Fast Label Embeddings via Randomized Linear Algebra

Author: Paul Mineiro and Nikos Karampatziakis
Presented By: Sriteja Reddy Pashya and Romica Raisinghani

MA8.401 Topics in Applied Optimization

5^{th} December 2023

Motivation and Problem Statement

- Modern multiclass and multilabel problems are characterized by increasingly large output spaces.
- The goal is to improve both computational and statistical efficiency in dealing with these problems.
- The authors propose the use of label embeddings to tackle problems with large output spaces.
- A fast label embedding algorithm is introduced that works in both the multiclass and multilabel settings.
- The algorithm uses techniques from randomized linear algebra to develop an efficient and scalable method for constructing the embeddings.
- The techniques are demonstrated on two large-scale public datasets, where they obtain state-of-the-art results.

Algorithm Notations

Let us have a look at some of the notations that will be used in the algorithm:

- Vectors are denoted by lowercase letters $x, y, ...$
- Matrices are denoted by uppercase letters $W, W, ...$
- Input dimension: d
- Output dimension: c
- Embedding dimension: k
- $X \in \mathbb{R}^{m \times n}$ denotes an $m \times n$ matrix.
- $\|X\|_F$ represents the Frobenius norm of matrix X.

For a given matrix A with dimensions $m \times n$, Frobenius norm is computed as follows:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Algorithm Notations Continued

- For multiclass problems, y is a one-hot vector.
- For multilabel problems, y is a binary vector.
- $X^{\dagger}$ denotes the pseudoinverse of matrix X.
- $\Pi_{X,L}$ represents the projection onto the left singular subspace of X.
- $X_{1:k}$ denotes the matrix obtained by taking the first k columns of X.
- X^{*} denotes a matrix obtained by solving an optimization problem over the matrix parameter X.
- The expectation of a random variable v is denoted by $\mathbb{E}[v]$.

Regular PCA Algorithm

Algorithm 1 Regular PCA

function $\mathrm{PCA}(k, X \in \mathbb{R}^{n \times d})$

$C \leftarrow X^T X$ {Compute the covariance matrix}

$(U, \Sigma, V^T) \leftarrow \mathrm{SVD}(C)$ {Singular Value Decomposition}

$U_k \leftarrow U[:, 1:k]$ {Select the first k columns of U}

$\Sigma_k \leftarrow \Sigma[1:k, 1:k]$ {Select the top k x k block of Σ}

$V_k^T \leftarrow V^T[1:k, :]$ {Select the first k rows of V^T}

$Y \leftarrow XV_k$ {Project X onto the space spanned by the top k eigenvectors}

return (Y, Σ_k)

end function

Randomized PCA Algorithm

Algorithm 2 Randomized PCA

function $\mathrm{RPCA}(k, X \in \mathbb{R}^{n \times d})$

$(p, q) \leftarrow (20, 1)$ {Hyperparameters}

$Q \leftarrow \mathrm{randn}(d, k + p)$

for $i \in \{1, \ldots, q\}$ **do**

$\Psi \leftarrow X^T X Q$ {Ψ computed in one pass}

$Q \leftarrow \mathrm{orthogonalize}(\Psi)$

end for

$F \leftarrow (X^T X Q)^T (X^T X Q)$

$(V, \Sigma^2) \leftarrow \mathrm{eig}(F, k)$

$V \leftarrow (X^T X Q) V \Sigma^\dagger$

return (V, Σ)

end function

Explanation: RPCA Algorithm

- Algorithm starts with a function RPCA that takes two parameters i.e k(number of desired principal components) and input matrix X(containing n examples and d features).
- Hyper-parameters p and q rarely need adjustment. p is used to oversample the number of random vectors to ensure a good range approximation, and q is the number of times the range finding loop will run.
- Generate a matrix Q of dimensions $d \times (k + p)$ filled with random numbers. This matrix is used to probe the range of the matrix $X^T X$.
- Algorithm runs a loop q times. In each iteration:
  1. Compute the matrix ψ as the product of $X^T X$ and Q.
  2. Update Q to an orthogonal basis for the range of ψ.

Explanation: RPCA Algorithm (Continued)

- After completing the q iterations, compute the matrix F as the product of $(X^T XQ)^T$ and $(X^T XQ)$. The matrix F is of size $(k+p) \times (k+p)$ and is small relative to the original data matrix.
- Perform eigendecomposition on the small matrix F to obtain the top k eigenvectors and eigenvalues. The eigenvectors are stored in V, and the eigenvalues (squared) are stored in $\sum^2$.
- Multiply $(X^T XQ)$ with V and scale by the pseudoinverse of $\sum$ (denoted $\sum^\dagger$). This step backs out the solution to find the approximate principal components.
- Return the matrices V (eigenvectors) and $\sum$ (square root of the eigenvalues), which represent the principal components and their associated variances.

Rembrandt Algorithm

Algorithm 3 : **R**esponse **EMB**edding via **RAND**omized **T**echniques

function REMBRANDT($k, X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^{n \times c}$)

$(p, q) \leftarrow (20, 1)$ {Hyperparameters}

$Q \leftarrow \text{randn}(c, k + p)$

for $i \in \{1, \ldots, q\}$ **do**

$Z \leftarrow \text{argmin} \| YQ - XZ \|_F^2$

$Q \leftarrow \text{orthogonalize}(YXZ)$

end for

$F \leftarrow (YXZ)(YXZ)^T$

$(V, \Sigma^2) \leftarrow \text{eig}(F, k)$

$V \leftarrow (YXZ)V\Sigma^\dagger$

return (V, Σ)

end function

Optimal Squared Loss Predictor with Low-Rank Constraint

Problem Formulation

We aim to find a weight matrix W^* for predicting a high-cardinality target vector $y \in \mathbb{R}^c$ from a high-dimensional feature vector $x \in \mathbb{R}^d$. The challenge is to minimize the squared loss, $\|Y - XW\|_F^2$, under a low-rank constraint $\text{rank}(W) \leq k$, where Y and X are the target and design matrices, respectively.

Solution via SVD

The solution W^* is obtained through the projection $\Pi_{X,L}$ onto the left singular subspace of X, and involves the optimal Frobenius norm rank-k approximation. The expression for W^* is derived using SVD, simplifying to $W^* = X^\dagger (YV_{1:k}) V_{1:k}^T$.

Reference

Friedland, S., Torokhti, A.: Generalized rank-constrained matrix approximations. SIAM Journal on Matrix Analysis and Applications 29(2), 656–659 (2007).

Rembrandt Algorithm: Computation Steps

The Rembrandt algorithm computes an optimal weight matrix W^* using a three-step procedure:

1. **Projection**: It projects the target matrix Y onto k dimensions using the top right singular vectors of $\Pi_{X,L} Y$, where $\Pi_{X,L}$ is the projection onto the left singular subspace of X.
2. **Least Squares Fit**: Then it fits the projected labels to the features using a least squares fit, which is a standard approach for finding the best-fit linear model that minimizes the sum of the squares of the errors.
3. **Mapping Predictions**: Finally, it maps these predictions back to the original output space using the transpose of the top k right singular vectors of $\Pi_{X,L} Y$.

This process efficiently approximates $W^* =$ without having to compute it directly, thus saving on computational resources while still capturing the essential structure of the data.

Label Embedding and Randomized Algorithms

The right singular vectors of $\Pi_{X,L} Y$ are utilized for label embedding, motivated by the predictions of the optimal unconstrained model:

$$Z^* = \arg \min_{Z \in \mathbb{R}^{d \times c}} \| Y - XZ \|_F^2,$$

$$\Pi_{X,L} Y = XZ^* \equiv \hat{Y}.$$

These vectors, V, of $\Pi_{X,L} Y$ are the eigenvectors of $\hat{Y}^T \hat{Y}$, the matrix of outer products of the model's predictions. Avoiding the direct computation of Z^*, the algorithm finds $\Pi_{X,L} YQ = XZ^* Q$ by solving:

$$Z^* Q = \arg \min_{Z \in \mathbb{R}^{d \times k}} \| YQ - XZ \|_F^2.$$

Bias-Variance Tradeoff and Model Generalization

Squared Loss Minimization

Squared loss, as a proper scoring rule, is minimized at the conditional mean. With sufficient data and model flexibility, $\frac{1}{n}\hat{Y}^T\hat{Y}$ converges to $\mathbb{E}[\mathbb{E}[y|x]^T\mathbb{E}[y|x]]$.

Law of Large Numbers

This convergence is assured by the strong law of large numbers.

$$\frac{1}{n}\hat{Y}^T\hat{Y} \xrightarrow{a.s.} \mathbb{E}[\mathbb{E}[y|x]^T\mathbb{E}[y|x]] \tag{1}$$

Eigendecomposition and Embedding Insights

Eigendecomposition

An embedding based on the eigendecomposition of $\mathbb{E}[\mathbb{E}[y|x]^T\mathbb{E}[y|x]]$ is not actionable but insightful, approximating it by the empirical label covariance Y^TY.

Generalization in High-Dimensional Spaces

For multiclass or multilabel cases, the low-rank constraint might not ensure good generalization if the model is overly flexible. The eigendecomposition may only capture the most frequent labels, missing out on label co-occurrence patterns.

Model Tuning

To better approximate $\mathbb{E}[Y|X]$ over the observed Y, one must trade off variance for bias, tuning the bias-variance tradeoff by the choice of model features, as used in Algorithm 2.

Conclusion

- The proposed label embedding techniques significantly improve computational and statistical efficiency in large output space problems.
- Experiments on large-scale datasets demonstrate that the Rembrandt algorithm coupled with logistic regression outperforms other methods.
- The results highlight the potential of label embeddings in dealing with multiclass and multilabel problems, achieving state-of-the-art performance.

THANK YOU !