinary least squares regression for 20 data points (x-values drawn from a uniform distribution, error is the product of two samples from a normal distribution). Left: classical robust regression, below: our approaches to best response regression ("FH", "FL", "FR": objective function; "boot", "grad", "resi", "newt": optimization method).



backpropagation with a(n initial) step with of $\eta = 0.001$, growth factor $c^+ = 1.2$ and shrink factor $c^- = 0.5$; and the Newton-Raphson method. All optimization methods were executed for 200 iterations and final parameters returned.

A selection of experimental results is shown in Figures 5 to 7, all of which are computed from 10 000 runs. For a comparison of the different objective functions and optimization approaches, we chose data with 20 data points, x-values drawn from a uniform distribution and noise computed as the product of two samples from a uniform distribution. Figures 5 and 6 show the win rate, that is, the percentage of sets of unseen data on which the objective function/method pair indicated at the top right of each diagram performed better than OLS regression, that is, won more points in Figure 5 or provided a lower sum of squared errors (SSE) in Figure 6. Note that in the diagram at the top left (robust regression) all bars have the same height (for easier comparison) as the parameters have no

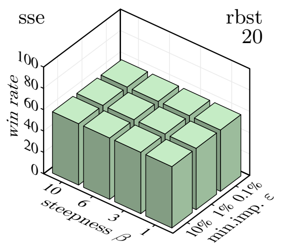
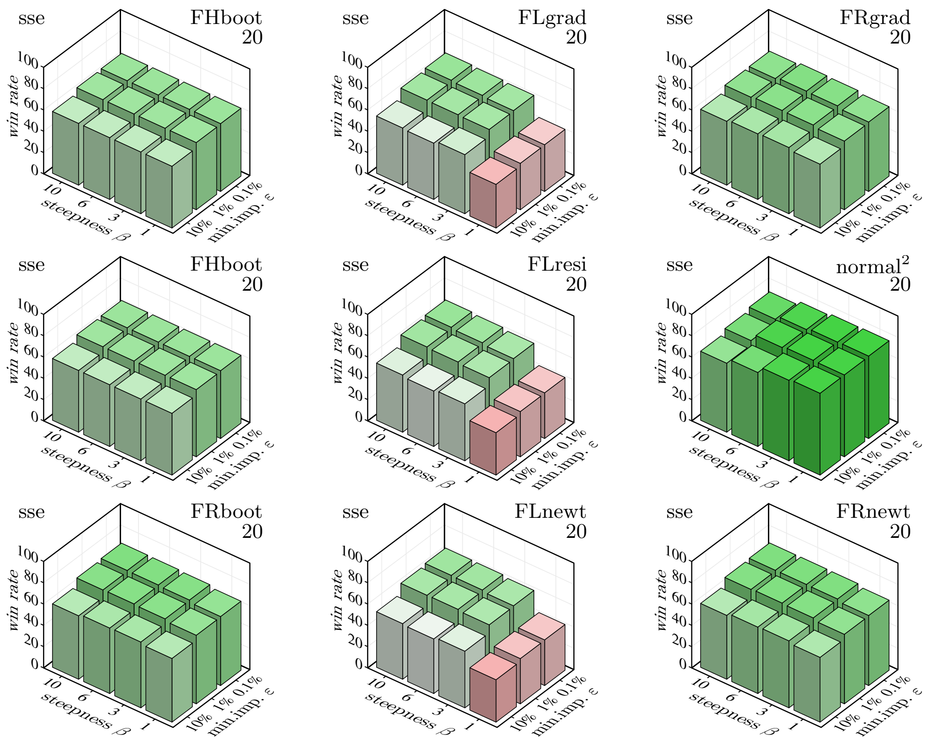


Fig. 6. Sum of squared errors win rate on test data relative to ordinary least squares regression for 20 data points (x-values drawn from a uniform distribution, error is the product of two samples from a normal distribution). Left: classical robust regression, below: our approaches to best response regression ("FH", "FL", "FR": objective function; "boot", "grad", "resi", "newt": optimization method).



influence. Note that Best Response Regression, at least with proper parameters, outperforms robust regression, especially with objective function F_R. Objective function F_L works slightly less well and requires a large steepness β.

Figure 7 shows the performance of best response regression for different noise distributions. For uniform noise and normally distributed noise, ordinary least squares (OLS) is clearly superior to best response regression, but for Laplace-distributed noise it can just edge out OLS (provided the required minimum prediction improvement ε is small). However, for noise computed as a product of two or three samples from a normal distribution and particularly for Cauchy-distributed noise, Best Response Regression clearly has the upper hand. As the full result diagrams[5] show, Best Response Regression can, for these distributions, usually also beat robust regression, although not always by a wide margin.

[5] All result diagrams are available at `www.borgelt.net/docs/brreg.pdf`.

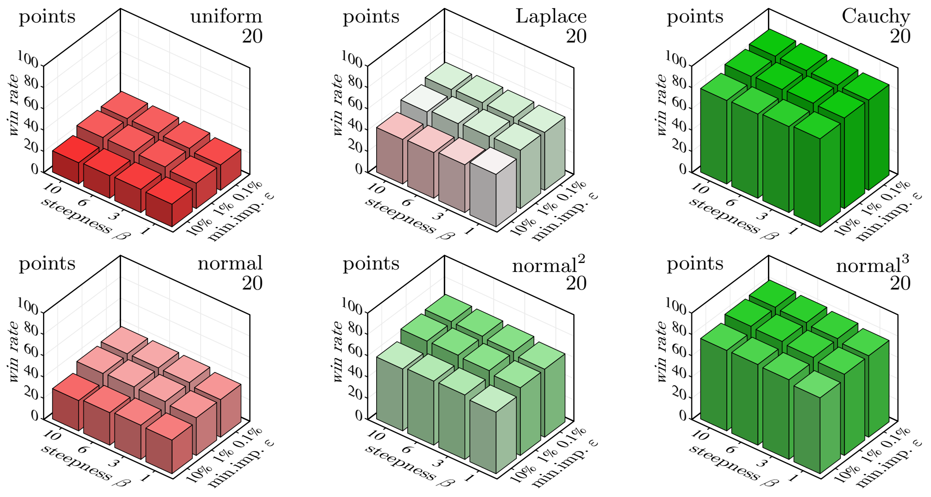


Fig. 7. Better prediction win rate on test data relative to ordinary least squares regression for 20 data points (x-values drawn from a uniform distribution) for different error distributions (indicated on top right, "normal^2" and "normal^3" refer to an error computed as a product of two or three samples from a normal distribution).

6 Conclusions

In our work we limited experiments and simulations to linear models. However, our gradient ascent approach for best response regression contains no assumptions about the regression function. In principle, our approach also works for, e.g., polynomial or logistic regression functions, which is an improvement over [2]. Our simulations show that best response regression is a very promising approach to optimize classical regression models. Moreover, our gradient ascent approach does not require any distribution assumptions for the error terms or the explanatory variables and therefore, our approach is less restrictive than classical regression models. Obviously, if there is a very strong linear dependence between the explanatory and target variables, it is very difficult for the best response regression to beat the "expert" (classical regression model, reference function). The other way round, if the "expert" performs badly, best response regression has far more potential to maximize the probability to predict better than the expert. In addition, best response regression has a clear advantage on small sample size data sets. Our experiments showed as a rule of thumb $n < 50$. With larger sample sizes, classical statistical methods have a greater advantage due to the underlying asymptotic results. Nevertheless, we would like to emphasize that with this work we could show that well known regression tasks, such as linear regression, can be reinterpreted. Ben-Porat and Tennenholtz motivated Best Response Regression as a game between an expert and an agent. In some way, we can interpret our new approach as optimizing linear regression predictions.

Acknowledgments The second author gratefully acknowledges the financial support from Land Salzburg within the WISS 2025 project IDA-Lab (20102-F1901166-KZP and 20204-WISS/225/197-2019).

References

1. A.E. Beaton and J.W. Tukey. The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-spectroscopic Data. *Technometrics* 16:147–185. American Society for Quality and the American Statistical Association, Milwaukee, WI and Boston, MA, USA 1974
2. O. Ben-Porat and M. Tennenholtz. Best Response Regression. *Advances in Neural Information Processing Systems 30 (NIPS 2017, Long Beach, CA)*, 1499–1508. Curran Associates, Red Hook, NY, USA 2017
3. P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling Extremal Events: For Insurance and Finance.* Springer Science & Business Media, Berlin, Germany 2013
4. T. Dozat. Incorporating Nesterov Momentum into Adam. *Proc. Int. Conf. on Learning Representations (ICLR Workshop 2016, San Juan, Puerto Rico).* `openreview.net` 2016
5. J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12:2121–2159. Microtome Publishing, Brookline, MA, USA 2011
6. S.E. Fahlman. An Empirical Study of Learning Speed in Backpropagation Networks. *Proc. of the Connectionist Models Summer School (Carnegie Mellon University).* Morgan Kaufman, San Mateo, CA, USA 1988
7. P.J. Huber. Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics* 35:73–101. Institute of Mathematical Statistics, 1964
8. P.J. Huber. *Robust Statistics.* J. Wiley & Sons, New York, NY, USA, 1981
9. D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *Proc. Int. Conf. on Learning Representations (ICLR 2015, San Diego, CA).* `openreview.net` 2015
10. Y.E. Nesterov. A Method of Solving a Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Soviet Mathematics Doklady* 27(2):372–376. 1983
11. B.T. Polyak. Some Methods of Speeding Up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, 1964
12. M. Riedmiller and H. Braun. Rprop — A Fast Adaptive Learning Algorithm. Technical Report, University of Karlsruhe, Karlsruhe, Germany 1992
13. M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *Int. Conf. on Neural Networks (ICNN-93, San Francisco, CA)*, 586–591. IEEE Press, Piscataway, NJ, USA 1993
14. T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning* 4. 2012
15. T. Tollenaere. SuperSAB: Fast Adaptive Backpropagation with Good Scaling Properties. *Neural Networks* 3:561–573, 1990
16. M.D. Zeiler. AdaDelta: An Adaptive Learning Rate Method. `arXiv:1212.5701`, 2012