JMLR: Workshop and Conference Proceedings 44 (2015) 260-276 NIPS 2015

The 1st International Workshop "Feature Extraction: Modern Questions and Challenges"

Minimum Description Length (MDL) Regularization for Online Learning

Gil I. Shamir GSHAMIR@GOOGLE.COM
Google Inc.
6425 Penn Ave. Suite 700
Pittsburgh, PA 15206-4037, USA

Editor: Dmitry Storcheus

Abstract

An approach inspired by the *Minimum Description Length (MDL)* principle is proposed for adaptively selecting features during online learning based on their usefulness in improving the objective. The approach eliminates noisy or useless features from the optimization process, leading to improved loss. Several algorithmic variations on the approach are presented. They are based on using a Bayesian mixture in each of the dimensions of the feature space. By utilizing the MDL principle, the mixture reduces the dimensionality of the feature space to its subspace with the lowest loss. Bounds on the loss, derived, show that the loss for that subspace is essentially achieved. The approach can be tuned for trading off between model size and the loss incurred. Empirical results on large scale real-world systems demonstrate how it improves such tradeoffs. Huge model size reductions can be achieved with no loss in performance relative to standard techniques, while moderate loss improvements (translating to large regret improvements) are achieved with moderate size reductions. The results also demonstrate that overfitting is eliminated by this approach.

Keywords: Minimum Description Length (MDL), online learning, regularization

1. Introduction and Overview

Model selection techniques (see, e.g., Hansen and Yu, 2001; Kadane and Lazar, 2004; Kearns et al., 1995; Quinlan and Rivest, 1989; Viswanathan et al., 1999) are limited in their ability to select the best features for an online model. They are also constrained on their ability to trade off between prediction model size and its expected accuracy, when size constraints are imposed. *Regularization* L_0 or L_1, (see, e.g., Shalev-Shwartz and Tewari, 2011; Vidaurre et al., 2013; Zou and Hastie, 2005), is often employed to limit model size. Various techniques were developed for L_1 regularization in online learning (Beck and Teboulle, 2009; McMahan, 2011; Tsuruoka et al., 2009), and in batch training (Duchi and Singer, 2009; Duchi et al., 2011; Ravikumar et al., 2010) for this purpose. These rely heavily on the magnitude of the *model coefficient* at each coordinate, but not directly on how useful the coordinates are for the objective. This can lead to suboptimal loss to sparsity tradeoffs as well as either *underfitting* or *overfitting* the model to the training data.

The *Minimum Description Length (MDL)* principle, initially developed by Rissanen (1978, 1984, 1986) (see also Barron et al., 1998; Grunwald, 2004; Grunwald et al., 2005; Grunwald, 2007), measures the optimal tradeoffs between the gain to the objective from

the *Maximum Likelihood (ML)* estimate of a model *parameter* and the additional regret necessary to learn the parameter. The *benefit* of a parameter is the overall sum of both terms. A model should consist only of parameters with positive benefit. Offline model selection techniques have been based on this principle (see, e.g., Hansen and Yu, 2001; Quinlan and Rivest, 1989; Kearns et al., 1995; Viswanathan et al., 1999).

To the best of our knowledge, no prior work applied the MDL principle per-parameter to online learning with a general convex loss, as it was not understood how the concept can be properly utilized. In this paper, we solve this problem. We start with the observation that any online learning algorithm can be modified to precisely compute an MDL *benefit* for all its features. For each feature, the model is updated with and without it, and the difference between the cumulative losses is accumulated as the *benefit score* of the feature. Only features with positive or sufficiently large benefit scores are used in prediction. This realization is the basis to the novel *online MDL regularization* presented, which leads to optimal sparsity tradeoffs. It is then shown that MDL regularization can be viewed as a low-complexity approximation of a Bayesian mixture algorithm that mixes over all possible subspaces of the complete feature space. This mixture has exponential complexity in the number of features. However, a per-feature linear complexity mixture which only negligibly increases the loss can be designed. The MDL Regularization algorithm essentially performs an algorithm very close to the per-feature mixture algorithm, dropping unuseful features.

The MDL algorithm wraps over some general *base algorithm*. We focus, though, on *Follow-The-Regularized-Leader (FTRL)* base algorithms (Auer et al., 2002; Hazan, 2012; Kakade and Shalev-Shwartz, 2008; Kalai and Vempala, 2005; McMahan and Streeter, 2010; McMahan, 2011; Rakhlin et al., 2005; Shalev-Shwartz, 2007; Shalev-Shwartz and Singer, 2007; Shalev-Shwartz et al., 2010; Shalev-Shwartz, 2012), specifically on *FTRL-Proximal* versions. We also focus on linearized versions (McMahan, 2011; McMahan et al., 2013; McMahan, 2014), which are more practical to implement. The analysis relies on some results from McMahan (2014). It shows that the loss of the MDL mixture algorithms converges to that of the subspace of the feature space with the smallest loss. Dropping lowest benefit features leads to optimal accuracy with imposed size constraints.

Paper Outline: Section 2 formulates the problem. In Section 3, MDL Regularization is presented. Section 4 presents various mixture approaches. Section 5 presents loss analysis for the mixture algorithms. Section 6 ties between the MDL Regularization algorithm of Section 3 and the mixture algorithms. Finally, real-world experimental results are presented in Section 7.

2. Problem Formulation, Background, and Notation

We consider online convex optimization over a series of rounds $t \in \{1, 2, \ldots, T\}$ as in (McMahan, 2014) (see also Boyd and Vandenberghe, 2004; Rockafellar, 1997; Shalev-Shwartz, 2012, for basic tools and results). Each round, a predictor $\mathbf{x}_t \in \mathbb{R}^k$ is selected. An adversary, then, selects a convex loss function f_t. The algorithm suffers loss $f_t(\mathbf{x}_t)$. The goal is to minimize $\mathcal{L}(\mathcal{A}) \stackrel{\triangle}{=} \sum_{t=1}^{T} f_t(\mathbf{x}_t)$ for algorithm $\mathcal{A}$ over all T rounds. The *regret* relative to a some fixed comparator $\mathbf{x}^*$, selected possibly with knowledge of the losses, is

$\text{Regret}(\mathbf{x}^*) \stackrel{\triangle}{=} \sum_{t=1}^T f_t(\mathbf{x}_t) - \sum_{t=1}^T f_t(\mathbf{x}^*)$. We also define $\text{Regret}(\mathcal{X}) \stackrel{\triangle}{=} \sup_{\mathbf{x}^* \in \mathcal{X}} \text{Regret}(\mathbf{x}^*)$. We use compressed summation notation, $f_{1:t}(\mathbf{x}) \stackrel{\triangle}{=} \sum_{\tau=1}^t f_\tau(\mathbf{x})$.

It is possible that the smallest possible *loss* can be achieved by optimizing over some sparse subspace of $\mathbb{R}^k$, designated by a (feature) *selector vector* $\mathbf{s} \in \{0,1\}^k$, whose 1 coordinates specify selected features. The dimension of the subspace is $\|\mathbf{s}\|_0 \leq k$. A vector $\mathbf{x}_{\mathbf{s},t}$ is constrained to have nonzero entries only at the nonzero indices of $\mathbf{s}$. We use $\mathbf{x}_t^{(\mathbf{s})}$ to denote a vector with nonzero entries only at coordinates specified by $\mathbf{s}$ which is played only with features selected by $\mathbf{s}$, attaining loss and regret $\mathcal{L}(\mathbf{s})$ and $\text{Regret}(\mathbf{s}, \mathbf{x}^*)$, respectively. An overall algorithm attempts to drive down its loss by utilizing the possibility that $\mathcal{L}(\mathbf{s}) \leq \mathcal{L}(\mathbf{1} \stackrel{\triangle}{=} (1, 1, \ldots, 1)^T)$. The idea is to force "bad" coordinates to 0 quickly without accruing the regret that is usually accrued while learning them.

Somewhat complicated notation must be defined for various vectors to describe the algorithms. Boldface letters denote (column) vectors. An *unregularized* (by MDL) k-dimensional vector played by a *base algorithm*, which can have its own internal regularization, is denoted by $\mathbf{x}_t$. A full dimensional MDL *mixed predictor* is denoted by $\tilde{\mathbf{x}}_t$. A *regularized predictor*, which is played after dropping low-benefit features from $\tilde{\mathbf{x}}_t$, will be denoted by $\mathbf{x}_t^R$. A *base predictor*, derived with the base algorithm for the MDL regularized algorithm, will be denoted by $\bar{\mathbf{x}}_t$. The ith component of $\mathbf{x}_t$ is denoted by $x_{i,t}$. The set of all nonzero indices in $\mathbf{s}$ is denoted by $\{\mathbf{s}\} \stackrel{\triangle}{=} \{i : s_i = 1\}$, $\mathbf{x}_{\mathbf{s},t} \stackrel{\triangle}{=} \text{Diag}(\mathbf{s}) \cdot \mathbf{x}_t$ designates $\mathbf{x}_t$ with $x_{i,t} = 0,\ \forall i \notin \{\mathbf{s}\}$ (where $\text{Diag}(\mathbf{s})$ is a diagonal matrix with diagonal $\mathbf{s}$). Similarly, $\mathbf{x}_t^{(\mathbf{s})}$ designates a vector obtained when playing the online problem only on the subspace of $\mathbb{R}^k$ spanned through $\mathbf{s}$. A selector vector whose only nonzero coordinate is at i will be denoted by $\mathbf{s}_i$, and $\mathbf{s}(\mathbf{x})$ denotes $\mathbf{s}$, such that, $s_i = 1 \Leftrightarrow x_i \neq 0$, and $s_i = 0 \Leftrightarrow x_i = 0$. Negative indices will be used to denote vectors for which the entries in the respective indices were set to 0, e.g., $\mathbf{x}_{-i,t}$ and $\mathbf{x}_t^{(-i)}$ denote $\mathbf{x}_t$ which was played with all components, but its ith component was then set to 0, and the vector played on the subspace spanned without the ith coordinate, respectively. A plus sign on index i indicates a mixed or a regularized vector, that takes the ith component of $\bar{\mathbf{x}}_t$, e.g., $\mathbf{x}_{+i,t}^R \stackrel{\triangle}{=} \mathbf{x}_{-i,t}^R + \bar{x}_{i,t}\mathbf{s}_i$ and $\tilde{\mathbf{x}}_{+i,t} \stackrel{\triangle}{=} \tilde{\mathbf{x}}_{-i,t} + \bar{x}_{i,t}\mathbf{s}_i$. Similarly, $\mathbf{x}_{+i}^{(\mathbf{s})} \stackrel{\triangle}{=} \mathbf{x}^{(\mathbf{s} \cup \mathbf{s}_i)}$.

A specific problem of interest is *logistic regression*, which predicts probabilities of binary labels. It is simpler to use to demonstrate the exponential mixture algorithm. In each round, a vector $\mathbf{a}_t \in \mathbb{R}^k$ represents the feature magnitudes. The positive label probability is predicted by $p_t = \sigma(\mathbf{a}_t \cdot \mathbf{x}_t)$, where '$\cdot$' denotes inner product, and $\sigma(z) \stackrel{\triangle}{=} 1/(1 + e^{-z})$ is the *Sigmoid* function. The adversary reveals the label $y_t \in \{0, 1\}$. The loss incurred is $\ell(p_t, y_t) = -y_t \log p_t - (1 - y_t) \log(1 - p_t)$. Logistic regression is a convex optimization problem with $f_t(\mathbf{x}) = \ell(\sigma(\mathbf{a}_t \cdot \mathbf{x}), y_t)$.

We make the standard assumptions for convexity (Boyd and Vandenberghe, 2004; McMahan, 2014; Shalev-Shwartz, 2012) of the loss functions. The function $\psi : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ over domain $\{\mathbf{x} : \psi(\mathbf{x}) < \infty\}$ has the sub-differential set $\partial\psi(\mathbf{x})$. A sub-gradient $\mathbf{g} \in \partial\psi(\mathbf{x})$ satisfies $\psi(\mathbf{y}) \geq \psi(\mathbf{x}) + \mathbf{g} \cdot (\mathbf{y} - \mathbf{x}); \forall \mathbf{y}$. A function $\psi(\mathbf{x})$ is *σ-strongly convex* w.r.t. a norm $\|\cdot\|$ on $\mathcal{X}$ if $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$ and any $\mathbf{g} \in \partial\psi(\mathbf{x})$, $\psi(\mathbf{y}) \geq \psi(\mathbf{x}) + \mathbf{g} \cdot (\mathbf{y} - \mathbf{x}) + \frac{\sigma}{2} \|\mathbf{y} - \mathbf{x}\|^2$.

We focus on the FTRL algorithms (see, e.g., McMahan, 2011; McMahan et al., 2013; McMahan, 2014)), specifically *Proximal* variants. Let $r_t(\mathbf{x}), t \in \{0, 1, \ldots, T\}$ denote a

strongly convex incremental regularizer $\forall t$, which also leads to a strongly convex composite objective $h_{0:t}(\mathbf{x}) \stackrel{\triangle}{=} f_{1:t}(\mathbf{x}) + r_{0:t}(\mathbf{x})$ (see Bartlett et al., 2007). Then, the algorithm plays $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} f_{1:t}(\mathbf{x}) + r_{0:t}(\mathbf{x})$. For FTRL-Proximal, the incremental regularizer r_t is globally minimized by $\mathbf{x}_t$.

Of specific practical interest are linearized algorithms, that bound the convex loss at any t by a linear function for convex differentiable functions f_t. These are particularly appealing because they are usually more practical to implement. Let $\mathbf{g}_t = \bigtriangledown f_t(\mathbf{x}_t)$, then, as observed in (Zinkevich, 2003), by convexity, for any comparator $\mathbf{x}^*$, $f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*) \leq \mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}^*)$. This implies that if $f_t(\mathbf{x})$ is replaced by $\bar{f}_t(\mathbf{x}) = \mathbf{g}_t \cdot \mathbf{x}_t$, the regret against $\bar{f}_t$ upper bounds that against f_t.

3. The MDL Regularization Algorithm

The extended version of this paper (Shamir, 2015) presents a more general algorithm. Here, we focus on the linearized algorithm, with a base FTRL (and later FTRL-Proximal) algorithm. We restrict the regularizer to regularizers for which minimizing per coordinate yields the global minimum, e.g., $\arg\min_{\mathbf{x}} r_{0:t}(\mathbf{x})|_i = \arg\min_{x_i} r_{0:t}(\mathbf{x}), \forall t$. General L_2 norm regularizers $r_{0:t}(\mathbf{x}) = \frac{1}{2\eta_t} \|\mathbf{x}\|_2^2$, where η_t is the algorithm's learning rate at round t, satisfy this condition. For FTRL-Proximal, we use $r_0(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{X}$ (∞ outside $\mathcal{X}$), and for all $t > 0$, $r_t(\mathbf{x}) = \frac{\varphi_t}{2} \|\mathbf{x} - \mathbf{x}_t\|_2^2, \eta_t \stackrel{\triangle}{=} \frac{1}{\varphi_{1:t}}$. With this r_t, as shown in (McMahan, 2011, 2014), the algorithm is a *Stochastic Gradient Descent (SGD)* algorithm with learning rate η_t (see, e.g., Bottou and Bousquet, 2008; Bottou, 2010; Shamir and Zhang, 2012, for some recent results on SGD). By adding $\lambda_2 \|\mathbf{x}_t\|_2^2$ to $r_t(\mathbf{x})$, we add L_2 regularization, and by adding $\lambda_1 \|\mathbf{x}_t\|_1$, L_1 regularization is added.

Algorithm 1 shown below is the *Linearized MDL Regularization* algorithm. It takes an inner regularization schema as a parameter, and four more parameters: μ - the threshold on the minimum benefit score a feature should have in order to be included in prediction; γ - a cap on the minimum benefit (if no such limit is desired, $\gamma = -\infty$); ρ - a scale factor applied to the benefit score to derive $\tilde{\mathbf{x}}_t$ from $\bar{\mathbf{x}}_t$; ξ - a bias (margin) added to the scaled benefit prior to using it as a mixing argument. Any of these can be configured globally or per-coordinate. We store a base predictor $\bar{\mathbf{x}}_t$ and a cumulative benefit score vector $\mathbf{B}_t$. Mixed and regularized vectors, $\tilde{\mathbf{x}}_t$ and $\mathbf{x}_t^R$, are derived from $\bar{\mathbf{x}}_t$ and $\mathbf{B}_t$. After playing $\mathbf{x}_t^R$ at t, the algorithm updates all coordinates of benefit scores $\mathbf{B}_{t+1}$ and base predictors $\bar{\mathbf{x}}_{t+1}$. It derives mixed predictors for all i with $B_{i,t+1} > \mu$ by mixing $\bar{x}_{i,t+1}$ with 0, using weights which are functions of $B_{i,t+1}$, and regularizes any coordinate with $B_{i,t+1} \leq \mu$ to 0. The benefit score of a coordinate accumulates the difference in losses with and without the coordinate.

Steps 6-7 compute a sub-gradient for each coordinate i (whether $x_{i,t}^R = 0$ or not) by taking $\mathbf{x}_t^R$, replacing the ith entry by $\bar{x}_{i,t}$, and computing the ith coordinate of the sub-gradient of the loss f_t at this constructed predictor. Then, the respective coordinates of the sub-gradients computed are assembled into a single vector $\bar{\mathbf{g}}_t$ (Step 7). In other words, the gradient of a base value of a coordinate is computed in the presence of the values of all other coordinates used in the actual prediction played by the algorithm. Results in Section 5 show why the update should be done in this manner. The benefit score is then updated in Step 8, using the same $\mathbf{x}_{+i,t}^R$ used to compute the gradient to compute the loss

Algorithm 1 Linearized FTRL MDL Regularization

1: **procedure** Linearized MDL Regularize(Parameters: r_0, schema for $r_{t;t>0}$, μ, γ, ρ, ξ)
2: $\bar{\mathbf{x}}_1, \tilde{\mathbf{x}}_1, \mathbf{x}_1^R \leftarrow \mathbf{0}$ ▷ Coefficients' initialization
3: $\mathbf{B}_1 \leftarrow \mathbf{0}$ ▷ Benefit initialization
4: **for** t=1,2,... **do**
5: Play $\mathbf{x}_t^R$, observe f_t, incur loss $f_t(\mathbf{x}_t^R)$
6: Compute sub-gradient $\bar{\mathbf{g}}_t^{(i)} \in \partial f_t\left(\mathbf{x}_{+i,t}^R\right), \forall i.$ ▷ Distinct gradient vector per coordinate
7: Assemble $\bar{\mathbf{g}}_t \stackrel{\triangle}{=} \left\{\bar{g}_{i,t}^{(i)}; i = 1, \ldots, k\right\}$ ▷ Gradient vector assembled
8: $B_{i,t+1} \leftarrow B_{i,t} - \left[f_t\left(\mathbf{x}_{+i,t}^R\right) - f_t\left(\mathbf{x}_{-i,t}^R\right)\right], \forall i$ ▷ Update benefit
9: If $B_{i,t+1} < \gamma$, $B_{i,t+1} \leftarrow \gamma$, $\forall i$ ▷ Set benefits to minimum limit
10: $\mathbf{s}_{t+1} \leftarrow \mathbf{s}(\mathbf{B}_{t+1} > \mu)$ ▷ Selector for (new) nonzero coordinates
11: Choose r_t, possibly using $\bar{\mathbf{x}}_t$ and $\bar{\mathbf{g}}_t$
12: $\bar{\mathbf{x}}_{t+1} \leftarrow \arg\min_{\bar{\mathbf{x}}} \left[\bar{\mathbf{g}}_{1:t} \cdot \bar{\mathbf{x}} + r_{0:t}\left(\bar{\mathbf{x}}\right)\right]$ ▷ Update base values
13: $\tilde{x}_{i,t+1} \leftarrow \sigma\left(\rho B_{i,t+1} + \xi\right) \cdot \bar{x}_{i,t+1}, \forall i \in \{\mathbf{s}_{t+1}\}$ ▷ Update mixed values
14: $x_{i,t+1}^R \leftarrow \begin{cases} \tilde{x}_{i,t+1}, & i \in \{\mathbf{s}_{t+1}\} \\ 0, & \text{otherwise} \end{cases}, \forall i$ ▷ Update regularized values
15: **end for**
16: **end procedure**

with the coordinate. The ith coordinate is removed for the loss without it. The actual f_t can be replaced in this step by the linearized objective $\bar{f}_t$. After thresholding the benefit scores, the set of (more beneficial) coordinates may have changed. It is updated in Step 10. An inner regularizer r_t for the next round can be chosen in Step 11, as function of $\bar{\mathbf{x}}_t$.

The base predictor $\bar{\mathbf{x}}_{t+1}$ is computed in Step 12 using linearization with the accumulated gradient $\bar{\mathbf{g}}_{1:t}$, updated in Step 7, adding the regularizer $r_t(\bar{\mathbf{x}})$ chosen in Step 11. For FTRL-Proximal with restricted regularizer w.r.t. $\bar{\mathbf{x}}$, Step 12 becomes as follows: $\bar{x}_{i,t+1} = \arg\min_{\bar{x}_i}\left\{\bar{g}_{i,1:t}\bar{x}_i + \sum_{\tau=1}^t \frac{\varphi_\tau}{2}\left(\bar{x}_i - \bar{x}_{i,\tau}\right)^2\right\}, \forall i$. If in addition, L_2 regularization is performed, an L_2 term of $\lambda_2 \bar{x}_i^2$ is added to the minimized objective, yielding $\bar{x}_{i,t+1} = \bar{x}_{i,t} - \frac{\eta_t \bar{g}_t}{1+2\lambda_2\eta_t}, \forall i$ ($\lambda_2 = 0$ with no L_2).

Next, the coordinates in $\{\mathbf{s}_{t+1}\}$ are mixed with 0 in Step 13 using the Sigmoid function. The argument of the Sigmoid, and thus $B_{i,t+1}$ as well, are regarded as logarithms of odds (*log-odds*). This is justified by viewing $\exp\{-\sum_\tau f_\tau(\mathbf{x}_{+i,\tau}^R)\}$ as a likelihood that the coefficient is nonzero. Similarly, $\exp\{-\sum_\tau f_\tau(\mathbf{x}_{-i,\tau}^R)\}$ is viewed as the likelihood that the coefficient is zero. Then, ξ is a prior for the binary distribution determining whether the coefficient is 0 or not. Thus the weight given to the nonzero value of the coefficient is essentially a *posterior* probability of the coefficient taking a nonzero value. A round is concluded with Step 14, which sets $\mathbf{x}_{t+1}^R$, performing the regularization, resetting coordinates with low benefit.

4. MDL Regularization as a Bayesian Mixture

MDL regularization is presented as an exponential Bayesian mixture over all feature subspaces. It is simplified to a per-feature linear mixture, concluding with a version that drops unuseful features.

4.1 Complete MDL Bayesian Mixture

The *Complete MDL Mixture* algorithm is presented below for logistic regression (for simplicity). Let $w_t^{(\mathbf{s})}$ denote the weight assigned to $p_t^{(\mathbf{s})} = \sigma(\mathbf{a}_t \cdot \mathbf{x}_t^{(\mathbf{s})})$ derived and played for $\mathbf{s}$ at t. The algorithm assigns a probability to label sequence $y_1^T \stackrel{\triangle}{=} (y_1, \ldots, y_T)$ for each $\mathbf{s}$. A final probability is then a mixture of these over all $\mathbf{s}$. The algorithm is an *ensemble* method (see, e.g., Dietterich, 2000) over the (exponential) ensemble of all $\mathbf{s}$. Let Λ_t denote the complete mixture probability of y_1^t, which is a normalizer for the probability at t. Algorithm 2 is shown below. It takes r_t and a single parameter α designating the prior probability of $x_i \neq 0$ (global or per-feature). It maintains a predictor $\mathbf{x}_t^{(\mathbf{s})}$ and a weight $w_t^{(\mathbf{s})}$ for each of the 2^k values of $\mathbf{s}$, as well as the normalizing sum Λ_t. Step 10, specified in a general manner, can be linearized as in Algorithm 1. Steps 12 and 13 update the weights for each $\mathbf{s}$ based on its predictions. Better $\mathbf{s}$ subsets will have larger portions in the mixture. In (Shamir, 2015), it is also shown how this algorithm can be applied to general losses.

Algorithm 2 Complete MDL Mixture for Logistic Regression

1: **procedure** Complete MDL Mix(Parameters: r_0, schema for $r_{t;t>0}$, α)
2: $\quad \mathbf{x}_1^{(\mathbf{s})} \leftarrow \mathbf{0},\ w_1^{(\mathbf{s})} = \alpha^{\|\mathbf{s}\|_0}(1-\alpha)^{k-\|\mathbf{s}\|_0},\ \forall \mathbf{s}$
3: $\quad \Lambda_1 \leftarrow 1$ ▷ Initialize normalizing factor
4: $\quad$ **for** t=1,2,... **do**
5: $\quad\quad$ Observe $\mathbf{a}_t$
6: $\quad\quad p_t^{(\mathbf{s})} \leftarrow \sigma(\mathbf{a}_t \cdot \mathbf{x}_t^{(\mathbf{s})})$
7: $\quad\quad p_t \leftarrow \sum_{\mathbf{s}} w_t^{(\mathbf{s})} p_t^{(\mathbf{s})}$
8: $\quad\quad$ Play p_t, observe $y_t \in \{0,1\}$, incur loss $f_t(p_t, y_t) = -y_t \log p_t - (1-y_t)\log(1-p_t)$
9: $\quad\quad$ Choose r_t, possibly using p_t
10: $\quad\quad \mathbf{x}_{t+1}^{(\mathbf{s})} \leftarrow \arg\min_{\mathbf{x}^{(\mathbf{s})}} \left[f_{1:t}(\mathbf{x}^{(\mathbf{s})}) + r_{0:t}(\mathbf{x}^{(\mathbf{s})}) \right],\ \forall \mathbf{s}$
11: $\quad\quad p_t^{(\mathbf{s})}(y_t) \leftarrow (p_t^{(\mathbf{s})})^{y_t} \cdot (1-p_t^{(\mathbf{s})})^{1-y_t},\ \forall \mathbf{s}$
12: $\quad\quad \Lambda_{t+1} \leftarrow \Lambda_t \cdot \sum_{\mathbf{s}} w_t^{(\mathbf{s})} \cdot p_t^{(\mathbf{s})}(y_t)$ ▷ Update normalizing factor
13: $\quad\quad w_{t+1}^{(\mathbf{s})} = \frac{\Lambda_t}{\Lambda_{t+1}} \cdot w_t^{(\mathbf{s})} \cdot p_t^{(\mathbf{s})}(y_t),\ \forall \mathbf{s}$ ▷ Update weights
14: $\quad$ **end for**
15: **end procedure**

4.2 Feature Level MDL Bayesian Mixture

The *Linearized MDL Mixture* algorithm, Algorithm 3, is presented below. Instead of the exponential mixture, it mixes per-coordinate with linear complexity in k. While very similar to Algorithm 1, it differs in playing all coordinates of $\tilde{\mathbf{x}}_t$ instead of the subset $\mathbf{x}_t^R$. The parameter ξ is replaced by the prior α, there is no minimum benefit threshold, and there

is no need for ρ. Mixing without forcing coordinates to 0 simplifies the analysis. The main result in the paper is derived for this algorithm, and drives the behavior of the other algorithms.

Algorithm 3 Linearized MDL Mixture

1: **procedure** Linearized MDL Mix(Parameters: r_0, schema for $r_{t;t>0}$, γ, α)
2: Play Algorithm 1 with $\rho = 1$, and $\xi = \log(\alpha) - \log(1-\alpha)$, replacing $\mathbf{x}_t^R$ by $\tilde{\mathbf{x}}_t$, removing Steps 10 and 14, and performing Step 13 for all i.
3: **end procedure**

4.3 Regularized MDL Mixture

For practical reasons, we may want to use only beneficial features. The *Linearized Regularized MDL Mixture* algorithm, Algorithm 4, below, limits prediction, as in Algorithm 3, only to features with $B_{i,t} > \mu$. Unlike Algorithm 1, Algorithm 4 still fully trains on all coordinates.

Algorithm 4 Linearized Regularized MDL Mixture

1: **procedure** Linearized Regularized MDL Mix(Parameters: r_0, schema for $r_{t;t>0}$, μ, γ, α)
2: Play Algorithm 1 with $\rho = 1$, and $\xi = \log(\alpha) - \log(1-\alpha)$, replacing $\mathbf{x}_t^R$ by $\tilde{\mathbf{x}}_t$ in Steps 6 and 8 but not in Step 5, and performing Step 13 for all i.
3: **end procedure**

5. Analysis and Loss Bounds for Mixture Algorithms

Loss bounds for the mixture algorithms are presented. More details are in (Shamir, 2015).

5.1 Loss for Complete MDL Mixture

The Complete MDL Mixture (CMM) algorithm (Algorithm 2), that plays $\mathbf{x}_t^{(\mathbf{s})}$ at each t for each $\mathbf{s}$, attains loss $\mathcal{L}\,(\text{CMM}) = -\log \sum_{\mathbf{s}} w_1^{(\mathbf{s})} \cdot \exp\left\{-\mathcal{L}\left(\{\mathbf{x}_t^{(\mathbf{s})}\}\right)\right\}$ where $\{\mathbf{x}_t^{(\mathbf{s})}\}$ is the sequence of predictors played at all rounds for $\mathbf{s}$, and $w_1^{(\mathbf{s})}$ is the prior assigned to $\mathbf{s}$. For simplicity, as formulated in Section 2, let $\mathcal{L}(\mathbf{s}) \stackrel{\triangle}{=} \mathcal{L}\left(\{\mathbf{x}_t^{(\mathbf{s})}\}\right)$ and $\text{Regret}(\mathbf{s}) \stackrel{\triangle}{=} \sup_{\mathbf{x}^{(\mathbf{s})}} \text{Regret}(\mathbf{x}^{(\mathbf{s})})$.

Proposition 1 *Define* $\mathbf{s}^* \stackrel{\triangle}{=} \arg\min_{\mathbf{s}} \mathcal{L}(\mathbf{s})$, *and* $m^* \stackrel{\triangle}{=} \|\mathbf{s}\|_0$. *Then, for Algorithm 2,*

$$\mathcal{L}\,(\textit{CMM}) \leq \mathcal{L}(\mathbf{s}^*) - m^* \log \alpha - (k - m^*) \log(1-\alpha). \tag{1}$$

Proposition 1 shows that with the complete (ensemble) mixture the loss of the best feature subspace is achieved with negligible penalty determined by the prior of $\mathbf{s}^*$. Unless all

features benefit the best loss, such loss is not achievable with a standard algorithm that pays a cost for learning all unbeneficial parameters. Regret bounds, as in (McMahan, 2014; Shalev-Shwartz, 2012), are expressed by bounds R and G on the L_2 norms of $\mathbf{x}$ and its sub-gradients $\mathbf{g}$, respectively, under the implicit assumption that R and G are fixed. The dimension k, however, is generally implicitly embedded in these bounds. Hence, for a subspace $\mathbf{s}^*$ of the full space, these bounds are essentially smaller. For example, the regret of the FTRL-Proximal is $O(RG\sqrt{T})$ (see, McMahan, 2014), which is at least $O(k\sqrt{T})$, when accounting for the dimension. With MDL, the regret is $O(m^*\sqrt{T}) + O(k)$, which is dominated by that of FTRL if $m^* \ll k$ (this is usually the case in many practical systems). Tuning α to $\alpha^* = m^*/k$ achieves the minimum loss in (1). While not always possible, in practice, we usually have some advance knowledge to approximate α^* from past data.

5.2 Loss for MDL Mixture

The main theorem, stated below, shows that the MDL Mixture (MM) algorithm (Algorithm 3) achieves loss only negligibly larger than Algorithm 2. This only requires a sufficient growth rate for the benefit scores of coordinates. Due to lack of space, the condition required is described briefly in Appendix A (and in more detail in Shamir, 2015).

Theorem 2 *Define $\mathbf{s}^*$ and m^* as in Proposition 1. Assume that benefit scores have sufficient growth rates (as in condition (4) in Appendix A). Then, for Algorithm 3 with $\gamma \to -\infty$,*

$$\mathcal{L}\left(MM\right) \leq \mathcal{L}(\mathbf{s}^*) + \left\{ C_1 \cdot \frac{\alpha}{1-\alpha} \cdot m^* + C_2 \cdot \frac{1-\alpha}{\alpha} \cdot (k - m^*) \right\} \cdot o\left\{Regret(\mathbf{s}^*)\right\}. \qquad (2)$$

for some constants C_1 and C_2.

Appendix A shows a sketch of the proof of Theorem 2, which is presented in (Shamir, 2015). By Theorem 2, the results described following Proposition 1 apply to Algorithm 3. The negligible mixture cost is usually larger than that achieved in Algorithm 2. If α is tuned as m^*/k or in proximity to that, the loss of Algorithm 3 is always somewhat greater than that of Algorithm 2.

5.3 Loss for Regularized MDL Mixture

Resulting from the analysis of Algorithm 3, the loss of Algorithm 4, the Linearized Regularized MDL Mixture (RMM) algorithm, can be bounded.

Corollary 3 *Let $\mathbf{s}^*$ and m^* be as defined in Proposition 1. Then, under the conditions of Theorem 2, for Algorithm 4,*

$$\mathcal{L}\left(RMM\right) \leq \mathcal{L}(\mathbf{s}^*) + \left\{ C_1 \cdot \frac{\alpha}{1-\alpha} \cdot m^* + C_2 \cdot \frac{1-\alpha}{\alpha} \cdot (k - m^*) + (\mu + C_3) \cdot m^* \right\} \cdot o\left\{Regret(\mathbf{s}^*)\right\}. \qquad (3)$$

for some constants C_1, C_2, and C_3.

Corollary 3 follows directly from Theorem 2. Features with $B_{i,t} \leq \mu$ are excluded from prediction until their benefits exceed μ. They incur additional loss μ beyond Algorithm 3 with an additional constant loss incurred in last rounds before they enter into the model. Algorithm 4 provides a tradeoff between model size and its accuracy. The threshold μ exchanges model size for accuracy.

6. MDL Regularization as Approximation of Regularized MDL Mixture

Algorithm 1 and Algorithm 4 both use the benefit score threshold to determine which features to include in prediction. However, Algorithm 1 trains jointly only features for which $B_{i,t} > \mu$, whereas Algorithm 4 trains all features jointly. This does not significantly affect coordinates with $B_{i,t} > \mu$, but it makes a difference on how features that are excluded from prediction train. In Algorithm 4, they train jointly with other features currently excluded, whereas in Algorithm 1, each low benefit feature trains in separation from other low benefit features. This makes a difference when there is correlation between low benefit features. Features that newly enter the prediction model in Algorithm 4, trained in tandem with other such features. Their coefficients are already adjusted to one another. On the other hand, learning correlated features results in smaller coefficients for each of the features, and in multiple training costs. In Algorithm 1, multiple training costs are avoided, and some unnecessary features correlated to ones already in the model may be kept out of the model, but there is risk of multiple features explaining the same effect entering the model simultaneously. To mitigate this problem, the parameter ρ scales down the benefit of coordinates newly used for prediction. It essentially splits the benefit in a group of correlated features among the members of the group. The parameter $\rho \in (0, 1]$ should thus be tuned based on a number of correlated features expected to enter the model simultaneously. (This can be approximated empirically.) Algorithm 4 thus has no need for ρ. In the real-world systems, where the two algorithms were tested, they seem to perform comparably, with slight advantage to Algorithm 1.

The parameter μ measures the tolerance to loss. It trades off between prediction model size and model prediction accuracy. The parameter ξ in Algorithm 1 determines the prior similarly to α in Algorithm 4. The optimal value shown in Theorem 2 should be used for determining ξ by the relation $\xi = \log(\alpha/(1-\alpha))$. If we have some prior knowledge of what fraction of the features become useful for prediction, ξ can be tuned by taking the values as shown following Theorem 2. Finally, the parameter γ has two different roles. First, it allows for nonstationarity. Second, in cases that only the loss on *test* is important, γ allows reducing the penalty imposed on features with past low-benefit, that are becoming more beneficial. While this increases prediction model size, experimental results demonstrate that using γ can improve performance measured in progressive validation on the latest batch of the examples.

7. Experimental Results

We report results with Algorithm 1 on real-world data, specifically, on large private data sets for ad Click-Through-Rate (CTR) prediction for sponsored advertisements. Hundreds of billions examples are visited, and the model feature sets consist of hundreds of millions to billions nonzero coefficients. As the standard practice in this domain (McMahan et al., 2013), progressive validation methodology (Blum et al., 1999) is used to assess performance on held-out data in an online setting. Results are reported in terms of AUC loss $(1 - \text{AUC})$, averaged on billions of the final examples seen by the model.

Figure 1 shows the trade off between model size and its AUC loss for three algorithms: an L_1 regularized FTRL-Proximal algorithm with no MDL regularization with a fixed L_1

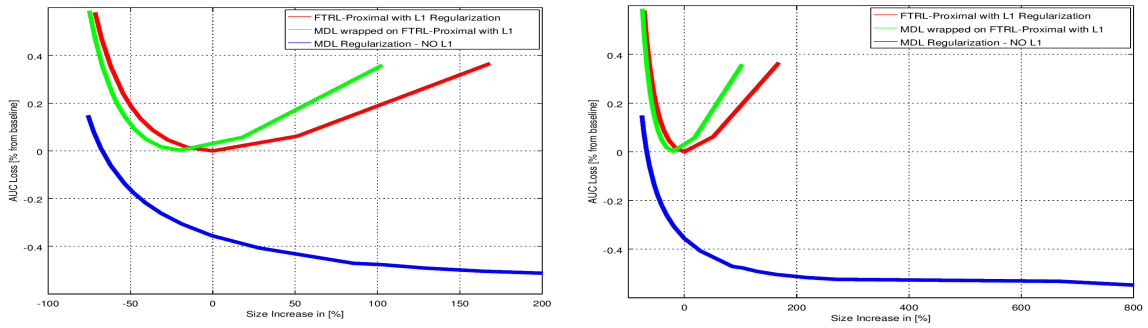

Figure 1: Experimental results on real-world AUC loss % change from baseline vs. model size % change from baseline. Full range shown on the right, and normal operating range on the left.

threshold, an L_1 regularized FTRL-Proximal algorithm wrapped with MDL regularization running Algorithm 1, and an FTRL-Proximal algorithm with no L_1 regularization, wrapped by Algorithm 1. All algorithms share all common parameters, such as learning rate, and L_2 regularization control. Both L_1 regularized algorithms are tested varying only the amount of L_1 regularization between the different points on the graph. The same grid of L_1 values is used for both algorithms. When L_1 is wrapped with the MDL Regularization, the MDL threshold μ is fixed to 0. For the MDL Regularization algorithm with no L_1 regularization, the only parameter varied is μ.

The x-axis in Figure 1 is the percent increase (decrease if negative) in size (nonzero coefficients' count) relative to a base point. The y-axis shows the percent increase (decrease if negative) in AUC-loss relative to the base point. The better the algorithm the more to the left or bottom of the graph it is. The base $(0, 0)$ reference is where the base L_1 regularized algorithm has its minimal AUC loss. The L_1 regularized algorithm exhibits a classic regularization curve. With larger L_1 regularization, it is smaller but less accurate. Decreasing the L_1 parameter initially improves accuracy with larger nonzero count. The curve reaches the optimal (base) point. However, then, the model starts overfitting. Decreasing the L_1 parameter farther still increases nonzero count, but degrades accuracy instead of improving it. Identical behavior is exhibited even when the L_1 threshold is allowed to increase with the number of examples seen by the feature. The AUC does not improve, but there is some size reduction of about $5 - 10\%$. Wrapping MDL regularization over the L_1 regularized algorithm shifts the whole curve left, decreasing model size, but with no change to accuracy. The base optimal point is shifted left by about 20%. Model size decrease is more moderate in the *underfitting* region to the left of the base point, while it is more extreme on the overfitting region, to its right, and is as large as 60% at the most extreme point tested.

Algorithm 1 with no L_1 regularization exhibits better behavior than the other algorithms. With equal model size, better accuracy is achieved, and for neutral accuracy smaller models are necessary. For accuracy neutrality with the optimal point of the L_1 algorithm, model, over 60% smaller, is sufficient. For equal model size, a moderate over 0.5% AUC loss improvement is exhibited. This improvement translates to substantial improvement on regret (which is the loss component the algorithm improves). Similar qualitative results were obtained for various systems the algorithm was tested on. Precise improvements, though,

vary among different systems. Even more interesting is the behavior of MDL closer to $\mu = 0$. Unlike L_1 regularization, the model keeps improving, exhibiting *no evidence of overfitting*. This is expected as discribed in Grunwald (2004), yet untypical with standard regularizations. It implies that tuning μ is not as crucial as tuning the L_1 parameter. Setting μ too low still improves accuracy, and does not degrade the model, unlike L_1.

The experimental results shown for the MDL Regularization algorithm can be qualitatively compared to Corollary 3. The far right end of the curve with $\mu = 0$ (with a prediction model about 650% larger than the reference base point in this case) is the one corresponding to the bound on the loss of Theorem 2. Take the loss for $\mu = 0$ as the reference loss $\mathcal{L}(\mu = 0)$ (with some abuse of notation). Then, for each point obtained for some other μ, $\mathcal{L}(\mu) = \mathcal{L}(0) + \kappa\mu$, for some κ, fixed for all μ. Then, there is a value of κ, for which the resulting curve overlaps the experimental one. This supports Theorem 2, Corollary 3, and the statement that Algorithm 1 approximates Algorithm 4.

8. Conclusions

A new paradigm, which uses the established MDL principle for online convex optimization in online learning algorithms was proposed. The paradigm uses the MDL principle to select features while in the process of online learning, based on their contributions to improving on the objective. Starting from an ensemble Bayesian mixture over all possible combinations of features out of a complete set, it was demonstrated that the algorithm can be simplified to a linear complexity (in time and number of active features) feasible algorithm that achieves the same performance accuracy as the complete exponential (in number of states) mixture algorithm. Several variations of the mixture and then of regularization algorithms, which eliminate useless features from the prediction model, reducing the actual storage/time costs of predicting with the algorithm, were described. Theoretical guarantees were then given. These show that MDL regularization can achieve the losses achieved by the best feature subspace of the feature space to first and second order (where second order is the order of the regret). Experimental results on real-world large scale systems demonstrated the advantage of the MDL based algorithms over classical feature reducing regularization techniques. Huge savings in prediction model size (of over 50% for neutral accuracy) and also accuracy savings (for neutral or smaller models) were demonstrated. In addition, the algorithm behavior showed no evidence of overfitting, as we allow more features (that are less beneficial, but still have some benefit) into the prediction model.

Acknowledgments

The author gratefully acknowledges G. Kochanski, G. Holt, A. Smola, and J. Dillon for numerous helpful discussions. Many helpful discussions with a long list of people also contributed to this work. The list includes: Y. Singer, V. Chaudhary, M. Young, J. Grady, D. Ebner, N. Shar, A. Afkanpour, D. Golovin, H. B. McMahan, A. Amini, J. F. Crespo, S. Gopal, D. Sculley, A. Ahmed, S. Finney, T. Tomokiyo, S. Siddiqi, Y. Wang, P. Castro, and C. O. Conaire.

References

H. Akaike. Information theory and an extension of the maximum likelihood principle. In *2nd International Symposium on Information Theory*, pages 267–281, 1971.

P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 2002.

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. Inf. Theory*, 44(6):2743–2760, Oct. 1998.

P. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *NIPS*, 2007.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2:183–202, 2009.

A. Blum, A. Kalai, and J. Langford. Beating the hold-out: bounds for k-fold and progressive cross-validation. In *COLT*, 1999.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

L. D. Davisson. Universal noiseless coding. *IEEE Trans. Inf. Theory*, IT-19(6):783–795, Nov. 1973.

T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, pages 1–15, 2000.

J. Duchi and Y. Singer. Efficient learning using forward-backward splitting. In *Advances in Neural Information Processing Systems*, volume 22, pages 495–503, 2009.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, Feb. 2011.

C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, pages 636–638, 2015.

M. Feder and N. Merhav. Hierarchical universal coding. *IEEE Trans. Inf. Theory*, 42(5): 1354–1364, Sep. 1996.

P. D. Grunwald. A tutorial introduction to the minimum description length principle. *CoRR*, abs/0406077, 2004. URL `http://arxiv.org/abs/math/0406077`.

P. D. Grunwald. *The Minimum Description Length Principle*. MIT Press, 2007.

P. D. Grunwald and J. Langford. Suboptimal behaviour of Bayes and MDL in classification under misspecification. In *COLT*, 2004.

P. D. Grunwald, I. J. Myung, and M. A. Pitt. *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

L. Györfi, I. Páli, and E. C. van der Meulen. There is no universal code for an infinite source alphabet. *IEEE Trans. Inf. Theory*, 40(1):267–271, Jan. 1994.

M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, pages 746–774, 2001.

T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer, 2009.

E. Hazan. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *COLT*, 2008.

E. Hazan. The convex optimization approach to regret minimization. In *S. Sra, S. Nowozin, and S. Wright, editors, Optimization for Machine Learning*, pages 287–303. MIT press, 2012.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Mach. Learn.*, 69:169–192, 2007.

J. B. Kadane and N. A. Lazar. Methods and criteria for model selection. *J. Amer. Statist. Soc.*, 99:279–290, 2004.

S. Kakade and S. Shalev-Shwartz. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, 2008.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and Systems Sciences*, 71(3), 2005.

M. J. Kearns, Y. Mansour, A. Y. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. In *COLT*, pages 21–30, 1995.

J. C. Kieffer. A unified approach to weak universal source coding. *IEEE Trans. Inform. Theory*, IT-24(6):674–682, Nov. 1978.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

N. Littlestone, P. M. Long, and M. K. Warmuth. On-line learning of linear functions. In *The 23rd Symposium on Theory of Computing*, pages 465–475, New York, NY, 1991. ACM Press.

H. B. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization. In *AISTATS*, 2011.

H. B. McMahan. Analysis techniques for adaptive online learning. *CoRR*, abs/1403.3465, 2014. URL `http://arxiv.org/abs/1403.3465`.

H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, 2010.

H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. Mar Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: A view from the trenches. In *KDD*, 2013.

T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, pages 227–248, 1989.

A. Rakhlin, S. Mukherjee, and T. Poggio. Stability results in learning theory. *Analysis and Applications*, 2005.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using L1-regularized logistic regression. *Annals of Statistics*, 2010.

J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. Inform. Theory*, IT-30(4):629–636, Jul. 1984.

J. Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14:1080–1100, 1986.

R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1997.

B. Ryabko. Twice-universal coding. *Problems of Information Transmission*, 20(3):173–177, 1984.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University of Jerusalem, 2007.

S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 2012.

S. Shalev-Shwartz and Y. Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 2007.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *JMLR*, 12:1865–1892, 2011.

S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Learnability, stability and uniform convergence. *JMLR*, 2010.

G. I. Shamir. On the MDL principle for i.i.d. sources with large alphabets. *IEEE Trans. Inform. Theory*, 52(5):1939–1955, May 2006.

G. I. Shamir. Minimum description length (MDL) regularization for online learning. in preparations, 2015.

O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *CoRR*, abs/1212.1824, 2012. URL `http://arxiv.org/abs/1212.1824`.

Y. M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23(3):3–17, Jul.-Sep. 1987.

M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society*, 39(1):44–47, 1977.

M. Streeter and H. B. McMahan. Less regret via online conditioning. *CoRR*, abs/002.4862, 2010. URL `http://arxiv.org/abs/1002.4862`.

Y. Tsuruoka, J. Tsujii, and S. Ananiadou. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 477–485, 2009.

D. Vidaurre, C. Bielza, and P. Larranaga. A survey of L1 regression. *International Statistical Review*, 81(3):361–387, 2013.

M. Viswanathan, C. S. Wallace, D. L. Dowe, and K. B. Korb. Finding outpoints in noisy binary sequences - A revised empirical evaluation. *Advanced Topics in Artificial Intelligence*, pages 405–416, 1999.

F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Trans. Inf. Theory*, 41(3):653–664, May 1995.

L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *NIPS*, 2009.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67:301–320, 2005.

Appendix A - Proof Sketch of Theorem 2

We begin with defining the condition that must be satifisfied. A stronger condition with more detailed discussions is presented in Shamir (2015). Let $B^*_{i,t}$ be defined as $B_{i,t}$ for $i \in \{\mathbf{s}^*\}$ and $-B_{i,t}$ for $i \notin \{\mathbf{s}^*\}$. Now, for some constant c, define the interval $I_j(c) \triangleq (cj, c(j+1)]\,, j = 0, 1, 2, \ldots$. Let $N_{i,j}(c) \triangleq \left|\left\{t : B^*_{i,t} \in I_j(c)\right\}\right|$ denote the number of elements over the time t of $B^*_{i,t}$ in interval $I_j(c)$. Let $\bar{x}^M_{i,j} \triangleq \max_{t:B^*_{i,t}\in I_j(c)} |\bar{x}_{i,t}|$, i.e., the maximal absolute base value played for coordinate i in the interval. Then, if there exist a fixed $c > 0$, such that

$$N_{i,j}(c) \cdot \bar{x}^M_{i,j} = o\left\{\text{Regret}\left(\mathbf{s}^*\right)\right\}, \tag{4}$$

then, Theorem 2 and Corollary 3 hold.

The condition descibed merely implies that the benefit of a coordinate that is included in $\mathbf{s}^*$ must grow in a sufficient rate in order for the bounds to hold. If the coordinate is not included in $\mathbf{s}^*$, the benefit of its 0 state must grow in a sufficient rate. However, because $\bar{x}_{i,t}$ also eventually diminishes, that rate can slow in later rounds. The condition holds for typical practical problems such as logistic regression. In the worst case, an adversary can alternate between losses that increase and decrease the benefit. However, it may be rather hard to guarantee that $\bar{x}_{i,t}$ does not converge to some average point and still accumulates benefit at a rate sufficient for the condition to hold. Similarly, if a coordinate should converge to 0, the benefit of the 0 state in the mixture is important mainly at early stages, where MDL speeds convergence to 0. Then, both $\bar{x}_{i,t}$ and the 0 state eventually become almost equal. The condition described here is weaker and simpler than the one described in (Shamir, 2015) in more detail.

To sketch the proof, we first address the sub-gradients $\bar{\mathbf{g}}_t$. Using the observation in (Zinkevich, 2003), the loss $f_t(\tilde{\mathbf{x}}_t)$ can be replaced by $\tilde{\mathbf{g}}_t \cdot \tilde{\mathbf{x}}_t$. We now fix the loss of every round t to be $\tilde{\mathbf{g}}_t \cdot \tilde{\mathbf{x}}_t$. Since, we optimize for $\bar{\mathbf{x}}_t$ in each round, this loss must be expressed in terms of $\bar{\mathbf{x}}_t$. This is done realizing that $\tilde{\mathbf{g}}_t \cdot \tilde{\mathbf{x}}_t = \bar{\mathbf{g}}_t \cdot \bar{\mathbf{x}}_t$ because $\tilde{\mathbf{x}}_t$ is a constant multiple of $\bar{\mathbf{x}}_t$. This yields that Step 12 in Algorithm 3 (shown in Algorithm 1) is, in fact, an FTRL update on a linearized loss. Hence playing $\bar{\mathbf{x}}_t$ on the loss, which is linearized in every round for $\tilde{\mathbf{x}}_t$, achieves the FTRL upper bounds for that loss, which upper bounds the loss f_t.

By the linearization of the loss we play on, and the condition on the regularizer described in Section 3, the update in Step 12 does not change if it is done per coordinate. This implies that $\bar{\mathbf{x}}_t$ will give the same coordinates in the indices selected by a vector $\mathbf{s}$, $\bar{\mathbf{x}}_{\mathbf{s},t}$, as an algorithm that plays $\bar{\mathbf{x}}^{(\mathbf{s})}_t$ only on the coordinates of $\mathbf{s}$. This ties the regret incurred by playing $\bar{\mathbf{x}}_t$ to that of playing $\bar{\mathbf{x}}^{(\mathbf{s})}_t$, specifically for $\mathbf{s} = \mathbf{s}^*$.

Now, we do not actually play $\bar{\mathbf{x}}_t$, but instead we play $\tilde{\mathbf{x}}_t$. It thus remains to show that the additional loss from mixing results in the additional terms to the bound. This applies to both types of terms, for coordinates in $\mathbf{s}^*$ and for coordinates outside of it. The Sigmoid over the benefit score, used to weight the coordinate of a feature in the mixture, in both cases (taking the coordinate for a valuable feature or taking 0 instead of the coordinate for a useless feature), decays exponentially for the less beneficial value of the coordinate in the extra loss of playing it (taking the left tail of the Sigmoid). Therefore, as the gap in losses increases, the weight of the wrong value of the coordinate diminishes. Since an adversary

can play loss functions that either increase or decrease the benefit, we must require a sufficient growth condition on the benefit. This condition implies that over a sufficient window of time, the loss with the wrong value of the coefficient increases sufficiently relative to the loss with the correct value of the coefficient. During rounds the benefit remains bounded within a given interval, loss linear in the length of the interval is incurred. Between intervals, however, the weight of the wrong value of the coefficient diminishes exponentially, and the growth of the overall loss then accumulates over all T rounds to some total which is upper bounded by a constant multiple of the largest $N_{i,j}(c) \cdot \bar{x}_{i,j}^{M}$ for each i. This value is scaled by a function of the prior, $\alpha/(1-\alpha)$ for coordinates for which $s_i^* = 1$, and by $(1-\alpha)/\alpha$ for ones with $s_i^* = 0$. (Note that features with cumulative benefit close to 0 can fall in either category, and the bounds apply in either case.) □