There's a Hole in My Data Space: Piecewise Predictors for Heterogeneous Learning Problems

Ofer Dekel
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA

Ohad Shamir
Microsoft Research
One Memorial Drive
Cambridge, MA 02142, USA

Abstract

We study statistical learning problems where the data space is multimodal and heterogeneous, and constructing a single global predictor is difficult. We address such problems by iteratively identifying high-error regions in the data space and learning specialized predictors for those regions. While the idea of composing localized predictors is not new, our approach is unique in that we actively seek out predictors that clump errors together, making it easier to isolate the problematic regions. When errors are clumped together they are also easier to interpret and resolve through appropriate feature engineering and data preprocessing. We present an error-clumping classification algorithm based on a convex optimization problem, and an efficient stochastic optimization algorithm for this problem. We theoretically motivate our approach with a novel sample complexity analysis for piecewise predictors, and empirically demonstrate its behavior on an illustrative classification problem.

1 Introduction

We consider the problem of statistical supervised learning where the data points are highly heterogeneous and diverse. In other words, the input distribution is a mosaic of multiple distinct components or modes. To give a concrete example, suppose that our goal is to learn a binary classifier that identifies web spam pages (junk pages that have no content other than ads) on the Internet. Of course, the Internet contains extremely varied websites: commerce websites, travel websites, financial websites, song lyric websites, personal homepages, online stores, product homepages, corporate homepages, etc. Web spam may look very different in each of these categories. A classical learning approach would ignore the multimodal nature of the data and attempt to address the web-spam classification problem all at once, across the entire problem domain. However, constructing an accurate global predictor in one shot is often difficult.

Another approach to the web-spam classification task is to partition web pages into topical categories and train a separate classifier within each category. The advantage of such an approach is that each region of the data space can receive a customized solution. However, the topical category of each web page in our training set is not given in the training supervision, and learning an accurate multiclass-multilabel categorizer is a challenge in itself. Moreover, it isn't even clear how to define an appropriate set of categories for this task.

Yet another alternative is to use a clustering algorithm to identify the modes in the training data, and to learn a classifier within each cluster of web pages. If many small clusters are found, the number of examples in each cluster may be very small and the resulting classifier will tend to overfit. If a few large clusters are found, it is unclear whether this rough partitioning of the input space would simplify the original binary classification task.

In this paper, we propose a novel approach to dealing with such heterogeneous problems. Rather than trying to find a classifier that simply minimizes the number of training errors, we attempt to find a classifier whose errors are tightly clustered; we refer to these clusters as *error clumps*. Next, we identify a region of the data space that contains a large error clump and we learn a separate classifier for that region. We view this as *punching a hole* in the data space and tailor-

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 22 of JMLR: W&CP 22.

ing a dedicated solution to that specific part of the problem. This process can be repeated until the error becomes sufficiently small. We refer to this process as *hole punching*. The result is a *piecewise* classifier, made up of a partitioning function that defines the regions in the data space and a classifier for each region.

We present a convex formulation of the binary classification problem that promotes error clusters by discounting the loss of errors that occur in close proximity to other errors. As a result, our error-clumping algorithm only attempts to make accurate predictions on some parts of the data space while forsaking other parts. Unlike the approaches mentioned above (and other localized learning methods proposed in the literature) we let the data determine which parts of the space require a specialized classifier, and avoid a needlessly elaborate partitioning when one is not needed.

The idea of encouraging and detecting clumped errors has many advantages, in addition to improving the accuracy of the final piecewise classifier. In particular, error clumps make a classifier's errors more *interpretable*. This is because close data points that are misclassified together are more likely to have some common properties. Observing multiple misclassified examples with similar properties makes it easier for a human to figure out what went wrong. In response, manual changes can be made to the learning process, say, by engineering new features that are specifically designed to avoid the observed errors.

We present a primal-dual stochastic optimization procedure that efficiently optimizes the convex error-clumping objective. This optimization algorithm may be of independent interest as a means of efficiently solving highly-constrained convex problems. We also derive a novel learning-theoretic analysis that formally motivates our approach. This analysis emphasizes the need to keep the number of holes to a minimum, and therefore justifies our efforts to contain the errors in small geometric regions. This statistical analysis improves upon existing related results and may be of independent interest. We conclude the paper with an illustrative empirical study, which demonstrates the unique behavior of our approach.

1.1 Related Work

The hole-punching loop is somewhat reminiscent of boosting, since we iteratively try to find a predictor for the "hard" examples that were misclassified on previous rounds. Moreover, some concrete implementations of our approach (such as the one used in our experiments in Sec. 5) actually have a boosting-like property that ensures progress on each round. Hole punching differs from traditional boosting algorithms in that it constructs a piecewise classifier, while traditional boosting algorithms typically construct an average or a majority vote of base classifiers. Another substantial difference is that the error-clumping algorithm, which is analogous to the weak learner in boosting, is aware that it is being called as part of the hole-punching loop, as it explicitly tries to make it easier to punch the next hole in the data space. In contrast, a weak learner in a boosting loop does not operate any differently than a standard learning algorithm.

Meir et al. [8] propose a nontraditional boosting algorithm called *Localized Boosting*, which constructs a linear combination of predictors in a locality-dependent manner. [8] also provides an extensive generalization analysis of localized boosting. However, the actual proposed algorithm does not come with the usual guarantees of boosting and is based on a non-convex joint optimization of the predictors as well as the coefficients of their linear combination. The generalization analysis cannot be applied to truly piecewise classifiers, and works only with a soft partition of the data space. Moreover, the bounds in [8] generally scale exponentially with the dimension of the space. Subsequent works (e.g. [1]) improve and generalize the analysis in [8], but the results are still not satisfactory for our purposes (see Sec. 4 for more details).

Kernel methods are also related to our setting. For example, a Gaussian kernel is effectively a localized classifier, as the predicted label of a test point depends mainly on the labels of its proximal support vectors. The main drawback of kernel machines is their inefficient training (often, requiring space and time that is quadratic in the sample size), and the inefficiency of evaluating the kernel classifier on test points. For this reason, kernel classifiers cannot be used for problems such as the large-scale web spam detection problem presented above. In comparison, our approach is both fast to train and the resulting piecewise classifier is fast to evaluate.

Finally, we note that there are quite a few previous papers that propose various heuristics to learn localized piecewise predictors (e.g. [3, 5, 7, 13, 6, 15]). However, none of these methods, nor any of the kernel or boosting methods we are aware of, attempt to *actively* clump the errors into small subsets of the data space. This puts our work on a different and incomparable track from previous approaches to "localized" learning. Moreover, many of these works do not capture the idea of localizing the prediction in the same way we do, and the algorithms do not come with formal guarantees.

2 The Error-Clumping Classifier

Let $\mathcal{X}$ be a data space and for simplicity assume[1] that $\mathcal{X} \subseteq \mathbb{R}^n$. Also, let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of m labeled examples, where each $\mathbf{x}_i \in \mathcal{X}$ and each $y_i \in \{-1, +1\}$.

We focus on linear predictors of the form $\mathbf{x} \mapsto \langle \mathbf{x}, \mathbf{w} \rangle$, parameterized by a vector $\mathbf{w}$, where the sign of $\langle \mathbf{x}, \mathbf{w} \rangle$ is taken to be the predicted value of the label y. Finding the predictor $\mathbf{w}$ that minimizes the classification error $\mathbf{1}_{\text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) \neq \mathbf{y}}$ on the training set is a hard combinatorial problem. A common way to deal with this is to replace the non-convex misclassification error by a convex surrogate, such as the hinge loss, $\max\{0, 1 - y\langle \mathbf{x}, \mathbf{w} \rangle\}$. As described in the introduction, we wish to derive a learning algorithm that takes the geometry of the errors into account. To that end, fix a parameter $\rho > 0$ and define the *neighborhood* of a training point $\mathbf{x}_i$ as

$$N_i \;=\; \{j \in [m] \setminus \{i\} \;:\; \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \rho\} \quad ,$$

where we use $[m]$ as shorthand for $\{1, \ldots, m\}$. Fix a second parameter $\eta \in [0, 1]$ and define the matrix $\Theta = (\theta_{i,j}) \in \mathbb{R}^{m \times m}$ as follows

$$\theta_{i,j} \;=\; \begin{cases} 1 & \text{if } \; j = i \\ -\frac{\eta}{m-1} & \text{if } \; j \in N_i \\ 0 & \text{otherwise} \end{cases} \quad .$$

The matrix Θ is positive semidefinite (the proof is presented in the supplementary material). Finally, fix a third parameter $\lambda \geq 0$ and consider the following SVM-like optimization problem

$$\begin{aligned} &\operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^m} \; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{2m}\boldsymbol{\xi}^\top \Theta \boldsymbol{\xi} \qquad (1) \\ &\text{s.t. } \forall i \;\; \xi_i \geq 0 \;\text{ and }\; 1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \xi_i \quad . \end{aligned}$$

Since Θ is positive semidefinite, Eq. (1) is a convex optimization problem. First, note that when $\eta = 0$, Θ is simply the identity matrix and $\boldsymbol{\xi}^\top \Theta \boldsymbol{\xi}$ equals $\sum_{i=1}^m \xi_i^2$. In this case, as in the standard SVM formulation, each $\xi_i = \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle\}$, and therefore $\boldsymbol{\xi}^\top \Theta \boldsymbol{\xi}$ equals the *squared-hinge-loss* incurred by the classifier $\mathbf{w}$ on S. On the other hand, when $\eta > 0$, the optimization problem in Eq. (1) is forgiving of classification errors that occur in close proximity to each other. To see this, note that $\frac{1}{2m}\boldsymbol{\xi}^\top \Theta \boldsymbol{\xi}$ can be rewritten as

$$\frac{1}{2m}\sum_{i=1}^m \xi_i^2 \;-\; \frac{\eta}{2m(m-1)}\,\xi_i \sum_{j \in N_i} \xi_j \quad .$$

[1] Our results extend to kernelized methods, where $\mathcal{X}$ is a subset of a reproducing kernel Hilbert space.

In other words, a non-zero ξ_i incurs a *penalty* proportional to ξ_i^2, but receives a *discount* proportional to $\xi_i \frac{\eta}{m-1} \sum_{j \in N_i} \xi_j$. Thus, a classification mistake made in proximity to other classification mistakes is not penalized as much as a classification mistake made in isolation.

Since Θ is positive semidefinite, $\boldsymbol{\xi}^\top \Theta \boldsymbol{\xi} \geq 0$ for all $\boldsymbol{\xi}$, and therefore the penalties always overpower the discounts. Moreover, it is still the case that $1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \xi_i$ for all $i \in [m]$ and that the optimal $\boldsymbol{\xi}$ is the zero vector.

Also observe that it is no longer necessarily the case that $\xi_i = 1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle$ for all $i \in [m]$. In fact, the examples for which $\xi_i > 1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle$ are the interesting ones, since the right hand side is free to increase slightly without incurring an additional penalty in the objective function.

3 An Efficient Implementation

While the optimization problem in Eq. (1) is convex and can be solved by generic convex optimization tools, these might be quite slow in practice. In recent years, simple and efficient first-order algorithms, such as stochastic gradient descent, have become increasingly popular for solving such problems (e.g. [11]). However, such methods require projection to the feasible set at each iteration. This is easy when the feasible set has a simple structure, such as a ball. However, in Eq. (1), the feasible set is a complex polyhedron (due to the m constraints of the form $1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \xi_i$), and projecting onto this set might not be easy. The standard trick of reducing the constrained optimization problem to an unconstrained problem by replacing ξ_i with $\max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle\}$ in the objective function (as in [11]) will not work here, since it is not necessarily true that $\xi_i = \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i \rangle\}$.

In this section, we present a primal-dual approach that retains the computational efficiency and scalability of first-order methods and is tailored to convex optimization problems with many constraints. Our algorithm directly finds a saddle-point of the Lagrangian, and differs from previous works in this direction (such as [9]) both algorithmically (for instance, we iteratively optimize the dual variables in closed-form) and in terms of the analysis.

Consider the general optimization problem

$$\min_{\mathbf{w} \in \mathcal{W}} \; f(\mathbf{w}) \;\text{ s.t. }\; \forall\, i \in [m] \;\; g_i(\mathbf{w}) \leq 0 \quad , \qquad (2)$$

where $f, g_1, \ldots, g_m$ are convex functions and $\mathcal{W}$ is a convex subset of Euclidean space. As in our case, it is often desirable to soften the constraints by introducing a vector of slack variables $\boldsymbol{\xi}$ and augmenting the

problem as follows

$$\min_{\mathbf{w}\in\mathcal{W},\boldsymbol{\xi}\in\mathbb{R}^m} \; f(\mathbf{w}) + h(\boldsymbol{\xi}) \qquad (3)$$
$$\text{s.t. } \forall\, i \in [m] \;\; g_i(\mathbf{w}) \le \xi_i \;\text{ and }\; \xi_i \ge 0 \;\;,$$

where h is a convex function that penalizes constraint violations. Our goal is to solve this relaxed problem. The Lagrangian of this problem is denoted by $\mathcal{L}(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})$ and defined as

$$f(\mathbf{w}) + h(\boldsymbol{\xi}) + \sum_{i=1}^{m} \alpha_i\big(g_i(\mathbf{w}) - \xi_i\big) - \sum_{i=1}^{m} \beta_i\xi_i, \qquad (4)$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of dual variables. The optimization problem in Eq. (3) is equivalent to the following Lagrangian saddle-point problem

$$\min_{\mathbf{w}\in\mathcal{W},\boldsymbol{\xi}\in\mathbb{R}^m}\; \max_{\boldsymbol{\alpha}\in\mathbb{R}^m,\boldsymbol{\beta}\in\mathbb{R}^m} \mathcal{L}(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) \qquad (5)$$
$$\text{s.t. } \forall\, i \in [m] \;\; \alpha_i \ge 0 \;\text{ and }\; \beta_i \ge 0.$$

We propose to solve Eq. (5) using Algorithm 1, which alternates between optimizing the dual variables and the primal variables. For the algorithm to converge, we add an additional constraint on the dual variables. Let $\Gamma(\mathbf{w},\boldsymbol{\xi})$ be the set of dual variables $(\boldsymbol{\alpha},\boldsymbol{\beta})$ that satisfy $\boldsymbol{\alpha}\ge 0$, $\boldsymbol{\beta}\ge 0$, $\boldsymbol{\alpha}+\boldsymbol{\beta} = \max\{0, \nabla h(\boldsymbol{\xi})\}$ and for all $i\in[m]$, if $g_i(\mathbf{w}) < \xi_i$ then $\alpha_i = 0$ and if $\xi_i > 0$ then $\beta_i = 0$. Let $(\mathbf{w}^\star,\boldsymbol{\xi}^\star,\boldsymbol{\alpha}^\star,\boldsymbol{\beta}^\star)$ denote a primal-dual solution to Eq. (5) and note that the KKT optimality conditions guarantee that $(\boldsymbol{\alpha}^\star,\boldsymbol{\beta}^\star) \in \Gamma(\mathbf{w}^\star,\boldsymbol{\xi}^\star)$. Therefore, adding the constraint $(\boldsymbol{\alpha},\boldsymbol{\beta})\in\Gamma(\mathbf{w},\boldsymbol{\xi})$ to Eq. (5) does not change the optimum of our problem.

We now describe each iteration of Algorithm 1 in more detail. We start iteration t with the primal-dual point $(\mathbf{w}_t,\boldsymbol{\xi}_t,\boldsymbol{\alpha}_t,\boldsymbol{\beta}_t)$ and the gradient accumulator variables $\mathbf{u}_t$ and $\boldsymbol{\mu}_t$. Although this is not explicit in the algorithm pseudo-code, we first compute optimal dual variables

$$(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1}) = \operatorname*{argmax}_{(\boldsymbol{\alpha},\boldsymbol{\beta})\in\Gamma(\mathbf{w}_t,\boldsymbol{\xi}_t)} \mathcal{L}(\mathbf{w}_t,\boldsymbol{\xi}_t,\boldsymbol{\alpha},\boldsymbol{\beta}).$$

Note that we require the dual variables to satisfy $(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1})\in\Gamma(\mathbf{w}_t,\boldsymbol{\xi}_t)$. This step has the following closed form solution:

$$\alpha_{t+1,i} = \begin{cases} \frac{\partial}{\partial\xi_i}h(\boldsymbol{\xi}_t) & \text{if } \xi_{t,i} = g_i(\mathbf{w}_t) \text{ and } \frac{\partial}{\partial\xi_i}h(\boldsymbol{\xi}_t) > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$\beta_{t+1,i} = \begin{cases} \frac{\partial}{\partial\xi_i}h(\boldsymbol{\xi}_t) & \text{if } \xi_{t,i} = 0 \text{ and } \frac{\partial}{\partial\xi_i}h(\boldsymbol{\xi}_t) > 0 \\ 0 & \text{otherwise} \end{cases} .$$

Next, we keep the dual variables constant and perform a stochastic dual-averaging update of the primal variables (as in [14, 10]). The dual-averaging update involves two substeps: first we choose a random sample $I_t \in [m]$ and add the stochastic primal gradients to our accumulator variables:

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \nabla_{\mathbf{w}}\big(f(\mathbf{w}_t) + m\,\alpha_{t+1,I_t}\,g_{I_t}(\mathbf{w}_t)\big)$$
$$\mu_{t+1,I_t} = \mu_{t,I_t} + m\Big(\frac{\partial}{\partial\xi_{I_t}}h(\boldsymbol{\xi}_t) - \alpha_{t+1,I_t} - \beta_{t+1,I_t}\Big).$$

The closed form for $(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1})$ and the definitions of $(\mathbf{u}_{t+1},\boldsymbol{\mu}_{t+1})$ can be combined into a single step as follows: if $\xi_{t,I_t} = g_{I_t}(\mathbf{w}_t)$ and $\frac{\partial}{\partial\xi_{I_t}}h(\boldsymbol{\xi}_t) > 0$ then we set

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \nabla f(\mathbf{w}_t) + m\nabla g_{I_t}(\mathbf{w}_t)\tfrac{\partial}{\partial\xi_{I_t}}h(\boldsymbol{\xi}_t),$$

and otherwise $\mathbf{u}_{t+1} = \mathbf{u}_t + \nabla f(\mathbf{w}_t)$. Similarly, if $\xi_{t,I_t} > [g_{I_t}(\mathbf{w}_t)]_+$ and $\frac{\partial}{\partial\xi_{I_t}}h(\boldsymbol{\xi}_t) > 0$, or if $\frac{\partial}{\partial\xi_{I_t}}h(\boldsymbol{\xi}_t) < 0$ then

$$\mu_{t+1,I_t} = \mu_{t,I_t} + m\tfrac{\partial}{\partial\xi_{I_t}}h(\boldsymbol{\xi}_t) \;\;,$$

and otherwise μ_{t+1,I_t} remains unchanged. This combined step is given in the top part of Algorithm 1.

Finally, we perform the second substep of dual-averaging: we scale the gradient accumulators $(\mathbf{u}_{t+1},\boldsymbol{\mu}_{t+1})$ by a learning rate η_t and project them onto the set of primal constraints to obtain $(\mathbf{w}_{t+1},\boldsymbol{\xi}_{t+1})$. The scaling and projection step appears at the bottom of Algorithm 1.

To analyze our algorithm, we observe that it is essentially applying dual-averaging [14, 10] to the sequence of functions $\mathcal{L}_1,\mathcal{L}_2,\ldots$, where $\mathcal{L}_t(\mathbf{w},\boldsymbol{\xi}) = \mathcal{L}(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1})$. Therefore, known regret bounds for dual-averaging (see section 3.1 in [14]) guarantee that for any T it holds that

$$\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}_t,\boldsymbol{\xi}_t) - \inf_{\mathbf{w}\in\mathcal{W},\boldsymbol{\xi}\in\mathbb{R}^m}\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w},\boldsymbol{\xi}) \le R(T) \;\;,$$

where $R(T)$ is $\theta(T^{-\frac{1}{2}})$. It remains to show that this bound implies that Algorithm 1 converges to the optimum of Eq. (3). Indeed, one can show that the regret bound given above implies that

$$f(\bar{\mathbf{w}}_T) + h(\bar{\boldsymbol{\xi}}_T) \;\le\; f(\mathbf{w}^\star) + h(\boldsymbol{\xi}^\star) + R(T) \;\;.$$

The proof appears in the supplementary material. Our proof is not specific to dual-averaging and applies to any other stochastic optimization procedure, provided that it comes with a sufficient online regret guarantee.

In the concrete setting of Eq. (1), we are interested in the case where $f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2$, $h(\boldsymbol{\xi}) = \frac{1}{2m}\boldsymbol{\xi}^\top\Theta\boldsymbol{\xi}$, and $g_i(\mathbf{w}) = 1 - y_i\langle\mathbf{w},\mathbf{x}_i\rangle$. Plugging these definitions into

Algorithm 1 Algorithm For solving Eq. (3)

Input: Learning rate η_t, number of rounds T
Initialize $\mathbf{w}_1 = 0,\ \boldsymbol{\xi}_1 = 0,\ \mathbf{u}_1 = 0,\ \boldsymbol{\mu}_1 = 0$
for $t = 1, 2, \ldots, T$ **do**
  Uniformly sample an index $I_t \in \{1, \ldots, m\}$
  $\mathbf{u}_{t+1} := \mathbf{u}_t + \nabla f(\mathbf{w}_t)$
    $+\ m \nabla g_{I_t}(\mathbf{w}_t) [\![\xi_{t,I_t} = g_{I_t}(\mathbf{w}_t)]\!] [\![\frac{\partial h(\boldsymbol{\xi}_t)}{\partial \xi_{I_t}} > 0]\!]$
    where $[\![\cdot]\!]$ denotes the indicator function.
  $\boldsymbol{\mu}_{t+1} := \boldsymbol{\mu}_t$
  if $\xi_{t,I_t} > \max\{0, g_{I_t}(\mathbf{w}_t)\}$ and $\frac{\partial}{\partial \xi_{I_t}} h(\boldsymbol{\xi}_t) > 0$,
    or $\frac{\partial}{\partial \xi_{I_t}} h(\boldsymbol{\xi}_t) < 0$ **then**
    $\mu_{t+1,I_t} := \mu_{t+1,I_t} + m \frac{\partial}{\partial \xi_{I_t}} h(\boldsymbol{\xi}_t)$.
  end if
  $\mathbf{w}_{t+1} := \operatorname{argmin}_{\mathbf{w} \in \mathcal{W}} \| - \eta_t \mathbf{u}_t - \mathbf{w} \|$
  $\forall\, i \quad \xi_{t+1,i} := \max\{-\eta_t \mu_{t+1,i},\ g_i(\mathbf{w}_{t+1}),\ 0\}$
end for
Return $(\bar{\mathbf{w}}_T, \bar{\boldsymbol{\xi}}_T) = (\frac{1}{T}\sum_{t=1}^T \mathbf{w}_t, \frac{1}{T}\sum_{t=1}^T \boldsymbol{\xi}_t)$

the equations in Algorithm 1 gives the update rule

$$\begin{aligned}
\mathbf{u}_{t+1} \ &\leftarrow\ \mathbf{u}_t + \lambda \mathbf{w}_t - \\
&\qquad \Theta_{I_t} \boldsymbol{\xi}_t \ y_{I_t} \mathbf{x}_{I_t} [\![\xi_{t,I_t} = g_{I_t}(\mathbf{w}_t)]\!] [\![\Theta_{I_t} \boldsymbol{\xi}_t > 0]\!] \\
\mu_{t+1,I_t} \ &\leftarrow\ \mu_{t,I_t} \ +\ \Theta_{I_t} \boldsymbol{\xi}_t \Big([\![\Theta_{I_t} \boldsymbol{\xi}_t < 0]\!] \ + \\
&[\![\xi_{t,I_t} > 0]\!] [\![\xi_{t,I_t} > 1 - y_{I_t} \langle \mathbf{w}_t, \mathbf{x}_{I_t} \rangle]\!] [\![\Theta_{I_t} \boldsymbol{\xi}_t > 0]\!] \Big),
\end{aligned}$$

where Θ_i is the i'th row of the matrix Θ, and where $[\![\cdot]\!]$ denotes the indicator function.

4 Sample Complexity of Piecewise Predictors

We now turn to the learning-theoretic properties of our approach. The hole-punching algorithm iteratively constructs a piecewise predictor, which consists of a data space partitioning function and a predictor for each piece of the space. The partitioning function associates each data point with one of the predictors and that predictor determines the label. The relevant learning-theoretic question is how well do such piecewise predictors generalize.

To formalize this, let $\mathcal{G}$ be a class of binary-valued functions on $\mathcal{X}$, and let $\mathcal{H}$ be a class of real-valued function on $\mathcal{X}$. $\mathcal{G}$ is the set of indicator functions of holes in our data space. For example, $\mathcal{G}$ could be the set of indicator functions of all the bounded-radius Euclidean balls in $\mathcal{X}$. $\mathcal{H}$ is the set of candidate predictors used within each region, and assume without loss of generality that $\sup_{\mathbf{x} \in \mathcal{X}} |h(\mathbf{x})| \leq 1$. We say that a set of functions $g_1, \ldots, g_k \in \mathcal{G}$ is *disjoint* if, for any $\mathbf{x} \in \mathcal{X}$, $g_j(\mathbf{x}) = 1$ for only a single j. A piecewise predictor can now be written as

$$h(\mathbf{x}) \bar{g}(\mathbf{x}_i) \ +\ \sum_{j=1}^k h_j(\mathbf{x}) g_j(\mathbf{x}) \quad , \tag{6}$$

where $h, h_1, \ldots, h_k \in \mathcal{H}$, $g_1, \ldots, g_k \in \mathcal{G}$ are disjoint, and $\bar{g}(\mathbf{x}) = 1 - \sum_{j=1}^k g_j(\mathbf{x})$. In words, we have an ambient classifier h that applies to data points that are not in any hole (a region indicated by $\bar{g}$) and k disjoint holes (indicated by $g_1, \ldots, g_k$) with respective predictors $h_1, \ldots, h_k$.

We analyze the generalization performance of piecewise predictors by bounding the Rademacher complexity [2] of this predictor class. We recall that for any predictor class $\mathcal{F}$, the empirical Rademacher complexity with respect to a sample $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is defined as $\mathcal{R}_m(\mathcal{F}) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \sigma_i f(\mathbf{x}_i)\right]$, where σ_i are i.i.d. random variables distributed uniformly on $\{-1, +1\}$. If the loss function is Lipschitz in the prediction value, then we can upper bound the difference between the expected loss and the average loss on the sample, uniformly for all predictors, by $\mathcal{O}\big(\mathcal{R}_m(\mathcal{F}) + \sqrt{\log(1/\delta)/m}\big)$, with probability at least $1 - \delta$ (see [4]).

Let $\mathcal{F}$ be our class of piecewise predictors, as in Eq. (6). By definition of $\bar{g}(\mathbf{x})$, each such predictor can be written as $h(\mathbf{x}) + \sum_{j=1}^k (h_j(\mathbf{x}) - h(\mathbf{x})) g_j(\mathbf{x})$. By definition of the Rademacher complexity, it is easy to verify that this implies that $\mathcal{R}_m(\mathcal{F}) \leq \mathcal{R}_m(\mathcal{H}) + 2\mathcal{R}_m(\mathcal{H} \otimes^k \mathcal{G})$, where $\mathcal{H} \otimes^k \mathcal{G}$ consists of all functions of the form $\sum_{j=1}^k h_j(\mathbf{x}) g_j(\mathbf{x})$ where $h_1, \ldots, h_k \in \mathcal{H}$ and $g_1, \ldots, g_k \in \mathcal{G}$ are disjoint. We will shortly assume that $\mathcal{R}_m(\mathcal{H})$ is bounded, so we can focus on bounding $\mathcal{R}_m(\mathcal{H} \otimes^k \mathcal{G})$.

Clearly, $\mathcal{R}_m(\mathcal{H} \otimes^k \mathcal{G})$ should depend somehow on the complexity of $\mathcal{H}$ and $\mathcal{G}$ and on the number k. We measure the complexity of $\mathcal{H}$ by its Rademacher complexity, $\mathcal{R}_m(\mathcal{H})$, which is a natural choice given that $\mathcal{H}$ is a set of real-valued functions. We measure the complexity of $\mathcal{G}$ by its VC-dimension $d_{\mathcal{G}}$. This is also the natural choice, since we think of $\mathcal{G}$ as a space of balls, and these classes are standard objects in VC theory. To compare our results with other results discussed below, we note that a VC dimension of $d_{\mathcal{G}}$ implies an upper bound of $\mathcal{O}(\sqrt{d_{\mathcal{G}} \log(m)/m})$ on the Rademacher complexity of $\mathcal{G}$, via Massart's lemma.

The existing literature already provides tools that can be used to bound $\mathcal{R}_m(\mathcal{H} \otimes^k \mathcal{G})$. Theorem 14 in [2] leads to the bound $k\sqrt{k}(\mathcal{R}_m(\mathcal{H}) + \mathcal{R}_m(\mathcal{G}))$, which has a rather disappointing dependence on k. A more refined approach first notes that $\mathcal{R}_m(\mathcal{H} \otimes^k \mathcal{G})$ can be upper

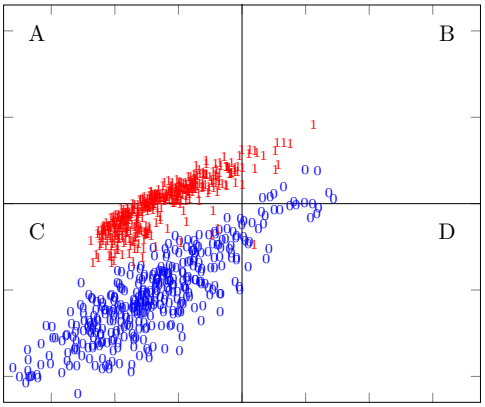

Figure 1: Illustration of 0's (blue) and 1 (red) in the training data (see main text).

bounded as follows:

$$\sup_{\substack{h_1,\ldots,h_k,g_1,\ldots,g_k \\ \text{disjoint}}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i \sum_{j=1}^{k} h_j(\mathbf{x}_i)g_j(\mathbf{x})$$
$$\leq \sum_{j=1}^{k} \sup_{h_j\in\mathcal{H},g_j\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i h_j(\mathbf{x}_i)g_j(\mathbf{x}) \quad .$$

We can then apply Thm. 14 from [2] to each summand separately, to get a bound of $\mathcal{R}_m(\mathcal{H}\otimes^k\mathcal{G}) \leq k(\mathcal{R}_m(\mathcal{H})+\mathcal{R}_m(\mathcal{G}))$. However, this decomposition ignores the constraint that $g_1,\ldots,g_k$ are disjoint and the resulting bound grows linearly with k. A similar problem is also considered in [1], and the resulting bound again grows linearly with k.

Explicitly using the fact that $g_1,\ldots,g_k$ are disjoint, we can prove a better bound that scales as $\sqrt{k}(\mathcal{R}_m(\mathcal{H})+\mathcal{R}_m(\mathcal{G}))$:

Theorem 1. *Suppose $\mathcal{R}_m(\mathcal{H})$ can be upper bounded by $\sqrt{d_{\mathcal{H}}/m}$ for all $m\geq 1$, and that $\mathcal{G}$ has VC-dimension at most $d_{\mathcal{G}}$, where $d_{\mathcal{G}}>2$. Then the Rademacher complexity of $\mathcal{H}\otimes^k\mathcal{G}$ is at most*

$$\left(\sup_{g_1,\ldots,g_k\in\mathcal{G}}\sum_{j=1}^{k}\sqrt{|i:g_j(\mathbf{x}_i)=1|}\right)\frac{\sqrt{d_{\mathcal{H}}}}{m} + 8\sqrt{\frac{kd_{\mathcal{G}}\log(m)}{m}},$$

which can be upper bounded in turn by $\sqrt{k}\left(\sqrt{d_{\mathcal{H}}/m}+8\sqrt{d_{\mathcal{G}}\log(m)/m}\right)$.

The proof, which is rather technical, appears in the supplementary material.

5 Experiments

We conducted a set of experiments using the MNIST dataset of $70K$ handwritten digits. We split the ten digits into two sets and the binary classification task was to distinguish between digits from the two sets. Although this binary classification task is somewhat synthetic, it perfectly simulates a situation where each binary class is composed of multiple distinct parts. This makes it a perfect problem on which to demonstrate the unique behavior of our algorithm.

We randomly split the $70K$ examples into equal size training and test sets. Each data point in the dataset was represented by a vector of pixel intensities, plus a constant feature. Each training example was given a binary label; the true 10-category digit label was hidden from the algorithm.

5.1 Paying with Ones and Gaining Zeros

We began with the task of distinguishing the digits $0,1,2,3,4$ from $5,6,7,8,9$. We ran the error-clumping algorithm with three different values of η, the parameter that controls the discount given to neighboring errors. First, we set $\eta=0$, which reduces our loss function to the vanilla squared-hinge-loss, and obtained the linear classifier $\mathbf{w}_0$. Then we reran our algorithm with $\eta=0.3$ and $\eta=0.5$, to obtain $\mathbf{w}_{0.3}$ and $\mathbf{w}_{0.5}$. Table 1 shows the per-category test error rate attained by each classifier. On the digit 1, $\mathbf{w}_0$ achieves a solid 0.066 error rate, while $\mathbf{w}_{0.5}$ attains an error rate that is 10 times larger. On the digit 0 we observe the opposite behavior: $\mathbf{w}_0$ attains a 0.208 error rate, while $\mathbf{w}_{0.5}$ reduces this error rate by a factor of 4.

The error-clumping algorithm with $\eta>0$ essentially gives up on the 1's for the sake of improving its performance on the 0's. It does this because the 1's are tightly clustered whereas the 0's are scattered. Therefore, errors on 1's are significantly discounted. The idea is that the tight cluster of 1's can be dealt with separately using a dedicated classifier.

To visualize the tight cluster of 1's, we must project the 784-dimensional data space onto two dimensions. We choose the two most natural dimensions: $\mathbf{w}_0$ and $\mathbf{w}_{0.5}$. The projected training set is depicted in Fig. 1. The x-axis represents $\langle\mathbf{w}_0,\mathbf{x}\rangle$ and the y-axis represents $\langle\mathbf{w}_{0.5},\mathbf{x}\rangle$.

The four quadrants in Fig. 1 are named A,B,C,D. Since both 0 and 1 are assigned negative binary labels, correct predictions have negative values. In other words, the prediction errors of $\mathbf{w}_0$ are contained in $B\cup D$ and the prediction errors of $\mathbf{w}_{0.5}$ are contained in $A\cup B$. Quadrant A contains the tight cluster of 1's that $\mathbf{w}_0$ predicted correctly and $\mathbf{w}_{0.5}$ forsook, while quadrant D contains those scattered 0's that $\mathbf{w}_0$ predicted incorrectly and $\mathbf{w}_{0.5}$ salvaged.

digit	0	1	2	3	4	5	6	7	8	9
$\mathbf{w}_0$	0.208	0.066	0.076	0.206	0.788	0.346	0.383	0.097	0.331	0.064
$\mathbf{w}_{0.3}$	0.102	0.185	0.052	0.236	0.750	0.299	0.487	0.097	0.254	0.081
$\mathbf{w}_{0.5}$	0.056	0.615	0.044	0.305	0.673	0.243	0.613	0.166	0.187	0.127

Table 1: Per-category test error rates of three classifiers, trained with different values of η.

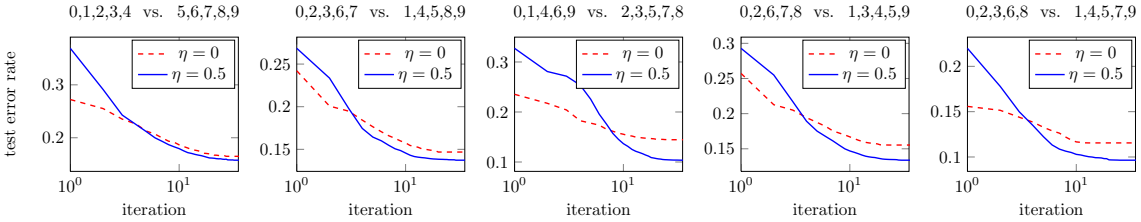

Figure 2: Test error rate of the hole-punching algorithm on 5 binary classification problems, with and without error clumping.

5.2 Hole-Punching with Error-Clumping

Next, we experimented with the hole-punching procedure. Again we focused on the binary classification problem of distinguishing between two sets of digits: $0, 1, 2, 3, 4$ versus $5, 6, 7, 8, 9$, as well as other random splits of the 10 digits into two sets.

We ran our iterative hole-punching algorithm using the error-clumping classifier within each hole. We defined holes in our data space using Euclidean balls. Namely, each hole-punching iteration finds a ball that contains many misclassified training points and trains a specialized classifier within that ball.

We used the following heuristic to define the ball on each iteration: We found the training example whose neighborhood (as defined by the matrix Θ) contains the largest number of classification errors; this example became the center of the ball. The radius of the ball was set to be the largest number such that the examples that fall inside the ball are more than 50% misclassified. Note that this procedure is linear in the number of training examples. It also ensures that the hole-punching loop is indeed a boosting algorithm: once a hole is identified, a trivial constant classifier already reduces the training error inside the hole from more than 50% to at most 50%. This guarantees that the training error will decline monotonically to zero. In the worst case, the balls will be tiny, and their number will be comparable to the number of training examples (leading to severe overfitting). Of course, we would expect better on realistic datasets.

We repeated the experiment with $\eta = 0$ (squared-hinge-loss) and $\eta = 0.5$ (significant discount given to error clumps). We tuned the other algorithm parameters appropriately. Fig. 2 shows the test error rate as a function of the hole-punching iteration, on a number of binary problems. As expected, the error-clumping algorithm delivers inferior results on the first few iterations, but quickly starts to pay off. The fact that errors are clumped together makes it easy for the hole-puncher to find large balls that contain many mistakes.

6 Discussion

We presented the hole-punching algorithm for learning piecewise predictors in heterogeneous learning problems. We built on the idea of finding predictors that prefer errors that occur in close proximity to other errors, making it easy to identify small problematic regions in the data space and to deal with these regions separately. We formulated the error-clumping idea as a convex optimization problem and solved it using an efficient new algorithm. We also presented a learning-theoretic analysis that shows how the number of pieces in a piecewise predictor and the complexity of these pieces affect generalization. This analysis justifies our desire to find small concentrated high-error regions of the data space.

The hole-punching procedure can be viewed as a boosting algorithm, with the error-clumping algorithm as its weak learner. To our knowledge, ours is the first boosting algorithm where the weak learner is aware that it is part of an interactive loop, and therefore behaves differently than a stand-alone learning algorithm.

Most of this paper focuses on binary classification problems, however, the ideas and concepts easily extend to other supervised learning problems. Similarly, we defined the neighborhood of each data point using Euclidean distance in the data space; in fact any other notion of neighborhood could be used without any change.

Beyond our specific contributions, we believe our paper also has an important conceptual message: instead of taking the simplistic viewpoint of learning as a one-shot process using some fixed dataset, one can view the entire learning pipeline as a whole, including the feature engineering and data pre-processing steps. In our case, we developed an algorithm that goes a step beyond minimizing classification error, and also tries to make the errors more interpretable. The error clumps expose problems in the data representation and help the user fix the learning process accordingly.

References

[1] A. Azran and R. Meir. Data dependent risk bounds for hierarchical mixture of experts classifiers. In *COLT*, 2004.

[2] P. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 3:463–482, 2002.

[3] L. Bobrowski. Piecewise-linear classifiers, formal neurons and separability of the learning sets. In *ICPR*, 1996.

[4] O. Bousquet Boucheron, S. and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323 – 375, 2005.

[5] J. Dai, S. Yan, X. Tang, and J. Kwok. Locally adaptive classification piloted by uncertainty. In *ICML*, 2006.

[6] P. Hartono. Ensemble of linear experts as an interpretable piecewise-linear classifier. *ICIC Express Letters*, 2(3), 2008.

[7] A. Kostin. A simple and fast multi-class piecewise linear pattern classifier. *Pattern Recognition*, 39:1949–1962, 2006.

[8] R. Meir, R. El-Yaniv, and S. Ben-David. Localized boosting. In *COLT*, 2000.

[9] A. Nedić and A. Ozdaglar. Subgradient methods for saddle point problems. *J. of Optimization Theory and Applications*, 142(1):205–228, 2009.

[10] Yu. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, August 2009.

[11] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *ICML*, 2007.

[12] O. Shamir. A variant of azuma's inequality for martingales with subgaussian tails. ArXiv Technical Report, 2011.

[13] M. Toussaint and S. Vijayakumar. Learning discontinuities with products-of-sigmoids for switching between local models. In *ICML*, 2005.

[14] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

[15] L. Yujian, L. Bo, Y. Xinwu, F. Yaozong, and L. Houjun. Multiconlitron: A general piecewise linear classifier. *IEEE Trans. Neural Networks*, 22(2):276–289, 2011.