Robust large-margin learning in hyperbolic space

Melanie Weber[*]
Princeton University
mw25@math.princeton.edu

Manzil Zaheer
Google Research
manzilzaheer@google.com

Ankit Singh Rawat
Google Research
ankitsrawat@google.com

Aditya Menon
Google Research
adityakmenon@google.com

Sanjiv Kumar
Google Research
sanjivk@google.com

Abstract

Recently, there has been a surge of interest in representation learning in hyperbolic spaces, driven by their ability to represent hierarchical data with significantly fewer dimensions than standard Euclidean spaces. However, the viability and benefits of hyperbolic spaces for downstream machine learning tasks have received less attention. In this paper, we present, to our knowledge, the first theoretical guarantees for learning a classifier in hyperbolic rather than Euclidean space. Specifically, we consider the problem of learning a *large-margin* classifier for data possessing a hierarchical structure. Our first contribution is a *hyperbolic perceptron* algorithm, which provably converges to a separating hyperplane. We then provide an algorithm to efficiently learn a large-margin hyperplane, relying on the careful injection of *adversarial examples*. Finally, we prove that for hierarchical data that embeds well into hyperbolic space, the low embedding dimension ensures superior guarantees when learning the classifier directly in hyperbolic space.

1 Introduction

Hyperbolic spaces have received sustained interest in recent years, owing to their ability to compactly represent data possessing hierarchical structure (e.g., trees and graphs). In terms of *representation learning*, hyperbolic spaces offer a provable advantage over Euclidean spaces for such data: objects requiring an *exponential* number of dimensions in Euclidean space can be represented in a *polynomial* number of dimensions in hyperbolic space [28]. This has motivated research into efficiently learning a suitable hyperbolic embedding for large-scale datasets [22, 4, 31].

Despite this impressive representation power, little is known about the benefits of hyperbolic spaces for downstream tasks. For example, suppose we wish to perform classification on data that is intrinsically hierarchical. One may naïvely ignore this structure, and use a standard Euclidean embedding and corresponding classifier (e.g., SVM). However, can we design classification algorithms that exploit the structure of hyperbolic space, and offer *provable* benefits in terms of *performance*? This fundamental question has received surprisingly limited attention. While some prior work has proposed specific algorithms for learning classifiers in hyperbolic space [6, 21], these have been primarily empirical in nature, and do not come equipped with theoretical guarantees on convergence and generalization.

In this paper, we take a first step towards addressing this question for the problem of learning a *large-margin* classifier. We provide a series of algorithms to *provably* learn such classifiers in hyperbolic space, and establish their superiority over the classifiers naïvely learned in Euclidean space. This shows that by using a hyperbolic space that better reflects the intrinsic geometry of the data, one can see gains in both representation size and performance. Specifically, our contributions are:

[*]Work performed while intern at Google Research, New York.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

(i) we provide a hyperbolic version of the classic perceptron algorithm and establish its convergence for data that is separable with a margin (Theorem 3.1).
(ii) we establish how suitable injection of *adversarial examples* to *gradient-based* loss minimization yields an algorithm which can *efficiently* learn a *large-margin* classifier (Theorem 4.3). We further establish that simply performing gradient descent or using adversarial examples alone does not suffice to efficiently yield such a classifier.
(iii) we compare the Euclidean and hyperbolic approaches for hierarchical data and analyze the trade-off between low embedding dimensions and low distortion (*dimension-distortion trade-off*) when learning robust classifiers on embedded data. For hierarchical data that embeds well into hyperbolic space, it suffices to use smaller embedding dimension while ensuring superior guarantees when we learn a classifier in hyperbolic space.

Contribution (i) establishes that it is possible to design classification algorithms that exploit the structure of hyperbolic space, while provably converging to *some* admissible (not necessarily large-margin) separator. Contribution (ii) establishes that it is further possible to design classification algorithms that provably converge to a *large-margin* separator, by suitably injecting adversarial examples. Contribution (iii) shows that the adaptation of algorithms to the intrinsic geometry of the data can enable efficient utilization of the embedding space without affecting the performance.

Related work. Our results can be seen as hyperbolic analogue of classic results for Euclidean spaces. The large-margin learning problem is well studied in the Euclidean setting. Classic algorithms for learning classifiers include the perceptron [25, 24, 12] and support vector machines [8]. Robust margin-learning has been widely studied; notably by Lanckriet et al. [16], El Ghaoui et al. [10], Kim et al. [15] and, recently, via adversarial approaches by Charles et al. [5], Ji and Telgarsky [14], Li et al. [18] and Soudry et al. [30]. Adversarial learning has recently gained interest through efforts to train more robust deep learning systems (see, e.g., [20, 11]).

Recently, the representation of (hierarchical) data in hyperbolic space has gained a surge of interest. The literature focuses mostly on learning representations in the Poincare [22, 4, 31] and Lorentz [23] models of hyperbolic space, as well as on analyzing representation trade-offs in hyperbolic embeddings [27, 32] . The body of work on performing downstream ML tasks in hyperbolic space is much smaller and mostly without theoretical guarantees. Monath et al. [21] study hierarchical clustering in hyperbolic space. Cho et al. [6] introduce a hyperbolic version of support vector machines for binary classification in hyperbolic space, albeit without theoretical guarantees. Ganea et al. [13] introduce a hyperbolic version of neural networks that shows empirical promise on downstream tasks, but likewise without theoretical guarantees. To the best of our knowledge, neither robust large-margin learning nor adversarial learning in hyperbolic spaces has been studied in the literature, including [6]. Furthermore, we are not aware of any theoretical analysis of dimension-distortion trade-offs in the related literature.

2 Background and notation

We begin by reviewing some background material on hyperbolic spaces, as well as embedding into and learning in these spaces.

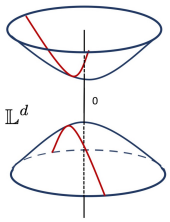

Figure 1: Lorentz model (geodesics in red).

2.1 Hyperbolic space

Hyperbolic spaces are smooth Riemannian manifolds $M = \mathsf{H}^d$ with constant negative curvature κ and are as such locally Euclidean spaces. There are several equivalent models of hyperbolic space, each highlighting a different geometric aspect. In this work, we mostly consider the *Lorentz model* (aka *hyperboloid model*), which we briefly introduce below, with more details provided Appendix A (see [3] for a comprehensive overview).

For $\boldsymbol{x}, \boldsymbol{x}' \in \mathsf{R}^{d+1}$, let $\boldsymbol{x} \quad \boldsymbol{x}' = x_0 x_0' \quad \sum_{i=1}^{d} x_i x_i'$ denote their *Minkowski product*. The Lorentz model is defined as $\mathsf{L}^d = \{\boldsymbol{x} \in \mathsf{R}^{d+1} : \boldsymbol{x} \quad \boldsymbol{x} = 1\}$ with distance measure $d_{\mathsf{L}}(\boldsymbol{x}, \boldsymbol{x}') = \mathsf{acosh}(\boldsymbol{x} \quad \boldsymbol{x}')$. Note that the distance $d_{\mathsf{L}}(\boldsymbol{x}, \boldsymbol{x}')$ corresponds to the length of the shortest line (*geodesic*) along the manifold connecting $\boldsymbol{x}$ and $\boldsymbol{x}'$ (cf. Fig. 1). We also point out that $(\mathsf{L}, d_{\mathsf{L}})$ forms a metric space.

2.2 Embeddability of hierarchical data

A map $\phi : X_1 \to X_2$ between metric spaces (X_1, d_1) and (X_2, d_2) is called an *embedding*. The *multiplicative* distortion of ϕ is defined to be the smallest constant $c_M \quad 1$ such that, $\forall \boldsymbol{x}, \boldsymbol{x}' \in X_1$,

$d_2\big(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\big)$ $d_1(\boldsymbol{x}, \boldsymbol{x}')$ c_M $d_2\big(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\big)$. When $c_M = 1$, ϕ is termed an *isometric embedding*. Since hierarchical data is tree-like, we can use classic embeddability results for trees as a reference point. Bourgain [2], Linial et al. [19] showed that an N-point metric X (i.e., $|X| = N$) embeds into Euclidean space $\mathbb{R}^{O(\log^2 N)}$ with $c_M = O(\log N)$. This bound is tight for trees in the sense that embedding them in a Euclidean space (of any dimension) must have $c_m = (\log N)$ [19]. In contrast, Sarkar [28] showed that trees embed quasi-isometrically with $c_M = O(1 + \epsilon)$ into hyperbolic space $\mathbb{H}^d$, even in the low-dimensional regime with the dimension as small as $d = 2$.

2.3 Classification in hyperbolic space

We consider classification problems of the following form: X $\mathbb{L}^d$ denotes the feature space, $Y = \{1\}$ the binary label space, and W $\mathbb{R}^{d+1}$ the model space. In the following, we denote the training set as S X Y.

We begin by defining *geodesic decision boundaries*. Consider the Lorentz space $\mathbb{L}^d$ with ambient space $\mathbb{R}^{d+1}$. Then every geodesic decision boundary is a hyperplane in $\mathbb{R}^d$ intersecting $\mathbb{L}^d$ and $\mathbb{R}^{d+1}$. Further, consider the set of linear separators or *decision functions* of the form

$$H = \{h_{\boldsymbol{w}} : \boldsymbol{w} \in \mathbb{R}^{d+1}, \boldsymbol{w} \quad \boldsymbol{w} < 0\}, \text{ where } h_{\boldsymbol{w}}(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{w} \quad \boldsymbol{x} > 0 \\ 1, & \text{otherwise.} \end{cases} \tag{2.1}$$

Note that the requirement $\boldsymbol{w} \quad \boldsymbol{w} < 0$ in (2.1) ensures that the intersection of $\mathbb{L}^d$ and the decision hyperplane $h_{\boldsymbol{w}}$ is not empty. The geodesic decision boundary corresponding to the decision function $h_{\boldsymbol{w}}$ is then given by $\partial H_{\boldsymbol{w}} = \{\boldsymbol{z} \in \mathbb{L}^d : \boldsymbol{w} \quad \boldsymbol{z} = 0\}$. The distance of a point $\boldsymbol{x} \in \mathbb{L}^d$ from the decision boundary $\partial H_{\boldsymbol{w}}$ can be computed as $d\big(\boldsymbol{x}, \partial H_{\boldsymbol{w}}\big) = \left|\operatorname{asinh}\left(\boldsymbol{w} \quad \boldsymbol{x} / \sqrt{\boldsymbol{w} \quad \boldsymbol{w}}\right)\right|$ [6].

2.4 Large-margin classification in hyperbolic space

In this paper, we are interested in learning a *large margin* classifier in a hyperbolic space. Analogous to the Euclidean setting, the natural notion of margin is the minimal distance to the decision boundary over all training samples:

$$\operatorname{margin}_S(w) = \inf_{(\boldsymbol{x},y)\in S} y h_{\boldsymbol{w}}(\boldsymbol{x}) \quad d(\boldsymbol{x}, \partial H_w) = \inf_{(\boldsymbol{x},y)\in S} \operatorname{asinh}\left(y(\boldsymbol{w} \quad \boldsymbol{x}) / \sqrt{\boldsymbol{w} \quad \boldsymbol{w}}\right). \tag{2.2}$$

For *large-margin classifier learning*, we aim to find $h_{\boldsymbol{w}}$ defined by $\boldsymbol{w}^* = \operatorname{argmax}_{\boldsymbol{w}\in C} \operatorname{margin}_S(\boldsymbol{w})$, where $C = \{\boldsymbol{w} \in \mathbb{R}^{d+1} : \boldsymbol{w} \quad \boldsymbol{w} < 0\}$ imposes a *nonconvex* constraint. This makes the problem computationally intractable using classical methods, unlike its Euclidean counterpart.

3 Hyperbolic linear separator learning

The first step towards the goal of learning a large-margin classifier is to establish that we can *provably* learn *some* separator. To this end, we present a hyperbolic version of the classic perceptron algorithm and establish that it will converge on data that is separable with a margin.

3.1 The hyperbolic perceptron algorithm

The hyperbolic perceptron (cf. Alg. 1) learns a binary classifier $\boldsymbol{w}$ with respect to the Minkowski product. This is implemented in the update rule $\boldsymbol{v}_t$ $\boldsymbol{w}_t + y\boldsymbol{x}$, similar to the Euclidean case. In contrast to the Euclidean case, the hyperbolic perceptron requires an additional normalization step $\boldsymbol{w}_{t+1}$ $\boldsymbol{v}_t / \sqrt{\boldsymbol{v}_t \quad \boldsymbol{v}_t}$. This ensures that $\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_{t+1} < 0$; as a result $\boldsymbol{w}_{t+1}$ defines a meaningful classifier, i.e., $\mathbb{L}^d \setminus \partial H_{\boldsymbol{w}_{t+1}}$ $\mathfrak{G}$;. For more details, see Appendix B.

While intuitive, it remains to establish that this algorithm converges, i.e., finds a solution which correctly classifies all the training samples. To this end, consider the following notion of *hyperbolic linear separability with a margin*: for X, X' $\mathbb{L}^d$, we say that X and X' are linearly separable with (hyperbolic) margin γ_H, if there exists a $\boldsymbol{w} \in \mathbb{R}^{d+1}$ with $\sqrt{\boldsymbol{w} \quad \boldsymbol{w}} = 1$ such that $\boldsymbol{w} \quad \boldsymbol{x} > \sinh(\gamma_H)$ $\forall\, \boldsymbol{x} \in X$ and $\boldsymbol{w} \quad \boldsymbol{x}' < \sinh(\gamma_H)$ $\forall\, \boldsymbol{x}' \in X'$. Assuming our training set is separable with a margin, the hyperbolic perceptron has the following convergence guarantee:

Theorem 3.1. *Assume that there is some* $\boldsymbol{w} \in \mathbb{R}^{d+1}$ *with* $\sqrt{\boldsymbol{w} \quad \boldsymbol{w}} = 1$ *and* $\boldsymbol{w}_0 \quad \boldsymbol{w}$ 0*, and some* $\gamma_H > 0$*, such that* $y_j(\boldsymbol{w} \quad \boldsymbol{x}_j)$ $\sinh(\gamma_H)$ *for* $j = 1, \ldots, |S|$*. Then, Alg. 1 converges in* $O\left(1/\sinh(\gamma_H)\right)$ *steps and returns a solution with margin* γ_H*.*

Algorithm 1 Hyperbolic perceptron

Initialize $\boldsymbol{w}_0 \in \mathbb{R}^{d+1}$.
for $t = 0, 1, \ldots, T-1$ **do**
  for $j = 1, \ldots, n$ **do**
    if $\operatorname{sgn}(\boldsymbol{x}_j \mathbin{\bullet} \boldsymbol{w}_t) \neq y_j$ **then**
      $\boldsymbol{v}_t \leftarrow \boldsymbol{w}_t + y_j \boldsymbol{x}_j$
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{v}_t / \min\{1, \sqrt{\boldsymbol{v}_t \mathbin{\bullet} \boldsymbol{v}_t}\}$
      break
    end if
  end for
end for
Output: $\boldsymbol{w}_T$

Algorithm 2 Adversarial Training

Initialize $\boldsymbol{w}_0 = 0, S' = \emptyset$.
for $t = 0, 1, \ldots, T-1$ **do**
  $S_t \leftarrow S$ iid with $|S_t| = m$; $S'_t \leftarrow \emptyset$.
  for $i = 0, 1, \ldots, m$ **do**
    $\boldsymbol{x}_i \leftarrow \operatorname{argmax}_{d_{\mathbb{L}}(\boldsymbol{x}_i, \boldsymbol{z}) \leq \alpha} l(\boldsymbol{z}, y_i; \boldsymbol{w}_t)$
  end for
  $S'_t \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$
  $S' \leftarrow S' \cup S'_t$
  $\boldsymbol{w}_{t+1} \leftarrow A(\boldsymbol{w}_t, S, S')$
end for
Output: $\boldsymbol{w}_T$

The proof of Thm. 3.1 (provided in Appendix B) follows the standard proof of the Euclidean perceptron and utilizes the Cauchy-Schwartz inequality for the Minkowski product. To verify that Algorithm 1 always converges to a valid hyperplane, i.e., such that $\mathbb{L}^d \setminus H_{\boldsymbol{v}_t} \neq \emptyset$, which happens iff $\boldsymbol{v}_t \mathbin{\bullet} \boldsymbol{v}_t < 0$, consider the following argument:

$$\boldsymbol{v}_t \mathbin{\bullet} \boldsymbol{v}_t = (\boldsymbol{w}_t + y_j \boldsymbol{x}_j) \mathbin{\bullet} (\boldsymbol{w}_t + y_j \boldsymbol{x}_j) = \underbrace{\boldsymbol{w}_t \mathbin{\bullet} \boldsymbol{w}_t}_{\overset{(i)}{\leq -1}} + 2 \underbrace{y_j (\boldsymbol{x}_j \mathbin{\bullet} \boldsymbol{w}_t)}_{\overset{(ii)}{< 0}} + \underbrace{y^2}_{=1} \underbrace{(\boldsymbol{x}_j \mathbin{\bullet} \boldsymbol{x}_j)}_{\overset{(iii)}{=} 1} < 0 \,.$$

Here, (i) is a consequence of the normalization step in Algorithm 1 and (iii) follows as $\boldsymbol{x}_j \mathbin{\bullet} \boldsymbol{x}_j = 1$, since $\boldsymbol{x}_j \in \mathbb{L}^d$. As for (ii), note that we perform the update $\boldsymbol{v}_t \leftarrow \boldsymbol{w}_t + y_j \boldsymbol{x}_j$ only when $y_j \neq \operatorname{sign}(\boldsymbol{x}_j \mathbin{\bullet} \boldsymbol{w})$ (cf. Algorithm 1).

Remark 3.2. Note that the perceptron algorithm in Euclidean space exhibits a $O(1/\gamma_E^2)$ convergence rate [24], where γ_E denotes the Euclidean margin. When $\gamma_E \to 0$, the $1/\sinh(\gamma_H)$ convergence rate for hyperbolic spaces can be significantly faster than $1/\gamma_E^2$, indicating that exploiting the structure of hyperbolic space can be beneficial.

3.2 The challenge of large-margin learning

Thm. 3.1 establishes that the hyperbolic perceptron converges to *some* linear separator. However, for the purposes of generalization, one would ideally like to converge to a *large-margin* separator. As with the classic Euclidean perceptron, no such guarantee is possible for the hyperbolic perceptron; this motivates us to ask whether a suitable modification can rectify this.

Drawing inspiration from the Euclidean setting, a natural way to proceed is to consider the use of *margin losses*, such as the logistic or hinge loss. Formally, let $l : \mathcal{X} \times \{\pm 1\} \to \mathbb{R}_+$ be a loss function:

$$l(\boldsymbol{x}, y; \boldsymbol{w}) = f(y \, (\boldsymbol{w} \mathbin{\bullet} \boldsymbol{x})), \tag{3.1}$$

where $f : \mathbb{R} \to \mathbb{R}_+$ is some convex, non-increasing function, e.g., the hinge loss. The empirical risk of the classifier parameterized by $\boldsymbol{w}$ on the training set $S \subseteq \mathcal{X} \times \{\pm 1\}$ is $L(\boldsymbol{w}; S) = \sum_{(x,y) \in \mathcal{S}} l(\boldsymbol{x}, y; \boldsymbol{w}) / |S|$. Commonly, we learn a classifier by minimizing $L(\boldsymbol{w}; S)$ via gradient descent with iterates

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \sum\nolimits_{(x,y) \in \mathcal{S}} \nabla l(\boldsymbol{x}, y; \boldsymbol{w}_t) / |S| \,, \tag{3.2}$$

where $\eta > 0$ denotes the learning rate. Unfortunately, while this will yield a large-margin solution, the following result demonstrates that the number of iterations required may be prohibitively large.

Theorem 3.3. *Let $\boldsymbol{e}_i \in \mathbb{R}^{d+1}$ be the i-th standard basis vector. Consider the training set $S = \{(\boldsymbol{e}_1, 1), (-\boldsymbol{e}_1, -1)\}$ and the initialization $\boldsymbol{w}_0 = \boldsymbol{e}_2$. Suppose $\{\boldsymbol{w}_t\}_{t \geq 0}$ is a sequence of iterates in* (3.2)*. Then, the number of iterations needed to achieve margin γ_H is $\Omega(\exp(\gamma_H))$.*

While this result is disheartening, fortunately, we now present a simple resolution: by suitably adding *adversarial examples*, the gradient descent converges to a large-margin solution in *polynomial time*.

4 Hyperbolic large-margin separator learning via adversarial examples

Thm. 3.3 reveals that gradient descent on a margin loss is insufficient to efficiently obtain a large-margin classifier. Adapting the approach proposed in [5] for the Euclidean setting, we show how to

Space	Perceptron	Adversarial margin	Adversarial ERM	Adversarial GD
Euclidean (prior work)	$O\left(\frac{1}{\gamma_E^2}\right)$	$\gamma_E \quad \alpha$	$(\exp(d))$	$\left(\text{poly}\left(\frac{1}{\gamma_E-\alpha}\right)\right)$
Hyperbolic (this paper)	$O\left(\frac{1}{\sinh(\gamma_H)}\right)$	$\frac{\gamma_H}{\cosh(\alpha)}$	$(\exp(d))$	$\left(\text{poly}\left(\frac{\cosh(\alpha)}{\sinh(\gamma_H)}\right)\right)$

Table 1: Comparison between Euclidean and hyperbolic spaces for Perceptron (cf. Alg. 1) and adversarial training (cf. Alg. 2). Recall that $\gamma_{E/H}$, α, and d denote the (Euclidean/ hyperbolic) margin of the training data, the adversarial perturbation budget, and the underlying dimension, respectively.

alleviate this problem by enriching the training set with *adversarial examples* before updating the classifier (cf. Alg. 2). In particular, we minimize a robust loss

$$\min_{w \in \mathbb{R}^{d+1}} L_{\text{rob}}(\boldsymbol{w}; S) := \frac{1}{|S|} \sum_{(x,y)\in S} l_{\text{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) \tag{4.1}$$

$$l_{\text{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) := \max_{\substack{z \in \mathbb{L}^d: \\ d_{\mathbb{L}}(x,z)\le\alpha}} l(\boldsymbol{z}, y; \boldsymbol{w}) \ . \tag{4.2}$$

The problem has a minimax structure, where the outer optimization minimizes the training error over S. The inner optimization generates an adversarial example by perturbing a given input feature $\boldsymbol{x}$ on the hyperbolic manifold. Note that the magnitude of the perturbation added to the original example is bounded by α, which we refer to as the *adversarial budget*. In particular, we want to construct a perturbation that maximizes the loss l, i.e., $\boldsymbol{x} \quad \text{argmax}_{d_{\mathbb{L}}(x,z)\le\alpha} l(\boldsymbol{z}, y; \boldsymbol{w})$. In this paper, we restrict ourselves to only those adversarial examples that lead to misclassification with respect to the current classifier, i.e., $h_w(\boldsymbol{x}) \neq h_w(\boldsymbol{x})$ (cf. Remark 4.2).

Adversarial example $\boldsymbol{x}$ can be generated efficiently by reducing the problem to an (Euclidean) linear program with a spherical constraint:

$$\text{(CERT)} \quad \max_{z\in\mathbb{R}^d} \quad w_0 z_0 + \sum_{i=1}^{d} w_i z_i \quad \text{s.t.} \quad \sum_{i=1}^{d} x_i z_i \quad \cosh(\alpha) \quad x_0 z_0, \ \|\boldsymbol{z}_{\setminus 0}\|^2 = z_0^2 \quad 1 \ . \tag{4.3}$$

Importantly, as detailed in Appendix C.2 and summarized next, (CERT) can be solved in closed-form.

Theorem 4.1. *Given the input example $(\boldsymbol{x}, y)$, let $\boldsymbol{x}_{\setminus 0} = (x_1, \ldots, x_d)$. We can efficiently compute a solution to* CERT *or decide that no solution exists. If a solution exists, then based on a guess of z_0, the solution has the form $\boldsymbol{z} = \left(z_0, \sqrt{z_0^2 \quad 1}\left(b_\alpha \boldsymbol{x} + \sqrt{1 \quad b_\alpha^2}\boldsymbol{x}^\perp\right)\right)$. Here, b_α depends on the adversarial budget α, and $\boldsymbol{x}^\perp$ is a unit vector orthogonal to $\boldsymbol{x} = \boldsymbol{x}_{\setminus 0}/\|\boldsymbol{x}_{\setminus 0}\|$ along $\boldsymbol{w}$.*

Remark 4.2. Note that according to Thm. 4.1, it is possible that, for a particular guess of z_0, we may not be able to find an adversarial example $\boldsymbol{x}$ that leads to a prediction that is inconsistent with $\boldsymbol{x}$, i.e., $h_w(\boldsymbol{x}) \neq h_w(\boldsymbol{x})$. Thus, for some t, we may have $|S_t'| < m$ in Alg. 2.

The minimization with respect to $\boldsymbol{w}$ in (4.1) can be performed by an iterative optimization procedure, which generates a sequence of classifiers $\{\boldsymbol{w}_t\}$. We update the classifier $\boldsymbol{w}_t$ according to an update rule A, which accepts as input the current estimate of the weight vector, the original training set, and an adversarial perturbation of the training set. The update rule produces as output a weight vector which approximately minimizes the robust loss L_{rob} in (4.1).

We now establish that for a gradient based update rule, the above adversarial training procedure will efficiently converge to a large-margin solution. Table 1 summarizes the results of this section and compares with the corresponding results in the Euclidean setting [5].

4.1 Fast convergence via gradient-based update

Consider Alg. 2 with $A(\boldsymbol{w}_t, S, S_t')$ being a gradient-based update with learning rate $\eta_t > 0$:

$$\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t \quad \eta_t/|S_t'| \sum_{(x,y)\in S_t'} \nabla_{w_t} l(\boldsymbol{x}, y; \boldsymbol{w}_t) \ ; \quad \boldsymbol{w}_{t+1} \quad \boldsymbol{w}_{t+1}/\sqrt{\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_{t+1}} \ , \tag{4.4}$$

where the normalization is performed to ensure that the update remains valid, i.e., $\mathbb{L}^d \setminus \partial H_w \neq \emptyset$.

To compute the update, we need to compute gradients of the outer minimization problem, i.e., $\nabla_w l_{\text{rob}}$ over S_t' (cf. (4.1)). However, this function is itself a maximization problem. We therefore compute

the gradient at the maximizer of this inner maximization problem. Danskin's Theorem [9, 1] ensures that this gives a valid descent direction. Given the closed form expression for the adversarial example $\boldsymbol{x}$ as per Thm. 4.1, the gradient of the loss is

$$\nabla_{\boldsymbol{w}} l(\boldsymbol{x}, y; \boldsymbol{w}) = f'(y(\boldsymbol{w}^T \boldsymbol{x}))\ \nabla_{\boldsymbol{w}} y(\boldsymbol{w}^T \boldsymbol{x}) = f'(y(\boldsymbol{w}^T \boldsymbol{x}))\ y\widehat{\boldsymbol{x}}^T,$$

where $y\widehat{\boldsymbol{x}}^T = y(\boldsymbol{x}_0^T, \boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_n^T)^T$. With Danskin's theorem, $\nabla l(\boldsymbol{x}, y; \boldsymbol{w}) \in \partial l_{\text{rob}}(\boldsymbol{x}, y; \boldsymbol{w})$, so we can compute the descent direction and perform the step in (4.4). We defer details to Appendix C.4.

4.1.1 Convergence analysis

We now establish that the above gradient-based update converges to a large-margin solution in polynomial time. For this analysis, we need the following assumptions:

Assumption 1. 1. The training set S is linearly separable with margin at least γ_H, i.e., there exists a $\boldsymbol{w} \in \mathbb{R}^{d+1}$, such that $y(\boldsymbol{w}^T \boldsymbol{x}) \geq \sinh(\gamma_H)$ for all $(\boldsymbol{x}, y) \in S$.

2. There exists constants $R_x, R_w \geq 0$, such that (i) $\|\boldsymbol{x}\| \leq R_x$; (ii) all possible adversarial perturbations remain within this constraint, i.e., $\|\boldsymbol{x}\| \leq R_x$; and (iii) $\|\boldsymbol{w}\| \leq R_w$. Let $R_\alpha := R_x R_w$.
3. the function $f(s)$, underlying the loss (cf. (3.1)), has the following properties: (i) $f(s) > 0\ \forall\ s$; (ii) $f'(s) < 0\ \forall\ s$; (iii) f is differentiable, and (iv) f is β-smooth.

In the rest of the section, we work with the following hyperbolic equivalent of the *logistic regression loss* that fulfills Assumption 1:

$$l(\boldsymbol{x}, y; \boldsymbol{w}) = \ln\left(1 + \exp\left(-\operatorname{asinh}\left(y(\boldsymbol{w}^T \boldsymbol{x})/2R_\alpha\right)\right)\right), \tag{4.5}$$

where R_α is as defined in Assumption 1. Other loss functions as well as the derivation of the hyperbolic logistic regression loss are discussed in Appendix C.1.

We first show that Alg. 2 with a gradient update is guaranteed to converge to a large-margin classifier.

Theorem 4.3. *With constant step size and A being the GD update with an initialization $\boldsymbol{w}_0$ with $\boldsymbol{w}_0^T \boldsymbol{w}_0 < 0$, $\lim_{t\to\infty} L(\boldsymbol{w}_t; S \cup S'_t) = 0$.*

The proof can be found in Appendix C.4. While this result guarantees convergence, it does not guarantee efficiency (e.g., by showing a polynomial convergence rate). The following result shows that Alg. 2 with a gradient based update obtains a max-margin classifier in polynomial time.

Theorem 4.4. *For a fixed constant $c \in (0, 1)$, let $\eta_t = \eta := c\ \frac{2\sinh^2(\gamma_H)}{\beta\sigma_{\max}^2 \cosh^2(\alpha) R_\alpha^2}$ with $\sigma_{\max}$ denoting an upper bound on the maximum singular value of the data matrix $\sum_{x\in\mathcal{S}^0} \boldsymbol{x}\boldsymbol{x}^T$, and A the GD update as defined in* (4.4)*. Then, Alg. 2 achieves the margin $\gamma_H/\cosh(\alpha)$ in $\left(\text{poly}\left(\cosh(\alpha)/\sinh(\gamma_H)\right)\right)$ steps.*

Below, we briefly sketch some of the proof ideas, but defer a detailed exposition to Appendix C.4 (cf. Thm. C.13 and C.14). To prove the gradient-based convergence result, we first analyze the convergence of an "adversarial perceptron", that resembles the adversarial GD in that it performs updates of the form $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + y\boldsymbol{x}$. We then extend the analysis to the adversarial GD, where the convergence analysis builds on classical ideas from convex optimization.

The following auxiliary lemma relates the adversarial margin to the max-margin classifier.

Lemma 4.5. *Let $\boldsymbol{w}$ be the max-margin classifier of S with margin γ_H. At each iteration of Algorithm 2, $\boldsymbol{w}$ linearly separates $S \cup S'$ with margin at least $\frac{\gamma_H}{\cosh(\alpha)}$.*

The result follows from geometric arguments, as discussed in Section C.3. With the help of this lemma, we can show the following bound on the sample complexity of the adversarial perceptron:

Theorem 4.6. *Assume that there is some $\boldsymbol{w} \in \mathbb{R}^{d+1}$ with $\sqrt{\boldsymbol{w}^T \boldsymbol{w}} = 1$ and $\boldsymbol{w}_0^T \boldsymbol{w} \geq 0$, and some $\gamma_H > 0$, such that $y_j(\boldsymbol{w}^T \boldsymbol{x}_j) \geq \sinh(\gamma_H)$ for $j = 1, \ldots, |S|$. Then, the adversarial perceptron (with adversarial budget α) converges after $O\left(\frac{\cosh(\alpha)}{\sinh(\gamma_H)}\right)$ steps, at which it has margin of at least $\frac{\gamma_H}{\cosh(\alpha)}$.*

The technical proof can be found in Section C.3.

4.2 On the necessity of combining gradient descent and adversarial training

We remark here that the enrichment of the training set with adversarial examples is critical for the polynomial-time convergence. Recall first that by Thm. 3.3, without adversarial training, we can construct a simple max-margin problem that cannot be solved in polynomial time. Interestingly, merely using adversarial examples by themselves does not suffice for fast convergence either.

Consider Alg. 2 with an ERM as the update rule $A(\boldsymbol{w}_t, S, S')$. In this case, the iterate $\boldsymbol{w}_{t+1}$ corresponds to an ERM solution for $S \cup S'$, i.e.,

$$\boldsymbol{w}_{t+1} \quad \text{argmin}_{w} \sum_{(x,y)\in S\cup S^0} l(\boldsymbol{x}, y; \boldsymbol{w}) \ . \tag{4.6}$$

Let $S_t = S$, i.e., we utilize the full power of the adversarial training in each step. The following result reveals that even under this optimistic setting, Alg. 2 may not converge to a solution with a non-trivial margin in polynomial time:

Theorem 4.7. *Suppose Alg. 2 (with an ERM update) outputs a linear separator of $S \cup S'$. In the worst case, the number of iteration required to achieve a margin at least ϵ is* $\exp(d)$.

We note that a similar result in the Euclidean setting appears in [5]. We establish Thm. 4.7 by extending the proof strategy of [5, Thm. 4] to hyperbolic spaces. In particular, given a spherical code in $\mathbb{R}^d$ with T codewords and $\theta \sinh(\epsilon)\cosh(\alpha)$ minimum separation, we construct a training set S and subsequently the adversarial examples $\{S'_t\}$ such that there exists a sequence of ERM solutions $\{\boldsymbol{w}_t\}_{t\le T}$ on $S \cup S'$ (cf. (4.6)) that has margin less than ϵ. Now the result in Thm. 4.7 follows by utilizing a lower bound [7] on the size of the spherical code with $T = \exp(d)$ codewords and $\theta \sinh(\epsilon)\cosh(\alpha)$ minimum separation. The proof of Thm. 4.7 is in Appendix C.5.

5 Dimension-distortion trade-off

So far we have focused on classifying data that is given in either Euclidean spaces $\mathbb{R}^d$ or Lorentz space $\mathbb{L}^{d^0}$. Now, consider data $(X, d_{\mathcal{X}})$ with similarity metric $d_{\mathcal{X}}$ that was embedded into the respective spaces. We assume access to maps $\phi_E : X \to \mathbb{R}^d$ and $\phi_H : X \to \mathbb{L}^{d^0}_+$ that embed X into the Euclidean space $\mathbb{R}^d$ and the upper sheet of the Lorentz space $\mathbb{L}^{d^0}_+$, respectively (cf. Remark A.1). Let c_E and c_H denote the multiplicative distortion induced by ϕ_E and ϕ_H, respectively (cf. § 2.2). Upper bounds on c_E and c_H can be estimated based on the structure of X and the embedding dimensions.

In this section, we address the natural question: *How does the distortion c_E, c_H impact our guarantees on the margin?* In the previous sections, we noticed that some of the guarantees scale with the dimension of the embedding space. Therefore, we want to analyze the trade-off between the higher distortion resulting from working with smaller embedding dimensions and the higher cost of training robust models due to working with larger embedding dimensions.

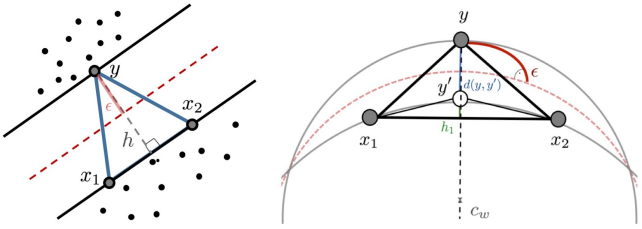

Figure 2: Margin as distance between support vectors. **Left:** Euclidean. **Right:** Hyperbolic.

We often encounter data sets in ML applications that are intrinsically hierarchical. Theoretical results on the embeddability of trees (cf. § 2.2) suggest that hyperbolic spaces are especially suitable to represent hierarchical data. We therefore restrict our analysis to such data. Further, we make the following assumptions on the underlying data X and the embedding maps, respectively.

Assumption 2. (1) Both $\phi_H(X)$ and $\phi_E(X)$ are linearly separable in the respective spaces, and (2) X is hierarchical, i.e., has a partial order relation.

Assumption 3. The maps ϕ_H, ϕ_E preserve the partial order relation in X and the root is mapped onto the origin of the embedding space.

Towards understanding the impact of the distortion of the embedding maps ϕ_H and ϕ_E on margin, we relate the distance between the support vectors to the size of the margin. The distortion of these distances via embedding then gives us the desired bounds on the margin.

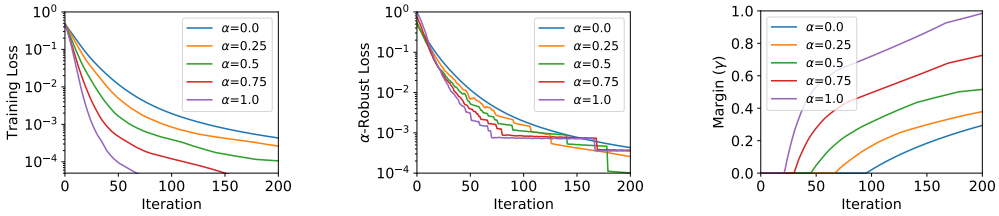

Figure 3: Performance of Adversarial GD. **Left:** Loss $L(\boldsymbol{w})$ on the original data. **Middle:** α-robust loss $L_\alpha(\boldsymbol{w})$. **Right:** Hyperbolic margin γ_H. We vary the adversarial budget α over $\{0, 0.25, 0.5, 0.75, 1.0\}$. Note that $\alpha = 0$ corresponds to the state of the art [6].

5.1 Euclidean case

In the Euclidean case, we relate the distance of the support vectors to the size of the margin via triangle relations. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ denote support vectors, such that $\langle \boldsymbol{x}, \boldsymbol{w} \rangle > 0$ and $\langle \boldsymbol{y}, \boldsymbol{w} \rangle < 0$ and $\text{margin}(\boldsymbol{w}) = \epsilon$. Note that we can rotate the decision boundary, such that the support vectors are not unique. So, without loss of generality, assume that $\boldsymbol{x}_1, \boldsymbol{x}_2$ are equidistant from the decision boundary and $\|\boldsymbol{w}\| = 1$ (cf. Fig. 2(left)). In this setting, we show the following relation between the margin with and without the influence of distortion:

Theorem 5.1. *Let ϵ' and ϵ denote the margin with and without distortion, respectively. If X is a tree embedded into $\mathbb{R}^{O(\log^2 |\mathcal{X}|)}$, then $\epsilon' = O\left(\epsilon / \log^3 |X|\right)$.*

The proof of Thm. 5.1 follows from a simple side length-altitude relations in the Euclidean triangle between support vectors (cf. Fig. 2(left)) and a simple application of Bourgain's result on embedding trees into $\mathbb{R}^d$. For more details see Appendix D.1.

5.2 Hyperbolic case

As in the Euclidean case, we want to relate the margin to the pairwise distances of the support vectors. Such a relation can be constructed both in the original and in the embedding space, which allows us to study the influence of distortion on the margin in terms of c_H. In the following, we will work with the half-space model $\mathbb{P}^{d^0}$ (cf. Appendix A.1). However, since the theoretical guarantees in the rest of the paper consider the Lorentz model $\mathbb{L}_+^{d^0}$, we have to map between the two spaces. We show in Appendix D.2 that such a mapping exists and preserves the Minkowski product, following [6].

The hyperbolic embedding ϕ_H has two sources of distortion: (1) the multiplicative distortion of pairwise distances, measured by the factor $1/c_H$; and (2) the distortion of order relations, in most embedding models captured by the alignment of ranks with the Euclidean norm. Under Assumption 3, order relationships are preserved and the root is mapped to the origin. Therefore, for $x \in X$, the distortion on the Euclidean norms is given as $\|\phi_H(x)\| = d_E(\phi_H(x), \phi_H(0)) = d_{\mathcal{X}}(x, 0)/c_H$, i.e., the distortion on both pairwise distances and norms is given by a factor $1/c_H$.

In $\mathbb{P}^{d^0}$, the decision hyperplane corresponds to a hypercircle K_w. We express its radius r_w in terms of the hyperbolic distance between a point on the decision boundary and one of the hypercircle's ideal points [6]. The support vectors $\boldsymbol{x}, \boldsymbol{y}$ lie on hypercircles K_x and K_y, which correspond to the set of points of hyperbolic distance ϵ (i.e., the margin) from the decision boundary. We again assume, without loss of generality, that at least one support vector is not unique and let $\boldsymbol{x}_1, \boldsymbol{x}_2 \in K_x$ and $\boldsymbol{y} \in K_y$ (cf. Fig. 2(right)). We now show that the distortion introduced by ϕ_H has a negligible effect on the margin.

Theorem 5.2. *Let ϵ' and ϵ denote the margin with and without distortion, respectively. If X is a tree embedded into $\mathbb{L}_+^2$, then ϵ' ϵ.*

The technical proof relies on a construction that reduces the problem to Euclidean geometry via circle inversion on the decision hypercircle. We defer all details to Appendix D.2.

6 Experiments

We now present empirical studies for hyperbolic linear separator learning to corroborate our theory. In particular, we evaluate our proposed Adversarial GD algorithm (§4) on data that is linearly separable in hyperbolic space and compare with the state of the art [6]. Furthermore, we analyze dimension-

distortion trade-offs (§5). As in our theory, we train hyperbolic linear classifiers $\boldsymbol{w}$ whose prediction on $\boldsymbol{x}$ is $y = \mathsf{sgn}(\boldsymbol{w} \quad \boldsymbol{x})$. Additional experimental results can be found in Appendix F.

We emphasise that our focus in this work is in theoretically understanding the benefits of hyperbolic spaces for classification. The above experiments serve to illustrate our theoretical results, which as a starting point were derived for linear models and separable data. While extensions to non-separable data and non-linear models are of practical interest, a detailed study is left for future work.

Data. We use the ImageNet ILSVRC 2012 dataset [26] along with its label hierarchy from wordnet. Hyperbolic embeddings in Lorentz space are obtained for the internal label nodes and leaf image nodes using Sarkar's construction [28]. For the first experiment, we verify the effectiveness of adversarial learning by picking two classes (n09246464 and n07831146), which allows for a data set that is linearly separable in hyperbolic space. In this set, there were 1,648 positive and 1,287 negative examples. For the second experiment, to showcase better representational power of hyperbolic spaces for hierarchical data, we pick two disjoint subtrees (n00021939 and n00015388) from the hierarchy.

Adversarial GD. In the following, we utilize the hyperbolic hinge loss (C.2), (see Appendix F for other loss functions). To verify the effectiveness of adversarial training, we compute three quantities: (i) loss on original data $L(\boldsymbol{w})$, (ii) α-robust loss $L_\alpha(\boldsymbol{w})$, and (iii) the hyperbolic margin γ. We vary the budget α over $\{0, 0.25, 0.5, 0.75, 1.0\}$, where $\alpha = 0$ corresponds to the setup in [6]. For a given budget, we obtain adversarial examples by solving the CERT problem (4.3), which is feasible for $z_0 \in (x_0 \cosh(\alpha) \quad , x_0 \cosh(\alpha) + \quad)$, where $= \sqrt{(x_0^2 \quad 1)(\cosh^2(\alpha) \quad 1)}$. We do a binary search in this range for z_0, solve CERT and check if we can obtain an adversarial example. We utilize the adversarial examples we find, and ignore other data points. In all experiments, we use a constant step-size $\eta_t = 0.01/\sqrt{t}$. The results are shown in Fig. 3. As α increases the problem becomes harder to solve (higher training robust loss) but we achieve a better margin. Notably, we observe strong performance gains over the training procedures without adversarial examples [6].

Dimensional efficiency. In this experiment, we illustrate the benefit of using hyperbolic space when the underlying data is truly hierarchical. To be more favourable to Euclidean setting, we subsample images from each subtree, such that in total we have 1000 vectors. We obtain Euclidean embeddings following the setup and code from Nickel and Kiela [22]. The Euclidean embeddings in 16 dimensions reach a mean rank (MR) 2, which indicates reasonable quality in preserving distance to few-hop neighbors. We observe superior classification performance at much lower dimensions by leveraging hyperbolic space (see Table 2 in Appendix F.3). In particular, our hyperbolic classifier achieves zero test error on 8-dimensional embeddings, whereas Euclidean logistic regression struggles even with 16-dimensional embeddings. This is consistent with our theoretical results (§5): Due to high distortion, lower-dimensional Euclidean embeddings struggle to capture the global structure among the data points that makes the data points easily separable.

7 Conclusion and future work

We studied the problem of learning robust classifiers with large margins in hyperbolic space. We introduced and analyzed a hyperbolic perceptron algorithm. Moreover, we explored multiple adversarial approaches to robust large-margin learning. The second part of the paper analyzed the role of geometry in learning robust classifiers. We compared Euclidean and hyperbolic approaches with respect to the intrinsic geometry of the data. For hierarchical data that embeds well into hyperbolic space, the lower embedding dimension ensures superior guarantees when learning the classifier in hyperbolic space. This result suggests that it can be highly beneficial to perform downstream machine learning and optimization tasks in a space that naturally reflects the intrinsic geometry of the data. Promising avenues for future research include (i) exploring the practicality of these results in broader machine learning and data science applications; and (ii) studying other related methods in non-Euclidean spaces, together with an evaluation of dimension-distortion trade-offs.

Acknowledgements

We thank Pranjal Awasthi for helpful discussions on computing adversarial examples and Eli Chien for helpful comments on an earlier version of the paper.

Broader Impact

The paper proposes novel algorithms for large-margin learning in hyperbolic space. The paper's scope is theoretical and does not discuss specific applications with societal consequences. Therefore, a discussion of the broader impact of this work is not applicable.

References

[1] D.P. Bertsekas. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 2016.

[2] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, Mar 1985.

[3] Martin R. Bridson and André Haefliger. *Metric Spaces of Non-Positive Curvature*, volume 319 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-642-08399-0 978-3-662-12494-9. URL `http://link.springer.com/10.1007/978-3-662-12494-9`.

[4] Benjamin Chamberlain, James Clough, and Marc Deisenroth. Neural embeddings of graphs in hyperbolic space. In *arXiv:1705.10359 [stat.ML]*, 2017.

[5] Zachary Charles, Shashank Rajput, Stephen Wright, and Dimitris Papailiopoulos. Convergence and margin of adversarial training on separable data. *arXiv:1905.09209*, 2019.

[6] Hyunghoon Cho, Benjamin DeMeo, Jian Peng, and Bonnie Berger. Large-margin classification in hyperbolic space. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1832–1840. PMLR, 16–18 Apr 2019.

[7] Henry Cohn and Yufei Zhao. Sphere packing bounds via spherical codes. *Duke Math. J.*, 163 (10):1965–2002, 07 2014.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[9] John M. Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664, 1966.

[10] Laurent El Ghaoui, Gert R. G. Lanckriet, and Georges Natsoulis. Robust classification with interval data. Technical Report UCB/CSD-03-1279, EECS Department, University of California, Berkeley, Oct 2003.

[11] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 1186–1195, 2018.

[12] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, Dec 1999.

[13] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in neural information processing systems*, pages 5345–5355, 2018.

[14] Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *ArXiv*, abs/1803.07300, 2018.

[15] Seung-Jean Kim, Alessandro Magnani, and Stephen Boyd. Robust fisher discriminant analysis. In *Advances in neural information processing systems*, pages 659–666, 2006.

[16] Gert R.G. Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I. Jordan. A robust minimax approach to classification. *J. Mach. Learn. Res.*, 3:555–582, March 2003. ISSN 1532-4435.

[17] Guy Lebanon and John Lafferty. Hyperplane margin classifiers on the multinomial manifold. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, 2004.

[18] Yan Li, Ethan X.Fang, Huan Xu, and Tuo Zhao. Implicit bias of gradient descent based adversarial training on separable data. In *International Conference on Learning Representations*, 2020.

[19] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICRL*, 2018.

[21] Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 714–722, 2019.

[22] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30*, pages 6338–6347, 2017.

[23] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, 2018.

[24] A. B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.

[25] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. In *International journal of computer vision*, 2015.

[27] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4460–4469, 2018.

[28] Rik Sarkar. Low distortion Delaunay embedding of trees in hyperbolic plane. In *Graph Drawing*, pages 355–366, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-25878-7.

[29] Claude E Shannon. Probability of error for optimal codes in a gaussian channel. *Bell System Technical Journal*, 38(3):611–656, 1959.

[30] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *JMLR*, 19:1–57, 2018.

[31] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. *ICRL*, 2019.

[32] Melanie Weber. Neighborhood growth determines geometric priors for relational representation learning. In *International Conference on Artificial Intelligence and Statistics*, volume 108, pages 266–276, 2020.

A Hyperbolic Space

Hyperbolic spaces are smooth Riemannian manifolds $M = \mathsf{H}^d$ and as such locally Euclidean spaces. In the following we introduce basic notation for three popular models of hyperbolic spaces. For a comprehensive overview see Bridson and Haefliger [3].

A.1 Models of hyperbolic spaces

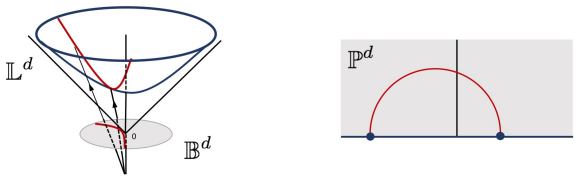

Figure 4: Models of hyperbolic space: The Lorentz model L^d, the Poincare ball B^d, and the Poincare half-plane P^d.

The Poincare ball defines a hyperbolic space within the Euclidean unit ball, i.e.

$$\mathsf{B}^d = \{\boldsymbol{x} \in \mathsf{R}^d : \|\boldsymbol{x}\| < 1\}$$
$$d_{\mathsf{B}}(\boldsymbol{x}, \boldsymbol{x}') = \operatorname{acosh}\left(1 + 2\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{(1 - \|\boldsymbol{x}\|^2)(1 - \|\boldsymbol{x}'\|^2)}\right) .$$

Here, $\|\cdot\|$ is the usual Euclidean norm.

The closely related Poincare half-plane model is defined as

$$\mathsf{P}^2 = \{\boldsymbol{x} \in \mathsf{R}^2 : x_1 > 0\}$$
$$d_{\mathsf{P}}(\boldsymbol{x}, \boldsymbol{x}') = \operatorname{acosh}\left(1 + \frac{(x'_0 - x_0)^2 + (x'_1 - x_1)^2}{2x_1 x'_1}\right) .$$

Note that if $x_0 = x'_0$, the metric simplifies as

$$d_{\mathsf{P}}(\boldsymbol{x}, \boldsymbol{x}') = d_{\mathsf{P}}((x_0, x_1), (x_0, x'_1)) = \left|\ln \frac{x'_1}{x_1}\right| .$$

The model can be generalized to higher dimensions with

$$\mathsf{P}^d = \{(x_0, \ldots, x_{d-1}) \in \mathsf{R}^d \mid x_{d-1} > 0\} ,$$

however, we will only use the two-dimensional model P^2 here. We further define the hyperboloid as

$$\mathsf{L}^d = \{\boldsymbol{x} \in \mathsf{R}^{d+1} : \boldsymbol{x} \bullet \boldsymbol{x} = 1\}$$
$$d_{\mathsf{L}}(\boldsymbol{x}, \boldsymbol{x}') = \operatorname{acosh}(\boldsymbol{x} \bullet \boldsymbol{x}') ,$$

where $\bullet$ denotes the Minkowski product $\boldsymbol{x} \bullet \boldsymbol{x}' = x_0 x'_0 - \sum_{i=1}^d x_i x'_i$.

Remark A.1. The Lorentz model

$$\mathsf{L}^d = \{x \in \mathsf{R}^{d+1} : \boldsymbol{x} \bullet \boldsymbol{x} = 1\} .$$

is also called double-sheet model. We use this more general setting in sections 2-4. For simplicity, we restrict ourselves to the upper sheet

$$\mathsf{L}^d_+ = \{\boldsymbol{x} \in \mathsf{R}^{d+1} : \boldsymbol{x} \bullet \boldsymbol{x} = 1,\ x_0 > 0\} ,$$

in section 5. All constructions of mappings between the different models of hyperbolic space can be extended to the double-sheet L^d.

A.2 Equivalence of different models of hyperbolic spaces

The Poincare ball $\mathbb{B}^d$ and the Lorentz model $\mathbb{L}_+^d$ are equivalent models of hyperbolic space. A mapping is given by

$$\pi_{LB} : \mathbb{L}_+^d \to \mathbb{B}^d$$
$$\boldsymbol{x} = (x_0, \ldots, x_d) \mapsto \left(\frac{x_1}{1 + x_0}, \ldots, \frac{x_d}{1 + x_0} \right) .$$

We can further construct a mapping from $\mathbb{B}^d$ to $\mathbb{P}^d$ by inversion on a circle centered at $(-1, 0, \ldots, 0)$:

$$\pi_{BP} : \mathbb{B}^d \to \mathbb{P}^d$$
$$\boldsymbol{x} = (x_0, \ldots, x_{d-1}) \mapsto \frac{(2x_1, \ldots, 2x_{d-1}, 1 - \|\boldsymbol{x}\|^2)}{1 + 2x_0 + \|\boldsymbol{x}\|^2} .$$

A.3 Embeddability

When analyzing the dimension-distortion trade-off, we make use of two key results on the embeddability (cf. §2.2) of trees into Euclidean and hyperbolic spaces. We state them below for reference.

Theorem A.2 ([2]). *An N-point metric X (i.e., $|X| = N$) embeds into Euclidean space $\mathbb{R}^{O(\log^2 N)}$ with the distortion $c_M = O(\log N)$.*

This bound in Theorem A.2 is tight for trees in the sense that embedding them in a Euclidean space (of any dimension) must incur the distortion $c_m = \Omega(\log N)$ [19].

Theorem A.3 ([28]). *Tree metrics embed quasi-isometrically with $c_M = O(1 + \epsilon)$ into $\mathbb{H}^d$.*

A.4 Spherical codes in hyperbolic space

Consider the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$. A *spherical code* is a subset of $\mathbb{S}^{d-1}$, such that any two distinct elements $\boldsymbol{x}, \boldsymbol{x}'$ are separated by at least an angle θ, i.e. $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle \leq \cos\theta$. We denote the size of the largest code as $A(d, \theta)$.

A similar construction of such "spherical caps" can be obtained in $\mathbb{H}^d$. Note that the induced geometry of these caps is spherical, hence they inherit a spherical geometric structure. This allows in particular the transfer of bounds on $A(d, \theta)$ to hyperbolic space [7]:

Theorem A.4 (Chabauty, Shannon, Wyner (see, e.g., [29])). $A(d, \theta) \geq (1 + o(1)) \sqrt{2\pi d} \frac{\cos\theta}{\sin^{d-1}\theta}$.

B Hyperbolic Perceptron

In this section we analyze the convergence and generalization properties of the hyperbolic perceptron (cf. Algorithm 1). Note that the update $\boldsymbol{v}_t \leftarrow \boldsymbol{w}_t + y_j \boldsymbol{x}_j$ in Algorithm 1 always leads to a valid hyperplane, i.e., $\mathbb{L}^d \setminus H_{\boldsymbol{v}_t} \neq \emptyset$, which happens iff $\boldsymbol{v}_t \circ \boldsymbol{v}_t < 0$. This can be verified as follows:

$$\boldsymbol{v}_t \circ \boldsymbol{v}_t = (\boldsymbol{w}_t + y_j \boldsymbol{x}_j) \circ (\boldsymbol{w}_t + y_j \boldsymbol{x}_j) = \underbrace{\boldsymbol{w}_t \circ \boldsymbol{w}_t}_{\overset{(i)}{\leq -1}} + 2 \underbrace{y_j (\boldsymbol{x}_j \circ \boldsymbol{w}_t)}_{\overset{(ii)}{< 0}} + \underbrace{y^2}_{=1} \underbrace{(\boldsymbol{x}_j \circ \boldsymbol{x}_j)}_{\overset{(iii)}{=} 1} < 0 ,$$

where (i) is a consequence of the normalization step in Algorithm 1 and (iii) follows as $\boldsymbol{x} \circ \boldsymbol{x} = 1$, since $\boldsymbol{x} \in \mathbb{L}^d$. As for (ii), note that we perform the update $\boldsymbol{v}_t \leftarrow \boldsymbol{w}_t + y_j \boldsymbol{x}_j$ only when $y_j \neq \operatorname{sign}(\boldsymbol{x}_j \circ \boldsymbol{w})$ (cf. Algorithm 1).

We now restate Theorem 3.1 and present a detailed proof of the result.

Theorem B.1 (Convergence hyperbolic Perceptron in Algorithm 1 (Theorem 3.1)). *Assume that there is some $\boldsymbol{w} \in \mathbb{R}^{d+1}$ with $\sqrt{\boldsymbol{w} \circ \boldsymbol{w}} = 1$ and $\boldsymbol{w}_0 \circ \boldsymbol{w} \geq 0$, and some $\gamma_H > 0$, such that $y_j(\boldsymbol{w} \circ \boldsymbol{x}_j) \geq \sinh(\gamma_H)$ for $j = 1, \ldots, |S|$. Then, Algorithm 1 converges in $O\left(\frac{1}{\sinh(\gamma_H)}\right)$ steps and returns a solution with margin γ_H.*

Proof. Assume wlog $\boldsymbol{w}_0 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^{d+1}$. Then $\boldsymbol{w}_0 \quad \boldsymbol{w}_0 = \quad 1$, i.e., $\mathbb{L}^d \setminus H_{w_0} \neq \emptyset$. Hence, $\boldsymbol{w}_0$ is a valid initialization. Furthermore, assume that the tth error is made at the jth sample, i.e. update $\boldsymbol{v}_t \quad \boldsymbol{w}_t + y_j \boldsymbol{x}_j$. For $\boldsymbol{u} \in \mathbb{R}^{d+1}$, let $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u} \quad \boldsymbol{u}}$.

Now let us consider two cases:

Case 1. In this case, we assume that the normalization is not performed in tth step, i.e.,

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_j \boldsymbol{x}_j.$$

Therefore,

$$\boldsymbol{w}_{t+1} \quad \boldsymbol{w} = (\boldsymbol{w}_t + y_j \boldsymbol{x}_j) \quad \boldsymbol{w} = \boldsymbol{w}_t \quad \boldsymbol{w} + y_j(\boldsymbol{x}_j \quad \boldsymbol{w}) \quad \boldsymbol{w}_t \quad \boldsymbol{w} + \underbrace{\gamma'_H}_{:=\sinh(\gamma_H)} . \tag{B.1}$$

Case 2. In this case, the normalization is performed in the tth step of Algorithm 1, i.e.,

$$\boldsymbol{w}_{t+1} = \frac{\boldsymbol{w}_t + y_j \boldsymbol{x}_j}{\|\boldsymbol{w}_t + y_j \boldsymbol{x}_j\|}.$$

Thus,

$$\begin{aligned} \boldsymbol{w}_{t+1} \quad \boldsymbol{w} = \frac{\boldsymbol{w}_t + y_j \boldsymbol{x}_j}{\|\boldsymbol{w}_t + y_j \boldsymbol{x}_j\|} \quad \boldsymbol{w} \overset{(i)}{} (\boldsymbol{w}_t + y_j \boldsymbol{x}_j) \quad \boldsymbol{w} \\ \boldsymbol{w}_t \quad \boldsymbol{w} + \underbrace{y_j(\boldsymbol{x}_j \quad \boldsymbol{w})}_{\geq \gamma_H^0} \quad \boldsymbol{w}_t \quad \boldsymbol{w} + \gamma'_H , \end{aligned} \tag{B.2}$$

where (i) follows as the normalization is performed only if $\|\boldsymbol{w}_t + y_j \boldsymbol{x}_j\| < 1$ and numerator is positive by induction.

By utilizing (B.1) and (B.2), we obtain the following telescoping sum

$$\begin{aligned} \sum_{k=0}^{T-1} (\quad \boldsymbol{w}_{k+1} + \boldsymbol{w}_k) \quad \boldsymbol{w} \quad \sum_{k=0}^{T-1} \quad \gamma'_H \\ ) \quad \boldsymbol{w}_T \quad \boldsymbol{w} \qquad \boldsymbol{w}_0 \quad \boldsymbol{w} \quad T\gamma'_H . \end{aligned} \tag{B.3}$$

Recall that, for the Minkowski product, we have

$$\cosh(\,]\,(\boldsymbol{u}, \boldsymbol{u}')) = \quad \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \frac{\boldsymbol{u}'}{\|\boldsymbol{u}'\|} \quad = \frac{\boldsymbol{u} \quad \boldsymbol{u}'}{\|\boldsymbol{u}\| \, \|\boldsymbol{u}'\|} . \tag{B.4}$$

By utilizing (B.4) with $(\boldsymbol{u}, \boldsymbol{u}') = (\boldsymbol{w}_T, \boldsymbol{w})$ and $(\boldsymbol{u}, \boldsymbol{u}') = (\boldsymbol{w}_0, \boldsymbol{w})$ in (B.3), we obtain that

$$\|\boldsymbol{w}_T\| \|\boldsymbol{w}\| \cosh(\,]\,(\boldsymbol{w}_T, \boldsymbol{w})) \quad \|\boldsymbol{w}_0\| \|\boldsymbol{w}\| \cosh(\,]\,(\boldsymbol{w}_0, \boldsymbol{w})) \quad T\gamma'_H. \tag{B.5}$$

Since, we have $\|\boldsymbol{w}\| = \|\boldsymbol{w}_0\| = 1$, it follows from (B.5) that

$$\|\boldsymbol{w}_T\| \cosh(\,]\,(\boldsymbol{w}_T, \boldsymbol{w})) \quad \cosh(\,]\,(\boldsymbol{w}_0, \boldsymbol{w})) \quad T\gamma'_H. \tag{B.6}$$

Further, using the facts that, due to normalization in Algorithm 1, $\|\boldsymbol{w}_T\| \quad 1$ and $\cosh(\,) \quad 1$, it follows from (B.6) that

$$1 \quad \cosh(\,]\,(\boldsymbol{w}_0, \boldsymbol{w})) \quad T\gamma'_H \overset{(ii)}{} C \quad T\gamma'_H, \tag{B.7}$$

where (ii) follows as $]\,(\boldsymbol{w}_i, \boldsymbol{w}) < \pi$, since the orientation is fixed by the requirement that $\mathbb{L}^d \setminus H_{w_i} \neq \emptyset$; as a result, we can find an upper bound $\cosh(\,]\,(\boldsymbol{w}_0, \boldsymbol{w})) < \cosh(\pi) = C$. Now, it follows from (B.7) that

$$T \quad \frac{C \quad 1}{\gamma'_H}, \tag{B.8}$$

which completes the proof of the convergence guarantee. The margin is given by

$$\text{margin}_{\mathcal{S}}(w) = \inf_{(x,y)\in\mathcal{S}} \operatorname{asinh}\left(\frac{y(\boldsymbol{w} \quad \boldsymbol{x})}{\|\boldsymbol{w}\| \, \|\boldsymbol{w}\|} \right) = \operatorname{asinh}(\gamma'_H) = \operatorname{asinh}\left(\sinh(\gamma_H)\right) = \gamma_H ,$$

which implies that a margin of γ_H is achieved in $O\left(\frac{1}{\sinh(\gamma_H)}\right)$ steps. □

C Adversarial Learning

C.1 Loss functions

For training the classifier, we consider the margin losses that have the following form

$$l(\boldsymbol{x}, y; \boldsymbol{w}) = f(y \quad (\boldsymbol{w} \quad \boldsymbol{x})), \tag{C.1}$$

where $f : \mathsf{R} \; ! \; \mathsf{R}_+$ is some convex, non-increasing function. Cho et al. [6] introduce the *hinge loss* in the hyperbolic setting which is defined by the (hyperbolic) hinge function $f(s) = \max\{0, \operatorname{asinh}(1) \; \operatorname{asinh}(s)\}$, i.e.,

$$l(\boldsymbol{x}, y; \boldsymbol{w}) = \max\{0, \operatorname{asinh}(1) \quad \operatorname{asinh}(y(\boldsymbol{w} \quad \boldsymbol{x}))\} . \tag{C.2}$$

A significant shortcoming of this notion is its non-smoothness and non-convexity. Therefore, we additionally consider a smoothed *least squares loss*:

$$l(\boldsymbol{x}_i, y_i; \boldsymbol{w}) = \begin{cases} \frac{1}{2}(\operatorname{asinh}(1) \quad \operatorname{asinh}(y_i(\boldsymbol{w} \quad \boldsymbol{x}_i))^2, & y_i(\boldsymbol{w} \quad \boldsymbol{x}_i) \quad 1 \\ 0, & \text{else} \end{cases} , \tag{C.3}$$

We present experimental results for both losses.

The majority of the paper employs a hyperbolic version of the logistic loss to introduce the logistic regression problem in hyperbolic space. First, recall the logistic regression problem in the Euclidean setting. Given an input $\boldsymbol{x}$ and a linear classifier defined by $\boldsymbol{w}$, the prediction of the classifier is defined as

$$p(y|\boldsymbol{x}; \boldsymbol{w}) = 1/\big(1 + \exp(\quad y\langle \boldsymbol{x}, \boldsymbol{w}\rangle)\big) \tag{C.4}$$

Thus the log-loss takes the following form

$$\begin{aligned} l(\boldsymbol{x}, y; \boldsymbol{w}) = \quad -\log p(y|\boldsymbol{x}; \boldsymbol{w}) &= \log\big(1 + \exp(\quad y\langle \boldsymbol{x}, \boldsymbol{w}\rangle)\big) \\ &= \log\big(1 + \exp(\quad y\|\boldsymbol{w}\|\langle \boldsymbol{x}, \tfrac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\rangle)\big) \\ &= \log\big(1 + \exp(\quad y\,\operatorname{sgn}(\langle \boldsymbol{x}, \boldsymbol{w}\rangle)\|\boldsymbol{w}\| d(\boldsymbol{x}, \partial H_{\boldsymbol{w}}))\big) \end{aligned} \tag{C.5}$$

where $\boldsymbol{w} = \boldsymbol{w}/\|\boldsymbol{w}\|$ and $d(\boldsymbol{x}, \partial H_{\boldsymbol{w}})$ is the distance of $\boldsymbol{x}$ from the decision boundary $\partial H_{\boldsymbol{w}} := \{\boldsymbol{z} \in \mathsf{R}^{d+1} : \langle \boldsymbol{z}, \boldsymbol{w}\rangle = 0\}$. Note that $y\,\operatorname{sgn}(\langle \boldsymbol{x}, \boldsymbol{w}\rangle) d(\boldsymbol{x}, \partial H_{\boldsymbol{w}}))$ denotes the Euclidean margin of the $(\boldsymbol{x}, y)$ with respect to the decision boundary defined by $\boldsymbol{w}$.

We can define a hyperbolic version of the logistic regression problem, where we replace the Euclidean margin with the hyperbolic margin with respect to the linear classifier $\boldsymbol{w}$. Recall that the hyperbolic margin has the following form (cf.. (2.2)):

$$y\,\operatorname{sgn}(\boldsymbol{x} \quad \boldsymbol{w}) d\big(\boldsymbol{x}, \partial H_{\boldsymbol{w}}\big) = y\,\operatorname{sgn}(\boldsymbol{x} \quad \boldsymbol{w}) \left|\operatorname{asinh}\left(\cancel{p}\frac{\boldsymbol{w} \quad \boldsymbol{x}}{\boldsymbol{w} \quad \boldsymbol{w}}\right)\right| = \operatorname{asinh}\left(\cancel{p}\frac{y(\boldsymbol{w} \quad \boldsymbol{x})}{\boldsymbol{w} \quad \boldsymbol{w}}\right) \tag{C.6}$$

Therefore, by combining (C.5) and (C.6), the hyperbolic logistic regression problem with a linear classifier corresponds to minimizing the following loss:

$$l(\boldsymbol{x}, y; \boldsymbol{w}) = \ln\left(1 + \exp\left(\quad \operatorname{asinh}\left(\cancel{p}\frac{y(\boldsymbol{w} \quad \boldsymbol{x})}{\boldsymbol{w} \quad \boldsymbol{w}}\right)\right)\right) . \tag{C.7}$$

Note that the hyperbolic logistic loss and the Euclidean logistic loss differ in the scaling factor $\|\boldsymbol{w}\|$. In order to ensure that the hyperbolic logistic loss satisfies Assumption 1, we introduce additional explicit scaling to obtain the following form of the loss.

$$l(\boldsymbol{x}, y; \boldsymbol{w}) = \ln\left(1 + \exp\left(\quad \operatorname{asinh}\left(\frac{y(\boldsymbol{w} \quad \boldsymbol{x})}{2R}\right)\right)\right) . \tag{C.8}$$

The following result verifies that the loss in (C.8) indeed satisfies Assumption 1.

Lemma C.1. *For valid inputs $(\boldsymbol{x}, y; \boldsymbol{w})$, the hyperbolic logistic loss in* (C.8) *fulfills Assumption 1.*

Proof. The robust loss (Eq. 4.1) is evaluated over inputs $(\boldsymbol{x}, y; \boldsymbol{w})$ only if $y(\boldsymbol{w} \quad \boldsymbol{x}) < 0$. A simple calculation shows, that Assumption 1.3 holds iff $\frac{|w*x|}{R_\alpha}$ 1, where R_α is as given in Assumption 1.2.

As a results, we want to show $\boldsymbol{jw} \quad \boldsymbol{xj} \quad R_\alpha$ for all allowable inputs $(\boldsymbol{x}, y; \boldsymbol{w})$. Recall that

$$\boldsymbol{w} \quad \boldsymbol{x} = w_0 x_0 \quad \sum_{i=1}^{d} w_i x_i$$

$$\boldsymbol{w} \quad \boldsymbol{x} = w_0 x_0 + \sum_{i=1}^{d} w_i x_i \ .$$

We consider the following cases:

1. $w_0 x_0 > 0$ *and* $\sum_{i=1}^{d} w_i x_i < 0$: $\boldsymbol{jw} \quad \boldsymbol{xj} \quad \boldsymbol{jw} \quad \boldsymbol{xj}$;
2. $w_0 x_0 > 0$ *and* $\sum_{i=1}^{d} w_i x_i > 0$: $\boldsymbol{jw} \quad \boldsymbol{xj} \quad \boldsymbol{jw} \quad \boldsymbol{xj}$;
3. $w_0 x_0 < 0$ *and* $\sum_{i=1}^{d} w_i x_i > 0$: $\boldsymbol{jw} \quad \boldsymbol{xj} \quad \boldsymbol{jw} \quad \boldsymbol{xj}$;
4. $w_0 x_0 < 0$ *and* $\sum_{i=1}^{d} w_i x_i < 0$: $\boldsymbol{jw} \quad \boldsymbol{xj} \quad \boldsymbol{jw} \quad \boldsymbol{xj}$.

In case (2) and (4) we have

$$\boldsymbol{jw} \quad \boldsymbol{xj} \quad \boldsymbol{jw} \quad \boldsymbol{xj} \overset{(i)}{} \boldsymbol{kwk}\, \boldsymbol{kxk} \overset{(ii)}{} R_x R_w = R_\alpha \ ,$$

where (i) follows from the Cauchy-Schwartz inequality and (ii) follows from Assumption 1.2. In case (1) and (3), we have

$$\boldsymbol{jw} \quad \boldsymbol{xj} = \boldsymbol{jw} \quad \hat{\boldsymbol{x}}\boldsymbol{j} \overset{(i)}{} \boldsymbol{kwk}\, \boldsymbol{k\hat{x}k} \overset{(ii)}{} R_x R_w = R_\alpha \ , \tag{C.9}$$

where $\hat{\boldsymbol{x}} = (x_0, \quad x_1, \ldots, \quad x_n)$ and (i) and (ii) again follow from the Cauchy-Schwartz inequality, respectively. This completes the proof. □

Remark C.2. A conceptually similar logistic loss is introduced in [17] for multinomial manifold. Max-margin learning with the above hyperbolic hinge loss was studied in [6].

C.2 Generating adversarial examples (Certification problem)

Recall that to train a classifier with large margin, we enrich the training set with adversarial examples (cf. Algorithm 2). For a classifier $\boldsymbol{w}$, an adversarial example $\boldsymbol{x}$ for a given $(\boldsymbol{x}, y)$ is generated by perturbing $\boldsymbol{x}$ in the hyperbolic space up to the maximum allowed perturbation budget α such that

$$\boldsymbol{x} \quad \underset{\substack{\boldsymbol{z} \in \mathbb{L}^d \\ d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{z}) \leq \alpha}}{\operatorname{argmax}} \ l(\boldsymbol{z}, y; \boldsymbol{w}) \ .$$

For the underlying loss function (cf. Section C.1), due to the monotonicity of asinh, the above problem can be equivalently expressed as

$$\begin{aligned} \boldsymbol{x} \quad & \underset{\substack{\boldsymbol{z} \in \mathbb{L}^d \\ d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{z}) \leq \alpha}}{\operatorname{argmin}} \ y \quad (\boldsymbol{w} \quad \boldsymbol{z}) = \underset{\substack{\boldsymbol{z} \in \mathbb{L}^d \\ d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{z}) \leq \alpha}}{\operatorname{argmax}} \quad \boldsymbol{w}' \quad \boldsymbol{z} \\ & = \underset{\substack{\boldsymbol{z} \in \mathbb{L}^d \\ d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{z}) \leq \alpha}}{\operatorname{argmax}} \quad w_0' z_0 + \sum_i w_i' z_i \end{aligned} \tag{C.10}$$

where $\boldsymbol{w}' = \quad y\boldsymbol{w}$. Since $\boldsymbol{w}', \boldsymbol{z} \ 2 \ \mathbb{R}^{d+1}$, we can rewrite (C.10) as a constraint optimization task in the ambient Euclidean space:

$$\begin{aligned} \max_{\boldsymbol{z} \in \mathbb{R}^{d+1}} \quad & w_0 z_0 + \sum_i w_i z_i \\ \text{s.t.} \quad & d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{z}) \quad \alpha \\ & z_0^2 \quad \sum_{i=1}^{d} z_i^2 = 1 \ . \end{aligned} \tag{C.11}$$

Assuming that we guess z_0 based on x_0, the constraint $z_0^2 \quad \sum_{i=1}^d z_i^2 = 1$ confines the solution space onto a d-dimensional sphere of radius $r = \sqrt{z_0^2 \quad 1}$, which also implies that $z_0 \quad 1$. On the other hand the constraint $d_{\mathsf{L}}(\boldsymbol{x}, \boldsymbol{z}) \quad \alpha$ is equivalent to

$$d_{\mathsf{L}}(\boldsymbol{x}, \boldsymbol{z}) = \mathsf{acosh}(\boldsymbol{x} \quad \boldsymbol{z}) = \mathsf{acosh}(x_0 z_0 \quad \sum_{i=1}^d x_i z_i) < \alpha \quad \text{or} \quad \sum_i \quad x_i z_i \quad \mathsf{cosh}(\alpha) \quad x_0 z_0 \ .$$

Thus, the problem in (C.11) reduces to the following linear program with a spherical constraint.

$$\begin{aligned} (\mathsf{CERT}) \quad \max_{\boldsymbol{z}_{n0} \in \mathsf{R}^d} \quad & w_0 z_0 + \sum_i w_i z_i \\ \text{s.t.} \quad & \sum_{i=1}^d \quad x_i z_i \quad \mathsf{cosh}(\alpha) \quad x_0 z_0 \\ & \|\boldsymbol{z}_{\setminus 0}\|^2 = z_0^2 \quad 1 \ , \end{aligned} \tag{C.12}$$

where $\boldsymbol{z}_{\setminus 0} = (z_1, \ldots, z_d)$. We now present a proof of Theorem 4.1 which characterizes a solution of the program in (C.12). For the sake of readability, we first restate the result from the main text:

Theorem C.3 (Theorem 4.1)**.** *Given the input example $(\boldsymbol{x}, y)$, let $\boldsymbol{x}_{\setminus 0} = (x_1, \ldots, x_d)$. We can efficiently compute a solution to (*CERT*) or decide that no solution exists. If a solution exists, then based on a guess of z_0 a maximizing adversarial example has the form $\boldsymbol{x} = \left(z_0, \sqrt{z_0^2 \quad 1}\left(b\boldsymbol{x} + \sqrt{1 \quad b^2}\boldsymbol{x}^\perp\right)\right)$. Here, $b = \frac{(\mathsf{cosh}(\alpha) \quad x_0 z_0)}{(\|x_{n0}\| \sqrt{z_0^2 - 1})}$ depends on the adversarial budget α, and $\boldsymbol{x}_{\setminus 0}^\perp$ is a unit vector orthogonal to $\boldsymbol{x} = \boldsymbol{x}_{\setminus 0}/\|\boldsymbol{x}_{\setminus 0}\|$ along $\boldsymbol{w}$.*

Proof. First, note that (CERT) can be rewritten as

$$\begin{aligned} (\mathsf{CERT}) \quad & \max \langle \boldsymbol{w}, \boldsymbol{z} \rangle \\ & s.t. \ \langle \boldsymbol{x}, \boldsymbol{z} \rangle \quad b \\ & \|\boldsymbol{z}\| = 1 \ , \end{aligned}$$

where $\boldsymbol{w} = \boldsymbol{w}_{\setminus 0}/\|\boldsymbol{w}_{\setminus 0}\|$, $\boldsymbol{x} = \quad \boldsymbol{x}_{\setminus 0}/\|\boldsymbol{x}_{\setminus 0}\|$, and $b = (\mathsf{cosh}(\alpha) \quad x_0 z_0)/(\|\boldsymbol{x}_{\setminus 0}\| \|\boldsymbol{z}_{\setminus 0}\|)$. We further set $\boldsymbol{z} = \boldsymbol{z}_{\setminus 0}/\|\boldsymbol{z}_{\setminus 0}\|$ so that the norm constraint confines the solution to the unit sphere to simplify the derivation. We can later rescale the solution to have the norm $\sqrt{z_0^2 \quad 1}$.

The solution of CERT lies on the cone $\langle \boldsymbol{x}, \boldsymbol{z} \rangle = b$. We decompose $\boldsymbol{w}$ along $\boldsymbol{x}$ and its orthogonal complement $\boldsymbol{x}^\perp$, i.e.

$$\boldsymbol{w} = \xi \boldsymbol{x} + \zeta \boldsymbol{x}^\perp \ .$$

with $\zeta \quad 0$ and $\|\boldsymbol{x}^\perp\| = 1$. Without loss of generality, such a decomposition always exists. Note that

$$\langle \boldsymbol{w}, \boldsymbol{z}^* \rangle = \xi \langle \boldsymbol{x}, \boldsymbol{z}^* \rangle + \zeta \langle \boldsymbol{x}^\perp, \boldsymbol{z}^* \rangle = \xi b + \zeta \langle \boldsymbol{x}^\perp, \boldsymbol{z}^* \rangle \ ,$$

where the second equality follows from $\langle \boldsymbol{x}, \boldsymbol{z}^* \rangle = b$. This implies that for the objective $\langle \boldsymbol{w}, \boldsymbol{z} \rangle$ to be maximized, $\boldsymbol{z}^*$ has to have all of its remaining mass along $\boldsymbol{x}^\perp$, i.e.,

$$\boldsymbol{z}^* = b\boldsymbol{x} + \sqrt{1 \quad b^2}\boldsymbol{x}^\perp.$$

After rescaling to satisfy the original norm constraint in CERT, the maximizing adversarial example (for a given z_0) is given as

$$\boldsymbol{x} = \left(z_0, \sqrt{z_0^2 \quad 1} \quad \boldsymbol{z}^*\right) = \left(z_0, \sqrt{z_0^2 \quad 1}\left(b\boldsymbol{x} + \sqrt{1 \quad b^2}\boldsymbol{x}^\perp\right)\right) .$$

□

C.3 Adversarial Perceptron

For the convergence analysis of the gradient-based update, we first need to analyze the convergence of the adversarial perceptron. We first state the following lemma that relates the adversarial margin to the max-margin classifier.

Lemma C.4. *Let $\boldsymbol{w}$ be the max-margin classifier of S with margin γ_H. At each iteration of Algorithm 2, $\boldsymbol{w}$ linearly separates $S \cup S'$ with margin at least $\frac{\gamma_H}{\cosh(\alpha)}$.*

Remark C.5. Note that this "adversarial Perceptron" corresponds to a gradient update of the form $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + y\boldsymbol{x}$, which resembles the adversarial SGD.

Proof. The proof reduces the problem to Euclidean geometry in the Poincare half plane. We defer the proof until Section E, since the respective geometric tools are introduced only in Section D.2. □

With this result, we can show the following bound on the sample complexity of the adversarial perceptron:

Theorem C.6. *Assume that there is some $\boldsymbol{w} \in \mathbb{R}^{d+1}$ with $\sqrt{\boldsymbol{w} \circ \boldsymbol{w}} = 1$ and $\boldsymbol{w}_0 \circ \boldsymbol{w} \geq 0$, and some $\gamma_H > 0$, such that $y_j(\boldsymbol{w} \circ \boldsymbol{x}_j) \geq \sinh(\gamma_H)$ for $j = 1, \ldots, |S|$. Then, adversarial perceptron (with adversarial budget α) converges after $O\left(\frac{\cosh(\alpha)}{\sinh(\gamma_H)}\right)$ steps, at which it has margin of at least $\frac{\gamma_H}{\cosh(\alpha)}$.*

Proof. Without loss of generality, we initialize the classifier as $w_0 = (0, 1, 0, \ldots, 0)$. Furthermore, assume that the tth error is made at the jth sample. For the ease of exposition, we assume that the normalization step is not performed at this update. (The case with normalization after the update can be handled as in the Proof of Theorem B.1.) Thus,

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + y_j \boldsymbol{x}_i \ ,$$

which implies that

$$(\boldsymbol{w}_{t+1} \circ \boldsymbol{w}_t) \circ \boldsymbol{w} = (y_j \boldsymbol{x}_j) \circ \boldsymbol{w} = y_j (\boldsymbol{x}_j \circ \boldsymbol{w}) \geq \frac{\gamma'_H}{\cosh(\alpha)} \ ,$$

where $\gamma'_H = \sinh(\gamma_H)$ and the last inequality follows from Lemma C.4. By summing and telescoping, we obtain that

$$\sum_{k=0}^{t} (\boldsymbol{w}_{k+1} \circ \boldsymbol{w}_k) \circ \boldsymbol{w} \geq \sum_{k=0}^{t} \frac{\gamma'_H}{\cosh(\alpha)}$$

$$) \quad (\boldsymbol{w}_{t+1} \circ \boldsymbol{w}_0) \circ \boldsymbol{w} \geq \frac{t\gamma'_H}{\cosh(\alpha)} \ .$$

Now, by multiplying both sides by -1 and rewriting the Minkowski product gives us that

$$\begin{aligned} \boldsymbol{w}_{t+1} \circ \boldsymbol{w} - \boldsymbol{w}_0 \circ \boldsymbol{w} &\leq \frac{t\gamma'_H}{\cosh(\alpha)} \\ \underbrace{|\boldsymbol{w}_0|}_{=1} \underbrace{|\boldsymbol{w}|}_{=1} \underbrace{\cosh(\rho)(\boldsymbol{w}_0, \boldsymbol{w})}_{\leq \cosh(\pi) =: C} &\leq \frac{t\gamma'_H}{\cosh(\alpha)} \\ C &\geq \frac{t\gamma'_H}{\cosh(\alpha)} \ . \end{aligned} \tag{C.13}$$

Now, note that

$$-1 \leq \cosh(\rho)(\boldsymbol{w}_{t+1}, \boldsymbol{w})) \leq \frac{\boldsymbol{w}_{t+1}}{\underbrace{|\boldsymbol{w}_{t+1}|}_{\geq 1}} \frac{\boldsymbol{w}}{\underbrace{|\boldsymbol{w}|}_{=1}} \overset{(i)}{\leq} C \leq \frac{t\gamma'_H}{\cosh(\alpha)} \ ,$$

where (i) utilizes (C.13). Now, solving for t gives us that

$$t \geq (C - 1) \frac{\cosh(\alpha)}{\gamma'_H}.$$

Further, it follows from (2.2) that an adversarial hyperbolic margin of $\frac{\gamma_H}{\cosh(\alpha)}$ is then achieved after $O\left(\frac{\cosh(\alpha)}{\sinh(\gamma_H)}\right)$ steps. □

C.4 Gradient-based update

Recall that, our objective in Algorithm 2 consists of an inner optimization (that computes the adversarial example) and an outer optimization (that updates the classifier). In particular, we consider

$$\min_{w \in \mathsf{R}^{d+1}} L_{\mathsf{rob}}(\boldsymbol{w}; S) := \frac{1}{jSj} \sum_{(x,y)\in\mathcal{S}} l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) \ ,$$

where the robust loss is given by

$$l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) := \max_{z \in \mathsf{L}^d, d_{\mathsf{L}}(x,z)\le\alpha} l(\boldsymbol{x}, y; \boldsymbol{w}) \ = \ l(\boldsymbol{x}, y; \boldsymbol{w}) \ ,$$

where $\boldsymbol{x} \ 2 \ \mathsf{argmax}_{z\in\mathsf{L}^d, d_{\mathsf{L}}(x,z)\le\alpha} \, l(\boldsymbol{x}, y; \boldsymbol{w})$.

Recall that, to compute the update, we need to compute gradients of the outer minimization problem, i.e., $r_{\ w} \ l_{\mathsf{rob}}$ over S. However, the function l_{rob} is itself a maximization problem (referred to as the inner maximization problem above). Therefore, we compute the gradient at the maximizer of the inner problem. Danskin's theorem ensures that this gives a valid decent direction. For the sake of completeness, we recall the Danskin's theorem here.

Theorem C.7 (Danskin [9], Bertsekas [1])**.** *Suppose X is a non-empty compact topological space and $g : \mathsf{R}^d \quad X \ ! \ \mathsf{R}$ is a continuous function such that $g(\ , \delta)$ is differentiable for every $\delta \ 2 \ X$. Let $\delta^*_{\boldsymbol{w}} \ = \ \mathsf{argmax}_{\delta\in X} \, g(\boldsymbol{w}, \delta)$. Then, the function $\psi(\boldsymbol{w}) \ = \ \mathsf{max}_{\delta\in X} \, g(\boldsymbol{w}, \delta)$ is subdifferentiable and the subdifferential is given by*

$$\partial\psi(\boldsymbol{w}) \ = \ \mathsf{conv} \ (f \, r_{\ w} \ g(\boldsymbol{w}, \delta) j \ \delta \ 2 \ \delta^*_{\boldsymbol{w}} g) \ .$$

This approach has been previously used in Madry et al. [20] and Charles et al. [5]. Note that when we find an adversarial example in Algorithm 2, we can write it in a closed form (cf. Theorem C.3). In particular,

$$l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) \ = \ \max_{d_{\mathsf{L}}(x,z)\le\alpha} l(\boldsymbol{z}, y; \boldsymbol{w}) \ = \ l(\boldsymbol{x}, y; \boldsymbol{w}) \quad \text{with } \boldsymbol{x} \ = \ \left(x_0, \sqrt{x_0^2 \quad 1} \left(b \, \boldsymbol{x} \ + \ \sqrt{1 \quad b^2} \boldsymbol{x}^{\perp} \right) \right) \ .$$

Note that

$$r_{\ w} \ l(\boldsymbol{x}, y; \boldsymbol{w}) \ = \ f'(y(\boldsymbol{w} \quad \boldsymbol{x})) \quad r_{\ w} \ y(\boldsymbol{w} \quad \boldsymbol{x}) \ = \ f'(y(\boldsymbol{w} \quad \boldsymbol{x})) \quad y(\widehat{\boldsymbol{x}})^T \ ,$$

where we have used the fact that $r_{\ w} \ y(\boldsymbol{w} \quad \boldsymbol{x}) = y(\widehat{\boldsymbol{x}})^T \ = y(\boldsymbol{x}_0, \quad \boldsymbol{x}_1, \ldots, \quad \boldsymbol{x}_n)^T$. From Danskin's theorem, we have $r_{\ w} l(\boldsymbol{x}, y; \boldsymbol{w}) \ 2 \ \partial \ l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w})$. This enables us to compute the decent direction and perform the update step with

$$r \ L(\boldsymbol{w}; S') \ = \ \frac{1}{jS'j} \sum_{(\boldsymbol{x},y)\in\mathcal{S}'} r \ l(\boldsymbol{x}, y; \boldsymbol{w}) \ 2 \ \partial L_{\mathsf{rob}}(\boldsymbol{w}; S) \ ,$$

Furthermore, we have

$$r^2_{\ w} \ l(\boldsymbol{x}, y; \boldsymbol{w}) \ = \ f''(y(\boldsymbol{w} \quad \boldsymbol{x}))\boldsymbol{x}\boldsymbol{x}^T \ 2 \ \partial^2 l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) \ , \tag{C.14}$$

which enable the computation of the Hessian of $L(\boldsymbol{w}; S')$.

The convergence results in this section build on hyperbolic analogues of comparable Euclidean results in [30, 14].

We first show a bound on the Hessian of the loss:

Lemma C.8.

$$r^2 L(\boldsymbol{w}_t; S'_t) \quad \beta\sigma^2_{\mathsf{max}} \quad I \ ,$$

where σ_{max} is an upper bound on the maximum singular value of the data matrix $\frac{1}{|\mathcal{S}'_t|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}'_t} \boldsymbol{x}\boldsymbol{x}^T$.

Proof.

$$r^2 L(\boldsymbol{w}_t; S'_t) \ = \ \frac{1}{jS'_tj} \sum_{(\boldsymbol{x},y)\in\mathcal{S}'_t} r^2 l(\boldsymbol{x}, y; \boldsymbol{w}_t) \overset{(i)}{=} \frac{1}{jS'_tj} \sum_{(\boldsymbol{x},y)\in\mathcal{S}'_t} f''(y(\boldsymbol{x} \quad \boldsymbol{w}_t))\boldsymbol{x}\boldsymbol{x}^T$$

$$\overset{(ii)}{\beta} \ \frac{1}{jS'_tj} \sum_{(\boldsymbol{x},y)\in\mathcal{S}'_t} \boldsymbol{x}\boldsymbol{x}^T \quad \beta\sigma^2_{\mathsf{max}} \quad I \ ,$$

where (i) and (ii) follow from (C.14) and the assumption that f is β-smooth. □

With the help of Lemma C.8, we can show the following result (a restatement of Theorem 4.4), which establishes that the gradient updates are guaranteed to converge to a large-margin classifier:

Theorem C.9 (Theorem 4.3)**.** *Let $\{\boldsymbol{w}_t\}$ be the GD iterates*

$$\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t \quad \frac{\eta}{|S'_t|} \sum_{(\boldsymbol{x},y)\in S'_t} \nabla l(\boldsymbol{x}, y; \boldsymbol{w})$$

$$\boldsymbol{w}_{t+1} \quad \frac{\boldsymbol{w}_{t+1}}{\sqrt{\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_{t+1}}}$$

with constant step size $\eta < \frac{2}{\beta\sigma_{\max}^2}$ and an initialization $\boldsymbol{w}_0$ with $\boldsymbol{w}_0 \quad \boldsymbol{w}_0 < 0$. Then, we have $\lim_{t\to\infty} L(\boldsymbol{w}_t; S \cup S'_t) = 0$.

Proof. By Assumption 1.1 we can find a $\boldsymbol{w}$ that linearly separates S. Then, we have

$$\begin{aligned}\langle \boldsymbol{w}, \nabla L(\boldsymbol{w}; S'_t)\rangle &= \langle \boldsymbol{w}, \frac{1}{|S'_t|} \sum_{(\boldsymbol{x},y)\in S'_t} f'(y(\boldsymbol{x} \quad \boldsymbol{w}_t))\, y\widehat{\boldsymbol{x}}\rangle \\ &= \underbrace{\left(\frac{1}{|S'_t|} \sum_{(x,y)\in S'_t} f'(y(\boldsymbol{x} \quad \boldsymbol{w}_t))\right)}_{<0} \underbrace{y\langle \boldsymbol{w}, \widehat{\boldsymbol{x}}\rangle}_{=y(w*x)<0} ,\end{aligned}$$

where the negativity of the first term follows from the assumptions on f (cf. Assumption 1.3) and the upper bound on the second term from the separability assumption. This implies that $\langle \boldsymbol{w}, \nabla L(\boldsymbol{w}; S'_t)\rangle \neq 0$ for any finite $\boldsymbol{w}$. Therefore, there are no finite critical points $\boldsymbol{w}$ for which $\nabla L(\boldsymbol{w}; S'_t) = 0$. However, GD is guaranteed to converge to a critical point for smooth objectives with an appropriate step size. Therefore, $\|\boldsymbol{w}_t\| \to 1$ and $y(\boldsymbol{w}_t \quad x) > 0\ \forall\ (\boldsymbol{x}, y) \in S \cup S'_t$ and large enough t. Then, we have $l(\boldsymbol{x}, y; \boldsymbol{w}_t) \to 0$, for all $(\boldsymbol{x}, y) \in S \cup S'_t$. This further implies that $L(\boldsymbol{w}_t; S \cup S'_t) = \frac{1}{|S\cup S'_t|} \sum_{(x,y)\in S\cup S'_t} l(\boldsymbol{x}, y; \boldsymbol{w}_t) \to 0$. □

We further show that the enrichment of the training set with adversarial examples is critical for polynomial-time convergence: Without adversarial training, we can construct a simple max-margin problem, that cannot be solved in polynomial time.

Theorem C.10 (Theorem 3.3)**.** *Consider $S = \{(\boldsymbol{e}_1, 1), (\quad \boldsymbol{e}_1, \quad 1)\} \quad \mathbb{R}^{d+1} \quad \{+1, \quad 1\}$ and a typical initialization $\boldsymbol{w}_0 = \boldsymbol{e}_2 \in \mathbb{R}^{d+1}$ (with the standard basis vectors $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{R}^{d+1}$). Let $\{\boldsymbol{w}_t\}_t$ is a sequence of classifiers generated by the GD updates (with fixed step size η)*

$$\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t \quad \frac{\eta}{|S|} \sum_{(x,y)\in S} \nabla l(\boldsymbol{x}, y; \boldsymbol{w})$$

$$\boldsymbol{w}_{t+1} \quad \frac{\boldsymbol{w}_{t+1}}{\sqrt{\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_{t+1}}} .$$

Then, the number of iterations needed to achieve margin γ_H is $\quad (\exp(\gamma_H))$.

Proof. First, note that the initialization $\boldsymbol{w}_0$ is valid as $\boldsymbol{w}_0 \quad \boldsymbol{w}_0 = \quad 1 < 0$. The gradient of the loss can be computed as

$$\nabla l(\boldsymbol{x}_i, y_i; \boldsymbol{w}_t) = f'(y_i(\boldsymbol{x}_i \quad \boldsymbol{w}_t)) y_i \widehat{\boldsymbol{x}}_i$$

where

$$f'(s) = \quad \frac{\exp\left(\quad \operatorname{asinh}\left(\frac{s}{2R}\right)\right)}{R\sqrt{\frac{s^2}{4R^2} + 1}\left(\exp\left(\quad \operatorname{asinh}\left(\frac{s}{2R}\right)\right) + 1\right)}$$

is the derivative of the hyperbolic logistic regression loss (cf. (4.5)). Note that due to the structure of S and $\boldsymbol{w}_0$, the GD update will produce the following iteration sequence

$$\begin{aligned} a_{t+1} &= a_t \quad f'(a_t) \\ \boldsymbol{w}_t &= (a_t, \sqrt{a_t^2 + 1}, 0, \ldots, 0) ,\end{aligned}$$

where the first coordinate is determined through the GD update and the second through normalization to ensure the validaty of the classifier, i.e., $\boldsymbol{w}_t \quad \boldsymbol{w}_t < 0$. In order to see this, note that $\boldsymbol{w}_t \quad \boldsymbol{w}_t = a_t^2 \quad (\sqrt{a_t^2 + 1})^2 = \quad 1 < 0$. We now want to show that

$$a_t \quad \sinh(\ln(t+1)) \; .$$

For the induction, note that $a_0 = 0 = \ln(1) = \sinh(\ln(1))$. Assume, that $a_t \quad \sinh(\ln(t+1))$. We want to show

$$a_{t+1} \quad \sinh(\ln(t+2)) \; .$$

Note, that

$$a_{t+1} = \underbrace{a_t}_{①} + \underbrace{\frac{\exp\left(\quad \operatorname{asinh}\left(\frac{a_t}{2R}\right)\right)}{R\sqrt{\frac{a_t^2}{4R^2}+1}\left(\exp\left(\quad \operatorname{asinh}\left(\frac{a_t}{2R}\right)\right)+1\right)}}_{②} \; .$$

Since $\exp\left(\quad \operatorname{asinh}\left(\frac{a_t}{2R}\right)\right) \quad \exp\left(\quad \operatorname{asinh}\left(\frac{a_t}{2R}\right)\right) + 1$ and $R\sqrt{\frac{a_t^2}{4R^2}+1} \quad 1$, clearly 2 is bounded by 1. Inserting this above and replacing 1 with the induction assumption, we have

$$a_{t+1} \quad \sinh(\ln(t+1)) + 1 \; .$$

Note that, by definition, $\sinh(z) = \frac{1}{2}(e^x \quad e^{-x})$. Thus,

$$\sinh(\ln(t+1)) = \frac{1}{2}(t+1 \quad (\quad (t+1))) = t+1 \; ,$$

which further implies that

$$a_{t+1} = \sinh(\ln(t+1)) + 1 \quad t+2 = \sinh(\ln(t+2)) \; . \tag{C.15}$$

This finishes the induction proof. Assuming a margin of at least γ_H, we have

$$\gamma_H \quad \text{margin}_{\mathcal{S}}(\boldsymbol{w}_t) = \operatorname{asinh}\left(\frac{y(\boldsymbol{x} \quad \boldsymbol{w}_t)}{\cancel{p}\sqrt{\boldsymbol{w}_t \quad \boldsymbol{w}_t}}\right) \overset{(i)}{=} \operatorname{asinh}(a_{t+1}) \overset{(ii)}{} \operatorname{asinh}(\sinh(\ln(t+2))) \quad \ln(t+2) \; ,$$

where (i) follows $\boldsymbol{w}_t \quad \boldsymbol{w}_t = \quad 1$ after normalization and (ii) from the upper bound in (C.15). Now, by solving for t, we obtain that $t = \quad (\exp(\gamma_H))$. □

Next, we quantify the convergence rate of adversarial training with GD updates (cf. (4.4)). We start by presenting some auxiliary results.

Lemma C.11 (Smoothness bound)**.** *Let $\eta_t =: \eta < \frac{2\sinh^2(\gamma_H)}{\beta\sigma_{\max}^2 \cosh^2(\alpha) R_\alpha^2}$ be the fixed step size and $\boldsymbol{w}_0$ a valid initialization, i.e. $\boldsymbol{w}_0 \quad \boldsymbol{w}_0 < 0$. Then, for the GD update (with fixed step size $\eta_t =: \eta$)*

$$\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t \quad \eta_t \underbrace{r\, L(\boldsymbol{w}_t; S_t')}_{\in \partial L_{\text{rob}}(w_t; \mathcal{S}_t)}$$

$$\boldsymbol{w}_{t+1} \quad \cancel{p}\frac{\boldsymbol{w}_{t+1}}{\sqrt{\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_{t+1}}} \; .$$

we have

1. $L_{\text{rob}}(\boldsymbol{w}_{t+1}; S) \quad L_{\text{rob}}(\boldsymbol{w}_t; S) \quad \eta\left(\frac{\sinh(\gamma_H)^2}{\cosh^2(\alpha) R_\alpha^2} \quad \frac{\beta\sigma_{\max}^2 \eta}{2}\right) k \underbrace{r\, L(\boldsymbol{w}_t; S_t')}_{\in \partial L_{\text{rob}}(w_t; \mathcal{S}_t)} k^2$;
2. $\sum_{k=0}^{\infty} k\, r\, L(\boldsymbol{w}_k; S_k')k^2 < 1$ *; as a result,* $\lim_{t\to\infty} k\, r\, L(\boldsymbol{w}_t); S_t' k^2 = 0$.

Proof. In Algorithm 2 with gradient update rule, we have

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t \quad \eta\, r\, L(\boldsymbol{w}_t; S_t') \\ &= \boldsymbol{w}_t \quad \frac{\eta}{j S_t' j} \sum_{(\boldsymbol{x},y)\in\mathcal{S}_t^0} l(\boldsymbol{x}, y; \boldsymbol{w}_t) \\ &= \boldsymbol{w}_t \quad \frac{\eta}{j S_t' j} \sum_{(\boldsymbol{x},y)\in\mathcal{S}_t^0} f'(y(\boldsymbol{x} \quad \boldsymbol{w}_t)) y \widehat{\boldsymbol{x}} \; . \end{aligned}$$

Now, consider the inner product $\langle \boldsymbol{w}_{t+1}, \boldsymbol{w}\rangle$, where $\boldsymbol{w}$ is the optimal classifier. With out loss of generality, we assume $\|\boldsymbol{w}\| = 1$.

$$\begin{aligned}\langle \boldsymbol{w}_{t+1}, \boldsymbol{w}\rangle &= \langle \boldsymbol{w}_t, \boldsymbol{w}\rangle - \frac{\eta}{|S'|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0} f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))y\langle \widehat{\boldsymbol{x}}, \boldsymbol{w}\rangle \\ &\overset{(i)}{=} \langle \boldsymbol{w}_t, \boldsymbol{w}\rangle - \frac{\eta}{|S'_t|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))y(\boldsymbol{x}^\top \boldsymbol{w}) \\ &\overset{(ii)}{\geq} \langle \boldsymbol{w}_t, \boldsymbol{w}\rangle - \frac{\eta\gamma'_H}{|S'_t|\cosh(\alpha)}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))\ ,\end{aligned}$$

where (i) and (ii) follow from $\langle \widehat{\boldsymbol{x}}, \boldsymbol{w}\rangle = \boldsymbol{x}^\top \boldsymbol{w}$ and $y(\boldsymbol{x}^\top \boldsymbol{w}) \geq \frac{\gamma^0_H}{\cosh(\alpha)}$ (cf. Lemma C.4), respectively. We use the shorthand $\gamma'_H = \sinh(\gamma_H)$. With the linearity of the inner product, we get

$$\langle \boldsymbol{w}_{t+1} - \boldsymbol{w}_t, \boldsymbol{w}\rangle \geq \frac{\eta\gamma'_H}{|S'_t|\cosh(\alpha)}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))\ .$$

Since f' is negative (cf. Assumption 1.3), we can replace $f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))$ with $|f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))|$ to get

$$\begin{aligned}\langle \boldsymbol{w}_t - \boldsymbol{w}_{t+1}, \boldsymbol{w}\rangle &\geq \frac{\eta\gamma'_H}{|S'_t|\cosh(\alpha)}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} |f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))| \\ &\overset{(i)}{=} \frac{\eta\gamma'_H}{\cosh(\alpha)R_\alpha}\|\nabla L(\boldsymbol{w}_t; S'_t)\|\ ,\end{aligned} \tag{C.16}$$

where (i) holds as follows: Recall, that $\|\nabla l(\boldsymbol{x}, y; \boldsymbol{w}_t)\| = |f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))|\|\widehat{\boldsymbol{x}}\|$. Thus,

$$\begin{aligned}\|\nabla L(\boldsymbol{w}_t; S'_t)\| = \Big\|\frac{1}{|S'_t|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} \nabla l(\boldsymbol{x}, y; \boldsymbol{w}_t)\Big\| &\leq \frac{1}{|S'_t|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} \|\nabla l(\boldsymbol{x}, y; \boldsymbol{w}_t)\| \\ &\leq \frac{1}{|S'_t|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} |f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))|\|\widehat{\boldsymbol{x}}\| \leq \frac{R_\alpha}{|S'_t|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} |f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))|\ .\end{aligned}$$

This implies that

$$\frac{1}{|S'_t|}\sum_{(\boldsymbol{x},y)\in\mathcal{S}^0_t} |f'(y(\boldsymbol{x}^\top \boldsymbol{w}_t))| = \frac{1}{R_\alpha}\|\nabla L(\boldsymbol{w}_t; S'_t)\|\ .$$

Applying Cauchy-Schwarz to the left hand side of (C.16) gives us that

$$\|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|\|\boldsymbol{w}\| \geq \langle \boldsymbol{w}_t - \boldsymbol{w}_{t+1}, \boldsymbol{w}\rangle \geq \frac{\eta\gamma'_H}{\cosh(\alpha)R_\alpha}\|\nabla L(\boldsymbol{w}_t; S'_t)\|\ . \tag{C.17}$$

Now, using the fact that $\|\boldsymbol{w}\| = 1$ in (C.17), we get

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \leq \frac{\eta\gamma'_H}{\cosh(\alpha)R_\alpha}\|\nabla L(\boldsymbol{w}_t; S'_t)\|\ . \tag{C.18}$$

Now, consider the following Taylor approximation:

$$\begin{aligned}L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S) = L_{\mathsf{rob}}(\boldsymbol{w}_t; S) + \langle \underbrace{\nabla L(\boldsymbol{w}_t; S'_t)}_{\in\partial L_{\mathsf{rob}}(\boldsymbol{w}_t;\mathcal{S})}, \boldsymbol{w}_{t+1} - \boldsymbol{w}_t\rangle + \\ (\boldsymbol{w}_{t+1} - \boldsymbol{w}_t)^T \underbrace{\nabla^2 L(\boldsymbol{v}; S'_t)}_{\in\partial^2 L_{\mathsf{rob}}(\boldsymbol{v};\mathcal{S})}(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t)/2,\end{aligned} \tag{C.19}$$

where $\boldsymbol{v} \in \mathsf{conv}(\boldsymbol{w}_{t+1}, \boldsymbol{w}_t)$. By utilizing Lemma C.8 in (C.19), we get that

$$L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S) \leq L_{\mathsf{rob}}(\boldsymbol{w}_t; S) + \langle \nabla L(\boldsymbol{w}_t; S'_t), \boldsymbol{w}_{t+1} - \boldsymbol{w}_t\rangle + \frac{\beta\sigma^2_{\mathsf{max}}}{2}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2\ . \tag{C.20}$$

Recall the update rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t \quad \eta\, r\, L(\boldsymbol{w}_t; S'_t) \tag{C.21}$$

$$) \ \boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t = \quad \eta\, r\, L(\boldsymbol{w}_t; S'_t)\ . \tag{C.22}$$

Inserting this in (C.20), we get

$$L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S) = L_{\mathsf{rob}}(\boldsymbol{w}_t; S) + h \quad \eta^{-1}(\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t),\, \boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t i + \frac{\beta\sigma_{\max}^2}{2} k\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t k^2$$

$$= L_{\mathsf{rob}}(\boldsymbol{w}_t; S) \quad \eta^{-1} k\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t k^2 + \frac{\beta\sigma_{\max}^2}{2} k\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t k^2\ . \tag{C.23}$$

By combining (C.18) and (C.23), we obtain that

$$L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}_t; S) \quad \frac{\eta\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} k\, r\, L(\boldsymbol{w}_t; S'_t) k^2 + \frac{\beta\sigma_{\max}^2}{2} k\boldsymbol{w}_{t+1} \quad \boldsymbol{w}_t k^2\ . \tag{C.24}$$

Again, utilizing (C.21), it follows from (C.24) that

$$L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}_t; S) \quad \frac{\eta\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} k\, r\, L(\boldsymbol{w}_t; S'_t) k^2 + \frac{\beta\sigma_{\max}^2 \eta^2}{2} k\, r\, L(\boldsymbol{w}_t; S'_t) k^2 \tag{C.25}$$

$$= L_{\mathsf{rob}}(\boldsymbol{w}_t; S) \quad \eta \left(\frac{\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} \quad \frac{\beta\sigma_{\max}^2 \eta}{2} \right) k\, r\, L(\boldsymbol{w}_t; S'_t) k^2\ . \tag{C.26}$$

This establishes the first claim of Lemma C.11. Now, we can rewrite (C.25) to obtain the following.

$$\frac{L_{\mathsf{rob}}(\boldsymbol{w}_t; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S)}{\eta \left(\frac{\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} \quad \frac{\beta\sigma_{\max}^2 \eta}{2} \right)} \quad k\, r\, L(\boldsymbol{w}_t; S'_t) k^2\ .$$

Note that our assumption on the step size η ensures that the denominator in (C.25) is $\neq 0$.

Next, summing and telescoping gives us that

$$\sum_{k=0}^{t} k\, r\, L(\boldsymbol{w}_k; S'_k) k^2 \quad \sum_{k=0}^{t} \frac{L_{\mathsf{rob}}(\boldsymbol{w}_k; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}_{k+1}; S)}{\eta \left(\frac{\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} \quad \frac{\beta\sigma_{\max}^2 \eta}{2} \right)} = \frac{L_{\mathsf{rob}}(\boldsymbol{w}_0; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S)}{\eta \left(\frac{\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} \quad \frac{\beta\sigma_{\max}^2 \eta}{2} \right)}\ ,$$

where the right term is bounded, since $L_{\mathsf{rob}}(\boldsymbol{w}_0; S) < 1$ and $0 \quad L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S)$. This establishes the second claim of Lemma C.11 as

$$\sum_{k=0}^{\infty} k\, r\, L(\boldsymbol{w}_k; S'_k) k^2 < 1 \quad) \quad \lim_{t\to\infty} k\, r\, L(\boldsymbol{w}_t; S'_t) k^2 = 0\ .$$

□

Lemma C.12. *With the assumptions of Lemma C.11, Lemma C.11.1 implies for all* $\boldsymbol{w} \in \mathsf{R}^{d+1}$

$$2 \sum_{k=0}^{t-1} \eta_k \big(L_{\mathsf{rob}}(\boldsymbol{w}_k; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}; S) \big) +$$

$$\sum_{k=0}^{t-1} \frac{\eta_k^2}{\eta_k} \big(L_{\mathsf{rob}}(\boldsymbol{w}_{k+1}; S) \quad L_{\mathsf{rob}}(\boldsymbol{w}_k; S) \big) \quad k\boldsymbol{w}_0 \quad \boldsymbol{w} k^2 \quad k\boldsymbol{w}_t \quad \boldsymbol{w} k^2\ ,$$

where $\eta_k = \eta_k \left(\frac{\gamma_H'^2}{\cosh(\alpha)^2 R_\alpha^2} \quad \frac{\beta\sigma_{\max}^2 \eta_k}{4} \right)$.

Proof. First, note that the GD update

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t \quad \eta_t \underbrace{r\, L(\boldsymbol{w}_t; S'_t)}_{\in \partial L_{\mathsf{rob}}(w_t; \mathcal{S})}$$

implies that

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}\|^2 = \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 - 2\eta_t \langle \nabla L(\boldsymbol{w}_t; S_t'), \boldsymbol{w}_t - \boldsymbol{w} \rangle + \eta_t^2 \|\nabla L(\boldsymbol{w}_t; S_t')\|^2 \tag{C.27}$$

$$= \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 + 2\eta_t \langle \nabla L(\boldsymbol{w}_t; S_t'), \boldsymbol{w} - \boldsymbol{w}_t \rangle + \eta_t^2 \|\nabla L(\boldsymbol{w}_t; S_t')\|^2 . \tag{C.28}$$

Note that the hyperbolic logistic regression loss $f(z)$ in (4.5) is convex for $z < 0$. As a consequence, $l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w})$ is convex for any adversarial example with $\mathsf{sgn}(\boldsymbol{w}_t^\top \boldsymbol{x}) \neq \mathsf{sgn}(\boldsymbol{w}_t^\top \boldsymbol{x})$. This implies that

$$l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}) \geq l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t) + \langle \partial l_{\mathsf{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t), \boldsymbol{w} - \boldsymbol{w}_t \rangle ,$$

for any $\boldsymbol{w} \in \mathsf{R}^{d+1}$ and any pair $(\boldsymbol{x}, y)$ for which an adversarial example exists.

Since the sum of convex function is convex, we further have

$$L_{\mathsf{rob}}(\boldsymbol{w}; S) \geq L_{\mathsf{rob}}(\boldsymbol{w}_t; S) + \langle \underbrace{\nabla L(\boldsymbol{w}_t; S_t')}_{\in \partial L_{\mathsf{rob}}(w_t; \mathcal{S})}, \boldsymbol{w} - \boldsymbol{w}_t \rangle . \tag{C.29}$$

By combining (C.27) and (C.29), we obtain that

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}\|^2 \leq \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 + 2\eta_t \big(L_{\mathsf{rob}}(\boldsymbol{w}; S) - L_{\mathsf{rob}}(\boldsymbol{w}_t; S)\big) + \eta_t^2 \|\nabla L(\boldsymbol{w}_t; S_t')\|^2$$

$$\overset{(i)}{\leq} \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 + 2\eta_t \big(L_{\mathsf{rob}}(\boldsymbol{w}; S) - L_{\mathsf{rob}}(\boldsymbol{w}_t; S)\big) + \frac{\eta_t^2 \big(L_{\mathsf{rob}}(\boldsymbol{w}_t; S) - L_{\mathsf{rob}}(\boldsymbol{w}_{t+1}; S)\big)}{\eta_t},$$

where (i) follows from the first claim in Lemma C.11 and $\eta_t := \eta_t \left(\frac{\gamma_H'^2}{\cosh^2(\alpha) R_\alpha^2} - \frac{\beta \sigma_{\max}^2 \eta_t}{4} \right)$.

Next, summing and telescoping gives us that

$$\sum_{k=0}^{t-1} \|\boldsymbol{w}_{k+1} - \boldsymbol{w}\|^2 \leq \|\boldsymbol{w}_k - \boldsymbol{w}\|^2 - \sum_{k=0}^{t-1} \Big[2\eta_k \big(L_{\mathsf{rob}}(\boldsymbol{w}; S) - L_{\mathsf{rob}}(\boldsymbol{w}_k; S)\big) + \frac{\eta_k^2}{\eta_k} \big(L_{\mathsf{rob}}(\boldsymbol{w}_k; S) - L_{\mathsf{rob}}(\boldsymbol{w}_{k+1}; S)\big) \Big]$$

or

$$\|\boldsymbol{w}_t - \boldsymbol{w}\|^2 \leq \|\boldsymbol{w}_0 - \boldsymbol{w}\|^2 - 2\sum_{k=0}^{t-1} \eta_k \big(L_{\mathsf{rob}}(\boldsymbol{w}; S) - L_{\mathsf{rob}}(\boldsymbol{w}_k; S)\big) + \sum_{k=0}^{t-1} \frac{\eta_k^2}{\eta_k} \big(L_{\mathsf{rob}}(\boldsymbol{w}_k; S) - L_{\mathsf{rob}}(\boldsymbol{w}_{k+1}; S)\big).$$

Now, multiplying both sides by -1 completes the proof as follow.

$$2\sum_{k=0}^{t-1} \eta_k \big(L_{\mathsf{rob}}(\boldsymbol{w}_k; S) - L_{\mathsf{rob}}(\boldsymbol{w}; S)\big) + \sum_{k=0}^{t-1} \frac{\eta_k^2}{\eta_k} \big(L_{\mathsf{rob}}(\boldsymbol{w}_{k+1}; S) - L_{\mathsf{rob}}(\boldsymbol{w}_k; S)\big) \leq \|\boldsymbol{w}_0 - \boldsymbol{w}\|^2 - \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 .$$

□

We are now in a position to present the desired convergence result.

Theorem C.13 (Convergence GD update, Algorithm 2)**.** *For a fixed constant $c \in (0, 1)$, let the step size $\eta_t := \eta = c \frac{2\sinh^2(\gamma_H)}{\beta \sigma_{\max}^2 \cosh^2(\alpha) R_\alpha^2}$ and A be the GD update as defined in* (4.4)*. Then, the iterates $\{\boldsymbol{w}_t\}$ in Algorithm 2 satisfy*

$$L_{\mathsf{rob}}(\boldsymbol{w}_t; S) = O\left(\frac{\sinh^2(\ln(t))}{t} \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)} \right)^{-4} \right) .$$

Proof. Without loss of generality, assume that $\boldsymbol{w}_0 = (0, \boldsymbol{e}_i)$ where $\boldsymbol{e}_i \in \mathbb{R}^d$ is a standard basis vector whose i-th coordinate is 1. Note that this is a valid initialization, since $\boldsymbol{w}_0 \quad \boldsymbol{w}_0 < 0$; furthermore, we have $\|\boldsymbol{w}_0\| = 1$. Let $\boldsymbol{w}^* \in \mathbb{R}^{d+1}$ be a classifier that achieves the margin γ_H on S, i.e., $\delta(\boldsymbol{x}, y) \in S$,

$$y(\boldsymbol{x} \quad \boldsymbol{w}^*) \quad \sinh(\gamma_H) \qquad ( ) \qquad \operatorname{asinh}\left(\frac{y(\boldsymbol{w}^* \quad \boldsymbol{x})}{\sqrt{\boldsymbol{w}^* \quad \boldsymbol{w}^*}} \right) \quad \gamma_H .$$

Without loss of generality, assume that $\|\boldsymbol{w}^*\| = 1$. Let $\boldsymbol{u}_t := \frac{2R_\alpha \sinh(\ln(t)) \cosh(\alpha)}{\sinh(\gamma_H)} \boldsymbol{w}^*$; then $\|\boldsymbol{u}_t\| = \frac{2R_\alpha \sinh(\ln(t)) \cosh(\alpha)}{\sinh(\gamma_H)}$. We have

$$\begin{aligned} L_{\text{rob}}(\boldsymbol{u}_t; S'_t) &= \frac{1}{|S'_t|} \sum_{(x,y) \in \mathcal{S}^0_t} l_{\text{rob}}(\boldsymbol{x}, y; \boldsymbol{u}_t) = \frac{1}{|S'_t|} \sum_{(x,y) \in \mathcal{S}^0_t} f(y(\boldsymbol{x} \quad \boldsymbol{u}_t)) \\ &\overset{(i)}{=} \frac{1}{|S'_t|} \sum_{(x,y) \in \mathcal{S}^0_t} f\big(2R_\alpha \sinh(\ln(t))\big) = f\big(2R_\alpha \sinh(\ln(t))\big) \\ &\overset{(ii)}{=} \ln\left(1 + \exp\left(\quad \ln(t)\right)\right) \overset{(iii)}{\quad} \frac{1}{t} , \end{aligned} \tag{C.30}$$

where (i) follows from

$$y(\boldsymbol{x} \quad \boldsymbol{u}_t) = \frac{2R_\alpha \sinh(\ln(t)) \cosh(\alpha)}{\sinh(\gamma_H)} \underbrace{y(\boldsymbol{x} \quad \boldsymbol{w}^*)}_{\geq \frac{\sinh(\gamma_H)}{\cosh(\alpha)}} \quad 2R_\alpha \sinh(\ln(t)) ,$$

(ii) from $\sqrt{\boldsymbol{u}_t \quad \boldsymbol{u}_t} \quad 1$ and (iii) follows from the fact that $\ln(1 + x) \quad x$.

Now, consider

$$\begin{aligned} &2\eta(t \quad 1)\big(L_{\text{rob}}(\boldsymbol{w}_t; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S)\big) \overset{(i)}{=} 2\sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_t; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S)\big) \\ &= 2\sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_t; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S) + L_{\text{rob}}(\boldsymbol{w}_k; S) \quad L_{\text{rob}}(\boldsymbol{w}_k; S)\big) \\ &= 2\sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_k; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S)) + 2\sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_t; S) \quad L_{\text{rob}}(\boldsymbol{w}_k; S)\big) \\ &\overset{(ii)}{} 2\sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_k; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S)) + \sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_{k+1}; S) \quad L_{\text{rob}}(\boldsymbol{w}_k; S)\big) \\ &\quad 2\sum_{k=0}^{t-1} \eta_k \big(L_{\text{rob}}(\boldsymbol{w}_k; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S)) + \sum_{k=0}^{t-1} \frac{\eta_k^2}{\eta_k} \big(L_{\text{rob}}(\boldsymbol{w}_{k+1}; S) \quad L_{\text{rob}}(\boldsymbol{w}_k; S)\big) \\ &\overset{(iii)}{} \|\boldsymbol{w}_0 \quad \boldsymbol{u}_t\|^2 \quad \|\boldsymbol{w}_t \quad \boldsymbol{u}_t\|^2 , \end{aligned}$$

where (i) holds as we have a constant step-size, i.e., $\eta_k = \eta$ and (ii) follows from the fact that

$$L_{\text{rob}}(\boldsymbol{w}_t; S) \quad L_{\text{rob}}(\boldsymbol{w}_{k+1}; S) \quad \text{for } 0 \quad k \quad t \quad 1.$$

The inequality in (iii) follows from Lemma C.12 with $\boldsymbol{w} = \boldsymbol{u}_t$.

We can rewrite this as

$$\begin{aligned} L_{\text{rob}}(\boldsymbol{w}_t; S) \quad L_{\text{rob}}(\boldsymbol{u}_t; S) &+ \frac{\|\boldsymbol{w}_0 \quad \boldsymbol{u}_t\|^2 \quad \|\boldsymbol{w}_t \quad \boldsymbol{u}_t\|^2}{2\sum_{k=0}^{t-1} \eta_k} \\ &\overset{(i)}{} \frac{1}{t} + \frac{\|\boldsymbol{w}_0 \quad \boldsymbol{u}_t\|^2}{2(t \quad 1)\eta} \\ &\overset{(ii)}{} \frac{1}{t} + \frac{2\|\boldsymbol{w}_0\|^2 + 2\|\boldsymbol{u}_t\|^2}{2(t \quad 1)\eta} = \frac{1}{t} + \frac{\|\boldsymbol{w}_0\|^2 + \|\boldsymbol{u}_t\|^2}{(t \quad 1)\eta} \end{aligned}$$

where (i) follows from (C.30) and (ii) follows from $(a + b)^2 \quad 2a^2 + 2b^2$. Now, using the fact that $\|\boldsymbol{w}_0\| = 1$ and $\|\boldsymbol{u}_t\| = \frac{2R_\alpha \sinh(\ln(t))\cosh(\alpha)}{\sinh(\gamma_H)}$, we obtain that

$$L_{\mathrm{rob}}(\boldsymbol{w}_t; S) \quad \frac{1}{t} + \frac{1 + 4R_\alpha^2 \left(\cosh(\alpha)/\sinh(\gamma_H)\right)^2 \quad \sinh^2(\ln(t))}{(t \quad 1)\eta} . \tag{C.31}$$

By substituting $\eta = c \quad \frac{2\sinh^2(\gamma_H)}{\beta\sigma_{\max}^2 \cosh^2(\alpha) R_\alpha^2}$, we get

$$L_{\mathrm{rob}}(\boldsymbol{w}_t; S) = O\left(\frac{\sinh^2(\ln(t))}{t} \quad \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)^{-4}\right) .$$

□

Theorem C.14 (Iteration complexity)**.** *Consider Algorithm 2 with* $\eta_t := \eta = c \quad \frac{2\sinh^2(\gamma_H)}{\beta\sigma_{\max}^2 \cosh^2(\alpha) R_\alpha^2}$ *and* A *being the GD update. Then Algorithm 2 converges as* $\left(\mathrm{poly}\left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)\right)$.

Proof. Let $\varrho = \frac{\ln(1+1/e)}{\ln(1+e)}$. We first argue that

$$L_{\mathrm{rob}}(\boldsymbol{w}_t; S) \quad \varrho \quad \ln\left(1 + \exp\left(\quad \frac{\gamma_H}{\cosh(\alpha)}\right)\right) \tag{C.32}$$

implies that $\boldsymbol{w}_t$ achieves margin $\gamma_H/\cosh(\alpha)$ on S. To see this, note that

$$\begin{aligned} L_{\mathrm{rob}}(\boldsymbol{w}_t; S) &= \frac{1}{|S|} \sum_{(x,y)\in\mathcal{S}} l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t) \\ &= \underbrace{\max_{(x,y)\in\mathcal{S}} l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t)}_{:= l_{\mathrm{rob}}^{\max}(\mathcal{S})} \frac{1}{|S|} \sum_{(x,y)\in\mathcal{S}} \frac{l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t)}{l_{\mathrm{rob}}^{\max}(S)} \\ & \quad l_{\mathrm{rob}}^{\max} \quad \frac{1}{|S|} \sum_{(x,y)\in\mathcal{S}} \varrho = \varrho \quad l_{\mathrm{rob}}^{\max}. \end{aligned} \tag{C.33}$$

The last inequality in (C.33) holds as, for each $(\boldsymbol{x}, y) \in S$, we have

$$\ln(1 + 1/e) \overset{(i)}{} \underbrace{\ln\left(1 + \exp\left(\quad \mathrm{asinh}\left(\frac{y(\boldsymbol{x} \quad \boldsymbol{w}_t)}{2R_\alpha}\right)\right)\right)}_{= l_{\mathrm{rob}}(x,y;w_t)} \quad \max_{(x,y)\in\mathcal{S}} l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t) \overset{(ii)}{} \ln(1 + e),$$

where (i) and (ii) follows from Assumption 1.2. Thus, for each $(\boldsymbol{x}, y) \in S$, we have

$$\frac{l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t)}{l_{\mathrm{rob}}^{\max}} \quad \frac{\ln(1 + 1/e)}{\ln(1 + e)} = \varrho. \tag{C.34}$$

Now, by combining (C.32) and (C.33), we obtain that

$$l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t) \quad \ln\left(1 + \exp\left(\quad \frac{\gamma_H}{\cosh(\alpha)}\right)\right)$$

for any $(\boldsymbol{x}, y) \in S$. Equivalently, for each $(\boldsymbol{x}, y) \in S$,

$$\begin{aligned} l_{\mathrm{rob}}(\boldsymbol{x}, y; \boldsymbol{w}_t) = \ln\left(1 + \exp\left(\quad \mathrm{asinh}\left(\frac{y(\boldsymbol{x} \quad \boldsymbol{w}_t)}{2R_\alpha}\right)\right)\right) \\ \ln\left(1 + \exp\left(\quad \frac{\gamma_H}{\cosh(\alpha)}\right)\right). \end{aligned} \tag{C.35}$$

Thus, for each $(\boldsymbol{x}, y) \in S$, we have

$$\mathrm{asinh}\left(\frac{y(\boldsymbol{x} \quad \boldsymbol{w}_t)}{\|\boldsymbol{w}_t\| \quad \|\boldsymbol{w}_t\|}\right) \overset{(i)}{} \mathrm{asinh}\left(\frac{y(\boldsymbol{x} \quad \boldsymbol{w}_t)}{2R_\alpha}\right) \overset{(ii)}{} \frac{\gamma_H}{\cosh(\alpha)}.$$

where (i) from the definition of R_α (cf. Assumption 1) and (ii) from Eq. C.35. Thus, $\boldsymbol{w}_t$ achieves margin $\gamma_H/\cosh(\alpha)$ on S.

Next, introduce the following constant:

$$C_q := \inf\{t \quad 2 : 2 + \ln(t)^2 \quad (t \quad 1)t^{-1/q}\} .$$

With this, for $t \quad C_q$, we can rewrite the bound in (C.31) as follows:

$$L_{\text{rob}}(\boldsymbol{w}_t; S) \quad \underbrace{\frac{1}{t}}_{\leq \frac{1}{(t \quad 1)\eta}} + \frac{1 + \sinh(\ln(t))^2 \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)^{-2}}{(t \quad 1)\eta} \quad \frac{2 + \sinh(\ln(t))^2 \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)^{-2}}{(t \quad 1)\eta}$$

$$\frac{2 + \ln(t)^2 \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)^{-2}}{(t \quad 1)\eta} \quad \frac{(t \quad 1)t^{-1/q}}{\eta(t \quad 1)} \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)^{-2} \quad \frac{t^{-1/q}}{\eta \left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)}\right)^2} .$$

Solving for t and plugging in the above bound on L_{rob} for which $\boldsymbol{w}_t$ achieves the desire margin, as well as $\eta = c \quad \frac{2\sinh^2(\gamma_H)}{\beta\sigma_{\max}^2 \cosh^2(\alpha) R_\alpha^2}$, we get

$$t = \max\{C_q, \quad \left(\left(\left(\sinh(\gamma_H)^4/\cosh(\alpha)^4\right)\right)^{-q}\right)\} ,$$

from which the claim follows directly. □

C.5 Algorithm 2 with an ERM update

Consider the unit sphere $S^{d-1} \quad \mathbb{R}^d$. A spherical code with minimum separation θ is a subset of S^{d-1}, such that any two distinct elements $\boldsymbol{u}, \boldsymbol{u}'$ in the subset are separated by at least an angle θ, i.e. $\langle \boldsymbol{u}, \boldsymbol{u}' \rangle \quad \cos\theta$. We denote the size of the largest such code as $A(d, \theta)$. A similar construction can be made in hyperbolic space, which allows the transfer of bounds on $A(d, \theta)$ to hyperbolic space [7].

The following lemma shows that a spherical code with a suitable minimum separation θ enables a simple pathological training set such that Algorithm 2 along with an ERM update rule cannot produce a classifier with a desired margin in a small number of iteration. In particular, the lemma shows that the number of iterations required to find the desire margin is lower-bounded by the size of the underlying spherical code.

Lemma C.15. *Consider $S = \{(\boldsymbol{x}_1, y_1) = \big((1, 0, \ldots, 0), 1\big), (\boldsymbol{x}_2, y_2) = \big((\quad 1, 0, \ldots, 0), \quad 1\big)\}$, where $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d$ and y_1, y_2 the corresponding labels. For any $\epsilon < \alpha$, there is an admissible sequence of classifiers $\{\boldsymbol{w}_t\}_{1 \leq t \leq T}$, with*

$$T = A\left(d, \arccos\left(\rho \quad \frac{\sinh(\epsilon)\cosh(\alpha)}{\sqrt{\cosh^2(\alpha) \quad 1}\sqrt{1 + \sinh^2(\epsilon)}}\right)\right)$$

Proof. First, note that $\boldsymbol{x}_1 \quad \boldsymbol{x}_1 = \boldsymbol{x}_2 \quad \boldsymbol{x}_2 = 1$, i.e., $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d$ as desired. Let $\epsilon' = \sinh(\epsilon)$ and $\boldsymbol{e}_i \in \mathbb{R}^{d+1}$ denotes the standard basis vector that has its i-th coordinate equal to 1. Now, consider classifiers of the form

$$\boldsymbol{w}_t = \left(\sqrt{\frac{\epsilon'}{1 + \epsilon'^2}}\, \boldsymbol{v}_t\right) \quad \text{where} \quad \boldsymbol{v}_t \in C\left(d, \arccos\left(\rho \quad \frac{\epsilon'}{\delta}\sqrt{\frac{1 + \delta^2}{1 + \epsilon'^2}}\right)\right) \quad \forall 1 \quad t \quad T , \quad \text{(C.36)}$$

where $\rho < 1$; and $C\left(d, \arccos\left(\rho \quad \frac{\epsilon^0\sqrt{1+\delta^2}}{\delta\sqrt{1+\epsilon^{0 2}}}\right)\right)$ be the spherical code with the minimum separation $\theta = \arccos\left(\rho \quad \frac{\epsilon^0\sqrt{1+\delta^2}}{\delta\sqrt{1+\epsilon^{0 2}}}\right)$ and size $A(d, \theta)$. Since $\boldsymbol{w}_t \quad \boldsymbol{w}_t = (\epsilon')^2 \quad 1 \quad (\epsilon')^2 = \quad 1$, we have $\boldsymbol{w}_t \quad \boldsymbol{w}_t < 0$ for all t. This guarantees that the intersections of the decision boundaries defined by $\{\boldsymbol{w}_t\}_t$ and $\mathbb{R}^d$ are not empty. Moreover, $\{\boldsymbol{w}_t\}$ is an admissible sequence of classifiers with margin ϵ. To see this, note that, for $t = 1, \ldots, T$,

$$\boldsymbol{w}_t \quad \boldsymbol{x}_1 = \epsilon' > 0$$
$$\boldsymbol{w}_t \quad \boldsymbol{x}_2 = \quad \epsilon' < 0 ,$$

i.e., $\{\boldsymbol{w}_t\}$ correctly classifies S. Furthermore, with $\boldsymbol{w}_t \boldsymbol{w}_t = 1$, we have

$$\operatorname{asinh}\left(\frac{y_1(\boldsymbol{w}_t \boldsymbol{x}_1)}{\sqrt{\boldsymbol{w}_t \boldsymbol{w}_t}}\right) = \operatorname{asinh}\left(\frac{y_2(\boldsymbol{w}_t \boldsymbol{x}_2)}{\sqrt{\boldsymbol{w}_t \boldsymbol{w}_t}}\right) = \epsilon ,$$

which gives $\operatorname{margin}_S(\boldsymbol{w}_t) = \epsilon$.

Now we perturb $\boldsymbol{x}_1, \boldsymbol{x}_2$ on $\mathbb{L}^d$ such that the magnitude of the perturbation is at most α, i.e., we want to find $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{L}^d$ such that both $d_{\mathbb{L}}(\boldsymbol{x}_1, \boldsymbol{x}_1)$ and $d_{\mathbb{L}}(\boldsymbol{x}_2, \boldsymbol{x}_2)$ are at most α. For $1 \le t \le T$, consider adversarial examples of the form

$$\boldsymbol{x}_{1t} = \begin{pmatrix} \sqrt{1+\delta^2} \\ \delta \boldsymbol{v}_t \end{pmatrix} \quad \text{and} \quad \boldsymbol{x}_{2t} = \begin{pmatrix} \sqrt{1+\delta^2} \\ \delta \boldsymbol{v}_t \end{pmatrix} .$$

Note that $\boldsymbol{x}_{1t}, \boldsymbol{x}_{2t} \in \mathbb{L}^d$ as $\boldsymbol{x}_{1i} \boldsymbol{x}_{1t} = \boldsymbol{x}_{2t} \boldsymbol{x}_{2i} = 1$. Let us verify the two conditions that we require the valid adversarial examples to satisfy:

Adversarial budget. Note that we have

$$d_{\mathbb{L}}(\boldsymbol{x}_1, \boldsymbol{x}_{1t}) = d_{\mathbb{L}}(\boldsymbol{x}_2, \boldsymbol{x}_{2t}) = \operatorname{acosh}(\sqrt{1+\delta^2}) .$$

Thus, by choosing $\delta = \sqrt{\cosh^2(\alpha) - 1}$, we achieve the maximal permitted perturbation α.

Inconsistent prediction for the current classifier, i.e., $h_{w_t}(\boldsymbol{x}_{1t/2t}) \neq h_w(\boldsymbol{x}_{1/2})$. Note that we have $\delta > \alpha > \epsilon$, which further implies that $\delta > \epsilon > \epsilon'$. In round t,

$$\begin{aligned} \boldsymbol{w}_t \boldsymbol{x}_{1t} &= \epsilon' \sqrt{1+\delta^2} - \delta \sqrt{1+\epsilon'^2} < 0 \\ \boldsymbol{w}_t \boldsymbol{x}_{2t} &= \epsilon' \sqrt{1+\delta^2} + \delta \sqrt{1+\epsilon'^2} > 0 , \end{aligned}$$

which is a consequence of the relation $\delta > \epsilon'$ as follows:

$$\begin{aligned} \delta^2 > \epsilon'^2 \;&\Rightarrow\; \delta^2 + \epsilon'^2\delta^2 > \epsilon'^2 + \epsilon'^2\delta^2 \;\Rightarrow\; \delta^2(1+\epsilon'^2) > \epsilon'^2(1+\delta^2) \\ &\Rightarrow\; \delta\sqrt{1+\epsilon'^2} > \epsilon'\sqrt{1+\delta^2} . \end{aligned}$$

Recall that, in each round of Algorithm 2 with an ERM update, we create adversarial examples and add them to the training set, i.e., after round t we have

$$S_{<t} = S \cup \bigcup_{i=0}^{t-1} \{(\boldsymbol{x}_{1i}, y_{1i}), (\boldsymbol{x}_{2i}, y_{2i})\} .$$

Now for each t and any $i < t$, we have

$$\begin{aligned} \boldsymbol{w}_t \boldsymbol{x}_{1i} &= \epsilon' \sqrt{1+\delta^2} - \delta \sqrt{1+\epsilon'^2} \cos(\theta) > 0 \\ \boldsymbol{w}_t \boldsymbol{x}_{2i} &= \epsilon' \sqrt{1+\delta^2} + \delta \sqrt{1+\epsilon'^2} \cos(\theta) < 0 , \end{aligned}$$

i.e., $\boldsymbol{w}_t$ linearly separates $S_{<t}$.

Therefore, $\{\boldsymbol{w}_t\}$ in (C.36) form an admissible sequence of the classifiers, where $\boldsymbol{w}_t$ linearly separates S_t while achieving the margin of at most ϵ on the original dataset S. The length of the sequence is bounded by the size of the spherical code $C\left(d, \epsilon' \cosh(\alpha)\right)$, which give us that

$$T = A\left(d, \arccos\left(\rho \; \frac{\epsilon'}{\delta}\sqrt{1+\delta^2}\sqrt{1+\epsilon'^2}\right)\right) = A\left(d, \arccos\left(\rho \; \frac{\sinh(\epsilon)\cosh(\alpha)}{\sqrt{\cosh^2(\alpha) - 1}\sqrt{1+\sinh^2(\epsilon)}}\right)\right) .$$

□

The following result (a restatement of Theorem 4.7 from the main text) then follows by applying a lower bound on the maximal size of spherical codes by Shannon.

Theorem C.16 (Theorem 4.7)**.** *Suppose Algorithm 2 (with an ERM update) outputs a linear seperator of $S \cup S'$. In the worst case, the number of iteration required to achieve the margin at least ϵ is* $\exp(d)$.

Proof. The statement of the theorem follows from combining Lemma C.15 with Shannon's lower bound (Theorem A.4) on the maximal size of spherical codes, namely

$$T \geq (1 + o(1)) \sqrt{2\pi d} \frac{\cos(\theta)}{\sin^{d-1}(\theta)} .$$

We introduce the shorthand $\theta =: \arccos\left(\frac{A}{B}\right)$, where $A = \rho \sinh(\epsilon)\cosh(\alpha)$ and $B = \sqrt{\cosh^2(\alpha) - 1}\sqrt{1 + \sinh^2(\epsilon)}$, as given by Lemma C.15. We then use two well-known trigonometric identities

$$\cos(\arccos z) = z \quad \text{and} \quad \sin(\arccos z) = \sqrt{1 - z^2}$$

to simplify the trignometric fraction in Shannon's bound:

$$\frac{\cos\theta}{\sin^{d-1}\theta} = \frac{A}{B\left(1 - \frac{A^2}{B^2}\right)^{\frac{d-1}{2}}} = \frac{AB^{d-2}}{(B^2 - A^2)^{\frac{d-1}{2}}} .$$

For the denominator, note that

$$\begin{aligned} B^2 - A^2 &= (\cosh^2(\alpha) - 1)(1 + \sinh^2(\epsilon)) - \rho^2 \sinh^2(\epsilon)\cosh^2(\alpha) \\ &= (1 - \rho^2)\sinh^2(\epsilon)\cosh^2(\alpha) + \cosh^2(\alpha) - 1 - \sinh^2(\epsilon) \\ &\overset{(i)}{\geq} \cosh^2(\alpha) - 1 - \sinh^2(\epsilon) \end{aligned}$$

where (i) follows from the fact that we can choose ρ arbitrary close to 1. Putting everything together, we have the lower bound

$$T \geq (1 + o(1)) \sqrt{2d} \frac{\rho \sinh(\epsilon)\cosh(\alpha)\left(\sqrt{\cosh^2(\alpha) - 1}\sqrt{1 + \sinh^2(\epsilon)}\right)^{d-2}}{\left(\cosh^2(\alpha) - 1 - \sinh^2(\epsilon)\right)^{\frac{d-1}{2}}} = \Omega(\exp d) ,$$

which is exponential in d. □

D Dimension-distortion trade-off

D.1 Euclidean case

In the Euclidean case, we relate the distance of the support vectors and the size of margin via side length - altitude relations. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ denote support vectors, such that $\langle \boldsymbol{x}, \boldsymbol{w} \rangle > 0$ and $\langle \boldsymbol{y}, \boldsymbol{w} \rangle < 0$ and $\text{margin}(\boldsymbol{w}) = \epsilon$. We can rotate the decision boundary, such that the support vectors are not unique. Wlog, assume that $\boldsymbol{x}_1, \boldsymbol{x}_2$ are equidistant from the decision boundary and $\|\boldsymbol{w}\| = 1$. In this setting, we show the following relation:

Theorem D.1 (Thm. 5.1)**.** $\epsilon' \geq \frac{\epsilon}{c_E^3}$.

Proof. Let $d_1 = d_{\mathcal{X}}(\phi_E^{-1}(\boldsymbol{x}_1), \phi_E^{-1}(\boldsymbol{y}))$, $d_2 = d_{\mathcal{X}}(\phi_E^{-1}(\boldsymbol{x}_2), \phi_E^{-1}(\boldsymbol{y}))$ and $d_3 = d_{\mathcal{X}}(\phi_E^{-1}(\boldsymbol{x}_1), \phi_E^{-1}(\boldsymbol{x}_2))$ the distances between the support vectors in the original space. In the Euclidean embedding space we have

$$\begin{aligned} d_1' &= d_E(\boldsymbol{x}_1, \boldsymbol{y}) \leq \frac{d_1}{c_E} \\ d_2' &= d_E(\boldsymbol{x}_2, \boldsymbol{y}) \leq \frac{d_2}{c_E} \\ d_3' &= d_E(\boldsymbol{x}_1, \boldsymbol{x}_2) \leq \frac{d_3}{c_E} . \end{aligned}$$

d'_1, d'_2, d'_3 are the side lengths of a triangle, whose altitude is given by the margin: $h = 2\epsilon'$. With Heron's equation we get

$$h = 2\epsilon' = \frac{2}{d'_3}\sqrt{s'(s' - d'_1)(s' - d'_2)(s' - d'_3)} ,$$

where $s' = \frac{1}{2}(d'_1 + d'_2 + d'_3)$. In X we have $s' = \frac{1}{2c_E}(d_1 + d_2 + d_3) = \frac{s}{c_E}$. Then we have with respect to the actual distance relations

$$h = 2\epsilon' \quad \frac{2}{c_E d_3}\sqrt{c_E^{-4} s(s - d_1)(s - d_2)(s - d_3)} = 2\frac{\epsilon}{c_E^3} ,$$

which gives the claim. □

D.2 Hyperbolic case

As in the Euclidean case, we want to relate the margin to the distance of the support vectors. Since the distortion can be expressed in terms of the distances of support vector in the original and the embedding space, this allows us to study the influence of distortion on the margin.

We will derive the relation in the half-space model ($\mathbb{P}^2$). However, since the theoretical guarantees above consider the upper sheet of the Lorentz model ($\mathbb{L}_+^{d^0}$), we have to map between the two spaces.

Assumption 4. We make the following assumptions on the underlying data X and the embedding ϕ_H:

1. X is linearly separable;
2. X is hierarchical, i.e., has a partial order relation;
3. ϕ_H preserves the partial order relation and the root is mapped onto the origin of the embedding space.

Under these assumptions, the hyperbolic embedding ϕ_H has two sources of distortion:

1. the (multiplicative) distortion of pairwise distances, measured by the factor $\frac{1}{c_H}$;
2. the distortion of order relations, in most embedding models captured by the alignment of ranks with the Euclidean norm.

Under Ass. 4, order relationships are preserved and the root is mapped to the origin. Therefore, the distortion on the Euclidean norms is given as follows:

$$\|\phi_H(x)\| = d_E(\phi_H(\boldsymbol{x}), \phi_H(0)) = \frac{d_{\mathcal{X}}(\boldsymbol{x}, 0)}{c_H} ,$$

i.e., the distortion on both pairwise distances and norms is given by a factor $\frac{1}{c_H}$.

Note on notation: In the following, a bar over any symbol indicates the Euclidean expression.

D.2.1 Mapping from $\mathbb{L}_+^{d^0}$ to $\mathbb{P}^2$

First, note that a transformation $\boldsymbol{v} \mapsto B\boldsymbol{v}$ with $B = \begin{pmatrix} 1 & 0 \\ 0 & A \end{pmatrix}$ and an orthogonal matrix A is isometric, i.e., it preserves the Minkowski product [6]:

$$(B\boldsymbol{u}) \quad (B\boldsymbol{v}) = u_0 v_0 \quad \boldsymbol{u}_{1:d^0}^T A^T A \boldsymbol{v}_{1:d^0} = u_0 v_0 \quad \boldsymbol{u}_{1:d^0}^T \boldsymbol{v}_{1:d^0} = \boldsymbol{u} \quad \boldsymbol{v} .$$

Setting the first column of A to $\frac{w_{1:d^0}}{\|w_{1:d^0}\|}$ we can isometrically transform the decision hyperplane as $\hat{\boldsymbol{w}} = B\boldsymbol{w} = (\hat{\boldsymbol{w}}_0, \|\hat{\boldsymbol{w}}_{1:d^0}\|, 0, \ldots, 0)$. Analogously, we can transform any point in $\mathbb{L}_+^{d^0}$. In the following, we will use the shorthand $\lambda = \frac{\hat{\boldsymbol{w}}_0}{\hat{\boldsymbol{w}}_1}$. We can then use the maps defined in section A.2 to map $\hat{\boldsymbol{x}} = Bx \in \mathbb{L}_+^2$ onto $\boldsymbol{z} \in \mathbb{P}^2$, i.e. applying $(\pi_{BP} \quad (\pi_{LB} \quad B))$ to any $\boldsymbol{x} \in \mathbb{L}_+^2$ gives $\boldsymbol{z} \in \mathbb{P}^2$.

Remark D.2 (Effect of hyperbolic distortion on Euclidean distances in the Poincare half plane)**.** Note that the hyperbolic distance in the Poincare half plane can be written as follows:

$$d_{\mathsf{P}}((x_0,x_1),(y_0,y_1)) = 2\,\mathrm{asinh}\left(\frac{1}{2}\sqrt{\frac{(x_0 \quad y_0)^2 + (x_1 \quad y_1)^2}{x_1 y_1}}\right)$$
$$= 2\,\mathrm{asinh}\left(\frac{1}{2}\frac{d_E((x_0,x_1),(y_0,y_1))}{\not{p}\overline{x_1 y_1}}\right) .$$

If c_H denotes the hyperbolic distortion, we get

$$d'_{\mathsf{P}} = \frac{d_{\mathsf{P}}}{c_H} = 2\,\mathrm{asinh}\left(\frac{1}{2}\not{p}\frac{d'_E}{\overline{x_1 y_1}}\right)$$
$$) \quad \frac{1}{2}\not{p}\frac{d'_E}{\overline{x_1 y_1}} = \sinh\left(\frac{2\,\mathrm{asinh}\left(\frac{1}{2}\frac{d_E}{\sqrt{x_1 y_1}}\right)}{2c_H}\right) \;\&\; \frac{1}{2}\frac{d_E}{c_H\not{p}\overline{x_1 y_1}} .$$

This suggests, that the effect of hyperbolic distortion on the Euclidean distances can be quantified by a comparable factor, i.e. $d'_E \;\&\; \frac{d_E}{c_H}$.

Lemma D.3 (Relation between h-margin and E-margin)**.** *Let γ_H be the margin of a hyperbolic classifier $\boldsymbol{w}$ 2 R^{d^0+1}. Then the Euclidean margin γ_E of $\boldsymbol{w}$ is bounded as follows: γ_E $\sinh(\gamma_H)$.*

Proof. We again write the hyperbolic distance in the Poincare half plane in terms of the Euclidean distance of the ambient space:

$$d_{\mathsf{P}}((x_0,x_1),(y_0,y_1)) = 2\,\mathrm{asinh}\left(\frac{1}{2}\sqrt{\frac{(x_0 \quad y_0)^2 + (x_1 \quad y_1)^2}{x_1 y_1}}\right)$$
$$= 2\,\mathrm{asinh}\left(\frac{1}{2}\frac{d_E((x_0,x_1),(y_0,y_1))}{\not{p}\overline{x_1 y_1}}\right) ,$$

where $\boldsymbol{y}$ 2 H_w is the point closest to the support vector $\boldsymbol{x}$ 2 $\mathsf{L}_+^{d^0}$ on the decision boundary. Therefore, the hyperbolic margin is $d_{\mathsf{P}}(\boldsymbol{x},\boldsymbol{y}) = \gamma_H$ and the Euclidean margin is $d_E(\boldsymbol{x},\boldsymbol{y}) = \gamma_E$.

Since we mapped the feature space onto the Poincare half plane, $\boldsymbol{y}$ has the coordinates $\boldsymbol{y} = (\not{y}_0, \not{y}_1, 0, \ldots, 0)$ where $\not{y}_0 = y_0$ and $\not{y}_1 = \frac{w^{0T}y^0}{\|w^0\|}$. Similarly, $\boldsymbol{x}$ has the coordinates $\boldsymbol{x} = (\not{x}_0, \not{x}_1, 0, \ldots, 0)$. The transformation preserves the Minkowski product. Therefore we have

$$\boldsymbol{y} \quad \boldsymbol{y} = y_0^2 \quad \boldsymbol{y}'^2 = \not{y}_0^2 \quad \Big(\underbrace{\frac{\boldsymbol{w}'^T\boldsymbol{y}'}{k\boldsymbol{w}'k}}_{=\not{y}_1}\Big)^2 = 1$$

and similarly $\boldsymbol{x} \quad \boldsymbol{x} = \not{x}_0^2 \quad \not{x}_1^2 = 1$. This implies

$$\textcircled{1} \quad \not{y}_1 = \sqrt{\not{y}_0^2 \quad 1},\ \not{x}_1 = \sqrt{\not{x}_0^2 \quad 1}\,,$$

and further

$$\textcircled{2} \quad \not{x}_0,\ \not{y}_0 \quad 1 .$$

We want to show that $\not{x}_1\not{y}_1 \quad 1$. For this, first, note that since $\boldsymbol{y}$ 2 H_w and the hyperbolic margin is γ_H, we have

$$0 = \boldsymbol{w} \quad \boldsymbol{y} = w_0 y_0 \quad \boldsymbol{w}'^T\boldsymbol{y}'$$
$$) \ \boldsymbol{w}'^T\boldsymbol{y}' = w_0 y_0 .$$

This gives

$$\boldsymbol{y} \quad \boldsymbol{y} = y_0^2 \quad \frac{w_0^2 y_0^2}{k\boldsymbol{w}'k^2} = 1$$
$$) \ 0 = y_0^2 \quad \frac{w_0^2 y_0^2}{k\boldsymbol{w}'k^2} \quad 1 ,$$

and therefore

$$\textcircled{3} \quad y_0 = \frac{1}{\sqrt{1 - \frac{w_0^2}{\|w^0\|}}} .$$

Since the hyperbolic margin is γ_H, we further have

$$d_{\mathsf{P}}(\boldsymbol{x}, \boldsymbol{y}) = \operatorname{acosh}(\boldsymbol{x} \quad \boldsymbol{y}) \quad \gamma_H \quad) \quad \boldsymbol{x} \quad \boldsymbol{y} \quad \cosh(\gamma_H) \quad 1 ,$$

and therefore

$$\begin{aligned} & x_0 y_0 \quad x_1 y_1 \quad 1 \\ x_0 y_0 \quad & \sqrt{x_0^2 \quad 1}\sqrt{y_0^2 \quad 1} \quad 1 \\ & (x_0 y_0 \quad 1)^2 \quad (x_0^2 \quad 1)(y_0^2 \quad 1) \\ & x_0^2 y_0^2 \quad 2 x_0 y_0 + 1 \quad x_0^2 y_0^2 \quad x_0^2 \quad y_0^2 + 1 \\ & \quad) \ 0 \quad (x_0 \quad y_0)^2 , \end{aligned}$$

which implies

$$\textcircled{4} \quad x_0 \quad y_0 .$$

This gives for $x_1 y_1$ the following:

$$x_1 y_1 \overset{\textcircled{1}}{=} \sqrt{x_0^2 \quad 1}\sqrt{y_0^2 \quad 1} \overset{\textcircled{4}}{} y_0^2 \quad 1 \overset{\textcircled{3}}{=} \frac{1}{1 \quad \frac{w_0^2}{\|w^0\|}} \quad 1 = \frac{w_0^2}{k\boldsymbol{w}' k^2 \quad w_0^2} .$$

By assumption we have $\boldsymbol{w} \quad \boldsymbol{w} = w_0^2 \quad k\boldsymbol{w}' k^2 = \quad 1$, which gives for the denominator $\quad w_0^2 + k\boldsymbol{w}' k^2 = 1$. It remains to show that $w_0^2 \quad 1$.

For this last step, we want to show that mass concentrates on w_0 as the classifier is updated, ensuring $w_0 \quad 1$. By construction, we have initially $\boldsymbol{w} \quad \boldsymbol{w} = \quad 1$. Wlog, assume that initially $w_0 \quad 1$. An initialization of this form can always be found, e.g., by setting $\boldsymbol{w} = (a, \sqrt{1 + a^2}, 0, \ldots, 0)$ for some $a \quad 0$. If the i^{th} update is negative ($y^i x_0^i < 0$), then $\|\boldsymbol{w}\|_*$ will initially decrease, but the normalization step will scale away the effect on w_0. However, if the i^{th} update is non-negative ($y^i x_0^i \quad 0$), it will increase w_0. Over time, the positive updates concentrate the mass on w_0. Since we initialized to $w_0 \quad 1$, the condition will always stay valid. With the arguments above, this implies $x_1 y_1 \quad 1$. Inserting the latter in the expression above, we get

$$\begin{aligned} d_H \ &= \ 2 \operatorname{asinh}\left(\frac{1}{2}\sqrt{\frac{d_E}{x_1 y_1}}\right) \quad 2 \operatorname{asinh}\left(\frac{d_E}{2}\right) \\ ) \ d_E \quad & 2 \sinh\left(\frac{d_H}{2}\right) \quad \sinh(d_H) . \end{aligned}$$

□

D.2.2 Characterizing the margin

In P^2 the decision hyperplane corresponding to $\hat{w} = Bw$ corresponds to a hypercircle K_w. One can show, that its radius is given by $r_w = \sqrt{\frac{1-\lambda}{1+\lambda}}$ [6], by computing the hyperbolic distance between a point on the decision boundary and one of the hypercircle's ideal points. Further note, that the support vectors lie on hypercircles K_x and K_y, which correspond to the set of points of hyperbolic distance ϵ (i.e., the margin) from the decision boundary. We again assume wlog that at least one support vector is not unique and let $x_1, x_2 \ 2 \ K_x$ and $y \ 2 \ K_y$ (see Fig. 5).

Theorem D.4 (Thm. 5.2)**.** $\epsilon' \quad \epsilon$.

Proof. Our proof consists of three steps:

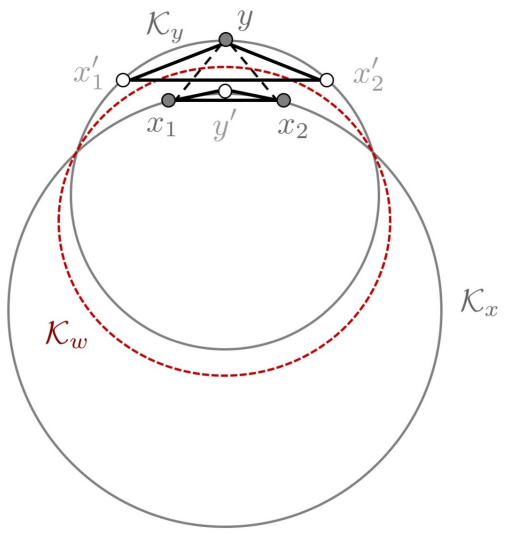

Figure 5: Support vectors on hypercircles K_x and K_y with decision hypercircle K_w.

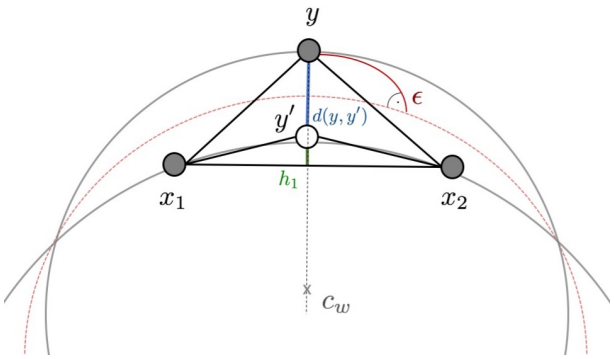

Figure 6: Margin as distance between hypercircles K_x and K_y.

Step 1: Find Euclidean radii and centers of hypercircles. The hypercircles K_x, K_y correspond to arcs of Euclidean circles K_x, K_y in the full plane that are related through circle inversion on the decision circle K_w (i.e., the Euclidean circle corresponding to K_w); see Fig. 5. We can construct a "mirror point" $y' \; 2 \; K_x$ of y by circle inversion on K_w. We have the following (Euclidean) distance relations: The circle inversion gives

$$d(y', c_w) \; d(y, c_w) = r_w^2 \; ,$$

where c_w denotes the center of K_w. Furthermore, we have (see Fig. 7)

$$d(y, c_w) = d(y', c_w) + d(y, y') \; .$$

Putting both together, we get an expression for the Euclidean distance of y and y':

$$\textcircled{1} \quad d(y, y') = d(c_w, y) \quad \frac{r_w^2}{d(c_w, y)} \; .$$

Here, we have by construction $c_w = (0, a, 0, \ldots, 0)$ with a free parameter a. Wlog, assume $c_w = (0, \quad 1, 0, \ldots, 0)$. Next, consider the triangle $\quad (x_1, x_2, y)$. We can express its altitude h in terms of the side length $d(x_1, x_2) =: d_1$, $d(x_1, y) =: d_2$ and $d(x_2, y) =: d_3$ via Heron's formula:

$$h = \frac{2}{d_1} \sqrt{s(s \quad d_1)(s \quad d_2)(s \quad d_3)} \; ,$$

where $s = \frac{1}{2}(d_1 + d_2 + d_3)$. Now, consider the triangle $\quad (x_1, x_2, y')$. Due to the relation between y and y' in $\textcircled{1}$, its altitude h_x is related to h as

$$\textcircled{2} \quad h_x = h \quad d(y, y') \; .$$

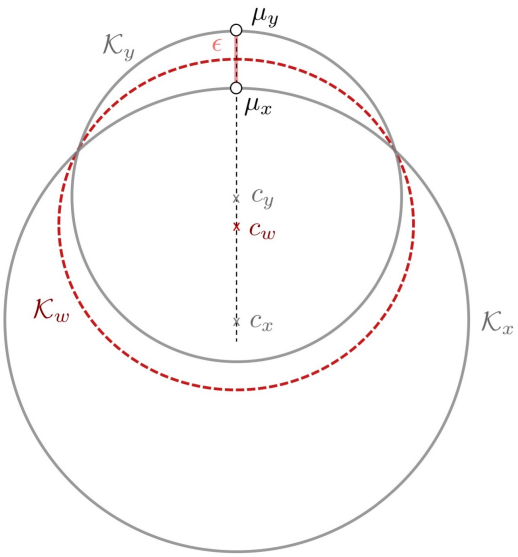

Figure 7: Geometric construction for computing the center and radius of the hypercircle K_x.

With the side length - altitude relations given in (x_1, x_2, y) and ②, we can compute the length of the other sides $d(x_1, y')$ and $d(x_2, y')$ as follows (with Pythagoras theorem):

$$d(x_1, y') = \left(h_x^2 + d(x_1, y)^2 \quad h^2\right)^{1/2}$$
$$d(x_2, y') = \left(h_x^2 + d(x_2, y)^2 \quad h^2\right)^{1/2} .$$

With that, we can compute the radius of K_x as follows: K_x circumscribes (x_1, x_2, y'), therefore its radius r_x can be computed via Heron's formula as

$$r_x = \frac{d(x_1, y') + d(x_2, y') + d(x_1, x_2)}{4A}$$
$$A = \sqrt{s(s \quad d(x_1, y'))(s \quad d(x_2, y'))(s \quad d(x_1, x_2))}$$

where $s = \frac{1}{2}(d(x_1, y') + d(x_2, y') + d(x_1, x_2))$. With an analog construction, we can compute the radius r_y of K_y as function of $d(x_1', x_2')$, $d(x_1', y)$ and $d(x_2', y)$ via relations in the triangle (x_1', x_2', y).

Step 2: Express h-margin as distance between hypercircles. As shown in Fig. 6, the margin is the hyperbolic distance from a point on K_x, K_y to K_w, corresponding to the length of a geodesic connecting the point with the closest point on K_w. Let $v \in K_x$ and $u \in K_w$ the closest point on the decision circle. From the geometry of the Poincare half plane we know that there exists a Möbius transform $\theta \in \text{Möb}(\mathsf{P}^2)$ such that the images $\theta(u) = i\mu$ and $\theta(v) = i\nu$ of u, v lie on the positive imaginary axis. Since the hyperbolic distance is invariant under Möbius transforms, we get

$$d(u, v) = d(\theta(u), \theta(v)) = d(i\mu, i\nu) = \left| \log \frac{\nu}{\mu} \right| .$$

Similarly, we can express the distance between between support vectors $x \in K_x$ and $y \in K$, which is twice the hyperbolic margin: Let $\theta(x) = i\mu_x$ and $\theta(y) = i\mu_y$, where μ_x, μ_y are given by the intersection points of K_x, K_y with the imaginary axis. Then

$$2\epsilon = d(x, y) = \left| \log \frac{\mu_y}{\mu_x} \right| .$$

We can express μ_x, μ_y in terms of the centers and radii of K_x, K_y as follows (Fig. 6)

$$\mu_x = c_x^{(2)} + r_x$$
$$\mu_y = c_y^{(2)} + r_y ,$$

where $c^{(2)}$ denotes the second coordinate of the point $c \in \mathbb{P}^m$. Putting everything together, we get the following expression for the margin:

$$\textcircled{3} \quad \epsilon = \frac{1}{2}\left|\log \frac{c_y^{(2)} + r_y}{c_x^{(2)} + r_x}\right| .$$

Step 3: Evaluate Distortion. As discussed above (Prop. 5.1), the influence of distortion on the altitude h in the triangle (x_1, x_2, y) is given by the factor $\frac{1}{c_H}$.

$$\textcircled{4} \quad h' = \frac{h}{c_H} .$$

r_x depends on pairwise distances between support vectors and h, which are distorted by a factor $\frac{1}{c_H}$ (by assumption on ϕ_H and ④). r_x depends further on h_x which in turn depends on $d(c_w, y)$. The latter depends on the Euclidean norm of the support vector y, i.e., $\|y\|$. With Ass. 4 the total multiplicative distortion is then at most of a factor $\frac{1}{c_H}$. We can derive an analogue result for r_y. For the center c_x note the following:

$$c_x^{(2)} = \frac{1}{2}\left[(1 \quad r_w^2)x_0 \quad (1 + r_w^2)x_1\right] ,$$

where $(x_0, x_1, 0, \ldots, 0) = x = (\pi_{BP} \quad (\pi_{LB} \quad B))$ and $r_w = \sqrt{\frac{1-\lambda}{1+\lambda}}$. Rewriting

$$(1 \quad r_w^2)x_0 = \frac{2w_0x_0}{w_1 + w_0}$$
$$(1 + r_w^2)x_1 = \frac{2w_1x_1}{w_1 + w_0} ,$$

we get

$$c_x^{(2)} = \frac{w^T x}{w_0 + w_1} .$$

Similarly, one can derive

$$c_y^{(2)} = \frac{w^T y}{w_0 + w_1} ,$$

for $(y_0, y_1, 0, \ldots, 0) = y = (\pi_{BP} \quad (\pi_{LB} \quad B))$. Both are only affected by distortion of the form (2), i.e. the multiplicative distortion is given by a factor $\frac{1}{c_H}$. Inserting this into the margin expression (④) gives

$$\epsilon' = \frac{1}{2}\left|\log \frac{c'_y + r'_y}{c'_x + r'_x}\right| \;\&\; \frac{1}{2}\left|\log \frac{\frac{c_y}{c_H} + \frac{r_y}{c_H}}{c_H c_x + c_H r_x}\right| = \frac{1}{2}\left|\log\left(\frac{1}{c_H^2}\frac{c_y + r_y}{c_x + r_x}\right)\right|$$
$$= \frac{1}{2}\Big|\underbrace{\log \frac{1}{c_H^2}}_{\approx 0} + \log \frac{c_y + r_y}{c_x + r_x}\Big| \;\dagger\; \frac{1}{2}\left|\log \frac{c_y + r_y}{c_x + r_x}\right| = \epsilon ,$$

where ($\dagger$) follows from $c_H = O(1 + \epsilon)$ with $\epsilon > 0$ small, by Thm. A.3. □

E Adversarial perceptron

With the geometric tools introduced in Appendix D, we can now also proof Lemma C.4. We restate the result from the main text:

Lemma E.1. *(Adversarial perceptron, Lem. C.4) Let w be the max-margin classifier of S with margin γ_H. At each iteration of Alg. 2, w linearly separates $S \cup S'$ with margin at least $\frac{\gamma_H}{\cosh(\alpha)}$.*

Proof. In the following, we again use the shorthand $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u}\quad\boldsymbol{u}}$, with "+", if $\boldsymbol{u}$ is space-like (i.e., $\boldsymbol{u}\quad\boldsymbol{u} > 0$) and "-", if $\boldsymbol{u}$ is time-like (i.e., $\boldsymbol{u}\quad\boldsymbol{u} < 0$). Since $\boldsymbol{w}$ is "time-like" and $\boldsymbol{x}, \boldsymbol{x}$ space-like, we have

$$\begin{aligned} \textcircled{1} \quad \|\boldsymbol{w}\quad\boldsymbol{x}\| &= \|\boldsymbol{w}\|\,\|\boldsymbol{x}\|\,\cosh\,]\,(\boldsymbol{w},\boldsymbol{x}) \\ \|\boldsymbol{w}\quad\boldsymbol{x}\| &= \|\boldsymbol{w}\|\,\|\boldsymbol{x}\|\,\cosh\,]\,(\boldsymbol{w},\boldsymbol{x})\ . \end{aligned}$$

To prove the statement, we first transform the problem from the Lorentz model L^d to the Poincare half plane P^2 using the map $(\pi_{BP}\quad(\pi_{LB}\quad B))$. Then the adversarial margin is given by the Euclidean distance of the hypercircle $K_{\boldsymbol{x}}$ through $\boldsymbol{x}$ and the decision hypercircle K_w. First, note that we can express this as the hyperbolic distance of the points $(0, \theta(\boldsymbol{x}))$ and $(0, r_w)$, where $\theta\ 2$ Möb(P^2) is a Möbius transform that maps $\boldsymbol{x}$ to the imaginary axis. Importantly, any such θ leaves the Minkowski product invariant. One can show [6] that

$$\theta(\boldsymbol{x}) = c_{\boldsymbol{x}} + \sqrt{c_{\boldsymbol{x}}^2 + r_w}$$

where $c_{\boldsymbol{x}} = \frac{1}{2}\left((1\quad r_w^2)\boldsymbol{x}_0\quad(1 + r_w^2)\boldsymbol{x}_1\right)$ is the Euclidean center of $K_{\boldsymbol{x}}$. The hyperbolic distance is then given by

$$\textcircled{2} \quad \left|\log\frac{\theta(\boldsymbol{x})}{r_w}\right| = \left|\log\left(\frac{c_{\boldsymbol{x}}}{r_w} + \sqrt{\frac{c_{\boldsymbol{x}}^2}{r_w^2} + 1}\right)\right| = \left|\operatorname{asinh}\left(\frac{c_{\boldsymbol{x}}}{r_w}\right)\right|.$$

Note, that

$$\frac{c_{\boldsymbol{x}}}{r_w} = \frac{1}{2}\left[\left(\frac{1}{r_w}\quad r_w\right)\boldsymbol{x}_0\quad\left(\frac{1}{r_w} + r_w\right)\boldsymbol{x}_1\right],$$

where

$$\begin{aligned} \frac{1}{r_w}\quad r_w &= \sqrt{\frac{1+\lambda}{1\quad\lambda}}\quad\sqrt{\frac{1\quad\lambda}{1+\lambda}} = \sqrt{\frac{2\lambda}{1\quad\lambda^2}} = \frac{2w_0}{\sqrt{w_1^2\quad w_0^2}} \\ \frac{1}{r_w} + r_w &= \sqrt{\frac{2}{1\quad\lambda^2}} = \frac{2w_1}{\sqrt{w_1^2\quad w_0^2}}\ . \end{aligned}$$

This gives

$$\textcircled{3} \quad \left|\operatorname{asinh}\left(\frac{c_{\boldsymbol{x}}}{r_w}\right)\right| = \left|\operatorname{asinh}\left(\frac{w_0\boldsymbol{x}_0\quad w_1\boldsymbol{x}_0}{\sqrt{w_1^2\quad w_0^2}}\right)\right| = \left|\operatorname{asinh}\left(\frac{\boldsymbol{w}\quad\boldsymbol{x}}{\|\boldsymbol{w}\|}\right)\right|.$$

Using ②, we can express the adversarial margin in terms of the margin and the distance between features and adversarial samples as follows:

$$\left|\operatorname{asinh}\left(\frac{c_{\boldsymbol{x}}}{\boldsymbol{w}}\right)\right| = \left|\operatorname{asinh}\begin{pmatrix}\frac{c_{\boldsymbol{x}}}{c_x} & \underbrace{\frac{c_x}{r_w}}_{\geq\sinh(\gamma_H)}\end{pmatrix}\right| \overset{\dagger}{\geq} \left|\operatorname{asinh}\left(\frac{c_{\boldsymbol{x}}}{c_x}\sinh(\gamma_H)\right)\right|,$$

where (y) follows from the assumption that $y(\boldsymbol{w}\quad\boldsymbol{x})\quad\sinh(\gamma_H)$ (with margin γ_H). We further show above that we can express the Euclidean centers as

$$c_x = \frac{\boldsymbol{w}\quad\boldsymbol{x}}{w_0 + w_1}\ ,\qquad c_{\boldsymbol{x}} = \frac{\boldsymbol{w}\quad\boldsymbol{x}}{w_0 + w_1}\ .$$

Wlog, assume that $\boldsymbol{w}\quad\boldsymbol{x} > 0$; then $\boldsymbol{w}\quad\boldsymbol{x} < 0$ and therefore

$$c_x = \frac{\|\boldsymbol{w}\|\quad\|\boldsymbol{x}\|}{w_0 + w_1}\ ,\qquad c_{\boldsymbol{x}} = \frac{\|\boldsymbol{w}\|\quad\|\boldsymbol{x}\|}{w_0 + w_1}\ .$$

Inserting ① above in ③, we get

$$\begin{aligned} \operatorname{asinh}\left(\frac{\|\boldsymbol{w}\|\quad\|\boldsymbol{x}\|}{\|\boldsymbol{w}\|\quad\|\boldsymbol{x}\|}\sinh(\gamma_H)\right) &\overset{①}{=} \operatorname{asinh}\left(\frac{\|\boldsymbol{w}\|\,\|\boldsymbol{x}\|\,\cosh(\,]\,(\boldsymbol{w},\boldsymbol{x}))}{\|\boldsymbol{w}\|\,\|\boldsymbol{x}\|\,\cosh(\,]\,(\boldsymbol{w},\boldsymbol{x}))}\sinh(\gamma_H)\right) \\ &= \operatorname{asinh}\left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\frac{\cosh(\,]\,(\boldsymbol{w},\boldsymbol{x}))}{\cosh(\,]\,(\boldsymbol{w},\boldsymbol{x}))}\sinh(\gamma_H)\right) \\ &\overset{\dagger}{\geq} \operatorname{asinh}\left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\sinh(\gamma_H)\right), \end{aligned}$$

where (γ) follows from $\boldsymbol{w}$ being a better classifier for $\boldsymbol{x}$ than for $\boldsymbol{x}$ by construction. Therefore, we have

$$\left| \operatorname{asinh}\left(\frac{j\boldsymbol{w} \quad \boldsymbol{x}j}{j\boldsymbol{w} \quad \boldsymbol{x}j} \sinh(\gamma_H) \right) \right| \quad \operatorname{asinh}\left(\frac{j\boldsymbol{x}j}{j\boldsymbol{x}j} \sinh(\gamma_H) \right) .$$

Furthermore, note, that by construction we have $d_{\mathsf{L}}(\boldsymbol{x}, \boldsymbol{x}) \quad \alpha$ and therefore:

$$\operatorname{acosh}(\boldsymbol{x} \quad \boldsymbol{x}) \quad \alpha \quad) \quad \boldsymbol{x} \quad \boldsymbol{x} \quad \cosh(\alpha) .$$

Since $\boldsymbol{x}, \boldsymbol{x}$ are both space-like, we further have $j\boldsymbol{x}j \; j\boldsymbol{x}j \quad \boldsymbol{x} \quad \boldsymbol{x}$. In summary, this gives

$$\textcircled{4} \quad j\boldsymbol{x}j \quad \frac{\cosh(\alpha)}{j\boldsymbol{x}j} .$$

Inserting $\textcircled{4}$ above, we get

$$\operatorname{asinh}\left(\frac{j\boldsymbol{x}j}{j\boldsymbol{x}j} \sinh(\gamma_H) \right) \overset{\textcircled{4}}{} \operatorname{asinh}\left(\frac{j\boldsymbol{x}j^2}{\cosh(\alpha)} \sinh(\gamma_H) \right)$$
$$\overset{\ddagger}{=} \operatorname{asinh}\left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)} \right) ,$$

where (ζ) follows from $j\boldsymbol{x}j^2 = \boldsymbol{x} \quad \boldsymbol{x} = 1$, since $\boldsymbol{x} \; 2 \; \mathsf{L}^m$. Finally, the claim follows from

$$\operatorname{asinh}\left(\frac{\sinh(\gamma_H)}{\cosh(\alpha)} \right) \quad \frac{\gamma_H}{\cosh(\alpha)} .$$

□

F Additional Experimental Results

F.1 Hyperbolic perceptron

To validate the hyperbolic perceptron algorithm, we performed two simple classification experiments. For the two-class data set (ImageNet n09246464 and n07831146), we observe that hyperbolic perceptron can successfully classify the points into the two groups, i.e., it achieves zero test error. In a second experiment, we try hyperbolic perceptron on a linearly non-separable dataset. The algorithm was still able to classify reasonably well.

F.2 Adversarial Gradient decent

F.2.1 Choice of loss function

Following the large body of work on large-margin learning in Euclidean space, we tested our approach with the classic hinge (Eq. C.2) and least squares losses (Eq. C.3). While both algorithms work well in practise (see § 6 and Section F.2.2), they do not fulfill Ass. 1 on the whole domain. Therefore, our theoretical guarantees are not valid for those loss functions.

We derive theoretical results for the hyperbolic logistic loss (Eq. C.8) instead, which fulfills Ass. 1. Unfortunately, the hyperparameter R_α is difficult to determine in practice. We therefore decided to omit validation experiments with the hyperbolic logistic loss.

For choosing an *adversarial budget* α in practice, note that Assumption 1(2) imposes a norm constraint on the adversarial examples, relative to the maximal norm of the training points. Given the constant R_x, one can estimate an upper bound on α. In addition, an upper bound on α depends on how separable the data set is, i.e., the maximal possible margin. Within these constraints, the choice of α is guided by a trade-off between better robustness and longer training time.

F.2.2 Adversarial GD via least squares loss

Using the same data set as described in § 6, we also try classification in hyperbolic space with adversarial examples using the least squares losses (Eq. C.3). We use the same procedure to find adversarial examples. The results are plotted in Figure 8 with similar conclusions.

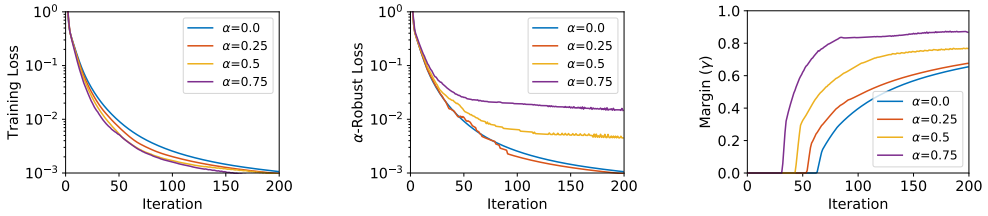

Figure 8: Performance of Adversarial GD using smoothed square loss (Eq. C.3). **Left:** Loss $L(\boldsymbol{w})$ on the original data. **Middle:** α-robust loss $L_\alpha(\boldsymbol{w})$. **Right:** Hyperbolic margin γ_H. We vary the adversarial budget α over $\{0, 0.25, 0.5, 0.75\}$. The case $\alpha = 0$ corresponds to the setup in [6].

F.3 Dimension-distortion trade-off

Euclidean embeddings computed using implementation in Nickel and Kiela [22] by Facebook Research[2].

d	Euclidean	Hyperbolic
4	0.54	0.51
8	0.53	1.00
16	0.68	1.00

Table 2: Classification performance (test error) in hyperbolic vs. Euclidean space of dimension d.

[2] `https://github.com/facebookresearch/poincare-embeddings`