University of Pittsburgh
Fall 2018

CMPBIO2065: Scalable Machine Learning Algorithms for Big-Data Biology

HW #1 - Random Projections and Matrix Inversion

DATE: *2nd September 2018*
DUE : *12th September 2018*

1 Random Projections for Low-Distortion Embeddings

RANDOM PROJECTIONS: Dimensionality reduction is the process of mapping a high dimensional dataset to a lower dimensional space, while preserving much of the important structure. One approach is to find a small number of directions which capture maximum variance in the dataset. Principal component analysis is a standard technique for this purpose.

In this assignment, we will consider a different approach to dimensionality reduction where the goal is to preserve pairwise distances between the data points. We present a technique, known as the random projection method, for solving this problem. The analysis of this technique is known as the Johnson-Lindenstrauss lemma.

Suppose we have a dataset of n points

$$\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\} \in \mathcal{R}^d.$$

We would like to *embed* them to a new dataset of n points

$$\{\vec{y}_1, \vec{y}_2, ..., \vec{y}_n\} \in \mathcal{R}^k,$$

where $k \ll n$, such that

$$\|y_j\| \approx \|x_j\| \qquad \forall j$$
$$\|y_j - y_{j'}\| \approx \|x_j - x_{j'}\| \qquad \forall j, j'$$

The notation $\|x\|$ refers to the usual Euclidean norm of the vector. Johnson-Lindenstrauss Lemma shows this can be accomplished while taking k to be surprisingly small.

JOHNSON-LINDENSTRAUSS LEMMA Let $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\} \in \mathcal{R}^d$ be arbitrary. Pick $\epsilon \in (0, 1)$. Then for some $k = O(log(n)/\epsilon^2$ there exists points $\{\vec{y}_1, \vec{y}_2, ..., \vec{y}_n\} \in \mathcal{R}^k$ such that

$$(1-\epsilon)\|x_j\| \leq \|y_j\| \leq (1+\epsilon)\|x_j\| \qquad \forall j$$
$$(1-\epsilon)\|x_j - x_j'\| \leq \|y_j - y_j'\| \leq (1+\epsilon)\|x_j - x_j'\| \qquad \forall j, j'$$

In polynomial time, it is easy to compute a linear transformation $L : \mathcal{R}^d \to \mathcal{R}^k$, such that, defining $y_j := L(x_j)$, the above inequalities are satisfied with probability at least $1 - 1/n$. Note the final embedded points y_j have no dependence on d. The surprising result is that the linear transformation L is simply multiplication by a matrix whose entries are independent Gaussian random variables. **In this assignment, we will empirically test the low-distortion properties of the embedding.**

To begin with, download the MNIST test dataset of handwritten digits from:

`http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz`

Note from `http://yann.lecun.com/exdb/mnist/`: "The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. Your browser may uncompress the gzipped files without telling you. If the files you downloaded have a larger size than mentioned in the webpage, they have been uncompressed by your browser. Simply rename them to remove the .gz extension. These files are not in any standard image format. You have to write your own (very simple) program to read them". The file format is described below:

```
TEST SET IMAGE FILE (t10k-images-idx3-ubyte):

[offset] [type]          [value]          [description]
0000     32 bit integer  0x00000803(2051) magic number
0004     32 bit integer  10000            number of images
0008     32 bit integer  28               number of rows
0012     32 bit integer  28               number of columns
0016     unsigned byte   ??               pixel
0017     unsigned byte   ??               pixel
........
xxxx     unsigned byte   ??               pixel
Pixels are organized row-wise. Pixel values are 0 to 255.
0 means background (white), 255 means foreground (black).
```

Using this dataset we will test the low-distortion properties of the Johnson-Lindenstrauss based embedding.

A. Create a data matrix $X \in \mathcal{R}^{d \times n}$ from the first $n = 2000$ digits of this dataset. The dimension of each digit is $d = 784$.

B. Find the ten nearest neighbors for each point in X, that is, compute $\mathcal{N}_{i,10}$ for $1 \leq i \leq n$, where $\mathcal{N}_{i,t}$ denotes the set of the t nearest neighbors for the i^{th} datapoint and nearest neighbors are defined with respect to the L_2 norm. Also, compute $\mathcal{N}_{i,50}$.

C. Generate $\widetilde{X} = RX$ where $R \in \mathcal{R}^{k \times d}$, $k = 100$ and entries of R are sampled independently from the standard normal distribution (zero-mean, unit-variance). Find the ten nearest neighbors for each point in $\widetilde{X}$, that is, compute $\widetilde{\mathcal{N}}_{i,10}$ for $1 \leq i \leq n$.

D. Report the quality of approximation by computing two scores:

$$\text{score}_{10} = \frac{1}{n} \sum_{i=1}^{n} \left| \mathcal{N}_{i,10} \cap \widetilde{\mathcal{N}}_{i,10} \right| , \tag{1.1}$$

$$\text{score}_{50} = \frac{1}{n} \sum_{i=1}^{n} \left| \mathcal{N}_{i,50} \cap \widetilde{\mathcal{N}}_{i,50} \right| . \tag{1.2}$$

The scores involve computing the intersection $\cap$ between two sets (i.e., how many elements are common between the two sets).

E. Generate two plots that show score_{10} and score_{50} as functions of k, i.e., perform steps (C.) and (D.) for $k = \{1, 10, 50, 100, 250, 500\}$. Provide a brief explanation of these plots.

F. Show similar plots as in (E.) using PCA (Principal Component Analysis) with various values of k to generate $\widetilde{X}$ and subsequently compute nearest neighbors. Are the nearest neighbor approximations generated via PCA better or worse that those generated via random projections? Explain.

2 Inverting Large-Scale Covariance Matrices with Low-Rank Substructures

We will explore one idea for inverting a large-scale covariance matrix with low-rank substructure. We will use a Factor Analysis model for demonstration.

The Factor Analysis model invokes a linear relationship between the latent variables $\vec{z}$ and the observables $\vec{x}$:

$$\vec{x} = B\vec{z} + \vec{\mu} + \vec{\epsilon}. \tag{2.1}$$

Here, the observables are $\vec{x} \in \mathcal{R}^d$, the latent variables $\vec{z} \in \mathcal{R}^l$ where $l \ll d$ are drawn from a zero-mean, unit variance Normal distribution $\mathcal{N}(0, I)$, B is a matrix of factor loadings with size: $d \times l$, $\vec{\mu}$ is the mean vector, $\vec{\epsilon}$ is the noise vector with a diagonal covariance matrix $\mathcal{N}(0, D)$.

The Factor Analysis model embeds a low-dimensional Gaussian distributed latent space of dimension l into a high-dimensional Gaussian distributed data space of dimension d

for the observables. The diagonal variance matrix D provides numerical stability (How? Also, what is the inverse of matrix D?).

A. DIMENSION CHECK The Gaussian probability distribution over the observables $\vec{x}$ is given by:

$$P(\vec{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}\det(\Sigma)^{1/2}} \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right\}, \tag{2.2}$$

where $\vec{\mu}$ is the mean vector and $\Sigma = BB^T + D$ is the covariance matrix (Make sure of this).

How many free parameters are there in this Factor Analysis model? Hint: $\vec{\mu}$ requires d parameters.

B. LIKELIHOOD COMPUTATION The log-likelihood of this probability model is given by

$$\log\left(P(\vec{x}|\mu, \Sigma\right) = -\frac{1}{2}\left[d\log(2\pi) + \log\det(\Sigma) + (\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu}).\right] \tag{2.3}$$

Our goal is to avoid computing the matrix inverse Σ^{-1}, which will be on the order $O(d^3)$. We can exploit the intrinsic structure of the covariance matrix $\Sigma = BB^T + D$ in computing its inverse. In particular, we can use Woodbury matrix inversion lemma to compute the inverse of an arbitrary system of matrices A, U, C, and V of appropriate sizes as given below:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}. \tag{2.4}$$

This formula is most appropriate when the inverse of A is easy to compute and the matrix product UCV introduces a low-rank perturbation to A.

Show how you will compute the inverse Σ^{-1} by the Woodbury matrix inversion lemma. You will discover that Σ^{-1} requires inverting a $l \times l$ matrix. As an aside, this new matrix also plays a role in evaluating the determinant term in the log-likelihood expression.

In addition to Σ^{-1}, we have to evaluate the Mahalanobis distance, which is the quadratic term: $(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})$. To perform this large scale matrix multiplication, we have to declare space (memory) of size $d \times d$ to hold Σ^{-1}*values*

Given your new understanding of the Woodbury inversion formula for inverting Σ, **describe how you will evaluate the Mahalanobis distance without storing the $d \times d$ matrix of Σ^{-1}.**