Distributionally Robust $k$-Nearest Neighbors for Few-Shot Learning

Shixiang Zhu  
Georgia Institute of Technology  
Atlanta, GA 30332  
shixiang.zhu@gatech.edu

Liyan Xie  
Georgia Institute of Technology  
Atlanta, GA 30332  
lxie49@gatech.edu

Minghe Zhang  
Georgia Institute of Technology  
Atlanta, GA 30332  
minghe_zhang@gatech.edu

Rui Gao  
University of Texas at Austin  
Austin, TX 78712  
rui.gao@mccombs.utexas.edu

Yao Xie  
Georgia Institute of Technology  
Atlanta, GA 30332  
yao.xie@isye.gatech.edu

Abstract

Learning a robust classifier from a few samples remains a key challenge in machine learning. A major thrust of research in few-shot classification has been based on metric learning to capture similarities between samples and then perform the $k$-nearest neighbor algorithm. To make such an algorithm more robust, in this paper, we propose a distributionally robust $k$-nearest neighbor algorithm Dr.$k$-NN, which features assigning minimax optimal weights to training samples when performing classification. We also couple it with neural-network-based feature embedding. We demonstrate the competitive performance of our algorithm comparing to the state-of-the-art in the few-shot learning setting with various real-data experiments.

1 Introduction

Machine learning has been proven to be successful in data-intensive applications, but is often hampered when the data set is small. For example, in breast mammography diagnosis for breast cancer screening [3], the diagnosis of the type of breast cancer requires specialized analysis by pathologists in a highly time- and cost-consuming task and often leads to non-consensual results. As a result, labeled data in digital pathology are generally very scarce; so do many other applications. Few-shot classification [35] aims to find a classifier given only a few labeled samples for each class.

In this paper, we focus on $k$-nearest neighbor ($k$-NN) algorithm for few-shot learning [16], a natural idea to tackle this problem and shows promising empirical performances. Notable contributions include matching network [24], prototypical network [31], and their extensions. They are primarily based on distance-weighted $k$-NN, which classifies an unseen sample (aka. query) by a weighted vote of its labeled neighbors and uses the distance between two data points in the embedding space as their weights.

The classification performance of distance-weighted $k$-NN critically depends on the choice of the distance measuring the similarity between samples, which is typically performed by metric learning that automatically constructing task-specific distance metrics from supervised data [16,17,22,24,34]. However, it has been recognized that metric learning based nearest neighbor algorithms are not robust.
Figure 1: Motivating example: a small training set of three samples of image-label pairs: the first image is a mop, the second image is a dog. However, the third image looks like a mop but is, in fact, a dog (dressing up as a mop). The query (the last image) is “closer” to the third one, and more likely to be misclassified as a dog if we use distance-weighted k-NN.

to outliers. An example to illustrate this issue is shown in Figure 1. The training set with only three labeled samples includes two categories we want to classify: mop and Komondor. As we can see, the query is visually closer to the third sample, and more likely to be misclassified as a mop with respect to some distance metric, even though it is actually a Komondor dressing up as a mop. This outlier misleads the metric learning model to capture irrelevant details (e.g., the bucket and the mop handle) for the category of Komondor. Such a problem can become even severe when the sample size is small.

To develop algorithms that are more robust to outliers, instead of using a weight for each sample measuring its distance to the query, we introduce a vector of weights, one for each class, for each sample in k-NN and perform a weighted majority vote. Ideally, the weight vectors are learned from training samples to represent their significance in different categories. An example is illustrated in Figure 2.

The above idea is formalized using ideas based on distributionally robust hypothesis testing [15]. Given the features of training samples, we solve a Wasserstein distributionally robust classification problem that finds the minimax optimal classifier among all possible classifiers. The resulting least favorable distributions on the training samples are then chosen to be the weight vector. We show that this procedure is equivalent to solving a distributionally robust k-NN. Further, using differentiable optimization [3, 2], we incorporate the minimax classifier into a neural network that jointly learns the feature embedding and the minimax optimal classifier. Numerical experiments show that our algorithm can effectively improve the performance of the few-shot classification on various data sets.

Figure 2: An illustration shows the difference between Dr. k-NN and standard k-NN. Each colored dot is a training sample, where the color indicates its class-membership. The horizontal/vertical bar represents the probability mass of one training sample under distribution $P_1, P_2$, respectively.
1.1 Related work

Recently, there has been much interest in the few-shot problem, see [35] for a recent survey. Many related machine learning approaches have been proposed, such as meta-learning [12, 17, 23, 27], embedding learning [5, 31–34], generative modeling [21, 25], and adversarial learning [36]. The main idea of our work is mostly related to the metric learning [16, 24, 31, 34, 17, 22], which essentially translates the hidden information carried by the limited data into a distance metric. For example, seminal work [16] and the follow-up non-linear version [24] propose neighborhood component analysis (NCA) that learns a distance metric which optimizes the expected leave-one-out classification error on the training data when used with a stochastic neighbor selection rule. In this paper, we take a different probabilistic approach to exploit information from the data: other than looking for a new distance metric directly, we construct an uncertainty set for each class using certain distance metric. In this way, we are able to find the least favorable distributions (LFDs) supported on the training set by solving a minimax problem. These LFDs reveal the significance of each sample in the worst case, thereby contributing effectively to decision making.

Wasserstein distributionally robust optimization [11, 1, 7, 14, 30, 6, 28, 13] is an emerging paradigm for statistical learning; see [19] for a recent survey. Our work is inspired by [15], a framework for Wasserstein robust hypothesis testing, but there are several major differences. First, [15] only studies two hypotheses and searches over deterministic test/classification rule; while we consider multiple classes and search over randomized classifiers and thereby we do not need to introduce a convex approximation for the error probabilities as did in [15]. Second, while [15] requires sample features as an input, we develop a scalable algorithmic framework to simultaneously learn the optimal feature extractor parameterized by neural networks and robust classifier to achieve the best performance. A recent work [8] studies distributionally robust k-nearest neighbor regression problem. Note that regression and classification are fundamentally different as different performance metrics are used: in [8] the objective is to minimize the mean square error, whereas in our work the performance metric is classification errors. In statistics, there has been a well-known work on optimal weighted nearest neighbor classifier [26], which assigns one weight to each sample when performing k-NN; the optimal weights minimize asymptotic expansion for the excess risk (regret). This is a very different approach from ours: [26] considers binary classifier to minimize the risk and each training sample is associated with one weight computed using density function; we consider minimax robust multi-class classification, each training sample is associated with different weights for different classes, whose calculation does not require the knowledge of density.

2 Proposed Algorithm Dr.k-NN

In this section, we introduce the model setup, state a Distributional Robust k-Nearest Neighbor (Dr.k-NN) algorithm and illustrate it by an example, but postpone its theoretical motivation to the next section.

Suppose there are $M$ classes, each of which has $n_m$ training samples $x_{m1}, \ldots, x_{mn_m}$ drawn from a space $\mathcal{X}, m = 1, \ldots, M$. Assume each sample can be represented as a vector in a feature space $\Omega \subset \mathbb{R}^d$. Let $\phi : \mathcal{X} \to \Omega$ be a feature extractor that embeds samples to the feature space $\Omega$ (in Section 4 we will train a neural network to learn $\phi$). We denote the samples in the feature space as $\hat{\omega}_im := \phi(x_{im}), i = 1, \ldots, n_m, \hat{\Omega}_m := \{\hat{\omega}_im, \ldots, \hat{\omega}_mn_m\}, m = 1, \ldots, M$.

We label all samples from 1 to $n := \sum_{m=1}^M n_m$ and set $\hat{\omega}_i := \phi(x_i), i = 1, \ldots, n, \hat{\Omega} := \cup_{1 \leq m \leq M} \hat{\Omega}_m$.

Let $c : \Omega \times \Omega \to \mathbb{R}_+$ be a norm that measures similarity between features. Define empirical distributions $\hat{P}_m := \frac{1}{n_m} \sum_{i=1}^{n_m} \delta_{\hat{\omega}_im}, m = 1, \ldots, M,$

where $\delta$ denotes the Dirac point mass. Let $\vartheta_1, \ldots, \vartheta_M > 0$ be a set of tuning parameters whose meaning will be made clear in (3). Our algorithm contains two steps.
Step 1. [Sample re-weighting] For each class \( m \), re-weight the \( n \) samples using a distribution \( P_m^* \), where \( (P_1^*, \ldots, P_M^*) \) is the \( p \)-component of the minimizer of the following convex program

\[
\begin{align*}
\min_{\gamma_1, \ldots, \gamma_M \in \mathbb{R}_+^{n \times n}} & \quad \sum_{i=1}^{n} \max_{1 \leq m \leq M} p_i^m \\
\text{subject to} & \quad \sum_{j=1}^{n} \gamma_{i,j}^m \leq \theta_m, \quad m = 1, \ldots, M, \\
& \quad \sum_{i=1}^{n} \gamma_{i,j}^m = \hat{P}_m(\hat{\omega}^j), \quad \forall 1 \leq i, j \leq N, \quad 1 \leq m \leq M.
\end{align*}
\]

This problem is derived from a distributionally robust problem that will be explained in Section 3.

Step 2. [k-NN] Given a query point \( \omega \), ordering the training samples according to their distance to \( \omega \):
\( c(\omega, \hat{\omega}^{(1)}) \leq c(\omega, \hat{\omega}^{(2)}) \leq \cdots \leq c(\omega, \hat{\omega}^{(n)}) \). Compute the weighted \( k \)-nearest neighbor votes,
\( \bar{P}_m(\omega) := \frac{1}{k} \sum_{i=1}^{k} P_m^*(\hat{\omega}^{(i)}), \quad m = 1, \ldots, M. \)

Decide the class for a query point \( \omega \) as \( \arg \max_{1 \leq m \leq M} \bar{P}_m(\omega) \), where the tie can be broken arbitrarily.

Figure 3 gives an illustration showing the probabilistic weights \( (1) \) and their corresponding results of Dr.k-NN using a small subset of MNIST (digit 4 (red), 6 (blue), 9 (green) and \( k = 5 \)). In (a)(b)(c), shaded areas indicate the kernel smoothing of \( P_1^*, P_2^*, P_3^* \) defined in (5). Raw samples are projected on a 2D feature space (\( d = 2 \)), with the color indicating their true class-membership. In (d), big dots represent the training points and small dots represent the query points, and their color depth suggests how likely the sample is being classified into the true category.

3 Theoretical Foundation of Dr.k-NN

In this section, we present the theoretical framework that justifies the sample re-weighting step in Dr.k-NN. In Section 3.1 we define a distributionally robust classification problem and show it is equivalent to (1) in 3.2. Thereby its least favorable distributions (LFDs) suggest a weighted majority vote for k-NN. In Section 3.3 we relate this problem to a distributionally robust k-NN problem.

3.1 Distributionally robust classification

We define a distributionally robust classification problem whose worst-case distributions will be used for re-weighting the empirical samples. Let \( \pi : \Omega \rightarrow \Delta_M \) be a randomized classifier that assigns class \( m \) with probability \( \pi_m(\omega) \) to a query \( \omega \), where \( \Delta_M \) is the probabilistic simplex
\( \Delta_M = \{ \pi \in \mathbb{R}_+^M : \sum_{m=1}^{M} \pi_m = 1 \} \). Suppose each class \( m \) is associated with a distributional uncertainty set \( \mathcal{P}_m^* \) which will be specified shortly. Given \( \mathcal{P}_1, \ldots, \mathcal{P}_m \), define the worst-case risk of a classifier \( \pi \) as the worst-case total error probabilities
\[
\max_{\pi_m \in \mathcal{P}_m, 1 \leq m \leq M} \left\{ \Psi(\pi; P_1, \ldots, P_M) := \sum_{m=1}^{M} \mathbb{E}_{\omega \sim P_m^*} [1 - \pi_m(\omega)] \right\}.
\]
We consider the following minimax robust classification problem that finds a classifier minimizing the worst-case risk.

\[
\min_{\pi : \Omega \rightarrow \Delta_M} \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \Psi(\pi; P_1, \ldots, P_M).
\]

(2)

Now we describe the uncertainty set \( \mathcal{P}_m \). First, since we are going to use the worst-case distributions to re-weight the few-shot samples, we will restrict the support of every distribution in \( \mathcal{P}_m \) to \( \hat{\Omega} \), the support of empirical points. Second, the uncertainty set is data-driven, containing the empirical distribution \( \hat{P}_m \) and distributions surrounding its neighborhood. Third, to measure the closeness between distributions, we would like to choose the Wasserstein metric, defined as \( \mathcal{W}(P, P') := \min_{\gamma} \mathbb{E}_{(\omega, \omega') \sim \gamma} [c(\omega, \omega')] \) for any two distributions \( P \) and \( P' \) on \( \hat{\Omega} \), where the minimization of \( \gamma \) is taken over the set of all probability distributions on \( \Omega \times \Omega \) with marginals \( P \) and \( P' \). One advantage of using Wasserstein metric is that it takes account of the geometry of the feature space by incorporating the metric \( c \) in its definition. Given the empirical distribution \( \hat{P}_m \), we define

\[
\mathcal{P}_m := \{ P_m \in \mathcal{P}(\hat{\Omega}) : \mathcal{W}(P_m, \hat{P}_m) \leq \vartheta_m \}, \quad m = 1, \ldots, M,
\]

(3)

where \( \mathcal{P}(\hat{\Omega}) \) denotes the set of all probability distributions on \( \hat{\Omega} \); \( \vartheta_m > 0 \) specifies the size of the uncertainty set for the \( m \)-th class that sets the amount of deviation we would like to control.

### 3.2 Re-weighting samples using LFDs

We establish the following theorem stating that the re-weighting vectors \( P_1^*, \ldots, P_M^* \) solved in (1) are exactly the LFDs to problem (2) with uncertainty set (3).

**Theorem 1.** For the uncertainty sets defined in (3), problems (1) and (2) yield the same optimal solution \( (P_1^*, \ldots, P_M^*) \).

**Remark 1 (Interpretation of (1)).** The decision variable \( \gamma_m \in \mathbb{R}_+^{n \times n} \) can be viewed as a joint distribution on \( n \) empirical points with marginal distributions \( \hat{P}_m \) and \( P_m \), represented by a vector \( p_m \in \mathbb{R}_+^n \). The inequality constraint controls the Wasserstein distance between \( P_m \) and \( \hat{P}_m \). Let \( y^i_m \in \{0, 1\} \) be the class indicator variable of sample \( \hat{\omega}^i \). The objective can be equivalent rewritten as minimization of total margin

\[
\sum_{i=1}^n \sum_{m=1}^M \left( \max_{1 \leq m' \leq M} p^{i}_{m'} - p^i_m \right),
\]

where \( \max_{1 \leq m' \leq M} p^{i}_{m'} - p^i_m \) measures the margin between the maximum likelihood of \( \hat{\omega}^i \) among all classes and the likelihood of the \( m \)-th class of \( \hat{\omega}^i \). When \( M = 2 \), the total margin reduces to the total variation distance. Also, observe that

\[
\sum_{i=1}^n \max_{1 \leq m \leq M} p^i_m = \lim_{\ell \to \infty} \left( \sum_{i=1}^n \sum_{m=1}^M y^i_m p^i_m - \frac{1}{\ell} \sum_{i=1}^n \sum_{m=1}^M y^i_m \log \frac{\exp(tp^i_m)}{\sum_{m=1}^M \exp(tp^i_m)} \right),
\]

where the second term on the right side represents the cross-entropy (or negative log-likelihood).

Therefore, problem (1) perturbs \( (\hat{P}_1, \ldots, \hat{P}_M) \) to LFDs \( (P_1^*, \ldots, P_M^*) \) so as to minimize the total margin as well as an upper bound on cross-entropy of LFDs; the smaller the margin (or cross-entropy) is, the more similar between classes and thus the harder to distinguish among them.

The proof of Theorem 1 is based on the following two lemmas.

**Lemma 1.** Fix probability distributions \( P_1, \ldots, P_M \in \mathcal{P}(\hat{\Omega}) \). Then

\[
\psi(P_1, \ldots, P_M) := \min_{\pi : \hat{\Omega} \rightarrow \Delta_M} \Psi(\pi; P_1, \ldots, P_M) = M - \max_{1 \leq m \leq M} P_m(\hat{\omega}^i) = n - \max_{1 \leq m \leq M} P_m(\hat{\omega}^i).
\]

Furthermore, the optimal classifier \( \pi^* \) satisfies \( \pi^*_m(\hat{\omega}) = 1 \) if \( m_0 \) is the smallest index such that

\[
\sum_{m=1}^M P_m(\hat{\omega}) = m_0 \in \arg \max_{1 \leq m \leq M} \frac{P_m(\hat{\omega})}{\sum_{m=1}^M P_m(\hat{\omega})}.
\]

This lemma gives a closed-form expression for the risk of optimal classifier if \( P_1, \ldots, P_M \) are known, and shows that the optimal decision \( \pi^* \) accepts the class with the maximum likelihood. Note that any other tie-breaking rule is valid.
Lemma 2. For the uncertainty sets defined in (3), the problem \( \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \psi(P_1, \ldots, P_M) \) is equivalent to (1).

Complete proofs are provided in the supplementary material.

3.3 Distributionally robust \( k \)-NN

Let \( Q_1, \ldots, Q_M \in \mathcal{P}(\Omega) \). In the sequel we use a shorthand notation \( Q := (Q_1, \ldots, Q_M) \). Let \( \omega \in \Omega \) and suppose \( c(\omega, \tilde{\omega}^{(1)}) \leq \cdots \leq c(\omega, \tilde{\omega}^{(n)}) \). Define

\[
m_0(\omega) := \min \left\{ m' : m' \in \arg \max_{1 \leq m \leq M} \frac{1}{k} \sum_{i=1}^{k} Q_m(\tilde{\omega}^{(i)}) \right\}.
\]

We define a weighted \( k \)-NN classifier \( \pi^{\text{knn}}(\omega; k, Q) : \Omega \to \Delta_M \) as

\[
\pi^{\text{knn}}_{m_0}(\omega; k, Q) = \begin{cases} 1, & m = m_0(\omega) \\ 0, & \text{otherwise}. \end{cases}
\]

That is, for any \( 1 \leq k \leq n \) and any vector of weights \( Q \), \( \pi^{\text{knn}}(\cdot; k, Q) \) defines a weighted \( k \)-NN classifier. Consider the following distributionally robust \( k \)-NN problem

\[
\min_{Q_1, \ldots, Q_M \in \mathcal{P}(\hat{\Omega})} \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \sum_{m=1}^{M} \mathbb{E}_{\omega_m \sim P_m} \left[ 1 - \pi^{\text{knn}}_{m_0}(\omega_m; k, Q) \right],
\]

which finds the best weighted \( k \)-NN classifier that minimizes the worst-case risk. The following theorem establishes the equivalence between (3) and (4) and together with Theorem 1 justifies the formulation (1).

Theorem 2. For the uncertainty set defined in (3), (2) and (4) have identical optimal values.

4 Jointly Learning Feature Mapping and Robust Classifier

In this section, we propose a framework that jointly learns the feature mapping and the robust classifier.

Figure 4: An overview of the framework, which consists of two cohesive components: (1) an architecture that is able to produce feature embedding \( \omega \) and least favorable distributions \( P^*_m \) for training set; (2) an Dr. \( k \)-NN makes decisions for any unseen sample \( \omega \) based on the estimated weight vector \( P^*_m(\omega) \) (probability mass on least favorable distributions).

We parameterize the feature mapping \( \phi(\cdot; \theta) \) by a neural network with weights \( \theta \) whose input is a batch of training samples (Figure 4), and then compose it with an optimization layer that packs the convex problem (1) as an output layer that outputs the LFDs of (2). The optimization layer is adopted from differentiable optimization [2][3]. To apply the mini-batch stochastic gradient descent, we also need to ensure that each batch comprises of multiple “mini-sets”, one for each class, containing at least one training sample from each class fed into the convex optimization layer.

In light of (1), the objective of our joint learning framework is

\[
J(\theta; P^*_1, \ldots, P^*_M) := \min_{\theta} \sum_{i=1}^{n} \max_{1 \leq m \leq M} P^*_m(\phi(x^i; \theta)),
\]
where \( \{ P_m, (\phi(; \theta)) \}_{1 \leq m \leq M} \) are the LFDs generated by the convex solver defined in (1) given input variables \( \{ \hat{\omega} = \phi(x^i; \theta) \}_{1 \leq i \leq n} \). The algorithm for carrying out the training procedure of our framework is presented in Algorithm 1.

**Algorithm 1: Learning algorithm for Dr. k-NN**

**Input:** \( D_m := \{ (x^i, y^i) : y^i = m, \forall i \} \subset \mathcal{D} \), \( m = 1, \ldots, M \);  
**Output:** The feature mapping \( \phi(; \theta) \) and the LFD \( P_1^*, \ldots, P_M^* \) supported on training samples;  
**Initialization:** \( \theta_0 \) is randomly initialized; \( n' < n \) is the size of “mini-set”; \( t = 0 \);  
**while** \( t < T \) **do**  
  **for** number of mini-sets do  
    Randomly generate \( M \) integers \( n_1, \ldots, n_M \) such that \( \sum_{m=1}^M n_m = n', n_m > 0 \);  
    Initialize two ordered sets \( \hat{\Omega} = \emptyset, \hat{\Phi} = \emptyset \);  
    for \( m \in \{ 1, \ldots, M \} \) **do**  
      \( X_m \leftarrow \) Randomly sample \( n_m \) data samples \( \{ x^i \} \) from \( D_m \);  
      \( \hat{\Omega}_m \leftarrow \{ \hat{\omega} := \phi(x; \theta_t) : x \in X_m \}; \hat{\Phi}_m \leftarrow \frac{1}{n_m} \sum_{i=1}^{n_m} \delta_{x^i}, \hat{\omega}_i \in \hat{\Omega}_m \}; \hat{\Omega} \leftarrow \hat{\Omega} \cup \hat{\Omega}_m; \hat{\Phi} \leftarrow \hat{\Phi} \cup \hat{\Phi}_m \);  
    end  
  end  
  Update the probability mass of LFDs \( P_1^*, \ldots, P_M^* \) on \( \hat{\Omega} \) by solving (1) given \( \hat{\Omega}, \hat{\Phi} \);  
\( \theta_{t+1} \leftarrow \theta_t + \alpha \nabla J(\theta_t; P_1^*, \ldots, P_M^*) \), where \( \alpha \) is the learning rate; \( t \leftarrow t + 1 \);  
end

5 Memory-efficient Implementation of Dr. k-NN in Data-Intensive Setting

For the sake of completeness, we extend our algorithm to non-few-shot setting. This can be particularly useful for the general classification problem with an arbitrary size of training set. In fact, k-NN methods notoriously suffer from computational inefficiency if the number of labeled samples \( n \) is large, since it has to store and search through the entire training set [16].

The main idea is to only keep the training points that are important in deciding the decision boundary based on the maximum entropy principle [9]. As a measure of importance, we choose the samples with the largest entropy across all categories, based on the intuition that the samples with higher entropy has larger uncertainty and will be more useful for classification purposes since they tend to lie on the decision boundary. The entropy of a sample is defined as follows. Consider a random variable which takes value \( m \) with probability \( \pi_m \), \( \sum_{i=1}^M \pi_m = 1 \); then the entropy of this random variable is define as \( H(\pi_1, \ldots, \pi_M) = -\sum_{m=1}^M \pi_m \log \pi_m \). As a simple example, for Bernoulli random variable (which can represent, e.g., the outcome for flipping a coin with bias \( p \)), the entropy function is \( H(p) = -p \log p - (1-p) \log (1-p) \), and it is a concave function achieving the maximum at \( p^* = 1/2 \), which means that the fair-coin has the maximum entropy; this is intuitive as indeed the outcome of a fair coin toss is the most difficult to predict. Now we use this entropy to define the “uncertainty” associated with each training points. With a little abuse of notation, define \( H(\hat{\omega}) := H(\pi_1(\hat{\omega}), \ldots, \pi_M(\hat{\omega})) \). Denote the minimal and maximal entropy of all the training points as \( H_{\text{min}} = \min \{ H(\hat{\omega}), \hat{\omega} \in \hat{\Omega} \}, H_{\text{max}} = \max \{ H(\hat{\omega}), \hat{\omega} \in \hat{\Omega} \} \), respectively. Define the \( \tau \)-truncated training set as \( \hat{\Omega}_\tau = \{ \hat{\omega} \in \hat{\Omega} : (H(\hat{\omega}) - H_{\text{min}})/(H_{\text{max}} - H_{\text{min}}) \geq \tau \}, \forall \tau \in [0, 1] \). The truncated Dr. k-NN is obtained similarly as Step 2 of Dr. k-NN by restricting the training set \( \hat{\Omega}_\tau \) only to the samples in \( \hat{\Omega} \) (samples with larger entropy). Figure 5 reveals that the most informative samples usually lie in between categories. We can see that a truncated Dr. k-NN classifier with \( \tau = 0.9 \) only uses 20\% samples with little performance loss. More experimental details will be presented in Section 6.

6 Experiments

In this section, we describe the results of few-shot experiments using small subsets of MNIST [20], CIFAR-10 [13], and Lung Cancer data set [10].

7
Truncated Dr. k-NN with an approach using kernel-smoothing of the LFDs (in contrast to using benchmark. We compare our method including results using only 20% training samples (Dr.k-NN) terms of the average test accuracy on all data sets. The truncated regions with \( \tau = 0.9 \), (b) shows \( \tau \)-truncated Dr. k-NN with \( \tau = 0.9 \), (c) shows the decision made by the truncated Dr. k-NN with \( \tau = 0.9 \). The depth of the shaded area shows the level of samples entropy.

Benchmark. We compare our method including Dr. k-NN and its truncated version with the following baselines: (1) k-NN based method with different dimension reduction techniques, including Principal Component Analysis (PCA+k-NN), Singular Value Decomposition (SVD+k-NN), and Neighbourhood Components Analysis (NCA+k-NN) [16], (2) matching networks [33]; (3) prototypical networks [31]. To make these methods comparable, we adopt the same naive neural network with a single CNN layer on matching network, prototypical network, and our model, respectively, where the kernel size is 3, the stride is 1 and the width of the output layer is \( d = 400 \). Here we also compare with an approach using kernel-smoothing of the LFDs (in contrast to using k-NN) for performing classification. Consider a Gaussian kernel \( \kappa(x) = |H_\kappa|^{-1/2} \kappa(H_\kappa^{-1/2} x) \), where \( H_\kappa = \kappa I \) is the isotropical kernel with bandwidth \( h \). Then we replace \( \bar{p}_m(\omega) \) in Step 2 of Dr. k-NN with the following

\[
\bar{p}_m(\omega) := \sum_{i=1}^{n} P^*_m(\omega^i) \kappa(\omega - \omega^i), \quad m = 1, \ldots, M, \quad \forall \omega \in \Omega.
\]

Experiment set-up. In our experiments, we focus on an \( M \)-way \( K \)-shot learning task. To generate the training data set, we randomly select \( M \) classes and for each class we take \( K \) random samples. So our training data set contains \( MK \) samples overall. We then aim to classify a disjoint batch of unseen samples into one of these \( M \) classes. Thus random performance on this task stands at \( 1/M \). We test the average performance of different methods using 1,000 unseen samples from the \( M \) classes. To obtain reliable results, we repeat each test 10 times and calculate the average accuracy. We use the Euclidean distance \( \epsilon(\omega, \omega') = ||\omega - \omega'||_2 \) throughout our experiment.

| Methods                  | MNIST       | CIFAR-10   | Lung Cancer |
|--------------------------|-------------|------------|-------------|
|                           | Two-ways    | Five-ways  | Two-ways    | Five-ways  | Three-ways |
|                           | 5-shot      | 10-shot    | 5-shot      | 10-shot    | 5-shot     |
| PCA+k-NN                 | 0.801       | 0.872      | 0.614       | 0.678      | 0.687      |
| SVD+k-NN                 | 0.749       | 0.790      | 0.524       | 0.567      | 0.680      |
| NCA+k-NN                 | 0.602       | 0.640      | 0.340       | 0.355      | 0.597      |
| Matching Net             | 0.732       | 0.830      | 0.625       | 0.732      | 0.632      |
| Prototypical Net         | 0.742       | 0.842      | 0.671       | 0.759      | 0.651      |
| LFDs with Kernel Smoothing | 0.777       | 0.873      | 0.559       | 0.579      | 0.642      |
| Dr.k-NN                  | 0.838       | 0.959      | 0.746       | 0.831      | 0.707      |
| Truncated Dr.k-NN        | 0.815       | 0.926      | 0.742       | 0.825      | 0.703      |

Results. We present the average test accuracy in Table 1 for the unseen samples with different \( M = 2, 5 \) and \( K = 5, 10 \) on small subsets of MNIST and CIFAR-10, and with \( M = 3 \) and \( K = 5, 8 \) on lung cancer data. Note that random performance for two-ways and five-ways are 0.5 and 0.2, respectively. The figures in the table show that Dr. k-NN \((k = 5)\) outperforms other baselines in terms of the average test accuracy on all data sets. The truncated Dr. k-NN also yields competitive results using only 20% training samples \((\tau = 0.9)\), compared to standard Dr. k-NN.

\(^{1}\)There are only three categories and 33 samples in the lung cancer data.
Figure 6: A comparison of the learned feature spaces and the corresponding decision boundaries. There are 10 training samples from two categories of MNIST identified as large dots and 1,000 query samples identified as small dots. The color of dots shows their true categories. The color of the region shows the decisions made by corresponding methods.

To visualize that the proposed learning framework will affect the distribution of hidden representation of data points, we show the training and query samples in a 2D feature space as well as the corresponding decision boundary in Figure 6. It turns out our framework finds a better feature representation in the 2D space with smooth decision boundary and reasonable decision confidence map (indicated by the color depth in Figure 6(a)).

It is worth pointing out that the experimental results show that it is important to use $k$-NN in our proposed algorithm, which significantly outperforms the parallel version using kernel smoothing (even after we optimize the kernel bandwidth). We find that the performance when using kernel smoothing heavily depends on selecting an appropriate kernel bandwidth $h$ as illustrated by Figure 8 in Appendix A. Moreover, the best kernel bandwidth may vary from one dataset to another. Therefore, the cross-validation is required to be carried out to find the best kernel bandwidth in practice, which is quite time-consuming. In contrast, choosing the hyper-parameter $k$ is an easy task, since we only have very limited choices of $k$ in few-shot learning problems and the performance of Dr. $k$-NN is insensitive to the choices of $k$ (see Appendix A Figure 7). More pictures and discussions can be found in Appendix A.
References

[1] Soroosh Shafieezadeh Abadeh, Peyman Mohajerin Mohajerin Esfahani, and Daniel Kuhn. Distributionally robust logistic regression. In Advances in Neural Information Processing Systems, pages 1576–1584, 2015.

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In Advances in Neural Information Processing Systems, pages 9558–9570, 2019.

[3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 136–145. JMLR. org, 2017.

[4] Guilherme Aresta, Teresa Araújo, Scotty Kwok, Sai Saketh Chennamsetty, Mohammed Safwan, Varghese Alex, Bahram Marami, Marcel Prastawa, Monica Chan, Michael Donovan, Gerardo Fernandez, Jack Zeineh, Matthias Kohl, Christoph Walz, Florian Ludwig, Stefan Braunewell, Maximilian Baust, Quoc Dang Vu, Minh Nguyen Nhat To, Eal Kim, Jin Tae Kwak, Sameh Galal, Veronica Sanchez-Freira, Nadia Brancati, Maria Frucci, Daniel Riccio, Yaqi Wang, Lingling Sun, Kaiqiang Ma, Jiannan Fang, Ismael Kone, Lahsen Boulmene, Aurélio Campilho, Catarina Eloy, António Polónia, and Paulo Aguiar. Bach: Grand challenge on breast cancer histology images. Medical Image Analysis, 56:122 – 139, 2019.

[5] Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems 29, pages 523–531. Curran Associates, Inc., 2016.

[6] Jose Blanchet, Yang Kang, and Karthyek Murthy. Robust wasserstein profile inference and applications to machine learning. Journal of Applied Probability, 56(3):830–857, 2019.

[7] Jose Blanchet and Karthyek Murthy. Quantifying distributional model risk via optimal transport. Mathematics of Operations Research, 44(2):565–600, 2019.

[8] Ruidi Chen and Ioannis Paschalidis. Selecting optimal decisions via distributionally robust nearest-neighbor regression. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 749–759. Curran Associates, Inc., 2019.

[9] Thomas M. Cover and Joy A. Thomas. Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience, USA, 2006.

[10] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[11] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. Mathematical Programming, 171(1-2):115–166, 2018.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[13] Rui Gao, Xi Chen, and Anton J Kleywegt. Wasserstein distributional robustness and regularization in statistical learning. arXiv preprint arXiv:1712.06050, 2017.

[14] Rui Gao and Anton J Kleywegt. Distributionally robust stochastic optimization with wasserstein distance. arXiv preprint arXiv:1604.02199, 2016.

[15] Rui Gao, Liyan Xie, Yao Xie, and Huan Xu. Robust hypothesis testing using wasserstein uncertainty sets. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS ’18, page 7913–7923, Red Hook, NY, USA, 2018. Curran Associates Inc.
[16] Jacob Goldberger, Geoffrey E Hinton, Sam T. Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, Advances in Neural Information Processing Systems 17, pages 513–520. MIT Press, 2005.

[17] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In ICML deep learning workshop, volume 2. Lille, 2015.

[18] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[19] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In Operations Research & Management Science in the Age of Analytics, pages 130–166. INFORMS, 2019.

[20] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[21] Fei-Fei Li, R. Fergus, and P. Perona. One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(4):594–611, 2006.

[22] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18, page 1095–1106, Red Hook, NY, USA, 2018. Curran Associates Inc.

[23] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[24] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In Marina Meila and Xiaotong Shen, editors, Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, volume 2 of Proceedings of Machine Learning Research, pages 412–419, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.

[25] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. One-shot learning with a hierarchical nonparametric bayesian model. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, Proceedings of ICML Workshop on Unsupervised and Transfer Learning, volume 27 of Proceedings of Machine Learning Research, pages 195–206, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[26] Richard J Samworth et al. Optimal weighted nearest neighbour classifiers. The Annals of Statistics, 40(5):2733–2763, 2012.

[27] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.

[28] Soroosh Shafieezadeh-Abadeh, Daniel Kuhn, and Peyman Mohajerin Esfahani. Regularization via mass transportation. Journal of Machine Learning Research, 20(103):1–68, 2019.

[29] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. Lectures on stochastic programming: modeling and theory. SIAM, 2014.

[30] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. arXiv preprint arXiv:1710.10571, 2017.

[31] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4077–4087. Curran Associates, Inc., 2017.
[32] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1199–1208, 2018.

[33] Kevin D. Tang, Marshall F. Tappen, Rahul Sukthankar, and Christoph H. Lampert. Optimizing one-shot recognition with micro-set learning. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3027–3034, 2010.

[34] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16, page 3637–3645, Red Hook, NY, USA, 2016. Curran Associates Inc.

[35] Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning, 2019.

[36] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In Advances in Neural Information Processing Systems, pages 2365–2374, 2018.
A Comparison to Kernel Smoothing

More visualization for comparing with kernel smoothing.

Figure 7: Dr. k-NN with different $k$.

Figure 8: Kernel smoothing with different bandwidth $h$

The performance of Dr. k-NN is insensitive to the choice of the number of selected neighbors $k$ as shown in Figure 7. In contrast, the performance using kernel smoothing method defined in (5) is heavily depended on selecting an appropriate kernel bandwidth $h$ as illustrated in Figure 8. Note that, to apply the method to different data sets, the best kernel bandwidth may vary from case to case. Therefore, the cross-validation is required to be carried out to find the best kernel bandwidth in practice, which is extremely time-consuming.
B Proof for Section 3

Proof of Lemma 1: We here prove a more general result for an arbitrary sample space $\Omega$. Note that each $P_m, 1 \leq m \leq M$, is absolutely continuous with respect to $P_1 + \cdots + P_M$, hence the Radon-Nikodym derivative $\frac{dP_m}{d(P_1 + \cdots + P_M)}$ exists. Using interchangeability principle [29], we have

$$
\min_{\pi \in \Delta_M} \Psi(\pi; P_1, \ldots, P_M) = \min_{\pi: \Omega \rightarrow \Delta_M} \int_{\Omega} \left[ \sum_{m=1}^{M} (1 - \pi_m(\omega)) \frac{dP_m}{d(P_1 + \cdots + P_M)}(\omega) \right] d(P_1 + \cdots + P_M)
$$

$$
= \int_{\Omega} \min_{\pi \in \Delta_M} \left[ \sum_{m=1}^{M} (1 - \pi_m(\omega)) \frac{dP_m}{d(P_1 + \cdots + P_M)}(\omega) \right] d(P_1 + \cdots + P_M)
$$

$$
= \int_{\Omega} \left[ 1 - \max_{1 \leq m \leq M} \frac{dP_m}{d(P_1 + \cdots + P_M)}(\omega) \right] d(P_1 + \cdots + P_M).
$$

where the third equality holds because for any $\omega$, the inner minimization attains its minimum at one of the vertices of $\Delta_M$.

Proof of Lemma 2: By the definition of uncertainty sets in [3], we can introduce additional variables $\gamma_m \in \mathbb{R}_{\pi}^{n \times n}$ which represents the joint distribution with marginals $P_m \in \mathcal{P}_m$ and $\hat{P}_m$. The constraints $\mathcal{W}(P, \hat{P}_m) \leq \vartheta_m$ in [3] can be rewritten using $\gamma_m$ as

$$
\sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_{ij} c(\hat{\omega}^i, \hat{\omega}^j) \leq \vartheta_m.
$$

Furthermore, the marginal distribution constraint of $\gamma_m$ reads

$$
\sum_{i=1}^{n} \gamma_{ij} = \hat{P}_m(\hat{\omega}^j), \quad \sum_{j=1}^{n} \gamma_{ij} = P_m(\hat{\omega}^i), \quad m = 1, \ldots, M.
$$

Thereby the problem $\max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \Psi(P_1, \ldots, P_M)$ is equivalent to the convex optimization formulation in [1].

Proof to Theorem 1: By Lemmas 1 and 2 we have

$$
\max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \min_{\pi: \Omega \rightarrow \Delta_M} \Psi(\pi; P_1, \ldots, P_M) = \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \Psi(\pi; P_1, \ldots, P_M) = \mathbb{1}.
$$

Moreover, we identify $\pi$ as $\{\pi^1, \ldots, \pi^n\}$, where $\pi^i \in \mathbb{R}^M$ satisfies $\sum_{m=1}^{M} \pi^i_m = 1$. Similar to the proof of Lemma 2 $P_m, 1 \leq m \leq M$, can also be identified as a vector in $\mathbb{R}^n$. Note that the objective function $\Psi(\pi; P_1, \ldots, P_M)$ is linear in $\{\pi^1, \ldots, \pi^n\}$ and concave in $\{P_1, \ldots, P_M\}$, and the Slater condition holds. Hence applying convex programming duality we can exchange max and min and thus the result follows.

Proof of Theorem 2: On the one hand, since $\pi_m^{knm}$ can be regarded as a special case of the general classifier $\pi: \Omega \rightarrow \Delta_M$, it holds that

$$
\min_{Q_1, \ldots, Q_M \in \mathcal{P}(\hat{\Omega})} \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \sum_{m=1}^{M} \mathbb{E}_{\omega_m \sim P_m} [1 - \pi_m^{knm}(\omega_m; k, Q)]
$$

$$
\geq \min_{\pi: \Omega \rightarrow \Delta_M} \max_{P_m \in \mathcal{P}_m, 1 \leq m \leq M} \sum_{m=1}^{M} \mathbb{E}_{\omega_m \sim P_m} [1 - \pi_m(\omega_m)].
$$

On the other hand, by Lemma 2 there exists an optimal solution to the minimax problem (2), denoted as $\{P^*_1, \ldots, P^*_M\}$, and the classifier $\pi^*$ as given in Lemma 1. Note that there exists $1 \leq k^* \leq n$ and weights $Q_1^*, \ldots, Q_M^* \in \mathcal{P}(\Omega)$ such that

$$
\pi_m^{knm}(\omega; k^*, Q^*) = \pi^*(\omega), \forall \omega \in \hat{\Omega},
$$

where the third equality holds because for any $\omega$, the inner minimization attains its minimum at one of the vertices of $\Delta_M$. 

\[ \square \]
for example, by taking $k^* = 1$ and $Q_m^* = P_m^*$, $\forall 1 \leq m \leq M$. This implies that

$$\min_{Q_1, \ldots, Q_M \in \mathcal{P} \left( \hat{\Omega} \right)} \max_{1 \leq k \leq K} \sum_{m=1}^{M} E_{\omega_m \sim P_m} \left[ 1 - \pi_{m}^{knn}(\omega_m; k, Q) \right]$$

$$\leq \max_{P_m \in P_m, 1 \leq m \leq M} \sum_{m=1}^{M} E_{\omega_m \sim P_m} \left[ 1 - \pi_{m}^{knn}(\omega_m; k^*, Q^*) \right]$$

$$= \max_{P_m \in P_m, 1 \leq m \leq M} \sum_{m=1}^{M} E_{\omega_m \sim P_m} \left[ 1 - \pi_{m}^*(\omega_m) \right]$$

$$= \min_{\pi: \hat{\Omega} \rightarrow \Delta_M} \max_{P_m \in P_m, 1 \leq m \leq M} \sum_{m=1}^{M} E_{\omega_m \sim P_m} \left[ 1 - \pi_{m}^*(\omega_m) \right].$$

Thereby we have shown the equivalence between (4) and (2).