8 Random Projection for Search Engines

In this project, you will need to implement random projection to accelerate nearest neighbor search. You will investigate to which extent the data dimension can be reduced with sacrificing the precision.

8.1 Data Set

MNIST[2] is a benchmark data set for classification. The full MNIST consists of 70,000 gray-scale images with size 28-by-28, each of which contains a digit from 0 to 9. In the experiments, we will use part of it for binary classification. We will also use the raw pixel as the feature vector $\boldsymbol{x}_i$. Therefore, the feature dimension $d = 28 \times 28 = 784$.

Download the "`mnist_568.mat`" in Canvas. This file is in Matlab format, and you will need to figure out a proper python API to load it. It stores four variables:

- $\boldsymbol{X}_{\text{train}}$: a 784×15000 matrix, with each column being the raw pixel values of an image, totally 15,000 training samples;
- $\boldsymbol{y}_{\text{train}}$: a 15000-dimensional vector, with each row being either 5, 6, or 8, indicating the labels for the images in $\boldsymbol{X}_{\text{train}}$;
- $\boldsymbol{X}_{\text{test}}$: a 784×2500 matrix, used for testing;
- $\boldsymbol{y}_{\text{test}}$: a 2500-dimensional vector, containing the labels for $\boldsymbol{X}_{\text{test}}$.

8.2 Data Preparation (10 pts)

The first step in scientific discovery is to visualize the data set, to check if the data is correct, and to perform proper pre-processing.

- Randomly pick 10 images from the training set, and another 10 images from the testing set. Use python to display them. Note that the images are stored as a 784-dimensional row vector. You need to reshape it into 28×28.
- Print the corresponding image labels in $\boldsymbol{y}_{\text{train}}$ and $\boldsymbol{y}_{\text{test}}$ to see if the data have correct annotation.
- Normalize all the feature vectors (i.e. the rows of $\boldsymbol{X}_{\text{train}}$ and $\boldsymbol{X}_{\text{test}}$) such that they have unit ℓ_2-norm.
- Normalize all the columns in $\boldsymbol{X}_{\text{train}}$ and $\boldsymbol{X}_{\text{test}}$ such that each column has unit ℓ_2 norm.

8.3 Evaluation Metric (30 pts)

Suppose that for each query $\boldsymbol{x}$ in $\boldsymbol{X}_{\text{test}}$, we are allowed to present k number of data points that are most related to it (you can think of k as number of links shown on the first page of Google). Here, the relevance is measured by the distance under ℓ_2-norm and k is a hyper-parameter we need to specify.

[2] `http://yann.lecun.com/exdb/mnist/`

- Implement the function `dist`($\boldsymbol{X}_{\text{train}}$, $\boldsymbol{X}_{\text{test}}$) which returns the distance matrix $\boldsymbol{M}$, where the (i, j)-th element represents the i-th data point in $\boldsymbol{X}_{\text{train}}$ and the j-th data point in $\boldsymbol{X}_{\text{test}}$.

- For the i-th point in $\boldsymbol{X}_{\text{test}}$, implement the function `retrieve_k`(i, $\boldsymbol{M}$, k) which returns the indices of k closest points in $\boldsymbol{X}_{\text{train}}$.

For a particular query $\boldsymbol{x}$ in $\boldsymbol{X}_{\text{test}}$, suppose that $I = \{i_1, \ldots, i_k\}$ are the indices returned. We will measure the search performance on $\boldsymbol{x}$ by precision@k:

$$\text{precision@}k := \frac{\sum_{j=1}^{k} \mathbf{1}_{\{y_{i_j}=y\}}}{k}. \tag{8.1}$$

In the above expression, y is the label of our query $\boldsymbol{x}$, $y_{i_1}, \ldots, y_{i_k}$ are the labels from $\boldsymbol{y}_{\text{train}}$ that are indexed by $i_1, \ldots, i_k$, and the indicator function $\mathbf{1}_{\{E\}}$ outputs 1 if event E is true and 0 otherwise.

- Implement the function `precision_k`(y, $\boldsymbol{y}_{\text{train}}$, I) where $I = \{i_1, \ldots, i_k\}$.

Now consider the whole query set $\boldsymbol{X}_{\text{test}}$. For each query, you have implemented the precision@k. The overall performance on the set is evaluated by the average.

- Implement the function `avg_precision_k`($\boldsymbol{X}_{\text{train}}$, $\boldsymbol{y}_{\text{train}}$, $\boldsymbol{X}_{\text{test}}$, $\boldsymbol{y}_{\text{test}}$, k).

8.4 Retrieval Performance with Original Data

For each $k \in \{1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\}$, calculate the average precision at k and the running time of retrieval. Plot the average precision curve against k and summarize your findings.