DIMENSION REDUCTION AND THE FENCHEL GAME

LUCAS TUCKER

Abstract. In this paper, we review the linear dimension reduction algorithm Principal Component Analysis (PCA), as well as the non-linear reduction algorithms Isomap and Laplacian Eigenmaps. We then examine a novel framework (Fenchel Game No Regret Dynamics) for converting convex function minimization into min-max style games, and apply one such method to a reformulation of PCA. This paper also serves as a reference for the theoretical underpinning of such popular dimension reduction algorithms (which is often left as an afterthought), as well as an attempt to clarify and build upon Wang-Abernathy-Levy's paper.

Contents

1. PCA

We first examine a commonly used linear method (linear transformation of data) to reduce dimensionality of a data matrix. This method, Principal Component Analysis (PCA), is most helpful for quick dimensionality reductions when features are highly correlated, as is demonstrated in the statement of Theorem 1.5.

Definition 1.1. A projection on a vector space V is a linear operator $P : V \to V$ such that $P^2 = P$. A projection on a Hilbert space V is an orthogonal projection if $\langle Px, y\rangle = \langle x, Py\rangle$

Definition 1.2. The "Frobenius norm", denoted by $||.||_F$ is a matrix norm defined over $\mathbb{R}^{m\times n}$ as

$$||\mathbf{M}||_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{M}_{ij}^2}$$

Date: September 15, 2023.

Definition 1.3. For a sample $S = (x_1, ..., x_m)$ and feature mapping $\mathbf{\Phi} : \mathcal{X} \to \mathbb{R}^n$, we define the data matrix $(\mathbf{\Phi}(x_1), ..., \mathbf{\Phi}(x_m)) =: \mathbf{X} \in \mathbb{R}^{n\times m}$. If $\mathbf{X}$ is a mean-centered data matrix ($\sum_{i=1}^m \mathbf{\Phi}(x_i) = \mathbf{0}$), let $\mathcal{P}_k$ denote the set of n-dimensional rank$-k$ orthogonal projection matrices. PCA (Principal Component Analysis) is defined by the orthogonal projection matrix

$$\mathbf{P}^* := \text{argmin}_{\mathbf{P}\in\mathcal{P}_k} ||\mathbf{PX} - \mathbf{X}||_F^2$$

Definition 1.4. The "top singular vector" of a matrix $\mathbf{M}$ is the vector $\mathbf{x}$ which maximizes the Rayleigh quotient

$$r(\mathbf{x}, \mathbf{M}) = \frac{\mathbf{x}^T\mathbf{M}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

Theorem 1.5. *Let $\mathbf{P}^* \in \mathcal{P}_k$ be the PCA solution for a centered (mean among the $\mathbf{\Phi}(x_i)$ is $\mathbf{0}$) data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$. Then, $\mathbf{P}^* = \mathbf{U}_k\mathbf{U}_k^T$, where $\mathbf{U}_k \in \mathbb{R}^{n\times k}$ is the matrix formed by the top k singular vectors of $\mathbf{C} := \frac{1}{m}\mathbf{X}\mathbf{X}^T$, the sample covariance matrix corresponding to $\mathbf{X}$. note that this is the sample covariance matrix since*

$$\frac{1}{m}(\mathbf{X}\mathbf{X}^T)_{ij} = \frac{1}{m}\sum_{\ell=1}^m \mathbf{X}_{i\ell}\mathbf{X}_{\ell j}^T = \frac{1}{m}\sum_{\ell=1}^m \mathbf{\Phi}(x_\ell)_i\mathbf{\Phi}(x_\ell)_j$$

$$= E[\mathbf{\Phi}(x)_i\mathbf{\Phi}(x)_j] = E[\mathbf{\Phi}(x)_i\mathbf{\Phi}(x)_j] - E[\mathbf{\Phi}(x)_i]E[\mathbf{\Phi}(x)_j] = Cov(\mathbf{\Phi}(x)_i, \mathbf{\Phi}(x)_j)$$

where the right hand term is the covariance between i-th and j-th coordinates of the feature output based on m samples. Moreover, the associated k-dimensional representation of $\mathbf{X}$ is given by $\mathbf{Y} = \mathbf{U}_k^T\mathbf{X}$.

Proof. For $\mathbf{P} = \mathbf{P}^T$ an orthogonal projection matrix, we seek to minimize

$$||\mathbf{PX} - \mathbf{X}||_F^2 = \sum_{i=1}^n\sum_{j=1}^n((\mathbf{PX} - \mathbf{X})_{ij})^2 = \text{Tr}[(\mathbf{PX} - \mathbf{X})^T(\mathbf{PX} - \mathbf{X})]$$

$$= \text{Tr}[\mathbf{X}^T\mathbf{P}^2\mathbf{X} - \mathbf{X}^T\mathbf{P}^T\mathbf{X} - \mathbf{X}^T\mathbf{PX} + \mathbf{X}^T\mathbf{X}] = \text{Tr}[\mathbf{X}^T\mathbf{PX} - 2\mathbf{X}^T\mathbf{PX} + \mathbf{X}^T\mathbf{X}]$$
$$= \text{Tr}[\mathbf{X}^2] - \text{Tr}[\mathbf{X}^T\mathbf{PX}]$$

hence we seek to maximize

$$\text{Tr}[\mathbf{X}^T\mathbf{PX}] = \text{Tr}[\mathbf{X}^T\mathbf{U}_k\mathbf{U}_k^T\mathbf{X}] = \text{Tr}[\mathbf{U}_k^T\mathbf{X}\mathbf{X}^T\mathbf{U}_k]$$

$$= \sum_{i=1}^k\Big(\sum_{j=1}^n(\mathbf{U}_k^T\mathbf{X}\mathbf{X}^T)_{ij}(\mathbf{U}_k)_{ji}\Big) = \sum_{i=1}^k\Big(\sum_{j=1}^n\Big(\sum_{\ell=1}^n(\mathbf{U}_k^T)_{i\ell}(\mathbf{X}\mathbf{X}^T)_{\ell j}\Big)(\mathbf{U}_k)_{ji}\Big)$$

so for $\mathbf{u}_i := ((\mathbf{U}_k)_{1i}, ..., (\mathbf{U}_k)_{ni})$,

$$= \sum_{i=1}^n\Big(\mathbf{u}_i^T\mathbf{X}\mathbf{X}^T\mathbf{u}_i\Big)$$

where

$$\mathbf{PX} = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X}$$

so that $\mathbf{Y} := \mathbf{U}_k^T\mathbf{X}$ is a k-dimensional representation of $\mathbf{X}$. □

Remark 1.6. The top singular vectors of $\mathbf{C}$ are the directions of maximal variance in the data, and the $\mathbf{u}_i$ are the variances, so that PCA may be understood as projection onto the subspace of maximal variance.

Remark 1.7. In Kernel principle component analysis (KPCA), the feature map Φ send $\mathcal{X}$ to an arbitrary Reproducing Kernel Hilbert Space (RKHS) equipped with its own inner product (kernel function K).

Theorem 1.8. *The following holds for the PCA gradient:*

$$\frac{\partial}{\partial \mathbf{U}_k}||\mathbf{U}_k\mathbf{U}_k^T\mathbf{X}-\mathbf{X}||_F^2 = 2(\mathbf{X}\mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{U}+\mathbf{U}\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U}-2\mathbf{X}\mathbf{X}^T\mathbf{U})$$

Proof. We find that

$$\frac{\partial}{\partial \mathbf{U}_k}||\mathbf{P}\mathbf{X}-\mathbf{X}||_F^2 = \frac{\partial}{\partial \mathbf{U}_k}\Big(\mathrm{Tr}[\mathbf{X}^T\mathbf{P}^2\mathbf{X}]-2\mathrm{Tr}[\mathbf{X}^T\mathbf{P}\mathbf{X}]+\mathrm{Tr}[\mathbf{X}^T\mathbf{X}]\Big)$$

We decompose the right hand side and relax notation as $\mathbf{U} := \mathbf{U}_k$ (since we know our desired dimension k):

$$\frac{\partial}{\partial \mathbf{U}}\mathrm{Tr}[\mathbf{X}^T\mathbf{P}^2\mathbf{X}] = \frac{\partial}{\partial \mathbf{U}}\mathrm{Tr}[\mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{U}\mathbf{U}^T\mathbf{X}]$$

$$= (\mathbf{X}^T)^T(\mathbf{U}^T\mathbf{U}\mathbf{U}^T\mathbf{X})^T+(\mathbf{U}\mathbf{U}^T\mathbf{X})(\mathbf{X}^T\mathbf{U})+(\mathbf{X}^T\mathbf{U}\mathbf{U}^T)(\mathbf{U}^T\mathbf{X})^T+(\mathbf{X})(\mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{U})$$
$$= 2\mathbf{X}\mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{U}+2\mathbf{U}\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U}$$

so that

$$\frac{\partial}{\partial \mathbf{U}}\Big(\mathrm{Tr}[\mathbf{X}^T\mathbf{P}^2\mathbf{X}]-2\mathrm{Tr}[\mathbf{X}^T\mathbf{P}\mathbf{X}]\Big)$$
$$= 2\mathbf{X}\mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{U}+2\mathbf{U}\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U}-2\Big((\mathbf{X}^T)^T(\mathbf{U}^T\mathbf{X})^T+(\mathbf{X})(\mathbf{X}^T\mathbf{U})\Big)$$
$$= 2(\mathbf{X}\mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{U}+\mathbf{U}\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U}-2\mathbf{X}\mathbf{X}^T\mathbf{U})$$

□

Remark 1.9. While the problem

$$\mathrm{argmin}_{\mathbf{P}\in\mathbb{R}^{n\times n}}||\mathbf{P}\mathbf{X}-\mathbf{X}||_F^2$$

minimizes a convex function, the assumption $\mathbf{P} = \mathbf{U}_k\mathbf{U}_k^T$ leads to a non-convex minimization

$$\mathrm{argmin}_{\mathbf{U}_k\in\mathbb{R}^{n\times k}}||\mathbf{U}_k\mathbf{U}_k^T\mathbf{X}-\mathbf{X}||_F^2$$

Further, after finding $\mathbf{U}_{\min}$ using gradient descent, we must find the closest orthogonal matrix $\mathbf{U} := \mathbf{A}\mathbf{B}^T$ for a Singular Value Decomposition $\mathbf{U}_{\min} = \mathbf{A}\mathbf{\Sigma}\mathbf{B}^T$.

2. Experiment: PCA

Define the t-similarity score to be the following: Given dataset $\mathbf{X}\in\mathbb{R}^{n\times m}$ of m points in $\mathbb{R}^n$, and a point $\mathbf{x}\in\mathbf{X}$, let $n_t(\mathbf{x})$ be the t closest points $y\in X$ to $\mathbf{x}$ under ℓ_2-distance. Let $f:\mathbb{R}^n\to\mathbb{R}^k$ be the dimensionality reduction map from full dimension n to low-dimension k. Define the *t-similarity score* for $x\in X$ to be $\ell_t(\mathbf{x})$ given by

$$\ell_{t,f}(\mathbf{x}) := |n_t(\mathbf{x})\cap n_t(f(\mathbf{x}))|$$

Let the *t-similarity score* of a dimensionality reduction technique f to be

$$\mathsf{score}_t(f) := \frac{1}{m}\cdot\sum_{\mathbf{x}\in X}\ell_{t,f}(\mathbf{x})$$

The experiment is as follows:

- Take $m = 2000$ points from MNIST ($n = 784$ pixels)

- Compute f^k_{random}, f^k_{pca}, and $f^k_{\mathsf{fast-pca}}$ given by random projection of $\mathbb{R}^n$ into $\mathbb{R}^k$, baseline PCA implementation, and fast gradient-based PCA implementation.
- Plot the three curves: where the x-axis is the dimension k ranging (in logarithmic steps from $k = 2$ to $k = 784$, and the y-axis is given by $\mathsf{score}_t(f^k_{\mathsf{random}})$ (blue), $\mathsf{score}_t(f^k_{\mathsf{pca}})$ (red), and $\mathsf{score}_t(f^k_{\mathsf{fast-pca}})$ (green) using $t = 10$.

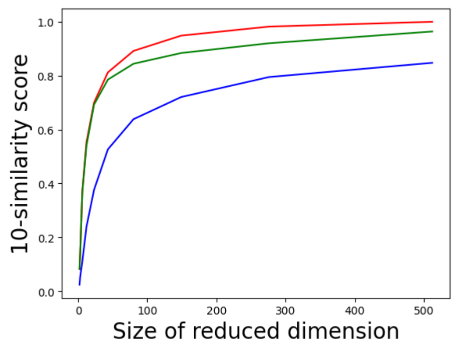
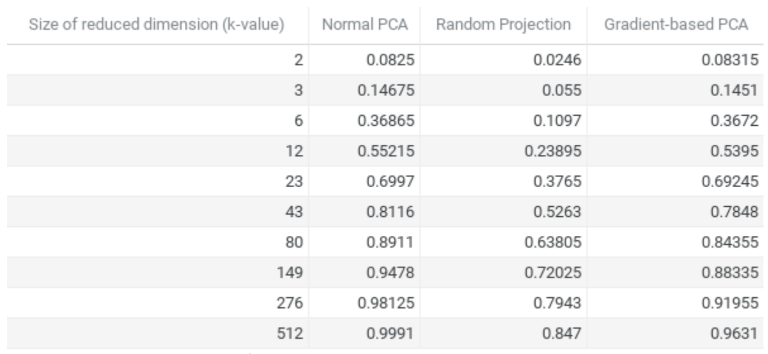


FIGURE 1. Comparison of similarity scores among the various PCA techniques as a function of dimension size



Size of reduced dimension (k-value)	Normal PCA	Random Projection	Gradient-based PCA
2	0.0825	0.0246	0.08315
3	0.14675	0.055	0.1451
6	0.36865	0.1097	0.3672
12	0.55215	0.23895	0.5395
23	0.6997	0.3765	0.69245
43	0.8116	0.5263	0.7848
80	0.8911	0.63805	0.84355
149	0.9478	0.72025	0.88335
276	0.98125	0.7943	0.91955
512	0.9991	0.847	0.9631

FIGURE 2. Values of the points in Figure 1, where the dimension size input is logarithmically spaced

Remark 2.1. As visualized above, the similarity scores (with respect to the data matrix X) of both the gradient-based PCA method and baseline implementation remained within 0.02 of one another up to a 40 dimensional representation, while the random projection's similarity score remained on average within 0.25 from the baseline PCA implementation.

Non-linear Methods

Rather than linearly transforming our data matrix $\mathbf{X}$ (through matrix multiplication), we may be interested in optimizing our representation with respect to a specific distance or weight matrix, or in preserving local structural information.

3. Laplacian Eigenmaps

Definition 3.1. The Laplacian Eigenmaps algorithm aims to find a k-dimensional representation of the data matrix $\mathbf{X}$ which best preserves the weighted neighborhood relations specified by a matrix $\mathbf{W}$:

Algorithm 1 Laplacian Eigenmaps

```
Require: points $\mathbf{x} \in \mathbb{R}^n$
Require: scaling parameter $\sigma$
Require: t-nearest neighbors function $N_t(\mathbf{x})$
  Define: $\mathbf{W} \in \mathbb{R}^{m\times m}$ as $\mathbf{W}_{ij} := \begin{cases} 0 & \mathbf{x}_i \notin N_t(\mathbf{x}_j), \mathbf{x}_j \notin N_t(\mathbf{x}_i) \\ e^{\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{\sigma^2}} & \text{otherwise} \end{cases}$
  Define: $\mathbf{D} \in \mathbb{R}^{m\times m}$ as $\mathbf{D}_{ii} = \sum_{j=1}^m \mathbf{W}_{ij}$
  Evaluate: $\mathbf{Y} \in \mathbb{R}^{k\times m}$ as $\mathbf{Y} = \text{argmin}_{\mathbf{Y}'}\Big\{ \sum_{i,j} \mathbf{W}_{ij}||\mathbf{y}'_i - \mathbf{y}'_j||_2^2 \Big\}$
```

Intuitively, the above minimization penalizes k-dimensional representations of neighbors that differ largely under the ℓ_2 norm.

Proposition 3.2. *The solution to the Laplacian Eigenmaps minimization is* $\mathbf{U}^T_{\mathbf{L},k}$*, where* $\mathbf{L} = \mathbf{D} - \mathbf{W}$ *is the "graph Laplacian" and* $\mathbf{U}^T_{\mathbf{L},k}$ *are the bottom* k *singular vectors of* $\mathbf{L}$ *(excluding* 0 *if the underlying neighborhood graph has connections).*

Proof. We find that, for $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{Y} \in \mathbb{R}^{k\times m}$ we have

$$(\mathbf{YLY}^T)_{ij} = \sum_{\ell=1}^m \mathbf{Y}^T_{i\ell}(\mathbf{LY})_{\ell j} = \sum_{\ell=1}^m \mathbf{Y}_{\ell i}\Big(\sum_{t=1}^m \mathbf{L}_{\ell t}\mathbf{Y}_{tj}\Big)$$

$$= \sum_{\ell=1} \mathbf{Y}_{\ell i} \sum_{t\neq \ell} \mathbf{W}_{\ell t}(\mathbf{Y}_{\ell j} - \mathbf{Y}_{tj}) \tag{3.3}$$

while

$$\sum_{i,\ell} \mathbf{W}_{i\ell}||\mathbf{y}'_i - \mathbf{y}'_\ell||_2^2 = \sum_{i=1}^m\sum_{\ell=1}^m \mathbf{W}_{i\ell}(\mathbf{y}'_i - \mathbf{y}'_\ell)^T(\mathbf{y}'_i - \mathbf{y}'_\ell)$$

$$= \sum_{i=1}^m\sum_{\ell=1}^m \mathbf{W}_{i\ell}((\mathbf{y}'_i)^2 - 2(\mathbf{y}'^T_\ell\mathbf{y}'_i) + (\mathbf{y}'_\ell)^2)$$

$$= \sum_{i=1}^m\sum_{\ell=1}^m \mathbf{W}_{i\ell}\Big(\sum_{j=1}^m (\mathbf{y}'_i)^2_j - 2(\mathbf{y}'_\ell)_j(\mathbf{y}'_i)_j + (\mathbf{y}'_\ell)^2_j\Big)$$

$$= \sum_{i=1}^m\sum_{\ell=1}^m \mathbf{W}_{i\ell}\Big(\sum_{j=1}^m \mathbf{Y}'_{ji}{}^2 - 2\mathbf{Y}'_{j\ell}\mathbf{Y}'_{ji} + \mathbf{Y}'_{j\ell}{}^2\Big)$$

hence by (3.3)

$$= \sum_{i=1}^k (\mathbf{Y}'\mathbf{L}\mathbf{Y}'^T)_{ii}$$

so for $\mathbf{Y} := \mathbf{Y}'^T$, by the final simplication used in Theorem 1.5,

$$= \sum_{i=1}^{k} \mathbf{y}_i^T \mathbf{L} \mathbf{y}_i$$

so that $\mathbf{U}_{\mathbf{L},k}^T$ are the bottom k singular vectors of $\mathbf{L}$. □

However, that when the data is not uniformly distributed, neighborhood connections based on the ℓ_2 norm are still weighted relative to those of other clusters in the data. As mentioned by W. Jiang et al., smaller radii in

"short-circuiting" or self-similar neighborhoods, while larger radii in sparser areas can prevent fragmentation.

Hence, we compare two Laplacian Eigenmaps algorithms with the baseline implementation. One of these algorithms is adapted from Jiang et. al. to measure local density, and the other extends this local density algorithm to variable neighbor size.

Definition 3.4. We first Fix a sample size $t \in \mathbb{N}$ and t-nearest neighbors function $N_t(\mathbf{x}_i)$, where the $\{\mathbf{x}_i\}_{i=1}^m \in \mathbb{R}^n$ comprise the data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$. Then, define the *average distance matrix* $A \in \mathbb{R}^{m\times 1}$ as

$$A_i = \frac{\sqrt{\sum_{j=1}^{m} ||\mathbf{x}_i - \mathbf{x}_j||_2^2}}{t}$$

Then, we define our distance matrix $D \in \mathbb{R}^{m\times m}$ as

$$D_{ij} = \begin{cases} \frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{\sqrt{A_i A_j}}, & \text{if } \mathbf{x}_i \in N_t(\mathbf{x}_j) \text{ and } \mathbf{x}_j \in N_t(\mathbf{x}_i) \\ 0, & \text{else} \end{cases}$$

Finally, we define the "Averaged Laplacian Eigenmaps" weight matrix $W^{\text{ALE}} \in \mathbb{R}^{m\times m}$ as

$$W_{ij}^{\text{ALE}} = e^{-\frac{D_{ij}^2}{t}}$$

Note that the solution to the Laplacian Eigenmaps minimization problem given by Proposition 3.1 still holds in this case.

Definition 3.5. We now extend Definition 3.4 to a novel Laplacian Eigenmaps algorithm with variable nearest neighbors. Instead of a fixed radius t we consider a maximum radius R, and assign the variable radius

$$r(\mathbf{x}_i) = R \cdot \left(1 - \frac{A_i}{1 + \max_{j\in[m]} A_j}\right)$$

Hence, the radius is shortened in sparser areas and extended in denser areas, with the distance matrix $D \in \mathbb{R}^{m\times m}$ defined as

$$D_{ij} = \begin{cases} \frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{\sqrt{A_i A_j}}, & \text{if } \mathbf{x}_i \in N_{r(\mathbf{x}_j)}(\mathbf{x}_j) \text{ and } \mathbf{x}_j \in N_{r(\mathbf{x}_i)}(\mathbf{x}_i) \\ 0, & \text{else} \end{cases}$$

We then define the "Variable Radius Laplacian Eigenmaps" weight matrix $W^{\text{VRLE}} \in \mathbb{R}^{m\times m}$ as

$$W_{ij}^{\text{VRLE}} = e^{-\frac{D_{ij}^2}{t}}$$

and solve the minimization problem by way of Proposition 3.1.

4. Experiment: Laplacian Eigenmaps

Remark 4.1. We now compare the the standard Laplacian Eigenmaps (LE) with both the "Averaged Laplacian Eigenmaps" (ALE) algorithm given by W. Jiang et al. and the extended "Variable Radius Laplacian Eigenmaps" algorithm. First we **(A)** visually analyze the dimension reduction of a colored helix (2d manifold) and then **(B)** train a simple neural network on LE, ALE, and VRLE reduced data and assess the results:

(A)

- Take $m = 2000$ points from the helical 3D plot specified by
$$x = \theta\cos(\theta)$$
$$y = \theta\sin(\theta)$$
$$z = r$$
for $r \in [0, 10]$ and $\theta \in [1.5\pi, 4.5\pi]$ and convert them to a data matrix $\mathbf{X} \in \mathbb{R}^{3\times 2000}$
- Compute ALE[$\mathbf{X}$], LE[$\mathbf{X}$] $\in \mathbb{R}^{2\times 2000}$ and visualize the plots. Vary the sample size t used by ALE.

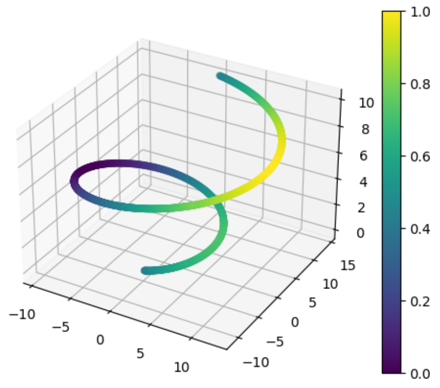


Figure 3. Original helix with color gradient

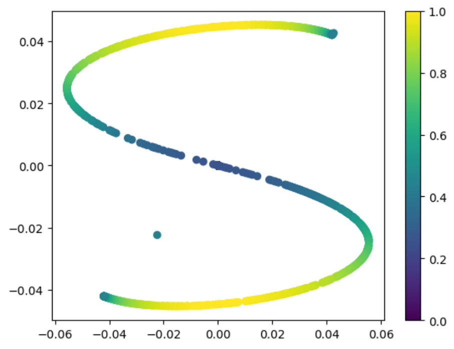


Figure 4. 2d dimension reduction based on Standard LE

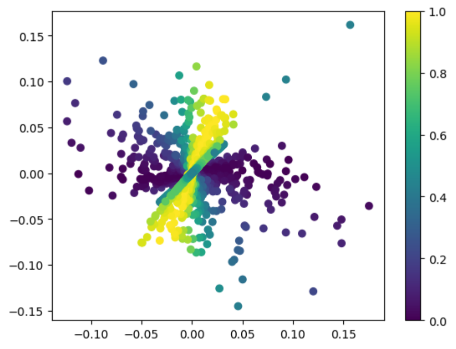


FIGURE 5. 2d dimension reduction based on ALE

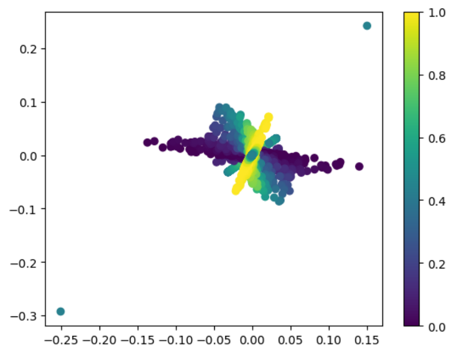


FIGURE 6. 2d dimension reduction based on VRLE

Remark 4.2. In the case of the helix, the standard LE algorithm properly unfolds the manifold and preserves the gradient well (Figure 4), while ALE and VRLE (Figures 5 and 6 respectively) seem to better capture the color gradient but lose some spatial information. In particular, the variable neighbor size VRLE adds to ALE seems to better ensure a lack of outliars in the reduced representation.

(B)

- Sample $m = 58000$ labeled data points from MNIST
- Train a neural network on both ALE- and LE-reduced data
- Compare the nets' performances on the dev set of 2000 labeled data points

5. Isomap

Definition 5.1. Isomap extracts the low-dimensional data that best preserves pairwise distances between inputs based on their geodesic distances along a manifold. The algorithm is specified as follows:

Remark 5.2. Note that $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector, $\mathbf{\Delta}$ is the squared distance matrix of the $\mathbf{x_i}$, and $N_t(\mathbf{x_i})$ are calculated based on the ℓ_2 norm difference in this case (the ℓ_2 norm also determines edge length in $\mathcal{G}$). Additionally, the calculation

$$\mathbf{K}_{\text{Iso}} = -\frac{1}{2}\mathbf{H\Delta H} \approx \mathbf{X}^{*\mathbf{T}}\mathbf{X}^{*} \tag{5.3}$$

Algorithm 2 Isomap

```
Require: Points $\mathbf{x_i} \in \mathbb{R}^n$
    Evaluate: $N_t(\mathbf{x_i})\ \forall \mathbf{x_i}$
    Construct: Undirected neighborhood graph $\mathcal{G}$
    Evaluate: Approximate $\mathbf{\Delta}_{ij}$ as shortest distance in $\mathcal{G}$
    Evaluate: $\mathbf{K}_{\text{Iso}} := -\frac{1}{2}(\mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T)\mathbf{\Delta}(\mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T) := -\frac{1}{2}\mathbf{H}\mathbf{\Delta}\mathbf{H}$
    Evaluate: $\mathbf{Y} \in \mathbb{R}^{k\times m},\ \mathbf{Y} := \text{argmin}_{\mathbf{Y'}\in\mathbb{R}^{k\times m}} \sum_{i,j}(||\mathbf{y}'_i - \mathbf{y}'_j||_2 - \mathbf{\Delta}_{ij})^2$ as $\mathbf{Y} = (\mathbf{\Sigma}_{\text{Iso, k}})^{\frac{1}{2}}\mathbf{U}^T_{\text{Iso,k}}$ (see Proposition 5.1)
```

(where $\mathbf{X}^*$ is mean-centered $\mathbf{X}$) is helpful when geodesic distances Δ_{ij} can be approximated and $\mathbf{X}^{*\mathbf{T}}\mathbf{X}^*$ is expensive to calculate directly. The Isomap minimization method is proven in Proposition 5.1, and Lemma 5.2 shows that $(*)$ is an equality when measuring distances with the Euclidean ℓ_2 norm.

Proposition 5.4. *The optimal k-dimensional representation $\mathbf{Y} \in \mathbb{R}^{k\times m}$ due to the Isomap algorithm is given by*

$$\mathbf{Y} = (\mathbf{\Sigma}_{Iso,\ k})^{\frac{1}{2}}\mathbf{U}^T_{Iso,k}$$

where $\mathbf{\Sigma}_{Iso,\ k}$ is the diagonal matrix of the top k singular values of $\mathbf{K}_{Iso}$ and $\mathbf{U}_{Iso,\ k}$ are the corresponding singular vectors.

Remark 5.5. Note that the top k eigenvectors of $\mathbf{K}_{\text{Iso}}$ give the optimal coordinates of the points in a lower k-dimensional space that preserve pairwise geodesic distances specified by $\mathbf{\Delta}$. We then scale the matrix according to the corresponding eigenvalues with $\sqrt{\mathbf{\Sigma}_{\text{Iso, k}}}$. Hence, $\mathbf{\Delta}$ in this case behaves as a covariance matrix for the space whose dimensions are defined by the data points (i.e. $m \times m$ in this case), and non-linearity is introduced through calculations of the interpoint distances in $\mathbf{\Delta}$.

Lemma 5.6. *We prove the correctness of double centering (i.e that we can compute $\mathbf{K}_{Iso}$) using Euclidean distance. Define $\mathbf{X}$ as in Theorem 1.5, and define $\mathbf{X}^*$ to have $\mathbf{x}^*_i := \mathbf{x}_i - \overline{\mathbf{x}}$ as its i-th column. Let $\mathbf{K} := \mathbf{X}^T\mathbf{X}$ and let $\mathbf{D}$ denote the Euclidean distance matrix with $\mathbf{D}_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. Further, let $\mathbf{\Delta}$ denote the squared distance matrix with $\mathbf{\Delta}_{ij} = \mathbf{D}^2_{ij}$.*

Proof. We find that

$$\begin{aligned}\mathbf{K}_{ij} = \sum_{\ell=1}^m \mathbf{X}^T_{i\ell}\mathbf{X}_{\ell j} &= \frac{1}{2}\Big(\sum_{\ell=1}^m \mathbf{X}^2_{\ell i} - \mathbf{X}^2_{\ell i} + \mathbf{X}^2_{\ell j} - \mathbf{X}^2_{\ell j} + 2\mathbf{X}_{\ell i}\mathbf{X}_{\ell j}\Big) \\ = \frac{1}{2}\Big(\sum_{\ell=1}^m \mathbf{X}^2_{\ell i} + \mathbf{X}^2_{\ell j} - (\mathbf{X}_{\ell j} - \mathbf{X}_{\ell i})^2\Big) &= \frac{1}{2}(\mathbf{K}_{ii} + \mathbf{K}_{jj} - ||\mathbf{x}_i - \mathbf{x}_j||^2) \\ &= \frac{1}{2}(\mathbf{K}_{ii} + \mathbf{K}_{jj} - \mathbf{D}^2_{ij})\end{aligned}$$

Let $\mathbf{K}^* := \mathbf{X}^{*T}\mathbf{X}^*$. We have

$$\frac{1}{m}(\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} = \frac{1}{m}\sum_{t=1}^m \mathbf{K}_{it} = \frac{1}{m}\sum_{t=1}^m\sum_{\ell=1}^m \mathbf{X}_{\ell i}\mathbf{X}_{\ell t} = \sum_{\ell=1}^m (\overline{\mathbf{x}})_\ell(\mathbf{x}_i)_\ell$$

$$\frac{1}{m}(\mathbf{1}\mathbf{1}^T\mathbf{K})_{ij} = \frac{1}{m}\sum_{t=1}^{m}\mathbf{K}_{tj} = \frac{1}{m}\sum_{t=1}^{m}\sum_{\ell=1}^{m}\mathbf{X}_{\ell t}\mathbf{X}_{\ell j} = \sum_{\ell=1}^{m}(\overline{\mathbf{x}})_\ell(\mathbf{x}_j)_\ell$$

and

$$\frac{1}{m^2}(\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} = \frac{1}{m^2}\sum_{t=1}^{m}(\mathbf{1}\mathbf{1}^T)_{it}(\mathbf{K}\mathbf{1}\mathbf{1}^T)_{tj} = \frac{1}{m}\sum_{t=1}^{m}\sum_{\ell=1}^{m}(\overline{\mathbf{x}})_\ell(\mathbf{x}_t)_\ell = \sum_{\ell=1}^{m}(\overline{\mathbf{x}}_\ell)^2$$

Then,

$$\begin{aligned}\mathbf{K}^*_{ij} &= \sum_{\ell=1}^{N}{\mathbf{X}^*_{i\ell}}^T\mathbf{X}^*_{\ell j} = \sum_{\ell=1}^{N}(\mathbf{x}_i-\overline{\mathbf{x}})_\ell(\mathbf{x}_j-\overline{\mathbf{x}})_\ell \\ &= \sum_{\ell=1}^{N}(\mathbf{x}_i)_\ell(\mathbf{x}_j)_\ell - (\mathbf{x}_i)_\ell(\overline{\mathbf{x}})_\ell - (\mathbf{x}_j)_\ell(\overline{\mathbf{x}})_\ell + (\overline{\mathbf{x}})^2_\ell \\ &= \mathbf{K}_{ij} - \frac{1}{m}(\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} - \frac{1}{m}(\mathbf{1}\mathbf{1}^T\mathbf{K})_{ij} + \frac{1}{m^2}(\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij}\end{aligned}$$

so that

$$\mathbf{K}^* = \mathbf{K} - \frac{1}{m}\mathbf{K}\mathbf{1}\mathbf{1}^T - \frac{1}{m}\mathbf{1}\mathbf{1}^T\mathbf{K} + \frac{1}{m^2}\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T$$

We find that

$$\begin{aligned}\mathbf{K}^*_{ij} &= \mathbf{K}_{ij} - \frac{1}{m}(\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} - \frac{1}{m}(\mathbf{1}\mathbf{1}^T\mathbf{K})_{ij} + \frac{1}{m^2}(\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} \\ &= \frac{1}{2}(\mathbf{K}_{ii}+\mathbf{K}_{jj}-\mathbf{D}^2_{ij}) - \frac{1}{m}(\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} - \frac{1}{m}(\mathbf{1}\mathbf{1}^T\mathbf{K})_{ij} + \frac{1}{m^2}(\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T)_{ij} \\ &= \frac{1}{2}(\mathbf{K}_{ii}+\mathbf{K}_{jj}-\mathbf{D}^2_{ij}) - \frac{1}{m}\sum_{t=1}^{m}\mathbf{K}_{it} - \frac{1}{m}\sum_{t=1}^{m}\mathbf{K}_{tj} + \frac{1}{m^2}\sum_{t=1}^{m}\sum_{\ell=1}^{m}\mathbf{K}_{t\ell} \\ &= \frac{1}{2}(\mathbf{K}_{ii}+\mathbf{K}_{jj}-\mathbf{D}^2_{ij}) - \frac{1}{2m}\sum_{t=1}^{m}\Big((\mathbf{K}_{ii}+\mathbf{K}_{tt}-\mathbf{D}^2_{it})+(\mathbf{K}_{tt}+\mathbf{K}_{jj}-\mathbf{D}^2_{tj}) - \frac{1}{m}\sum_{\ell=1}^{m}(\mathbf{K}_{tt}+\mathbf{K}_{\ell\ell}-\mathbf{D}^2_{t\ell})\Big) \\ &= \frac{1}{2}(-\mathbf{D}^2_{ij}) - \frac{1}{2m}\sum_{t=1}^{m}\Big((\mathbf{K}_{tt}-\mathbf{D}^2_{it}) - \mathbf{D}^2_{tj} - \frac{1}{m}\sum_{\ell=1}^{m}(\mathbf{K}_{\ell\ell}-\mathbf{D}^2_{t\ell})\Big) \\ &= \frac{1}{2}\Big(-\mathbf{D}^2_{ij} - \frac{1}{m}\sum_{t=1}^{m}(\mathbf{K}_{tt}-\mathbf{D}^2_{it}-\mathbf{D}^2_{tj}) + \frac{1}{m^2}\sum_{t=1}^{m}\sum_{\ell=1}^{m}(\mathbf{K}_{\ell\ell}-\mathbf{D}^2_{t\ell})\Big) \\ &= -\frac{1}{2}\Big(\mathbf{D}^2_{ij} - \frac{1}{m}\sum_{t=1}^{m}(\mathbf{D}^2_{it}+\mathbf{D}^2_{tj}) + \frac{1}{m^2}\sum_{t=1}^{m}\sum_{\ell=1}^{m}\mathbf{D}^2_{t\ell}\Big)\end{aligned}$$

Finally, we have

$$(\mathbf{\Delta}(\mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T))_{\ell j} = \Delta_{\ell j} - \frac{1}{m}\sum_{t=1}^{m}\Delta_{\ell t}$$

hence we may solve for $(\mathbf{H}\Delta\mathbf{H})_{ij}$ as

$$\begin{aligned}((\mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T)\mathbf{\Delta}(\mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T))_{ij} &= \Delta_{ij} - \frac{1}{m}\sum_{t=1}^{m}\Delta_{it} - \frac{1}{m}\sum_{\ell=1}^{m}(\Delta_{\ell j} - \frac{1}{m}\sum_{t=1}^{m}\Delta_{\ell t}) \\ &= -2\mathbf{K}^*_{ij} \Rightarrow \mathbf{K}^* = -\frac{1}{2}\mathbf{H}\mathbf{\Delta}\mathbf{H}\end{aligned}$$

□

6. Fenchel Game No-Regret Dynamics (FGNRD)

We now seek to understand such algorithms in the context of the Fenchel game no-regret dynamics framework (FGNRD) introduced by Wang-Abernethy-Levy.

Definition 6.1. For a function $f : \mathcal{K} \to \mathbb{R} \cup \infty$ where $\mathcal{K} \subset \mathbb{R}^d$, we define its conjugate $f^* : \mathbb{R}^d \to \mathbb{R} \cup \infty$ as

$$f^*(y) := \sup_{x \in D} \{\langle y, x\rangle - f(x)\}$$

Proposition 6.2. *Conjugate functions of convex functions are convex.*

Proof. For $f : \mathcal{K} \to \mathbb{R}$ convex where $\mathcal{K} \subset \mathbb{R}^d$, we find that

$$\begin{aligned} f^*(\lambda x + (1-\lambda)y) &= \sup_{x' \in \mathcal{K}} \{\langle x', \lambda x + (1-\lambda)y\rangle - f(x')\} \\ &= \sup_{x' \in \mathcal{K}} \{\langle x', \lambda x + (1-\lambda)y\rangle - f(x')\} \\ &= \sup_{x' \in \mathcal{K}} \{\langle x', \lambda x\rangle + \langle x', y\rangle - \lambda\langle x', y\rangle - f(x')\} \\ &= \sup_{x' \in \mathcal{K}} \{\lambda\langle x, x'\rangle - \lambda f(x') + \langle y, x'\rangle - f(x') - \lambda\langle y, x'\rangle + \lambda f(x')\} \\ &= \sup_{x' \in \mathcal{K}} \{\lambda(\langle x, x'\rangle - f(x')) + (1-\lambda)(\langle y, x'\rangle - f(x'))\} \\ &\leq \lambda \sup_{x' \in \mathcal{K}} \{\langle x, x'\rangle - f(x')\} + (1-\lambda) \sup_{x'' \in \mathcal{K}} \{\langle y, x''\rangle - f(x'')\} \\ &= \lambda f^*(x) + (1-\lambda) f^*(y) \end{aligned}$$

□

Definition 6.3. The subdifferential $\partial f(x)$ is the set of all subgradients of f at x, i.e.

$$\partial f(x) = \{f_x : f(z) \geq \langle f_x, z - x\rangle + f(x),\ \forall z \in \mathcal{K}\}$$

Theorem 6.4. *For a closed convex function $f : \mathbb{R}^d \to \mathbb{R}$, the following are equivalent:*

$$\textit{I. } y \in \partial f(x)$$
$$\textit{II. } x \in \partial f^*(y)$$
$$\textit{III. } \langle x, y\rangle = f(x) + f^*(y)$$

Proof. We first show I ⇒ II. Suppose $y \in \partial f(x)$. Then, for any $z \in \mathcal{K}$ we have

$$f(z) - f(x) \geq \langle y, z - x\rangle \Rightarrow \langle y, x\rangle - f(x) \geq \langle y, z\rangle - f(z)$$

so that

$$\langle y, x\rangle - f(x) \geq f^*(y)$$

Then,

$$\begin{aligned} \langle z, x\rangle - f(x) \leq f^*(z) &\Rightarrow \langle y, x\rangle - f(x) \leq f^*(z) - \langle x, z - y\rangle \\ &\Rightarrow f^*(y) \leq f^*(z) - \langle x, z - y\rangle \Rightarrow x \in \partial f^*(y) \end{aligned}$$

To show II ⇒ III, we find that, for any $z \in \mathcal{K}$

$$\langle x, y\rangle - f^*(y) \geq \langle x, z\rangle - f^*(z)$$

hence

$$\langle x, y\rangle - f^*(y) \geq f^{**}(x) = \sup_{y' \in \mathcal{K}} \langle x, y'\rangle - \sup_{x' \in \mathcal{K}} (\langle y', x'\rangle - f(x'))$$

so since f is closed, $\sup_{x'\in\mathcal{K}}(\langle y', x'\rangle - f(x'))$ is attained by some x' as

$$\geq \langle x, \nabla f(x)\rangle - \langle \nabla f(x), x'\rangle + f(x') = f(x') - \langle \nabla f(x), x' - x\rangle \geq f(x)$$
$$\Rightarrow f^*(y) \leq \langle x, y\rangle - f(x) \Rightarrow f^*(y) = \langle x, y\rangle - f(x)$$

Finally, III $\Rightarrow$ I as

$$\langle x, y\rangle \geq f(x) + f^*(y) \Rightarrow \langle x, y\rangle - f(x) \geq \langle y, z\rangle - f(z),\ \forall z \in \mathcal{K}$$
$$\Rightarrow f(z) - f(x) - \langle y, z - x\rangle \geq 0,\ \forall z \in \mathcal{K}$$

so that all three statements are equivalent. □

Definition 6.5. We define our two-input "payoff" function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as

$$g(x, y) := \langle x, y\rangle - f^*(y)$$

We will understand this function as a zero-sum game in which, if player 1 selects action x and player 2 selects action y, $g(x, y)$ is the "cost" for player 1 and the "gain" for player 2.

Definition 6.6. Given a zero-sum game with a payoff function $g(x, y)$ which is convex in x and concave in y, we define

$$V^* := \inf_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} g(x, y)$$

We further define an "ϵ-equilibrium" of $g(.,.)$ as a pair $\widehat{x}, \widehat{y}$ for which

$$V^* - \epsilon \leq \inf_{x\in\mathcal{X}} g(x, \widehat{y}) \leq V^* \leq \sup_{y\in\mathcal{Y}} g(\widehat{x}, y) \leq V^* + \epsilon$$

where $\mathcal{X}$ and $\mathcal{Y}$ are convex decision spaces of the x-player and y-player respectively.

Definition 6.7. To solve for $\inf_{x\in D} f(x)$, we define $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ as

$$g(x, y) := \langle x, y\rangle - f^*(y) = \langle x, y\rangle - \sup_{x'\in\mathcal{K}}\{\langle x', y\rangle - f(x')\}$$

and attempt to find an ϵ-equilibrium for $g(x, y)$.

Proposition 6.8. *An equilibrium for the Fenchel game function solves the minimization problem* $\inf_{x\in D} f(x)$.

Proof. For an ϵ-equilibrium $\widehat{x}, \widehat{y}$ of g defined as above, we have

$$\inf_{x\in\mathcal{K}} f(x) = -\sup_{x\in\mathcal{K}}\{-f(x)\} = -\sup_{x'\in\mathcal{K}}\{\langle x', y\rangle - \langle x', y\rangle - f(x')\} =: h(y)$$

so that

$$\inf_{x\in\mathcal{K}}\left\{\langle x, \widehat{y}\rangle - \sup_{x'\in\mathcal{K}}\{\langle x', \widehat{y}\rangle - f(x')\}\right\} \leq h(\widehat{y}) \leq \sup_{y\in\mathcal{Y}}\left\{\langle \widehat{x}, y\rangle - \sup_{x'\in\mathcal{K}}\{\langle x', y\rangle - f(x')\}\right\}$$

hence

$$(*) \quad |V^* - h(y)| \leq 2\epsilon$$

where

$$V^* = \inf_{x\in\mathcal{K}} \sup_{y\in\mathcal{Y}}\left\{\langle x, y\rangle - \sup_{x'\in\mathcal{K}}\{\langle x', y\rangle - f(x')\}\right\}$$

and as $\epsilon \to 0$ we have

$$V^* = \sup_{y\in\mathcal{Y}}\left\{\langle \widehat{x}, y\rangle - \sup_{x'\in\mathcal{K}}\{\langle x', y\rangle - f(x')\}\right\}$$
$$= \sup_{y\in\mathcal{Y}}\{\langle \widehat{x}, y\rangle - f^*(y)\} = f(\widehat{x})$$

which follows from Theorem 7.4. □

Corollary 6.9. *If $(\widehat{x}, \widehat{y})$ is an ϵ-equilibrium of the Fenchel Game as defined above, then*

$$|f(\widehat{x}) - \inf_{x \in \mathcal{K}} f(x)| \leq \epsilon$$

Proof. Follows from $(*)$ above for $\epsilon' := \frac{\epsilon}{2}$. □

Definition 6.10. "Online convex optimization" works as follows. At each round t (of T many), the learner selects a point $z_t \in \mathcal{Z}$ and suffers a loss $\alpha_t \ell_t(z_t)$ for this selection, where $\boldsymbol{\alpha}$ is the weight vector and $\mathcal{Z} \subset \mathbb{R}^d$ is a convex decision set of actions.

Remark 6.11. In general it is assumed that, upon selecting z_t during round t, the learner has observed all loss functions $\alpha_1 \ell_1(.), ..., \alpha_{t-1}\ell_{t-1}(.)$ up to but not including time t. An exception to this are the "prescient" learners (whose algorithms, marked with a "+" superscript, have access to the loss ℓ_t prior to selecting z_t) maintain knowledge of the t-th loss function.

Algorithm 3 Protocol for weighted online convex optimization

```
Require: convex decision set $\mathcal{Z} \subset \mathbb{R}^d$
Require: number of rounds $T$
Require: weights $\alpha_1, \alpha_2, ..., \alpha_T > 0$
Require: algorithm OAlg
  for $t = 1, 2, \ldots, T$ do
    Return: $z_t \leftarrow$ OAlg
    Receive: $\alpha_t, \ell_t(\cdot) \rightarrow$ OAlg
    Evaluate: Loss $\leftarrow$ Loss + $\alpha_t \ell_t(z_t)$
  end for
```

Remark 6.12. The "OAlg" referenced above refers to an algorithm performed within the current algorithm, and "OAlg^X" will refer to the algorithm updating the x coordinate in the Fenchel Game No Regret Dynamics.

Definition 6.13. We define a learner's "regret" as

$$\boldsymbol{\alpha}\text{-REG}^z(z^*) := \sum_{t=1}^{T} \alpha_t \ell_t(z_t) - \sum_{t=1}^{T} \alpha_t \ell_t(z^*)$$

where $z^* \in \mathcal{Z}$ is the "comparator" to which the online learner is compared. We further define "average regret" as that normalized by the time weight $A_T : \sum_{t=1}^{T} \alpha_t$ and denote it by

$$\overline{\boldsymbol{\alpha}\text{-REG}}^z(z^*) := \frac{\boldsymbol{\alpha}\text{-REG}^z(z^*)}{A_T}$$

Finally, "no-regret algorithms" guarantee $\overline{\boldsymbol{\alpha}\text{-REG}}^z(z^*) \rightarrow 0$ as $A_T \rightarrow \infty$

Remark 6.14. The following batch-style online-learning strategies modify the central algorithm Follow The Leader (FTL)

Algorithm 4 Online Learning Strategies

```
Require: convex set $\mathcal{Z}$, initial point $z_{\text{init}} \in \mathcal{Z}$
Require: $\alpha_1, ..., \alpha_T > 0,\ \ell_1, ..., \ell_T : \mathcal{Z} \to \mathbb{R}$

  FTL[$z_{\text{init}}$]:
    $z_t \leftarrow z_{\text{init}}$ if $t = 1$, else
    $z_t \leftarrow \text{argmin}_{z \in \mathcal{Z}} \Big( \sum_{s=1}^{t-1} \alpha_s \ell_s(z) \Big)$
  FTL$^+$:
    $z_t \leftarrow \text{argmin}_{z \in \mathcal{Z}} \Big( \sum_{s=1}^{t} \alpha_s \ell_s(z) \Big)$
  FTRL[$R(.), \eta$]:
    $z_t \leftarrow \text{argmin}_{z \in \mathcal{Z}} \Big( \sum_{s=1}^{t} \alpha_s \ell_s(z) + \frac{1}{\eta} R(z) \Big)$
```

Definition 6.15. A first-order oracle for a function $f : \mathbb{R}^n \to \mathbb{R}$ is a primitive that, given $x \in \mathbb{Q}^n$, outputs the value $f(x) \in \mathbb{Q}$ and a vector $h(x) \in \mathbb{Q}^n$ such that, for any $z \in \mathbb{R}^n$,

$$f(z) \geq f(x) + \langle h(x), z - x \rangle$$

so $h(x) = \nabla f(x)$ for f differentiable, else it is a subgradient of f at x.

We now examine online mirror descent and its prescient counterpart before recovering gradient descent from the fenchel game framework.

Definition 6.16. For fixed $\epsilon > 0$ and norm $||.||$, a differentiable function $f : \mathcal{K} \to \mathbb{R}$ where $\mathcal{K} \subset \mathbb{R}^d$ is convex, is considered "ϵ-strongly convex with respect to $||.||$" if

$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle + \frac{\epsilon}{2} ||y - x||^2$$

Definition 6.17. We define the Bregman divergence $D_z^\phi(\cdot)$ centered at z with respect to a β-strongly convex distance generating function $\phi(\cdot)$ as

$$D_z^\phi(x) := \phi(x) - \langle \nabla \phi(z), x - z \rangle - \phi(z)$$

Algorithm 5 Update-style online learning strategies

```
Require: convex set $\mathcal{Z}$, initial point $z_0 \in \mathcal{Z}$
Require: $\alpha_1, ..., \alpha_T > 0,\ \ell_1, ..., \ell_T : \mathcal{Z} \to \mathbb{R}$

  OMD[$\phi(\cdot), z_0, \gamma$]:
    $z_t \leftarrow \text{argmin}_{z \in \mathcal{Z}} \Big( \alpha_{t-1} \ell_{t-1}(z) + \frac{1}{\gamma} D^\phi_{z_{t-1}}(z) \Big)$
  OMD$^+$[$\phi(\cdot), z_0, \gamma$]:
    $z_t \leftarrow \text{argmin}_{z \in \mathcal{Z}} \Big( \alpha_t \ell_t(z) + \frac{1}{\gamma} D^\phi_{z_{t-1}}(z) \Big)$
```

Remark 6.18. To introduce the FGNRD framework, we show vanilla gradient descent can be understood as a two player Fenchel game. For $f(\cdot)$ convex, let

$$G = \max_{y \in \partial f(w),\ w \in \mathcal{K}} ||y||^2$$

and let $R \in \mathbb{R}$ be an upper bound to $||w_0 - w^*||$ where $w^* := \text{argmin}_{w\in\mathcal{K}} f(w)$.

Algorithm 6 Vanilla gradient descent and its FGNRD equivalent

```
Require: Convex function $f(\cdot)$ and iterations $T$
  Initialize: $w_0 = x_0 \in \mathcal{K} \subseteq \mathbb{R}^d$
  Initialize: $\gamma = \begin{cases} \frac{R}{G\sqrt{T}}, & \text{if } f(\cdot) \text{ is non-smooth} \\ \frac{1}{2L}, & \text{if } f(\cdot) \text{ is } L\text{-smooth} \end{cases}$
  Gradient Descent[$w_0$]:
    $w_t := w_{t-1} - \gamma\delta_{t-1}$ for $\delta_{t-1} \in \partial f(w_{t-1})$
    $\overline{w}_t := \frac{1}{t}\sum_{s=1}^t w_s$
  FGNRD Equivalent:
    $g(x, y) := \langle x, y\rangle - f^*(y)$
    $\alpha_t := 1$ for $t \in \{1, ..., T\}$
    $\text{OAlg}^X := \text{OMD}[\frac{1}{2}||\cdot||_2^2, x_0, \gamma]$
    $\text{OAlg}^Y := \text{BESTRESP}^+$
```

Remark 6.19. Note that the use of Bregman divergence in the FGNRD Equivalent (which helps specify the subgradient y_t selected) includes the Bregman divergence term $\frac{1}{\gamma}D^{\phi}_{z_{t-1}}(z)$ as in Algorithm 5 to ensure we remain within a neighborhood of our previous iteration upon descent.

Theorem 6.20. *The FGNRD formulation is equivalent to vanilla gradient descent, i.e.* $w_t = x_t$ *at every time step, hence* $\overline{w}_t = \frac{1}{t}\sum_{s=1}^t w_s = \frac{1}{t}\sum_{s=1}^t x_s = \overline{x}_t$.

Proof. We find that

$$\begin{aligned}\text{OAlg}^x &:= \text{argmin}_{x\in\mathcal{K}} \alpha_{t-1}\ell_{t-1}(x) + \frac{G\sqrt{T}}{R}\Big(\frac{1}{2}||x||_2^2 - \frac{1}{2}||x_{t-1}||_2^2 - \langle x_{t-1}, x - x_{t-1}\rangle\Big) \\ &= \text{argmin}_{x\in\mathcal{K}} \langle x, y_{t-1}\rangle - f^*(y) + \frac{G\sqrt{T}}{R}\Big(\frac{1}{2}||x||_2^2 - \langle x_{t-1}, x - x_{t-1}\rangle\Big) \\ &= \text{argmin}_{x\in\mathcal{K}} \langle x, y_{t-1} - \frac{G\sqrt{T}}{R}x_{t-1}\rangle + \frac{G\sqrt{T}}{2R}||x||_2^2 := \text{argmin}_{x\in\mathcal{K}} F(x)\end{aligned}$$

so since $\mathcal{K}$ is convex, the minimum is reached when $\nabla F = \mathbf{0}$, i.e.

$$\mathbf{0} = y_{t-1} - \frac{G\sqrt{T}}{R}x_{t-1} + \frac{G\sqrt{T}}{R}x_t \Rightarrow x_t = x_{t-1} - \frac{R}{G\sqrt{T}}y_{t-1}$$

Now it suffices to show $y_t \in \partial f(x_t)$:

$$\begin{aligned}y_t &= \text{argmin}_{y\in\mathcal{K}} - (\alpha_t\langle x_t, y\rangle - f^*(y)) \\ &= \text{argmax}_{y\in\mathcal{K}}(\langle x_t, y\rangle - f^*(y))\end{aligned}$$

Then, since, for any $z \in \mathcal{K}$ we have

$$\langle x_t, y_t\rangle - f^*(y_t) \geq \langle x_t, z\rangle - f^*(z)$$

$$\Rightarrow f^*(z) \geq f^*(y_t) + \langle x_t, z - y_t\rangle \Rightarrow x_t \in \partial f^*(y_t)$$

By Theorem 6.4 we thus have

$$y_t \in \partial f(x_t)$$

□

Lemma 6.21. *Let $\phi(.)$ be a β-strongly convex function with respect to the norm $||\cdot||_*$, and consider a sequence of lower semi-continuous convex loss functions $\{\alpha_t \ell_t(.)\}_{t=1}^T$. Then, for any comparator $z^* \in \mathcal{Z}$, OMD$[\phi(.), z_0, \gamma]$ satisfies*

$$\boldsymbol{\alpha}\text{-}REG^z(z^*) \leq \frac{1}{\gamma} D^{\phi}_{z_1}(z^*) + \frac{\gamma}{2\beta}\sum_{t=1}^{T} ||\alpha_t \delta_t||_*^2$$

for $\delta_t \in \partial \ell_t(z_t)$

Proof. We wish to show that

$$\alpha_t \ell_t(\text{argmin}_{z\in\mathcal{Z}}\{\alpha_{t-1}\ell_{t-1}(z) + D^{\phi}_{z_{t-1}}(z)\}) - \alpha_t \ell_t(z^*) \leq \frac{1}{\gamma} D^{\phi}_{z_1}(z^*) + \frac{\gamma}{2\beta}||\alpha_t \delta_t||_*^2$$

By the minimality of $\alpha_{t-1}\ell_{t-1}(z_{t+1}) + D^{\phi}_{z_{t-1}}(z_{t+1})$ we have

$$\alpha_t(\ell_t(z_{t+1}) - \ell_t(z^*)) \leq \frac{1}{\gamma}(\phi(z^*) - \phi(z_{t+1}) + \langle \nabla\phi(z_t), z_{t+1} - z^*\rangle)$$

hence

$$(*) \quad \langle \alpha_t \delta_t, z_{t+1} - z^*\rangle \leq \langle \frac{1}{\gamma}(\nabla\phi(z_t) - \nabla\phi(z_{t+1})), z_{t+1} - z^*\rangle$$

since the convexity of ℓ_t ensures the existence of a $\delta_t \in \partial\ell_t(x_t)$ with $\delta_t \in \partial\ell_t(z^*)$.

We now find that

$$\alpha_t \ell_t(z_t) - \alpha_t \ell_t(z^*) \leq \langle \alpha_t \delta_t, z_t - z^*\rangle$$

$$= \langle \frac{1}{\gamma}(\nabla\phi(z_{t+1}) - \nabla\phi(z_t)), z^* - z_{t+1}\rangle + \langle \frac{1}{\gamma}(\nabla\phi(z_t) - \nabla\phi(z_{t+1})) - \alpha_t\delta_t, z^* - z_{t+1}\rangle$$

$$+\langle \alpha_t \delta_t, z_t - z_{t+1}\rangle$$

so by $(*)$ we have

$$\leq \langle \frac{1}{\gamma}(\nabla\phi(z_{t+1}) - \nabla\phi(z_t)), z^* - z_{t+1}\rangle + \langle \alpha_t\delta_t, z_t - z_{t+1}\rangle$$

$$= \frac{1}{\gamma}(D^{\phi}_{z_t}(z^*) - D^{\phi}_{z_{t+1}}(z^*) - D^{\phi}_{z_t}(z_{t+1})) + \langle \alpha_t\delta_t, z_t - z_{t+1}\rangle$$

Then, since $\langle \alpha_t \nabla\ell_t(z_t), z_t - z_{t+1}\rangle \leq \frac{\gamma}{2\beta}||\alpha_t\delta_t||_*^2 + \frac{\beta}{2\gamma}||z_t - z_{t+1}||^2$,

$$\leq \frac{1}{\gamma}(D^{\phi}_{z_t}(z^*) - D^{\phi}_{z_{t+1}}(z^*)) + \frac{\gamma}{2\beta} D^{\phi}_{z_t}(z^*)$$

We now sum from $t = 1$ to $t = T$ and the result follows. □

Lemma 6.22. *Algorithm 6 satisfies $f(\overline{w}_T) - \min_{w\in\mathcal{K}} f(w) = O(\frac{GR}{\sqrt{T}})$.*

Proof. By Lemma 6.17, OAlg^X satisfies

$$\boldsymbol{\alpha}\text{-REG}^x(x^*) \le \frac{1}{\gamma} D^{\phi}_{x_0}(x^*) + \frac{\gamma}{2}\sum_{t=1}^{T} ||\alpha_t y_t||_2^2$$

while OAlg^Y suffers no regret (prescient). Further, since

$$||y||_2^2 - ||x||_2^2 - 2\langle x, y-x\rangle = ||y-x||_2^2$$

we have that $\phi(x) = \frac{1}{2}||x||_2^2$ is 1-strongly convex with respect to the Euclidean norm $||\cdot||_2$, hence

$$\begin{aligned}\overline{\boldsymbol{\alpha}\text{-REG}}^x[\text{OMD}] + \overline{\boldsymbol{\alpha}\text{-REG}}^x[\text{BESTRESP}^+] &\le \frac{1}{A_t}\left(\frac{1}{\gamma}D^{\phi}_{x_0}(x^*) + \sum_{t=1}^{T}\frac{\gamma}{2}||\alpha_t y_t||^2\right)\\ &\le \frac{1}{T}\Big(\frac{R^2}{\gamma} + \frac{\gamma T G^2}{2}\Big)\\ &= \frac{1}{T}\Big(RG\sqrt{T} + \frac{RG\sqrt{T}}{2}\Big)\\ &= O\Big(\frac{GR}{\sqrt{T}}\Big)\end{aligned}$$

□

Acknowledgments

I would like to thank my mentor, Antares Chen, for helping direct my research throughout the summer. I would also like to thank Peter May for organizing this REU.

7. Bibliography

References

[1] http://www.ams.org/publications/authors/tex/amslatex

[2] Jun-Kun Wang, Jacob Abernethy, Kfir Y. Levy. No-Regret Dynamics in the Fenchel Game: A Unified Framework for Algorithmic Convex Optimization https://arxiv.org/abs/2111.11309

[3] Mehryar Mohri. Foundations of Machine Learning (second edition) MIT Press. 2018.

[4] Wei Jiang, Nan Li, Hongpeng Yin and Yi Chai. An Improved Laplacian Eigenmaps Algorithm for Nonlinear Dimensionality Reduction. Proceedings of the 2015 Chinese Intelligent Systems Conference