Smooth ε-Insensitive Regression by Loss Symmetrization

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer

School of Computer Science, Hebrew University, Jerusalem, Israel
{oferd,shais,singer}@cs.huji.ac.il

Abstract. We describe a framework for solving regression problems by reduction to classification. Our reduction is based on symmetrization of margin-based loss functions commonly used in boosting algorithms, namely, the logistic loss and the exponential loss. Our construction yields a smooth version of the ε-insensitive hinge loss that is used in support vector regression. A byproduct of this construction is a new simple form of regularization for boosting-based classification and regression algorithms. We present two parametric families of batch learning algorithms for minimizing these losses. The first family employs a log-additive update and is based on recent boosting algorithms while the second family uses a new form of additive update. We also describe and analyze online gradient descent (GD) and exponentiated gradient (EG) algorithms for the ε-insensitive logistic loss. Our regression framework also has implications on classification algorithms, namely, a new additive batch algorithm for the log-loss and exp-loss used in boosting.

1 Introduction

The focus of the paper is supervised learning of real-valued regression functions. In the settings we discuss in this paper, we observe a sequence $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of instance-target pairs. For concreteness, we assume that the instances are vectors in $\mathbb{R}^n$ and that the targets are real-valued scalars, $y_i \in \mathbb{R}$. We denote the j'th component of an instance $\mathbf{x}_i$ by $x_{i,j}$. Our goal is to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ which provides a good approximation of target values from their corresponding instance vectors. Such a function is often referred to as a regression function or a regressor for short. In this paper we focus on learning linear regressors, that is, f is of the form $f(\mathbf{x}) = \boldsymbol{\lambda} \cdot \mathbf{x}$. This setting is also suitable for learning a linear combination of base regressors of the form $f(\mathbf{x}) = \sum_{j=1}^{l} \lambda_j h_j(\mathbf{x}) = \boldsymbol{\lambda} \cdot \mathbf{h}(\mathbf{x})$ where $h_j : X \to \mathbb{R}$, X is an instance domain, and $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_l(\mathbf{x}))$. The latter form enables us to employ kernels by setting $h_j(\mathbf{x}) = K(\mathbf{x}_j, \mathbf{x})$. Since the class of regressors we consider is rather restricted and due to the existence of noise, a perfect mapping such that for all $(\mathbf{x}_i, y_i) \in S$, $f(\mathbf{x}_i) = y_i$ might not exist. Hence, we employ a loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ which measures the discrepancy between the *predicted* target, $f(\mathbf{x})$, and the *true* (observed) target y. As we discuss shortly, the loss functions we consider in this paper depend only on the discrepancy $\delta = f(\mathbf{x}) - y$ hence

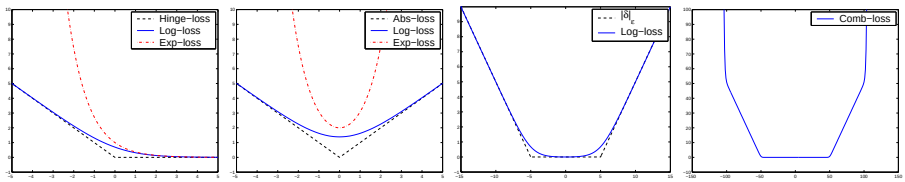

Fig. 1. Constructing regression losses (second-left) by symmetrization of margin losses (left). The smooth ε-insensitive log-loss (second-right) and the combined loss (right).

L can be viewed as a function from $\mathbb{R}$ into $\mathbb{R}_+$ and is denoted by $L(\delta)$. Given a loss function L, the goal of a regression algorithm is to find a regressor f which attains a small *cumulative* loss on the training set S,

$$\text{Loss}(\boldsymbol{\lambda}, S) \;=\; \sum_{i=1}^{m} L(f(\mathbf{x}_i) - y_i) \;=\; \sum_{i=1}^{m} L(\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i) \quad .$$

Regression problems have long been the focus of many research papers in statistics and learning theory. See for instance the book by Hastie, Tibshirani, and Friedman [6] and the references therein. Denote the discrepancy $\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i$ by δ_i. Two common approaches for regression minimize either the sum of the absolute discrepancies over the sample ($\sum_i |\delta_i|$) or the square of the discrepancies ($\sum_i \delta_i^2$). It has been argued that the squared loss is sensitive to outliers, hence robust regression algorithms often employ the absolute loss [7].

Furthermore, it is often the case that the *exact* discrepancy between $\boldsymbol{\lambda} \cdot \mathbf{x}$ and y is unimportant so long as this discrepancy falls below an insensitivity parameter ε. Formally, the ε-insensitive hinge loss, denoted $|\delta|_\varepsilon$, is zero if $|\delta| \leq \varepsilon$ and is $|\delta| - \varepsilon$ for $|\delta| > \varepsilon$ (see also Fig. 1). Whether $\varepsilon = 0$ or not, the ε-insensitive hinge loss is not smooth as its derivative is discontinuous at $\delta = \pm\varepsilon$. While the non-smooth nature of the ε-insensitive hinge loss led to the design and analysis of efficient batch learning algorithms (cf. [14, 13]), it also poses algorithmic difficulties. In this paper we discuss and analyze a smooth approximation of the ε-insensitive hinge loss. Formally, we define the following loss,

$$L_{\log}(\delta; \varepsilon) = \log\left(1 + e^{\delta - \varepsilon}\right) + \log\left(1 + e^{-\delta - \varepsilon}\right) + \kappa \quad . \tag{1}$$

Whenever it is clear from context we simply denote the loss as $L_{\log}(\delta)$. We term this loss the *symmetric ε-insensitive logistic loss* and for brevity, we refer to it simply as the log-loss. The constant κ equals $2\log(1 + e^{-\varepsilon})$ and is set so that $L_{\log}(0) = 0$. Since additive constants do not change the form of the optimal regressor we henceforth omit this constant. In Fig. 1 we depict the above loss along with the ε-insensitive hinge loss for $\varepsilon = 5$. Note that the ε-insensitive log-loss provides a smooth upper bound on the ε-insensitive hinge loss, and that with this particular choice of ε we get that for $|\delta| < 2$ and $|\delta| > 8$ the smooth and non-smooth ε-insensitive losses are graphically indistinguishable.

To motivate our construction, let us make a short detour and discuss a recent view of classification algorithms. In the binary classification setting discussed

in [5, 2, 10], we are provided with *instance-label* pairs, $(\mathbf{x}, y)$, where, in contrast to regression, each label takes one of two values, namely $y \in \{-1, +1\}$. A real-valued classifier is a function f into the reals such that $\mathrm{sign}(f(\mathbf{x}))$ is the predicted label and the magnitude, $|f(\mathbf{x})|$, is the confidence of f in its prediction. The product $yf(\mathbf{x})$ is called the (signed) margin of the instance-label pair $(\mathbf{x}, y)$. The goal of a margin-based classifier is to attain large margin values on as many instances as possible. Learning algorithms for margin-based classifiers typically employ a margin-based loss function $L_{\mathrm{C}}(yf(\mathbf{x}))$ and attempt to minimize the cumulative loss over a given sample. One of the margin losses discussed is the logistic loss, that takes the form

$$L_{\mathrm{C}}(yf(\mathbf{x})) = \log\left(1 + e^{-yf(\mathbf{x})}\right) \quad . \tag{2}$$

We use the loss in Eq. (2) as a building block in the construction of the regression loss (Eq. (1)). Denote by $[\mathbf{u}\,;\,v]$ the concatenation of an additional element v to the end of a vector $\mathbf{u}$. We replace every instance-*target* pair $(\mathbf{x}, y)$ from the regression problem with *two* classification instance-*label* pairs,

$$(\mathbf{x}, y) \;\mapsto\; \begin{cases} ([\mathbf{x}\,;\,-y+\varepsilon]\,,\,+1) \\ ([\mathbf{x}\,;\,-y-\varepsilon]\,,\,-1) \end{cases} \quad .$$

In words, we duplicate each regression instance and create two classification instances. We then increase the dimension by one and concatenate $-y+\varepsilon$ to the first newly created instance and set its label to $+1$. Symmetrically, we concatenate $-y-\varepsilon$ to the second duplicate and set its label to -1. We define the linear *classifier* to be the vector $[\boldsymbol{\lambda}\,;\,1] \in \mathbb{R}^{n+1}$. It is simple to verify that,

$$L_{\log}(\boldsymbol{\lambda}\cdot\mathbf{x} - y\,;\,\varepsilon) = L_{\mathrm{C}}([\boldsymbol{\lambda}\,;\,1]\cdot[\mathbf{x}\,;\,-y+\varepsilon]) + L_{\mathrm{C}}(-[\boldsymbol{\lambda}\,;\,1]\cdot[\mathbf{x}\,;\,-y-\varepsilon]) \quad .$$

In Fig. 1 we give an illustration of the above construction. We have thus reduced a regression problem of m instances in $\mathbb{R}^n$ and targets in $\mathbb{R}$ to a classification problem with $2m$ instances in $\mathbb{R}^{n+1}$ and binary labels.

The work in [2] gave a unified view of two margin losses: the logistic loss defined by Eq. (2) and an exponential loss. An immediate benefit of our construction is a similar unified account of the two respective regression losses. Formally, we define the *symmetric exponential loss* for regression as follows,

$$L_{\exp}(\delta) \;=\; e^{\delta} + e^{-\delta} \quad .$$

As for the log-loss, we simply refer to this loss as the exp-loss. The exp-loss was first presented and analyzed by Duffy and Helmbold [3] in their pioneering work on leveraging regressors. Their view though is somewhat different as it builds upon the notion of weak-learnability, yielding a different (sequential) algorithm for regression. The exp-loss is by far less forgiving than the log-loss, i.e. small discrepancies are amplified exponentially. While this property might be undesirable in regression problems with numerous outliers, it can also serve as a barrier that prevents the existence of any large discrepancy on the training set.

A nice property of this loss is that it provides a bound on the maximal absolute discrepancy in the sample. To see this, note that the minimizer of $\sum_i L_{\exp}(\delta_i)$ is also the minimizer of $\log(\sum_i L_{\exp}(\delta_i))$ which is a smooth approximation to $\max_i |\delta_i|$. We can also combine the log-loss and the exp-loss with two different insensitivity parameters and benefit both from a discrepancy insensitivity region and from enforcing a smooth barrier on the maximal discrepancy. Formally, let $\varepsilon_1 > 0$ and $\varepsilon_2 > \varepsilon_1$ be two insensitivity parameters. We define the *combined loss*, abbreviated as *comb-loss*, by $L_{\text{comb}}(\delta; \varepsilon_1, \varepsilon_2) = L_{\log}(\delta; \varepsilon_1) + L_{\exp}(\delta; \varepsilon_2)$, where $L_{\exp}(\delta; \varepsilon_2) = e^{-\varepsilon_2} L_{\exp}(\delta)$. An illustration of the combined loss with $\varepsilon_1 = 50$ and $\varepsilon_2 = 100$ is given in Fig. 1.

The paper is organized as follows. In Sec. 2 we describe a simple use of the symmetric losses defined above as a means of regularizing $\boldsymbol{\lambda}$. In Sec. 3 we describe and analyze a family of log-additive update algorithms for batch learning settings. The algorithms are derived using the reduction outlined above and by adapting proof techniques from [2] to our setting. In Sec. 4 we describe a new family of additive update regression algorithms based on modified gradient descent. For both the log-additive and the additive updates, we provide a boosting-style analysis of the decrease in loss. In Sec. 5 we shift our attention to *online* learning algorithms for the ε-insensitive log-loss. Our algorithms and regularization technique have implications on *classification* algorithms with both the log-loss and the exp-loss. We briefly discuss these implications, illustrate the merits of the two losses and conclude in Sec. 6.

2 Regularization

Regularization is a means of controlling the complexity of the regressor being learned. In particular for linear regressors, regularization serves as a soft limit on the magnitude of the elements of $\boldsymbol{\lambda}$ (cf. [11]). The losses we discussed in the previous section can provide a new form of regularization. For the log-loss, the regularization applied to the j'th coordinate of $\boldsymbol{\lambda}$ is, $\log\left(1 + e^{\lambda_j}\right) + \log\left(1 + e^{-\lambda_j}\right)$. The minimum of the above equation is obtained at $\lambda_j = 0$. It is straightforward to show that the regularization term above is bounded below by $|\lambda_j|$ and above by $|\lambda_j| + 2$. Therefore, summing over all possible indices j, the regularization term on $\boldsymbol{\lambda}$ lies between $\|\boldsymbol{\lambda}\|_1$ and $\|\boldsymbol{\lambda}\|_1 + 2n$. Thus, this form of regularization can be viewed as a smooth approximation to the ℓ_1 norm. A similar regularization can be imposed using the exp-loss, namely, $e^{\lambda_j} + e^{-\lambda_j}$. For both losses, the j'th regularization term equals $L(\lambda_j\,; 0)$. An equivalent way to impose this form of regularization is to introduce a set of pseudo examples $S_{\text{reg}} = \{\mathbf{x}_k, 0\}_{k=1}^n$ where $\mathbf{x}_k = \mathbf{1}_k$ (a vector with 1 at its k'th position and zeros elsewhere).

Let $\nu > 0$ denote a regularization parameter that governs the relative importance of the regularization term with respect to the empirical loss. The sample loss with regularization becomes, $\text{Loss}(\boldsymbol{\lambda}, \nu, S) = \text{Loss}(\boldsymbol{\lambda}, S) + \nu\,\text{Loss}(\boldsymbol{\lambda}, S_{\text{reg}})$. The batch algorithms we describe in the sequel easily accommodate a weighted sample. Therefore, by introducing a set of n pseudo-examples weighted by ν, we can incorporate regularization into our batch algorithms without any modification to the core of the algorithms. It is simple to verify that the above regular-

INPUT: Training set $S = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}\}_{i=1}^m$; Insensitivity $\varepsilon \in \mathbb{R}_+$
Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t. $\forall \mathbf{a} \in \mathcal{A}\ \ \max_i \left(\sum_{j=1}^n a_j |x_{i,j}|\right) \leq 1$

INITIALIZE: $\boldsymbol{\lambda}_1 = (0, 0, \ldots, 0)$

ITERATE: **For** $t = 1, 2, \ldots$

$$\delta_{t,i} = \boldsymbol{\lambda}_t \cdot \mathbf{x}_i - y_i \qquad q_{t,i}^- = \frac{1}{1 + e^{-\delta_{t,i}+\varepsilon}} \qquad q_{t,i}^+ = \frac{1}{1 + e^{\delta_{t,i}+\varepsilon}} \qquad (1 \leq i \leq m)$$

$$W_{t,j}^- = \sum_{i:x_{i,j}\geq 0} q_{t,i}^-\ x_{i,j} - \sum_{i:x_{i,j}<0} q_{t,i}^+\ x_{i,j} \qquad (1 \leq j \leq n)$$

$$W_{t,j}^+ = \sum_{i:x_{i,j}\geq 0} q_{t,i}^+\ x_{i,j} - \sum_{i:x_{i,j}<0} q_{t,i}^-\ x_{i,j} \qquad (1 \leq j \leq n)$$

$$\mathbf{a}_t = \operatorname*{argmax}_{\mathbf{a}\in\mathcal{A}} \sum_{j=1}^n a_j \left(\sqrt{W_{t,j}^-} - \sqrt{W_{t,j}^+}\right)^2$$

$$\Lambda_{t,j} = \frac{a_{t,j}}{2} \log\left(\frac{W_{t,j}^+}{W_{t,j}^-}\right) \qquad (1 \leq j \leq n)$$

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t$$

Fig. 2. A log-additive update algorithm for minimizing the log-loss.

ization forces the optimal solution $\boldsymbol{\lambda}^\star$ to be unique and with finite elements. We use this property in the convergence analysis of the batch algorithms.

3 Log-additive Update for Batch Regression

In the previous section we discussed a general reduction from regression problems to margin-based classification problems. As a first application of this reduction, we devise a family of batch regression learning algorithms based on boosting techniques. We term these algorithms *log-additive update* algorithms as they iteratively add to $\boldsymbol{\lambda}$ a logarithmic function of the gradient of the loss.

Our implicit goal is to obtain the (global) minimizer of the empirical loss function $\sum_{i=1}^m L(\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i)$ where L is either the log-loss, the exp-loss or the comb-loss. For the sake of clarity, we present algorithms and proofs only for the log-loss. We then complete our presentation with a brief discussion on how everything carries over to settings that employ the exp-loss or the comb-loss.

Following the general paradigm of boosting, we make the assumption that we have access to a set of predefined base regressors. These base regressors are analogous to the weak hypotheses commonly discussed in boosting. The goal of the learning algorithm is to select a subset of base regressors and combine them linearly to obtain a highly accurate strong regressor. We assume that the set of base regressors is of finite cardinality though our algorithms can be generalized to a countably infinite number of base regressors. In the finite case we can simply

map each input instance to the vector of images with respect to each of the base-regressors, $\mathbf{x} \mapsto (h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$ where n is the number or base-regressors. Using this transformation, each input instance is a vector $\mathbf{x}_i \in \mathbb{R}^n$ and the strong regressor's prediction is $\boldsymbol{\lambda} \cdot \mathbf{x}$.

Boosting was initially described and analyzed as a *sequential* algorithm that iteratively selects a base-hypothesis or a feature and changes its weight. All of the elements of $\boldsymbol{\lambda}$ are initialized to be zero, so after performing T sequential update iterations, at most T elements of $\boldsymbol{\lambda}$ are non-zero. Thus, the sequential update can be used for feature selection as well as loss optimization. An alternative approach is to simultaneously update all of the elements of $\boldsymbol{\lambda}$ on every iteration. This approach is the more common among regression algorithms. Collins et al. [2] described a unified framework of boosting algorithms for *classification*. In that framework, the sequential and parallel update schemes are actually two extremes of a general approach for applying iterative updates to $\boldsymbol{\lambda}$. Following Collins et al. we describe and analyze an algorithm that employs *update templates* to determine specifically which subsets of the coordinates of $\boldsymbol{\lambda}$ may be updated in parallel. This algorithm includes both sequential update and parallel update paradigms as special cases by setting the templates accordingly, and allows us to discuss and prove correctness of both algorithms in a unified manner.

In this unified approach, we are required to pre-specify to the algorithm which subsets of the coordinates of $\boldsymbol{\lambda}$ may be updated simultaneously. Formally, the algorithm is given a set of update templates $\mathcal{A}$, where every template $\mathbf{a} \in \mathcal{A}$ is a vector in $\mathbb{R}^n_+$. On every iteration, the algorithm selects a template $\mathbf{a} \in \mathcal{A}$ and updates only those elements λ_j for which a_j is non-zero. We require that every $\mathbf{a} \in \mathcal{A}$ conform with the constraint $\sum_j a_j |x_{i,j}| \leq 1$ for every instance $\mathbf{x}_i$ in the training set. The purpose of this requirement will become apparent in the proof of Thm. 1. The parallel update is obtained by setting $\mathcal{A}$ to contain the single vector $(\rho, \ldots, \rho)$ where $\rho = (\max_i \|\mathbf{x}_i\|_1)^{-1}$. The sequential update is obtained by setting $\mathcal{A}$ to be the set of vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ defined by

$$a_{k,j} = \begin{cases} (\max_i |x_{i,j}|)^{-1} & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases} .$$

The algorithm that we discuss is outlined in Fig. 2 and operates as follows. During the process of building $\boldsymbol{\lambda}$, we may encounter two different types of discrepancies: underestimation and overestimation. If the predicted target $\boldsymbol{\lambda} \cdot \mathbf{x}_i$ is less than the correct target y_i, we say that $\boldsymbol{\lambda}$ underestimates y_i, if it is greater we say that $\boldsymbol{\lambda}$ overestimates y_i. For every instance-target pair in the training set, we use a pair of weights $q^-_{t,i}$ and $q^+_{t,i}$ to represent its discrepancies: $q^-_{t,i}$ represents the degree to which y_i is overestimated by $\boldsymbol{\lambda}_t$ and analogously $q^+_{t,i}$ represents the degree to which y_i is underestimated by $\boldsymbol{\lambda}_t$. We then proceed to calculate two weighted sums over each coordinate of the instances: $W^-_{t,j}$ can be thought of as the degree to which $\boldsymbol{\lambda}_{t,j}$ should be decreased in order to compensate for overestimation discrepancies. Symmetrically, $W^+_{t,j}$ represents the degree to which $\boldsymbol{\lambda}_{t,j}$ should be increased. At this point, the algorithm selects the update template $\mathbf{a}_t \in \mathcal{A}$ with respect to which it will apply the update to $\boldsymbol{\lambda}$. $\mathbf{a}_t$ is selected so as

to maximize the decrease in loss, according to a criterion that follows directly from Thm. 1.

Finally, the update applied to each coordinate of $\boldsymbol{\lambda}_t$ is half the log ratio between the respective elements of W_t^+ and W_t^-, times the scaling factor $a_{t,j}$.

In the following theorem we prove a lower bound on the decrease in loss on every iteration of the algorithm. We later use this bound to show that the algorithm converges to the unique globally optimal regressor $\boldsymbol{\lambda}^\star$.

Theorem 1. *Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of instance-target pairs where for all i in $1, \ldots, m$, $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$. Then using the notation defined in the algorithm outlined in Fig. 2, on every iteration t the decrease in the log-loss satisfies,*

$$\mathrm{Loss}(\boldsymbol{\lambda}_t, S) \;-\; \mathrm{Loss}(\boldsymbol{\lambda}_{t+1}, S) \;\geq\; \sum_{j=1}^{n} a_{t,j} \left(\sqrt{W_{t,j}^-} - \sqrt{W_{t,j}^+} \right)^2 \quad .$$

Proof Define $\Delta_t(i)$ to be the difference between the loss attained by $\boldsymbol{\lambda}_t$ and that attained by $\boldsymbol{\lambda}_{t+1}$ on an instance-target pair $(\mathbf{x}_i, y_i)$ in the training set, namely $\Delta_t(i) \;=\; L_{\log}(\delta_{t,i}) - L_{\log}(\delta_{t+1,i})$. Since $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t$ then $\delta_{t+1,i} = \delta_{t,i} + \boldsymbol{\Lambda}_t \cdot \mathbf{x}_i$. Using this equality, and the identity $1/(1+e^{\alpha}) = 1 - 1/(1+e^{-\alpha})$, $\Delta_t(i)$ can be rewritten as follows,

$$\begin{aligned}
\Delta_t(i) &= -\log\left(\frac{1+e^{\delta_{t+1,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}}\right) - \log\left(\frac{1+e^{-\delta_{t+1,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}}\right) \\
&= -\log\left(1 - \frac{1}{1+e^{-(\delta_{t,i}-\varepsilon)}} + \frac{e^{\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}}{1+e^{-(\delta_{t,i}-\varepsilon)}}\right) \\
&\quad -\log\left(1 - \frac{1}{1+e^{-(-\delta_{t,i}-\varepsilon)}} + \frac{e^{-\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}}{1+e^{-(-\delta_{t,i}-\varepsilon)}}\right) \quad .
\end{aligned}$$

We can now plug the definitions of $q_{t,i}^+$ and $q_{t,i}^-$ into this expression to get

$$\Delta_t(i) \;=\; -\log\left(1 - q_{t,i}^-\left(1 - e^{\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right)\right) - \log\left(1 - q_{t,i}^+\left(1 - e^{-\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right)\right) \quad .$$

Next we apply the inequality $-\log(1-\alpha) \geq \alpha$ (which holds wherever $\log(1-\alpha)$ is defined):

$$\Delta_t(i) \;\geq\; q_{t,i}^-\left(1 - e^{\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right) \;+\; q_{t,i}^+\left(1 - e^{-\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right) \quad . \tag{3}$$

We rewrite the scalar product $\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i$ in a more convenient form,

$$\begin{aligned}
\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i &= \sum_{j=1}^{n} \frac{a_{t,j}}{2} \log\left(W_{t,j}^+ / W_{t,j}^-\right) x_{i,j} \\
&= \sum_{j=1}^{n} \left(a_{t,j}|x_{i,j}|\right)\; \mathrm{sign}(x_{i,j})\; \log\left(\sqrt{W_{t,j}^+ / W_{t,j}^-}\right) \quad .
\end{aligned} \tag{4}$$

We recall the assumptions made on the vectors in $\mathcal{A}$, namely that $\mathbf{a}_t$ and $\mathbf{x}_i$ comply with $\sum_{j=1}^n a_{t,j}|x_{i,j}| \leq 1$ and that $a_{t,j}|x_{i,j}|$ is non-negative. We now use

the fact that $(1 - e^{\alpha})$ is a concave function and is equal to zero at $\alpha = 0$. Replacing $\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i$ in Eq. (3) with the form given by Eq. (4) we get,

$$\begin{aligned}
\Delta_t(i) &\geq q^-_{t,i}\left(1 - e^{\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right) + q^+_{t,i}\left(1 - e^{-\boldsymbol{\Lambda}_t \cdot \mathbf{x}_i}\right) \\
&\geq \sum_{j=1}^{n} a_{t,j} q^-_{t,i} |x_{i,j}| \left(1 - e^{\mathrm{sign}(x_{i,j}) \log\left(\sqrt{W^+_{t,j}/W^-_{t,j}}\right)}\right) \\
&+ \sum_{j=1}^{n} a_{t,j} q^+_{t,i} |x_{i,j}| \left(1 - e^{-\mathrm{sign}(x_{i,j}) \log\left(\sqrt{W^+_{t,j}/W^-_{t,j}}\right)}\right) \quad .
\end{aligned}$$

We now rewrite,

$$\begin{aligned}
\Delta_t(i) \geq & \sum_{j:x_{i,j}>0} a_{t,j} q^-_{t,i} |x_{i,j}| \left(1 - \sqrt{\frac{W^+_{t,j}}{W^-_{t,j}}}\right) + \sum_{j:x_{i,j}<0} a_{t,j} q^-_{t,i} |x_{i,j}| \left(1 - \sqrt{\frac{W^-_{t,j}}{W^+_{t,j}}}\right) \\
+ & \sum_{j:x_{i,j}>0} a_{t,j} q^+_{t,i} |x_{i,j}| \left(1 - \sqrt{\frac{W^-_{t,j}}{W^+_{t,j}}}\right) + \sum_{j:x_{i,j}<0} a_{t,j} q^+_{t,i} |x_{i,j}| \left(1 - \sqrt{\frac{W^+_{t,j}}{W^-_{t,j}}}\right) .
\end{aligned}$$

Summing $\Delta_t(i)$ over i and using the definition of the q's and W's we finally get that,

$$\begin{aligned}
\sum_{i=1}^{m} \Delta_t(i) &\geq \sum_{j=1}^{n} a_{t,j} \left(W^-_{t,j}\left(1 - \sqrt{W^+_{t,j}/W^-_{t,j}}\right) + W^+_{t,j}\left(1 - \sqrt{W^-_{t,j}/W^+_{t,j}}\right)\right) \\
&= \sum_{j=1}^{n} a_{t,j} \left(\sqrt{W^-_{t,j}} - \sqrt{W^+_{t,j}}\right)^2 \quad .
\end{aligned}$$

This concludes the proof. □

For the remainder of this section, we assume that the set of update templates $\mathcal{A}$ is not degenerate, in the sense that every coordinate of $\boldsymbol{\lambda}$ is accessible. We now show that the incorporation of a regularization term (Sec. 2) into the loss function implies that the algorithm converges to the unique global minimizer of the loss. First, it is easily verified that the regularization term guarantees that the loss function is strictly convex and attains its unique minimum at the point denoted $\boldsymbol{\lambda}^\star$. Second, the regularization term guarantees that all admissible values for $\boldsymbol{\lambda}_t$ lie within a compact set C. To see this, note that the initial loss with regularization is

$$\mathrm{Loss}(\mathbf{0}, \nu, S) = \mathrm{Loss}(\mathbf{0}, S) \; + \; \nu \, \mathrm{Loss}(\mathbf{0}, S_{\mathrm{reg}}) \quad .$$

Denote the initial loss above by $\mathcal{L}_0$. Since the loss attained by the algorithm on every iteration is non-increasing, the contribution of the regularization term to the total loss cannot exceed $\mathcal{L}_0/\nu$. Also, the regularization term for both the exp-loss and the log-loss bounds the ℓ_∞ norm of $\boldsymbol{\lambda}_t$ by

$$\|\boldsymbol{\lambda}_t\|_\infty \; \leq \; \mathrm{Loss}(\boldsymbol{\lambda}_t, S_{\mathrm{reg}}) \; \leq \; \mathrm{Loss}(\boldsymbol{\lambda}_t, \nu, S)/\nu \; \leq \; \mathcal{L}_0/\nu \quad .$$

Therefore, the compact set C of admissible values for $\boldsymbol{\lambda}_t$ is $\{\boldsymbol{\lambda} : \|\boldsymbol{\lambda}\|_\infty \leq \mathcal{L}_0/\nu\}$. The lower bound on the decrease in loss given in Thm. 1 can be thought of as a function of the current regressor $\boldsymbol{\lambda}_t$ and is equal to zero only when the gradient of the loss function equals zero, that is, at $\boldsymbol{\lambda}^\star$.

Assume by contradiction that the sequence of regressors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ does *not* converge to $\boldsymbol{\lambda}^\star$. An immediate consequence of this assumption is that there exists $\gamma > 0$ such that an infinite subsequence of regressors $\boldsymbol{\lambda}_{s_1}, \boldsymbol{\lambda}_{s_2}, \ldots$ remains outside of $B(\boldsymbol{\lambda}^\star, \gamma)$, the open ball of radius γ centered at $\boldsymbol{\lambda}^\star$. The set $C \setminus B(\boldsymbol{\lambda}^\star, \gamma)$ is also compact. Therefore, the lower bound from Thm. 1 attains a minimum value over $C \setminus B(\boldsymbol{\lambda}^\star, \gamma)$ at $\tilde{\boldsymbol{\lambda}} \neq \boldsymbol{\lambda}^\star$. Denoting this minimum by μ, we conclude that μ is a positive lower bound on the decrease in loss on each of the iterations $s_1, s_2, \ldots$. If the loss decreases by at least μ an unbounded number of times then it must eventually become negative. We therefore get a contradiction since the loss is a non-negative function. We thus conclude that the sequence $\boldsymbol{\lambda}_t$ converges to $\boldsymbol{\lambda}^\star$.

So far, we have focused on the log-loss function. The algorithm described in Fig. 2 can easily be adapted to minimize the exp-loss or the comb-loss by simply redefining the overestimation and underestimation weights q^- and q^+. For exp-loss regression problems, we define $q^-_{t,i} = e^{\delta_{t,i}}$ and $q^+_{t,i} = e^{-\delta_{t,i}}$.

Similarly, we can redefine q^- and q^+ to minimize the comb-loss. Recall that the comb-loss function is defined by a pair of insensitivity parameters, ε_1 and ε_2. To minimize the comb-loss, we define

$$q^-_{t,i} \;=\; \frac{e^{\delta_{t,i}-\varepsilon_1}}{1+e^{\delta_{t,i}-\varepsilon_1}} \;+\; e^{\delta_{t,i}-\varepsilon_2} \qquad q^+_{t,i} \;=\; \frac{e^{-\delta_{t,i}-\varepsilon_1}}{1+e^{-\delta_{t,i}-\varepsilon_1}} \;+\; e^{-\delta_{t,i}-\varepsilon_2} \quad .$$

All of the formal discussion given in this section carries over to the exp-loss and the comb-loss cases with only minor technical adaptations necessary.

4 Additive Update for Batch Regression

In this section we describe a family of additive batch learning algorithms that advance on each iteration in a direction which is a linear transformation of the gradient of the loss. We term these algorithm *additive update* algorithms. These algorithms bear a resemblance to the log-additive algorithms described in the previous section, as do their proofs of progress. As in the previous section, we first restrict the discussion to the log-loss and then outline the adaptation to the exp-loss at the end of the section.

We again devise a template-based family of updates. This family includes a parallel update which modifies all the elements of $\boldsymbol{\lambda}$ simultaneously and a sequential update which updates a single element of $\boldsymbol{\lambda}$ on each iteration. We denote the set of update templates by $\mathcal{A}$ and assume that every $\mathbf{a} \in \mathcal{A}$ is a vector in $\mathbb{R}^n_+$. For each $\mathbf{a} \in \mathcal{A}$ we require that $\sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \leq 2$.

The pseudo-code of the additive update algorithm is given in Fig. 3. Intuitively, on each iteration t, the algorithm computes the negative of the gradient with respect to $\boldsymbol{\lambda}_t$, denoted $(W_{t,1}, \ldots, W_{t,n})$. It then selects the update template $\mathbf{a}_t \in \mathcal{A}$ which, as we shortly show in Thm. 2, guarantees a maximal drop in the loss.

INPUT: Training set $S = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}\}_{i=1}^m$; Insensitivity $\varepsilon \in \mathbb{R}_+$
Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t. $\forall \mathbf{a} \in \mathcal{A}\ \ \sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \leq 2$

INITIALIZE: $\boldsymbol{\lambda}_1 = (0, 0, \ldots, 0)$

ITERATE: **For** $t = 1, 2, \ldots$

$$\delta_{t,i} = \boldsymbol{\lambda}_t \cdot \mathbf{x}_i - y_i \qquad q_{t,i}^- = \frac{1}{1 + e^{-\delta_{t,i}+\varepsilon}} \qquad q_{t,i}^+ = \frac{1}{1 + e^{\delta_{t,i}+\varepsilon}} \qquad (1 \leq i \leq m)$$

$$W_{t,j} \ = \ \sum_{i=1}^n \left(q_{t,i}^+ - q_{t,i}^-\right) x_{i,j} \qquad (1 \leq j \leq n)$$

$$\mathbf{a}_t = \operatorname*{argmax}_{\mathbf{a} \in \mathcal{A}} \sum_{j=1}^n a_j W_{t,j}^2$$

$$\Lambda_{t,j} \ = \ a_{t,j} W_{t,j} \qquad (1 \leq j \leq n)$$

$$\boldsymbol{\lambda}_{t+1} \ = \ \boldsymbol{\lambda}_t \ + \ \boldsymbol{\Lambda}_t$$

Fig. 3. An additive update algorithm for minimizing the log-loss.

Theorem 2. *Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of instance-target pairs where for all i in $1, \ldots, m$, $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$. Then using the notation defined in the algorithm outlined in Fig. 3, on every iteration t the decrease in the log-loss, denoted Δ_t, satisfies*

$$\Delta_t \ = \ \mathrm{Loss}(\boldsymbol{\lambda}_t, S) \ - \ \mathrm{Loss}(\boldsymbol{\lambda}_{t+1}, S) \ \geq \ \frac{1}{2} \sum_{j=1}^n a_{t,j} W_{t,j}^2 \ .$$

Proof To prove the theorem we construct a parametric quadratic function $Q : \mathbb{R} \to \mathbb{R}$ which bounds the log-loss along the direction $\boldsymbol{\Lambda}$ from $\boldsymbol{\lambda}$. Concretely, the function Q is defined as

$$Q_{\boldsymbol{\lambda},\boldsymbol{\Lambda}}(\alpha) \ = \ \mathrm{Loss}(\boldsymbol{\lambda}, S) \ + \ \left(\nabla \mathrm{Loss}(\boldsymbol{\lambda}, S) \cdot \boldsymbol{\Lambda}\right)\left(\alpha - \alpha^2/2\right) \ . \tag{5}$$

Next, we show that for all α, $Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(\alpha) \geq \mathrm{Loss}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda}_t, S)$ where $\boldsymbol{\Lambda}_t$ is defined as in Fig. 3. For convenience, we define $\Gamma(\alpha) = Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}(\alpha) - \mathrm{Loss}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda}_t, S)$ and prove that Γ is a non-negative function.

By construction, we get that $\Gamma(0) = 0$. Since the derivative of $Q_{\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}_t}$ at zero is equal to $\nabla \mathrm{Loss}(\boldsymbol{\lambda}_t, S) \cdot \boldsymbol{\Lambda}_t$, we get that the derivative of Γ at zero is also zero. To prove that Γ is non-negative it remains to show that Γ is convex and thus $\alpha = 0$ attains its global minimum. To prove convexity it is sufficient to show that the second derivative of Γ (denoted Γ'') is non-negative. Routine calculations yield that,

$$\Gamma''(\alpha) = -\boldsymbol{\Lambda} \cdot \nabla \mathrm{Loss}(\boldsymbol{\lambda}, S) - \boldsymbol{\Lambda}^{\mathrm{T}} H \boldsymbol{\Lambda} \ , \tag{6}$$

where $H = \sum_{i=1}^m L_{\log}''(\boldsymbol{\lambda} + \alpha\boldsymbol{\Lambda})\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}$ and $L_{\log}''$ is the second derivative of the log-loss. It is simple to show that this derivative is in $[0, 1/2]$. Plugging the value of

H into Eq. (6) we get that,

$$\Gamma''(\alpha) \geq -\boldsymbol{\Lambda} \cdot \nabla \mathrm{Loss}(\boldsymbol{\lambda}, S) - \frac{1}{2} \sum_{i=1}^{m} (\boldsymbol{\Lambda} \cdot \mathbf{x}_i)^2 \quad . \tag{7}$$

Note that on the t'th iteration, the j'th element of $\boldsymbol{\Lambda}_t$ equals $a_{t,j} W_{t,j}$ where $W_{t,j} = -\nabla_j \mathrm{Loss}(\boldsymbol{\lambda}_t, S)$. Therefore, we rewrite Eq. (7) as,

$$\begin{aligned} \Gamma''(\alpha) &\geq \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 - \frac{1}{2} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{t,j} W_{t,j} x_{i,j} \right)^2 \\ &= \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 - \frac{1}{2} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \sqrt{a_{t,j}}\, W_{t,j}\, \sqrt{a_{t,j}}\, x_{i,j} \right)^2 \quad . \end{aligned} \tag{8}$$

Using Cauchy-Schwartz inequality ($\mathbf{u} \cdot \mathbf{v} \leq \|\mathbf{u}\| \|\mathbf{v}\|$) we further bound Γ'' as,

$$\begin{aligned} \Gamma''(\alpha) &\geq \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 - \frac{1}{2} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{t,j} W_{t,j}^2 \right) \left(\sum_{k=1}^{n} a_{t,k} x_{i,k}^2 \right) \\ &= \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 \left(1 - \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{n} a_{t,k} x_{i,k}^2 \right) \quad . \end{aligned} \tag{9}$$

Finally, we use the constraint $\sum_i \sum_{k=1}^{n} a_{t,k} x_{i,k}^2 \leq 2$ which immediately implies that $\Gamma''(\alpha) \geq 0$.

Summing up, we have shown that $\mathrm{Loss}(\boldsymbol{\lambda}_t + \alpha \boldsymbol{\Lambda}_t, S)$ is upper bounded by $Q_{\boldsymbol{\lambda}_t, \boldsymbol{\Lambda}_t}(\alpha)$. Therefore, $\mathrm{Loss}(\boldsymbol{\lambda}_{t+1}, S) = \mathrm{Loss}(\boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t, S) \leq Q_{\boldsymbol{\lambda}_t, \boldsymbol{\Lambda}_t}(1)$, hence,

$$\Delta_t \geq \mathrm{Loss}(\boldsymbol{\lambda}_t, S) - Q_{\boldsymbol{\lambda}_t, \boldsymbol{\Lambda}_t}(1) = \frac{1}{2} \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 \quad .$$

This concludes the proof. □

The proof of convergence for the additive update algorithm follows identical lines as the proof of convergence for the log-additive update algorithm. If the loss function includes a regularization term then the lower bound discussed in Thm. 2 attains a value of zero only at $\boldsymbol{\lambda}^\star$, the unique global optimum of $\mathrm{Loss}(\boldsymbol{\lambda}, \nu, S)$. This fact implies convergence of $\boldsymbol{\lambda}$ to $\boldsymbol{\lambda}^\star$.

To conclude this section, we briefly outline the adaptation of the additive update algorithm to the exp-loss. Since the gradient of the exp-loss is itself exponential, we cannot hope to minimize the exp-loss by straightforward gradient descent. However, we can apply a gradient descent approach to the *logarithm* of the exp-loss on the entire sample, as both the empirical exp-loss and its logarithm share the same global optimum. For this modified loss, we can also apply the proof technique of Thm. 2.

Online EG

INPUT: upper bound X
insensitivity parameter ε
INITIALIZE: $\boldsymbol{\lambda}_1 = (\frac{1}{n}, \ldots, \frac{1}{n})$

For $t = 1, 2, \ldots$
Receive an example $\mathbf{x}_t$
Predict $\boldsymbol{\lambda}_t \cdot \mathbf{x}_t$
Receive target y_t and update:
$\delta_t = \boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t$
$L'_{\log}(\delta_t) = \frac{1}{1+e^{-\delta_t+\varepsilon}} - \frac{1}{1+e^{\delta_t+\varepsilon}}$
$\beta_t = L'_{\log}(\delta_t)/X^2$
For $1 \leq j \leq n$:
$\lambda_{t+1,j} = \frac{\lambda_{t,j} e^{-\beta_t x_{t,j}}}{\sum_{k=1}^{n} \lambda_{t,k} e^{-\beta_t x_{t,k}}}$

Online GD

INPUT: upper bound R
insensitivity parameter ε
INITIALIZE: $\boldsymbol{\lambda}_1 = (0, 0, \ldots, 0)$

For $t = 1, 2, \ldots$
Receive an example $\mathbf{x}_t$
Predict $\boldsymbol{\lambda}_t \cdot \mathbf{x}_t$
Receive target y_t and update:
$\delta_t = \boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t$
$L'_{\log}(\delta_t) = \frac{1}{1+e^{-\delta_t+\varepsilon}} - \frac{1}{1+e^{\delta_t+\varepsilon}}$
$\beta_t = L'_{\log}(\delta_t)/(2R^2)$
$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \beta_t \mathbf{x}_t$

Fig. 4. The EG and GD algorithms for online regression with the log-loss.

5 Online Regression Algorithms

In this section we describe online regression algorithms for the log-loss. We follow the notation and techniques used in [9, 8, 1]. In online learning settings, we observe a sequence of instance-target pairs, in rounds, one by one. On round t we first receive an instance $\mathbf{x}_t$. Based on the current regressor, $\boldsymbol{\lambda}_t$, we extend a prediction $\boldsymbol{\lambda}_t \cdot \mathbf{x}_t$. We then receive the true target y_t and suffer an instantaneous loss which is in our case, $L_{\log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t)$. Our goal is to suffer a small cumulative loss. The learning algorithm employs an *update rule* which modifies its current regressor after each round. We describe and analyze two online regression algorithms that differ in the update rules that they employ. The first is additive in the gradient of the loss and is thus called *Gradient Descent* (GD) while the second is exponential in the gradient of the loss and is analogously called *Exponentiated Gradient* (EG).

The GD algorithm: The pseudo-code of the algorithm is given on the right hand side of Fig. 4. Note that the GD algorithm updates its current regressor, $\boldsymbol{\lambda}_t$, by subtracting the gradient of the loss function from it. The GD algorithm assumes an upper bound R on the norm of the instances, that is, $\|\mathbf{x}_t\|_2 \leq R$. In the following analysis we give a bound on the cumulative loss for any number of rounds. However, rather than bounding the loss per se we bound the cumulative loss *relative* to the cumulative loss suffered by a *fixed* regressor $\boldsymbol{\mu}$. The bound holds for any linear regressor $\boldsymbol{\mu}$ and any number of rounds, hence we get that the GD algorithm is competitive with the optimal (fixed) linear regressor for any number of rounds. Formally, the following theorem states that the cumulative

loss attained by the GD algorithm is at most twice the cumulative loss of any fixed linear regressor plus an additive constant.

Theorem 3. *Let $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ be a sequence of instance-target pairs such that $\forall t : \|x_t\|_2 \leq R$ and let $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T$ be the regressors generated by the GD online algorithm (Fig. 4) on the sequence. Then for any fixed linear regressor $\boldsymbol{\mu} \in \mathbb{R}^n$ we have*

$$\sum_{t=1}^{T} L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) \;\leq\; 2 \sum_{t=1}^{T} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;+\; 2R^2 \|\boldsymbol{\mu}\|_2^2 \quad . \tag{10}$$

The proof of the theorem is based on the following lemma that underscores an invariant property of the update rule.

Lemma 1. *Consider the setting of Thm. 3, then for each round t we have*

$$L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - 2L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \leq 2R^2 \left(\|\boldsymbol{\lambda}_t - \boldsymbol{\mu}\|_2^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\mu}\|_2^2 \right) \quad . \tag{11}$$

The proof of the lemma is omitted due to the lack of space. Intuitively, the lemma states that if the loss of GD on round t is greater than the loss of a fixed regressor $\boldsymbol{\mu}$, then the algorithm updates its regressor so that $\boldsymbol{\lambda}_{t+1}$ gets closer to $\boldsymbol{\mu}$ than $\boldsymbol{\lambda}_t$. In contrast, if the loss of $\boldsymbol{\mu}$ is greater than the loss of GD, the algorithm may move its regressor away from $\boldsymbol{\mu}$. With Lemma 1 handy it is almost immediate to prove Thm. 3.

Proof of Theorem 3: Summing Eq. (11) for $t = 1, ..., T$ we get

$$\begin{aligned} \sum_{t=1}^{T} L_{\log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - 2 \sum_{t=1}^{T} L_{\log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) &\leq 2R^2 \left(\|\boldsymbol{\lambda}_1 - \boldsymbol{\mu}\|_2^2 - \|\boldsymbol{\lambda}_{T+1} - \boldsymbol{\mu}\|_2^2 \right) \\ &\leq 2R^2 \|\boldsymbol{\lambda}_1 - \boldsymbol{\mu}\|_2^2 \\ &= 2R^2 \|\boldsymbol{\mu}\|_2^2 \quad , \end{aligned}$$

where in the last equality we use the fact that the initial regressor, $\boldsymbol{\lambda}_1$, is the zero vector. □

The EG algorithm: The algorithm is described on the left hand side of Fig. 4. The algorithm assumes that the regressor is in the probability simplex, $\boldsymbol{\lambda} \in \mathbb{P}^n$ where $\mathbb{P}^n = \{\boldsymbol{\mu} \;:\; \boldsymbol{\mu} \in \mathbb{R}_+^n, \; \sum_{j=1}^n \mu_j = 1\}$. We would like to note in passing that following an analogous construction to the one employed in [9, 8], it is possible to derive a generalized version of EG in which the elements of $\boldsymbol{\lambda}$ can be either negative or positive, so long as the sum of their absolute values is less than 1. The EG algorithm assumes an upper bound, denoted X, on the difference between the maximal value and minimal value of any two elements in all of the instances it receives, $X \geq (\max_j x_{t,j} - \min_j x_{t,j})$. Since EG maintains a

regressor from the probability simplex, we measure the cumulative loss of the EG algorithm *relative* to the cumulative loss achieved by any fixed regressor from the probability simplex. The following theorem gives a bound on the loss of the EG algorithm relative to the loss of any fixed regressor from $\mathbb{P}^n$.

Theorem 4. *Let* $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ *be a sequence of instance-target pairs, and let* $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T$ *be the regressors generated by the EG online algorithm (Fig. 4) on the sequence. Then, for any fixed regressor* $\boldsymbol{\mu} \in \mathbb{P}^n$ *we have*

$$\sum_{t=1}^{T} L_{\log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) \;\leq\; \frac{4}{3} \sum_{t=1}^{T} L_{\log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;+\; \frac{4}{3} X^2 D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_1) \quad , \qquad (12)$$

where $D_{RE}(p, q) = \sum_j p_j \log(p_j / q_j)$ *is the relative entropy.*

The proof of the theorem is analogous to the proof of Thm. 3 and employs a relative entropy based progress lemma.

6 Discussion

We described a framework for solving regression problems by a symmetrization of margin loss functions. Our approach naturally lent itself to a shifted and symmetric loss function which is approximately zero in a pre-specified interval and can thus be used as a smooth alternative to the ε-insensitive hinge loss. We presented both batch and online algorithms for solving the resulting regression problems. The updates of the batch algorithms we presented have a log-additive and an additive form. Our framework also results in a new and very simple to implement regularization scheme. As a byproduct, we tacitly derived a new additive algorithm for boosting-based classification, which can be used in conjunction with the newly introduced regularization scheme. There are numerous extensions of this work. One of them is the application of Thms. 1 and 2 as splitting criteria for learning regression trees. Another interesting direction is the marriage of the loss symmetrization technique with other boosting related techniques such as drifting games [12, 4].

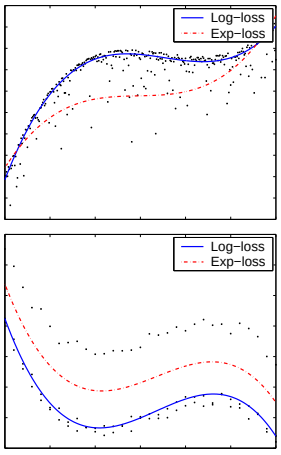

Fig. 5. A comparison of log-loss and exp-loss.

We conclude the paper with a synthetic example that underscores the different merits of the log-loss and the exp-loss. In Fig. 5 we show results obtained for both losses on two synthetic datasets. Each dataset was generated by uniformly sampling from a univariate third degree polynomial. One-sided noise, generated by taking minus the absolute value of a normal random variable, was added to the first dataset (top plot in Fig. 5). Regressors were learned using both the log-loss and the exp-loss, using a degree 3 polynomial kernel. The regressor obtained by minimizing the log-loss is very close to the function used to generate

the data, demonstrating the robustness of the log-loss to noise. The regressor attained by minimizing the exp-loss, however, attempts to minimize the maximal discrepancy over the entire data set and therefore lies significantly below. The other facet of this behavior is illustrated on the bottom plot of Fig. 5. For this dataset, a third of the targets were shifted by a positive constant. The regressor obtained by minimizing the exp-loss lies between the two groups of points and as such approximately minimizes the ℓ_∞ regression loss on the sample. The regressor found by minimizing the log-loss practically ignores the shifted third of the sample. The log-loss shares the same asymptotic behavior as the absolute loss and as such its solution resembles the median. The different merits of the two losses can be exploited in more complex decision tasks such as ranking problems. We leave this and the extensions mentioned above for future research.

Acknowledgements: We are in debt to Rob Schapire for making the connection to regularization and for numerous comments. Part of this research was funded by the Bi-national Science Foundation grant no. 1999-038.

References

1. Nicolò Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, 59(3):392–411, 1999.
2. M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3):253–285, 2002.
3. N. Duffy and D. Helmbold. Leveraging for regression. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*. ACM, 2000.
4. Y. Freund and M. Opper. Drifting games and Brownian motion. *Journal of Computer and System Sciences*, 64:113–132, 2002.
5. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, April 2000.
6. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.
7. P.J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.
8. J. Kivinen, D.P Helmbold, and M. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 1999.
9. J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.
10. G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2001.
11. T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), 1990.
12. Robert E. Schapire. Drifting games. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.
13. A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2, 1998.
14. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.