1. Idea for prototype selection

We can first do some preprocessing on the dataset. Next, extract some important data points from the whole dataset.

Idea:

1. Conduct PCA on the dataset to reduce the dimensionality.
2. Separate training data into 10 different clusters according to their labels
3. For each cluster, conduct K-means and extract all clusters’ centers as representatives
4. Combine all centers and form a group of condensed training data.
5. Pseudocode

|  |
| --- |
| Algorithm of creating subset of training data |
| **Input**: training dataset and labels, number M |
| **Output**: subset of size M as condensed training dataset |
| **procedure buildSubset**(*traingData*, *M*) |
| initialize 10 *buckets* to store data with 10 different labels |
| **for** each *d* **in** *trainingData*: |
| append this training sample *d* to corresponding *bucket* |
| **for** each bucket: |
| conduct k-means on samples in this bucket |
| cluster them into *M*/10 clusters |
| *trainingSubset* ← cluster centers |
| **return** *trainingSubset* |

1. Experimental results

Implementation details are as follows:

1. When conducting PCA, we need to **import PCA** from *sklearn.decomposition*. Here we condense the dimension from 784 to 100.
2. We need to **import KMeans** from *sklearn.cluster* so as to create subset of training dataset. We can set *init = ‘random’* such that the initialization would be random.

Numerical results are shown below:

**Table 1: The correctness rate of uniform-random selection and prototype**

|  |  |  |  |
| --- | --- | --- | --- |
| M | 1000 | 5000 | 10000 |
| Uniform-random | 89.02% | 93.82% | 94.97% |
| Prototype | 95.6% | 96.76% | 97.12% |

**Table 2: Error bars for uniform-random selection (20 trials)**

|  |  |  |  |
| --- | --- | --- | --- |
| M | 1000 | 5000 | 10000 |
| Mean | 89.32% | 93.93% | 95.14% |
| Standard Deviation | 0.33% | 0.21% | 0.19% |
| Error Bars | 88.99% to 89.65% | 93.72% to 94.14% | 94.93% to 95.33% |
| Confidence interval (0.95) | 88.67% to 89.97% | 93.51% to 94.35% | 94.74% to 95.52% |

Where

Mean = , standard deviation

Error bars =

For the confidence interval when confidence ratio is 0.95:

**Table 3: Error bars for Prototype selection**

|  |  |  |  |
| --- | --- | --- | --- |
| M | 1000 | 5000 | 10000 |
| # of trials | 20 | 10 | 5 |
| Mean | 95.83% | 96.82% | 96.99% |
| Standard Deviation | 1.03% | 0.07% | 0.08% |
| Error Bars | 95.72% to 95.93% | 96.75% to 96.89% | 96.90% to 97.07% |
| Confidence Interval  (0.95) | 95.63% to 96.03% | 96.67% to 96.97% | 96.81% to 97.15% |

1. Critical evaluation

From the table above, we can see that this prototype improves the performance in terms of correctness rate compared to uniform-random selection significantly when M is relatively small (M = 1000).

To better improve this, we can do the following:

1. utilize k-NN instead of 1-NN
2. Consider another innovative method: Stochastic Neighbor Compression
3. implement another model instead of NN, for instance, auto encoder