Metric Learning for Dynamic Text Classification

Jeremy Wohlwend **Ethan R. Elenberg** **Samuel Altschul** **Shawn Henry** **Tao Lei**

ASAPP, Inc.

{jeremy,eelenberg,saltschul,shawn,tao}@asapp.com

Abstract

Traditional text classifiers are limited to predicting over a fixed set of labels. However, in many real-world applications the label set is frequently changing. For example, in intent classification, new intents may be added over time while others are removed.

We propose to address the problem of dynamic text classification by replacing the traditional, fixed-size output layer with a learned, semantically meaningful metric space. Here the distances between textual inputs are optimized to perform nearest-neighbor classification across overlapping label sets. Changing the label set does not involve removing parameters, but rather simply adding or removing support points in the metric space. Then the learned metric can be fine-tuned with only a few additional training examples.

We demonstrate that this simple strategy is robust to changes in the label space. Furthermore, our results show that learning a non-Euclidean metric can improve performance in the low data regime, suggesting that further work on metric spaces may benefit low-resource research.

1 Introduction

Text classification often assumes a static set of labels. While this assumption holds for tasks such as sentiment analysis and part-of-speech tagging (Pang and Lee, 2005; Kim, 2014; Brants, 2000; Collins, 2002; Toutanova et al., 2003), it is rarely true for real-world applications. Consider the example of news categorization in Figure 1 (a). A domain expert may decide that the *Sports* class should be separated into two distinct *Soccer* and *Baseball* sub-classes, and conversely merge the two *Cars* and *Motorcycles* classes into a single *Auto* category. Another example is user intent classification in task-oriented dialog systems. In

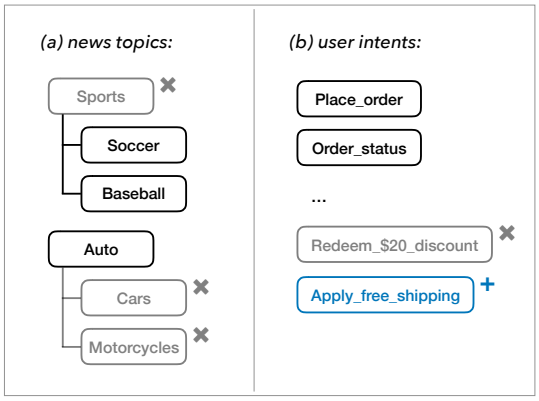


Figure 1: Examples of dynamic classification. In the hierarchical setting (left), new labels are created by splitting and merging old labels. In the flat setting (right), arbitrary labels can be added or removed.

Figure 1 (b) for example, an intent to redeem a reward can be removed when the option is no longer available, while a new intent to apply free shipping can be added to the system. In all of these applications, the classifier must remain applicable for *dynamic classification*, a task where the label set is rapidly evolving.

Several factors make the dynamic classification problem difficult. First, traditional classifiers are not suited to changes in the label space. These classifiers produce a fixed sized output which aligns each of the dimensions to an existing label. Thus, adding or removing any label requires changing the model architecture. Second, while it is possible to retain some model parameters, such as in hierarchical classification models, these architectures must still learn separate weights for every new class or sub-class (Cai and Hofmann, 2004; Kowsari et al., 2017). This is problematic because the new class labels often come with very few training examples, providing insufficient information for learning accurate model weights. Furthermore, these models do not leverage infor-

Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo), pages 143–152

https://doi.org/10.18653/v1/P17

mation across similar labels, which weakens their ability to adapt to new target labels (Kowsari et al., 2017; Tsochantaridis et al., 2005; Cai and Hofmann, 2004).

We propose to address these issues by learning an embedding function which maps input text into a semantically meaningful metric space. The parameterized metric space, once trained on an initial set of labeled data, can be used to perform classification in a nearest-neighbor fashion (by comparing the distance from the input text to reference texts with known label). As a result, the classifier becomes agnostic to changes in the label set. One remaining design challenge, however, is to learn a representation that best leverages the relationship between old and new labels. In particular, the label split example in Figure 1 (b) shows that new labels are often formed by partitioning an old label. This suggests that the classifier may benefit from a metric space that can better represent the structural relationships between labels. Given the hierarchical relationship between the old and new labels, we choose a space of negative curvature (hyperbolic), which has been shown to better embed tree-like structure (Nickel and Kiela, 2017; Sala et al., 2018; Gu et al., 2019).

Our two main contributions are outlined below:

1. We design an experimental framework for *dynamic text classification*, and propose a classification strategy based on prototypical networks, a simple but powerful metric learning technique (Snell et al., 2017).

2. We construct a novel prototypical network adapted to hyperbolic geometry. This requires deriving useful prototypes to represent a set of points on the negatively curved Riemannian manifold. We state sufficient theoretical conditions for the resulting optimization problem to converge. To the best of our knowledge, this is the first application of hyperbolic geometry to text classification beyond the word level.

We perform a thorough experimental analysis by considering the model improvements across several aspects – low-resource fine-tuning, impact of pretraining, and ability to learn new classes. We find that the metric learning approach adapts more gracefully to changes in the label distribution, and outperforms traditional, fixed size classifiers in every aspect of the analysis. Furthermore, our proposed hyperbolic prototypical network outperforms its Euclidean counterpart in the low-resource setting, when fewer than 10 examples per class are available.

2 Related Work

Prototypical Networks and Manifold Learning: This paper builds on the prototypical network architecture (Snell et al., 2017), which was originally proposed in the context of few-shot learning. In both their work and ours, the goal is to embed training data in a space such that the distance to *prototype* centroids of points with the same label define good decision boundaries for labeling test data with a nearest neighbor classifier. Building on earlier work in metric learning (Vinyals et al., 2016; Ravi and Larochelle, 2017), the authors show that learned prototype points also help the network classify inputs into test classes for which minimal data exists. This architecture has found success in computer vision applications such as image and video classification (Weinberger and Saul, 2009; Ustinova and Lempitsky, 2016; Luo et al., 2017). Very recently, prototypical network architectures have shown promising results on relational classification tasks (Han et al., 2018; Gao et al., 2019). To the best of our knowledge, our work is the first application of prototypical network architectures to text classification using non-Euclidean geometry.[1]

Concurrent with the writing of this paper, (Khrulkov et al., 2019) applied several hyperbolic neural networks to few-shot image classification tasks. However, their prototypical network uses the Einstein midpoint rather than the Karcher mean we use in Section 3.3. In (Chen et al., 2019) the authors embed the labels and data separately, then predict hierarchical class membership using an interaction model. Our model directly links embedding distances to model predictions, and thus learns an embedded space that is more amenable to low-resource, dynamic classification tasks.

Hyperbolic geometry has been deeply explored in classical works of differential geometry (Thurston, 2002; Cannon et al., 1997; Berger,

[1] Snell et al. (2017) discuss their formulation in the context of Euclidean distance, cosine distance (spherical manifold), and general Bregman divergences; however, classical Bregman divergence does not easily generalize to hyperbolic space (Section 3.3).

2003). More recently, hyperbolic space has been studied in the context of developing neural networks with hyperbolic parameters (Ganea et al., 2018b). In particular, recent work has successfully applied hyperbolic geometry to graph embeddings (Sarkar, 2011; Nickel and Kiela, 2017, 2018; Sala et al., 2018; Ganea et al., 2018a; Gu et al., 2019). In all of these prior works, the model's parameters correspond to node vectors in hyperbolic space that require Riemannian optimization. In our case, only the model's outputs live in hyperbolic space—not its parameters, which avoids propagating gradients in hyperbolic space and facilitates optimization. This is explained in more detail in Section 3.3.

Hierarchical or Few-shot Text Classification: Many classical models for multi-class classification incorporate a hierarchical label structure (Tsochantaridis et al., 2005; Cai and Hofmann, 2004; Yen et al., 2016; Naik et al., 2013; Sinha et al., 2018). Most models proceed in a top-down manner: a separate classifier (logistic regression, SVM, etc.) is trained to predict the correct child label at each node in the label hierarchy. For instance, HDLTex (Kowsari et al., 2017) addresses large hierarchical label sets explicitly by training a stacked, hierarchical neural network architecture. Such approaches do not scale well to deep and large label hierarchies, while our method can adapt to more flexible settings, such as adding or removing labels, without adding extra parameters.

Our work also relates to text classification in a low-resource setting. While a wide range of methods improve accuracy by leveraging external data such as multi-task training (Miyato et al., 2016; Chen et al., 2018; Yu et al., 2018; Guo et al., 2018), semi-supervised pretraining (Dai and Le, 2015), and unsupervised pretraining (Peters et al., 2018; Devlin et al., 2018), our method makes use of the structure of the data via metric learning. As a result, our method can be easily combined with any of these methods to further improve model performance.

3 Model Framework

This section provides the details of each component of our framework, starting with a more detailed formulation of *dynamic classification*. We then provide some background on prototypical networks, before introducing our hyperbolic variant and its theoretical guarantees.

3.1 Dynamic Classification

Mathematically, we formulate *dynamic classification* as the following problem: given access to an old, labeled training corpus $(x_i, y_i) \in \mathcal{X}_{old} \times \mathcal{Y}_{old}$, we are interested in training a classifier $h : \mathcal{X}_{new} \mapsto \mathcal{Y}_{new}$ with a few examples $(x_j, y_j) \in \mathcal{X}_{new} \times \mathcal{Y}_{new}$. Unlike few-shot learning, the old and new datasets need not be disjoint ($\mathcal{X}_{old} \cap \mathcal{X}_{new} \neq \emptyset, \mathcal{Y}_{old} \cap \mathcal{Y}_{new} \neq \emptyset$).

We consider two different cases: 1) new labels arrive as a consequence of new input data $\mathcal{X}_{new} \setminus \mathcal{X}_{old}$, and 2) during label splitting/merging, some new examples may be constructed by relabeling old examples from $y_i \in \mathcal{Y}_{old}$ to $y_j \in \mathcal{Y}_{new} \setminus \mathcal{Y}_{old}$. This latter case is of particular interest as the classifier may be able to leverage its knowledge of old labels in learning to classify new ones.

There are many natural approaches to this problem. First, a fixed model trained on $\mathcal{X}_{old} \times \mathcal{Y}_{old}$ may be applied directly to classify examples in $\mathcal{X}_{new} \times \mathcal{Y}_{new}$, which we refer to as an *un-tuned* model. Alternately, a pretrained model may also be fine-tuned on $\mathcal{X}_{new} \times \mathcal{Y}_{new}$. Finally, it is also possible to train from scratch on $\mathcal{X}_{new} \times \mathcal{Y}_{new}$, disregarding the model weights trained on the old data distribution. We compare strategies in Sections 4–5.

3.2 Episodic Training

The standard prototypical network is trained using episodic training, as described in (Snell et al., 2017). We view our model as an embedding function which takes textual inputs and outputs points in the metric space. Let $d(x, y)$ denote the distance between two points x and y in our metric space, and let f denote our embedding function. At each iteration, we form a new episode by sampling a set of target labels, as well as support and query points for each of the sampled labels. Let N_C, N_S, and N_Q, be the number of classes tested, the number of support points used, and the number of query points used in each episode, respectively.

For each episode, we first sample N_C classes, $C = \{c_i | i = 1, \ldots, N_C\}$, uniformly across all training labels. We then build a set of support points $S_i = \{s_{i,j} | j = 1, \ldots, N_S\}$ for each of the selected classes by sampling N_S training examples from each selected class. For each support

set, we compute a prototype vector p_i^*. For the standard Euclidean prototypical network, we use the mean of the embedded support set:

$$p_i^* = \frac{1}{N_S} \sum_{j=1}^{N_S} f(s_{i,j}) \ . \tag{1}$$

To compute the loss for an episode, we further sample N_Q query points $Q = \{x_{i,j} | j = 1, \ldots, N_Q\}$ which do not appear in the support set of the episode, for each selected class c_i. We then encode each query sequence and apply a softmax function over the negative distances from the query points to the episode's class prototypes. This yields a probability distribution over classes, and we take the negative log probability of the true class, averaged over the query points, to get the loss for the episode.

$$\frac{-1}{N_C N_Q} \sum_{i=1}^{N_C} \sum_{j=1}^{N_Q} \log \left[\frac{\exp(-d(f(x_{i,j}), p_i^*))}{\sum_k \exp(-d(f(x_{i,j}), p_k^*))} \right],$$

where k in the denominator ranges from 1 to N_C. The steps of a single episode are summarized in Algorithm 1.

Once episodic training is finished, the prototype vectors for a class can be computed as the mean of the embeddings of any number of items in the class. In our experiments, we use the whole training set to compute the final class prototypes, but under lower resources, fewer support points could also be used.

3.3 Hyperbolic Prototypical Networks

In this section we discuss the hyperbolic prototypical network which can better model structural relationships between labels. We first review the hyperboloid model of hyperbolic space and its distance formula. Then we describe the main technical challenge of computing good prototypes in hyperbolic space. Proofs of our uniqueness and convergence will be provided in an extended version. We also describe a second, *distinct* method for computing prototypes which is used to initialize our main method during experiments (a detailed discussion of this point will be provided in an extended version).

Hyperbolic space can be interpreted as a continuous analogue of a tree (Cannon et al., 1997; Krioukov et al., 2010). While trees on n vertices can be embedded in Euclidean space with $\log(n)$ dimensions, hyperbolic space needs only

Algorithm 1 Prototypical Training Episode

Input: D – set of (x, y) pairs
D_i – all pairs with $y = i$
N_C – number of classes sampled each episode
N_S – number of support points
N_Q – number of query points

procedure EPISODE(D, N_C,N_S, N_Q)
  $C \leftarrow$ SAMPLE(D, N_C)
  for $i \in C$ **do**
    $S_i \leftarrow$ SAMPLE(D_i, N_S)
    $Q_i \leftarrow$ SAMPLE($D_i \setminus S_i, N_Q$)
    $c_i \leftarrow$ PROTOTYPE(S_i)
  $P \leftarrow$ CONCAT(c_0; c_1; ...; c_{N_C})
  Loss $\leftarrow 0$
  for each Q_i **do**
    $d_i \leftarrow$ PAIRWISEDIST(Q_i, P)
    Loss $\leftarrow$ Loss $- \frac{1}{N_C N_Q} \log \left[\frac{e^{-d_i}}{\sum_j e^{-d_j}} \right]$

2 dimensions. Additionally, the circumference of a hyperbolic disk grows exponentially with its radius. Therefore, hyperbolic models have room to place many prototypes equidistant from a common parent while maintaining separability from other classes. We argue that this property helps text classification with latent hierarchical structures (e.g. dynamic label splitting).

The reader is referred to Section 2.6 of (Thurston, 2002) for a detailed introduction to hyperbolic geometry, and to (Cannon et al., 1997) for a more gentle introduction. In this section we have adopted the sign convention of (Sala et al., 2018).

Hyperbolic space in d dimensions is the unique, simply connected, d-dimensional, Riemannian manifold with constant curvature -1. The hyperboloid (or Lorentz) model realizes d-dimensional hyperbolic space as an isometric embedding inside $\mathbb{R}^{d+1}$ endowed with a signature $(1, d)$ bilinear form. Specifically, let the coordinates of any $a \in \mathbb{R}^{d+1}$ be $a = (a_0, a_1, ..., a_d)$. Then we can define a bilinear form on $\mathbb{R}^{d+1}$ by

$$B(x, y) = x_0 y_0 - \sum_{j=1}^{d} x_j y_j \ , \tag{2}$$

which allows us to define the hyperboloid to be the set $\{x \in \mathbb{R}^{d+1} | B(x,x) = 1 \text{ and } x_0 > 0\}$. We induce a Riemannian metric on the hyperboloid by restricting $B(\cdot,\cdot)$ to the hyperboloid's tangent space. The resulting Riemannian manifold is hyperbolic space $\mathbb{H}^d$. For $x, y \in \mathbb{H}^d$ the hyperbolic distance is given by

$$d_{\mathbb{H}}(x,y) = \operatorname{arccosh}(B(x,y)). \tag{3}$$

There are several equivalent ways of defining hyperbolic space. We choose to work primarily in the hyperboloid model over other models (*e.g.* Poincaré disk model) for improved numerical stability. We use the d-dimensional output vector h of our network and project it on the hyperboloid embedded in $d+1$ dimensions:

$$h_0 = \sqrt{\sum_{i=1}^{d} h_i^2 + 1}\ , \qquad \bar{h} = [h_0; h]\ . \tag{4}$$

A key algorithmic difference between the Euclidean and the hyperbolic model is the computation of prototype vectors. There are multiple definitions that generalize the notion of a mean to general Riemannian manifolds. One sensible mean $p_X^\star$ of a set X is given by the point which minimizes the sum of squared distances to each point in X.

$$\begin{aligned} p_X^\star &= \arg\min_{p \in \mathbb{H}^d} \phi_X(p) \\ &= \arg\min_{p \in \mathbb{H}^d} \sum_{x \in X} d_{\mathbb{H}^d}(p,x)^2\ . \end{aligned} \tag{5}$$

A proof for the following proposition will be provided in an extended version. We note that concurrent with the writing of this paper, a generalized version of our result appeared in (Gu et al., 2019) as Lemma 2.

Proposition 1. *Every finite collection of points X in $\mathbb{H}^d$ has a unique mean $p_X^\star$. Furthermore, solving the optimization problem (5) with Riemannian gradient descent will converge to $p_X^\star$.*

In an effort to derive a closed form for $p_X^\star$ (rather than solve a Riemannian optimization problem), we conjecture that the following expression is a good approximation. It is computed by averaging the vectors in X and scaling them by the constant which projects this average back to the hyperboloid:

$$\hat{p} = \frac{1}{|X|} \sum_{x \in X} x, \quad \tilde{p} = \frac{\hat{p}}{\sqrt{B(\hat{p},\hat{p})}}. \tag{6}$$

$p_X^\star \neq \tilde{p}$ can be shown to differ through a simple counterexample, although in practice we find little difference between their values during experiments. The proof will be provided in an extended version.

3.4 Implementation and Stability

Our final hyperbolic prototypical model combines both definitions with the following heuristic: initialize problem (5) with $\tilde{p}$ and then run several iterations of Riemannian gradient descent. We find that it is possible to backpropagate through a few steps of the gradient descent procedure described above during prototypical model training. However, we also find that the model can be trained successfully when detaching the gradients with respect to the support points. This suggests that prototypical models can be trained in metric spaces where the mean or its gradient cannot be computed efficiently. Further experimental details are provided in the next section.

Our prototypical network loss function uses both squared Euclidean distance and squared hyperbolic distance for similar reasons. Namely, the distance between two close points is much less numerically stable than the squared distance. In the Euclidean case, the derivative of $\sqrt{s}$ is undefined at zero. In the hyperbolic case, the derivative of $\operatorname{arccosh}(s)$ at 1 is undefined, and $B(x,x) = 1$ for points on the hyperboloid. If we instead use the *squared* hyperbolic distance, L'Hôpital's rule implies that the derivative of $\operatorname{arccosh}(b)^2$ as $b \to 1+$ is 2, allowing gradients to backpropagate through the squared hyperbolic distance without issue.

4 Experiments

We evaluate the performance of our framework on several text classification benchmarks, two of which exhibit a hierarchical label set. We only use the label hierarchy to simulate the label splitting discussed in Figure 1 (a). The models are not trained with explicit knowledge of the hierarchy, as we assume that the full hierarchy is not known *a priori* in the dynamic classification setting. A description of the datasets is provided below:

- 20 Newsgroups (NEWS): This dataset is composed of nearly 20,000 documents, distributed across 20 news categories. We use the provided label hierarchy to form the depth 3 tree used throughout our experiments. We

use 9,044 documents for training, 2,668 for validation, and 7,531 for testing.

- Web of Science (WOS): This dataset was used in two previous works on hierarchical text classification (Kowsari et al., 2017; Sinha et al., 2018). It contains 134 topics, split across 7 parent categories. It contains 46,985 documents collected from the Web of Science citation index. We use 25,182 documents for training, 6,295 for validation, and 15,503 for testing.

- Twitter Airline Sentiment (SENT): This dataset consists of public tweets from customers to American-based airlines labeled with one of 10 reasons for negative sentiment (*e.g.* Late Flight, Lost Luggage).[2] We preprocess the data by keeping only the negative tweets with confidence over 60%. This dataset is non-hierarchical and composed of nearly 7500 documents. We use 5,975 documents for training, 742 for validation, and 754 for testing.

Dynamic Setup: We construct training data for the task of dynamic classification as follows. First, we split our training data in half. The first half is used for pretraining and the second for fine-tuning. To simulate a change in the label space, we randomly remove $p > 0$ fraction of labels in the pretraining data. This procedure yields two label sets, with $\mathcal{Y}_{old}$ (pretraining) $\subset \mathcal{Y}_{new}$ (fine-tuning). In our experiments, we further vary the amount of data available in the fine-tuning set. For the flat dataset, the labels to be removed are sampled uniformly. In the hierarchical case, we create $\mathcal{Y}_{old}$ by randomly collapsing leaf labels into their parent classes, as shown previously in Figure 1.

Hyperparameters and Implementation Details: We apply the same encoder architecture throughout all experiments. We use a 4 layer recurrent neural network, with SRU cells (Lei et al., 2018) and a hidden size of 128. We use pretrained GloVe embeddings (Pennington et al., 2014), which are fixed during training. A sequence level embedding is computed by passing a sequence of word embeddings through the recurrent encoder, and taking the embedding for the last token to represent the sequence. We use the ADAM optimizer with default learning of 0.001, and train for 100 epochs for the baseline models and 10,000 episodes for the prototypical models, with early stopping. In our experiments, we use $N_S = 4$, $N_Q = 64$. We use the full label set every episode for all datasets except WOS, for which we use $N_C = 50$. We use a dropout rate of 0.5 on NEWS and SENT, and 0.3 for the larger WOS dataset. We tuned the learning rate and dropout for each model on a held-out validation set.

For the hyperbolic prototypical network, we follow the initialization and update procedure outlined at the end of Section 3.3 with 5 iterations of Riemannian gradient descent during training and 100 iterations during evaluation. We utilize negative squared distance in the softmax computation in order to improve numerical stability. The means are computed via (5) during both training and model inference. However, this computation is treated as a constant during backpropagation as described in Section 3.3.

Baseline: Our baseline model consists of the same recurrent encoder and an extra linear output layer which computes the final probabilities over the target classes. In order to fine-tune this multilayer perceptron (MLP) model on a new label ontology, we reuse the encoder, and learn a new output layer. This differs from the prototypical models for which the architecture is kept unchanged.

Evaluation: We evaluate the performance of our models using accuracy with respect to the new label set $\mathcal{Y}_{new}$. We also highlight accuracy on only the classes introduced during label addition/splitting, *i.e.* $\mathcal{Y}_{new} \setminus \mathcal{Y}_{old}$. All results are averaged over 5 random label splits with $p = 0.3$.

Results: Table 1 shows the accuracy of the fine tuned models for all three methods. The SENT dataset shows performance in the case where completely new labels are added during fine tuning. In the NEWS and WOS datasets new labels originate from the splits of old labels.

In all cases, the prototypical models outperform the baseline MLP model significantly, especially when the data in the new label distribution is in the low-resource regime (+5–15% accuracy). We also see an increase in performance in the high data regime of up to 5%.

Table 1 further shows that the hyperbolic model outperforms its Euclidean counterpart in the low

[2] https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Dataset	Model	$n_{fine} = 5$	$n_{fine} = 10$	$n_{fine} = 20$	$n_{fine} = 100$
SENT	MLP	37.3 ± 2.9	43.8 ± 3.5	45.7 ± 3.8	57.4 ± 3.5
	EUC	39.6 ± 6.4	45.5 ± 1.8	47.7 ± 4.7	$\mathbf{62.7 \pm 2.1}$
	HYP	$\mathbf{42.2 \pm 3.5}$	$\mathbf{47.1 \pm 4.8}$	$\mathbf{53.0 \pm 2.3}$	$\mathbf{62.7 \pm 2.2}$
NEWS	MLP	49.2 ± 1.0	55.9 ± 2.5	68.5 ± 1.1	76.3 ± 0.5
	EUC	56.5 ± 0.4	65.6 ± 1.0	$\mathbf{74.2 \pm 0.6}$	$\mathbf{79.8 \pm 0.2}$
	HYP	$\mathbf{64.8 \pm 2.8}$	$\mathbf{69.7 \pm 1.0}$	72.9 ± 0.5	78.8 ± 0.4
WOS	MLP	36.6 ± 1.1	46.8 ± 1.2	62.8 ± 0.6	68.9 ± 0.5
	EUC	49.4 ± 1.0	59.2 ± 0.4	$\mathbf{70.4 \pm 0.4}$	73.3 ± 0.2
	HYP	$\mathbf{54.5 \pm 1.4}$	$\mathbf{60.7 \pm 0.9}$	70.2 ± 0.7	$\mathbf{73.5 \pm 0.5}$

Table 1: Test accuracy for each dataset and method. Columns indicate the number of examples per label n_{fine} used in the fine tuning stage. In all cases, the prototypical models outperform the baseline. The hyperbolic model performs best in the low data regime, but both metrics perform comparably when data is abundant.

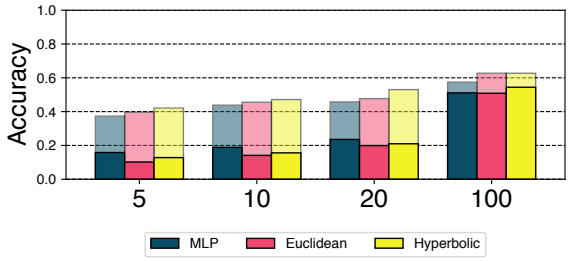
data regime on the NEWS and WOS datasets. This is consistent with our hypothesis (and previous work) that hyperbolic geometry is well suited for hierarchical data. Interestingly, the hyperbolic model also performs better on the non-hierarchical SENT dataset when given few examples, which implies that certain metric spaces may be generally stronger in the low-resource setting. In the high data regime, however, both prototypical models perform comparably.

5 Analysis

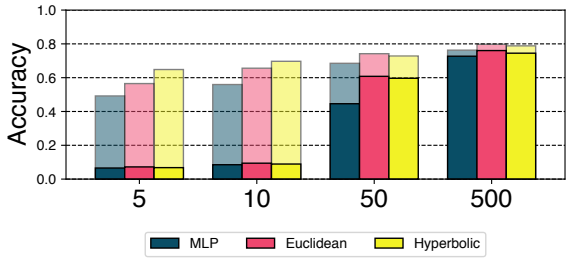
In this section, we examine several aspects of our experimental setup more closely, and use the SENT and NEWS datasets for this analysis.

Benefits of Pretraining We wish to isolate the effect of pretraining on an older label set by measuring the performance of our models on the new label distribution with and without pretraining. Figure 2 shows accuracy without pretraining as solid bars, with the gains due to pretraining shown as translucent bars above them. In the low-data regime without pretraining, all models often perform similarly. Nevertheless, our models do improve substantially over the baseline once pretraining is introduced.

With only a few new examples, our models better leverage knowledge gained from old pretraining data. On the NEWS dataset in particular, with only 5 fine-tune examples per class, the relative reduction in classification error for metric learning models exceeds 53% (Euclidean) and 62% (hyperbolic), while the baseline only reduces relative error by about 45%. This shows that the prototypical network, and particularly the hyperbolic



(a) SENT



(b) NEWS

Figure 2: Accuracy gains from pretraining as a function of the number of examples per class available in the new label distribution. While the models are comparable in the pretraining stage (solid bards), the prototypical models make better use of pretraining, showing higher gain during fine-tuning in both the low and high data data regimes (translucent bars).

model can adapt more quickly to dynamic label shifts. Furthermore, the prototypical models conserve their advantage over the baseline in the high data regime, though the margins become smaller.

Benefits of Fine-tuning An important advantage of the prototypical model is its ability to predict classes that were unseen during training with as few as a single support point for the new class. A natural question is whether fine-tuning on these new class labels immediately improves per-

Model	5	10	20	100
MLP (un-tuned)	38.2	46.7	42.4	46.3
EUC	39.6	45.5	47.7	**62.7**
EUC (un-tuned)	43.4	51.2	47.6	55.8
HYP	42.2	47.1	53.0	**62.7**
HYP (un-tuned)	**45.7**	**52.4**	**53.3**	53.1

(a) SENT

Model	5	10	50	500
MLP (un-tuned)	29.5	34.6	40.3	42.7
EUC	56.5	65.6	**74.2**	**79.8**
EUC (un-tuned)	53.2	56.5	59.6	60.8
HYP	**64.8**	**69.7**	72.9	78.8
HYP (un-tuned)	60.1	62.9	65.4	66.7

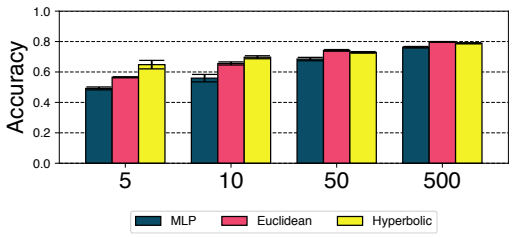
(b) NEWS

Table 2: Test accuracy for each dataset and method. Columns indicate the number of examples per label used for fine-tuning and/or creating prototype vectors.

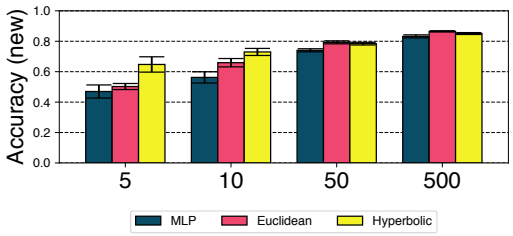
formance, or whether fine-tuning should only be done once a significant amount of data has been obtained from the new distribution. We study this question by comparing the performance of tuned and un-tuned models on the new label distribution.

Table 2 compares the accuracy of two types of pretrained prototypical models provided with a variable number of new examples. The fine-tuned model uses this data for both additional training and for constructing new prototypes. The *un-tuned* model constructs prototypes using the pretrained model's representations without additional training. We also construct an un-tuned MLP baseline by fitting a nearest neighbor classifier (KNN) on the encodings of the penultimate layer of the network. We experimented with fitting the KNN on the output predictions but found that using the penultimate layer was more effective.

We find that the models generally benefit from fine-tuning once a significant amount of data for the new classes is provided (> 20). In the low data regime, however, the results are less consistent, and suggests that the performance may be very dataset dependant. We note however that all metric learning models significantly outperform the MLP-KNN baseline in both the low and high data regimes. This shows that regardless of fine-tuning, our approach is more robust on previously unseen classes.



(a) Accuracy with respect to the full label set



(b) Accuracy with respect to new classes only

Figure 3: Accuracy on the NEWS Dataset against number of fine tune examples: (a) all classes and (b) newly introduced classes only. The mean is taken over 5 random label splits, and error bars are given at ± 1 standard deviation. The gap between the hyperbolic models and the others is even larger on the new classes.

Learning New Classes An important factor in the dynamic classification setup is the ability for the model to not only keep performing well on the old classes, but also to smoothly adapt to new ones. We highlight the performance of the models on the newly introduced labels in Figure 3, where we see that the improvement in accuracy is dominated by the performance on the new classes.

6 Conclusions

We propose a framework for dynamic text classification in which the label space is considered flexible and subject to frequent changes. We apply a metric learning method, namely prototypical network, and demonstrate its robustness for this task in a variety of data regimes. Motivated by the idea that new labels often originate from label splits, we extend prototypical networks to hyperbolic geometry, derive expressions for hyperbolic prototypes, and demonstrate the effectiveness of our model in the low-resource setting. Our experimental findings suggest that metric learning improves dynamic text classification models, and offer insights on how to combine low-resource training data from overlapping label sets. In the future we hope to explore other applications of metric learning to low-resource research, possibly in combination with explicit models for label entailment (tree learning, fuzzy sets), and/or Wasserstein distance.

References

Marcel Berger. 2003. *A Panoramic View of Riemannian Geometry*. Springer-Verlag Berlin Heidelberg, Heidelberg, Germany.

Thorsten Brants. 2000. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics.

Lijuan Cai and Thomas Hofmann. 2004. Hierarchical document categorization with support vector machines. In *CKIM*, pages 78–87.

James Cannon, William Floyd, Richard Kenyon, and Walter Parry. 1997. Hyperbolic geometry. http://library.msri.org/books/Book31/files/cannon.pdf.

Boli Chen, Xin Huang, Lin Xiao, Zixin Cai, and Liping Jing. 2019. Hyperbolic interaction model for hierarchical multi-label classification. https://arxiv.org/pdf/1905.10802.pdf.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NeurIPS*, pages 3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. 2018a. Hyperbolic entailment cones for learning hierarchical embeddings. In *ICML*.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. 2018b. Hyperbolic neural networks. In *NeurIPS*.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *AAAI*.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2019. Learning mixed-curvature representations in products of model spaces. In *ICLR*.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *EMNLP*. Association for Computational Linguistics.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*.

Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. 2019. Hyperbolic image embeddings. https://arxiv.org/pdf/1904.02239.pdf.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, page 17461751.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. HDLTex: hierarchical deep learning for text classification. In *IEEE ICMLA*, pages 364–371.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. 2010. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *EMNLP*.

Zelun Luo, Yuliang Zou, Judy Hoffman, and Li Fei-Fei. 2017. Label efficient learning of transferable representations across domains and tasks. In *NeurIPS*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Azad Naik, Anveshi Charuvaka, and Huzefa Rangwala. 2013. Classifying documents within multiple hierarchical datasets using multi-task learning. In *IEEE International Conference on Tools with Artificial Intelligence*.

Maximilian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*, pages 3776–3785.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*, pages 6338–6347.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the annual meeting on assocation for computational linguistics*, pages 115–224.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *ICLR*.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. 2018. Representation tradeoffs for hyperbolic embeddings. In *ICML*, pages 4460–4469.

R. Sarkar. 2011. Low distortion Delaunay embedding of trees in hyperbolic plane. In *Proc. of the International Symposium on Graph Drawing (GD 2011)*, pages 355–366.

Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung, and Derek Ruths. 2018. A hierarchical neural attention-based text classifier. In *EMNLP*, pages 817–823.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087.

William Thurston. 2002. The geometry and topology of three-manifolds: Chapter 2, elliptic and hyperbolic geometry. http://library.msri.org/books/gt3m/PDF/2.pdf.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, pages 173–180. Association for Computational Linguistics.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.

Evgeniya Ustinova and Victor Lempitsky. 2016. Learning deep embeddings with histogram loss. In *NeurIPS*, pages 4170–4178.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638.

Kilian Weinberger and Lawrence Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244.

Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*, pages 3069–3077.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *NAACL*.

Hongyi Zhang and Suvrit Sra. 2016. First-order methods for geodesically convex optimization. In *COLT*, volume 49, pages 1617–1638.