Journal of Machine Learning Research 6 (2005) 711–741 Submitted 1/04; Revised 2/05; Published 5/05

Smooth ε-Insensitive Regression by Loss Symmetrization

Ofer Dekel OFERD@CS.HUJI.AC.IL
Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem, 91904, Israel

Editors: Kristin P. Bennett and Nicolò Cesa-Bianchi

Abstract

We describe new loss functions for regression problems along with an accompanying algorithmic framework which utilizes these functions. These loss functions are derived by symmetrization of margin-based losses commonly used in boosting algorithms, namely, the logistic loss and the exponential loss. The resulting symmetric logistic loss can be viewed as a smooth approximation to the ε-insensitive hinge loss used in support vector regression. We describe and analyze two parametric families of batch learning algorithms for minimizing these symmetric losses. The first family employs an iterative *log-additive* update which can be viewed as a regression counterpart to recent boosting algorithms. The second family utilizes an iterative *additive* update step. We also describe and analyze online gradient descent (GD) and exponentiated gradient (EG) algorithms for the symmetric logistic loss. A byproduct of our work is a new simple form of regularization for boosting-based classification and regression algorithms. Our regression framework also has implications on classification algorithms, namely, a new additive update boosting algorithm for classification. We demonstrate the merits of our algorithms in a series of experiments.

1. Introduction

The focus of this paper is supervised learning of real-valued functions. We observe a sequence $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of instance-target pairs, where the instances are vectors in $\mathbb{R}^n$ and the targets are real-valued scalars, $y_i \in \mathbb{R}$. Our goal is to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ which provides a good approximation of the target values from their corresponding instance vectors. Such a function is often referred to as a regression function or a regressor for short. Regression problems have long been the focus of research papers in statistics and learning theory (see for instance the book by Hastie, Tibshirani, and Friedman (2001) and the references therein). In this paper we discuss learning of 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 $f(\mathbf{x}) = \sum_{j=1}^{l} \lambda_j h_j(\mathbf{x})$ where each base regressor h_j is a mapping from an instance domain X into $\mathbb{R}$. The latter form enables us to employ kernels by setting $h_j(\mathbf{x}) = K(\mathbf{x}_j, \mathbf{x})$.

The class of linear regressors is rather restricted. Furthermore, in real applications both the instances and the target values are often corrupted by noise and a perfect mapping such that for all $(\mathbf{x}_i, y_i) \in S$, $f(\mathbf{x}_i) = y_i$ is usually unobtainable. Hence, we employ a loss

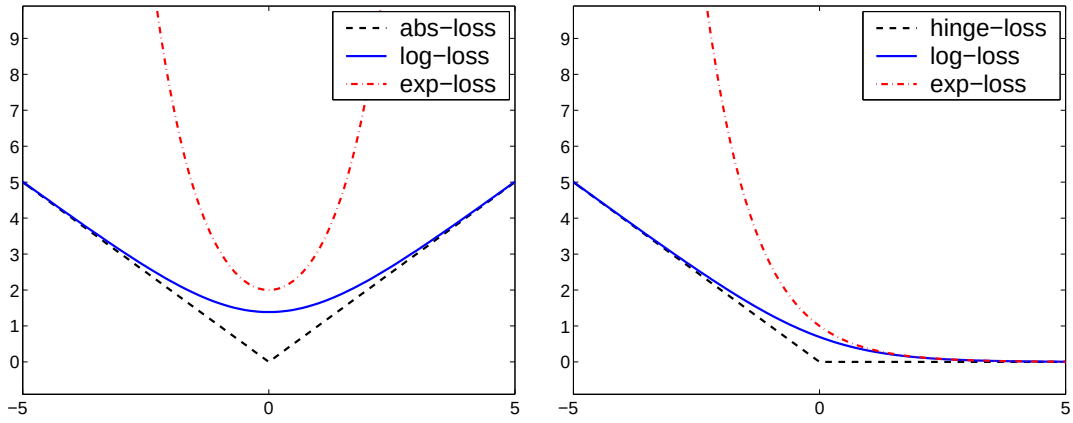

Figure 1: Constructing regression losses (left) by symmetrization of margin losses (right).

function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ which determines the penalty for a 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 between the predicted target and the true target $\delta = f(\mathbf{x}) - y$, hence L can be viewed as a function from $\mathbb{R}$ into $\mathbb{R}_+$. We therefore allow ourselves to overload our notation and denote $L(\delta) = L(f(\mathbf{x}), y)$.

Given a loss function L, the goal of a regression algorithm is to find a regressor f which attains a small total loss on the training set S,

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

Denoting the discrepancy $\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i$ by δ_i, we note that two common approaches to solving regression problems are to minimize either the sum of the absolute discrepancies over the sample ($\sum_i |\delta_i|$) or the sum of squared 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 (Huber, 1981). Furthermore, it is often the case that the *exact* discrepancy between $\boldsymbol{\lambda} \cdot \mathbf{x}$ and y is unimportant so long as it 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 the left hand side of Figure 2). The ε-insensitive hinge loss is not smooth as its derivative is discontinuous at $\delta = \pm\varepsilon$. Several batch learning algorithms have been proposed for minimizing the ε-insensitive hinge loss (see for example Vapnik, 1998; Smola and Schölkopf, 1998). However, these algorithms are based on rather complex constrained optimization techniques since the ε-insensitive hinge loss is a non-smooth function.

The first loss function presented in this paper is a smooth approximation to the ε-insensitive hinge loss. Define the *symmetric ε-insensitive logistic loss*, or *log-loss* for short, to be,

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

Whenever it is clear from context we omit the insensitivity parameter ε and denote this loss by $L_{\log}(\delta)$. The constant κ in Eq. (1) equals $2\log(1 + e^{-\varepsilon})$ and is set such that $L_{\log}(0) = 0$. Since additive constants do not affect the value of the minimizer of $L_{\log}(\delta)$, we omit κ

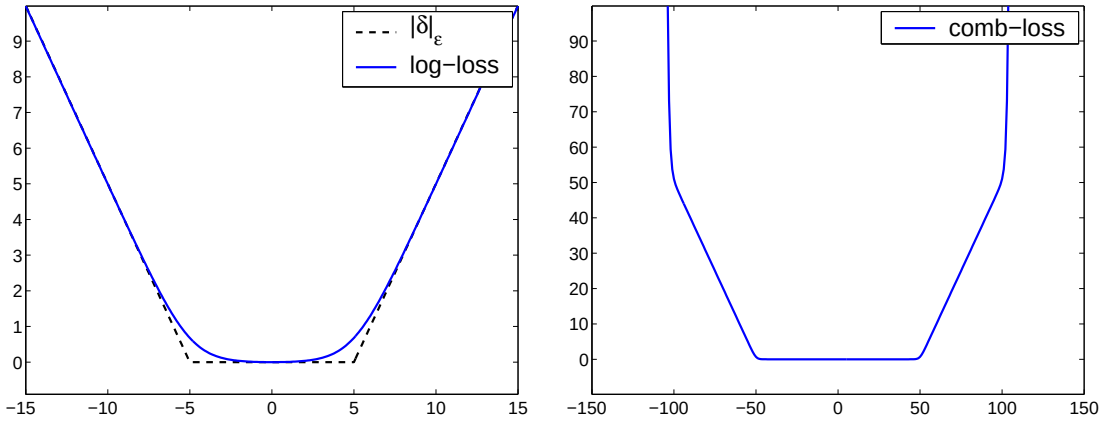

Figure 2: The smooth ε-insensitive log-loss (left) and the comb-loss (right).

henceforth. In Figure 2 we depict the ε-insensitive log-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. Moreover, note that for this particular choice of ε and for $|\delta| < 2$ and $|\delta| > 8$ the log-loss and hinge-loss are graphically indistinguishable.

To motivate our construction, let us take a short detour and discuss a recent view of margin-based classification algorithms. In the binary classification setting discussed in Friedman et al. (2000), Collins et al. (2002) and Lebanon and Lafferty (2001), 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 $|f(\mathbf{x})|$ 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 total loss over all instances in a given sample. One of the margin losses discussed is the logistic loss, which takes the form

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

We discuss a general technique for reducing a regression problem to a margin-based classification problem called *loss symmetrization*. The symmetric log-loss given in Eq. (1) is obtained by applying this technique to the classification logistic loss in Eq. (2). The technique of loss symmetrization was previously discussed in Bi and Bennett (2003) in the context of support vector regression.

Formally, let $[\mathbf{u}\,;\,v]$ denote 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,

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

In words, we duplicate each regression instance and create two instances of a classification problem. We then increase the dimension of the instance vectors 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 copy of the instance 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,

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

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

The work in Collins et al. (2002) 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*, or *exp-loss* for short, as

$$L_{\exp}(\delta) \;=\; e^{\delta} + e^{-\delta}. \tag{3}$$

The exp-loss was first presented and analyzed by Duffy and Helmbold (2000) in their pioneering work on leveraging regressors. However, their view is somewhat different than ours 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 in the training set. 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

$$L_{\mathrm{comb}}(\delta;\varepsilon_1,\varepsilon_2) \;=\; L_{\log}(\delta;\varepsilon_1) + L_{\exp}(\delta;\varepsilon_2),$$

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 on the right hand side of Figure 2.

The paper is organized as follows. In Section 2 we describe and analyze a family of *log-additive update* algorithms for batch learning settings. The algorithms in this family are in essence boosting algorithms for regression problems. The symmetrization technique outlined above is used to derive these algorithms and to adapt proof techniques from Collins et al. (2002). In Section 3 we describe another 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. Then, in Section 4, we describe a simple use of the symmetric losses defined above as a means of regularizing our batch learning algorithms and other boosting algorithms as well. In Section 5 we discuss the convergence properties of both log-additive and additive update algorithms, when applied with regularization. We then show the implications of our work on classification problems in Section 6. Specifically, we show how both the additive update algorithm of Section 3 and the regularization scheme of Section 4 extend to the setting of classification. In Section 7 we shift our attention to *online* learning algorithms for the ε-insensitive log-loss. In Section 8

we complement our formal discussion with a set of experimental results obtained on real and synthetic data sets. Specifically, we demonstrate the different properties of the log-loss and exp-loss functions, we compare the different algorithms presented in this paper under different settings and discuss the effect of regularization on the generalization abilities of our algorithms. In this section, we also present a detailed example of boosting a weak-learning regression algorithm using our techniques. We conclude the paper in Section 9.

2. 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 update $\boldsymbol{\lambda}$ by 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. We first prove that progress is made on every iteration of the learning algorithm. For the sake of clarity, the main theorem of this section is stated and proven only for the log-loss. We then complete our presentation with a brief discussion on how the theorem is easily adapted to the exp-loss and comb-loss cases. In Section 5 we show how progress on every iteration leads to convergence to the global minimum of the respective loss function.

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 deal with a countably infinite number of base regressors. In the finite case we can simply map each input instance to the vector of images generated by each of the base-regressors, $\mathbf{x} \mapsto (h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$, where n is the number of 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 $\boldsymbol{\lambda} \cdot \mathbf{x}$. The j'th element of $\boldsymbol{\lambda}$, namely λ_j, should be regarded as the weight associated with the base regressor h_j.

Boosting was initially described and analyzed as a *sequential* algorithm that iteratively selects a single base-hypothesis or feature h_j and updates its weight λ_j. All of the elements of $\boldsymbol{\lambda}$ are initialized to zero, so after performing T sequential update iterations, at most T elements of $\boldsymbol{\lambda}$ are non-zero. Thus, this form of sequential update can be used for feature selection as well as loss minimization. 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. (2002) described a unified framework of boosting algorithms for *classification*. In that framework, the sequential and parallel update schemes become 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

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** $t = 1, 2, \ldots$

$\delta_{t,i} = \boldsymbol{\lambda}_t \cdot \mathbf{x}_i - y_i$

[if log-loss] $q_{t,i}^- = \dfrac{e^{\delta_{t,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}}$ $q_{t,i}^+ = \dfrac{e^{-\delta_{t,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}}$ $(1 \leq i \leq m)$

[if exp-loss] $q_{t,i}^- = e^{\delta_{t,i}}$ $q_{t,i}^+ = e^{-\delta_{t,i}}$ $(1 \leq i \leq m)$

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

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

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

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

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

Figure 3: A log-additive update algorithm for minimizing either the log-loss or the exp-loss.

by setting the templates accordingly, and allows us to discuss and prove the correctness of both paradigms 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 all of the instances $\mathbf{x}_i$ in the training set. The purpose of this requirement will become apparent in the proof of Theorem 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 Figure 3 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 and 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 Theorem 1 below. In the sequential version of the algorithm, selecting an update template is equivalent to selecting a single base regressor and updating its weight. In this case, the template selection criterion should be viewed as the *weak learning criterion* of the boosting procedure. The algorithm's iteration concludes with an update of $\boldsymbol{\lambda}$. Each element λ_j is updated by half the log ratio between the respective elements of W^+_t and W^-_t, times the scaling factor $a_{t,j}$.

The following theorem states a non-negative lower bound on the decrease in loss on every iteration of the algorithm for the case of the log-loss.

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 Figure 3, on every iteration t the decrease in the log-loss satisfies,*

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

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

$$\Delta_t(i) \;=\; L_{\log}(\delta_{t,i}) - L_{\log}(\delta_{t+1,i}). \tag{4}$$

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

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

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

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

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

$$\Delta_t(i) \;\geq\; q^-_{t,i}\left(1-e^{\mathbf{\Lambda}_t\cdot\mathbf{x}_i}\right) \;+\; q^+_{t,i}\left(1-e^{-\mathbf{\Lambda}_t\cdot\mathbf{x}_i}\right). \tag{5}$$

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

$$\begin{aligned}\mathbf{\Lambda}_t\cdot\mathbf{x}_i &= \sum_{j=1}^{n}\frac{a_{t,j}}{2}\log\left(W^+_{t,j}/W^-_{t,j}\right)x_{i,j}\\ &= \sum_{j=1}^{n}(a_{t,j}|x_{i,j}|)\;\mathrm{sign}(x_{i,j})\;\log\left(\sqrt{W^+_{t,j}/W^-_{t,j}}\right).\end{aligned} \tag{6}$$

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. This assumption is used in conjunction with the fact that $(1-e^{\alpha})$ is a concave function and is equal to zero at $\alpha = 0$. We can replace $\mathbf{\Lambda}_t \cdot \mathbf{x}_i$ in Eq. (5) with the form given in Eq. (6) and use Jensen's inequality to get,

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

We now rewrite,

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

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

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

This concludes the proof. ■

Theorem 1 focuses on the log-loss function, but is easily adapted to case of the exp-loss. Note that the only difference between the log-loss and exp-loss cases in the algorithm pseudo-code (Figure 3) is in the definitions of the overestimation and underestimation weights q^- and q^+. When our goal is to minimize the exp-loss, we define

$$q^-_{t,i} \;=\; e^{\delta_{t,i}} \qquad\qquad q^+_{t,i} \;=\; e^{-\delta_{t,i}}. \tag{7}$$

To show that Theorem 1 still holds for the exp-loss we modify the definition of $\Delta_t(i)$ from Eq. (4) in accordance to the change in the loss. Specifically, let

$$\begin{aligned}\Delta_t(i) &= L_{\exp}(\delta_{t,i}) - L_{\exp}(\delta_{t+1,i}) \\ &= e^{\delta_{t,i}} - e^{\delta_{t+1,i}} + e^{-\delta_{t,i}} - e^{-\delta_{t+1,i}}.\end{aligned}$$

As before, we plug the definitions of $q^+_{t,i}$ and $q^-_{t,i}$ from Eq. (7) into the above and rewrite Δ_t as,

$$\Delta_t(i) = q^-_{t,i}\left(1 - e^{\mathbf{\Lambda}_t \cdot \mathbf{x}_i}\right) + q^+_{t,i}\left(1 - e^{-\mathbf{\Lambda}_t \cdot \mathbf{x}_i}\right).$$

Eq. (5) in the proof of Theorem 1 now holds with equality and the rest of the proof proceeds as before. Consequently, we get the same lower bound for the exp-loss as was stated in Theorem 1 for the log-loss.

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

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

Again, the formal discussion given in this section carries over to the comb-loss case with only minor technical adaptations necessary.

To conclude this section, we note that the log-additive algorithm can be used verbatim in the case of a *weighted* loss. In Section 4 we use this extension to devise a simple regularization scheme. Formally, let $\boldsymbol{\nu} \in \mathbb{R}^m_+$ be a vector of non-negative weights such that ν_i is the weight of the i'th example. The weighted loss is defined as,

$$\mathrm{Loss}(\boldsymbol{\lambda}, \boldsymbol{\nu}, S) = \sum_{i=1}^{m} \nu_i L(\boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i),$$

where $L(\cdot)$ is any of the loss functions discussed above. The sole change to the algorithm resides in the calculation of the weights $q^+_{t,i}$ and $q^-_{t,i}$ which must now be scaled by ν_i, namely, $q^+_{t,i} \leftarrow \nu_i q^+_{t,i}$ and $q^-_{t,i} \leftarrow \nu_i q^-_{t,i}$. It is easy to verify that Theorem 1 still holds for this extended definition of weighted-loss.

3. 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 algorithms *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. The parallel update amounts to a gradient descent approach to minimizing the loss. The sequential update applied to an element

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. $\forall \mathbf{a} \in \mathcal{A}\ \ \sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \le 2$

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

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

$$\delta_{t,i} = \boldsymbol{\lambda}_t \cdot \mathbf{x}_i - y_i$$

[IF LOG-LOSS] $$q_{t,i}^- = \frac{e^{\delta_{t,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}} \qquad q_{t,i}^+ = \frac{e^{-\delta_{t,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}} \qquad (1 \le i \le m)$$

[IF EXP-LOSS] $$q_{t,i}^- = \frac{e^{\delta_{t,i}}}{Z_t} \qquad q_{t,i}^+ = \frac{e^{-\delta_{t,i}}}{Z_t} \qquad (1 \le i \le m)$$

$$\text{where } Z_t = \sum_{i=1}^m \left(e^{\delta_{t,i}} + e^{-\delta_{t,i}} + 2\right)$$

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

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

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

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

Figure 4: An additive update algorithm for minimizing either the log-loss or the exp-loss.

λ_j is an axis-parallel gradient descent step. 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 $\sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \le 2$.

The pseudo-code of the additive update algorithm is given in Figure 4. 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 Theorem 2, guarantees a maximal drop in the loss. Finally, $\lambda_{t,j}$ is updated by $a_{t,j} W_{t,j}$.

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 Figure 4, on every iteration t the decrease in the log-loss, denoted Δ_t, satisfies*

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

Proof We begin by defining a quadratic function $Q : \mathbb{R} \to \mathbb{R}$ which is parameterized by two parameters, $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}$. $Q_{\boldsymbol{\lambda},\boldsymbol{\Lambda}}$ will be shown to be an upper bound on the log-loss along the direction $\boldsymbol{\Lambda}$ from $\boldsymbol{\lambda}$. Concretely, $Q_{\boldsymbol{\lambda},\boldsymbol{\Lambda}}$ is defined as,

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

Formally, 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 Figure 4. 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 instead 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 a non-negative function 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,

$$\Gamma''(\alpha) \;=\; -\boldsymbol{\Lambda} \cdot \nabla \mathrm{Loss}(\boldsymbol{\lambda}, S) - \boldsymbol{\Lambda}^{\mathrm{T}} H \boldsymbol{\Lambda}, \tag{8}$$

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 function. It is simple to show that this derivative is bounded in $[0, 1/2]$. Plugging the value of H into Eq. (8) we get that,

$$\Gamma''(\alpha) \;\geq\; -\boldsymbol{\Lambda} \cdot \nabla \mathrm{Loss}(\boldsymbol{\lambda}, S) - \frac{1}{2}\sum_{i=1}^{m} (\boldsymbol{\Lambda} \cdot \mathbf{x}_i)^2. \tag{9}$$

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. (9) as,

$$\begin{aligned}
\Gamma''(\alpha) \quad &\geq \quad \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 - \frac{1}{2}\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{t,j} W_{t,j} x_{i,j}\right)^2 \\
&= \quad \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 - \frac{1}{2}\sum_{i=1}^{m} \left(\sum_{j=1}^{n} \sqrt{a_{t,j}}\, W_{t,j}\, \sqrt{a_{t,j}}\, x_{i,j}\right)^2 .
\end{aligned} \tag{10}$$

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

$$\begin{aligned}
\Gamma''(\alpha) \quad &\geq \quad \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 - \frac{1}{2}\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{t,j} W_{t,j}^2\right)\left(\sum_{k=1}^{n} a_{t,k} x_{i,k}^2\right) \\
&= \quad \sum_{j=1}^{n} a_{t,j} W_{t,j}^2 \left(1 - \frac{1}{2}\sum_{i=1}^{m}\sum_{k=1}^{n} a_{t,k} x_{i,k}^2\right).
\end{aligned} \tag{11}$$

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,

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

This concludes the proof. ■

To conclude this section, we briefly outline the adaptation of the additive update algorithm to the exp-loss. Recall that in the exp-loss setting, our goal is to minimize,

$$\sum_{i=1}^{m} \left(e^{\delta_i} + e^{-\delta_i} \right) \quad \text{where} \quad \delta_i = \boldsymbol{\lambda} \cdot \mathbf{x}_i - y_i .$$

Since the gradient of the exp-loss is itself exponential, we cannot hope to minimize the exp-loss by straightforward gradient descent. However, instead of minimizing the exp-loss function over the sample, we can minimize the loss,

$$\log \left(\sum_{i=1}^{m} \left(e^{\delta_i} + e^{-\delta_i} + 2 \right) \right) . \tag{12}$$

Clearly, both functions attain the same (global) minimum. We can now repeat verbatim the proof technique of Theorem 2 using q^- and q^+ as defined for the exp-loss case in Figure 4.

The additive update family of algorithms can accommodate a weighted loss just as log-additive update algorithms do. The algorithm is adapted to cope with weights in the same way that the log-additive algorithm was adapted in the end of Section 2, namely by an appropriate rescaling of the weights q^- and q^+.

4. 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. (Poggio and Girosi, 1990)). The loss functions discussed in the previous sections can also be used as a new form of regularization. Using the log-loss, we can apply the following regularization to the j'th coordinate of $\boldsymbol{\lambda}$,

$$\log \left(1 + e^{\lambda_j} \right) + \log \left(1 + e^{-\lambda_j} \right) .$$

The minimum of the above equation is obtained at $\lambda_j = 0$. It is straightforward to show that the regularization term above is bounded from below by $|\lambda_j|$ and from 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 of $\boldsymbol{\lambda}$. A similar form of regularization can be imposed using the exp-loss, namely,

$$e^{\lambda_j} + e^{-\lambda_j} .$$

For both losses, the j'th regularization term equals $L(\lambda_j \,; 0)$. When the set of base hypotheses is finite, 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 $\mathbf{x}_k = \mathbf{1}_k$ (the vector with 1 in its k'th position and zeros elsewhere). Let $\nu > 0$ be a regularization parameter that governs the relative importance of the regularization term with respect to the empirical loss. Slightly overloading our notation, let $\text{Loss}(\boldsymbol{\lambda}, \nu, S)$ denote the regularized empirical loss, defined by,

$$\text{Loss}(\boldsymbol{\lambda}, S) \;+\; \nu \, \text{Loss}(\boldsymbol{\lambda}, S_{\text{reg}}).$$

As noted in Section 2 and Section 3, both the log-additive and additive update batch algorithms easily accommodate a weighted loss. Therefore, by introducing a set of n pseudo-examples, each of which weighted by ν, we can incorporate regularization into our batch algorithms without any modification to the algorithm core. Concretely, we set the weight of each example in S to 1 and of each pseudo-example in S_{reg} to ν. We can now use either the log-additive or the additive algorithm to minimize the weighted loss.

In practice, we do not need to explicitly add pseudo-examples to our sample in order to incorporate a regularization term into the loss function. A more efficient way of achieving the same effect is to modify our algorithms to behave as if such a pseudo sample was presented to them. For instance, for the log-additive log-loss update (Figure 3) the term $\nu/(1+e^{-\lambda_{t,j}})$ should be added to the definition of $W^-_{t,j}$ for every coordinate being updated. Analogously, the term $\nu/(1+e^{\lambda_{t,j}})$ should be added to $W^+_{t,j}$. Applying this modification is equivalent to adding pseudo-examples which correspond to the coordinates being updated.

Another useful property of this regularized loss is that it is *strictly* convex. To see that $\text{Loss}(\boldsymbol{\lambda}, \nu, S)$ is strictly convex it suffices to show that its Hessian is positive definite. The Hessian of $\text{Loss}(\boldsymbol{\lambda}, \nu, S)$ can be written as a sum of two matrices $H + H_{\text{reg}}$ where the first is the matrix of second order derivatives of $\text{Loss}(\boldsymbol{\lambda}, S)$ and the second contains the second order derivatives of $\nu\,\text{Loss}(\boldsymbol{\lambda}, S_{\text{reg}})$. Since $\text{Loss}(\boldsymbol{\lambda}, S)$ is the sum of convex losses, H is positive semi-definite. It is simple to verify that the matrix H_{reg} is a diagonal matrix with $H_{i,i} = 2\nu/\left(1+e^{-\lambda_i}\right)\left(1+e^{\lambda_i}\right)$ for the log-loss or $H_{i,i} = \nu\left(e^{-\lambda_i}+e^{\lambda_i}\right)$ for the exp-loss. Clearly, the diagonal elements are strictly positive for both losses for any finite $\boldsymbol{\lambda}$. Therefore, H_{reg} is positive definite and thus $H + H_{\text{reg}}$ is positive definite as well. Furthermore, since the regularization term tends to infinity at least as fast as $\|\boldsymbol{\lambda}\|_1$, the regularized loss has an *attainable* global minimum. In other words, this form of regularization enforces uniqueness of the solution in our loss minimization problem. We denote the unique global minimum of $\text{Loss}(\boldsymbol{\lambda}, \nu, S)$ by $\boldsymbol{\lambda}^\star$. We use the uniqueness of $\boldsymbol{\lambda}^\star$ in the next section where the convergence of our batch algorithms is discussed.

5. Convergence

In the previous section we have argued that the regularized loss attains a unique minimum at the point denoted $\boldsymbol{\lambda}^\star$. In this section we show that the batch algorithms described so far converge to this unique minimizer. For simplicity, we assume that the set of templates $\mathcal{A}$ spans $\mathbb{R}^n$. The following theorem can be tediously generalized to the case where the space spanned by $\mathcal{A}$ is any linear-subspace of $\mathbb{R}^n$ in which case convergence is to the optimal value within this subspace.

Theorem 3 *Assume that the vectors in $\mathcal{A}$ span the entire space $\mathbb{R}^n$. Let $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_t, \ldots$ be the sequence of vectors generated by the log-additive (Figure 3) or the additive (Figure 4) updates, using either of the regularized loss functions discussed in this paper. Then this sequence converges to $\boldsymbol{\lambda}^\star$, the global minimizer of the regularized loss $L(\boldsymbol{\lambda}, \nu, S)$.*

Proof Due to the introduction of the regularization term, the loss function is strictly convex and attains its unique minimum at the point denoted $\boldsymbol{\lambda}^\star$, as argued in the previous section. In addition, the regularization term guarantees that the entire sequence $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \ldots$ lies

within a compact set C. To see this, note that $\boldsymbol{\lambda}$ is initialized to be the zero vector and therefore the initial regularized loss is

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

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 certainly does not 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

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

Therefore, we can define $C = \{\boldsymbol{\lambda} : \|\boldsymbol{\lambda}\|_\infty \leq \mathcal{L}_0/\nu\}$ and assert that the sequence $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ is contained in C. Next, note that the lower bound on the decrease in loss given in Theorem 1 and Theorem 2 can be thought of as a function of the current regressor $\boldsymbol{\lambda}_t$ and the chosen template $\mathbf{a}_t$. If the bound on the decrease equals zero for all possible $\mathbf{a} \in \mathcal{A}$ then $\boldsymbol{\lambda}_t$ must be equal to $\boldsymbol{\lambda}^\star$. Otherwise, there exists $\mathbf{a} \in \mathcal{A}$ for which the decrease bound is strictly positive. To see this, note that if the decrease bound for the log-additive update is 0 then,

$$\forall j : \quad \left(\sqrt{W_{t,j}^-} - \sqrt{W_{t,j}^+}\right)^2 = 0,$$

which implies that $W_{t,j}^- - W_{t,j}^+ = 0$. Note that $W_{t,j}^- - W_{t,j}^+$ is the j'th partial derivative of the loss function being minimized (log-loss, exp-loss, or comb-loss). Since the regularized loss function is strictly convex, a zero gradient vector is attained only at the optimal point $\boldsymbol{\lambda}^\star$. A similar argument holds for the additive update.

Assume now 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 a compact set and therefore the lower bound from Theorem 1 (or equivalently Theorem 2) attains a minimum value over $C \setminus B(\boldsymbol{\lambda}^\star, \gamma)$ at some point $\tilde{\boldsymbol{\lambda}}$. Denote this minimum by μ. Since $\tilde{\boldsymbol{\lambda}} \notin B(\boldsymbol{\lambda}^\star, \gamma)$ it necessarily follows that $\tilde{\boldsymbol{\lambda}} \neq \boldsymbol{\lambda}^\star$ and therefore μ must be strictly positive. Thus, on each of the iterations $s_1, s_2, \ldots$ the decrease in loss is at least $\mu > 0$. The subsequence $s_1, s_2, \ldots$ is infinite and as a consequence the loss must eventually become negative. We get a contradiction since the loss is a non-negative function. We conclude that the sequence $\boldsymbol{\lambda}_t$ must converge to $\boldsymbol{\lambda}^\star$. ■

6. Back to Classification

To conclude the part of the paper which discusses batch algorithms we would like to briefly draw connections to boosting algorithms for classification. The reader mainly interested in regression problems may skip this section.

The algorithms of previous sections can also be used in classification settings. The log-additive updates simply reduce to the algorithms described in (Collins et al., 2002). The additive update results in a new boosting procedure for classification accompanied with a matching criterion for selecting a base hypothesis. Concretely, in the binary classification

setting we receive a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ where each target y_i is either -1 or $+1$. As in the case of regression, $\mathbf{x}$ is the mapping of an instance into its image under the set of base-classifiers, $\mathbf{x} \mapsto (h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$ and the goal is to find a function $f(\mathbf{x})$ that attains a small loss. The function $f(\mathbf{x})$ is a weighted combination of base-hypotheses, $f(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j h_j(\mathbf{x}) = \boldsymbol{\lambda} \cdot \mathbf{x}$. The skeleton of the additive algorithm for classification is almost the same as the one for regression. In the classification case we define a single (unnormalized) distribution over the examples, setting the weight of the i'th example to,

$$q_{t,i} = e^{-\delta_{t,i}}/Z_t \quad [\text{ EXP-LOSS }] \quad ; \quad q_{t,i} = \frac{1}{1 + e^{-\delta_{t,i}}} \quad [\text{ LOG-LOSS }],$$

where $\delta_i = y_i \boldsymbol{\lambda}_t \cdot \mathbf{x}_i$ and $Z_t = 1 + \sum_{i=1}^{m} e^{-\delta_{t,i}}$. On round t we set each variable $W_{t,j}$ to be $W_{t,j} = \sum_{i=1}^{n} q_{t,i}\, x_{i,j}$. The rest of the algorithm, including the constraint $\sum_{i,j} a_j x_{i,j}^2 \leq 2$, is kept intact. The result is a new boosting-type procedure where base-hypotheses are selected so as to maximize $\sum_j a_j W_{i,j}^2$.

The regularization technique discussed in Section 4 can be used in classification tasks as well. It is worth noting that Schapire et al. (2002) suggested a procedure for incorporating prior knowledge into log-loss boosting which can also be used for regularization. We compare the two regularization techniques in the log-loss case. Using the notation of Section 4, the regularization technique of Schapire et al. can also be described via the introduction of a pseudo-sample. Given a training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ with $y_i \in \{-1, +1\}$ define the pseudo-sample $\bar{S} = \{\mathbf{x}_i, -y_i\}$ and use log-loss boosting to train a classifier whose task is to minimize the loss,

$$(1 - \nu)\text{Loss}(\boldsymbol{\lambda}, S) + \nu \text{Loss}(\boldsymbol{\lambda}, \bar{S}),$$

where, as before, ν is a regularization parameter. In this case ν is restricted to the interval $[0, 1/2]$. This construction of a regularization sample implies that even when there exists a strong-hypothesis which attains zero classification error on S the extended sample $S \cup \bar{S}$ is inseparable. If the space spanned by the examples is of a full rank, then this regularization scheme guarantees a unique and attainable global minimizer $\boldsymbol{\lambda}^\star$. However, the two optima due to the two different regularization schemes will be achieved at different points. The regularization scheme presented in this paper penalizes large values of $|\lambda_j|$ whereas Schapire et al. penalize overconfident predictions.

7. Online Regression Algorithms

In this section we describe online regression algorithms for the log-loss defined in Eq. (1). In the previous sections we allowed ourselves to ignore the constant κ which appears in the definition of the log-loss since this did not alter the global minimum of our problem. However, in the online learning setting this constant should not be ignored. Inclusion of κ does not affect the online algorithms themselves as they depend only on the gradient of the loss function, but it will play a role in their analysis.

We follow the notation and techniques presented in (Kivinen and Warmuth, 1997; Cesa-Bianchi, 1999). 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 equal to $L_{\log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t)$. Our goal is to suffer a small cumulative loss.

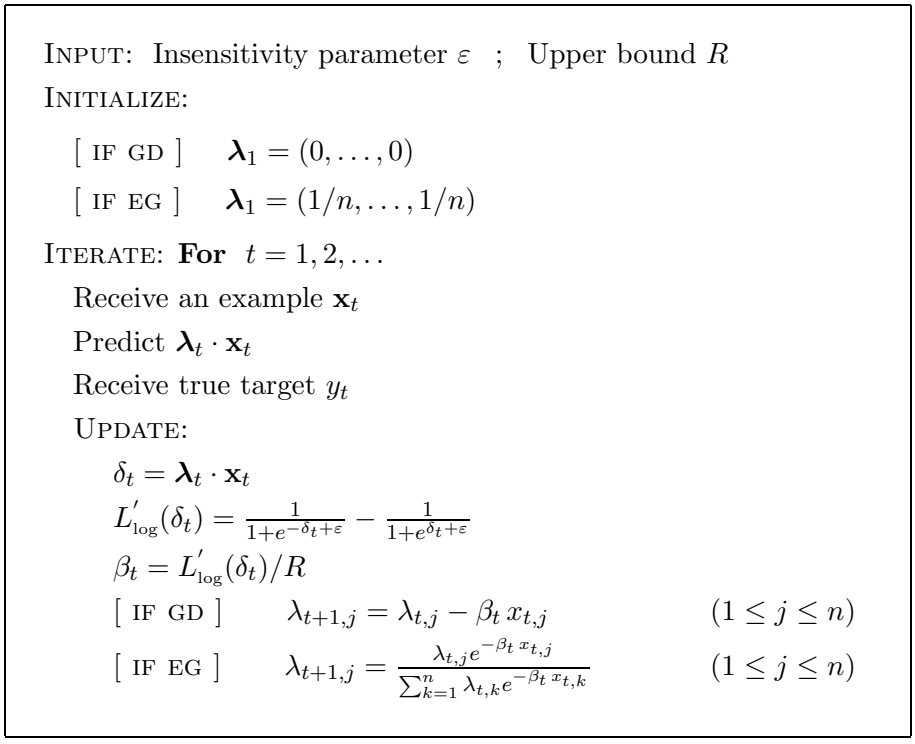

Figure 5: The GD and EG algorithms for online regression with the log-loss.

The learning algorithm employs an update rule which modifies its current regressor after each round. We describe and analyze two online regression algorithms for the log-loss 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 in Figure 5. 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 twice the squared norm of all the instances, that is, $2\|\mathbf{x}_t\|_2^2 \le R$. In the following analysis we give a bound on the cumulative loss attained on 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 4 *Let $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ be a sequence of instance-target pairs such that $\forall t : 2\|x_t\|_2^2 \le R$ and let $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T$ be the regressors generated by the GD online algorithm*

(Figure 5) on the sequence. Then for any fixed linear regressor $\boldsymbol{\mu} \in \mathbb{R}^n$ we have

$$\sum_{t=1}^{T} L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) \;\leq\; 2 \sum_{t=1}^{T} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;+\; R \left\|\boldsymbol{\mu}\right\|_2^2 \,. \tag{13}$$

Note that the statement of the theorem changes if the constant κ is excluded from the definition of the log-loss in Eq. (1), resulting in a looser bound. The proof of the theorem is based on the following lemma that underscores an invariant property of the update rule.

Lemma 5 *Consider the setting of Theorem 4, then for each round t we have*

$$L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - 2 L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;\leq\; R \left(\left\|\boldsymbol{\lambda}_t - \boldsymbol{\mu}\right\|_2^2 - \left\|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\mu}\right\|_2^2 \right) \,. \tag{14}$$

The proof of the lemma is given in Appendix A. Intuitively, the lemma states that if the loss attained by $\boldsymbol{\lambda}_t$ on round t is greater than the loss of a fixed regressor $\boldsymbol{\mu}$, then the algorithm will update $\boldsymbol{\lambda}_t$ such that it gets closer to $\boldsymbol{\mu}$. 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 5 handy, the proof of Theorem 4 is almost immediate.

Proof of Theorem 4: Summing Eq. (14) for $t = 1, ..., T$ we get

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

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 in Figure 5 and works under the assumption that the regressor $\boldsymbol{\lambda}$ is contained in the probability simplex, namely $\boldsymbol{\lambda} \in \mathbb{P}^n$ where $\mathbb{P}^n = \{\boldsymbol{\mu} \;:\; \boldsymbol{\mu} \in \mathbb{R}_+^n,\; \sum_{j=1}^n \mu_j = 1\}$. We note in passing that following a construction described in (Kivinen and Warmuth, 1997), 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 on the squared difference between the maximal and minimal coordinates of the instances it receives, $R \geq (\max_j x_{t,j} - \min_j x_{t,j})^2$. 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.

Theorem 6 *Let $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ be a sequence of instance-target pairs such that $\forall t : (\max_j x_{t,j} - \min_j x_{t,j})^2 \leq R$ and let $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T$ be the regressors generated by the EG online algorithm (Figure 5) on the sequence. Then, for any fixed regressor $\boldsymbol{\mu} \in \mathbb{P}^n$ we have*

$$\sum_{t=1}^{T} L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) \;\leq\; \frac{4}{3} \sum_{t=1}^{T} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;+\; \frac{4R}{3} \, D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_1) \,, \tag{15}$$

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

The proof of the theorem is analogous to the proof of Theorem 4 and employs the following relative entropy based progress lemma.

Lemma 7 *Consider the setting of Theorem 6, then for each round t we have*

$$L_{log}(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) - \frac{4}{3} L_{log}(\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t) \;\leq\; \frac{4R}{3} \left(D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_t) - D_{RE}(\boldsymbol{\mu}, \boldsymbol{\lambda}_{t+1}) \right) \;. \qquad (16)$$

The proof of the lemma is given in Appendix A.

8. Experiments

In this section we present experimental results that demonstrate different aspects of our algorithms in the light of their formal analysis. In Section 8.1, we start with a synthetic example that underscores the different properties of the log-loss and the exp-loss functions. We then turn to a comparison of the different algorithmic approaches to minimizing these losses. In Section 8.2 we compare the log-additive and additive batch updates by examining how certain properties of the training data influence the rates of convergence of the two updates. Next we demonstrate the different benefits of the sequential and parallel update paradigms. In Section 8.3 we use the sequential form of our update as a boosting procedure which uses regression stumps as base regressors. We compare this boosting technique to the LAD algorithm (Friedman, 2001) on a natural data set. In Section 8.4 we turn our attention to the parallel update paradigm. When using the parallel update, some form of regularization is essential to avoid over-fitting. We therefore demonstrate the effectiveness of the regularization scheme presented in Section 4 and show the effect of the regularization parameter on the generalization ability of our algorithms. More specifically, we learn a kernel-based regressor and compare our log-loss regularization to Support Vector Regression with l_1 regularization. The last two experiments illustrate some properties of our online algorithms and are presented in Section 8.5. In these experiments, we compare the cumulative loss of the online GD algorithm with its theoretical bound given in Theorem 4. We also compare the sensitivity of the online GD and EG regression algorithms to the number of relevant coordinates in the data, demonstrating that EG vastly outperforms GD when the number of relevant coordinates is small.

8.1 Comparison of the Exp-Loss and the Log-Loss

In this section we describe an experiment that underscores the different merits of the log-loss and the exp-loss functions. The end result is that the solution obtained by minimizing the log-loss shares the same asymptotic behavior as the ℓ_1 regression loss ($\sum_i |\delta_i|$). On the other hand, the solution found by minimizing the exp-loss approximately minimizes the l_∞ regression loss on the sample. To exemplify the above properties we created two synthetic data sets and for each one we found the two regressors that minimize the log-loss and the exp-loss respectively. The two data sets and the resulting regressors are depicted in Figure 6. Each of the two data sets was generated by sampling points on the curve of a univariate third degree polynomial, resulting in a sample $S = \{(x_i, y_i)\}$ where $x_i, y_i \in \mathbb{R}$. Then, the target y_i of each point $\mathbf{x}_i$ was contaminated with a small additive noise distributed

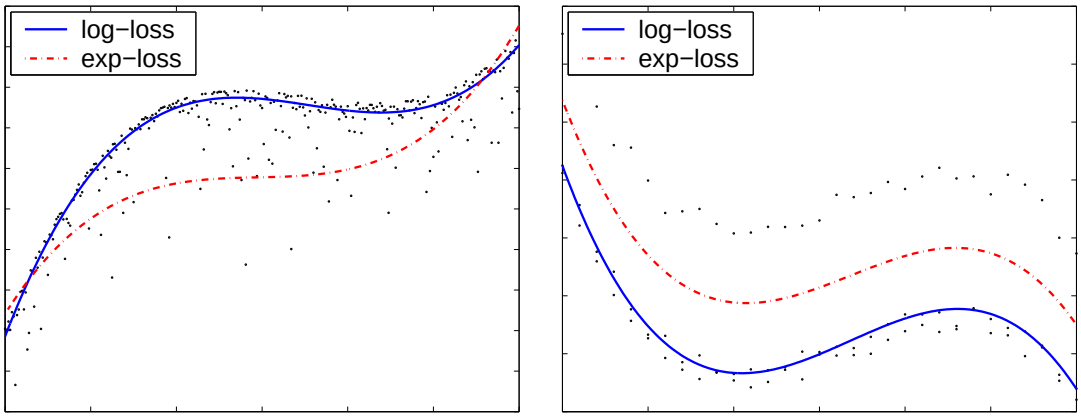

Figure 6: A comparison of log-loss and exp-loss on synthetic data.

normally with a zero mean and a variance of 0.1. In the first experiment, the targets were further contaminated by adding one-sided noise which was generated by subtracting the absolute value of a normal variable with a zero mean and a unit variance (Figure 6, left). Each instance $\mathbf{x}_i$ was expanded by taking powers of x_i, i.e. we performed the mapping $x_i \mapsto (1, x_i, x_i^2, x_i^3)$. This expansion enables us to use our linear algorithms to learn degree three polynomials. It is clear from the figure that the regressor obtained by minimizing the log-loss is very close to the polynomial generating the data, demonstrating the robustness of the log-loss to biased noise. The regressor attained by minimizing the exp-loss, however, approximately minimizes the maximal discrepancy over the entire data set and therefore lies significantly below. The other facet of this behavior is illustrated on the right hand side of Figure 6. In this data set, the additional one sided noise was set to *one* with a probability of 1/3 and otherwise it was set to zero. Thus, about a third of the targets were shifted up by 1. Here, 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 samples that were shifted by 1 and as such approximately minimizes the ℓ_1 regression loss on the sample.

8.2 A Comparison of the Log-Additive and Additive Updates

In this section we compare the performance of the log-additive update from Figure 3 to that of the additive update from Figure 4. An important difference between the two updates is the type of constraint imposed on the norm of the update templates. In the following, we demonstrate the effect of this difference on the convergence rates of the two update strategies. To make the comparison as simple as possible, we chose the instance space to be $\mathbb{R}^1$. Therefore, the instances are scalars and there is a single update template $a \in \mathbb{R}$. For the log-additive update the constraint on a becomes $a \max_i |x_i| \leq 1$ while for the additive update the constraint is $a \sum_i x_i^2 \leq 2$. To demonstrate the implications of the different norm constraints, we generated two synthetic data sets. The target of each instance was set for both data sets to be equal to the input instance, that is, $y_i = x_i$. Each data set consists of 26 instance-target pairs. For both data sets, we set the value of the first 25 instances to equal 0.01. In the first data set we set the last instance to 0.5 whereas in the second data set

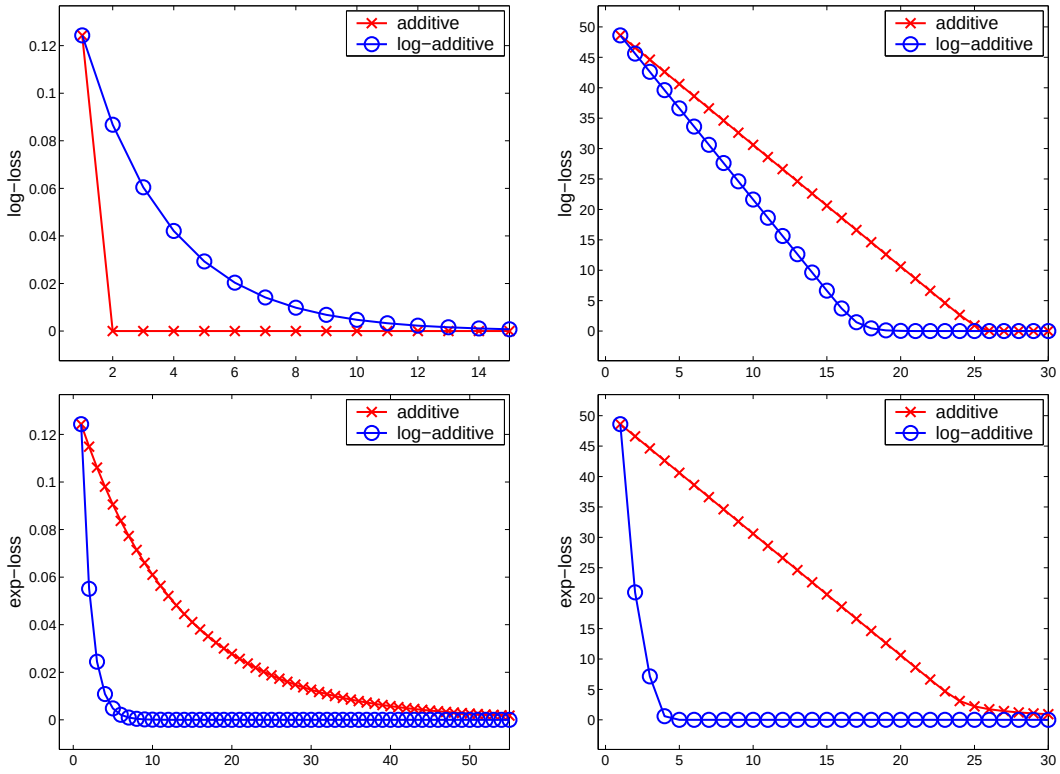

Figure 7: Comparison of the convergence rates of the log-additive and additive updates on two different data sets (see text). The left column corresponds to the first data set and the right column to the second. The top row presents results for the log-loss while the bottom row presents results for the exp-loss.

we set it to 10. Therefore, for the first data set, the constraint on a reduces to $a \leq 2$ when using the log-additive update and to $a \leq 4$ when the additive update is used. In the case of the second data set, the constraint on a becomes $a \leq 0.02$ for the log-additive update, and $a \leq 0.0008$ for the additive one. We would like to note in passing that for the additive update, a typically decreases as the number of examples increases. Hence, the steps that additive update takes are likely to be smaller in large data sets. The end result is slower convergence rates as both the log-additive and the additive updates scale linearly with the value of the template used. Put another way, a small value of a yields an update which changes $\boldsymbol{\lambda}$ rather conservatively. Therefore, in the settings discussed in this section, the additive update should converge faster on the first data set while the log-additive update should converge faster on the second data set. The top row of Figure 7 shows the log-loss obtained on the training set as a function of the number of iterations for the two data sets. It is clear from the graphs that our expectations are met and that the additive update converges faster than the log-additive update on the first data set and slower on the second data set.

Another important difference between the two updates is the construction of q_i^+ and q_i^- when minimizing the exp-loss. Recall that in the case of the log-additive update, the weights of the examples are $q_i^+ = e^{\delta_i}$ and $q_i^- = e^{-\delta_i}$ while for the additive update we further divide these weights by Z. Therefore, when the data set contains examples for which the exp-loss cannot be made small, the value of Z is likely to be rather large. Unlike the log-additive update, the additive form is *sensitive* to scaling of the weights q_i^+ and q_i^-. Thus, whenever Z is large, the resulting normalized weights will be small and therefore the corresponding step sizes taken by the additive update will also be small. The bottom row of Figure 7 reflects this sensitivity of the additive update to scaling. For both data sets described above the log-additive update exhibits much faster convergence than the additive-one.

8.3 Boosting Regression Stumps

The next experiment demonstrates the effectiveness of the log-additive and additive updates in their sequential form, when they are applied as boosting procedures. As in the classic boosting setting, our algorithm has access to an external learning procedure called a base or weak learner. The goal of the boosting algorithm is to construct a highly accurate regressor by combining base regressors obtained from consecutive calls to the base learner. On every boosting iteration the base learner receives the training set along with the weights $q_{t,i}^+$ and $q_{t,i}^-$ generated by the boosting algorithm. The goal of the base learner is to construct a regressor which maximizes the decrease in loss. We denote by $h_t : \mathbb{R}^n \to \mathbb{R}$, the regressor returned by the base learner on round t. We use either the bound in Theorem 1 or the bound in Theorem 2 as the criterion for selecting a base regressor using the log-additive and additive updates respectively. That is, the base learner attempts to maximize the lower bound on the decrease in loss given in Theorem 1 or Theorem 2.

In our experiments, we use *regression stumps* as base regressors. Like decision stumps which are depth-one decision trees, regression stumps are the simplest form of regression trees (cf. Friedman (2001)). Each stump is characterized by two parameters: a feature index parameter, $\ell \in \{1, \ldots, n\}$ and a threshold parameter $\theta \in \mathbb{R}$. The prediction of each stump is either -1 or $+1$ and is defined as $h(\mathbf{x}) = \text{sign}(\theta - x_\ell)$. We now describe the specific base learner we use. Given a training set $S = \{(\mathbf{x}_i, y_i)\}$ of m instance-target pairs, we construct for each feature index $\ell \in \{1, \ldots, n\}$ a set of candidate thresholds. Each set consists of all possible mid-points between two consecutive values of that feature on the training set. Formally, let Θ_ℓ denote the candidate thresholds set for feature ℓ and let $x_{i,\ell}$ denote the ℓth feature of the ith instance in S. Then, the set Θ_ℓ is defined as,

$$\Theta_\ell = \left\{ (x_{i_1,\ell} + x_{i_2,\ell})/2 \,|\, x_{i_1,\ell} < x_{i_2,\ell} \text{ and } \nexists r \text{ s.t. } x_{i_1,\ell} < x_{r,\ell} < x_{i_2,\ell} \right\}. \tag{17}$$

Note that each set Θ_ℓ may contain at most $m-1$ different thresholds and can be *pre-computed* efficiently in time $m \log(m)$ by sorting the training set independently for each feature.

Given the current set of weights, $q_{t,i}^+$ and $q_{t,i}^-$, the base learner constructs a regression stump by choosing a feature index ℓ and a threshold value $\theta \in \Theta_\ell$. This pair is chosen so as to maximize the bound on the decrease in the log-loss as defined in Theorem 1 or in Theorem 2. It is easy to verify that the value of an update template for each base regressor is either $2/m$ in the case of the additive update or 1 in the case of the log-additive update

Input: Training set $S = \{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}\}_{i=1}^m$; Insensitivity $\varepsilon \in \mathbb{R}_+$; Number of iterations T

Initialization: compute the set of admissible thresholds Θ_ℓ $(\ell = 1, \ldots, d)$

Iterate:

For $t = 1, 2, \ldots, T$

$$\delta_{t,i} = \sum_{j=1}^{t-1} \boldsymbol{\lambda}_j h_j(\mathbf{x}_i) - y_i$$

$$q^-_{t,i} = \frac{e^{\delta_{t,i}-\varepsilon}}{1+e^{\delta_{t,i}-\varepsilon}} \quad , \quad q^+_{t,i} = \frac{e^{-\delta_{t,i}-\varepsilon}}{1+e^{-\delta_{t,i}-\varepsilon}} \qquad (1 \le i \le m)$$

Define:

[if log-additive]

$$\Delta(\theta, \ell) = \left(\sqrt{\sum_{i\,:\,x_{i,l}>\theta} q^-_{t,i} + \sum_{i\,:\,x_{i,l}<\theta} q^+_{t,i}} - \sqrt{\sum_{i\,:\,x_{i,l}>\theta} q^+_{t,i} + \sum_{i\,:\,x_{i,l}<\theta} q^-_{t,i}} \right)^2$$

[if additive]

$$\Delta(\theta, \ell) = \frac{1}{m} \left(\sum_{i\,:\,x_{i,l}>\theta} (q^+_{t,i} - q^-_{t,i}) + \sum_{i\,:\,x_{i,l}<\theta} (q^-_{t,i} - q^+_{t,i}) \right)^2$$

Set $(\theta^\star, \ell^\star) = \operatorname*{argmax}_{(\theta,\ell)} \Delta(\theta, \ell)$

Set $h_t(\mathbf{x}) = \operatorname{sign}(\theta^\star - x_{\ell^\star})$

Update:

[if log-additive] $\lambda_t = \frac{1}{2} \log \left(\frac{\sum_{i:h_t(\mathbf{x}_i)\ge 0} q^+_{t,i} + \sum_{i:h_t(\mathbf{x}_i)<0} q^-_{t,i}}{\sum_{i:h_t(\mathbf{x}_i)\ge 0} q^-_{t,i} + \sum_{i:h_t(\mathbf{x}_i)<0} q^+_{t,i}} \right)$

[if additive] $\lambda_t = \frac{2}{m} \sum_{i=1}^m (q^+_{t,i} - q^-_{t,i}) h_t(\mathbf{x}_i)$

Output: $f(\mathbf{x}) = \sum_{t=1}^T \lambda_t h_t(\mathbf{x})$

Figure 8: The stumps-based regression algorithm.

since the output of the base regressors is either $+1$ or -1. Hence, given a candidate feature index ℓ and a threshold $\theta \in \Theta_\ell$ the bound on the decrease in loss for the additive update is,

$$\frac{1}{m}\left(\sum_{i\,:\,x_{i,l}>\theta}(q^+_{t,i}-q^-_{t,i})+\sum_{i\,:\,x_{i,l}<\theta}(q^-_{t,i}-q^+_{t,i})\right)^2, \tag{18}$$

and for the log-additive update the bound is,

$$\left(\sqrt{\sum_{i\,:\,x_{i,l}>\theta}q^-_{t,i}+\sum_{i\,:\,x_{i,l}<\theta}q^+_{t,i}}-\sqrt{\sum_{i\,:\,x_{i,l}>\theta}q^+_{t,i}+\sum_{i\,:\,x_{i,l}<\theta}q^-_{t,i}}\right)^2. \tag{19}$$

As mentioned above, the base learner evaluates one of the above terms (depending on the update) for each possible ℓ and $\theta \in \Theta_\ell$. It then chooses the pair which maximizes either Eq. (18) or Eq. (19). The pseudocode of the regression learning algorithm using stumps for both the additive and the log-additive updates is given in Figure 8.

We compared the regression algorithm with stumps to an algorithm named Least Absolute Deviation (LAD) due to Friedman (2001). LAD is a boosting-style algorithm which attempts to minimize the hinge-loss by fitting a base hypothesis to the residual error, the approximation error left after applying the combination of base hypotheses found so far. It is not obvious how to conduct a fair comparison between our approach and LAD since the direct objective of our regression learning algorithm is to minimize the log-loss while the goal of LAD is to minimize the hinge-loss. To remove any doubt on the validity of the results, we evaluate both algorithms using the hinge-loss, thus giving a slight advantage to LAD in our empirical evaluation. In addition, we compared the mean squared errors (MSE) of the algorithms.

We ran experiments on two standard data sets for regression: the *Boston housing* data set from the UCI repository and the *body fat* data set (Penrose et al., 1985). To evaluate our results, we used a 10-fold cross validation technique. The plots on the top row of Figure 9 depict the average hinge-loss on the two data sets as a function of the number of sequential iterations while the plots on the bottom row correspond to the mean squared error obtained by the algorithms on the same data sets. The plots underscore a few interesting phenomena. The LAD algorithm appears to be able to decrease the hinge loss and the MSE much faster than our algorithm on both the training data (not shown) and the test data. In no more than 3 iterations, LDA is able to achieve rather low loss. It takes about an order of magnitude more iterations for our algorithm to obtain the same performance LAD achieves after 2 or 3 iterations. This behavior can be partially attributed to the fact that LAD is designed to directly maximize the decrease in loss while our algorithm maximizes a lower bound on the decrease in loss. However, despite its initial performance, LDA seems to "get stuck" rather quickly and the final regressor it obtains has substantially higher loss on both data sets compared to our algorithm, whether it is trained with the log-additive update or the additive one. The improved generalization performance may be attributed to the following behavior that is common to boosting algorithms: as more base regressors are added, the regression error obtained on most of the examples is rather small. Thus, the weights q^+_i and q^-_i for most of the examples are also small and do not contribute too much to further

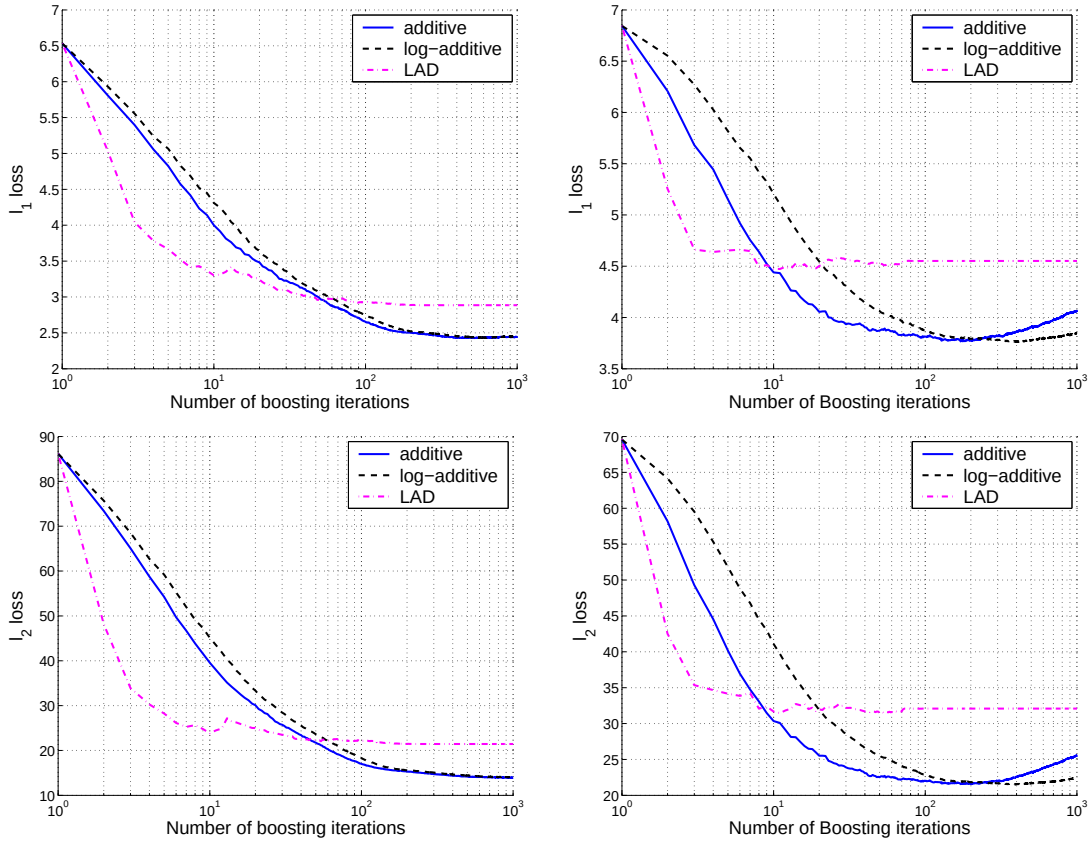

Figure 9: A comparison of the ℓ_1 and MSE losses obtained by the regression algorithm with stumps and the LAD algorithm (see text) on Boston housing data set (left) and body-fat (right) data sets.

decreases in the loss. Thus, even simple regressors such as decision stumps can further reduce the loss on the remaining examples for which the loss is still high. Indeed, we see that some over-fitting takes place when our regression algorithm is run for more than 200 iterations.

Comparing the performance of the additive and the log-additive updates in this experiment, it is apparent that the former seems more effective in reducing the loss but is also more susceptible to over-fitting. The accuracy of each update strategy seems to be problem dependent. We leave further theoretical and empirical research on the generalization properties of the two updates to future research.

8.4 Examining the Effect of Regularization

So far we have focused on experiments which illustrate the different facets of empirical loss minimization using different update schemes. In this section we shift our focus to the effects of the regularization technique discussed in Section 4. The role of regularization is to control the complexity of the final regressor. Indeed, as we demonstrate empirically, proper

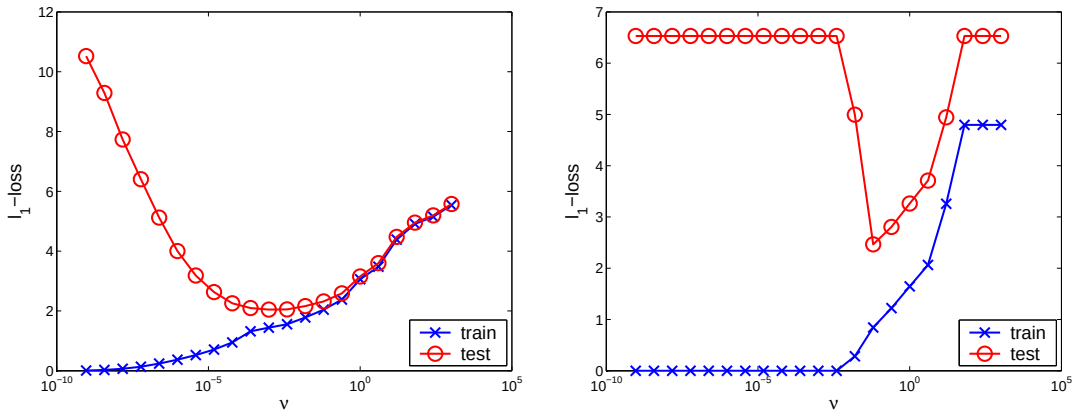

Figure 10: The training and test losses as a function of the regularization parameter (ν) for the log-loss (left) and Support Vector Regression (right).

regularization can ensure that the generalization loss (i.e. the regression loss suffered on test examples) would not greatly exceed the loss obtained on the training set. The feature space we use in this experiment is based on kernel operators. Concretely, the regressors we construct take the form

$$f_{\boldsymbol{\lambda}}(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j k(\mathbf{x}_j, \mathbf{x}),$$

where $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ are the instances in the training set and k is a kernel function. We used the log-additive and additive update algorithms to minimize the following regularized loss,

$$\text{Loss}(\boldsymbol{\lambda}, \nu, S) \;=\; \sum_{i=1}^{m} L_{\log}(f_{\boldsymbol{\lambda}}(\mathbf{x}_i) - y_i \,;\, \varepsilon) \;+\; \nu \sum_{j=1}^{m} L_{\log}(\lambda_j). \tag{20}$$

As illustrated in Figure 1, the log-loss can be interpreted as a smooth approximation to the ε-insensitive hinge loss used by Support Vector Regression (SVR). SVR is a technique for non-linear regression which uses kernel functions. For a thorough review of SVR, see for instance (Smola and Schölkopf, 1998). In our setting, the regularized loss from Eq. (20) can be viewed as a smooth approximation to the hinge-loss with l_1 regularization that is used in Linear Programming Support Vector Regression (LP-SVR), namely,

$$\sum_{i=1}^{m} |f_{\boldsymbol{\lambda}}(\mathbf{x}_i) - y_i|_{\varepsilon} \;+\; \nu \sum_{j=1}^{m} |\lambda_j|. \tag{21}$$

We ran experiments using the *Boston Housing* data set from the UCI Machine Learning Repository. Following Bi and Bennett (2003), we chose to use a Gaussian kernel with $2\sigma^2 = 3.9$. We ran experiments with ϵ set to $0, 1, 2, 3$. The preprocessing we performed consisted of shifting and scaling the input variables to the unit hypercube. Specifically, let $r_j = \min_i x_{i,j}$ and $s_j = \max_i x_{i,j}$, then $x_{i,j}$ was transformed to $(x_{i,j} - r_j)/(s_j - r_j)$. As in the

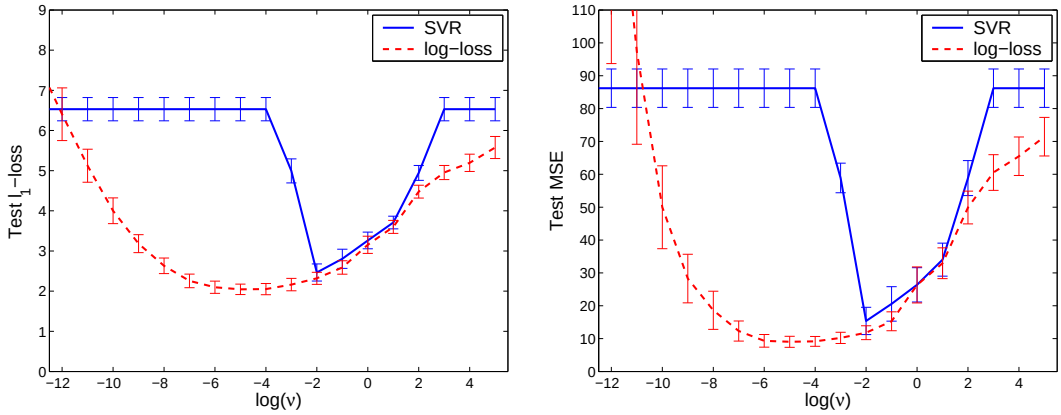

Figure 11: Comparisons of the test losses obtained by the regularized log-loss minimization procedure and by SVR as a function of ν. The losses used for evaluation are the ℓ_1 loss (left) and the mean squared error (right). The standard deviation of over the cross validation fold is depicted as error bars.

previous experiment, we used 10-fold cross validation to evaluate the results and measured the hinge-loss and the mean-squared error (MSE).

The train and test hinge losses for different values of the regularization parameter ν are depicted in Figure 10. As anticipated, the training loss for both algorithms is monotonically increasing in the regularization parameter ν while the difference between the test and training loss is monotonically decreasing in ν. This behavior is typical of regularization techniques. On the left hand side of Figure 11 we directly compare the test error obtained by the two algorithms. The standard deviation over the ten folds is shown using error bars. The MSE of the algorithms is given on the right hand side of Figure 11. As can be seen form the figure, the lowest test loss attained by SVR is very close to the value attained by the log-loss (with a slight advantage to the latter). However, the regressors obtained by the log-loss seem to be less sensitive to the particular choice of ν than the regressors obtained by SVR. Indeed, for ν in $[10^{-5}, 1]$, the discrepancy between the losses of the regressors found by the log-loss is less than 1 while in the same range the losses of SVR can be as much as 3 units apart. This behavior suggests that regression methods which use the smooth log-loss function may give a viable alternative to SVR as they are less sensitive to the particular choice of the regression parameter.

8.5 Online Experiments

We conclude the experiments section with two experiments which use our online algorithms. Theorem 4 states that the GD online algorithm attains a cumulative log-loss which is at most twice the loss of any fixed regressor $\boldsymbol{\mu}$, up to a constant additive factor. For any finite number of online rounds T, the theorem in particular holds for $\boldsymbol{\mu} = \boldsymbol{\lambda}^\star_T$, the regressor which attains the minimal log-loss on the first T examples in the sequence. In practice, however, we have found that GD performs much better than the theoretical guarantee in Theorem 4.

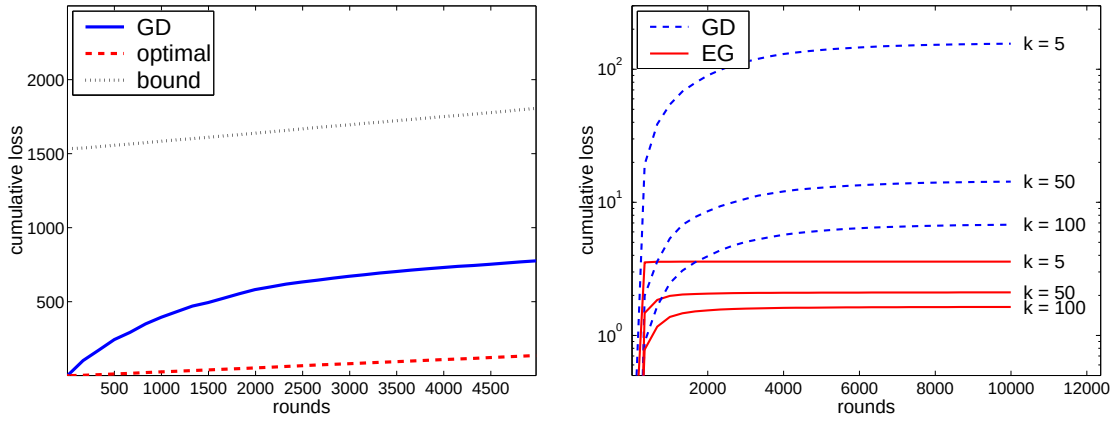

Figure 12: (Left) Cumulative loss of the GD online algorithm compared with the cumulative loss of the optimal fixed regressor and the worst case bound in Theorem 4. (Right) The cumulative loss of EG and GD for different numbers of relevant features.

To demonstrate this, we randomly generated instances and selected the target value for each instance according to a predefined linear function. We added random Gaussian noise to the target values and presented the sequence to the GD algorithm. The cumulative loss of the GD algorithm is depicted on the left hand side of Figure 12, along with the cumulative loss of $\boldsymbol{\lambda}_T^\star$ and the worst-case guarantee attained from Theorem 4 with $\boldsymbol{\mu} = \boldsymbol{\lambda}_T^\star$. Clearly, the cumulative loss of the GD algorithm lies significantly below the worst case bound. Moreover, despite the simplicity of the algorithm, its de facto performance is competitive with the best regressor on each round.

Our last experiment compares the performance of the EG and GD online algorithms. We randomly generated instances in $\{-1, 1\}^{1000}$, and generated noise-free targets according to a linear function with only k non-zero components (for $k = 5, 50, 100$). For each value of k, the first k elements of the target function were set to be $1/k$ while the rest of the elements were set to zero. The results are depicted in Figure 12 (right). The cumulative loss curves indicate that the EG algorithm is faster to converge than GD. In fact, for $k = 5$ it takes less than 10 iterations for EG to cease making any significant regression errors. Indeed, simple calculations yield that the excess loss as given in the analyses of the online algorithms is $O(\log(n/k))$ for EG while for GD it is $O(n/k)$. This type of behavior is clearly observed on the right hand side of Figure 12: the higher k becomes the smaller the difference between GD and EG.

9. Discussion

We described a framework for solving regression problems by a symmetrization of margin-based loss functions commonly used in boosting techniques. 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 take either a log-additive or an additive form. Our framework also results in a new and very simple to implement regularization scheme for regression and classification boosting algorithms. As a byproduct, we obtained a new additive algorithm for boosting-style 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 regression-tree learning algorithms. Another interesting direction is the marriage of the loss symmetrization technique with other boosting related techniques such as drifting games (Schapire, 1999; Freund and Opper, 2002).

Acknowledgments

We are in debt to Rob Schapire for making the connection to regularization and for numerous comments. We also would like to thank the anonymous reviewers for their constructive comments. This research was funded by NSF ITR Award 0205594 and by EU PASCAL Network of Excellence.

Appendix A. Technical Proofs

Proof of Lemma 5: Recall that we denote the discrepancy between the target predicted by the online algorithm and the true target by $\delta_t = \boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t$. For brevity, let $\bar{\delta}_t$ denote $\boldsymbol{\mu} \cdot \mathbf{x}_t - y_t$. Given the form of the update rule of GD, we can expand $\boldsymbol{\lambda}_{t+1}$ to get the following lower bound on the progress given in Lemma 5,

$$\begin{aligned}
& R\left(\|\boldsymbol{\lambda}_t - \boldsymbol{\mu}\|_2^2 - \|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\mu}\|_2^2\right) \\
&= R\left(2(\boldsymbol{\lambda}_t - \boldsymbol{\mu}) \cdot \frac{1}{R} L_{\log}'(\delta_t)\ \mathbf{x}_t - \left\| \frac{1}{R} L_{\log}'(\delta_t)\ \mathbf{x}_t \right\|^2\right) \\
&= 2L_{\log}'(\delta_t)(\boldsymbol{\lambda}_t - \boldsymbol{\mu}) \cdot \mathbf{x}_t - \frac{1}{R}\left(L_{\log}'(\delta_t)\right)^2 \|\mathbf{x}_t\|_2^2 \\
&\geq 2L_{\log}'(\delta_t)(\boldsymbol{\lambda}_t - \boldsymbol{\mu}) \cdot \mathbf{x}_t - \frac{1}{2}\left(L_{\log}'(\delta_t)\right)^2, \qquad (22)
\end{aligned}$$

where $L_{\log}'$ denotes the derivative of $L_{\log}$ and we used the fact that $2\|\mathbf{x}_t\|_2^2 \leq R$ in the last inequality. Since $(\boldsymbol{\lambda}_t - \boldsymbol{\mu}) \cdot \mathbf{x}_t = (\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t) + (y_t - \boldsymbol{\mu} \cdot \mathbf{x}_t) = \delta_t - \bar{\delta}_t$, we can rewrite Eq. (22) as

$$2L_{\log}'(\delta_t)(\delta_t - \bar{\delta}_t) - \frac{1}{2}(L_{\log}'(\delta_t))^2.$$

Therefore, it is sufficient to prove that the following function is non-negative

$$F(\delta, \bar{\delta}) = 2L_{\log}'(\delta)(\delta - \bar{\delta}) - \frac{1}{2}(L_{\log}'(\delta))^2 - L_{\log}(\delta) + 2L_{\log}(\bar{\delta}).$$

The partial derivative of F with respect to $\bar{\delta}$ is

$$\frac{\partial F(\delta, \bar{\delta})}{\partial \bar{\delta}} = -2L_{\log}'(\delta) + 2L_{\log}'(\bar{\delta}).$$

The only assignment of $\bar{\delta}$ for which this derivative equals 0 is $\bar{\delta} = \delta$. It is straightforward to verify that the second derivative of F with respect to $\bar{\delta}$ is positive since $L_{\log}(\cdot)$ is a convex function. Therefore, fixing δ, F attains its minimum at $\bar{\delta} = \delta$. Put another way, we have shown that for any $\delta, \bar{\delta} \in \mathbb{R}$, $F(\delta, \delta) \leq F(\delta, \bar{\delta})$. Denoting $K(\delta) = F(\delta, \delta)$ and simplifying K we get,

$$K(\delta) = -\frac{1}{2}\left(L'_{\log}(\delta)\right)^2 + L_{\log}(\delta).$$

It is left to show that $K(\delta)$ is non-negative. We prove this by showing that for all δ $K(\delta) \geq K(0) = 0$. The derivative of K with respect to δ is $\frac{dK(\delta)}{d\delta} = L'_{\log}(\delta)(1 - L''_{\log}(\delta))$, where $L''_{\log}(\delta)$ is the second derivative of $L_{\log}(\delta)$, namely,

$$\begin{aligned} L''_{\log}(\delta) &= \frac{e^{-\delta+\varepsilon}}{(1+e^{-\delta+\varepsilon})^2} + \frac{e^{\delta+\varepsilon}}{(1+e^{\delta+\varepsilon})^2} \\ &= \left(1 - \frac{1}{1+e^{-\delta+\varepsilon}}\right)\frac{1}{1+e^{-\delta+\varepsilon}} + \left(1 - \frac{1}{1+e^{\delta+\varepsilon}}\right)\frac{1}{1+e^{\delta+\varepsilon}}. \end{aligned}$$

The above equation implies that $L''_{\log}(\delta)$ is the sum of two numbers, each of which is in $[0, 1/4]$ and therefore $0 \leq L''_{\log}(\delta) \leq \frac{1}{2}$ for all $\delta \in \mathbb{R}$. Therefore, $1 - L''_{\log}(\delta) \geq 0$. We complete the proof by noticing that $L'_{\log}(\delta)$ is a monotonically increasing function and $L'_{\log}(0) = 0$. Therefore, $\delta = 0$ is the single extreme point of $K(\delta)$ and since $K(1) > K(0) = 0$ we get that for all δ $K(\delta) \geq K(0) = 0$.

■

Proof of Lemma 7: Recall that the EG update rule is

$$\lambda_{t+1,j} = \frac{\lambda_{t,j} e^{-\beta_t x_{t,j}}}{\sum_k \lambda_{t,k} e^{-\beta_t x_{t,k}}}, \tag{23}$$

where $\beta_t = L'(\boldsymbol{\lambda}_t \cdot \mathbf{x}_t - y_t)/R$ and $R \geq (\max_j x_{t,j} - \min_j x_{t,j})^2$. First note that, without loss of generality, we can assume that $\min_j x_{t,j} = 0$. This is true since we can replace each instance-target pair $(\mathbf{x}_t, y_t)$ with the pair $(\mathbf{x}_t - (\min_j x_{t,j})\ ,\ y_t - (\min_j x_{t,j}))$. Since we consider only regressors in the probability simplex this transformation does not change discrepancy values. In addition, it is simple to verify that the update given in Eq. (23) is invariant to this shifting.

We now prove the bound in the Lemma, starting with the left-hand side of Eq. (16). Using the definition of the relative entropy we get

$$D_{\mathrm{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_t) - D_{\mathrm{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_{t+1}) = -\beta \boldsymbol{\mu} \cdot \mathbf{x}_t - \log\left(\sum_{j=1}^{n} \lambda_{t,j} e^{-\beta x_{t,j}}\right).$$

We employ the inequality $\alpha^z \leq 1 - z(1-\alpha)$ which holds for every $\alpha \geq 0$ and $z \in [0,1]$. Applying this inequality with $\alpha = e^{-\beta\sqrt{R}}$ and $z = x_{t,j}/\sqrt{R}$ yields that

$$e^{-\beta x_{t,j}} \leq 1 - \frac{x_{t,j}}{\sqrt{R}}\left(1 - e^{-\beta\sqrt{R}}\right),$$

and summing over j results in the bound,

$$\sum_{j=1}^{n} \lambda_{t,j} e^{-\beta x_{t,j}} \leq 1 - \frac{\boldsymbol{\lambda}_t \cdot \mathbf{x}_t}{\sqrt{R}} \left(1 - e^{-\beta\sqrt{R}}\right).$$

Hence,

$$D_{\mathrm{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_t) - D_{\mathrm{RE}}(\boldsymbol{\mu}, \boldsymbol{\lambda}_{t+1}) \geq -\beta \boldsymbol{\mu} \cdot \mathbf{x}_t - \log\left(1 - \frac{\boldsymbol{\lambda}_t \cdot \mathbf{x}_t}{\sqrt{R}}(1 - e^{-\beta\sqrt{R}})\right).$$

Therefore, to prove Eq. (16) it suffices to show that $F(y, \boldsymbol{\lambda} \cdot \mathbf{x}, \boldsymbol{\mu} \cdot \mathbf{x}) \geq 0$ where

$$F(y, p, r) = \frac{4}{3} R \left(-\beta r - \log\left(1 - \frac{p}{\sqrt{R}}(1 - e^{-\beta\sqrt{R}})\right)\right) + \frac{4}{3} L_{\log}(r - y) - L_{\log}(p - y).$$

We now show that for any $p, r \in \mathbb{R}$ we have $F(y, p, r) \leq F(y, p, p)$. The partial derivative of F with respect to the variable r is

$$\frac{\partial F(y, p, r)}{\partial r} = \frac{4}{3}\left(-R\beta + L'_{\log}(r - y)\right) = \frac{4}{3}\left(-L'_{\log}(p - y) + L'_{\log}(r - y)\right).$$

The only assignment of r for which this derivative is equal to 0 is $r = p$. The second derivative of F with respect to r is $4/3 L''_{\log}(r - y)$ which is non-negative (see the end of the proof of Lemma 5). Hence, given p, F attains a global minimum at $r = p$. We thus have shown that for any $p, r \in \mathbb{R}$, $F(y, p, r) \leq F(y, p, p)$. $F(y, p, p)$ reduces to

$$\begin{aligned} F(y, p, p) &= \frac{4}{3}\left(-L'_{\log}(p - y)p - R\log\left(1 - \frac{p}{\sqrt{R}}(1 - e^{-L'_{\log}(p-y)/\sqrt{R}})\right)\right) \\ &+ \frac{1}{3} L_{\log}(p - y). \end{aligned}$$

It is left to show that $F(y, p, p)$ is non-negative. Let $z = p/\sqrt{R}$ and $\delta = p - y$. The definition of $\sqrt{R}$ implies that $z \in [0, 1]$. Therefore, it is sufficient to prove that the function $G(z, \delta)$ is non-negative for all $z \in [0, 1]$ and $\delta \in \mathbb{R}$ where,

$$G(z, \delta) = L_{\log}(\delta) - 4\sqrt{R}\left(z L'_{\log}(\delta) + \sqrt{R}\log\left(1 - z\left(1 - e^{-L'_{\log}(\delta)/\sqrt{R}}\right)\right)\right).$$

We now apply the inequality $\log(1 - z(1 - e^p)) \leq zp + p^2/8$, which holds for $z \in [0, 1]$ and $p \in \mathbb{R}$. We get

$$G(z, \delta) \geq L_{\log}(\delta) - \frac{1}{2}\left(L'_{\log}(\delta)\right)^2.$$

The term lower-bounding $G(z, \delta)$ is equal to $K(\delta)$ where $K(\delta)$ was defined in Lemma 5. In that lemma we proved that $K(\delta) \geq 0$. Therefore, $G(z, \delta) \geq 0$ as required. ■

References

J. Bi and K.P. Bennett. A geometric approach to support vector regression. In *Neurocomputing, special issue on support vector machines*, volume 55, pages 79–108, September 2003.

N. Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, 59(3):392–411, 1999.

M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3):253–285, 2002.

N. Duffy and D. Helmbold. Leveraging for regression. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*. ACM, 2000.

Y. Freund and M. Opper. Drifting games and Brownian motion. *Journal of Computer and System Sciences*, 64:113–132, 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, April 2000.

J.H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

P.J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

J. Kivinen and M.K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2001.

K.W. Penrose, A.G. Nelson, and A.G. Fisher. Generalized body composition prediction equation for men using simple measurement techniques. *Medicine and Science in Sports and Exercise*, 17(2):189, 1985.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), 1990.

R.E. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Machine Learning: Proceedings of the Nineteenth International Conference*, 2002.

R.E. Schapire. Drifting games. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2, 1998.

V.N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.