Prototype selection for nearest neighbor
Latent KMeans Prototype Selection

Xinyuan Wang
UC San Diego
xiw136@ucsd.edu

Abstract

Choosing a representative subset of "prototypes" from the training set is crucial for accelerating nearest neighbor classifiers. This project proposes projecting the data into a latent space using a pretrained embedder, applying in-class KMeans Clustering to the latent representations and taking the clustering centers as the selected prototypes. Experiments demonstrate that the performance of the nearest neighbor classifier with latent KMeans clustering prototypes significantly outperforms other baselines, even with a training set containing as few as 10 points. An ablation study on the embedder's pretraining domain reveals taht using in-domain data for training greatly affects the quality of the representations.

1 Description

The performance of nearest neighbor classifier is heavily dependent on the distribution of the training data. Original data, such as image data, often contains a large amount of high-frequency noise, which can impair the effectiveness of the classifier. The perceptual distance in the latent space has been shown to offer advantages over per-pixel distance in the original data space (Johnson et al., 2016), so it is beneficial to project the data into a meaningful latent space using a pretrained embedder. However, regardless of the space in which the data resides, there can be multiple outliers in each class that would directly affect the nearest neighbor classifier's accuracy. Therefore, this project proposes using the centers derived from latent KMeans clustering as the training data for the nearest neighbor classifier. First, the entire dataset is projected into the latent space using an embedder built from a pretrained classifier. Then, for each class, KMeans Clustering is applied to represent the data points with M/N clustering centers (M is the size of the sub-training set. N is the class number of the classification task). The clustering centers of all classes compose the final training set of the nearest neighbor classifier.

2 Pseudocode

Prototype Selection Algorithm 1 calculates the KMeans Clustering centers on each class' embedding data. The original training set is X with N classes. E is the pretrained embedder. S_i is the sub-training set whose label is i. The new training set size is M. C_i is the clustering centers of S_i.

Algorithm 1 Prototype Selection

Input: Training set X, embedder E, size M, class number N

$S \leftarrow \emptyset$
$X \leftarrow E(X)$ ▷ Project X to the latent space.
for $i \leftarrow 1$ to $\mathcal{N}$ **do**
    $S_i \leftarrow \text{GETCLASSDATA}(X, i)$ ▷ Get the data points whose class label is i.
    $C_i \leftarrow \text{KMEANS}(S_i, M//N)$ ▷ Calculate KMeans Clustering centers for this class.
    $S \leftarrow S \cup S_i$ ▷ Add class centers to the set
end for
return S

3 Experiment

Dataset. MNIST dataset is an image classification dataset of 10 handwritten digits. The includes 60000 training samples and 10000 testing samples, which are all 28×28 grayscale image (768 dimensions in total).

Baselines. The baseline strategy of prototype selection are random sampling and KMeans clustering on the original dataset. Random sampling method randomly selects a subset of points from the training set. KMeans clustering method applies

KMeans on the data of each class and takes the clustering centers as the prototypes.

Implementation details. The experiments are conducted with 5 different values of M: 10, 500, 1000, 5000, 10000. The nearest neighbor classifier is implemented using sklearn.neighbors.KNeighborsClassifier with a brute-force search and Euclidean distance. Due to the randomness of sampling, each experiment is repeated 5 times with different random seeds. The 95% error margin and t-confidence intervals can be calculated using the formula 1 and formula 2, where $\bar{x}$ is the mean of accuracy, s is the sample standard deviation, n is the number of iterations, and t is the t-Value. For simplicity, only error margin is recorded. The embedder is constructed from a ResNet-18 (He et al., 2016) pretrained on MNIST by removing the last fully-connected layer. The embedding dimension is 512. In the ablation study of embedder's pretraining domain, ResNet-18 pretrained on ImageNet1k (Deng et al., 2009) is used as embedder for comparison.

$$ErrorMargin = t \times \frac{s}{\sqrt{n}} \tag{1}$$

$$ConfidenceInterval = \bar{x} \pm ErrorMargin \tag{2}$$

4 Results

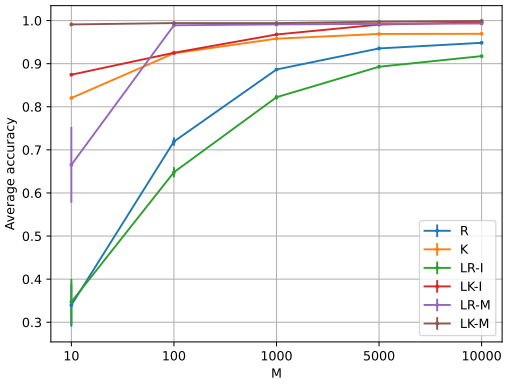

Figure 1: Average accuracy and error bars.

The basic prototype selection methods includes random selection and KMeans clustering method. They are applied across three different spaces: the original data space, the latent space of ImageNet1k pretrained embedder, and the latent space

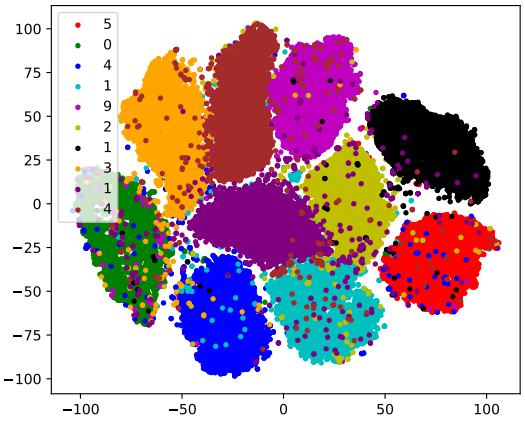

Figure 2: T-SNE visualization of the original data distribution.

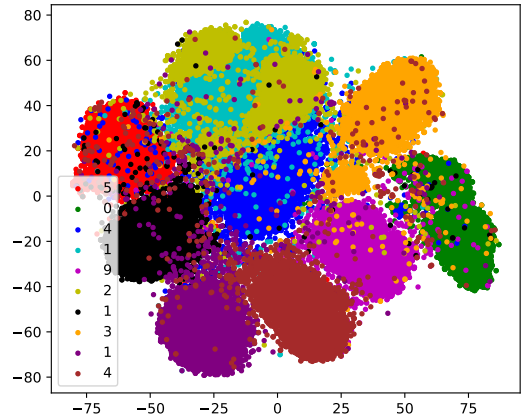

Figure 3: T-SNE visualization of the latent data distribution using ImageNet1k embedder.

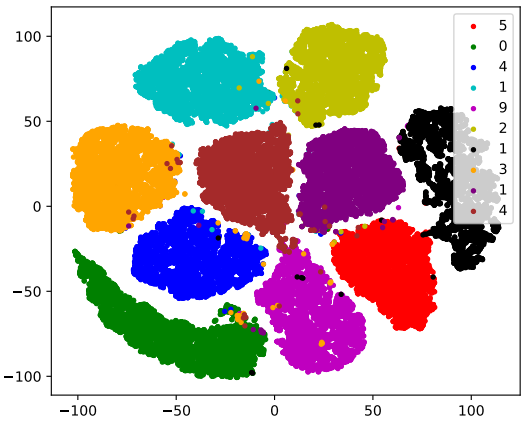

Figure 4: T-SNE visualization of the latent data distribution using MNIST embedder.

Table 1: Average accuracy of different methods.

Method	M=10	M=100	M=1000	M=5000	M=10000	Avg.
R	0.340	0.719	0.886	0.935	0.948	0.766
K	0.820	0.924	0.958	0.969	0.969	0.928
LR-I	0.347	0.648	0.822	0.893	0.918	0.726
LK-I	0.874	0.925	0.968	0.991	0.996	0.951
LR-M	0.665	0.989	0.991	0.992	0.993	0.926
LK-M	**0.991**	**0.994**	**0.994**	**0.997**	**0.999**	**0.995**

Table 2: Standard Deviation and Confidence Interval Half Widths

Method	M=10		M=100		M=1000		M=5000		M=10000		Average	
	Std	EM	Std	EM	Std	EM	Std	EM	Std	EM	Avg. Std	Avg. EM
R	4.7e-02	5.0e-02	9.0e-03	9.6e-03	2.1e-03	2.3e-03	2.4e-03	2.6e-03	2.9e-03	3.1e-03	1.3e-02	1.4e-02
K	1.1e-16	0.0e+00	2.1e-03	2.3e-03	3.5e-04	3.8e-04	4.4e-04	4.7e-04	5.7e-04	6.1e-04	**7.0e-04**	**7.4e-04**
LR-a	5.0e-02	5.3e-02	1.1e-02	1.2e-02	5.0e-03	5.4e-03	1.6e-03	1.7e-03	2.5e-03	2.7e-03	1.4e-02	1.5e-02
LK-a	0.0e+00	0.0e+00	1.0e-03	1.1e-03	2.2e-04	2.3e-04	1.8e-04	1.9e-04	1.6e-04	1.7e-04	**3.2e-04**	**3.4e-04**
LR-b	8.3e-02	8.8e-02	9.8e-04	1.0e-03	1.0e-03	1.1e-03	2.8e-04	3.0e-04	2.9e-04	3.1e-04	1.7e-02	1.8e-02
LK-b	0.0e+00	0.0e+00	1.1e-04	1.2e-04	2.2e-04	2.3e-04	2.6e-04	2.8e-04	7.8e-05	8.0e-05	**1.3e-04**	**1.4e-04**

of MNIST pretrained embedder. There are 6 experiments in total: Random selection (R), KMeans clustering (K), Latent random selection (ImageNet1k) (LR-I), Latent KMeans clustering (ImageNet1k) (LK-I), Latent random selection (MNIST) (LR-M), and Latent KMeans clustering (MNIST) (LK-M). Each experiment is conducted with M values of $10, 100, 1000, 5000, 10000$, repeated 5 times using different random seeds. Figure 1 plots the average accuracy and error bars for all the methods across all the M values. The detailed accuracy, standard deviation (Std), and error margin (EM) are presented in Table 1 and Table 2. Since the confidence interval can be calculated by the mean and error margin, Table 2 only shows the error margin for each experiment. The variances of some of the accuracy measures are very small, so their error margins are approximated to 0.

Evaluation. It can be observed in Figure 1 that the latent KMeans clustering using MNIST embedder method (LK-M) beats or matches all other methods with nearly 100% accuracy across all experimented M values. Latent Random Selection with MNIST (LR-M) also outperforms random selection and KMeans baseline when $M > 10$. Both of them demonstrate the superiority of the proposed method. In general, larger M yields better classification performance. All the methods have a relatively decent performance when $M >= 1000$, but when $M = 10$ or 100, their performance drops significantly, except for LK-M remaining at 99% accuracy. The highly variable data distribution makes it difficult for random selection method to coincidentally pick the good representatives. However, the centering effect of KMeans methods can mitigate this problem, which explains they outperforms random selection methods when M is small. Figure 2 3 4 are the t-SNE (Van der Maaten and Hinton, 2008) visualizations of the original data, latent data using ImageNet1k embedder and latent data using MNIST embedder. We can see that after the projection using the MNIST embedder, the number of outliers significantly reduces and the boundaries between different classes are more clearer than the other two. This explains the advantage of using a meaningful latent space, which leads to a better classifier at last. Besides the classification performance, since the embedded data's dimension is reduced to 512, so the proposed method also needs 24% less computation than the baseline, leading to a faster classifier.

Ablation study of embedder's pretraining domain. The embedder projects the original data to a high-dimensional latent space to make the nearest neighbor distance metric more effective. This project reveals that the data on which the embedder is pretrained greatly affects the downstream nearest neighbor classifier's performance. LR-I and LK-I uses the ImageNet1k pretrained ResNet18, which means the embedder is pretrained using out-of-domain data. It can be observed in Figure 1 that LR-I's performance is even worse than baseline random selection, while LK-I outperforms KMeans baseline by a small margin. This indicates that

simply using an embedder without tuning on the target data may project the data into a meaningless distribution, detrimental for random selection. However, after centering using a clustering method, the distribution becomes more suitable for the distance metric with less variance and outliers. It can be observed in Table 2 that KMeans methods have smaller standard deviation and error margin than random selection methods. There can be a second explanation that the nearest neighbor and KMeans method both operate in the Euclidean space, so the preprocessing of KMeans re-project the latent data to a meaningful space for the classifier. However, when the embedder is directly pretrained on the in-domain data, even random selection benefits from the projection, leading to a 98.9% accuracy, and the average accuracy of Latent Random method (LRM) almost matches the KMeans baseline in Table 1. When applying the KMeans method, the average accuracy is boosted to 99.5%. Surprisingly, when $M = 10$, it still has a 99.1% accuracy, but even using the entire 60000 original data as training set only results in an accuracy of 96.9%, which means representing each class with a single point in the latent space can be more effective than using the whole training set in the original space!

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).