Distributionally Robust Weighted k-Nearest Neighbors

Shixiang (Woody) Zhu[1], Liyan Xie[2], Minghe Zhang[3], Rui Gao[4], Yao Xie[3]

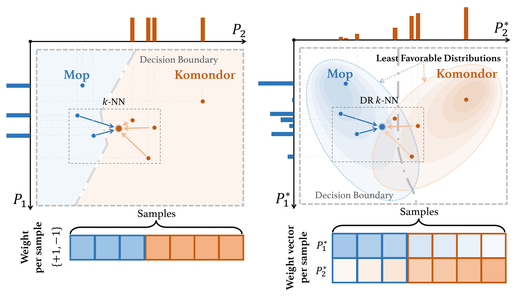
Carnegie Mellon University[1], The Chinese University of Hong Kong, Shenzhen[2],
Georgia Institute of Technology[3], University of Texas at Austin[4]

Introduction

Learning a robust classifier from a few samples remains a key challenge in machine learning. A major thrust of research has been focused on developing k-nearest neighbor (k-NN) based algorithms combined with metric learning that captures similarities between samples. When the samples are limited, robustness is especially crucial to ensure the generalization capability of the classifier. In this paper, we study a minimax distributionally robust formulation of weighted k-nearest neighbors, which aims to find the optimal weighted k-NN classifiers that hedge against feature uncertainties. We develop an algorithm, `Dr.k-NN`, that efficiently solves this functional optimization problem and features in assigning minimax optimal weights to training samples when performing classification. These weights are class-dependent, and are determined by the similarities of sample features under the least favorable scenarios. The proposed framework can be shown to be equivalent to a Lipschitz norm regularization problem. We also couple our framework with neural-network-based feature embedding. We demonstrate the competitive performance of our algorithm compared to the state-of-the-art in the few-training-sample setting with various real-data experiments.

Motivating Example

Figure: Motivating example: a small training set of three image-label pairs: the first image is a mop; the second image is a dog; the third image looks like a mop but is, in fact, a dog (dressing up as a mop).



(a) Vanilla k-NN (b) `Dr.k-NN`

Figure: An illustrative comparison of `Dr.k-NN` and vanilla k-NN.

Key Concepts

Setup:

- Training samples: $\{(x^1, y^1), \ldots, (x^n, y^n)\}$;
- Empirical distributions: $\widehat{P}_m := \frac{1}{|\{i:y^i=m\}|} \sum_{i=1}^n \delta_{\xi^i} \mathbb{I}\{y^i = m\}$;

Randomized classifier π:

$\pi : \Xi \to \Delta_M$ assigns class $m \in \{1, \ldots, M\}$ with probability $\pi_m(\xi)$ to a query feature vector $\xi \in \Xi$, where Δ_M is the probabilistic simplex $\Delta_M = \{\pi \in \mathbb{R}_+^M : \sum_{m=1}^M \pi_m = 1\}$.

Uncertainty Set $\mathcal{P}_m$:

$$\mathcal{P}_m := \left\{P_m \in \mathscr{P}(\widehat{\Xi}) : \mathcal{W}(P_m, \widehat{P}_m) \le \vartheta_m\right\}, \tag{1}$$

- $\mathscr{P}(\widehat{\Xi})$ denotes the set of all probability distributions on $\widehat{\Xi}$;
- $\vartheta_m \ge 0$ specifies the size of the uncertainty set for the m-th class that specifies the amount of deviation we would like to control;
- $\mathcal{W}(P, P') := \min_\gamma \ \mathbb{E}_{(\xi,\xi')\sim\gamma}[c(\xi, \xi')]$.

Overview of End-to-end Architecture

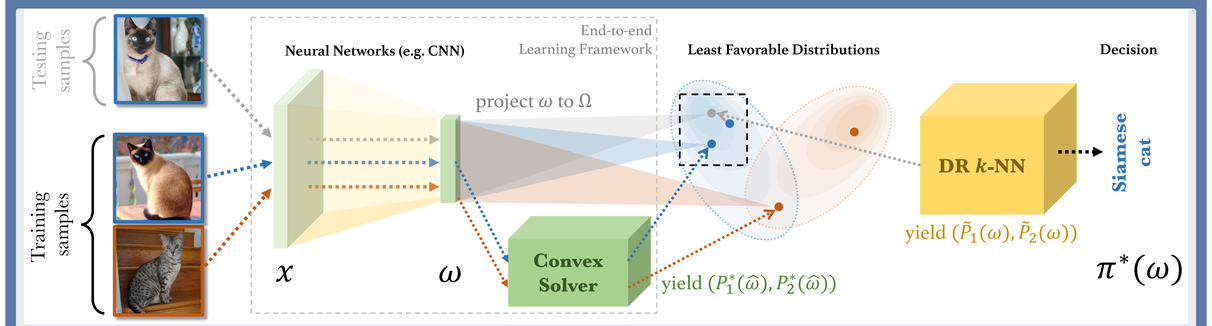


Figure: (1) an architecture that is able to produce feature embedding ξ and least favorable distributions P_m^* for training set; (2) an `Dr.k-NN` makes decisions for any unseen sample ξ based on the estimated weight vector $\widetilde{p}_m(\xi)$ (probability mass on least favorable distributions).

Distributionally robust k-NN

Minimax Robust Classification Problem:

$$\min_{\pi:\Xi\to\Delta_M} \ \max_{P_m\in\mathcal{P}_m, 1\le m\le M} \ \Psi(\pi; P_1, \ldots, P_M). \tag{2}$$

Suppose the features in each class m follows a distribution P_m. We define the *risk* of a classifier π as the total error probabilities:

$$\Psi(\pi; P_1, \ldots, P_M) := \sum_{m=1}^M \mathbb{E}_{\xi\sim P_m}[1 - \pi_m(\xi)].$$

Least Favorable Distributions (LFD):

The optimal solution $P_1^*, \ldots, P_M^*$ to the inner maximization problem is also called *least favorable distributions (LFD)*.

Theorem 1: Finite-dimensional Convex Programming Reformulation

For the uncertainty sets defined in (1), the least favorable distribution of problem (2) can be obtained by solving the following problem:

$$\begin{aligned} \min_{\substack{p_1,\ldots,p_M\in\mathbb{R}_+^n \\ \gamma_1,\ldots,\gamma_M\in\mathbb{R}_+^{n\times n}}} \ & \sum_{i=1}^n \max_{1\le m\le M} p_m^i \\ \text{subject to} \ & \sum_{i=1}^n \sum_{j=1}^n \gamma_m^{i,j} c(\xi^i, \xi^j) \le \vartheta_m, \\ & \sum_{i=1}^n \gamma_m^{i,j} = \widehat{P}_m(\xi^j), \quad \sum_{j=1}^n \gamma_m^{i,j} = p_m^i, \\ & \forall 1 \le i, j \le N, \ 1 \le m \le M. \end{aligned} \tag{3}$$

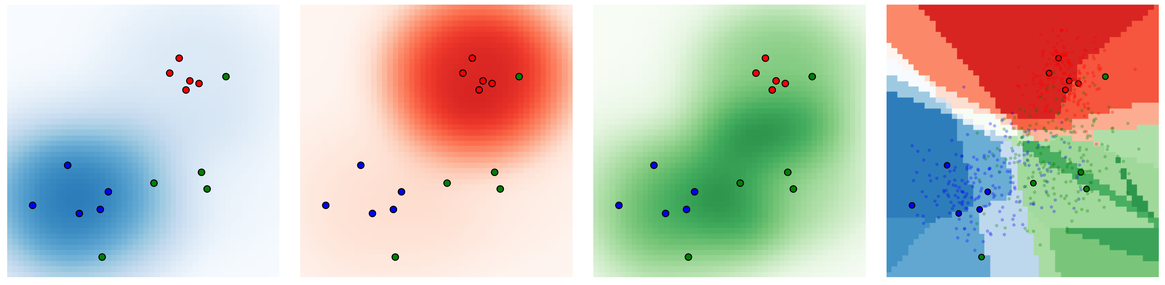
Joint Learning Framework

The objective is $\min_\theta J(\theta; P_1^*, \ldots, P_M^*) := \sum_{i=1}^n \max_{1\le m\le M} P_m^*(\phi(x^i; \theta))$,

- $\{P_m^*(\phi(\cdot;\theta))\}_{1\le m\le M}$ are the LFDs generated by the convex solver (3) given input variables $\{\xi^i = \phi(x^i;\theta)\}_{1\le i\le n}$;
- Feature mapping $\phi : \mathcal{X} \to \Xi$ is a neural network parameterized by θ;
- Optimization layer packs the convex problem (3) as an output layer generating LFDs of (2) by *differentiable optimization* (Agrawal et al.).

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

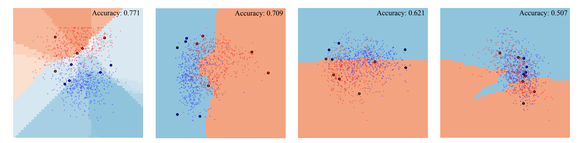
Input: $S_m := \{(x^i, y^i) : y^i = m, \forall i\} \subset S, \ m = 1, \ldots, M$;
Output: The feature mapping $\phi(\cdot;\theta)$ and the LFD $P_1^*, \ldots, P_M^*$;
Initialization: $t = 0$; $n' < n$ is the size of "mini-set";
while $t < T$ **do**
  for *number of mini-sets* **do**
    Randomly generate M integers $n_1, \ldots, n_M$ such that $\sum_{m=1}^M n_m = n', n_m > 0, \forall m$;
    Initialize two ordered sets $\widehat{\Xi} = \emptyset, \widehat{P} = \emptyset$;
    for $m \in \{1, \ldots, M\}$ **do**
      $\mathcal{X}_m \leftarrow$ Randomly sample n_m points from S_m;
      $\widehat{\Xi}_m \leftarrow \{\xi := \phi(x;\theta_t) : x \in \mathcal{X}_m\}$;
      $\widehat{P}_m \leftarrow \frac{1}{n_m}\sum_{i=1}^{n_m} \delta_{\xi_m^i}, \ \xi_m^i \in \widehat{\Xi}_m$;
      $\widehat{\Xi} \leftarrow \widehat{\Xi} \cup \widehat{\Xi}_m$; $\widehat{P} \leftarrow \widehat{P} \cup \widehat{P}_m$;
    end
    Update the LFDs $P_1^*, \ldots, P_M^*$ on $\widehat{\Xi}$ by solving (3) given $\widehat{\Xi}, \widehat{P}$;
  end
  $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla J(\theta_t; P_1^*, \ldots, P_M^*)$, where α is the learning rate;
  $t \leftarrow t + 1$;
end



(a) P_1^* (b) P_2^* (c) P_3^* (d) π^*

Figure: An example of the weights P_1^*, P_2^*, P_3^* yielding from (3) and the corresponding results of `Dr.k-NN` using a small subset of MNIST (digit 4 (red), 6 (blue), 9 (green) and $k = 5$). Raw samples are projected on a 2D feature space ($d = 2$), with the color indicating their true class-membership.

Numerical Results



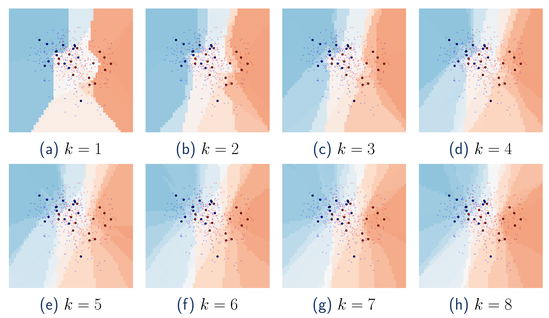
(a) `Dr.k-NN` (b) PCA + k-NN (c) SVD + k-NN (d) NCA.

Figure: 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.

Table: Comparison of classification accuracy in the few-training-sample setting

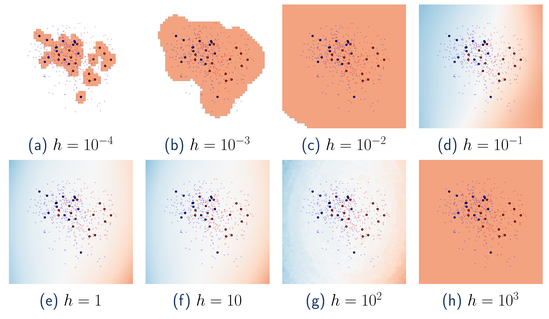
Methods	MNIST				mini ImageNet				CIFAR-10				Omniglot				Lung Cancer		COVID-19 CT	
	$M=2$		$M=5$		$M=2$		$M=5$		$M=2$		$M=5$		$M=2$		$M=5$		$M=3$		$M=2$	
	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=8$	$K=5$	$K=10$
PCA+k-NN	0.801	0.872	0.614	0.678	0.578	0.667	0.268	0.277	0.687	0.711	0.262	0.270	0.597	0.638	0.309	0.358	0.617	0.647	0.658	0.719
SVD+k-NN	0.749	0.790	0.524	0.567	0.587	0.675	0.268	0.283	0.680	0.701	0.259	0.266	0.591	0.618	0.305	0.413	0.624	0.648	0.646	0.715
NCA+k-NN	0.602	0.640	0.340	0.355	0.547	0.578	0.245	0.258	0.597	0.616	0.232	0.236	0.549	0.574	0.267	0.346	0.575	0.582	0.612	0.624
Matching Net	0.732	0.830	0.625	0.732	0.687	0.703	0.286	**0.360**	0.632	0.641	0.241	0.247	0.735	0.769	0.412	0.433	0.621	0.635	0.715	0.732
Prototypical Net	0.742	0.842	0.671	0.759	0.710	0.725	0.296	0.348	0.651	0.664	0.254	0.259	**0.769**	**0.836**	**0.448**	0.532	0.632	0.644	**0.729**	**0.744**
MetaOptNet	0.725	0.843	0.658	0.790	0.732	0.741	0.255	0.363	0.702	0.713	0.257	0.298	0.742	0.755	0.412	0.453	0.638	0.642	0.713	0.739
Feature embedding + k-NN	0.792	0.798	0.546	0.551	0.738	0.742	**0.490**	**0.486**	0.689	0.691	**0.492**	**0.494**	0.725	0.751	0.445	0.495	**0.664**	0.691	0.701	0.710
Kernel Smoothing	0.777	0.873	0.559	0.579	0.593	0.601	0.272	0.278	0.642	0.661	0.272	0.282	0.520	0.565	0.240	0.285	0.367	0.370	0.502	0.604
Truncated Dr.k-NN	**0.815**	**0.926**	**0.742**	**0.825**	**0.746**	**0.753**	0.295	0.340	**0.703**	**0.719**	0.297	0.305	0.755	0.825	0.425	**0.542**	0.652	**0.693**	0.722	0.741
Dr.k-NN	**0.838**	**0.959**	**0.746**	**0.831**	**0.752**	**0.786**	**0.306**	0.358	**0.707**	**0.728**	**0.309**	**0.311**	**0.765**	**0.850**	**0.465**	**0.580**	**0.667**	**0.704**	**0.734**	**0.752**

Comparison to Kernel Smoothing



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$
(e) $k = 5$ (f) $k = 6$ (g) $k = 7$ (h) $k = 8$

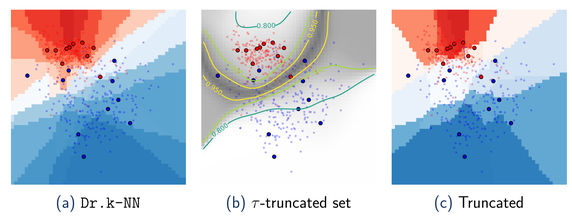
Figure: `Dr.k-NN` with different k.



(a) $h = 10^{-4}$ (b) $h = 10^{-3}$ (c) $h = 10^{-2}$ (d) $h = 10^{-1}$
(e) $h = 1$ (f) $h = 10$ (g) $h = 10^2$ (h) $h = 10^3$

Figure: Kernel smoothing with different bandwidth h.

Truncated Dr.k-NN for Large-scale Data



(a) `Dr.k-NN` (b) τ-truncated set (c) Truncated

Figure: An example of the truncated `Dr.k-NN` using MNIST (digit 4 (red) and 9 (blue)). Big dots between the lines are selected training samples under different τ. The depth of the shaded area shows the level of samples entropy.