Journal of Machine Learning Research 1 (2017) — Submitted —; Published —

Depth without distress

Shiva Kaul
SKKAUL@CS.CMU.EDU
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Geoffrey J. Gordon
GGORDON@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Editor: —-

Abstract

Upgrading a linear classifier to a nonlinear one leads to statistical or computational compromise. For example, lifting its dimension, as in kernel methods, may lead to overfitting. Nonconvex optimization algorithms, used for (deep) neural networks or decision trees, may not obtain high-quality solutions or even converge. Similarly, boosting can get stuck on increasingly difficult subproblems. We avoid concerns about overfitting and convergence with new nonlinear classifiers — smooth lists of halfspaces — and a new learning algorithm — the sequence of averages. Smooth lists have unbounded depth, yet do not need more data, in the worst case, than single linear classifiers. Theory suggests the algorithm never gets prematurely stuck and monotonically improves the classifier. In experiments inspired by challenging problems in computational learning theory, it performs well when only a moderate amount of data and time are available, and there is non-trivial noise or inconsistency. At a high level, we isolate a kind of deep learning which is easier to reason about but retains some of the advantages of depth.

1. Introduction

A classifier takes an input x and outputs a binary value in $\{-1, 1\}$. Supervised learning is the task of looking at data of the form "input: x, output: y" and choosing a classifier c which, given a new input, returns a (typically) correct output. We hope to complete this task using as little time and data as possible. The data are randomly, independently generated by an unknown probability distribution $\mathcal{D}$. We wish to maximize the correlation of the classifier c with $\mathcal{D}$ (i.e., minimize the error probability):

$$\chi(c) = \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{D}}(c(x)y) = 1 - 2 \mathop{\mathbf{P}}_{(x,y)\sim\mathcal{D}}(c(x) \neq y) \tag{1}$$

Linear classifiers, defined by vectors $w \in \mathbb{R}^n$, operate upon inputs $x \in \mathbb{R}^n$ only via an inner product $\langle w, x\rangle = \sum_{i=1}^{n} w_i x_i$. The archetypal linear classifier is a halfspace, which returns -1 or 1 according to the sign of the inner product: $\mathrm{sgn}(\langle w, x\rangle)$. Linear classifiers are convenient, but they may not be expressive enough to achieve high correlation. Even if a good one exists, finding it may be computationally intractable due to noise (Feldman et al., 2009; Zhang et al., 2015). In limited scenarios — e.g. if the inputs have logconcave distribution and the noise probability is bounded — modern algorithms manage to learn a good linear classifier (Awasthi et al., 2016; Zhang et al., 2015, 2017). Nonetheless, the prevalent approach to fitting more complicated data or avoiding computational intractability is to use nonlinear classifiers; this is called 'improper learning' when linear classifiers are kept as a competitive baseline. Unfortunately, it leads to two new concerns: overfitting and reliable convergence.

Lifting (as in kernel SVM and polynomial regression) increases the dimension of the linear classifier by mapping each input x to a a feature vector $\phi(x)$ in a reproducing kernel Hilbert space, and minimizing a convex objective thereof. Such algorithms can be robust to arbitrary noise, and in some scenarios, are provably the only such methods (Kalai et al., 2008; Daniely, 2014; Dachman-Soled et al., 2014). Solutions to the convex optimization may be simply characterized and reliably found. However, lifting may require substantially more data (Bun and Steinke, 2015; Poggio et al.; Daniely et al., 2012). This tradeoff is witnessed in practice and illustrated by experiments in this paper.

Composition (as in decision trees or deep neural networks) produces expressive and succinct classifiers. Some decision tree algorithms achieve optimal error rate asymptotically as the amount of data increases (Scott and Nowak, 2006); however, they may need much more data as the dimension grows (Bengio et al., 2010). Characterizing the data needs of neural networks is a complex, open question. Neural networks are extremely capable of overfitting, but typically don't due to some benign and poorly understood properties of natural data (Zhang et al., 2016; Karolina Dziugaite and Roy, 2017). Because learning deep classifiers typically involves nonconvex optimization, it is difficult to analyze the convergence behavior of the learning algorithm, or guarantee the quality of the resultant classifier. In some scenarios, local minima may be satisfactory (Kawaguchi, 2016; Soltanolkotabi et al., 2017), but avoiding saddle points still requires care (Du et al., 2017; Jin et al., 2017). Even as we improve our understanding of deep classifiers in general, we seek an analytically simple subset which retains some of the advantages of depth. Indeed, prior analysis restricts the architecture of the network, the activation functions, the eigenvalues of saddle points, etc.

Boosting algorithms (such as AdaBoost) produce convex or linear combinations ('ensembles') by iteratively (1) reweighting the data so the current ensemble has low correlation, (2) finding a classifier at least weakly correlated with the reweighted data, (3) adding this new classifier to the ensemble. Weak learning in (2) is left for a separate algorithm to perform. The amount of data required by an ensemble can be controlled (Koltchinskii et al., 2003). Many boosting algorithms may be understood as convex optimization over ensembles, which is not robust to noise (Long and Servedio, 2010). By introducing nonconvexity, these algorithms become robust but computationally intractable (Freund, 2001; Hanbo Li and Bradic, 2015). 'Smooth' boosting limits the amount of reweighting; with a carefully chosen weak learner, such algorithms are robust to limited amounts of noise (Servedio, 2003; Klivans et al., 2009). General-purpose boosting algorithms merely assume that weak learning is always possible (Ben-David et al., 2001; Kanade and Kalai, 2009; Chen et al., 2015). This assumption is equivalent to a kind of linear separability, which is typically thought of as "non-realistic" (Shalev-Shwartz and Singer, 2010). In summary, boosting may be stymied by noise or may get stuck on insoluble subproblems.

```
For $t = 1, \ldots, T$:
    $w_t \leftarrow \frac{1}{m}\sum_{i=1}^m x_i y_i$
    $w_t \leftarrow \beta_t(w_t / ||w_t||)$
    $y_i \leftarrow e^{-|\langle w_t, x_i\rangle|} y_i$
Return $w_1, \ldots, w_T$
```

```
For $t = 1, \ldots, T$:
    With probability $1 - e^{-|\langle w_t, x\rangle|}$, return $\operatorname{sgn}(\langle w_t, x\rangle)$
Return $-1$ or $1$ uniformly at random.
```

Figure 1: The learning algorithm (left) operates upon data $\{x_i, y_i\}_{i=1}^m$ for T iterations, and computes a sequence of averages. It uses a sequence of positive step sizes $\{\beta_t\}_{t=1}^T$. On each iteration, it computes, rescales (by Euclidean norm $||\cdot||$), and stores the average of all the data. It reduces the weight of data having high inner product with the average. The weight reduction corresponds to a passing probability in the smooth list of halfspaces (right), which operates upon an input x. A stored average w_t is used to classify an input if they have high inner product; otherwise, the input is passed to the next average.

1.1 Our contributions

This paper revisits the following basic questions.

What classifiers should the algorithm return? Smooth lists of halfspaces are a novel generalization of halfspaces. They involve a sequence of halfspaces, each employed with probability depending on its confidence on the input; see section 2 for their definition. They are more flexible than halfspaces due to their unbounded depth; see figure 3 for an illustration of a complicated, nonlinear function which they can (weakly) fit.

How should the algorithm fit such a classifier to the data? The sequence of averages (SoA) is a hybrid formulation of learning as iterative optimization. It never revises previously added elements, much as decision trees are recursively constructed. A step size smoothly limits the modification of the list through the probability that the element is invoked, much as gradient descent smoothly updates parameters.

Our classifier and learning algorithm (defined in figure 1) blunt the compromises of improper learning. Even though smooth lists of halfspaces have unbounded depth, they do not require more data to learn (in the worst case) than halfspaces. That is, an amount of data large enough to bind together the training and true correlations of every n-dimensional halfspace also suffices for smooth lists of halfspaces.

Theorem 1 *Let the input distribution be absolutely continuous with respect to Lebesgue measure, and let $\mathcal{F}$ be all n-dimensional smooth lists.* $\mathbf{E}\sup_{f\in\mathcal{F}}(\chi(f) - \hat{\chi}(f)) \le \epsilon$ *with $m = O(n/\epsilon^2)$ data,.*

Since smooth lists strictly generalize halfspaces, this bound is tight. The following bound is more appropriate when the dimension n is large compared to the number of iterations, or the size of the steps therein.

Theorem 2 *Let $\mathcal{D}$ be supported on the unit ball, and let $\mathcal{F}_\beta$ be all smooth lists defined by vectors $w_1, \ldots, w_T$ satisfying $\sum_{t=1}^T ||w_t|| \le \beta$.* $\mathbf{E}\sup_{f\in\mathcal{F}_\beta}(\chi(f) - \hat{\chi}(f)) \le \epsilon$ *with $m = O(\beta^2/\epsilon^2)$ data.*

These theorems are reminiscent of previous results, which bound the additional complexity of later classifiers by the probability of actually reaching them (DeSalvo et al., 2015).

Because SoA is so simple, it may be expressed as a concrete (albeit nonsmooth) dynamical system (section 3). This yields insight about its convergence. Notice that, once the smooth list returns with probability 1 for every input, appending new elements has no effect. Such convergence occurs when every y_i is shrunk to zero by line 4 of SoA. It is not evident that SoA converges; it may 'get stuck' at iteration t if $w_t = 0$ yet $y_i > 0$ for some i. It is easy to see (in section 3) that, when $m > n$, there are infinitely many points at which this may occur. Such sticking points are analogous to pathological distributions in boosting. Fortunately (under reasonable smoothness conditions) SoA never leads to such points.

Theorem 3 *Let the input distribution be absolutely continuous with respect to Lebesgue measure, and scale the training inputs to have unit norm. In the training algorithm, replace the (nonsmooth) absolute value function with a smooth ϵ-approximation z, as defined in (4). Run the training algorithm for T iterations with any constant step size $\beta > 0$. As $\epsilon \to 0$ and $T \to \infty$, $y \to 0$ with probability 1 over the randomness of the training data.*

By analyzing an (arbitrarily minor) smoothing of SoA, we can apply powerful tools from dynamical systems which have also recently been applied to nonconvex optimization (Lee et al., 2016).

In experiments, SoA fulfills the original purpose of nonlinear classification: to quickly fit complicated, possibly noisy data. These experiments involve some of the most intensely studied, yet elusively challenging, problems in computational learning theory. Overall, it seems SoA achieves a better tradeoff among time, data, and robustness, in a restricted but significant regime: where only a moderate amount of time and data are available, yet the classifier must tolerate a non-trivial amount of inconsistent data.

2. Smooth lists of halfspaces

Begin by approximating the sign function $\operatorname{sgn}(\cdot)$ with a differentiable sigmoid function of slope at most β. This paper uses a sigmoid derived from the CDF of the Laplace distribution:

$$\psi(a) = \begin{cases} 1 - e^{-\beta a} & a \geq 0 \\ -1 + e^{\beta a}, & \text{otherwise} \end{cases} \qquad\qquad \begin{aligned} |\psi(a)| &= 1 - e^{-\beta|a|} \\ \psi'(a) &= \beta e^{-\beta|a|} = \beta(1 - |\psi(a)|) \end{aligned}$$

This function is numerically stable and twice differentiable. Despite its simplicity and numerical appeal, the Laplace sigmoid is rare in machine learning literature; (Clevert et al., 2015) uses it for negative inputs, and a linear function for positive inputs. Applying such an approximation to halfspaces yields smooth halfspaces:

$$\mathcal{H} = \{h_w(x) = \psi(\langle w, x\rangle) : w \in \mathbb{R}^n\}$$

The magnitude of the real-valued output, within $[-1, 1]$, can be used as a probability for a randomized classifier which operates as follows: "with probability $|h_w(x)|$, return $\operatorname{sgn}(\langle w, x\rangle)$. Otherwise guess -1 or 1 uniformly at random." The correlation of h_w equals the expected correlation of this randomized classifier. As $\beta \to \infty$, a halfspace is recovered, which we will denote as $h_{\infty w}$.

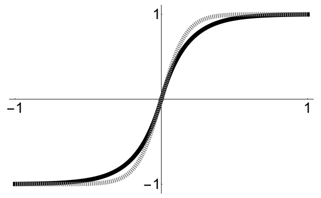

Figure 2: With the same maximum slope $\beta = 6$, the Laplace sigmoid (in black) and the logistic sigmoid (dashed). The former's singly-exponential tail is algebraically useful for proving the convergence of the algorithm in theorem 3.

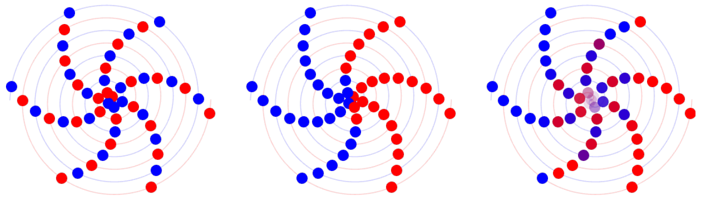

Figure 3: Points on two intertwined logarithmic spirals (left) of the form $r = e^{\theta}$. For visual clarity, the radii are plotted logarithmically. The best halfspace (middle) achieves correlation ≈ 0.07 from a negligible imbalance in the sampling of points. By contrast, a smooth list (right) achieves ≈ 0.27; lower opacity denotes lower return probability. Intuitively, outer rungs are exponentially farther from the origin than the inner ones. The smooth list can correctly classify the former with reasonable return probability, while essentially ignoring the latter. It then recurses on closer 'flipped' rungs.

A smooth list of halfspaces, defined by a list of vectors $w_1, \ldots, w_T$, operates as described on the right side of figure 1. It is a randomized classifier which has the same correlation as the following real-valued, recursively-defined function:

$$\begin{aligned} f_{\emptyset}(x) &= 0 \\ f_{w_1,\ldots,w_T}(x) &= h_{w_1}(x) + (1 - |h_{w_1}(x)|) f_{w_2,\ldots,w_T}(x) \end{aligned}$$

Clearly a smooth halfspace is a smooth list of length 1. Interestingly, the smooth halfspace defined by w can be decomposed into an arbitrary-length smooth list of halfspaces defined by scalings of w; the short proof is in the appendix.

Theorem 4 *$h_w = f_{\beta_1 w,\ldots,\beta_T w}$ for any $\beta_1, \ldots, \beta_T$ satisfying $\beta_t \geq 0$ and $\sum_{t=1}^{T} \beta_t = 1$.*

Smooth lists are related to previously proposed classifiers. For example, decision lists (Rivest, 1987) operate as follows: "For $t = 1, \ldots, T$: if the deterministic function $\pi_t(x) = 1$, return the fixed value $v_t \in [-1, 1]$. Return 1 at the end of the list." Decision lists are more expressive than smooth lists. For example, if π_t are halfspaces, decision lists are intersections of halfspaces, which cannot be represented as smooth lists. However, the complexity of decision lists grows with their length, which discourages appending many elements, and thereby constrains the algorithm. (This difficulty was partially overcome by the notable algorithm of Blum et al. (1998), as described in section 3.) As theorems 1 and 2 prove, the complexity of a smooth list is independent of its length.

If each $v_t = -1$, then a decision list is called a cascade of classifiers (Viola and Jones, 2001). These can be fast to evaluate in applications with imbalanced outputs, such as computer vision: obvious inputs are classified early, and further processing is reserved for the occasional output 1. Smooth lists behave similarly, even without imbalanced outputs: inputs far from decision boundaries tend to be classified earlier. However, we focus on the resources needed to train smooth lists, not evaluate them. If each $v_t > 0$ and $v_1 > v_2 > \ldots v_T$, then a decision list is called a falling rule list (Wang and Rudin, 2015). These are easily interpretable for a similar reason: obviously-true inputs must be classified early. Smooth lists may also be easily interpreted, since they tend to classify extreme inputs early. However, we focus on the algorithmic benefits of smooth lists, not their interpretability.

Smooth lists are reminiscent of other functions besides classifiers. Feedforward neural networks also involve a composition of linear functions and nonlinear activation functions. They typically transform the input x to another vector x'. For example, each layer of a cascade correlation network (Fahlman and Lebiere, 1990) computes a nonlinear function of the previous layer and copies through the input. Smooth lists lack the latter 'skip connections', in the parlance of modern deep learning (He et al., 2016). Smooth lists are more comparable to the final classification layer of a neural network, which is typically a linear classifier learned by relaxation. In general, hypotheses are often averaged according to a fixed probability distribution. Such combinations are called ensembles. Smooth lists are not ensembles, since the distribution of which classifier returns depends on the input x.

The statistical properties of smooth lists are different than previously proposed nonlinear classifiers, as the following sections describe.

2.1 On 'nice' data, smooth lists need more data than halfspaces

Since the distribution $\mathcal{D}$ is unknown, learning is performed on a sample of m data. This defines a training distribution $\hat{\mathcal{D}}$ and a training correlation $\hat{\chi}$:

$$\hat{\chi}(c) = \hat{\mathbf{E}}_{x,y}\left(c(x)y\right) = \frac{1}{m}\sum_{i=1}^{m} c(x_i)y_i \tag{2}$$

This section upper bounds the amount of data needed to ensure the training and true correlations are close for smooth lists of halfspaces. The standard approach, for any hypothesis class $\mathcal{F}$, is to bound the Rademacher complexity: the maximum correlation, over $\mathcal{F}$, with m uniformly random outputs $\sigma_1, \ldots, \sigma_m$ on m inputs $x_1, \ldots, x_m$ drawn from $\mathcal{D}$.

$$\mathcal{R}(\mathcal{F}) = \mathop{\mathbf{E}}_{\substack{x_1,\ldots,x_m\sim\mathcal{D}\\ \sigma_1,\ldots,\sigma_m\sim\{-1,1\}}}\left(\sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^{m} f(x_i)\sigma_i\right)$$

It seems the Rademacher complexity of smooth lists of halfspaces exceeds that of halfspaces when $\mathcal{D}$ is normally distributed. Figure 4 shows experimental evidence.

2.2 On worst-case data, smooth lists don't need more data than halfspaces

In most scenarios, however, $\mathcal{D}$ is not known, and one resorts to 'distribution-independent' bounds which hold over a family of $\mathcal{D}$. As reviewed in the appendix, the Rademacher complexity of smooth halfspaces $\mathcal{R}(\mathcal{H})$ may be bounded independently of the dimension or distribution of the inputs.

Theorem 5 *If $||x|| \leq 1$, then for any $\beta > 0$, $\sup_{\mathcal{D}} \mathcal{R}(\mathcal{H}) \leq \mathcal{R}_\beta := \beta\sqrt{2/m}$ (Kakade et al., 2008 Theorem 1 and Example 3.1.1).*

Theorem 2 is a straightforward consequence of the following.

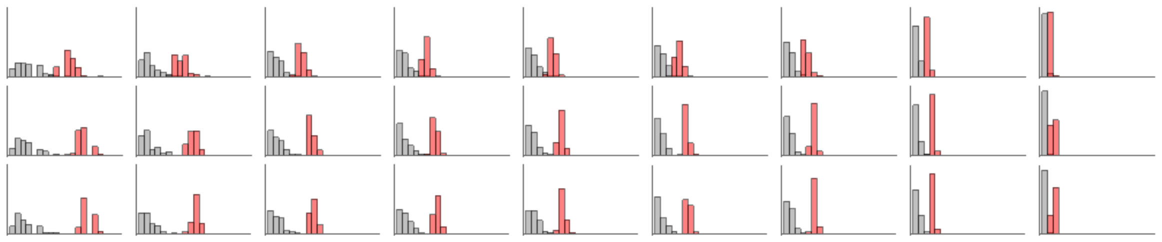

Figure 4: Estimated Rademacher complexities of halfspaces and thresholded smooth lists. Each histogram consists of 150 samples of the following process: r random classifiers of each kind are generated, m standard normal inputs in 128 dimensions are sampled, random outputs are assigned, and the maximum correlation obtained by each set of classifiers is returned; larger correlation is plotted further right. The rows increase r from 250 (top) to 25000 (bottom). The columns increase m from 32 (left) to 15336 (right). Since the 2nd and 3rd rows are similar, increasing r further should have no effect. For small m, the smooth lists achieve substantially higher correlation (i.e. demonstrate more potential for overfitting); this persists as m increases. All of these estimated Rademacher complexities are far below the worst-case bounds.

Theorem 6 *Under the conditions of theorem 2,* $\mathcal{R}(\mathcal{F}_\beta) \leq \mathcal{R}_\beta$.

Theorem 1 is analogously based on the distribution-independent Vapnik-Chervonenkis bound for halfspaces, which are denoted by $h_{\infty w}$ for normal vector w.

Theorem 7 *If* $m = O(n/\epsilon^2)$, *then* $\sup_{\mathcal{D}} \mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}} (\sup_w \chi(h_{\infty w}) - \hat{\chi}(h_{\infty w})) \leq \epsilon$ *(Boucheron et al., 2005, Theorem 3.4).*

The proofs of these results are in the appendix. The high-level insight is that an early element of the smooth list (say, w_1) interacts with a later element (say, w_2) only through the conditional distribution over inputs which it passes to w_2. The correlation of h_{w_2} on the conditional distribution is handled by the distribution-independent bound:

$$\sup_{w_1} \mathop{\mathbf{E}}_{\mathcal{D}|w_1 \text{ passes}} \left(\sup_{w_2} \chi(h_{w_2}) - \hat{\chi}(h_{w_2}) \right) \leq \sup_{\mathcal{D}_2} \mathop{\mathbf{E}}_{\mathcal{D}_2} \left(\sup_{w_2} \chi(h_{w_2}) - \hat{\chi}(h_{w_2}) \right)$$

We can apply the bound separately for each element without introducing dependence on T because the sum of norms is at most β (for theorem 2) or the total return probability is at most 1 (for theorem 1). Due to the previous section, it is not surprising that these proofs depend critically on distribution-independent bounds.

3. The sequence of averages

The algorithm (fig. 1) trains smooth lists the same way they are used. It appends a vector to the list, reweights the data by the probability they would pass to the next vector, and repeats. The reweighting in line 4 is superficially similar to AdaBoost or multiplicative weights, which are based on the (signed) margin: $y_i \propto e^{-\langle w_t, x_i\rangle y_i}$. Their reweighting is sensitive to noise: if

$|\langle w_t, x_i\rangle|$ is large, flipping the sign of y_i affects the reweighting substantially more than a linear classifier's correlation. By contrast, line 4 depends on the confidence $|\langle w_t, x_i\rangle|$ of w_t on x_i — not whether it was correctly classified. This helps explain the robustness observed in section 4.

Furthermore, parts of this algorithm are reminiscent of previous ones designed to resist noise. The algorithm of Klivans et al. (2009) computes the same average vector w_t at each iteration. However, it forms a combination of halfspaces via boosting rather than a smooth list. The algorithm of Blum et al. (1998) produces a decision list (as defined in section 2) of halfspaces defined by vectors $w_1, \ldots, w_T$. It returns $\operatorname{sgn}(\langle w_t, x\rangle)$ if $|\langle w_t, x\rangle|$ is larger than some threshold, and otherwise proceeds to the next element. It trains each halfspace with the perceptron algorithm on a subset of the data which would have (on average) large margin; it passes the remaining data to subsequent steps. By contrast, we smooth the 'return' event and pick a vector according to a simpler criterion.

Our algorithm may be viewed a sequence of locally optimal decisions of arbitrarily small impact. The training correlation, at iteration T, of the smooth halfspace h_w is:

$$\hat{\chi}_T(h_w) = \frac{1}{m}\sum_{i=1}^{m} h_w(x_i) \prod_{t=1}^{T-1} (1 - |h_{w_t}(x_i)|) y_i$$

Each average w_t is instantaneously optimal, assuming it is nonzero: for some $\beta > 0$, versus any competing direction v scaled to the same norm β_t, $\hat{\chi}(f_{w_1,\ldots,w_t}) \geq \hat{\chi}(f_{w_1,\ldots,v})$. This is because the average is the derivative of the correlation at the origin:

$$\chi'(0) = \frac{d}{dw}\chi(h_w)\bigg|_{w=0} = \mathbf{E}\left(\frac{d}{dw}\psi(\langle w, x\rangle) y\right)\bigg|_{w=0} = \mathbf{E}\left(\psi'(\langle w, x\rangle) x \cdot y\right)\big|_{w=0} = \mathbf{E}\left(x \cdot y\right)$$

Since w_t is the direction of instantaneous steepest ascent, by continuity, it maintains some advantage over all competing directions for some length. We can choose $\beta_t > 0$ so that $\hat{\chi}_t(h_{w_t}) > 0$ for all t. This makes the algorithm monotonically improving in that $\hat{\chi}(f_{t+1}) > \hat{\chi}(f_t)$.

Rather than explicitly computing the averages w_t by manipulating the inputs, it is possible to directly solve for the inner products used to classify and reweight data, in terms of the kernel matrix $K_{i,j} = \frac{1}{m}\langle x_i, x_j\rangle$. Let the norm of the sum be $G = ||\sum_i x_i \cdot y_i||$, g the unit vector in that direction, and $||y||_K$ the kernel seminorm:

$$\begin{aligned}
(Ky)_i &= \sum_j K_{i,j} y_j \\
G^2 &= \left\langle \sum_i y_i x_i, \sum_j y_j x_j \right\rangle = \sum_{i,j} y_i y_j K_{i,j} = y^T K y = ||y||_K^2 \\
\langle Gg, x_i\rangle &= \left\langle \sum_j y_j x_j, x_i \right\rangle = \sum_j y_j K_{i,j} = (Ky)_i \\
\langle g, x_i\rangle &= \langle Gg, x_i\rangle / G = (Ky)_i / ||y||_K
\end{aligned}$$

The dual update to y is a continuous function α:

$$y_i \leftarrow \alpha(y_i) := e^{-\beta_t |\langle g, x_i\rangle|} y_i = e^{-\beta_t |Ky|_i / ||y||_K} y_i \tag{3}$$

As a convention, multiply the sign of y_i into x_i, so each y_i is initially 1. This does not affect the correlation of any linear classifier, nor the convergence of y to 0.

3.1 Convergence

This section outlines why smoothed SoA converges, per theorem 3; details appear in section 6.5 y is initially the all-ones vector; each component decreases monotonically and must remain nonnegative. So, y converges to some vector, and that vector must be nonnegative. If y converges to anything, we must have $\alpha(y) = y$ by continuity, i.e., we must be at a fixed point. We say $\tilde{y}$ is stuck if it is a nonzero fixed point. Equivalently:

Definition 8 *$\tilde{y}$ is called a stuck point if $||\tilde{y}||_K = 0$ but $\tilde{y} \neq 0$.*

Let ϕ be a smooth map that closely approximates α and shares the exact same stuck points $\tilde{y}$. Unlike α, ϕ is a diffeomorphism locally at $\tilde{y}$. Using the stable manifold theorem, we show the 'bad' manifold, which contains nearby points that converge to $\tilde{y}$, is null (has measure zero). Its preimage is also null; iterating, the corresponding 'bad' initializations are null. The only remaining possibility is convergence to 0.

The strategy of the proof is similar to that of (Lee et al., 2016), but the technical details differ considerably. Their map is smooth and has a well-defined inverse over $\mathbb{R}^m$, which allows immediate application of the stable manifold theorem. By contrast, even after smoothing α to ϕ, it isn't invertible, so we identify the most general conditions which enable the 'iterated preimage' strategy. Finally, we must handle some technical complications relating to strictly positive y and uncountably many $\tilde{y}$.

I denotes the set of initial points which converge to $\tilde{y}$, also known as the basin of attraction of $\tilde{y}$.

$$I(\tilde{y}) = \left\{ y_0 \in \mathbb{R}^m : \lim_{t \to \infty} \alpha^t(y_0) = \tilde{y} \right\}$$

It is easier to analyze convergence of smooth maps, so we approximate the absolute value function with a smooth function parametrized by $L > 0$:

$$z(a) = \frac{1}{L} \log \left(\frac{1}{2} \left(e^{-La} + e^{La} \right) \right) \tag{4}$$

For all $a \in \mathbb{R}$, this function satisfies $z(a) \leq |a| \leq z(a) + \frac{1}{L} \log 2$. Accordingly define a smooth analogue of α.

Lemma 9 *Smoothness: for all $L > 0$,*

$$\phi(y) = e^{-\beta z(Ky)/||y||_K} \circ y$$

(where $\circ$ denotes entrywise product) is continuously differentiable.

(The lemmas in this section are proved in the appendix.) Let I_L replace α with ϕ in the definition of I. The singular points of ϕ determine if the preimage of any null set is null.

Theorem 10 *For continuously differentiable ϕ, the set of singular points*

$$S = \{y : \phi'(y) \text{ is not invertible}\}$$

is null iff the preimage $\{y_0 : \phi(y_0) \in Z\}$ of every null set Z is null. (Ponomarev, 1987, theorem 1).

The following result validates the proof strategy.

Lemma 11 *Almost diffeomorphism: for all $L > 0$, S is null.*

Outside of S, for sufficiently large L, ϕ is a C^1 local diffeomorphism: within some ball B around y, $\phi(Y)$ is open, and $\phi : B \to \phi(B)$ is continuously differentiable, invertible with a continuously differentiable inverse (by the inverse function theorem, since ϕ' is invertible), and bijective (since the inverse is defined at every $\phi(y)$). The next lemma shows all of the stuck points are outside of S. Its proof involves a slight technical complication related to strict positivity of y.

Lemma 12 *Local diffeomorphism: for each strictly positive stuck point $\tilde{y}$, there is an $L_0(\tilde{y}) > 0$ such that for all $L > L_0$, $\phi'(\tilde{y})$ is invertible.*

The stable manifold theorem describes the (bad) manifolds of convergence around stuck points.

Theorem 13 *Let $\tilde{y}$ be a fixed point of the C^1 local diffeomorphism $\phi : \mathbb{R}^m \to \mathbb{R}^m$. Let $\tilde{m}$ be the dimension of the span of the eigenvectors of $\phi'(\tilde{y})$ whose corresponding eigenvalues are at most 1 in absolute value. The center stable manifold W is an embedded C^1 disk* [1] *of dimension at most $\tilde{m}$. For some open ball B around $\tilde{y}$ and some constant $k < 1$: (1) $\phi(W) \cap B \subset W$, and (2) $\phi^t(y) \in B$ for all $t \geq 0$ only if $y \in W$ (which implies $I_L(\tilde{y}) \cap B \subset W$.) That is, under the action of ϕ, for some distance $\epsilon > 0$ from $\tilde{y}$: (1) all y in W which become ϵ-close remain in W, and (2) all ϵ-close y which remain ϵ-close are in W (which implies all trajectories converging to $\tilde{y}$ must enter W). (Shub et al., 1987, Theorem III.7, page 65)*

Each stuck point $\tilde{y}$ is unstable (having $\tilde{m} < m$), and therefore its manifold of convergence is null:

Lemma 14 *Instability: at every stuck $\tilde{y}$ and $L > 0$, $\phi'(\tilde{y})$ has an eigenvalue with absolute value strictly greater than 1.*

This implies the smooth map does not get stuck.

Lemma 15 *Not stuck here: at each stuck $\tilde{y}$, for all $L > L_0(\tilde{y})$, $I_L(\tilde{y})$ has probability 0.*

Extending this result to uncountably infinite stuck points requires a covering argument.

Lemma 16 *Not stuck anywhere: $\cup_{\text{stuck } \tilde{y}} \cup_{L > L_0(\tilde{y})} I_L(\tilde{y})$ has probability 0.*

Finally, not getting stuck implies convergence; this completes the proof of theorem 3.

Lemma 17 *Strict decrease: $\phi^t(y)$ strictly decreases to 0 with probability 1.*

Since the smooth map doesn't get stuck for sufficiently large L, it may seem obvious that the algorithm doesn't get stuck either. However, extending results about smooth maps to nonsmooth maps is technically subtle; we leave this extension for future work.

1. A set which is C^1-diffeomorphic to the unit Euclidean ball.

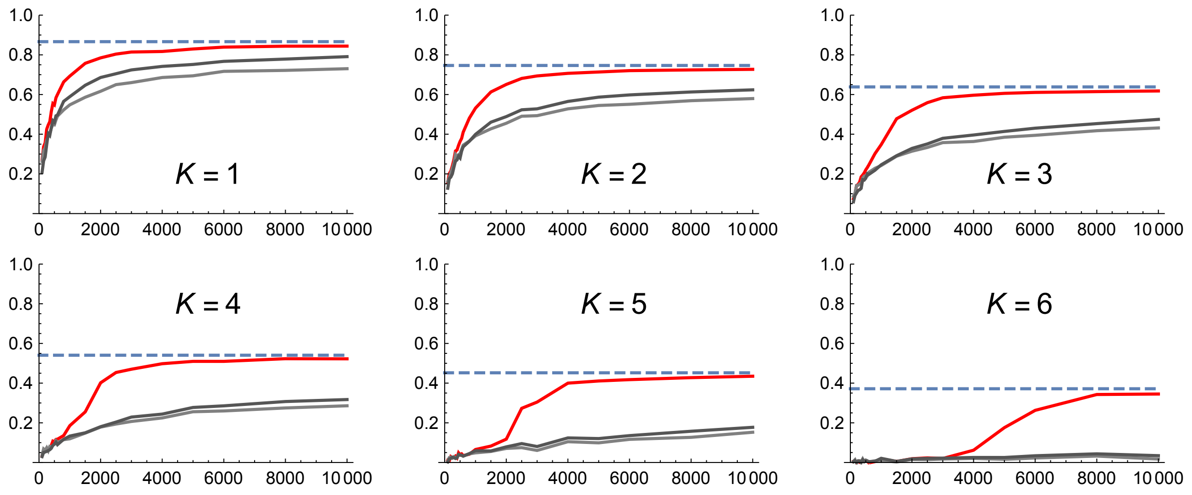

Figure 5: For different noise levels increasing with k, correlation (vertical) varying with the amount of data (horizontal). SVM (Pegasos), ℓ_2-regularized logistic regression (LIBLINEAR), and SoA, relative to the analytically-calculated correlation of the optimal halfspace.

4. Experiments

These experiments evaluate SoA's ability to fit noisy or complicated data. This requires efficiently coping with inconsistency while avoiding overfitting. As we shall see, SoA is the only algorithm which does both, and thus avoids the compromises of linear classifiers and previous nonlinear classifiers.

4.1 Increasing margin noise for linear classifiers

It may be tempting to think of noise as a theoretical nuisance which only hurts the performance of classifiers in complicated situations. This experiment dispels that misconception: very simple noise can utterly foil algorithms based on convex relaxation, whereas SoA copes gracefully. The inputs are standard normals of dimension $n = 128$. The outputs are generated by a halfspace $h_{\infty w}$ and flipped with probability increasing with their unsigned margin (their distance from the decision boundary).

$$y_i = \operatorname{sgn}(\langle w, x_i \rangle) \cdot \begin{cases} 1 & \text{with probability } e^{-k\sqrt{n}\left|\left\langle \frac{w}{||w||}, x_i \right\rangle\right|} \\ -1 & \text{otherwise} \end{cases}$$

In high dimension, the overwhelming majority of points have small margin, so the parameter k exponentially increases the flip probability. A simple integral calculates the correlation of the initial halfspace, labeled in blue. Relaxation cannot cope with higher levels of noise. By contrast, SoA converges to the calculated optimum. SoA's ability to cope with this sort of noise is not novel, but is rather a prerequisite to the complications of the next experiment.

4.2 Juntas and $\cap$ halfspaces

This experiment shows SoA can (weakly) fit very challenging functions without overfitting. Juntas are boolean functions $\{-1,1\}^n \to \{-1,1\}$ that depend on only $k < n$ input coordinates. Since a junta is expressible as $\operatorname{sgn}(p(x))$, where p is a degree-k polynomial, it is possible to fit them with lifting (or simply exhaustive search) in $n^{O(k)}$ time. The challenge is to fit lower-dimensional classifiers while avoiding computational intractability. In the presence of noise, this is considered the most fundamental (and vexingly unresolved) problem in learning theory (Mossel et al., 2003). Section 6.6 briefly reviews juntas and their various difficulties. Intuitively, they are difficult because they don't have local structure which may be exploited by, say, a well-architected neural network. It is challenging to devise good features besides degree-k products; lower-order algorithms must be sensitive to tiny correlations. (Intersections of halfspaces are not juntas, but are difficult in the same way, so they are included as well.)

4.2.1 Experiment design

SoA is compared to practical algorithms which fit linear classifiers. As observed in the first experiment, fitting them by convex relaxation is susceptible to inconsistent outputs. However, when the inputs are nicely distributed, some algorithms can cope with limited inconsistency. For example, iteratively filtering outliers via "localization" works when the inputs have logconcave distribution (Awasthi et al., 2016). The following two randomized algorithms are practical and achieve state-of-the-art guarantees:

- Least-squares initialization (LSQI): on a small subsample, guess labels for the dataset (which will be correct with some exponentially small probability), fit a least-squares vector to them, and use it as initialization for a first-order method (Zhang et al., 2015).
- Stochastic gradient Langevin dynamics (SGLD), which is stochastic gradient descent with step size η and Gaussian noise with variance proportional to $1/\eta$ (Zhang et al., 2017).

All three of these algorithms are unbiased. However, conjunctions and intersections of halfspaces are very biased: the number of positive examples drops exponentially with k. On the other juntas, we also compare to the two following deep classifiers:

- XGBoost, a large-scale implementation of boosted decision trees that is a mainstay of machine learning competitions (Chen and Guestrin, 2016).
- A multilayer perceptron (MLP) consisting of $10n$ hidden units, followed by a ReLu nonlinearity, fully connected to an output layer followed by a `tanh` sigmoid. It is trained by minimizing hinge loss with a stochastic first-order method. It incorporates a dropout probability of 0.1.

For conjunctions and parities, the average vector $\mathbf{E}\,(x \cdot y)$ is zero; as described in section 5, this halts first-order methods as well as SoA. We break symmetry by partitioning the inputs according to a uniformly random halfspace, and then run the training algorithms separately on each partition.

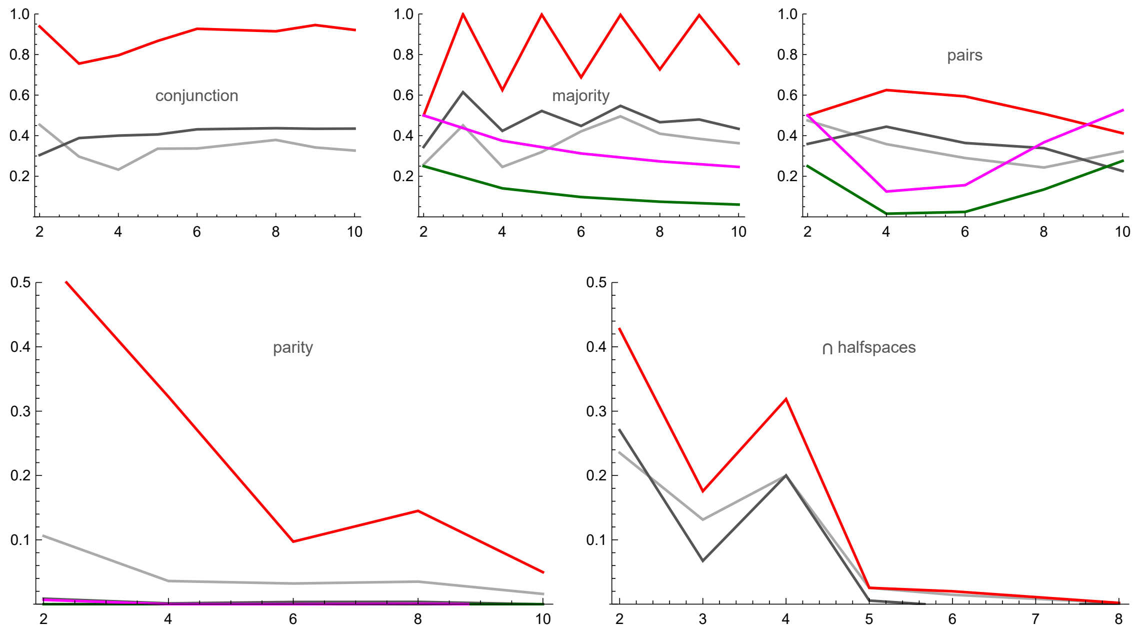

Figure 6: On low-dimensional ($n = 12$) data, correlation (vertical) varying with the complexity parameter k (horizontal), for LSQI, SGLD, SoA, and (on the unbiased data), XGBoost and MLP. For majority, when k is odd, the outputs are slightly imbalanced.

Low-dimensional ($n = 12$) experiments with all $m = 2^n$ training inputs assess the fitting capacity of the algorithms (fig. 6). Higher-dimensional experiments ($n = 64$) with just $m = 2048$ training inputs assess susceptibility to overfitting and asymmetry (fig. 7). All inputs are sampled uniformly from $\{-1, 1\}^n$ and the outputs have no noise. k will vary between 2 to 10 to assess how the algorithms cope with increasingly complex functions. Each experiment is averaged over 8 runs.

4.2.2 Results

On the easier juntas (conjunctions, majority, and pairs), none of the methods seemed particularly susceptible to overfitting. For each algorithm, the train-true deviation seems proportional to its true correlation. MLP's test correlation is sometimes higher than its trainin due to the randomness of dropout. On parity or intersections of halfspaces, SoA and SGLD seems to overfit for $k \geq 4$. The next experiment sees how much data is needed to ameliorate this.

The algorithms distinguish themselves by their capacity to fit. Despite employing deep, expressive classifiers, XGBoost and MLP failed. For larger k, pairs becomes imbalanced, so XGBoost and MLP 'unfairly' improve. SGLD generally outperformed LSQI. SoA consistently demonstrated the most capacity, particularly on conjunctions. This performance, though notable, does not insinuate that it learns juntas, or that doing so is even possible. Overall, the experiments suggest some juntas are easier than others.

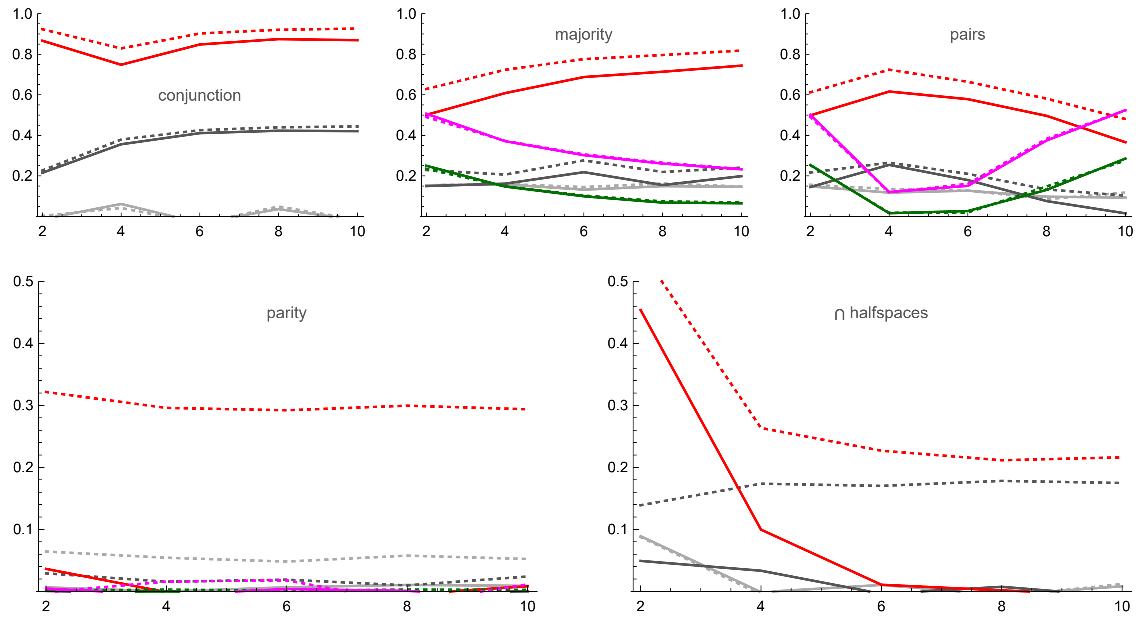

Figure 7: On high-dimensional ($n = 64$) data, correlation (vertical) varying with the complexity parameter k (horizontal), for LSQI, SGLD, SoA, and (on the unbiased data), XGBoost and MLP. Training and true correlations are dotted and solid, respectively. The majority plot is smooth because the output balance does not manifest in the training data.

We did not record training time, but generally observed that XGBoost was the fastest (due to its high-quality implementation), followed by SoA and MLP. SGLD and LSQI were substantially slower due to the large number of stochastic gradient steps and random initializations, respectively.

4.3 Revisiting parity

In the previous experiment, SoA is the only method which achieves nontrivial correlation with parity. However, the train-test deviation is extreme. This leaves open the possibility that, with enough training data, SoA can achieve nontrivial true correlation. This experiment shows SoA succeeds in doing so, whereas relaxation (still) fails and lifting needs too much time and data. This experiment adopts the setup of (Klivans and Kothari, 2014). The inputs are standard normals of varying extrinsic dimension n. The outputs are generated by a parity of fixed size $k = 3$; that is, the product of the sign of k coordinates. Relaxation is no better than random guessing. Kernel SVM with a degree-$(k + 1)$ polynomial is reliable because it subsumes the degree-k parity function. However, as the extrinsic dimension increases, it requires an overwhelming amount of training data. SoA uses a modicum of data and reliably achieves a nontrivial correlation. This does not insinuate that SoA learns parities, which would require maintaining correlation as k increases.

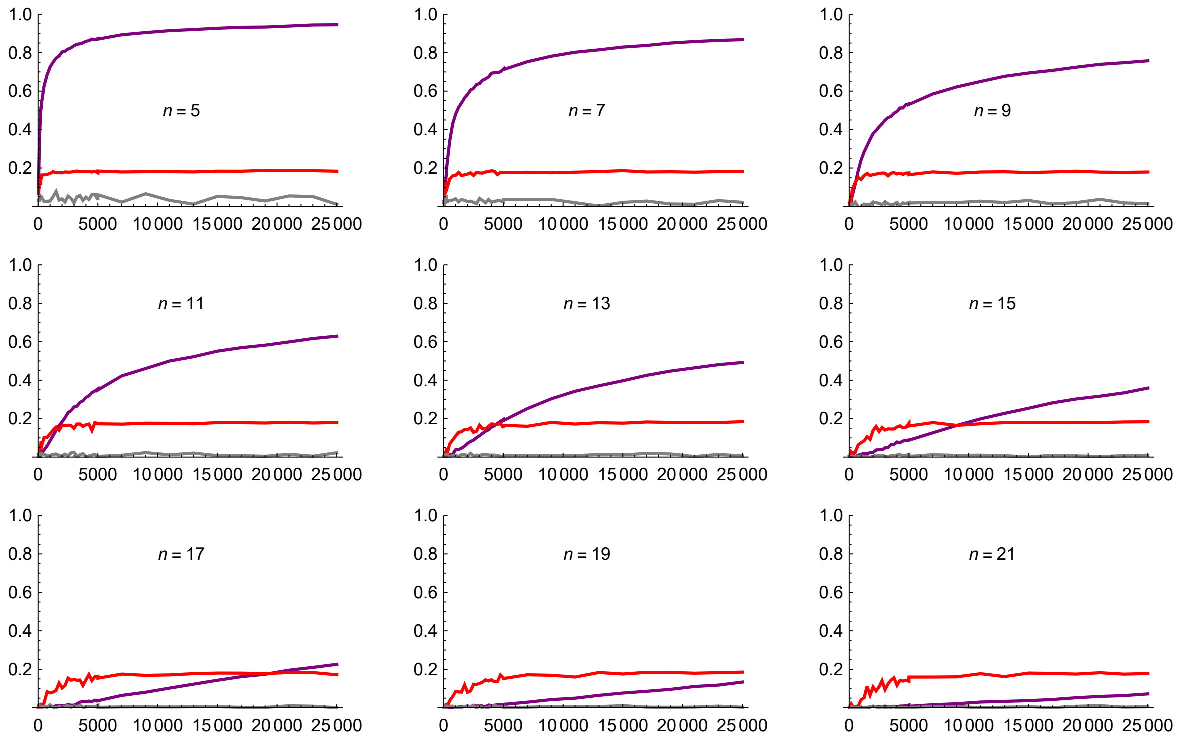

Figure 8: Correlation (vertical) varying with the amount of training data (horizontal) for SVM (LIBLINEAR), degree-4 polynomial SVM (LIBSVM), and SoA.

5. Conclusion

This paper proposed a new approach to classification. It is based on new classifiers (smooth lists of halfspaces) and a new iterative learning algorithm (the sequence of averages). We theoretically examined it with basic statistical tools (based on VC dimension and Rademacher complexity) and more involved algorithmic tools (from dynamical systems). We hope the algorithm's qualitative novelty and quantitative performance motivate further research of the following questions.

Are smooth lists provably worthwhile? Enlarging the set of classifiers beyond linear classifiers gives the training algorithm some flexibility to avoid computational intractability. SoA is experimentally superior to state-of-the-art linear classifiers, but the latter might be improvable. Is there data which provably can be fit by SoA, but cannot be efficiently fit by any linear classifier? Conjunctions are a possible candidate.

When does SoA yield good classifiers? Theorem 3 guarantees the classifier is not prematurely truncated, but does not lower bound its correlation, nor does it upper bound its depth. Due to the relationship with smooth halfspaces elucidated in theorem 2, we wonder if, with a reasonable amount of time and data, the classifier's correlation exceeds that of the best smooth halfspace. This would require restrictions on the distribution of inputs and outputs, since agnostically learning smooth halfspaces is conjectured to be computationally intractable (Shalev-Shwartz et al., 2011; Daniely, 2016).

Are alternatives to averaging useful? The average vector is fast to compute and allows the dual updates to be expressed as a continuous map. However, the corresponding halfspace may have poor correlation, even if the inputs are normally distributed (Awasthi et al., 2015 theorem 5). It is worth considering alternatives to the average, but some natural choices have caveats. First-order maximization, which yields a local maximum of the correlation, is inadvisable, since it causes the algorithm to immediately get stuck. If w_1 is a critical point of the correlation, then the average vector is subsequently zero:

$$0 = \frac{d}{dw}\chi(h_w)\bigg|_{w=w_1} = \mathbf{E}\left(e^{-|\langle w_1, x\rangle|} x \cdot y\right) = w_2$$

Relaxation (minimizing a convex loss ℓ) makes progress only if the average does. That is, if the average is zero, then all relaxations fail in the sense that the minimizer of ℓ is always zero. To contradict the existence of a solution w with lower loss, apply convexity twice:

$$\begin{aligned} 0 &< \ell(0) - \mathbf{E}\left(\ell(\langle w, x\rangle y)\right) \leq \ell(0) - \ell(\mathbf{E}\left(\langle w, x\rangle y\right)) \\ &\leq -\ell'(0)\mathbf{E}\left(\langle w, x\rangle y\right) = -\ell'(0)0 \end{aligned}$$

The first inequality is the assumption to be contradicted; the second is the zero-order definition of convexity; the third is the first-order definition of convexity. Substituting the zero joint average yields the contradiction.

How should learning be formulated as iterative optimization? There are two usual answers:

1. with classifiers of fixed depth (such as support vector machines, neural networks, and boosted ensembles) whose parameters are iteratively optimized (via, e.g., gradient descent) to maximize correlation. Each iteration may have arbitrarily small size. This smoothness confers many advantages: for example, each iteration may involve only a fraction of the data, as in stochastic gradient descent.

2. with classifiers of growing depth (such as decision trees and lists) trained in discrete, effectively irreversible steps. Many algorithms for training decision trees are consistent in the sense that, as the amount of data increases to infinity, along with the depth of the tree, the correlation of c approaches the maximum achievable by any classifier. However, trees of large depth may not be practical, since their size may be exponential in their depth.

We have proposed a classifier and algorithm which combine aspects of both: it has unbounded depth and never revises previously added layers; however, it is smoothly trained in steps of arbitrary granularity. We hope to understand how this qualitative difference affects fundamental tradeoffs in learning.

References

P. Awasthi, M.-F. Balcan, N. Haghtalab, and R. Urner. Efficient learning of linear separators under bounded noise. In *Proceedings of the 28th Conference on Learning Theory*, volume 40, pages 167–190. JMLR, 2015. URL `http://jmlr.org/proceedings/papers/v40/Awasthi15b.html`.

Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Hongyang Zhang. Learning and 1-bit compressed sensing under asymmetric noise. In *Proceedings of the 29th Annual Conference on Learning Theory (COLT)*, 2016.

Shai Ben-David, Philip M Long, and Yishay Mansour. Agnostic boosting. In *Computational Learning Theory*, pages 507–516. Springer, 2001.

Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 408–416. IEEE, 1985.

Yoshua Bengio, Olivier Delalleau, and Clarence Simard. Decision trees do not generalize to new variations. *Computational Intelligence*, 26(4):449–467, 2010. ISSN 1467-8640. doi: 10.1111/j.1467-8640.2010.00366.x. URL `http://dx.doi.org/10.1111/j.1467-8640.2010.00366.x`.

A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1):35–52, 1998. ISSN 1432-0541. doi: 10.1007/PL00013833. URL `http://dx.doi.org/10.1007/PL00013833`.

Stephane Boucheron, Olivier Bousquet, and Gabor Lugosi. Theory of classification: a survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 11 2005. ISSN 1262-3318. doi: 10.1051/ps:2005018.

Mark Bun and Thomas Steinke. Weighted Polynomial Approximations: Limits for Learning and Pseudorandomness. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 625–644, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-89-7. doi: http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2015.625. URL `http://drops.dagstuhl.de/opus/volltexte/2015/5327`.

Shang-Tse Chen, Maria-Florina Balcan, and Duen Horng Chau. Communication efficient distributed agnostic boosting. *CoRR*, abs/1506.06318, 2015. URL `http://arxiv.org/abs/1506.06318`.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Dana Dachman-Soled, Vitaly Feldman, Li-Yang Tan, Andrew Wan, and Karl Wimmer. Approximate resilience, monotonicity, and the complexity of agnostic learning. *CoRR*, abs/1405.5268, 2014. URL `http://arxiv.org/abs/1405.5268`.

A. Daniely. A PTAS for Agnostically Learning Halfspaces. *ArXiv e-prints*, October 2014.

A. Daniely, N. Linial, and S. Shalev-Shwartz. The complexity of learning halfspaces using generalized linear methods. *ArXiv e-prints*, November 2012.

Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 105–117, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4132-5. doi: 10.1145/2897518.2897520. URL `http://doi.acm.org/10.1145/2897518.2897520`.

Giulia DeSalvo, Mehryar Mohri, and Umar Syed. Learning with deep cascades. In *Proceedings of the Twenty-Sixth International Conference on Algorithmic Learning Theory (ALT 2015)*, 2015.

Simon S. Du, Chi Jin, Jason D. Lee, Michael I. Jordan, Barnabás Póczos, and Aarti Singh. Gradient descent can take exponential time to escape saddle points. *CoRR*, abs/1705.10412, 2017. URL `http://arxiv.org/abs/1705.10412`.

Scott E. Fahlman and Christian Lebiere. Advances in neural information processing systems 2. chapter The Cascade-correlation Learning Architecture, pages 524–532. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-100-7. URL `http://dl.acm.org/citation.cfm?id=109230.107380`.

Vitaly Feldman and Pravesh Kothari. Agnostic learning of disjunctions on symmetric distributions. *Journal of Machine Learning Research*, 16:3455–3467, 2015.

Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM Journal on Computing*, 39 (2):606–645, 2009.

Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012.

Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine learning*, 43 (3):293–318, 2001.

A. Hanbo Li and J. Bradic. Boosting in the presence of outliers: adaptive classification with non-convex loss functions. *ArXiv e-prints*, October 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1724–1732, 2017. URL `http://proceedings.mlr.press/v70/jin17a.html`.

Sham M. Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 793–800. Curran Associates, Inc., 2008.

Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008.

Varun Kanade and Adam Kalai. Potential-based agnostic boosting. In *Advances in Neural Information Processing Systems 22*, pages 880–888. 2009. URL `http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips22/NIPS2009_0346.pdf`.

Daniel Kane. The average sensitivity of an intersection of half spaces. *Research in the Mathematical Sciences*, 1(1):13, 2014.

G. Karolina Dziugaite and D. M. Roy. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. *ArXiv e-prints*, March 2017.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

Adam Klivans and Pravesh Kothari. Embedding hard learning problems into gaussian space. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:63, 2014.

Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 553–562. IEEE, 2006.

Adam R Klivans, Ryan O'Donnell, and Rocco A Servedio. Learning intersections and thresholds of halfspaces. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 177–186. IEEE, 2002.

Adam R Klivans, Philip M Long, and Rocco A Servedio. Learning halfspaces with malicious noise. *The Journal of Machine Learning Research*, 10:2715–2740, 2009.

Vladimir Koltchinskii, Dmitriy Panchenko, and Fernando Lozano. Bounding the generalization error of convex combinations of classifiers: balancing the dimensionality and the margins. *Annals of Applied Probability*, pages 213–252, 2003.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 1246–1257, 2016. URL `http://jmlr.org/proceedings/papers/v49/lee16.html`.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Mach. Learn.*, 78(3):287–304, March 2010. ISSN 0885-6125. doi: 10.1007/s10994-009-5165-z. URL `http://dx.doi.org/10.1007/s10994-009-5165-z`.

Elchanan Mossel and Ryan O'Donnell. On the noise sensitivity of monotone functions. In *Mathematics and Computer Science II*, pages 481–495. Springer, 2002.

Elchanan Mossel, Ryan O'Donnell, and Rocco P. Servedio. Learning juntas. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 206–212, New York, NY, USA, 2003. ACM. ISBN 1-58113-674-9. doi: 10.1145/780542.780574. URL `http://doi.acm.org/10.1145/780542.780574`.

Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 21–30. IEEE, 2005.

I. Panageas and G. Piliouras. Gradient Descent Only Converges to Minimizers: Non-Isolated Critical Points and Invariant Regions. *ArXiv e-prints*, May 2016.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, pages 1–17.

S. P. Ponomarev. Submersions and preimages of sets of measure zero. *Siberian Mathematical Journal*, 28(1):153–163, 1987. ISSN 1573-9260. doi: 10.1007/BF00970225. URL `http://dx.doi.org/10.1007/BF00970225`.

Ronald L. Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

Clayton Scott and Robert D Nowak. Minimax-optimal classification with dyadic decision trees. *IEEE transactions on information theory*, 52(4):1335–1353, 2006.

Rocco A. Servedio. Smooth boosting and learning with malicious noise. *J. Mach. Learn. Res.*, 4:633–648, December 2003. ISSN 1532-4435. doi: 10.1162/153244304773936072. URL `http://dx.doi.org/10.1162/153244304773936072`.

Shai Shalev-Shwartz and Yoram Singer. On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. *Machine learning*, 80(2-3): 141–163, 2010.

Shai Shalev-Shwartz, Ohad Shamir, and Karthik Sridharan. Learning kernel-based halfspaces with the 0-1 loss. *SIAM J. Comput.*, 40(6):1623–1646, 2011.

Alexander A Sherstov. Optimal bounds for sign-representing the intersection of two halfspaces by polynomials. *Combinatorica*, 33(1):73–96, 2013.

M. Shub, A. Fathi, and R. Langevin. *Global stability of dynamical systems*. Springer-Verlag, 1987. ISBN 9780387962955. URL `https://books.google.com/books?id=KFLvAAAAMAAJ`.

Mahdi Soltanolkotabi, Adel Javanmard, and Jason D. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *CoRR*, abs/1707.04926, 2017. URL `http://arxiv.org/abs/1707.04926`.

Paul A. Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001. doi: 10.1109/CVPR.2001.990517.

Fulton Wang and Cynthia Rudin. Falling rule lists. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 1013–1022, 2015.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Yuchen Zhang, Jason D. Lee, Martin J. Wainwright, and Michael I. Jordan. Learning halfspaces and neural networks with random initialization. *CoRR*, abs/1511.07948, 2015. URL `http://arxiv.org/abs/1511.07948`.

Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. *CoRR*, abs/1702.05575, 2017. URL `http://arxiv.org/abs/1702.05575`.

6. Appendix

6.1 Proof of theorem 4

Assume $a = \langle w, x\rangle \geq 0$; the case $a < 0$ is symmetric.

$$\begin{aligned}
&f_{\beta w,(1-\beta)w}(x) \\
&= (1 - e^{-\beta a}) + (1 - (1 - e^{-\beta a}))(1 - e^{-(1-\beta)a}) \\
&= 1 - e^{-\beta a} + e^{-\beta a}(1 - e^{-(1-\beta)a}) \\
&= 1 - e^{-\beta a-(1-\beta)a} = h_w(x)
\end{aligned}$$

6.2 VC and Rademacher complexity review

Recall the Vapnik-Chervonenkis bound for halfspaces, which is denoted by $h_{\infty w}$ for normal vector w.

Theorem 18 *If* $m = O(n/\epsilon^2)$, *then* $\sup_{\mathcal{D}} \underset{\hat{\mathcal{D}}\sim\mathcal{D}}{\mathbf{E}} \left(\sup_w \chi(h_{\infty w}) - \hat{\chi}(h_{\infty w})\right) \leq \epsilon$ *(Boucheron et al., 2005, Theorem 3.4).*

Bounding the Rademacher complexity bounds the required amount of data.

Theorem 19 *With probability* $1 - \delta$ *over the sample* $\hat{\mathcal{D}} \sim \mathcal{D}$:

$$m \geq \frac{8\log(1/\delta)}{(4\mathcal{R}(\mathcal{F}) - \epsilon)^2} \quad \Longrightarrow \quad \sup_{f\in\mathcal{F}} |\chi(f) - \hat{\chi}(f)| \leq \epsilon$$

(Boucheron et al. (2005) Theorem 3.2).

6.3 Proof of theorem 2

Proof by induction on the list length T. In the base case $T = 2$, let $||w_1|| = \beta_1$ and $||w_2|| = \beta - \beta_1$. The Rademacher complexity $\mathcal{R}(\mathcal{F}_\beta)$ is $\underset{\hat{\mathcal{D}},\sigma}{\mathbf{E}}\left(\hat{\mathcal{R}}(\mathcal{F}_\beta)\right)$, the expectation of the training Rademacher complexity:

$$\begin{aligned}
\hat{\mathcal{R}}(\mathcal{F}_\beta) &= \sup_{w_1,w_2} \underset{x}{\hat{\mathbf{E}}}\left(h_{w_1}(x)\sigma_x + (1 - |h_{w_1}(x)|)h_{w_2}(x)\sigma_x\right) \\
&\leq \sup_{w_1} \underset{x}{\hat{\mathbf{E}}}\left(h_{w_1}(x)\sigma_x\right) + \sup_{w_1,w_2} \underset{x}{\hat{\mathbf{E}}}\left((1 - |h_{w_1}(x)|)h_{w_2}(x)\sigma_x\right) \\
&= \sup_{w_1} \underset{x}{\hat{\mathbf{E}}}\left(h_{w_1}(x)\sigma_x\right) + \sup_{w_1,w_2} \hat{\mathbf{P}}\left(w_1 \text{ passes}\right) \underset{x}{\hat{\mathbf{E}}}\left(h_{w_2}(x)\sigma_x \mid w_1 \text{ passes}\right) \\
&\leq \sup_{w_1} \underset{x}{\hat{\mathbf{E}}}\left(h_{w_1}(x)\sigma_x\right) + \sup_{w_1,w_2} \underset{x}{\hat{\mathbf{E}}}\left(h_{w_2}(x)\sigma_x \mid w_1 \text{ passes}\right)
\end{aligned}$$

The first inequality is the triangle inequality. w_1 affects the second correlation only through reweighting of the outputs, which is equivalently written in terms of conditional expectation. Conditioning the probability of input x on the event 'w_1 passes' means multiplying the probability of x by $1 - |h_{w_1}(x)|$, the probability that w_1 passes on x, and normalizing by the overall pass probability $\underset{x}{\hat{\mathbf{E}}}\left(1 - |h_{w_1}(x)|\right) = \hat{\mathbf{P}}\left(w_1 \text{ passes}\right)$. The final inequality drops the pass probability of at most 1. To bound the Rademacher complexity, apply the worst-case bound theorem 5 to the different distributions:

$$\mathcal{R}(\mathcal{F}_\beta) \le \mathcal{R}_{\beta_1} + \mathcal{R}_{\beta-\beta_1} = \mathcal{R}_\beta$$

This bound seems crudely pessimistic. Intuitively, a smooth halfspace at the end of a list has lower complexity than an independent one, but the bound does not reflect this. Algebraically, later correlations have decreasing weight (in the lower-magnitude outputs), but the proof does not exploit this. Without restrictions on the distribution of inputs, this result is tight even for $T = 1$. Furthermore, we have already seen numerical evidence against substantially tighter bounds, even when the inputs are normally distributed.

6.4 Proof of theorem 1

Let $\mathcal{D}_t$ be the conditional distribution of inputs that pass to element t in the list. (That is, elements $w_1, \ldots w_{t-1}$ do not return.) Let χ_t be the corresponding correlation, so we may decompose the correlation of the smooth list as correlations of smooth halfspaces.

$$\mathcal{D}_t = \mathcal{D} \mid \left(\bigwedge_{i=1}^{t-1} w_i \text{ passes} \right)$$

$$\chi(f_T) = \sum_t \mathbf{P}\left(\bigwedge_{i=1}^{t-1} w_i \text{ passes} \right) \chi_t(h_{w_t})$$

Let 'use w_t' mean the first $t-1$ elements of the list pass on the input to element t, which returns. Then we can decompose the smooth list correlation as conditional correlations over halfspaces.

$$\text{use } w_t = \left(\bigwedge_{i=1}^{t-1} w_i \text{ passes} \right) \wedge w_t \text{ returns}$$

$$\chi(f_T) = \sum_t \mathbf{P}\left(\text{use } w_t\right) \chi(h_{\infty w_t} \mid \text{use } w_t) = \sum_t \mathbf{P}\left(\text{use } w_t\right) \chi_t(h_{\infty w_t} \mid w_t \text{ returns})$$

Note the probability an element passes or returns on an input doesn't depend on the associated output. If the empirical and true correlations shared the same distribution over inputs, we could individually bound the deviation between conditional correlations. Of course, the empirical input distribution is discrete, whereas the true input distribution has density. We can still employ this proof strategy, albeit with some mild technical complications, by introducing a 'tilde' correlation which closely approximates the empirical correlation, yet does have the true distribution over inputs. Let $\tilde{y}(x)$ be a continuous function which places (signed) Gaussian bumps at the input data, and cancels out the probability density:

$$\tilde{y}(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{y_i}{\mathcal{D}(x_i)} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-||x_i - x||^2/\sigma^2}$$

Since x_i is drawn from $\mathcal{D}$, the density in the denominator is strictly positive. As $\sigma \to 0$, the bump approaches the Dirac delta of the distance. So within the correlation integral:

$$c(x)\tilde{y}(x)\mathcal{D}(x) \leftrightarrow \frac{1}{m} c(x_i) y_i$$

at $x = x_i$ and 0 at all other x. The empirical correlation may be reformulated (up to arbitrarily small error) to share the true input distribution by replacing y with $\tilde{y}$:

$$\left(\tilde{\chi}(c) = \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left(c(x)\tilde{y}(x)\right) = \sum_t \mathbf{P}\left(w_t \text{ returns}\right)\tilde{\chi}_t(h_{\infty w_t} \mid w_t \text{ returns})\right) \leftrightarrow \hat{\chi}(c) \text{ as } \sigma \to 0$$

Conditioning $\mathcal{D}$ on 'w_t passes' or 'w_t returns' preserves density with respect to Lebesgue measure, so by the same argument:

$$\tilde{\chi}_t(c \mid w_t \text{ returns}) \leftrightarrow \hat{\chi}_t(c \mid w_t \text{ returns}) \text{ as } \sigma \to 0$$

Taking $\sigma \to 0$:

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}}\left(\sup_{\{w_t\}} \chi(f_T) - \hat{\chi}(f_T)\right) \\
\leftrightarrow &\mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}}\left(\sup_{\{w_t\}} \chi(f_T) - \tilde{\chi}(f_T)\right) \\
= &\mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}}\left(\sup_{\{w_t\}} \sum_t \mathbf{P}\left(\text{use } w_t\right)\left(\chi_t(h_{\infty w_t} \mid w_t \text{ returns}) \ - \tilde{\chi}_t(h_{\infty w_t} \mid w_t \text{ returns})\right)\right) \\
\leq &\mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}}\left(\sup_u \sup_{\{w_t\}} \sum_t u_t\left(\chi_t(h_{\infty w_t} \mid w_t \text{ returns}) \ - \tilde{\chi}_t(h_{\infty w_t} \mid w_t \text{ returns})\right)\right) \\
\leq &\sup_u \sum_t u_t \mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}}\left(\sup_{\{w_t\}} \chi_t(h_{\infty w_t} \mid w_t \text{ returns}) \ - \tilde{\chi}_t(h_{\infty w_t} \mid w_t \text{ returns})\right) \\
\leftrightarrow &\sup_u \sum_t u_t \mathop{\mathbf{E}}_{\hat{D}\sim\mathcal{D}}\left(\sup_{\{w_t\}} \chi_t(h_{\infty w_t} \mid w_t \text{ returns}) \ - \hat{\chi}_t(h_{\infty w_t} \mid w_t \text{ returns})\right) \\
\leq &\sup_u \sum_t u_t \epsilon = \epsilon
\end{aligned}$$

The 3rd line conditions on 'use w_t', which is the same for the true and tilde correlations. The 4th line relaxes the supremum over $\{\mathbf{P}\left(\text{use } w_t\right)\}$ to all discrete distributions u. The 5th line pulls supremums outside expectations. The 6th line replaces the empirical conditional correlations. Let us examine how the final line invokes theorem 18. Note that w_t only affects the remainder of the list through $\mathcal{D}_t$, which is just a conditional distribution to which the theorem applies. The further conditioning on 'w_t returns' just multiplies each term of the deviation by the same number (at most 1), which is handled by the VC analysis.

6.5 Proof of theorem 3

Proof of smoothness

Some basic algebraic properties of the smooth absolute value function z:

- for all a,$|a| \geq z(a)$ and$|a| - z(a) \leq \frac{1}{L}\log 2$,

$$\begin{aligned}
\frac{d}{d\epsilon} z'(K(\tilde{y}+\epsilon\hat{y})) &= z''(0) \circ K\hat{y} = LK\hat{y} \\
\frac{d}{d\epsilon}(\tilde{y}+\epsilon\hat{y}) \circ z'(K(\tilde{y}+\epsilon\hat{y})) &= \hat{y} \circ z'\left(K\epsilon\hat{y}\right) + (\tilde{y}+\epsilon\hat{y}) \circ z''(K\epsilon\hat{y}) \circ K\hat{y} \\
\frac{d}{d\epsilon} z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right) &= z'\left(\epsilon K\hat{y}\right) \circ K\hat{y} \\
\frac{d}{d\epsilon} z(K(\tilde{y}+\epsilon\hat{y})) \circ (\tilde{y}+\epsilon\hat{y}) &= z'\left(\epsilon K\hat{y}\right) \circ K\hat{y} \circ (\tilde{y}+\epsilon\hat{y}) + z(K\left(\tilde{y}+\epsilon\hat{y}\right)) \circ \epsilon\hat{y} \\
\frac{d}{dy}\|y\|_K &= \frac{d}{dy}\langle y, Ky\rangle^{1/2} = \frac{1}{2\left\|y\right\|_K} 2Ky = \frac{Ky}{\|y\|_K} \\
\frac{d}{dy}\left|\left|y\right|\right|_K^2 &= 2\left|\left|y\right|\right|_K \left(\frac{d}{\mathrm{dy}}\|y\|_K\right) = 2Ky \\
\frac{d}{dy}\frac{1}{\|y\|_K} &= -\frac{1}{\|y\|_K^2}\left(\frac{d}{dy}\|y\|_K\right) = -\frac{Ky}{\|y\|_K} \Big/ \|y\|_K^2 \\
\frac{d}{d\epsilon}\left|\left|\tilde{y}+\epsilon\hat{y}\right|\right| &= \frac{d}{d\epsilon}\sqrt{\langle\tilde{y}, K\tilde{y}\rangle + 2\left\langle\tilde{y}, K\epsilon\hat{y}\right\rangle + \langle\epsilon\hat{y}, K\epsilon\hat{y}\rangle} = \frac{\langle\tilde{y}, K\hat{y}\rangle + \epsilon\left\langle\hat{y}, K\hat{y}\right\rangle}{\left|\left|\tilde{y}+\epsilon\hat{y}\right|\right|}
\end{aligned}$$

Figure 9: Some elementary derivatives for completeness.

$$\begin{aligned}
\lim_{\epsilon\to 0}\frac{z'\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right)}{\|\tilde{y}+\epsilon\hat{y}\|_K} &= \lim_{\epsilon\to 0}\frac{z''\left(K\epsilon\hat{y}\right)\circ K\hat{y}}{\|y\|_K} = \frac{LK\hat{y}}{\|y\|_K} \\
\lim_{\epsilon\to 0}\frac{z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right)}{\epsilon^2\|\hat{y}\|_K} &= \lim_{\epsilon\to 0}\frac{z'\left(\epsilon K\hat{y}\right)\circ K\hat{y}}{2\epsilon\left\|\hat{y}\right\|_K} = \frac{LK\hat{y}\circ K\hat{y}}{2\left\|\hat{y}\right\|_K} \\
\lim_{\epsilon\to 0}\frac{z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right)}{\left\|\left(\tilde{y}+\epsilon\hat{y}\right)\right\|_K} &= \lim_{\epsilon\to 0}\frac{z'\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right)\circ K\hat{y}}{\|\hat{y}\|_K} = 0
\end{aligned}$$

Figure 10: Applications of L'Hopital's rule at $||\tilde{y}||_K = 0$.

- $1 \leq z'(a) \leq 1$,
- $z''(0) = L$,
- $\lim_{a\to 0}\frac{z(a)}{a} = 0$,
- $\lim_{a\to 0}\frac{z(a)}{a^2} = L/2$,
- $\lim_{a\to 0}\frac{z'(a)}{a} = L$.

The blocks of equations in section 6.5 and fig. 11 compute the derivative of ϕ.

$$
\begin{aligned}
&\frac{d}{dy_j}\phi(y)_i \\
=&\frac{d}{dy_j} e^{-\frac{\beta}{\|y\|_K} z(Ky)_i} \circ y_i \\
=&e^{-\frac{\beta}{\|y\|_K} z(Ky)_i} \left(\left(\frac{d}{dy_j} y_i \right) - y_i \beta \left(\frac{1}{\|y\|_K} \left(z'(Ky)_i \frac{d}{dy_j} \sum_{j'} K_{i,j'} y_{j'} \right) + z(Ky)_i \left(\frac{d}{dy_j} \frac{1}{\|y\|_K} \right) \right) \right) \\
=&e^{-\frac{\beta}{\|y\|_K} z(Ky)_i} \left(1(i=j) - \beta y_i \left(\frac{1}{\|y\|_K} z'(Ky)_i K_{i,j} - \frac{z(Ky)_i}{\|y\|_K} \frac{(Ky)_j}{\|y\|_K{}^2} \right) \right) \\
=&e^{-\frac{\beta}{\|y\|_K} z(Ky)_i} \left(1(i=j) - \beta \frac{y_i}{\|y\|_K} \left(z'(Ky)_i K_{i,j} - \frac{z(Ky)_i}{\|y\|_K} \frac{(Ky)_j}{\|y\|_K} \right) \right) \\
\phi'(y) =&\mathrm{diag}\left(e^{-\frac{\beta}{\|y\|_K} z(Ky)} \right) \left(I - \beta \mathrm{diag}\left(\frac{y}{\|y\|_K} \right) \left(\mathrm{diag}(z'(Ky))K - \frac{z(Ky)}{\|y\|_K} \left(\frac{Ky}{\|y\|_K} \right)^T \right) \right)
\end{aligned}
$$

$$
\begin{aligned}
&\lim_{\epsilon\to 0} \frac{d}{d\epsilon} e^{-\beta \frac{z(K(\tilde{y}+\epsilon\hat{y}))}{\|(\tilde{y}+\epsilon\hat{y})\|_K}} \circ (\tilde{y}+\epsilon\hat{y}) \\
=&\lim_{\epsilon\to 0} e^{-\beta \frac{z(K(\tilde{y}+\epsilon\hat{y}))}{\|(\tilde{y}+\epsilon\hat{y})\|_K}} \circ \left(\frac{d}{d\epsilon} \tilde{y} + \epsilon\hat{y} \right) - e^{-\beta \frac{z(K(\tilde{y}+\epsilon\hat{y}))}{\|(\tilde{y}+\epsilon\hat{y})\|_K}} \circ \beta \left(\frac{d}{d\epsilon} \frac{z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right)}{\|(\tilde{y}+\epsilon\hat{y})\|_K} \right) \circ (\tilde{y}+\epsilon\hat{y}) \\
=&\left(\lim_{\epsilon\to 0} e^{-\beta \frac{z(K(\tilde{y}+\epsilon\hat{y}))}{\|(\tilde{y}+\epsilon\hat{y})\|_K}} \right) \circ \left(\hat{y} - \lim_{\epsilon\to 0} \beta \left(\frac{d}{d\epsilon} \frac{z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right)}{\|\tilde{y}+\epsilon\hat{y}\|_K} \right) \circ (\tilde{y}+\epsilon\hat{y}) \right) \\
=&\hat{y} - \lim_{\epsilon\to 0} \beta \left(\left(\frac{1}{\|\tilde{y}+\epsilon\hat{y}\|_K} \frac{d}{d\epsilon} z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right) \right) + \left(z\left(K\left(\tilde{y}+\epsilon\hat{y}\right)\right) \frac{d}{d\epsilon} \frac{1}{\|\tilde{y}+\epsilon\hat{y}\|_K} \right) \circ (\tilde{y}+\epsilon\hat{y}) \right) \\
=&\hat{y} - \lim_{\epsilon\to 0} \beta \left(\left(\frac{1}{\epsilon\|\hat{y}\|_K} \frac{d}{d\epsilon} z\left(\epsilon K\hat{y}\right) \right) + \left(z\left(\epsilon K\hat{y}\right) \frac{d}{d\epsilon} \frac{1}{\|\epsilon\hat{y}\|_K} \right) \right) \circ (\tilde{y}+\epsilon\hat{y}) \\
=&\hat{y} - \beta \left(\lim_{\epsilon\to 0} \frac{z'\left(\epsilon K\hat{y}\right) \circ K\hat{y}}{\epsilon\|\hat{y}\|_K} - \frac{z\left(\epsilon K\hat{y}\right)}{\epsilon^2\|\hat{y}\|_K} \right) \circ \tilde{y} \\
=&\hat{y} - \beta \left(L\frac{K\hat{y}}{\|\hat{y}\|_K} - L\frac{K\hat{y}\circ K\hat{y}}{2\|\hat{y}\|_K} \right) \circ \tilde{y} \\
=&\hat{y} - \beta L\tilde{y} \circ \left(\frac{K\hat{y}}{\|\hat{y}\|_K} \circ \left(1 - \frac{K\hat{y}}{2} \right) \right) \\
\phi'(\tilde{y}) =&I - \beta L \mathrm{diag}\left(\tilde{y}\right) \left(K - \frac{1}{2} K \circ K \right)
\end{aligned}
$$

Figure 11: The derivative of ϕ at $||y||_K > 0$, and the derivative at $||\tilde{y}||_K = 0$ in the direction of $\hat{y}$. Take coordinate directions $\hat{y}_j = e_j$. Note $\|\hat{y}_j\|_K = \sqrt{\langle e_j, Ke_j\rangle} = \sqrt{K_{j,j}} = 1$.

Proof of almost diffeomorphism

$e^{-\frac{\beta}{\|y\|_K} z(Ky)}$ is strictly positive; by Sylvester's lemma, it suffices to show there is a direction r such that, for all sufficiently small $\epsilon > 0$,

$$J(y + \epsilon r) = I - \beta \text{diag}(y + \epsilon r) Q(y + \epsilon r)$$

where

$$Q(y) = \frac{1}{\|y\|_K} \left(\text{diag}(z'(Ky))K - \frac{z(Ky)}{\|y\|_K} \left(\frac{Ky}{\|y\|_K} \right)^T \right)$$

is full rank. $\det(J(y)) = 0$, and we will show its derivative in the direction of r is nonzero. Set r to a random vector with iid $N(0,1)$ coordinates. Orthogonally decompose $r = \hat{r} + \tilde{r}$ where $||\tilde{r}||_K = 0$. Note that Q does not depend on components $\tilde{r}$ in the nullspace of K; that is, $Q(r) = Q(\hat{r})$. By the product rule:

$$\begin{aligned}
& -\frac{d}{d\epsilon} J(y + \epsilon r) \\
=& \frac{d}{d\epsilon} \text{diag}(y + \epsilon r) Q(y + \epsilon r) \\
=& \text{diag}(r) Q(y + \epsilon r) + \text{diag}(y + \epsilon r) \left(\frac{d}{d\epsilon} Q(y + \epsilon r) \right) \\
=& \text{diag}(\hat{r} + \tilde{r}) Q(y + \epsilon \hat{r}) + \text{diag}(y + \epsilon(\hat{r} + \tilde{r})) \left(\frac{d}{d\epsilon} Q(y + \epsilon \hat{r}) \right) \\
=& \text{diag}(\hat{r} + \tilde{r}) \left(Q(y + \epsilon \hat{r}) + \epsilon \frac{d}{d\epsilon} Q(y + \epsilon \hat{r}) \right) + \text{diag}(y) \left(\frac{d}{d\epsilon} Q(y + \epsilon \hat{r}) \right)
\end{aligned}$$

As $\epsilon \to 0$, this approaches: $\text{diag}(r)Q(y) + R(\hat{r})$ where

$$R(\hat{r}) = \text{diag}(y) \left(\lim_{\epsilon \to 0} \frac{d}{d\epsilon} Q(y + \epsilon \hat{r}) \right)$$

By Jacobi's formula and linearity of trace:

$$\begin{aligned}
\lim_{\epsilon \to 0} \frac{d}{d\epsilon} \det(J(y + \epsilon r)) =& \lim_{\epsilon \to 0} \text{tr} \left(\text{adj}(J(y + \epsilon r)) \left(\frac{d}{d\epsilon} J(y + \epsilon r) \right) \right) \\
=& \text{tr}(\text{adj}(J(y)) \text{diag}(r) Q(y)) + \text{tr}(\text{adj}(J(y)) R(\hat{r}))
\end{aligned}$$

The first trace is an inner product between a normal vector and a fixed vector, and is almost surely nonzero. Conditioning on $\hat{r}$, r still has the same normal distribution, so the traces are independent. Therefore the value of the first trace is almost surely not equal to the second.

Proof of local diffeomorphism

Figure 11 computes $J = \phi'(\tilde{y}) = I - LYQ$, for a fixed matrix Q and $Y = \text{diag}(\tilde{y})$. To make J invertible, it suffices to choose L larger than the inverse of the smallest nonzero eigenvalue of YQ. Let us show this eigenvalue is at least $\delta\lambda$, where δ is the value of the smallest coordinate $\tilde{y}_i$ of $\tilde{y}$, and λ is the smallest eigenvalue of Q. Since $\tilde{y}$ is strictly positive, YQ has the same eigenvalues as $\sqrt{Y}Q\sqrt{Y}$. In the worst case, the smallest eigenvector of Q is e_i:

$$e_i^T\sqrt{Y}Q\sqrt{Y}e_i = \sqrt{\delta}e_i^TQ\sqrt{\delta}e_i \geq \delta\lambda$$

Therefore $L > (\delta\lambda)^{-1}$ suffices.

Proof of instability

Lemma 20 *Not exactly orthogonal: with probability* 1, $K_{i,j} = \langle x_i, x_j\rangle\, y_i y_j \neq 0$ *for all* $i, j \leq m$.

Proof For fixed x, $\{x' : \langle x', x\rangle = 0\}$ has probability 0. Similarly, for countably many x_i, $\{x' : \vee_i \langle x', x_i\rangle = 0\} = \cup_i \{x' : \langle x', x_i\rangle = 0\}$ has probability 0. ■

Figure 11 computes $J = \phi'(\tilde{y})$. Let a be the minimum value of $K \circ K$, which, by the immediately preceding lemma, is strictly greater than 0 with probability 1. $K \circ K$ and $\tilde{y}$ have strictly positive entries, so

$$\tilde{y}^T(K \circ K)1 \geq m^2 a\|\tilde{y}\|_1$$

Observe $1^TJ1 > 1^T1 = m$, which implies J has an eigenvalue with absolute value strictly greater than 1:

$$\begin{aligned}1^TJ1 &= m - \beta L\left(K\tilde{y} - (K \circ K)\tilde{y}\right)^T 1\\ &= m + \beta L\tilde{y}^T(K \circ K)1\\ &> m\end{aligned}$$

Proof of not stuck here

The following lemma encapsulates the core idea of “running the algorithm in reverse” away from null regions of convergence.

Lemma 21 *Not stuck in null sets: if* $I_L(\tilde{y}) \cap B$ *is null for any ball* B *around* $\tilde{y}$*, then* $I_L(\tilde{y})$ *has probability zero.*

Proof For each $y \in I_L(\tilde{y})$, there is an iteration T after which the map remains in B and keeps on converging to $\tilde{y}$: $\phi^T(y) \in I_L(\tilde{y}) \cap B$. Let T^* be the largest such iteration over all y, so $\phi^{T^*}(I_L(\tilde{y})) \subseteq I_L(\tilde{y}) \cap B$. By theorem 10 and Singularity, ϕ^{-1} preserves null sets. Iterating, $\phi^{-T^*}(I_L(\tilde{y}) \cap B)$ is null and contains $I_L(\tilde{y})$. ■

First, let us assume $\tilde{y}$ is strictly positive.

Lemma 22 *Null around strictly positive: if* $\tilde{y}$ *is strictly positive, then* $I_L(\tilde{y}) \cap B$ *has codimension* 1 *(is null) for some* B *around* $\tilde{y}$.

Proof By Local Hyperbolicity, for sufficiently large L, ϕ is a C^1 diffeomorphism locally at $\tilde{y}$. Applying the stable manifold theorem, $(I_L(\tilde{y}) \cap B) \subset W$. By Instability, these sets have codimension 1. ■

Now let us reduce the general case to the strictly positive case.

Lemma 23 *Null around zeros: Suppose a stuck $\tilde{y}$ has k zero coordinates. Let $\pi : \mathbb{R}^m : \mathbb{R}^{m-k}$ project them away, overload $\phi : \mathbb{R}^{m-k} \to \mathbb{R}^{m-k}$ to drop the corresponding rows and columns of K from the definition of ϕ, and similarly overload I_L. Then for all $L > 0$, for some ball B around $\tilde{y}$, $\pi(I_L(\tilde{y}) \cap B) = I(\pi(\tilde{y})) \cap \pi(B)$, so $I(\tilde{y}) \cap B$ is null.*

Proof To ease notation, let $k = 1$ and have the first coordinate be zero. Note $\phi([0, \pi(y)]) = [0, \phi(\pi(y))]$. Since ϕ is continuous, taking y_1 to zero interchanges:

$$\lim_{y_1 \to 0} \phi(y) = \phi(\lim_{y_1 \to 0} y) = [0, \phi(\pi(y))]$$

For all $y \in I(\tilde{y})$ and $\epsilon > 0$, there is an iteration $T > 0$ such that for all additional $t \geq 0$, $\phi^t(\pi(\phi^T(y)))$ is $\epsilon/2$-close to $\pi(\phi^{T+t}(y))$, which is in turn $\epsilon/2$-close $\pi(\tilde{y})$. The first statement is because of the previously described interchange of projection. The second is because $y \in B$ implies $\pi(y) \in \pi(B)$. So, for $y \in I_L(\tilde{y}) \cap B$, $\phi^t(\pi(y))$ remains ϵ-close to $\pi(\tilde{y})$, i.e. $\pi(y) \in I_L(\pi(\tilde{y})) \cap \pi(B)$. This set has codimension 1 by theorem 22 applied to the lower-dimensional map. Since π reduces the dimension by exactly k, the result follows. ■

Proof of not stuck anywhere

Since ϕ is continuous (the preimage of open sets is open), each $I_L(\tilde{y})$ is open. According to Lindelöf's lemma, every open cover has a countable subcover. Therefore there is a countable subset $\tilde{Y}$ such that

$$\mathbf{P}\left(\cup_{\text{stuck } \tilde{y}} \cup_{L \geq L_0(\tilde{y})} I_L(\tilde{y})\right) \leq \mathbf{P}\left(\cup_{\tilde{y} \in \tilde{Y}} \cup_{L \geq L_0(\tilde{y})} I_L(\tilde{y})\right) = 0$$

where the last equality is due to theorem 15. A similar reduction was employed by Panageas and Piliouras (2016).

Proof of strict decrease

Note $y = e^{\frac{\beta}{\|y\|_K} z(Ky)} \circ \phi(y)$ and $\phi(y) > 0$. By the definition of the ℓ_1 norm, Jensen's inequality with a convex function on the left and a distribution on the right, and the first-order approximation of the exponential function:

$$\begin{aligned}
&\frac{\|y\|_1}{\|\phi(y)\|_1} \\
=&\left\langle e^{\frac{\beta}{\|y\|_K} z(Ky)}, \frac{\phi(y)}{\|\phi(y)\|_1} \right\rangle \\
\geq& e^{\frac{\beta}{\|y\|_K} \left\langle z(Ky), \frac{\phi(y)}{\|\phi(y)\|_1} \right\rangle} \\
\geq& 1 + \frac{\beta}{\|y\|_K \cdot \|\phi(y)\|_1} \langle z(Ky), \phi(y) \rangle
\end{aligned}$$

This rate of decrease is strictly greater than 1; if it is exactly 1, then $||y||_K = 0$, which is disallowed (with probability 1) by theorem 16. This inequality at $||y||_K = 0$ extends to a lower bound for other values of $||y||_K$. (Or, alternatively, an upper bound on $||y||_K$ for other values of the rate.) Formally, $\frac{\|y\|_1}{\|\phi(y)\|_1} = \rho^\beta$ (for $\rho \geq 1$) implies an upper bound on $\|y\|_K$ that goes to 0 as $r \to 1$. Begin with the second-to-last line from above, take the logarithm of both sides, and then substitute $1/\left|\left|\phi(y)\right|\right|_1 = \rho^\beta / \left|\left|y\right|\right|_1$ and the definition of ϕ on the right hand side:

$$\begin{aligned}\beta \log(\rho) \geq & \frac{\beta}{\|y\|_K} \left\langle z(Ky), \frac{\phi(y)}{\|\phi(y)\|_1} \right\rangle \\ 1 \geq & \frac{\rho^\beta}{\log(\rho)} \left\langle \frac{z(Ky)}{\|y\|_K}, e^{\frac{\beta}{\|y\|_K} z(Ky)} \circ \frac{y}{\|y\|_1} \right\rangle\end{aligned}$$

$\frac{z(Ky)}{\|y\|_K} \to 0$ if and only if $\|y\|_K \to 0$. This is necessary to satisfy the above inequality, since $\frac{\rho^\beta}{\log(\rho)} \to \infty$ as $\rho \to 1$. Therefore $\left\|\phi^t(y)\right\|_1$ is strictly decreasing unless $\left\|\phi^t(y)\right\|_K = 0$, which (by theorem 16) implies $\phi^t(y) = 0$. Note, since $|a| > z(a)$, the same result holds for α as well as ϕ.

6.6 Brief introduction to juntas

Our experiments involve the following well-studied boolean functions, ordered roughly in terms of complexity or difficulty.

Conjunctions are 1 if all the coordinates in S are 1, and -1 otherwise. They are (affine) halfspaces $\text{sgn}(\sum_{i \in S} x_i - k)$. When the inputs are 'symmetrically' distributed on the hypercube, lifting takes just $n^{O(\log 1/\epsilon)}$ time, which is formally independent of k despite being exponential in the excess error ϵ (Feldman and Kothari, 2015). For other distributions on the hypercube, it is NP-hard to even weakly fit a halfspace (Feldman et al., 2012). They have low noise sensitivity because they are monotone.

Majorities take the 'majority vote' $\text{sgn}(\sum_i x_i)$ within S; they are halfspaces defined by the indicator vector of S. Fitting majorities very accurately is likely difficult since they are significantly correlated with parity functions (Kalai et al., 2008). Majority is , in a sense, the most noise-stable boolean function (Mossel et al., 2005), so overfitting may not be a concern.

Pairs (aka 2-tribes) are defined, for even k, as the disjunction ("or") of $\frac{k}{2}$ disjoint conjunctions of pairs of coordinates. Tribes are not well-studied in learning theory; they were recently devised as a boolean function where each coordinate individually exhibits the least 'influence' on the output (Ben-Or and Linial, 1985). For larger tribe sizes, these functions are the most sensitive to large amounts of noise (Mossel and O'Donnell, 2002), but this may not manifest for pairs.

Parities $\prod_{i \in S} x_i$ are perhaps the most notorious functions in learning theory; many difficult, open problems reduce to learning parities with noise (Feldman et al., 2009). They are also the most noise-sensitive boolean functions. We expect all the learning algorithms to degrade as k increases.

Intersections of halfspaces are not juntas, but they may be ϵ-approximately learned by lifting in roughly $n^{O\left(k^2/\epsilon^2\right)}$ time (Klivans et al., 2002). For a perfect fit, an $\Omega(n)$-degree polynomial is required even for $k = 2$ (Sherstov, 2013). Furthermore, for non-uniform distributions on $\{-1, 1\}^n$, if $k = n^c$ (for any constant $c > 0$), learning is cryptographically hard (Klivans and Sherstov, 2006). We expect all the learning algorithms to degrade as k

increases. However, intersections of random halfspaces have relatively low noise sensitivity (Kane, 2014), so we might expect train-test deviation to be tame.