Provably Accurate and Scalable Linear Classifiers in Hyperbolic Spaces

Chao Pan[*], Eli Chien[*], Puoya Tabaghi, Jianhao Peng, Olgica Milenkovic

ECE, University of Illinois Urbana-Champaign, 306 N Wright St, Urbana, Illinois 61801, USA

{chaopan2, ichien3, tabaghi2, jianhao2, milenkov}@illinois.edu

Abstract—Many high-dimensional practical data sets have hierarchical structures induced by graphs or time series. Such data sets are hard to process in Euclidean spaces and one often seeks low-dimensional embeddings in other space forms to perform the required learning tasks. For hierarchical data, the space of choice is a hyperbolic space because it guarantees low-distortion embeddings for tree-like structures. The geometry of hyperbolic spaces has properties not encountered in Euclidean spaces that pose challenges when trying to rigorously analyze algorithmic solutions. We propose a unified framework for learning scalable and simple hyperbolic linear classifiers with provable performance guarantees. The gist of our approach is to focus on Poincaré ball models and formulate the classification problems using tangent space formalisms. Our results include a new hyperbolic perceptron algorithm as well as an efficient and highly accurate convex optimization setup for hyperbolic support vector machine classifiers. Furthermore, we adapt our approach to accommodate second-order perceptrons, where data is preprocessed based on second-order information (correlation) to accelerate convergence, and strategic perceptrons, where potentially manipulated data arrives in an online manner and decisions are made sequentially. The excellent performance of the Poincaré second-order and strategic perceptrons shows that the proposed framework can be extended to general machine learning problems in hyperbolic spaces. Our experimental results, pertaining to synthetic, single-cell RNA-seq expression measurements, CIFAR10, Fashion-MNIST and mini-ImageNet, establish that all algorithms provably converge and have complexity comparable to those of their Euclidean counterparts. Accompanying codes can be found at: https://github.com/thupchnsky/PoincareLinearClassification[1].

Index Terms—Hyperbolic, Support Vector Machine, Perceptron, Online Learning

I. INTRODUCTION

Representation learning in hyperbolic spaces has received significant interest due to its effectiveness in capturing latent hierarchical structures [2]–[7]. It is known that arbitrarily low-distortion embeddings of tree-structures in Euclidean spaces is impossible even when using an unbounded number of dimensions [8]. In contrast, precise and simple embeddings are possible in the Poincaré disk, a hyperbolic space model with only two dimensions [3].

Despite their representational power, hyperbolic spaces are still lacking foundational analytical results and algorithmic solutions for a wide variety of downstream machine learning tasks. In particular, the question of designing highly-scalable classification algorithms with provable performance guarantees that exploit the structure of hyperbolic spaces remains open. While a few prior works have proposed specific algorithms for learning classifiers in hyperbolic space, they are primarily empirical in nature and do not come with theoretical convergence guarantees [9], [10]. The work [11] described the first attempt to establish performance guarantees for the hyperboloid perceptron, but the proposed algorithm is not transparent and fails to converge in practice. Furthermore, the methodology used does not naturally generalize to other important classification methods such as support vector machines (SVMs) [12]. Hence, a natural question arises: Is there a unified framework that allows one to generalize classification algorithms for Euclidean spaces to hyperbolic spaces, make them highly scalable and rigorously establish their performance guarantees?

We give an affirmative answer to this question for a wide variety of classification algorithms. By redefining the notion of separation hyperplanes in hyperbolic spaces, we describe the first known Poincaré ball perceptron and SVM methods with provable performance guarantees. Our perceptron algorithm resolves convergence problems associated with the perceptron method in [11]. It is of importance as they represent a form of online learning in hyperbolic spaces and are basic components of hyperbolic neural networks. On the other hand, our Poincaré SVM method successfully addresses issues associated with solving and analyzing a nontrivial nonconvex optimization problem used to formulate hyperboloid SVMs in [9]. In the latter case, a global optimum may not be attainable using projected gradient descent methods and consequently this SVM method does not provide tangible guarantees. Our proposed algorithms may be viewed as "shallow" one-layer neural networks for hyperbolic spaces that are not only scalable but also (unlike deep networks [13], [14]) exhibit an extremely small storage-footprint. They are also of significant relevance for few-shot meta-learning [15] and for applications such as single-cell subtyping and image data processing as described in our experimental analysis (see Figure 1 for the Poincaré embedding of these data sets).

For our algorithmic solutions we choose to work with the Poincaré ball model for several practical and mathematical reasons. First, this model lends itself to ease of data visualization and it is known to be conformal. Furthermore, many recent deep learning models are designed to operate on the Poincaré ball model, and our detailed analysis and

---

*equal contribution

[1] A shorter version of this work [1] was presented as a regular paper at the International Conference on Data Mining (ICDM), 2021.
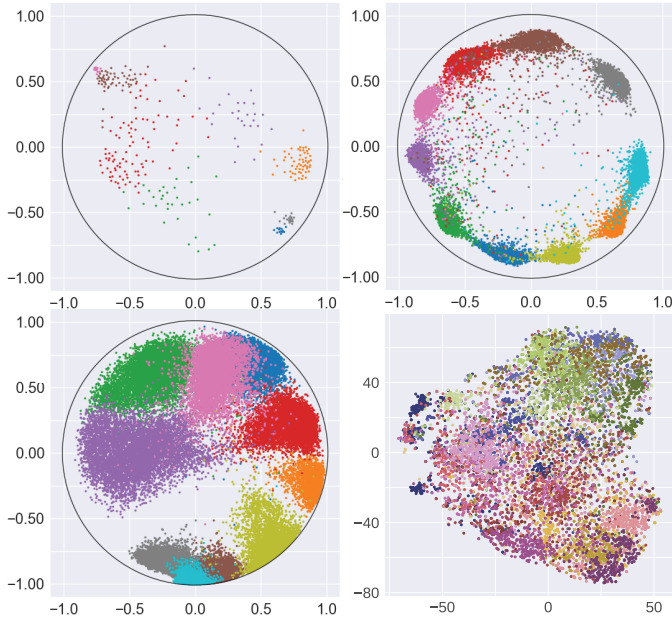
Fig. 1. Visualization of four embedded data sets: Olsson's single-cell RNA expression data (top left, $K = 8, d = 2$), CIFAR10 (top right, $K = 10, d = 2$), Fashion-MNIST (bottom left, $K = 10, d = 2$) and mini-ImageNet (bottom right, $K = 20, d = 512$). Here K stands for the number of classes and d stands for the dimension of embedded Poincaré ball. Data points from mini-ImageNet are mapped into 2 dimensions using tSNE for viewing purposes only and thus may not lie in the unit Poincaré disk. Different colors represent different classes.

experimental evaluation of the perceptron, SVM and related algorithms can improve our understanding of these learning methods. The key insight is that *tangent spaces* of points in the Poincaré ball model are Euclidean. This, along with the fact that logarithmic and exponential maps are readily available to switch between different relevant spaces simplifies otherwise complicated derivations and allows for addressing classification tasks in a unified manner using convex programs. To estimate the reference points for tangent spaces we make use of convex hull algorithms over the Poincaré ball model and explain how to select the free parameters of the classifiers. Further contributions include generalizations of the work [16] on second-order perceptrons and the work [17] on strategic perceptrons in Euclidean spaces.

The proposed perceptron and SVM methods easily operate on massive synthetic data sets comprising millions of points and up to one thousand dimensions. All perceptron algorithms converge to an error-free result provided that the data satisfies an ε-margin assumption. The second-order Poincaré perceptron converges using significantly fewer iterations than the perceptron method, which matches the advantages offered by its Euclidean counterpart [16]. This is of particular interest in online learning settings and it also results in lower excess risk [18]. Our Poincaré SVM formulation, which unlike [9] comes with provable performance guarantees, also operates significantly faster than its nonconvex counterpart (1 minute versus 9 hours on a set of 10^6 points, which is $540\times$ faster) and also offers improved classification accuracy as high as 50%.

Real-world data experiments involve single-cell RNA expression measurements [19], CIFAR10 [20], Fashion-MNIST [21] and mini-ImageNet [22]. These data sets have challenging overlapping-class structures that are hard to process in Euclidean spaces, while Poincaré SVMs still offer outstanding classification accuracy with gains as high as 47.91% compared to their Euclidean counterparts.

This paper is organized as follows. Section II describes an initial set of experimental results illustrating the scalability and high-quality performance of our Poincaré SVM method compared to the corresponding Euclidean and other hyperbolic classifiers. This section also contains a discussion of prior works on hyperbolic perceptrons and SVMs that do not use the tangent space formalism and hence fail to converge and/or provide provable convergence guarantees, as well as a review of variants of perceptron algorithms. Section III describes relevant concepts from differential geometry needed for the analysis at hand. Section IV contains our main results, analytical convergence guarantees for the proposed Poincaré perceptron and SVM learners. Section V includes examples pertaining to generalizations of two variants of perceptron algorithms in Euclidean spaces. A more detailed set of experimental results, pertaining to real-world single-cell RNA expression measurements for cell-typing and three collections of image data sets is presented in Section VI. These results illustrate the expression power of hyperbolic spaces for hierarchical data and highlight the unique feature and performance of our techniques.

II. RELEVANCE AND RELATED WORK

To motivate the need for new classification methods in hyperbolic spaces we start by presenting illustrative numerical results for synthetic data sets. We compare the performance of our Poincaré SVM with the previously proposed hyperboloid SVM [9] and Euclidean SVM. The hyperbolic perceptron outlined in [11] does not converge and is hence not used in our comparative study. Rigorous descriptions of all mathematical concepts and pertinent proofs are postponed to the next sections of this paper.

One can clearly observe from Figure 2 that the accuracy of Euclidean SVMs may be significantly below 100%, as the data points are not linearly separable in Euclidean but rather only in the hyperbolic space. Furthermore, the nonconvex SVM method of [9] does not scale well as the number of points increases: It takes roughly 9 hours to complete the classification process on 10^6 points while our Poincaré SVM takes only 1 minute. Furthermore, the algorithm breaks down when the data dimension increases to $1,000$ due to its intrinsic non-stability. Only our Poincaré SVM can achieve nearly optimal (100%) classification accuracy with extremely low time complexity for all data sets considered. More extensive experimental results on synthetic and real-world data can be found in Section VI.

The exposition in our subsequent sections explains what makes our classifiers as fast and accurate as demonstrated, especially when compared to the handful of other existing
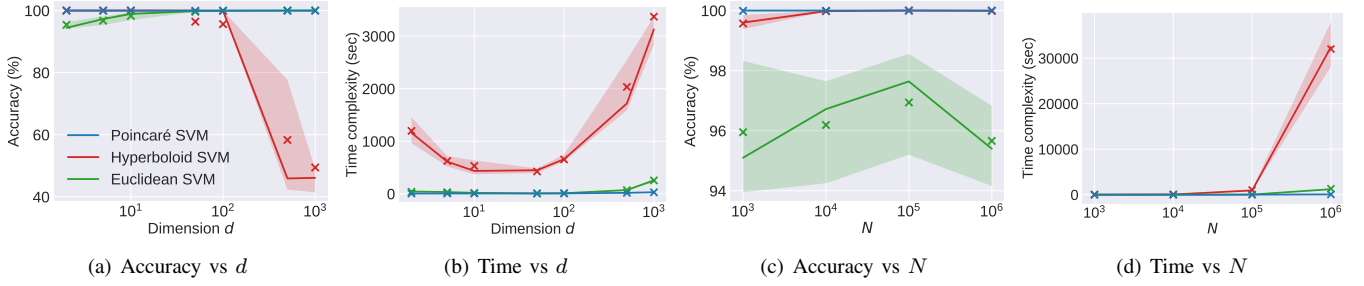
Fig. 2. Classification of N points in d dimensions selected uniformly at random in the Poincaré ball. The upper and lower boundaries of the shaded region represent the first and third quantile, respectively. The line shows the medium (second quantile) and the marker $\times$ indicates the mean. Detailed explanations pertaining to the test results can be found in Section VI.

hyperbolic space methods. In the first line of work to address SVMs in hyperbolic spaces [9] the authors chose to work with the hyperboloid model of hyperbolic spaces which resulted in a nonconvex optimization problem formulation. The nonconvex problem was solved via projected gradient descent which is known to be able to provably find only a *local* optimum. In contrast, as we will show, our Poincaré SVM provably converges to a *global* optimum. The second related line of work [11] studied hyperbolic perceptrons and a hyperbolic version of robust large-margin classifiers for which a performance analysis was included. This work also solely focused on the hyperboloid model and the hyperbolic perceptron method outlined therein does not converge. The main difference between this work and previous works is that we resort to a straightforward, universal and simple proof techniques that "transfers" the classification problem from a Poincaré ball back to a Euclidean space through the use of tangent space formalisms. Our analytical convergence results are extensively validated experimentally, as described above and in Section VI.

There exists many variants of perceptron-type algorithms in the literature. One variant uses second-order information (correlation) in samples to accelerate the convergence of standard perceptron algorithms [16]. The method, termed the *second-order perceptron*, makes use of the data correlation matrix to ensure fast convergence to the optimal perceptron classifier. Another line of work is related to strategic classification [17]. Strategic classification deals with the problem of learning a classier when the learner relies on data that is provided by strategic agents in an online manner [23], [24]. This problem is of great importance in decision theory and fair learning. It was shown in [17] that standard perceptrons can oscillate and fail to converge when the data is manipulated, and the strategic perceptron is proposed to mitigate this problem. We successfully extend our framework to these two settings and describe Poincaré second-order and strategic perceptrons based on their Euclidean counterparts.

In addition to perceptrons and SVM described above, a number of hyperbolic neural networks solutions have been put forward as well [13], [14]. These networks were built upon the idea of Poincaré hyperplanes and motivated our

approach for designing Poincaré-type perceptrons and SVMs. One should also point out that there are several other deep learning methods specifically designed for the Poincaré ball model, including hyperbolic graph neural networks [25] and Variational Autoencoders [26]–[28]. Despite the excellent empirical performance of these methods theoretical guarantees are still unavailable due to the complex formalism of deep learners. Our algorithms and proof techniques illustrate for the first time why elementary components of such networks, such as perceptrons, perform exceptionally well when properly formulated for a Poincaré ball.

III. REVIEW OF HYPERBOLIC SPACES

We start with a review of basic notions pertinent to hyperbolic spaces. We then proceed to introduce the notion of *separation hyperplanes* in the Poincaré ball model of hyperbolic space which is crucial for all our subsequent derivations. The relevant notation used is summarized in Table I.

TABLE I
NOTATION AND DEFINITIONS.

Notation	Definition
n	Dimension of Poincaré ball
c	Absolute value of the negative curvature, $c > 0$
$\|x\|$	$\sqrt{\sum_{i=1}^{n} x_i^2}$
$\langle x, y \rangle$	$\sum_{i=1}^{n} x_i y_i$
$\mathbb{B}_c^n$	Poincaré ball model, $\mathbb{B}_c^n = \{x \in \mathbb{R}^n : \sqrt{c}\,\|x\| < 1\}$
$g_p^{\mathbb{B}_c}(x, y)$	$\frac{2}{1 - c\|p\|^2} \langle x, y \rangle$

The Poincaré ball model. Despite the existence of a multitude of equivalent models for hyperbolic spaces, Poincaré ball models have received the broadest attention in the machine learning and data mining communities. This is due to the fact that the Poincaré ball model provides conformal representations of shapes and point sets, i.e., in other words, it preserves Euclidean angles of shapes. The model has also been successfully used for designing hyperbolic neural networks [13], [14] with excellent heuristic performance. Nevertheless, the field of learning in hyperbolic spaces – under the Poincaré or other models – still remains largely unexplored.

The Poincaré ball model $(\mathbb{B}_c^n, g^{\mathbb{B}})$ is a Riemannian manifold. For the absolute value of the curvature $c > 0$, its domain is the open ball of radius $1/\sqrt{c}$:

$$\mathbb{B}_c^n = \{x \in \mathbb{R}^n : \sqrt{c}\,\|x\| < 1\},$$

here and elsewhere $\|\cdot\|$ stands for the ℓ_2 norm and $\langle\cdot,\cdot\rangle$ stands for the standard inner product. The Riemannian metric of Poincaré model is defined as

$$g_x^{\mathbb{B}_c}(\cdot,\cdot) = (\sigma_x^c)^2 \langle\cdot,\cdot\rangle, \ \ \sigma_x^c = 2/(1 - c\|x\|^2).$$

For $c = 0$, we recover the Euclidean space, i.e., $\mathbb{B}_0^n = \mathbb{R}^n$. For simplicity, we focus on the case $c = 1$ in this paper albeit our results can be generalized to hold for arbitrary $c > 0$. Furthermore, for a reference point $p \in \mathbb{B}^n$, we denote its tangent space, the first order linear approximation of $\mathbb{B}^n$ around p, by $T_p\mathbb{B}^n$.

In the following, we introduce Möbius addition and scalar multiplication — two basic operators in the Poincaré ball [29]. These operators represent analogues of vector addition and scalar-vector multiplication in Euclidean spaces. The Möbius addition of $x, y \in \mathbb{B}^n$ is defined as

$$x \oplus y = \frac{(1 + 2\langle x,y\rangle + \|y\|^2)x + (1 - \|x\|^2)y}{1 + 2\langle x,y\rangle + \|x\|^2\|y\|^2}. \quad (1)$$

Unlike its vector-space counterpart, this addition is noncommutative and nonassociative. The Möbius version of multiplication of $x \in \mathbb{B}^n \setminus \{0\}$ by a scalar $r \in \mathbb{R}$ is defined according to

$$r \otimes x = \tanh(r\tanh^{-1}(\|x\|))\frac{x}{\|x\|} \text{ and } r \otimes 0 = 0. \quad (2)$$

For detailed properties of these operations, see [13], [30]. The distance function in the Poincaré model is

$$d(x,y) = 2\tanh^{-1}(\|(-x) \oplus y\|). \quad (3)$$

Using Möbius operations one can also describe geodesics (analogues of straight lines in Euclidean spaces) in $\mathbb{B}^n$. The geodesics connecting two points $x, y \in \mathbb{B}^n$ is given by

$$\gamma_{x \to y}(t) = x \oplus (t \otimes ((-x) \oplus y)). \quad (4)$$

Note that $t \in [0,1]$ and $\gamma_{x \to y}(0) = x$ and $\gamma_{x \to y}(1) = y$.

The following result explains how to construct a geodesic with a given starting point and a tangent vector.

Lemma 3.1 (Lemma 1 in [13]): For any $p \in \mathbb{B}^n$ and $v \in T_p\mathbb{B}^n$ s.t. $g_p(v,v) = 1$, the geodesic starting at p with tangent vector v equals:

$$\gamma_{p,v}(t) = p \oplus \left(\tanh\left(\frac{t}{2}\right)\frac{v}{\|v\|}\right), \quad (5)$$
$$\text{where } \gamma_{p,v}(0) = p \text{ and } \dot{\gamma}_{p,v}(0) = v.$$

We complete the overview by introducing logarithmic and exponential maps.

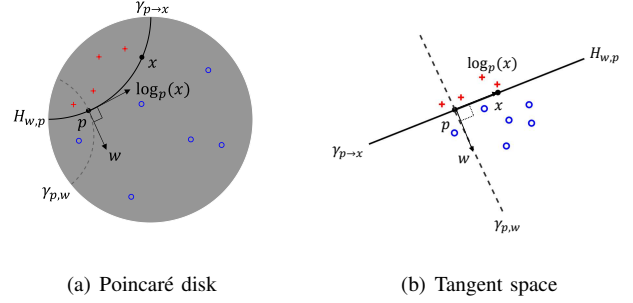
(a) Poincaré disk   (b) Tangent space

Fig. 3. Figure (a): A linear classifier in Poincaré disk $\mathbb{B}^2$. Figure (b): Corresponding tangent space $T_p\mathbb{B}^2$.

Lemma 3.2 (Lemma 2 in [13]): For any point $p \in \mathbb{B}^n$ the exponential map $\exp_p : T_p\mathbb{B}^n \mapsto \mathbb{B}^n$ and the logarithmic map $\log_p : \mathbb{B}^n \mapsto T_p\mathbb{B}^n$ are given for $v \neq 0$ and $x \neq p$ by:

$$\exp_p(v) = p \oplus \left(\tanh\left(\frac{\sigma_p\|v\|}{2}\right)\frac{v}{\|v\|}\right), \quad (6)$$

$$\log_p(x) = \frac{2}{\sigma_p}\tanh^{-1}(\|(-p) \oplus x\|)\frac{(-p) \oplus x}{\|(-p) \oplus x\|}. \quad (7)$$

The geometric interpretation of $\log_p(x)$ is that it gives the tangent vector v of x for starting point p. On the other hand, $\exp_p(v)$ returns the destination point x if one starts at the point p with tangent vector v. Hence, a geodesic from p to x may be written as

$$\gamma_{p \to x}(t) = \exp_p(t\log_p(x)), \quad t \in [0,1]. \quad (8)$$

See Figure 3 for the visual illustration.

IV. CLASSIFICATION IN HYPERBOLIC SPACES

A. Classification Algorithms for Poincaré Balls

Classification with Poincaré hyperplanes. The recent work [13] introduced the notion of a Poincaré hyperplane which generalizes the concept of a hyperplane in Euclidean space. The Poincaré hyperplane with reference point $p \in \mathbb{B}^n$ and normal vector $w \in T_p\mathbb{B}^n$ in the above context is defined as

$$H_{w,p} = \{x \in \mathbb{B}^n : \langle\log_p(x), w\rangle_p = 0\}$$
$$= \{x \in \mathbb{B}^n : \langle(-p) \oplus x, w\rangle = 0\}, \quad (9)$$

where $\langle x,y\rangle_p = (\sigma_p)^2\langle x,y\rangle$. The minimum distance of a point $x \in \mathbb{B}^n$ to $H_{w,p}$ has the following close form

$$d(x, H_{w,p}) = \sinh^{-1}\left(\frac{2|\langle(-p) \oplus x, w\rangle|}{(1 - \|(-p) \oplus x\|^2)\|w\|}\right). \quad (10)$$

We find it useful to restate (10) so that it only depends on vectors in the tangent space $T_p\mathbb{B}^n$ as follows.

Lemma 4.1: Let $v = \log_p(x)$ (and thus $x = \exp_p(v)$), then we have

$$d(x, H_{w,p}) = \sinh^{-1}\left(\frac{2\tanh(\frac{\sigma_p\|v\|}{2})|\langle v,w\rangle|}{(1 - \tanh(\frac{\sigma_p\|v\|}{2})^2)\|w\|\|v\|}\right) \quad (11)$$

Equipped with the above definitions, we now focus on binary classification in Poincaré models. To this end, let $\{(x_i, y_i)\}_{i=1}^N$ be a set of N data points, where $x_i \in \mathbb{B}^n$ and $y_i \in \{\pm 1\}$ represent the true labels. Note that based on Lemma 4.1, the decision function based on $H_{w,p}$ is $h_{w,p}(x) = \text{sgn}\left(\langle \log_p(x_i), w \rangle\right)$. This is due to the fact that $\sinh^{-1}$ does not change the sign of its input and that all other terms in (11) are positive if $v_i = \log_p(x_i)$ and $w \neq 0$. For the case that either $\log_p(x_i)$ or w is 0, $\langle v, w \rangle = 0$ and thus the sign remains unchanged. For linear classification, the goal is to learn w that correctly classifies all points. For large margin classification, we further required that the learnt w achieves the largest possible margin,

$$\max_{w \in T_p \mathbb{B}^n} \min_{i \in [N]} y_i h_{w,p}(x_i) d(x_i, H_{w,p}). \quad (12)$$

In what follows we outline the key idea behind our approach to classification and analysis of the underlying algorithms. We start with the perceptron classifier, which is the simplest approach yet of relevance for online settings. We then proceed to describe our SVM method which offers excellent performance with provable guarantees.

Our approach builds upon the result of Lemma 4.1. For each $x_i \in \mathbb{B}^n$, let $v_i = \log_p(x_i)$. We assign a corresponding weight as

$$\eta_i = \frac{2 \tanh\left(\frac{\sigma_p \|v_i\|}{2}\right)}{\left(1 - \tanh\left(\frac{\sigma_p \|v_i\|}{2}\right)^2\right) \|v_i\|}. \quad (13)$$

Without loss of generality, we also assume that the optimal normal vector has unit norm $\|w^\star\| = 1$. Then (11) can be rewritten as

$$d(x_i, H_{w^\star, p}) = \sinh^{-1}\left(\eta_i |\langle v_i, w^\star \rangle|\right). \quad (14)$$

Note that $\eta_i \geq 0$ and $\eta_i = 0$ if and only if $v_i = 0$, which corresponds to the case $x_i = p$. Nevertheless, this "border" case can be easily eliminated under a margin assumption. Hence, the problem of finding an optimal classifier becomes similar to the Euclidean case if one focuses on the *tangent space* of the Poincaré ball model (see Figure 3 for an illustration).

B. The Poincaré Perceptron

We first restate the standard assumptions needed for the analysis of the perceptron algorithm in Euclidean space for the Poincaré model.

Assumption 4.1:

$$\exists w^\star \in T_p \mathbb{B}^n,\ y_i \left\langle \log_p(x_i), w^\star \right\rangle > 0\ \forall i \in [N], \quad (15)$$
$$\exists \epsilon > 0 \text{ s.t. } d(x_i, H_{w^\star, p}) \geq \varepsilon\ \forall i \in [N], \quad (16)$$
$$\|x_i\| \leq R < 1\ \forall i \in [N]. \quad (17)$$

The first assumption (15) postulates the existence of a classifier that correctly classifies every points. The margin assumption is listed in (16), while (17) ensures that points lie in a bounded region.

Using (14) we can easily design the Poincaré perceptron update rule. If the kth mistake happens at instance (x_{i_k}, y_{i_k}) (i.e., $y_{i_k} \left\langle \log_p(x_{i_k}), w_k \right\rangle \leq 0$), then

$$w_{k+1} = w_k + \eta_{i_k} y_{i_k} \log_p(x_{i_k}),\ w_1 = 0. \quad (18)$$

The complete Poincaré perceptron is then shown in Algorithm 1.

Algorithm 1: Poincaré Perceptron

Input: Data points $\{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, reference point p.

1 Initialization: $w_1 = 0, k = 1$.
2 **for** $t = 1, 2, \ldots$ **do**
3   Get (x_t, y_t), compute $v_t = \log_p(x_t)$ and
    $\eta_t = \frac{2 \tanh\left(\frac{\sigma_p \|v_t\|}{2}\right)}{\left(1 - \tanh\left(\frac{\sigma_p \|v_t\|}{2}\right)^2\right) \|v_t\|}$.
4   Predict $\hat{y}_t = \text{sgn}(\langle w_k, v_t \rangle)$.
5   **if** $\hat{y}_t \neq y_t$ **then**
6         $w_{k+1} = w_k + \eta_t y_t v_t,\ k = k + 1$.
7   **end**
8 **end**

Algorithm 1 comes with the following convergence guarantees.

Theorem 4.1: Under Assumption 4.1, the Poincaré perceptron Algorithm 1 will correctly classify all points with at most $\left(\frac{2R_p}{(1 - R_p^2) \sinh(\varepsilon)}\right)^2$ updates, where $R_p = \frac{\|p\| + R}{1 + \|p\| R}$.

Proof. To prove Theorem 4.1, we need the technical lemma below.

Lemma 4.2: Let $a \in \mathbb{B}^n$. Then

$$\underset{b \in \mathbb{B}^n : \|b\| \leq R}{\text{argmax}} \|a \oplus b\| = R \frac{a}{\|a\|}, \quad (19)$$
$$\underset{b \in \mathbb{B}^n : \|b\| \leq R}{\text{argmax}} \|b \oplus a\| = R \frac{a}{\|a\|}. \quad (20)$$

If we replace $\oplus$ with ordinary vector addition, we can basically interpret the result as follows: The norm of $\|a + b\|$ is maximized when a has the same direction as b. This can be easily proved by invoking the Cauchy-Schwartz inequality. However, it is nontrivial to show the result under Möbius addition. The proof of Lemma 4.2 is shown in Appendix A.

As already mentioned in Section IV-A, the key idea is to work in the tangent space $T_p \mathbb{B}^n$, in which case the Poincaré perceptron becomes similar to the Euclidean perceptron. First, we establish the boundedness of the tangent vectors $v_i = \log_p(x_i)$ in $T_p \mathbb{B}^n$ by invoking the definition of σ_p from Section III:

$$\|v_i\| = \|\log_p(x_i)\| = \frac{2}{\sigma_p} \tanh^{-1}(\|(-p) \oplus x_i\|)$$
$$\overset{(a)}{\leq} \frac{2}{\sigma_p} \tanh^{-1}(R_p) \Leftrightarrow \tanh\left(\frac{\sigma_p \|v_i\|}{2}\right) \leq R_p, \quad (21)$$

where (a) can be shown to hold true by involving Lemma 4.2 and performing some algebraic manipulations. Details are as follows.

$$\|(-p) \oplus x_i\| \leq \|(-p) \oplus \frac{R}{\|p\|}(-p))\| \quad \text{By Lemma 4.2}$$

$$= \left\| \frac{(1 + 2R\|p\| + R^2)(-p) + (1 - \|p\|^2)\frac{R(-p)}{\|p\|}}{(R + \|p\|^2)^2} \right\| \quad \text{By (1)}$$

$$= \frac{\|p\| + R^2\|p\| + R\|p\|^2 + R}{(R + \|p\|^2)^2} = \frac{(R + \|p\|)(1 + R\|p\|)}{(R + \|p\|^2)^2}$$

$$= \frac{1 + R\|p\|}{R + \|p\|^2} = R_p.$$

Combining this with the fact that $\tanh^{-1}(\cdot)$ is non-decreasing in $[0, 1)$, we arrive at the inequality (a).

The remainder of the analysis is similar to that of the standard Euclidean perceptron. We first lower bound $\|w_{k+1}\|$ as

$$\|w_{k+1}\| \overset{(b)}{\geq} \langle w_{k+1}, w^\star \rangle = \langle w_k, w^\star \rangle + \eta_{i_k} y_{i_k} \langle v_{i_k}, w^\star \rangle$$
$$\overset{(c)}{\geq} \langle w_k, w^\star \rangle + \sinh(\varepsilon) \geq \cdots \geq k \sinh(\varepsilon), \quad (22)$$

where (b) follows from the Cauchy-Schwartz inequality and (c) is a consequence of the margin assumption (16). Next, we upper bound $\|w_{k+1}\|$ as

$$\|w_{k+1}\|^2 \leq \|w_k\|^2 + \left(\frac{2\tanh(\frac{\sigma_p \|v_{i_k}\|}{2})}{1 - \tanh(\frac{\sigma_p \|v_{i_k}\|}{2})^2} \right)^2$$
$$\overset{(d)}{\leq} \|w_k\|^2 + \left(\frac{2R_p}{1 - R_p^2} \right)^2 \leq \cdots \leq k \left(\frac{2R_p}{1 - R_p^2} \right)^2, \quad (23)$$

where (d) is a consequence of (21) and the fact that the function $f(x) = \frac{x}{1-x^2}$ is nondecreasing for $x \in (0, 1)$. Note that $R_p < 1$ since $\|p\| < 1, R < 1$ and

$$(1 - \|p\|)(1 - R) = 1 + \|p\|R - (\|p\| + R) > 0.$$

Combining (22) and (23) we have

$$k^2 \sinh^2(\varepsilon) \leq k \left(\frac{2R_p}{1 - R_p^2} \right)^2 \Leftrightarrow k \leq \left(\frac{2R_p}{(1 - R_p^2)\sinh(\varepsilon)} \right)^2,$$

which completes the proof.

C. Discussion

It is worth pointing out that the authors of [11] also designed and analyzed a different version of a hyperbolic perceptron in the hyperboloid model $\mathbb{L}^d = \{x \in \mathbb{R}^{d+1} : [x, x] = -1\}$, where $[x, y] = x^T H y, H = \text{diag}(-1, 1, 1, \ldots, 1)$ denotes the Minkowski inner product (A detailed description of the hyperboloid model can be found in Appendix 5). Their proposed update rule is

$$u_k = w_k + y_n x_n \quad \text{if} \ - y_n[w_k, x_n] < 0, \quad (24)$$
$$w_{k+1} = u_k / \min\{1, \sqrt{[u_k, u_k]}\}, \quad (25)$$

where (25) is a "normalization" step. Although a convergence result was claimed in [11], we demonstrate by simple counterexamples that their hyperbolic perceptron do not converge, mainly due to the choice of the update direction. This can be easily seen by using the proposed update rule with $w_0 = e_2$ and $x_1 = e_1 \in \mathbb{L}^2$ with label $y_1 = 1$, where e_j are standard basis vectors (the particular choice leads to an ill-defined normalization). Other counterexamples $w_0 = 0$ involve normalization with complex numbers for arbitrary $x_1 \in \mathbb{L}^2$ which is not acceptable.

It appears that the algorithm [11] does not converge for most of the data sets tested. The results on synthetic data sets are shown in Figure 4. For this test, data points satisfying a $\varepsilon-$margin assumption are generated in a hyperboloid model $\mathbb{L}^2$ and then further converted into a Poincaré ball model for use with Algorithm 1. The two accuracy plots shown in Figure 4 (red and green) represent the best achievable results for their corresponding algorithms within the theoretical upper bound on the number of updates of the weight vector. The experiment was performed for 100 different values of ε. From the generated results, one can easily conclude that our algorithm always converge within the theoretical upper bound provided in Theorem 4.1, while the other algorithm is unstable.
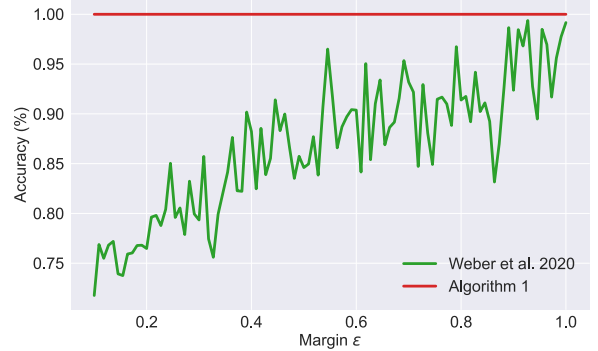

Fig. 4. A comparison between the classification accuracy of our Poincaré perceptron Algorithm 1 and the algorithm in [11], for different values of the margin ε. The classification accuracy is the average value over five independent random trials. The stopping criterion is to either achieve a 100% classification accuracy or reach the corresponding theoretical upper bound of updates on the weight vector.

D. Learning Reference Points

So far we have tacitly assumes that the reference point p is known in advance. While the reference point and normal vector can be learned in a simple manner in Euclidean spaces, this is not the case for the Poincaré ball model due to the non-linearity of its logarithmic map and Möbius addition, as illustrated in Figure 5.

Importantly, we have the following observation: A hyperplane correctly classifies all points if and only if it separates their convex hulls (the definition of "convexity" in hyperbolic spaces follows from replacing lines with geodesics [31]). Hence we can easily generalize known convex hull algorithms
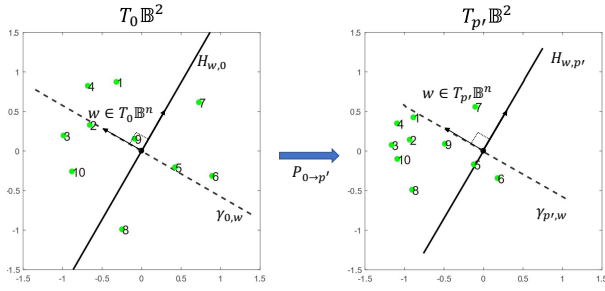
Fig. 5. The effect of changing the reference point via parallel transport in corresponding tangent spaces $T_0\mathbb{B}^2$ and $T_{p'}\mathbb{B}^2$, for $p' \in \mathbb{B}^2$. Note that the images of data points, $\log_p(x_i)$, change in the tangent spaces with the choice of the reference point.
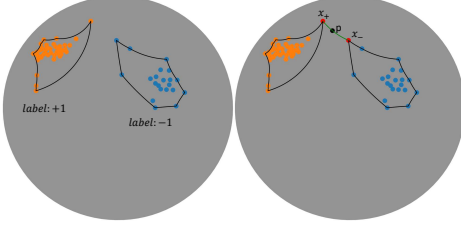

Fig. 6. Learning a reference point p. Step 1 (left): Construct convex hull for each cluster. Black lines are geodesics defining the surface of convex hull. Step 2 (right): Find a minimum distance pair and choose p as their midpoint.

to the Poincaré ball model, including the Graham scan [32] (Algorithm 2) and Quickhull [33] (see the Appendix A). Note that in a two-dimensional space (i.e., $\mathbb{B}^2$) the described convex hull algorithm has complexity $O(N \log N)$ and is hence very efficient. Next, denote the set of points on the convex hull of the class labeled by $+1$ (-1) as CH_+ (CH_-). A minimum distance pair x_+, x_- can be found as

$$d(x_+, x_-) = \min_{x \in CH_+, y \in CH_-} d(x, y). \tag{26}$$

Then, the hyperbolic midpoint of x_+, x_- corresponds to the reference point $p = \gamma_{x_+ \to x_-}(0.5)$ (see Figure 6). Our strategy of learning p works well on real world data set, see Section VI.

It is important to point out that the computational cost of learning a reference point scales with the dimension of the ambient space, which is not desirable. To resolve this issue we propose a different type of hyperbolic perceptron that operates in the hyperboloid model [34] and discuss it in Appendix 5 for completeness.

E. The Poincaré SVM

We conclude our theoretical analysis by describing how to formulate and solve SVMs in the Poincaré ball model with performance guarantees. For simplicity, we only consider binary classification. Techniques for dealing with multiple classes are given in Section VI.

When data points from two different classes are linearly separable the goal of SVM is to find a "max-margin hyperplane" that correctly classifies all data points and has the maximum distance from the nearest point. This is equivalent to selecting

---

Algorithm 2: Poincaré Graham scan

Input: Data points $X = \{x_i\}_{i=1}^N \in \mathbb{B}^2$.
1 Initialization: Set $S = \emptyset$.
2 Find p_0 with minimum y coordinate. If multiple options exist, choose the smallest x-coordinate one.
3 In the $T_{p_0}\mathbb{B}^2$, sort X by the angle between $\log_{p_0}(x_i)$ and the x-axis (counterclockwise, ascending).
4 Append an additional p_0 to the end of X.
5 **for** $x \in X$ **do**
6   **while** $|S| > 1$ *and*
    outer-product$\left(\log_{S[-2]}(S[-1]), \log_{S[-2]}(x)\right) < 0$
7   Pop S;
8   **end**
9   Push x to S;
10 **end**
Output: S.

---

two parallel hyperplanes with maximum distance that can separate two classes. Following this approach and assuming that the data points are normalized, we can choose these two hyperplanes as $\langle \log_p(x_i), w \rangle = 1$ and $\langle \log_p(x_i), w \rangle = -1$ with $w \in T_p\mathbb{B}^n$. Points lying on these two hyperplanes are referred to as support vectors following the convention for Euclidean SVM. They are critical for the process of selecting w.

Let $v_i = \log_p(x_i) \in T_p\mathbb{B}^n$ be such that $|\langle v_i, w \rangle| = 1$. Therefore, by Cauchy-Schwartz inequality the support vectors satisfy

$$\|v_i\|\|w\| \geq |\langle v_i, w \rangle| = 1 \Rightarrow \|v_i\| \geq 1/\|w\|. \tag{27}$$

Combing the above result with (14) leads to a lower bound on the distance of a data point x_i to the hyperplane $H_{w,p}$

$$d(x_i, H_{w,p}) \geq \sinh^{-1}\left(\frac{2\tanh(\sigma_p/2\|w\|)}{1 - \tanh^2(\sigma_p/2\|w\|)}\right), \tag{28}$$

where equality is achieved for $v_i = kw$ ($k \in \mathbb{R}$). Thus we can obtain a max-margin classifier in the Poincaré ball that can correctly classify all data points by maximizing the lower bound in (28). Through a sequence of simple reformulations, the optimization problem of maximizing the lower bound (28) can be cast as an easily-solvable **convex** problem described in Theorem 4.2.

Theorem 4.2: Maximizing the margin (28) is equivalent to solving the convex problem of either primal (P) or dual (D) form:

$$(P)\ \min_w \frac{1}{2}\|w\|^2 \quad \text{s.t. } y_i\langle v_i, w \rangle \geq 1\ \forall i \in [N]; \tag{29}$$

$$(D)\ \max_{\alpha \geq 0} \sum_{i=1}^N \alpha_i - \frac{1}{2}\left\|\sum_{i=1}^N \alpha_i y_i v_i\right\|^2 \quad \text{s.t. } \sum_{i=1}^N \alpha_i y_i = 0, \tag{30}$$

which is guaranteed to achieve a global optimum with linear convergence rate by stochastic gradient descent.

The Poincaré SVM formulation from Theorem 4.2 is *inherently different* from the only other known hyperbolic SVM

approach [9]. There, the problem is nonconvex and thus does not offer convergence guarantees to a global optimum when using projected gradient descent. In contrast, since both (P) and (D) are smooth and strongly convex, variants of stochastic gradient descent is guaranteed to reach a global optimum with a linear convergence rate. This makes the Poincaré SVM numerically more stable and scalable to millions of data points. Another advantage of our formulation is that a solution to (D) directly produces the support vectors, i.e., the data points with corresponding $\alpha_i \neq 0$ that are critical for the classification problem.

When two data classes are not linearly separable (i.e., the problem is soft- rather than hard-margin), the goal of the SVM method is to maximize the margin while controlling the number of misclassified data points. Below we define a soft-margin Poincaré SVM that trades-off the margin and classification accuracy.

Theorem 4.3: Solving soft-margin Poincaré SVM is equivalent to solving the convex problem of either primal (P) or dual (D) form:

$$(P) \min_w \frac{1}{2}\|w\|^2 + C\sum_{i=1}^N \max(0, 1 - y_i\langle v_i, w\rangle); \quad (31)$$

$$(D) \max_{0 \leq \alpha \leq C} \sum_{i=1}^N \alpha_i - \frac{1}{2}\left\|\sum_{i=1}^N \alpha_i y_i v_i\right\|^2 \text{ s.t. } \sum_{i=1}^n \alpha_i y_i = 0, \quad (32)$$

which is guaranteed to achieve a global optimum with sublinear convergence rate by stochastic gradient descent.

The algorithmic procedure behind the soft-margin Poincaré SVM is depicted in Algorithm 3.

Algorithm 3: Soft-margin Poincaré SVM

Input: Data points $\{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, reference point p, tolerance ε, maximum iteration number T.

1 Initialization: $w_0 \leftarrow 0$, $f(w_0) = NC$.
2 **for** $t = 1, 2, \ldots$ **do**
3      $f(w_t) = \frac{1}{2}\|w_t\|^2 + C\sum_{i=1}^N \max(0, 1 - y_i\langle v_i, w_t\rangle)$.
4   **if** $f(w_t)$ - $f(w_{t-1}) \leq \varepsilon$ *or* $t > T$ **then**
5   **break;**
6   **end**
7   Sample (x_t, y_t) from data set randomly and compute $v_t = \log_p(x_t)$.
8   **if** $1 - y_t\langle v_t, w_{t-1}\rangle < 0$ **then**
9          $\nabla w = w_{t-1}$.
10   **else**
11         $\nabla w = w_{t-1} - NCy_t v_t$.
12   **end**
13     $w_t = w_{t-1} - \frac{1}{t+1000}\nabla w$.
14 **end**

V. Perceptron variants in hyperbolic spaces

A. The Poincaré Second-Order Perceptron

The reason behind our interest in the second-order perceptron is that it leads to fewer mistakes during training compared to the classical perceptron. It has been shown in [18] that the error bounds have corresponding statistical risk bounds in online learning settings, which strongly motivates the use of second-order perceptrons for online classification. The performance improvement of the modified perceptron comes from accounting for second-order data information, as the standard perceptron is essentially a gradient descent (first-order) method.

Equipped with the key idea of our unified analysis, we compute the scaled tangent vectors $z_i = \eta_i v_i = \eta_i \log_p(x_i)$. Following the same idea, we can extend the second order perceptron to Poincaré ball model. Our Poincaré second-order perceptron is described in Algorithm 4 which has the following theoretical guarantee.

Algorithm 4: Poincaré second order perceptron

Input: Data points $\{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, reference point p, parameter $a > 0$.

1 Initialization: $\xi_0 \leftarrow 0$, $X_0 = \emptyset$, $k = 1$.
2 **for** $t = 1, 2, \ldots$ **do**
3   Get (x_t, y_t), compute z_t, set $S_t \leftarrow [X_{k-1}\ z_t]$.
4   Predict $\hat{y}_t = sign(\langle w_t, z_t\rangle)$, where $w_t = (aI + S_t S_t^T)^{-1}\xi_{k-1}$.
5   **if** $\hat{y}_i \neq y_i$ **then**
6        $\xi_k = \xi_{k-1} + y_t z_t$, $X_k \leftarrow S_t$, $k \leftarrow k + 1$.
7   **end**
8 **end**

Theorem 5.1: For all sequences $((x_1, y_1), (x_2, y_2), \ldots)$ with assumption 4.1, the total number of mistakes k for Poincaré second order perceptron satisfies

$$k \leq \frac{1}{\sinh(\varepsilon)}\sqrt{(a + \lambda_{w^\star})\left(\sum_{j \in [n]} \log(1 + \frac{\lambda_j}{a})\right)}, \quad (33)$$

where $\lambda_w = w^T X_k X_k^T w$ and λ_j are the eigenvalues of $X_k X_k^T$.

The bound in Theorem 5.1 has a form that is almost identical to that of its Euclidean counterpart [16]. However, it is important to observe that the geometry of the Poincaré ball model plays a important role when evaluating the eigenvalues λ_j and λ_w. Another important observation is that our tangent-space analysis is not restricted to first-order analysis of perceptrons.

B. The Poincaré Strategic Perceptron

Strategic classification deals with the problem of learning a classifier when the learner relies on data that is provided by strategic agents in an online manner, meaning that the observed data can be manipulated in a controlled manner, based on

the utilities of agents. This is a challenging yet important problem in practice because the learner can only observe potentially modified data; however, it has been shown in [17] that standard perceptron algorithms in this setting can fail to converge and may make an unbounded number of mistakes in Euclidean spaces, even when a perfect classifier exists. The authors of [17] thus proposes a modified version of Euclidean space perceptrons to deal with this problem. Following the same idea, we can extend this strategic perceptron to Poincaré ball model and establish performance guarantees.

Before introducing our algorithm, we introduce some additional notation and discuss relevant modeling assumptions. We again consider a binary classification problem, but this time in a strategic setting. In this case, true (unmanipulated) features $\{x_t\}$ from different agents arrive in order, with the corresponding binary labels $\{y_t\}$ in $\{+1, -1\}$. The assumption is that all agents want to be classified as positive regardless of their true labels; to achieve this goal, they can choose to manipulate their data to change the classifier, and the decision if to manipulate or not is made based on the gain and the cost of manipulation. The classifier can only receive observed data points $\{z_t\}$, which are either manipulated or not. The first assumption in this problem is that all agents are assumed to be utility maximizers, where utility is defined as the gain minus cost. More precisely, we assume for simplicity that all agents have a gain equal to 1 when being classified as positive, and 0 otherwise. To be more specific, an agent with true data x_t will modify their data to $z_t = \arg\max_z(\text{val}(z) - \text{cost}(x_t, z))$, where $\text{val}(z_t) = 1$ if z_t is classified as "positive" by the current classifier and $\text{val}(z_t) = 0$ otherwise; here, $\text{cost}(x_t, z_t)$ refers to the cost of changing (manipulating) x_t to z_t. Second, the cost we consider is proportional to the magnitude of movement between $u_t = \log_p(x_t)$ and $v_t = \log_p(z_t)$ in the Poincaré ball model; i.e., $\text{cost}(x_t, z_t) = s\sqrt{g_p(u_t - v_t, u_t - v_t)} = s\sigma_p\|u_t - v_t\|$, where s is the cost per unit of movement and $\alpha = 1/s$ is the largest amount of movement that a rational agent would take. For simplicity, we also assume that p is known in advance, otherwise we can estimate p from the data.

One simple example illustrating why Algorithm 1 can fail to converge in a strategic classification setting is as follows.

Consider $A = (-\tanh(1), 0), B = (0, -\tanh(1)), C = \left(-\frac{\tanh(\sqrt{5}/2)}{\sqrt{5}}, -\frac{2\tanh(\sqrt{5}/2)}{\sqrt{5}}\right)$ from $\mathbb{B}_1^2$ and let $p = \mathbf{0}$, where the points A, C are classified as negative, and B as positive and $\alpha = 1$. Suppose A is the first data point to arrive, following B and C. Upon observing A, w_2 is set to $\left(\frac{2\tanh(1)}{1-\tanh^2(1)}, 0\right)$ based on (18). Observing B will not change the classifier since it is correctly classified as positive. The next input, C, can be manipulated from $\left(-\frac{\tanh(\sqrt{5}/2)}{\sqrt{5}}, -\frac{2\tanh(\sqrt{5}/2)}{\sqrt{5}}\right)$ to $(0, -\tanh(1))$ to confuse the current classifier to misclassify it as positive. In this case the manipulation cost is 1 which is within the budget. After the learner receives the true label of C, it performs the update $w_3 = \left(\frac{2\tanh(1)}{1-\tanh^2(1)}, \frac{2\tanh(1)}{1-\tanh^2(1)}\right)$. However, when B is reexamined next, it will be classified as negative and no manipulation will be possible because the

minimum cost for B to be classified as positive is $\sqrt{2}$. Hence, the learner will perform the update $w_4 = \left(\frac{2\tanh(1)}{1-\tanh^2(1)}, 0\right) = w_2$. This causes an infinite loop of updates and the upper bound for the updates on the weight vector in Theorem 4.1 does not hold. Although in this case a perfect classifier with $w = \left(\frac{2\tanh(1)}{1-\tanh^2(1)}, \frac{\tanh(1)}{1-\tanh^2(1)}\right)$ exists, Algorithm 1 cannot successfully identify it.

For this reason, we propose the following Poincaré strategic perceptron described in Algorithm 5, based on the idea described in [17].

Algorithm 5: Poincaré Strategic Perceptron

Input: Observed data points $\{z_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, reference point p, manipulation budget α.

1 Initialization: $w_1 = \mathbf{0}, k = 1$.
2 **for** $t = 1, 2, \dots$ **do**
3   Get (z_t, y_t); compute $v_t = \log_p(z_t)$ and
$$\eta_t = \frac{2\tanh\left(\frac{\sigma_p\|v_t\|}{2}\right)}{\left(1-\tanh\left(\frac{\sigma_p\|v_t\|}{2}\right)^2\right)\|v_t\|}.$$
4   **if** $w_k = \mathbf{0}$ **then**
5   Predict $\hat{y}_t = +1$.
6   **if** $\hat{y}_t \neq y_t$ **then**
7           $w_{k+1} = w_k - \eta_t v_t, k \leftarrow k+1$.
8   **end**
9   **end**
10   **else**
11   Predict $\hat{y}_t = \text{sgn}\left(\frac{\langle w_k, v_t\rangle}{\|w_k\|} - \frac{\alpha}{\sigma_p}\right)$.
12   **if** $\hat{y}_t \neq y_t$ **then**
13           $\tilde{v}_t =$
$$\begin{cases} v_t - \frac{\alpha w_k}{\sigma_p\|w_k\|}, & \text{if } y_t = -1, \frac{\langle w_k, v_t\rangle}{\|w_k\|} = \frac{\alpha}{\sigma_p} \\ v_t, & \text{otherwise.} \end{cases}$$
14           $w_{k+1} = w_k + \eta_t y_t \tilde{v}_t, k \leftarrow k+1$.
15   **end**
16   **end**
17 **end**

Theorem 5.2: If Assumption 4.1 holds for unmanipulated data points $\{x_i\}$, then Algorithm 5 makes at most $\left(\frac{2R_p\sigma_p + \alpha(1-R_p^2)}{\sigma_p(1-R_p^2)\sinh(\varepsilon)}\right)^2$ mistakes in the strategic setting for a given cost parameter α. Here, $R_p = \frac{\|p\|+R}{1+\|p\|R}$.

The proof of Theorem 5.2 is delegated to Appendix 7, but two observations are in place to explain the intuition behind Algorithm 5:

- The updated decision hyperplane at each step will take the form $\frac{\langle w_k, v_t\rangle}{\|w_k\|} - \frac{\alpha}{\sigma_p} = 0$ and v_t is a manipulated data point only if $\frac{\langle w_k, v_t\rangle}{\|w_k\|} = \frac{\alpha}{\sigma_p}$. This is due to the fact that all agents are utility maximizers, so they will only move in the direction of w_k if needed. More specifically,

$$v_t = \begin{cases} u_t + \left(\frac{\alpha}{\sigma_p} - \frac{\langle w_k, u_t\rangle}{\|w_k\|}\right)\frac{w_k}{\|w_k\|}, & \text{if } 0 \leq \frac{\langle w_k, u_t\rangle}{\|w_k\|} \leq \frac{\alpha}{\sigma_p} \\ u_t, & \text{otherwise.} \end{cases}$$
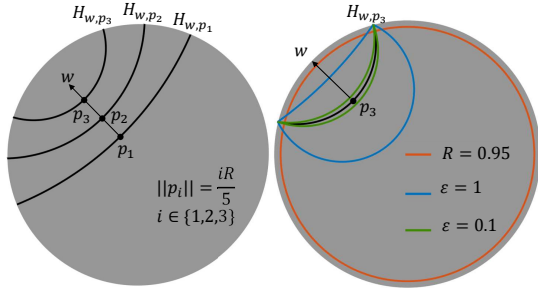
Fig. 7. (left) Decision boundaries for different choices of p. (right) Geometry of different choices of margin ε.

- No observed data point will fall in the region $0 < \frac{\langle w_k, v_t \rangle}{\|w_k\|} < \frac{\alpha}{\sigma_p}$. This can be shown by contradiction. A v_t lying in this region must be either manipulated or not. If v_t is manipulated, this implies that the agent is not rational as the point is still classified as negative after manipulation; if v_t is not manipulated, this also implies that the agent is not rational as the cost of modifying the data to be classified as positive is less than α, which is within the budget. Since v_t can either be manipulated or not, this shows that no observed data point satisfies $0 < \frac{\langle w_k, v_t \rangle}{\|w_k\|} < \frac{\alpha}{\sigma_p}$.

For the cases where the manipulation budget α is unknown, one can follow the same estimation procedure presented in [17] which is independent of the curvature of the hyperbolic space.

VI. Experiments

To put the performance of our proposed algorithms in the context of existing works on hyperbolic classification, we perform extensive numerical experiments on both synthetic and real-world data sets. In particular, we compare our Poincaré perceptron, second-order perceptron and SVM method with the hyperboloid SVM of [9] and the Euclidean SVM. Detailed descriptions of the experimental settings are provided in the Appendix.

A. Synthetic Data Sets

In the first set of experiments, we generate N points uniformly at random on the Poincaré disk and perform binary classification task. To satisfy our Assumption 4.1, we restrict the points to have norm at most $R = 0.95$ (boundedness condition). For a decision boundary $H_{w,p}$, we remove all points within margin ε (margin assumption). Note that the decision boundary looks more "curved" when $\|p\|$ is larger, which makes it more different from the Euclidean case (Figure 7). When $\|p\| = 0$ then the optimal decision boundary is also linear in Euclidean sense. On the other hand, if we choose $\|p\|$ too large then it is likely that all points are assigned with the same label. Hence, we consider the case $\|p\| = \frac{R}{5}, \frac{2R}{5}$ and $\frac{3R}{5}$ and let the direction of w to be generated uniformly at random. Results for case $\|p\| = \frac{2R}{5}$ are demonstrated in Figure 2 while the others are in Figure 8. All results are averaged over 20 independent runs.

We first vary N from 10^3 to 10^6 and fix $(d, \varepsilon) = (2, 0.01)$. The accuracy and time complexity are shown in Figure 8 (e)-(h). One can clearly observe that the Euclidean SVM fails to achieve a 100% accuracy as data points are not linearly separable in the Euclidean, but only in hyperbolic sense. This phenomenon becomes even more obvious when $\|p\|$ increases due to the geometry of Poincaré disk (Figure 7). On the other hand, the hyperboloid SVM is not scalable to accommodate such a large number of points. As an example, it takes 6 *hours* (9 hours for the case $\|p\| = \frac{2R}{5}$, Figure 2) to process $N = 10^6$ points; in comparison, our Poincaré SVM only takes 1 *minute*. Hence, only the Poincaré SVM is highly scalable and offers the highest accuracy achievable.

Next we vary the margin ε from 1 to 0.001 and fix $(d, N) = (2, 10^5)$. The accuracy and time complexity are shown in Figure 8 (i)-(l). As the margin reduces, the accuracy of the Euclidean SVM deteriorates. This is again due to the geometry of Poincaré disk (Figure 7) and the fact that the classifier needs to be tailor-made for hyperbolic spaces. Interestingly, the hyperboloid SVM performs poorly for $\|p\| = 3R/5$ and a margin value $\varepsilon = 1$, as its accuracy is significantly below 100%. This may be attributed to the fact that the cluster sizes are highly unbalanced, which causes numerical issue with the underlying optimization process. Once again, the Poincaré SVM outperforms all other methods in terms of accuracy and time complexity.

Finally, we examined the influence of the data point dimension on the performance of the classifiers. To this end, we varied the dimension d from 2 to $1,000$ and fixed $(N, \varepsilon) = (10^5, 0.01)$. The accuracy and time complexity results are shown in Figure 8 (a)-(d). Surprisingly, the hyperboloid SVM fails to learn well when d is large and close to $1,000$. This again reaffirms the importance of the convex formulation of our Poincaré SVM, which is guaranteed to converge to a global optimum independent of the choice d, N and ε. We also find that Euclidean SVM improves its performance as d increases, albeit at the price of high execution time.

We now we turn our attention to the evaluation of our perceptron algorithms which are online algorithms in nature. Results are summarized in Table II. There, one can observe that the Poincaré second-order perceptron requires a significantly smaller number of updates compared to Poincaré perceptron, especially when the margin is small. This parallels the results observed in Euclidean spaces [16]. Furthermore, we validate our Theorem 4.1 which provide an upper bound on the number of updates for the worst case.

B. Real-World Data Sets

For real-world data sets, we choose to work with more than two collections of points. To enable K-class classification for $K \geq 2$, we use K binary classifiers that are independently trained on the same training set to separate each single class from the remaining classes. For each classifier, we transform the resulting prediction scores into probabilities via the Platt scaling technique [35]. The predicted labels are then decided
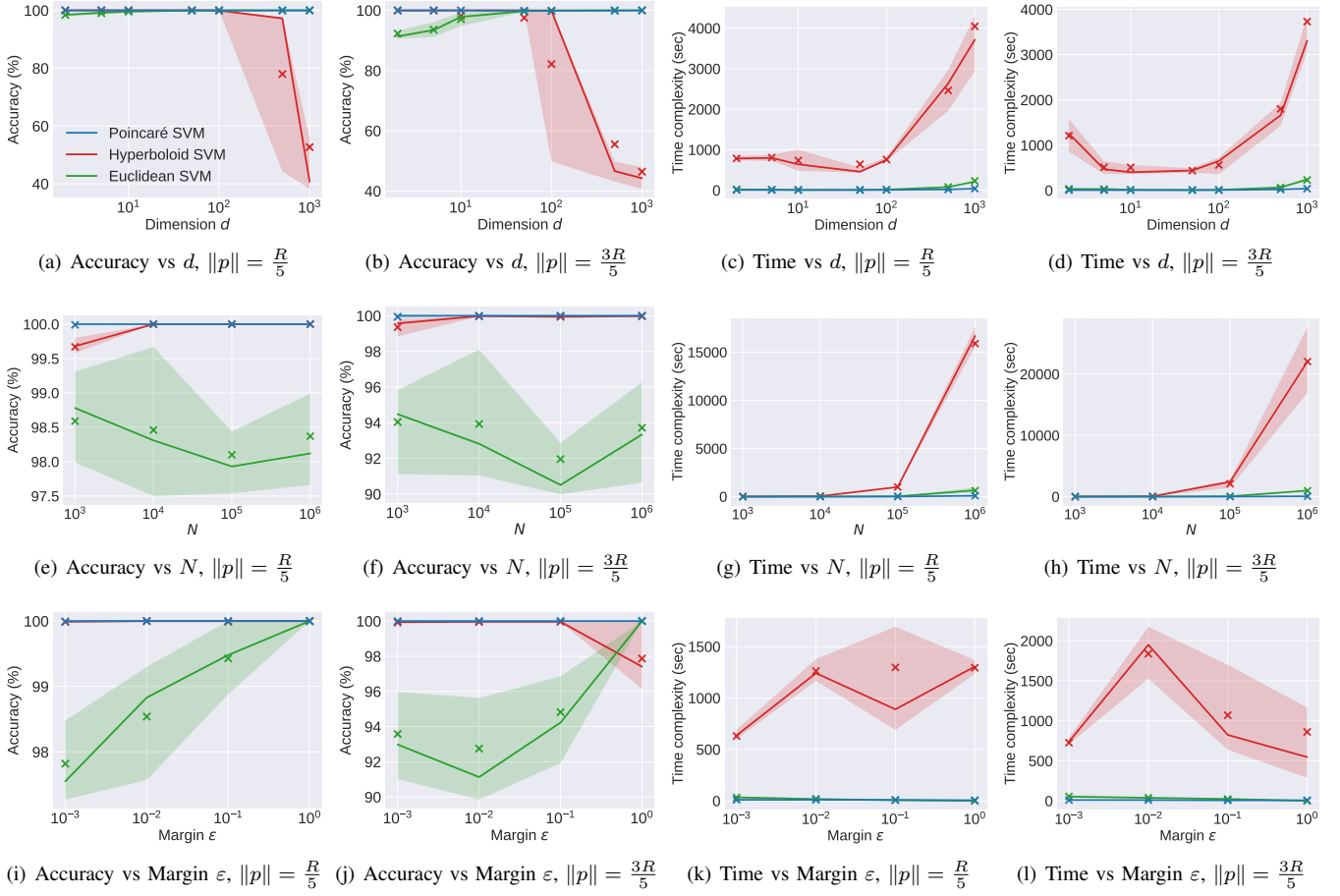
Fig. 8. Experiments on synthetic data sets and $\|p\| \in \{\frac{R}{5}, \frac{3R}{5}\}$. The upper and lower boundaries of the shaded region represent the first and third quantile, respectively. The line itself corresponds to the medium (second quantile) and the marker $\times$ indicates the mean. The first two columns plot the accuracy of the SVM methods while the last two columns plot the corresponding time complexity. For the first row we vary the dimension d from 2 to $1,000$. For the second row we vary the number of points N from 10^3 to 10^6. In the third row we vary the margin ε from 1 to 0.001. The default setting for (d, N, ε) is $(2, 10^5, 0.01)$.

TABLE II
AVERAGED NUMBER OF UPDATES FOR THE POINCARÉ SECOND-ORDER PERCEPTRON (S-PERCEPTRON) AND POINCARÉ PERCEPTRON FOR A VARYING MARGIN ε AND FIXED $(N, d) = (10^4, 10)$. BOLD NUMBERS INDICATE THE BEST RESULTS, WITH THE MAXIMUM NUMBER OF UPDATES OVER 20 RUNS SHOWN IN PARENTHESIS. ALSO SHOWN IS A THEORETICAL UPPER BOUND ON THE NUMBER OF UPDATES FOR THE POINCARÉ PERCEPTRON BASED ON THEOREM 4.1.

	Margin ε	1	0.1	0.01	0.001
	S-perceptron	**26**	**82**	**342**	**818**
		(34)	(124)	(548)	(1,505)
$\|p\| = \frac{R}{5}$	perceptron	51	1,495	1.96×10^4	1.34×10^5
		(65)	(2,495)	(3.38×10^4)	(3.56×10^5)
	Theorem 4.1	594	81,749	8.2×10^6	8.2×10^8
	S-perceptron	**29**	**101**	**340**	**545**
		(41)	(159)	(748)	(986)
$\|p\| = \frac{3R}{5}$	perceptron	82	1,158	1.68×10^4	1.46×10^5
		(138)	(2,240)	(6.65×10^4)	(7.68×10^5)
	Theorem 4.1	3,670	5.05×10^5	5.07×10^7	5.07×10^9

by a maximum a posteriori criteria based on the probability of each class.

The data sets of interest include Olsson's single-cell expression profiles, containing single-cell (sc) RNA-seq expression data from 8 classes (cell types) [19], CIFAR10, containing images from common objects falling into 10 classes [20], Fashion-MNIST, containing Zalando's article images with 10 classes [21], and mini-ImageNet, containing subsamples of images from original ImageNet data set and containing 20 classes [22]. Following the procedure described in [36], [37], we embed single-cell, CIFAR10 and Fashion-MNIST data sets into a 2-dimensional Poincaré disk, and mini-ImageNet into a 512-dimensional Poincaré ball, all with curvature -1 (Note that our methods can be easily adapted to work with other curvature values as well), see Figure 1. Other details about the data sets including the number of samples and splitting strategy of training and testing set is described in the Appendix. Since the real-world embedded data sets are not linearly separable we only report classification results for the Poincaré SVM method.

We compare the performance of the Poincaré SVM, Hyperbolid SVM and Euclidean SVM for soft-margin classification

TABLE III
PERFORMANCE OF THE SVM ALGORITHMS GENERATED BASED ON 5
INDEPENDENT TRIALS.

	Algorithm	Accuracy (%)	Time (sec)
Olsson's scRNA-seq	Euclidean SVM	71.59 ± 0.00	0.06 ± 0.00
	Hyperboloid SVM	71.97 ± 0.54	4.49 ± 0.05
	Poincaré SVM	$\mathbf{89.77 \pm 0.00}$	0.16 ± 0.00
CIFAR10	Euclidean SVM	47.66 ± 0.00	26.03 ± 5.29
	Hyperboloid SVM	89.87 ± 0.01	707.69 ± 15.33
	Poincaré SVM	$\mathbf{91.84 \pm 0.00}$	45.55 ± 0.43
Fashion-MNIST	Euclidean SVM	39.91 ± 0.01	32.70 ± 11.11
	Hyperboloid SVM	76.78 ± 0.06	898.82 ± 11.35
	Poincaré SVM	$\mathbf{87.82 \pm 0.00}$	67.28 ± 8.63
mini-ImageNet	Euclidean SVM	63.33 ± 0.00	7.94 ± 0.09
	Hyperboloid SVM	31.75 ± 1.91	618.23 ± 10.92
	Poincaré SVM	$\mathbf{63.59 \pm 0.00}$	18.78 ± 0.42

of the above described data points. For the Poincaré SVM the reference point $p^{(i)}$ for each binary classifier is estimated via our technique introduced in Section IV-D. The resulting classification accuracy and time complexity are shown in Table III. From the results one can easily see that our Poincaré SVM consistently achieves the best classification accuracy over all data sets while being roughly 10x faster than the hyperboloid SVM. It is also worth pointing out that for most data sets embedded into the Poincaré ball model, the Euclidean SVM method does not perform well as it does not exploit the geometry of data; however, the good performance of the Euclidean SVM algorithm on mini-ImageNet can be attributed to the implicit Euclidean metric used in the embedding framework of [37]. Note that since the Poincaré SVM and Euclidean SVM are guaranteed to achieve a global optimum, the standard deviation of classification accuracy is zero.

VII. CONCLUSION

We generalized classification algorithms such as the second-order& strategic perceptron and the SVM method to Poincaré balls. Our Poincaré classification algorithms comes with theoretical guarantees that ensure convergence to a global optimum. Our Poincaré classification algorithms were validated experimentally on both synthetic and real-world datasets. The developed methodology appears to be well-suited for extensions to other machine learning problems in hyperbolic spaces, an example of which is classification in mixed constant curvature spaces [34].

ACKNOWLEDGMENT

REFERENCES

[1] E. Chien, C. Pan, P. Tabaghi, and O. Milenkovic, "Highly scalable and provably accurate classification in poincaré balls," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 61–70.

[2] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, 2010.

[3] R. Sarkar, "Low distortion delaunay embedding of trees in hyperbolic plane," in *International Symposium on Graph Drawing*. Springer, 2011, pp. 355–366.

[4] F. Sala, C. De Sa, A. Gu, and C. Re, "Representation tradeoffs for hyperbolic embeddings," in *International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 4460–4469.

[5] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in Neural Information Processing Systems*, 2017, pp. 6338–6347.

[6] F. Papadopoulos, R. Aldecoa, and D. Krioukov, "Network geometry inference using common neighbors," *Physical Review E*, vol. 92, no. 2, 2015.

[7] A. Tifrea, G. Becigneul, and O.-E. Ganea, "Poincaré glove: hyperbolic word embeddings," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Ske5r3AqK7

[8] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, no. 2, pp. 215–245, 1995.

[9] H. Cho, B. DeMeo, J. Peng, and B. Berger, "Large-margin classification in hyperbolic space," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1832–1840.

[10] N. Monath, M. Zaheer, D. Silva, A. McCallum, and A. Ahmed, "Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 714–722.

[11] M. Weber, M. Zaheer, A. S. Rawat, A. Menon, and S. Kumar, "Robust large-margin learning in hyperbolic space," in *Advances in Neural Information Processing Systems*, 2020.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5345–5355.

[14] R. Shimizu, Y. Mukuta, and T. Harada, "Hyperbolic neural networks++," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Ec85b0tUwbA

[15] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[16] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "A second-order perceptron algorithm," *SIAM Journal on Computing*, vol. 34, no. 3, pp. 640–668, 2005.

[17] S. Ahmadi, H. Beyhaghi, A. Blum, and K. Naggita, "The strategic perceptron," in *Proceedings of the 22nd ACM Conference on Economics and Computation*, 2021, pp. 6–25.

[18] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "On the generalization ability of online learning algorithms," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2050–2057, 2004.

[19] A. Olsson, M. Venkatasubramanian, V. K. Chaudhri, B. J. Aronow, N. Salomonis, H. Singh, and H. L. Grimes, "Single-cell analysis of mixed-lineage states leading to a binary cell fate choice," *Nature*, vol. 537, no. 7622, pp. 698–702, 2016.

[20] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[21] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[22] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rJY0-Kcll

[23] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 547–555.

[24] M. Hardt, N. Megiddo, C. Papadimitriou, and M. Wootters, "Strategic classification," in *Proceedings of the 2016 ACM conference on innovations in theoretical computer science*, 2016, pp. 111–122.

[25] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 8230–8241.

[26] Y. Nagano, S. Yamaguchi, Y. Fujita, and M. Koyama, "A wrapped normal distribution on hyperbolic space for gradient-based learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4693–4702.

[27] E. Mathieu, C. L. Lan, C. J. Maddison, R. Tomioka, and Y. W. Teh, "Continuous hierarchical representations with poincaré variational auto-encoders," in *Advances in Neural Information Processing Systems*, 2019.

[28] O. Skopek, O.-E. Ganea, and G. Bécigneul, "Mixed-curvature variational autoencoders," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1g6xeSKDS

[29] A. A. Ungar, *Analytic hyperbolic geometry and Albert Einstein's special theory of relativity*. World Scientific, 2008.

[30] J. Vermeer, "A geometric interpretation of ungar's addition and of gyration in the hyperbolic plane," *Topology and its Applications*, vol. 152, no. 3, pp. 226–242, 2005.

[31] J. G. Ratcliffe, S. Axler, and K. Ribet, *Foundations of hyperbolic manifolds*. Springer, 2006, vol. 149.

[32] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Info. Pro. Lett.*, vol. 1, pp. 132–133, 1972.

[33] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[34] P. Tabaghi, C. Pan, E. Chien, J. Peng, and O. Milenković, "Linear classifiers in product space forms," *arXiv preprint arXiv:2102.10204*, 2021.

[35] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[36] A. Klimovskaia, D. Lopez-Paz, L. Bottou, and M. Nickel, "Poincaré maps for analyzing complex hierarchies in single-cell data," *Nature communications*, vol. 11, no. 1, pp. 1–9, 2020.

[37] V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, and V. Lempitsky, "Hyperbolic image embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6418–6428.

[38] J. W. Cannon, W. J. Floyd, R. Kenyon, W. R. Parry *et al.*, "Hyperbolic geometry," *Flavors of geometry*, vol. 31, no. 59-115, p. 2, 1997.

[39] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950.

PROOF OF LEMMA 4.2

By the definition of Möbius addition, we have

$$a \oplus b = \frac{1 + 2a^T b + \|b\|^2}{1 + 2a^T b + \|a\|^2 \|b\|^2} a + \frac{1 - \|a\|^2}{1 + 2a^T b + \|a\|^2 \|b\|^2} b.$$

Thus,

$$\|a \oplus b\|^2 = \frac{\|(1 + 2a^T b + \|b\|^2)a + (1 - \|a\|^2)b\|^2}{\left(1 + 2a^T b + \|a\|^2 \|b\|^2\right)^2}$$

$$= \frac{\|a + b\|^2 \left(1 + \|b\|^2 \|a\|^2 + 2a^T b\right)}{\left(1 + 2a^T b + \|a\|^2 \|b\|^2\right)^2} = \frac{\|a + b\|^2}{1 + 2a^T b + \|a\|^2 \|b\|^2}. \tag{34}$$

Next, use $\|b\| = r$ and $a^T b = r\|a\| \cos(\theta)$ in the above expression:

$$\|a \oplus b\|^2 = \frac{\|a\|^2 + 2r\|a\| \cos(\theta) + r^2}{1 + 2r\|x\| \cos(\theta) + \|a\|^2 r^2}$$

$$= 1 - \frac{(1 - r^2)(1 - \|a\|^2)}{1 + 2r\|x\| \cos(\theta) + \|a\|^2 r^2}. \tag{35}$$

The function in (35) attains its maximum at $\theta = 0$ and $r = R$. We also observe that (34) is symmetric in a, b. Thus, the same argument holds for $\|b \oplus a\|$.

CONVEX HULL ALGORITHMS IN POINCARÉ BALL MODEL

We introduce next a generalization of the Graham scan and Quickhull algorithms for the Poincaré ball model. In a nutshell, we replace lines with geodesics and vectors $\overrightarrow{AB}$ with tangent vectors $\log_A(B)$ or equivalently $(-A) \oplus B$. The pseudo code for the Poincaré version of the Graham scan is listed in Algorithm 2, while Quickhull is listed in Algorithm 6 (both for the two-dimensional case). The Graham scan has worst-case time complexity $O(N \log N)$, while Quickhull has complexity $O(N \log N)$ in expectation and $O(N^2)$ in the worst-case. The Graham scan only works for two-dimensional points while Quickhull can be generalized for higher dimensions [33].

Algorithm 6: Poincaré Quickhull

Input: Data points $X = \{x_i\}_{i=1}^N \in \mathbb{B}^2$.
1 Initialization: Set $S = \emptyset$.
2 Find the left- and right-most points A, B. Add them to S.
   // $\gamma_{A \to B}$ splits the remaining points into two groups, S_1 and S_2.
3 $p_0 = \gamma_{A \to B}(\frac{1}{2})$, $v = \log_{p_0}(B)$, $w = v^\perp$;
4 **for** $x \in X$ **do**
5   **if** $\langle w, \log_{p_0}(x) \rangle \geq 0$ **then**
6   Add x to S_1;
7   **end**
8   **if** $\langle w, \log_{p_0}(x) \rangle \leq 0$ **then**
9   Add x to S_2;
10   **end**
11 **end**
12 S_L = FindHull(S_1,A,B), S_R = FindHull(S_2,B,A);
   Output: Union(S_L,S_R).

HYPERBOLOID PERCEPTRON

The hyperboloid model and the definition of hyperplanes. The hyperboloid model $\mathbb{L}_c^n$ is another model for representing points in a n-dimensional hyperbolic space with curvature $-c$ $(c > 0)$. Specifically, it is a Riemannian manifold $(\mathbb{L}_c^n, g^{\mathbb{L}})$ for which

$$\mathbb{L}_c^n = \left\{ x \in \mathbb{R}^{n+1} : [x, x] = -\frac{1}{c}, x_0 > 0 \right\},$$

$$g^{\mathbb{L}}(u, v) = [u, v] = u^\top H v, \ u, v \in T_p \mathbb{L}_c^n, \ H = \begin{pmatrix} -1 & 0^\top \\ 0 & I_d \end{pmatrix}.$$

Algorithm 7: FindHull

Input: Set of points S_k, points P and Q.

1 **if** $|S_k|$ *is* 0 **then**
  | **Output:** $\emptyset$
2 **end**
3 Find the furthest point F from $\gamma_{P \to Q}$.
4 Partition S_k into three set S_0, S_1, S_2: S_0 contains points in ΔPFQ; S_1 contains points outside of $\gamma_{P \to F}$; and S_2 contains points outside of $\gamma_{F \to Q}$.
5 S_L = FindHull(S_1,P,F), S_R = FindHull(S_2,F,Q)
Output: Union(F,S_L,S_R);

Throughout the remainder of this section, we restrict our attention to $c = 1$; all results can be easily generalized to arbitrary values of c.

We make use of the following bijection between the Poincaré ball model and hyperboloid model, given by

$$(x_0, \ldots, x_n) \in \mathbb{L}^n \Leftrightarrow \left(\frac{x_1}{1 + x_0}, \ldots, \frac{x_n}{1 + x_0} \right) \in \mathbb{B}^n. \tag{36}$$

Additional properties of the hyperboloid model can be found in [38].

The recent work [34] introduced the notion of a hyperboloid hyperplane of the form

$$H_w = \{x \in \mathbb{L}^n : \text{asinh}([w, x]) = 0\}$$
$$= \{x \in \mathbb{L}^n : [w, x] = 0\} \tag{37}$$

where $w \in \mathbb{R}^{n+1}$ and $[w, w] = 1$. The second equation is a consequence of the fact that $\text{asinh}(\cdot)$ is an increasing function and will not change the sign of the argument. Thus the classification result based on the hyperplane is given by $\text{sgn}([w, x])$ for some weight vector w, as shown in Figure 9.

The hyperboloid perceptron. The definition of a linear classifier in the hyperboloid model is inherently different from that of a classifier in the Poincaré ball model, as the former is independent of the choice of reference point p. Using the decision hyperplane defined in (37), a hyperboloid perceptron [34] described in Algorithm 8 can be shown to have easily established performance guarantee.

Algorithm 8: Hyperboloid Perceptron

Input: Data points $\{x_i\}_{i=1}^N \in \mathbb{L}^n$, labels $\{y_i\}_{i=1}^N \in \{-1, +1\}$.

1 Initialization: $w_1 = 0, k = 1$.
2 **for** $t = 1, 2, \ldots$ **do**
3 | Predict $\hat{y}_t = \text{sgn}([w_k, x_t])$.
4 | **if** $\hat{y}_t \neq y_t$ **then**
5 | | $w_{k+1} = w_k + y_t H x_t, k = k + 1$.
6 | **end**
7 **end**

Theorem A.1: Let $(x_i, y_i)_{i=1}^N$ be a labeled data set from a bounded subset of $\mathbb{L}^n$ such that $\|x_i\| \leq R \ \forall i \in [N]$. Assume that there exists an optimal linear classifier with weight vector $w^\star$ such that $y_i \text{asinh}([w^\star, x_i]) \geq \varepsilon$ (ε-margin). Then, the hyperboloid perceptron in Algorithm 8 will correctly classify all points with at most $O\left(\frac{1}{\sinh^2(\varepsilon)} \right)$ updates.

Proof. According to the assumption, the optimal normal vector $w^\star$ satisfies $y_t \text{asinh}([w^\star, x_t]) \geq \varepsilon$ and $[w^\star, w^\star] = 1$. So we have

$$\begin{aligned} \langle w^\star, w_{k+1} \rangle &= \langle w^\star, w^k \rangle + y_t [w^\star, x_t] \\ &\geq \langle w^\star, w^k \rangle + \sinh(\varepsilon) \\ &\geq \ldots \geq k \sinh(\varepsilon), \end{aligned} \tag{38}$$

where the first inequality holds due to the ε-margin assumption and because $y_t \in \{-1, +1\}$. We can also upper bound $\|w_{k+1}\|$ as

$$
\begin{aligned}
\|w_{k+1}\|^2 &= \|w_k + y_t H x_t\|^2 \\
&= \|w_k\|^2 + \|x_t\|^2 + 2y_t [w_k, x_t] \\
&\leq \|w_k\|^2 + R^2 \\
&\leq \ldots \leq kR^2,
\end{aligned}
\tag{39}
$$

where the first inequality follows from $y_t [w_k, x_t] \leq 0$, corresponding to the case when the classifier makes a mistake. Combining (38) and (39), we have

$$
k \sinh(\varepsilon) \leq \langle w^\star, w_{k+1} \rangle \leq \|w^\star\| \|w_{k+1}\| \leq \|w^\star\| \sqrt{k} R
$$

$$
\Leftrightarrow k \leq \left(\frac{R\|w^\star\|}{\sinh(\varepsilon)} \right)^2,
\tag{40}
$$

which completes the proof. In practice we can always perform data processing to control the norm of $w^\star$. Also, for small classification margins ε, we have $\sinh(\varepsilon) \sim \varepsilon$. As a result, for data points that are very close to the decision boundary (ε is small), Theorem A.1 shows that the hyperbolid perceptron has roughly the same convergence rate as its Euclidean counterpart $\left(\frac{R}{\varepsilon} \right)^2$.

To experimentally confirm the convergence of Algorithm 8, we run synthetic data experiments similar to those described in Section IV. More precisely, we first randomly generate a $w^\star$ such that $[w^\star, w^\star] = 1$. Then, we generate a random set of $N = 5,000$ points $x_i{}_{i=1}^N$ in $\mathbb{L}^2$. For margin values $\varepsilon \in [0.1, 1]$, we remove points that violate the required constraint on the distance to the classifier (parameterized by $w^\star$). Then, we assign binary labels to each data point according to the optimal classifier so that $y_i = \text{sgn}([w^\star, x_i])$. We repeat this process for 100 different values of ε and compare the classification results with those of Algorithm 1 of [11]. Since the theoretical upper bound $O\left(\frac{1}{\sinh(\varepsilon)} \right)$ claimed in [11] is smaller than $O\left(\frac{1}{\sinh^2(\varepsilon)} \right)$ in Theorem A.1, we also plot the upper bound for comparison. From Figure 9 one can conclude that (1) Algorithm 8 always converges within the theoretical upper bound provided in Theorem A.1, and (2) both methods disagree with the theoretical convergence rate results of [11].
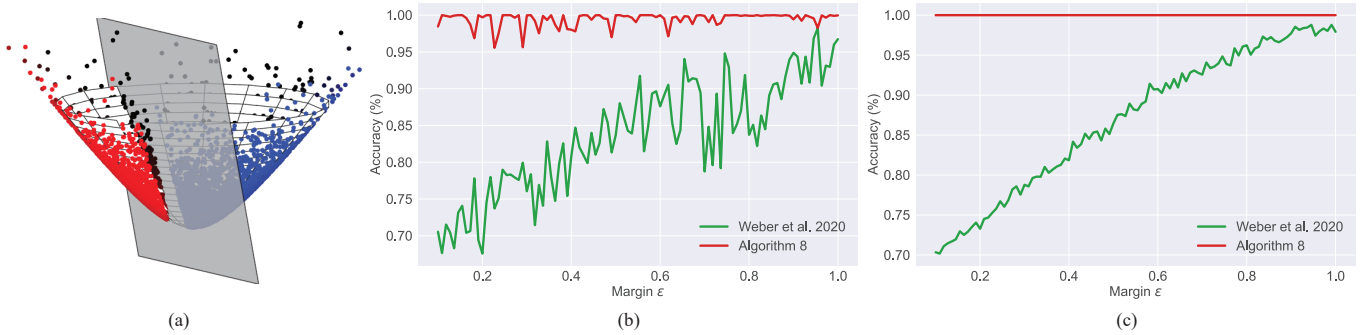

Fig. 9. (a) Visualization of a linear classifier in the hyperboloid model. Colors are indicative of the classes while the gray region represents the decision hyperplane; (b), (c) A comparison between the classification accuracy of the hyperboloid perceptron from Algorithm 8 and the perceptron of Algorithm 1 of [11], for different values of the margin ε. The classification accuracy is the average of five independent random trials. The stopping criterion is to either achieve a 100% classification accuracy or to reach the theoretical upper bound on the number of updates of the weight vector from Theorem 3.1 in [11] (Figure (b)), and Theorem A.1 (Figure (c)).

PROOF OF THEOREM 4.2 AND 4.3

Let $x_i \in \mathbb{B}^n$ and let $v_i = \log_p(x_i)$ be its its logarithmic map value. The distance between the point and the hyperplane defined by $w \in T_p\mathbb{B}^n$ and $p \in \mathbb{B}^n$ can be written as (see also (14))

$$
d(x, H_{w,p}) = \sinh^{-1} \left(\frac{2 \tanh(\sigma_p \|v_i\|/2) |\langle v_i, w \rangle|}{(1 - \tanh^2(\sigma_p \|v_i\|/2)) \|w\| \sigma_p \|v_i\|/2} \cdot \frac{\sigma_p}{2} \right).
\tag{41}
$$

For support vectors, $|\langle v_i, w \rangle| = 1$ and $\|v_i\| \geq 1/\|w\|$. Note that $f(x) = \frac{2\tanh(x)}{x(1-\tanh^2(x))}$ is an increasing function in x for $x > 0$ and $g(y) = \sinh^{-1}(y)$ is an increasing function in y for $y \in \mathbb{R}$. Thus, the distance in (41) can be lower bounded by

$$d(x_i, H_{w,p}) \geq \sinh^{-1}\left(\frac{2\tanh(\sigma_p/2\|w\|)}{1 - \tanh^2(\sigma_p/2\|w\|)} \right). \tag{42}$$

The goal is to maximize the distance in (42). To this end observe that $h(x) = \frac{2x}{1-x^2}$ is an increasing function in x for $x \in (0,1)$ and $\tanh(\sigma_p/2\|w\|) \in (0,1)$. So maximizing the distance is equivalent to minimizing $\|w\|$ (or $\|w\|^2$), provided σ_p is fixed. Thus the Poincaré SVM problem can be converted into the convex problem of Theorem 4.2; the constraints are added to force the hyperplane to correctly classify all points in the hard-margin setting. The formulation in Theorem 4.3 can also be seen as arising from a relaxation of the constraints and consideration of the trade-off between margin values and classification accuracy.

PROOF OF THEOREM 5.1

We generalize the arguments in [16] to hyperbolic spaces. Let $A_0 = aI$. The matrix A_k can be recursively computed from $A_k = A_{k-1} + z_t z_t^T$, or equivalently $A_k = aI + X_k X_k^T$. Without loss of generality, let t_k be the time index of the kth error.

$$\begin{aligned}
\xi_k^T A_k^{-1} \xi_k &= (\xi_{k-1} + y_{t_k} z_{t_k})^T A_k^{-1} (\xi_{k-1} + y_{t_k} z_{t_k}) \\
&= \xi_{k-1}^T A_k^{-1} \xi_{k-1} + z_{t_k}^T A_k^{-1} z_{t_k} + 2 y_{t_k} (A_k^{-1} \xi_{k-1})^T z_{t_k} \\
&= \xi_{k-1}^T A_k^{-1} \xi_{k-1} + z_{t_k}^T A_k^{-1} z_{t_k} + 2 y_{t_k} (w_{t_k})^T z_{t_k} \\
&\leq \xi_{k-1}^T A_k^{-1} \xi_{k-1} + z_{t_k}^T A_k^{-1} z_{t_k} \\
&\overset{(a)}{=} \xi_{k-1}^T A_{k-1}^{-1} \xi_{k-1} - \frac{(\xi_{k-1}^T A_{k-1}^{-1} z_{t_k})^2}{1 + z_{t_k}^T A_{k-1}^{-1} z_{t_k}} + z_{t_k}^T A_k^{-1} z_{t_k} \\
&\leq \xi_{k-1}^T A_{k-1}^{-1} \xi_{k-1} + z_{t_k}^T A_k^{-1} z_{t_k}
\end{aligned}$$

where (a) is due to the Sherman-Morrison formula [39] below.

Lemma A.1 ([39]): Let A be an arbitrary $n \times n$ positive-definite matrix. Let $x \in \mathbb{R}^n$. Then $B = A + xx^T$ is also a positive-definite matrix and

$$B^{-1} = A^{-1} - \frac{(A^{-1}x)(A^{-1}x)^T}{1 + x^T A^{-1} x}. \tag{43}$$

Note that the inequality holds since A_{k-1} is a positive-definite matrix and thus so is its inverse. Therefore, we have

$$\begin{aligned}
\xi_k^T A_k^{-1} \xi_k &\leq \xi_{k-1}^T A_{k-1}^{-1} \xi_{k-1} + z_{t_k}^T A_k^{-1} z_{t_k} \leq \sum_{j \in [k]} z_{t_j}^T A_j^{-1} z_{t_j} \\
&\overset{(b)}{=} \sum_{j \in [k]} \left(1 - \frac{\det(A_{j-1})}{\det(A_j)} \right) \overset{(c)}{\leq} \sum_{j \in [k]} \log\left(\frac{\det(A_j)}{\det(A_{j-1})} \right) \\
&= \log\left(\frac{\det(A_k)}{\det(A_0)} \right) = \log\left(\frac{\det(aI + X_k X_k^T)}{\det(aI)} \right) = \sum_{i \in [n]} \log\left(1 + \frac{\lambda_i}{a} \right),
\end{aligned}$$

where λ_i are the eigenvalues of $X_k X_k^T$. Claim (b) follows from Lemma A.2 while (c) is due to the fact $1 - x \leq -\log(x), \forall x > 0$.

Lemma A.2 ([16]): Let A be an arbitrary $n \times n$ positive-semidefinite matrix. Let $x \in \mathbb{R}^n$ and $B = A - xx^T$. Then

$$x^T A^\dagger x = \begin{cases} 1 & \text{if } x \notin span(B) \\ 1 - \frac{\det_{\neq 0}(B)}{\det_{\neq 0}(A)} < 1 & \text{if } x \in span(B) \end{cases}, \tag{44}$$

where $\det_{\neq 0}(B)$ is the product of non-zero eigenvalues of B.

This leads to the upper bound for $\xi_k^T A_k^{-1} \xi_k$. For the lower bound, we have

$$\sqrt{\xi_k^T A_k^{-1} \xi_k} \geq \left\langle A_k^{-1/2} \xi_k, \frac{A_k^{1/2} w^\star}{\|A_k^{1/2} w^\star\|} \right\rangle = \frac{\langle \xi_k, w^\star \rangle}{\|A_k^{1/2} w^\star\|} \geq \frac{k\varepsilon'}{\|A_k^{1/2} w^\star\|}.$$

Also, recall $\xi_k = \sum_{j \in [k]} y_{\sigma(j)} z_{\sigma(j)}$. Combining the bounds we get

$$\left(\frac{k\varepsilon'}{\|A_k^{1/2} w^\star\|} \right)^2 \leq \xi_k^T A_k^{-1} \xi_k \leq \sum_{i \in [n]} \log\left(1 + \frac{\lambda_i}{a} \right).$$

This leads to the bound $k \leq \frac{\|A_k^{1/2} w^\star\|}{\varepsilon'} \sqrt{\sum_{i \in [n]} \log(1 + \frac{\lambda_i}{a})}$. Finally, since $\|w^\star\| = 1$, we have

$$\|A_k^{1/2} w^\star\|^2 = (w^\star)^T (aI + X_k X_k^T) w^\star = a + \lambda_{w^\star},$$

which follows from the definition of $\lambda_{w^\star}$. Hence,

$$k \leq \frac{1}{\varepsilon'} \sqrt{(a + \lambda_{w^\star}) \sum_{i \in [n]} \log(1 + \frac{\lambda_i}{a})}, \tag{45}$$

which completes the proof.

PROOF OF THEOREM 5.2

To prove Theorem 5.2, we need the following lemmas A.3 and A.4.

Lemma A.3: For any $\tilde{v}_t$ in the update rule, we have $\eta_t y_t \langle \tilde{v}_t, w^\star \rangle \geq \sinh(\varepsilon)$, where $w^\star$ stands for the optimal classifier in Assumption 4.1. Also, for any w_k, we have $\langle w_k, w^\star \rangle \geq 0$.

Proof. The proof is by induction. Initially, $w_1 = \mathbf{0}$ and all arriving points get classified as positive. The first mistake occurs when the first negative point z_t arrives, which gets classified as positive. In this case, $w_2 = -\eta_t v_t$, where $v_t = \log_p(z_t)$ and $\eta_t = \frac{2 \tanh\left(\frac{\sigma_p \|v_t\|}{2}\right)}{\left(1 - \tanh\left(\frac{\sigma_p \|v_t\|}{2}\right)^2\right) \|v_t\|}$. Also, v_t must be unmanipulated (i.e., $v_t = u_t$) since it will always be classified as positive. Therefore, based on Assumption 4.1, we have

$$\eta_t \langle v_t, w^\star \rangle = \eta_t \langle u_t, w^\star \rangle \leq -\sinh(\varepsilon), \quad \langle w_2, w^\star \rangle = -\eta_t \langle v_t, w^\star \rangle \geq 0. \tag{46}$$

Next, suppose that w_{t-1} denotes the weight vector at the end of step $t - 1$ and $\langle w_{t-1}, w^\star \rangle \geq 0$. We need to show that $\eta_t y_t \langle \tilde{v}_t, w^\star \rangle \geq \sinh(\varepsilon)$. By definition, for any point such that $\frac{\langle v_t, w_{t-1} \rangle}{w_{t-1}} \neq \frac{\alpha}{\sigma_p}$, $\tilde{v}_t = v_t$. According to Observation 1, those points are also not manipulated, i.e. $\tilde{v}_t = v_t = u_t$. Therefore the claim holds. For data points such that $\frac{\langle v_t, w_{t-1} \rangle}{w_{t-1}} = \frac{\alpha}{\sigma_p}$, if they are positive, we have $\tilde{v}_t = v_t = u_t + \beta \frac{w_{t-1}}{\|w_{t-1}\|}$, where $0 \leq \beta \leq \frac{\alpha}{\sigma_p}$. The reason behind β always being positive is that all rational agents want to be classified as positive so the only possible direction of change is w_{t-1}. Hence,

$$\eta_t \langle \tilde{v}_t, w^\star \rangle = \eta_t \left\langle u_t + \beta \frac{w_{t-1}}{\|w_{t-1}\|}, w^\star \right\rangle \geq \eta_t \langle u_t, w^\star \rangle \geq \sinh(\varepsilon). \tag{47}$$

For data points with negative labels such that $\frac{\langle v_t, w_{t-1} \rangle}{w_{t-1}} = \frac{\alpha}{\sigma_p}$, $\tilde{v}_t = v_t - \frac{\alpha w_{t-1}}{\sigma_p \|w_{t-1}\|}$ and $v_t = u_t + \beta \frac{w_{t-1}}{\|w_{t-1}\|}$. This implies that $\tilde{v}_t = u_t + \left(\beta - \frac{\alpha}{\sigma_p}\right) \frac{w_{t-1}}{\|w_{t-1}\|}$. Therefore,

$$\eta_t \langle \tilde{v}_t, w^\star \rangle = \eta_t \left\langle u_t + \left(\beta - \frac{\alpha}{\sigma_p}\right) \frac{w_{t-1}}{\|w_{t-1}\|}, w^\star \right\rangle \leq \eta_t \langle u_t, w^\star \rangle \leq -\sinh(\varepsilon). \tag{48}$$

Combining the above two claims we get $\eta_t y_t \langle \tilde{v}_t, w^\star \rangle \geq \sinh(\varepsilon)$.

The last step is to assume $\langle w_{t-1}, w^\star \rangle \geq 0$ and $\eta_t y_t \langle \tilde{v}_t, w^\star \rangle \geq \sinh(\varepsilon)$. I this case, we need to show $\langle w_t, w^\star \rangle \geq 0$. If the classifier does not make a mistake at step t, the claim is obviously true since $w_{t-1} = w_t$. If he classifier makes a mistake, we have

$$\langle w_t, w^\star \rangle = \langle w_{t-1} + \eta_t y_t \tilde{v}_t, w^\star \rangle \geq \langle w_{t-1}, w^\star \rangle \geq 0. \tag{49}$$

This completes the proof.

Lemma A.4: If Algorithm 5 makes a mistake on an observed data point v_t then $y_t \langle \tilde{v}_t, w_{t-1} \rangle \leq 0$.

Proof. If the algorithm makes a mistake on a positive example, we have $\frac{\langle v_t, w_{t-1} \rangle}{\|w_{t-1}\|} < \frac{\alpha}{\sigma_p}$. By Observation 2, no point will fall within the region $0 < \frac{\langle v_t, w_{t-1} \rangle}{\|w_{t-1}\|} < \frac{\alpha}{\sigma_p}$. Thus one must have $\frac{\langle v_t, w_{t-1} \rangle}{\|w_{t-1}\|} \leq 0$. Since $y_t = +1$, $\tilde{v}_t = v_t$. Therefore, $\langle \tilde{v}_t, w_{t-1} \rangle \leq 0$. If the algorithm makes a mistake on a negative point, we have $\frac{\langle v_t, w_{t-1} \rangle}{\|w_{t-1}\|} \geq \frac{\alpha}{\sigma_p}$. For the case $\frac{\langle v_t, w_{t-1} \rangle}{\|w_{t-1}\|} > \frac{\alpha}{\sigma_p}$, we have $\tilde{v}_t = v_t$. In this case, $\langle \tilde{v}_t, w_{t-1} \rangle \geq 0$ obviously holds. For $\frac{\langle v_t, w_{t-1} \rangle}{\|w_{t-1}\|} = \frac{\alpha}{\sigma_p}$, we have

$$\langle \tilde{v}_t, w_{t-1} \rangle = \left\langle v_t - \frac{\alpha w_{t-1}}{\sigma_p \|w_{t-1}\|}, w_{t-1} \right\rangle = 0. \tag{50}$$

The above equality implies that for a negative sample we have $\langle \tilde{v}_t, w_{t-1} \rangle \geq 0$. Therefore, for any mistaken data point, $y_t \langle \tilde{v}_t, w_{t-1} \rangle \leq 0$.

We are now ready to prove Theorem 5.2.

Proof. The analysis follows along the same lines as that for the standard Poincaré perceptron algorithm described in Section IV. We first lower bound $\|w_{k+1}\|$ as

$$\begin{aligned}
\|w_{k+1}\| &\geq \langle w_{k+1}, w^\star \rangle \\
&= \langle w_k, w^\star \rangle + \eta_{i_k} y_{i_k} \langle \tilde{v}_{i_k}, w^\star \rangle \\
&\geq \langle w_k, w^\star \rangle + \sinh(\varepsilon) \geq \cdots \geq k \sinh(\varepsilon),
\end{aligned} \tag{51}$$

where the first bound follows from the Cauchy-Schwartz inequality, while the second inequality was established in Lemma A.3. Next, we upper bound $\|w_{k+1}\|$ as

$$\begin{aligned}
\|w_{k+1}\|^2 &= \|w_k + \eta_{i_k} y_{i_k} \tilde{v}_{i_k}\|^2 \\
&= \|w_k\|^2 + 2\eta_{i_k} y_{i_k} \langle w_k, \tilde{v}_{i_k} \rangle + \|\tilde{v}_{i_k}\|^2 \\
&\leq \|w_k\|^2 + \|\tilde{v}_{i_k}\|^2 \\
&\leq \|w_k\|^2 + \left(\frac{2\tanh(\frac{\sigma_p \|v_{i_k}\|}{2})}{1 - \tanh(\frac{\sigma_p \|v_{i_k}\|}{2})^2} + \frac{\alpha}{\sigma_p} \right)^2 \\
&\leq \|w_k\|^2 + \left(\frac{2R_p}{1 - R_p^2} + \frac{\alpha}{\sigma_p} \right)^2 \leq \cdots \leq k \left(\frac{2R_p \sigma_p + \alpha(1 - R_p^2)}{\sigma_p(1 - R_p^2)} \right)^2,
\end{aligned} \tag{52}$$

where the first inequality was established Lemma A.4 while the second inequality follows from the fact that the manipulation budget is α.

Combining (51) and (52) we obtain

$$k^2 \sinh(\varepsilon)^2 \leq k \left(\frac{2R_p \sigma_p + \alpha(1 - R_p^2)}{\sigma_p(1 - R_p^2)} \right)^2$$

$$k \leq \left(\frac{2R_p \sigma_p + \alpha(1 - R_p^2)}{\sigma_p(1 - R_p^2) \sinh(\varepsilon)} \right)^2, \tag{53}$$

which completes the proof.

DETAILED EXPERIMENTAL SETTING

For the first set of experiments, we have the following hyperparameters. For the Poincaré perceptron, there are no hyperparameters to choose. For the Poincaré second-order perceptron, we adopt the strategy proposed in [16]. That is, instead of tuning the parameter a, we set it to 0 and change the matrix inverse to pseudo-inverse. For the Poincaré SVM and the Euclidean SVM, we set $C = 1000$ for all data sets. This theoretically forces SVM to have a hard decision boundary. For the hyperboloid SVM, we surprisingly find that choosing $C = 1000$ makes the algorithm unstable. Empirically, $C = 10$ in general produces better results despite the fact that it still leads to softer decision boundaries and still breaks down when the point dimensions are large. As the hyperboloid SVM works in the hyperboloid model of a hyperbolic space, we map points from the Poincaré ball to points in the hyperboloid model as follows. Let $x \in \mathbb{B}^n$ and $z \in \mathbb{L}^n$ be its corresponding point in the hyperboloid model. Then,

$$z_0 = \frac{1 - \sum_{i=0}^{n-1} x_i^2}{1 + \sum_{i=1}^n x_i^2}, \ z_j = \frac{2x_j}{1 + \sum_{i=1}^n x_i^2} \ \forall j \in [n]. \tag{54}$$

On the other hand, Olsson's scRNA-seq data contains 319 points from 8 classes and we perform a $70\%/30\%$ random split to obtain training (231) and test (88) point sets. CIFAR10 contains $50,000$ training points and $10,000$ testing points from 10 classes. Fashion-MNIST contains $60,000$ training points and $10,000$ testing points from 10 classes. Mini-ImageNet contains $8,000$ data points from 20 classes and we do $70\%/30\%$ random split to obtain training $(5,600)$ and test $(2,400)$ point sets. For all data sets we choose the trade-off coefficient $C = 5$, and use it with all three SVM algorithms to ensure a fair comparison. We also find that in practice the performance of all three algorithms remains stable when $C \in [1, 10]$.