Random Projections and Dimension Reduction

Rishi Advani[1] Madison Crim[2] Sean O'Hagan[3]

[1]Cornell University

[2]Salisbury University

[3]University of Connecticut

Summer@ICERM, July 2020

Acknowledgements

Thank you to our organizers, Akil Narayan and Yanlai Chen, along with our TAs, Justin Baker and Liu Yang, for supporting us throughout this program

Table of Contents

Johnson-Lindenstrauss Lemma

If we have n data points in $\mathbb{R}^d$, there exists a linear map into $\mathbb{R}^k$ such that pairwise distances between data points can be preserved up to an ϵ tolerance, provided $k > C\varepsilon^{-2} \log n$, where C is some constant. The proof follows three steps:

- Define a random linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ by $f(u) = \frac{1}{\sqrt{k}} R \cdot u$, where $R \in \mathbb{R}^{k \times d}$ drawn elementwise from a standard normal distribution
- If $u \in \mathbb{R}^d$, show $\mathbb{E}[\|f(u)\|_2^2] = \|u\|_2^2$
- Show that the random variable $\|f(u)\|_2^2$ concentrates around $\|u\|_2^2$, and construct a union bound over all pairwise distances.

Johnson-Lindenstrauss Lemma: Demonstration

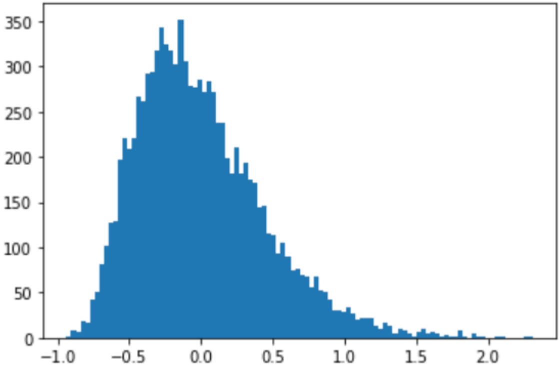


Figure: Histogram of $\|u\|_2^2 - \|f(u)\|_2^2$ for a fixed $u \in \mathbb{R}^{1000}$, $f(u) \in \mathbb{R}^{10}$

Table of Contents

Deterministic Interpolative Decomposition

Given a matrix $A \in \mathbb{R}^{m \times n}$ we can come up with a low-rank matrix approximation that uses $A's$ own columns. The ID can be computed using the column-pivoted QR factorization:

$$AP = QR\,.$$

To obtain our low-rank approximation we form the submatrix Q_k formed by the first k columns of Q. Thus we have the approximation:

$$A \approx Q_k Q_k^* A$$

which gives us a particular rank k projection of A.

Randomized Interpolative Decomposition

This method constructs a subset S of randomly selected distinct $p > k$ columns from the n columns of A. The algorithm then performs the column-pivoted QR factorization on the p columns of A:

$$A_{(:,S)}P = QR$$

Accordingly we have the following rank k projection of A:

$$A \approx Q_k Q_k^* A \,,$$

where Q_k is the submatrix formed by the first k columns of Q.

Table of Contents

Deterministic Singular Value Decomposition

- Recall the singular value decomposition of a matrix:

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V_{n\times n}^T,$$

  where U and V are orthogonal matrices, and Σ is a diagonal matrix with positive diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$, where r is the rank of the matrix A.
- The σ_is are called the singular values of A.

Randomized Singular Value Decomposition

Utilizing ideas from [HMT09], our algorithm executes the following steps to compute the randomized SVD:

1. Construct a $n \times k$ random Gaussian matrix Ω
2. Form $Y = A\Omega$
3. Construct a matrix Q whose columns form an orthonormal basis for the column space of Y
4. Set $B = Q^* A$
5. Compute the SVD: $B = U' \Sigma V^*$,
6. Construct the SVD approximation: $A \approx QQ^* A = QB = QU' \Sigma V^*$

Table of Contents

Results - Testing 620 $\times$ 187500 Matrix

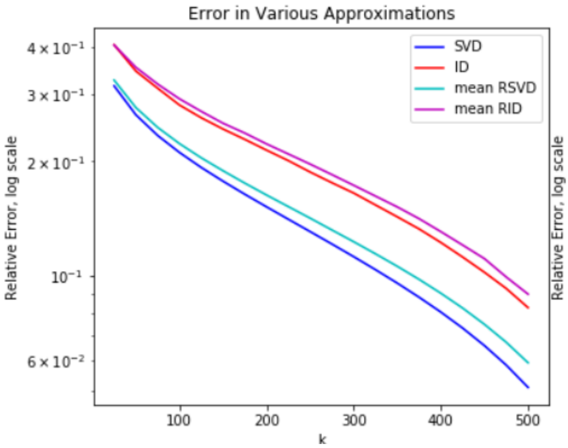


Figure: Error Relative to Original Data

Results - Testing 620×187500 Matrix

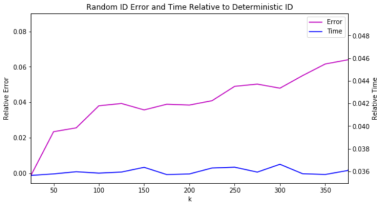


Figure: Random ID Error and Time Relative to Deterministic ID

Figure: Random SVD Error and Time Relative to Deterministic SVD

Table of Contents

Eigenfaces

- Using ideas from [BKP15] our eigenfaces experiment tests the LFW dataset [Hua+07]. This dataset contains more then 13,000 RGB images of faces where each image is a 250×250.
- We can flatten each image to represent it as vector of length $250 \cdot 250 \cdot 3 = 187500$.
- In our experiment we will only use 620 images from the LFW dataset. This gives us a data matrix A of size 187500×620.
- We then can perform SVD on the mean-subtracted columns of A.

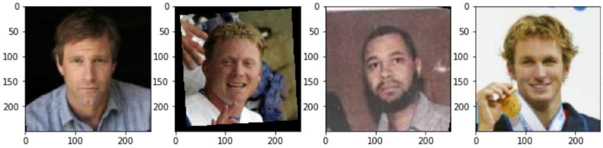


Figure: Original LFW Images

Image Results

We obtain the following eigenfaces from the columns of the matrix U:

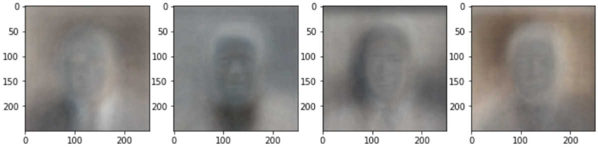


Figure: Eigenfaces Obtained using Deterministic SVD

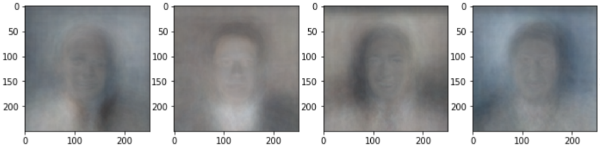


Figure: Eigenfaces Obtained using Randomized SVD

Table of Contents

Kernel Methods

- Kernel methods describe mapping the data into a high-dimensional space to add more structure and encourage linear separability.

$$\phi : \mathbb{R}^n \to \mathbb{R}^m, \quad m > n$$

- The 'kernel trick' allows for only the inner products to be computed, as opposed to the explicit high-dimensional mappings.

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

- Kernel methods include kernel PCA, kernel SVM, and more.

Randomized Fourier Features Kernel

Introduced in [RR08], we can sample random Fourier features to approximate a kernel. Let $k(\mathbf{x}, \mathbf{y})$ denote our kernel, and $p(w)$ the probability distribution corresponding to the inverse Fourier transform of k.

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{y}) &= \int_{\mathbb{R}^d} p(w) e^{-\mathrm{j} w^T (\mathbf{x} - \mathbf{y})} \mathrm{d} w \\
&\approx \frac{1}{m} \sum_{i=1}^{m} \cos(w_i^T \mathbf{x} + b_i) \cos(w_i^T \mathbf{y} + b_i)
\end{aligned}
$$

where $w_i \sim p(w)$, $b_i \sim \text{Uniform}(0, 2\pi)$. For a given m, define

$$
z(x) = \sum_{i=1}^{m} \cos(w_i^T x + b_i)
$$

to yield the approximation $k(\mathbf{x}, \mathbf{y}) \approx \frac{1}{m} z(x) z(y)^T$.

Table of Contents

Data for Kernel PCA Experiments

To test kernel PCA methods, we used a dataset that is not linearly separable– a cloud of points surrounded by a circle:

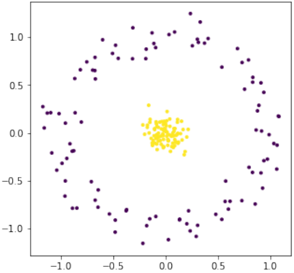


Figure: Data used to test kernel PCA methods

Randomized Kernel PCA Results

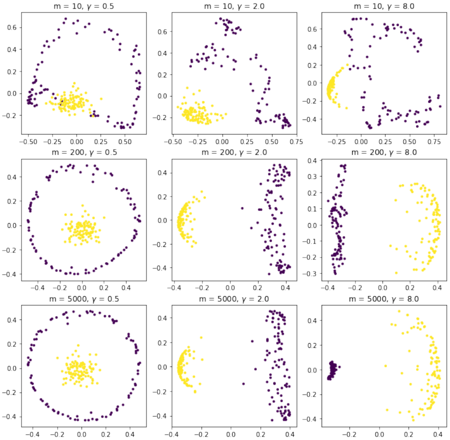


Figure: Random Fourier features KPCA results

Table of Contents

Kernel SVM

- We may also use kernel methods for support vector machines (SVM).
- The goal of an SVM is to find the $(d-1)$-hyperplane that best separates two clusters of d-dimensional data points.
- In two dimensions, this is a line separating two clusters of points in a plane.
- Using the kernel trick, we can project inseparable points into a higher dimension and run an SVM algorithm on the resulting points.

Randomized Kernel SVM

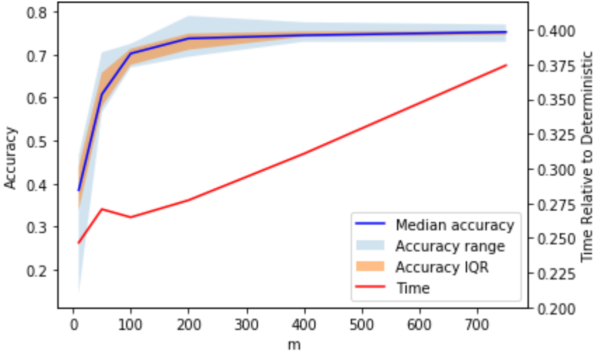


Figure: Randomized Kernel SVM Accuracy and time results as m varies

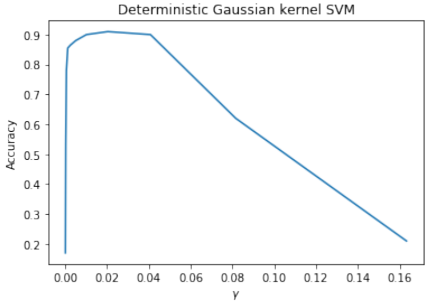
Comparison of Deterministic and Randomized Kernel SVM

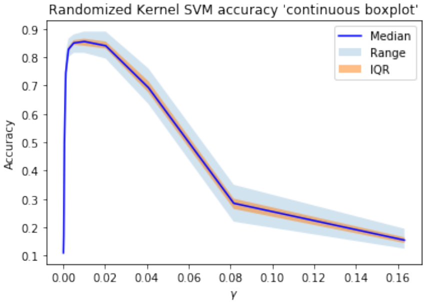
Using the MNIST dataset [LC10] we test 10000 images (784 features), for a **fixed** γ:

- Deterministic Kernel
  - Accuracy: 0.9195
  - Time: 37.99s
- Randomized Kernel
  - Accuracy: Mean: 0.891, St. dev. 0.0042
    Min: 0.881, Max: 0.9005
  - Mean Time: 2.14s

Comparison of Deterministic and Randomized Kernel SVM

On 1000 MNIST images, we plot the accuracies of the deterministic and random kernel SVMs as γ varies:





Application of Randomized Kernel SVM: Grid Search

$$\hat{\mathbf{K}} = \frac{1}{m} z(\mathbf{X}) z(\mathbf{X})^T$$

Testing 100 γ values to identify the best one:

- Deterministic Kernel, Series: 133.03s
- Randomized Kernel, Series: 78.97s
- Randomize Kernel, Parallel: 41.18s
- Best γ value obtained from randomized method corresponds with either best or second best deterministic γ (3 trials)

References

Brunton, Kutz, and Proctor. *Eigenfaces Example*. 2015. URL: http://faculty.washington.edu/sbrunton/me565/pdf/L29secure.pdf.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*. 2009. arXiv: 0909.4061 [math.NA].

Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

Ali Rahimi and Benjamin Recht. *Random Features for Large-Scale Kernel Machines*. Ed. by J. C. Platt et al. 2008.

Website

To explore more visit our website at the following link:
https://rishi1999.github.io/random-projections/