Machine Learning
https://doi.org/10.1007/s10994-021-06025-3

S.I. : ECML PKDD 2021

AgFlow: fast model selection of penalized PCA via implicit regularization effects of gradient flow

Haiyan Jiang[1] · Haoyi Xiong[1] · Dongrui Wu[2] · Ji Liu[1] · Dejing Dou[1]

Received: 22 November 2020 / Revised: 12 April 2021 / Accepted: 16 June 2021

Abstract

Principal component analysis (PCA) has been widely used as an effective technique for feature extraction and dimension reduction. In the High Dimension Low Sample Size setting, one may prefer modified principal components, with penalized loadings, and automated penalty selection by implementing model selection among these different models with varying penalties. The earlier work (Zou et al. in J Comput Graph Stat 15(2):265–286, 2006; Gaynanova et al. in J Comput Graph Stat 26(2):379–387, 2017) has proposed penalized PCA, indicating the feasibility of model selection in ℓ_2-penalized PCA through the solution path of Ridge regression, however, it is extremely time-consuming because of the intensive calculation of matrix inverse. In this paper, we propose a fast model selection method for penalized PCA, named approximated gradient flow (`AgFlow`), which lowers the computation complexity through incorporating the implicit regularization effect introduced by (stochastic) gradient flow (Ali et al. in: The 22nd international conference on artificial intelligence and statistics, pp 1370–1378, 2019; Ali et al. in: International conference on machine learning, 2020) and obtains the complete solution path of ℓ_2-penalized PCA under varying ℓ_2-regularization. We perform extensive experiments on real-world datasets. `AgFlow` outperforms existing methods (Oja and Karhunen in J Math Anal Appl 106(1):69–84, 1985; Hardt and Price in: Advances in neural information processing systems, pp 2861–2869, 2014; Shamir in: International conference on machine learning, pp 144–152, PMLR, 2015; and the vanilla Ridge estimators) in terms of computation costs.

Keywords Model selection · Gradient flow · Implicit regularization · Penalized PCA · Ridge

Editors: Annalisa Appice, Sergio Escalera, Jose A. Gamez, Heike Trautmann.

✉ Haoyi Xiong
xionghaoyi@baidu.com

[1] Big Data Lab, Baidu Research, Beijing, China

[2] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China

Published online: 07 July 2021

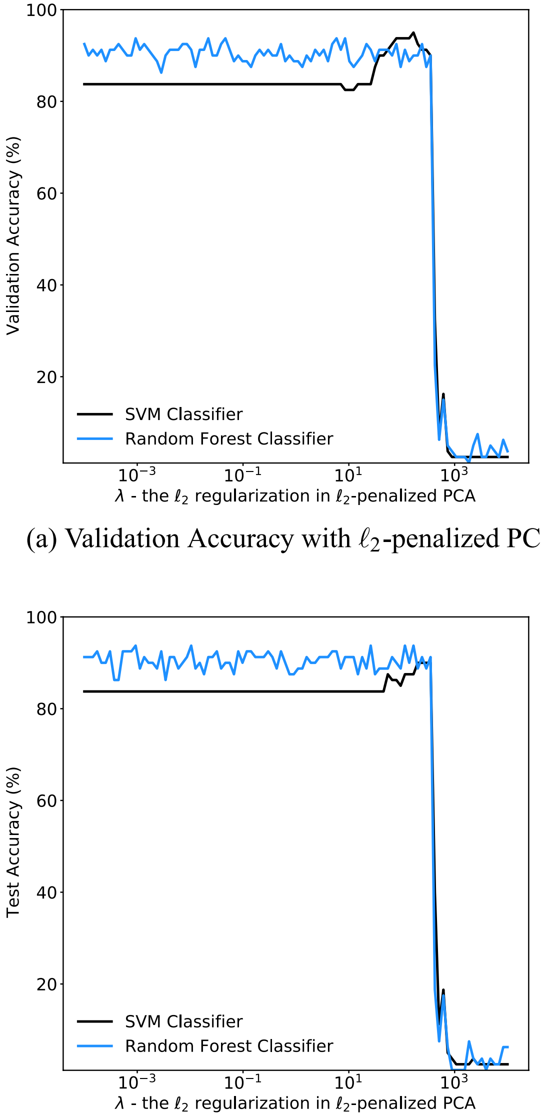
1 Introduction

Principal component analysis (PCA) (Jolliffe 1986; Dutta et al. 2019) is widely used as an effective technique for feature transformation, data processing and dimension reduction in unsupervised data analysis, with numerous applications in machine learning and statistics such as handwritten digits classification (LeCun et al. 1995; Hastie et al. 2009), human faces recognition (Huang et al. 2008; Mohammed et al. 2011), and gene expression data analysis (Yeung and Ruzzo 2001; Zhu et al. 2007). Generally, given a data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, where n refers to the number of samples and d refers to the number of variables in each sample, PCA can be formulated as a problem of projecting samples to a lower d' -dimensional subspace ($d' \ll d$) with variances maximized. To achieve the goal, numerous algorithms, such as Oja's algorithm (Oja and Karhunen 1985), power iteration algorithm (Hardt and Price 2014), and stochastic/incremental algorithm (Shamir et al. 2015; Arora et al. 2012; Mitliagkas et al. 2013; De Sa et al. 2015) have been proposed, and the convergence behaviors of these algorithms have also been intensively investigated. In summary, given the matrix of raw data samples, the eigensolvers above output d'-dimensional vectors which are linear combinations of the original predictors, projecting original samples into the d'-dimensional subspace desired while capturing maximal variances.

In addition to the above estimates of PCA, penalized PCA has been proposed (Zou et al. 2006; Irina et al. 2017; Witten et al. 2009; Lee et al. 2012) to improve its performance using regularization. For example, Zou et al. (2006) introduced a direct estimation of ℓ_2 -penalized PCA using Ridge estimator (see also in Theorem 1 in Section 3.1 of (Zou et al. 2006)), where an ℓ_2-regularization hyper-parameter (denoted as λ) has been used to balance the error term for fitting and the penalty term for regularization. Whereas an ℓ_1-regularization is usually introduced for achieving sparsity (Irina et al. 2017). Though the effects of λ in ℓ_2-penalized PCA would be waived by normalization when the sample covariance matrix is non-singular (i.e., $d < n$), the penalty term indeed regularizes the sample covariance matrix (Witten et al. 2009) for a stable inverse under High Dimension Low Sample Size (HDLSS) settings. Therefore, it is desirably necessary to do model selection to get the optimal λ on the solution path, where the solution path is formed by all the solutions corresponding to all the candidate λ-s in the ℓ_2-penalized problem, and each model is determined by the parameter λ. Thus, given the datasets for training and validation, it is only needed to retrieve the complete solution path (Friedman et al. 2010; Zou and Hastie 2005) for penalized PCA using the training dataset, where each solution corresponds to an Ridge estimator, and iterate every model on the solution path for validation and model selection. An example of ℓ_2-penalized PCA for dimension reduction over the solution path is listed in Fig. 1, where we can see that both the validation and testing accuracy are heavily affected by the value of λ, the ℓ_2 regularization in ℓ_2-penalized PCA, no matter what kind of classifier is employed after dimension reduction.

While a solution path for penalized PCA is highly required, the computational complexity of estimating a large number of models using grid searching of hyper-parameters is usually unacceptable. Specifically, to obtain the complete solution path or the models for ℓ_2-penalized PCA, it is needed to repeatedly solve the Ridge estimator with a wide range of values for λ, where the matrix inverse to get a shrunken sample covariance matrix is required in $O(d^3)$ complexity for every possible setting of λ. To lower the complexity, inspired by the recent progress on implicit regularization effects of gradient descent (GD) and stochastic gradient descent (SGD) in solving Ordinary Least-Square (OLS) problems (Ali et al. 2019, 2020), we propose a fast model selection method, named Approximated

Fig. 1 A Running Example of Performance Tuning for Classification with ℓ_2-penalized PCA Dimension-Reduced Data. We reduce the dimension of FACES dataset (Huang et al. 2008) from $d = 4096$ to $d' = 20$, using ℓ_2 -penalized PCA (Ridge-based), and try to select models over the solution path of varying λ for classification tasks. The model (i.e., the ℓ_2-penalized PCA estimation with a fixed λ) that achieves better performance in validation set empirically works better on the testing set



(a) Validation Accuracy with ℓ_2-penalized PCA

(b) Testing Accuracy with ℓ_2-penalized PCA

Gradient Flow (`AgFlow`), which is an efficient and effective algorithm to accelerate model selection of ℓ_2-penalized PCA with varying penalties.

Our contributions. We make three technical contributions as follows.

- We study the problem of lowering the computational complexity while accelerating the model selection for penalized PCA under varying penalties, where we particularly pay attention to ℓ_2-penalized PCA via the commonly-used Ridge estimator (Zou et al. 2006) under High Dimension Low Sample Size (HDLSS) settings.

- We propose AgFlow to do fast model selection in ℓ_2-penalized PCA with $O(K \cdot d^2)$ complexity, where K refers to the total number of models estimated for selection, and d is the number of dimension. More specifically, AgFlow first adopts algorithms in Shamir et al. (2015) to sketch d' principal subspaces, then retrieves the complete solution path of the corresponding loadings for every principal subspace. Especially, AgFlow incorporates the implicit ℓ_2-regularization of approximated (stochastic) gradient flow over the Ordinary Least Squares (OLS) to screen and validate the ℓ_2-penalized loadings, under varying λ from $+\infty \rightarrow 0^+$, using the training and validation datasets respectively.
- We conduct extensive experiments to evaluate AgFlow, where we compare the proposed algorithm with vanilla PCA, including (Hardt and Price 2014; Shamir et al. 2015; Oja and Karhunen 1985) and ℓ_2-penalized PCA via Ridge estimator (Zou et al. 2006) on real-world datasets. Specifically, the experiments are all based on HDLSS settings, where a limited number of high-dimensional samples have been given for PCA estimation and model selection. The results showed that the proposed algorithm can significantly outperform the vanilla PCA algorithms (Hardt and Price 2014; Shamir et al. 2015; Oja and Karhunen 1985) with better performance on validation/testing datasets gained by the flexibility of performance tuning (i.e., model selection). On the other hand, AgFlow consumes even less computation time to select models from 50 times more models compared to Ridge-based estimator (Zou et al. 2006).

Note that we don't intend to propose the "off-the-shelf" estimators to reduce the computational complexity of PCA estimation. Instead, we study the problem of model selection for ℓ_2-regularized PCA, where we combine the existing algorithms (Zou et al. 2006; Ali et al. 2019; Shamir et al. 2015) to lower the complexity of model selection and accelerate the procedure. The unique contribution made here is to incorporate with the novel continuous-time dynamics of gradient descent (gradient flow) Ali et al. (2019, 2020) to obtain the time-varying implicit regularization effects of ℓ_2-type for PCA model selection purposes.

Notations The following key notations are used in the rest of the paper. Let $\mathbf{x} \in \mathbb{R}^d$ be the d-dimensional predictors and $y \in \mathbb{R}$ be the response, and denote $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top = [X_1, \ldots, X_d]$ and $Y = [y_1, \ldots, y_n]^\top$, where n is the sample size and d is the number of variables. Without loss of generality, assume X_j, $j = 1, \ldots, d$ and Y are centered. Given a d-dimensional vector $\mathbf{z} \in \mathbb{R}^d$, denote the ℓ_2 vector-norm $\|\mathbf{z}\|_2 = \left(\sum_{i=1}^d |z_i|^2\right)^{1/2}$.

2 Preliminaries

In this section, firstly the Ordinary Least Squares and Ridge Regression is briefly introduced, then followed with the formulation that PCA is rewritten as a regression-type optimization problem with an explicit ℓ_2 regularization parameter λ, and lastly ended up with the introduction of the implicit regularization effect introduced by the (stochastic) gradient flow.

2.1 Ordinary least squares and ridge regression

Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $Y \in \mathbb{R}^n$ be a matrix of predictors (or features) and a response vector, respectively, with n observations and d predictors. Assume the columns of $\mathbf{X}$ and Y are centered. Consider the ordinary least squares (linear) regression problem

$$\hat{\beta}_{\text{OLS}} = \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{2n} \|Y - \mathbf{X}\beta\|_2^2 . \tag{1}$$

To enhance the solution of OLS for linear regression, regularization is commonly used as a popular technique in optimization problems in order to achieve a sparse solution or alleviate the multicollinearity problem (Friedman et al. 2010; Zou and Hastie 2005; Tibshirani 1996; Fan and Li 2001; Yuan and Lin 2006; Candes and Tao 2007). Recently an enormous amount of literature has focused on the related regularization methods, such as the lasso (Tibshirani 1996) which is friendly to interpretability with a sparse solution, the grouped lasso (Yuan and Lin 2006) where variables are included or excluded in groups, the elastic net (Zou and Hastie 2005) for correlated variables which compromises ℓ_1 and ℓ_2 penalties, the Dantzig selector (Candes and Tao 2007) which serves as a slightly modified version of the lasso, and some variants (Fan and Li 2001).

The ridge regression is the ℓ_2-regularized version of the linear regression in Eq. (1), imposing an explicit ℓ_2 regularization on the coefficients (Hoerl and Kennard 1970; Hoerl et al. 1975). Thus, the ridge estimator $\hat{\beta}_{\text{ridge}}(\lambda)$, a penalized least squares estimator, can be obtained by minimizing the ridge criterion

$$\hat{\beta}_{\text{ridge}}(\lambda) = \arg\min_{\beta \in \mathbb{R}^d} \left\{ \frac{1}{2n} \|Y - \mathbf{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right\}. \tag{2}$$

The solution of the ridge regression has an explicit closed-form,

$$\hat{\beta}_{\text{ridge}}(\lambda) = (\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I})^{-1} \mathbf{X}^\top Y. \tag{3}$$

We can see that the ridge estimator, Eq. (3), applies a type of shrinkage in comparison to the OLS solution $\hat{\beta}_{\text{OLS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top Y$, which shrinks the coefficients of correlated predictors towards each other and thus alleviates the multicolinearity problem.

2.2 PCA as ridge regression

PCA can be formulated as a regression-type optimization problem which was first proposed by Zou et al. (2006), where the loadings could be recovered by regressing the principal components on the d variables given the principal subspace.

Consider the j^{th} principal component. Let $\mathbf{Y}^j$ be a given $n \times 1$ vector referring to the estimate of the j^{th} principal subspace. For any $\lambda \geq 0$, the Ridge-based estimator (Theorem 1 in Zou et al. (2006)) of ℓ_2-penalized PCA is defined as

$$\bar{\beta}^j(\lambda) = \arg\min_{\beta \in \mathbb{R}^d} \left\{ \frac{1}{2n} \left\| \mathbf{Y}^j - \mathbf{X}\beta \right\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right\}. \tag{4}$$

Obviously, the estimator above highly depends on the estimate of the principal subspace $\mathbf{Y}^j$. Given the original data matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$, we could obtain its singular value decomposition as $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ and the estimate of subspace could be $\mathbf{Y}^j = \mathbf{U}_j \mathbf{S}_j$, where $\mathbf{U}_j$ and $\mathbf{S}_j$ refers to the j^{th} columns of the corresponding matrices, respectively. Then the normalized vector can be used as the penalized loadings of the j^{th} principal component

$$\hat{\beta}^j(\lambda) = \frac{\bar{\beta}^j(\lambda)}{\|\bar{\beta}^j(\lambda)\|_2}. \tag{5}$$

Note that, when the sample covariance matrix $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$ is nonsingular ($d \leq n$), $\hat{\beta}^j(\lambda)$ would be invariant on λ and $\hat{\beta}^j(\lambda) \propto \mathbf{V}_j$. When the sample covariance matrix is singular ($d > n$), the ℓ_2-norm penalty would regularize the inverse of shrunken covariance matrix (Witten et al. 2009) with respect to the strength of λ.

2.3 Implicit regularization with (stochastic) gradient flow

The implicit regularization effect of an estimation method means that the method produces an estimate exhibiting a kind of regularization, even though the method does not employ an explicit regularizer (Ali et al. 2019, 2020; Friedman and Popescu 2003, 2004). Consider gradient descent applied to Eq. (1), with initialization value $\beta_0 = \mathbf{0}$, and a constant step size $\eta > 0$, which gives the iterations

$$\beta_k = \beta_{k-1} + \frac{\eta}{n}\mathbf{X}^\top(Y - \mathbf{X}\beta_{k-1}) \,, \tag{6}$$

for $k = 1, 2, 3, \ldots$. With simply rearrangement, we get

$$\frac{\beta_k - \beta_{k-1}}{\eta} = \frac{\mathbf{X}^\top(Y - \mathbf{X}\beta_{k-1})}{n} \,. \tag{7}$$

To adopt a continuous-time (gradient flow) view, consider infinitesimal step size in gradient descent, i.e., $\eta \to 0$. The *gradient flow* differential equation for the OLS problem can be obtained with the following equation,

$$\dot{\beta}(t) = \frac{1}{n}\mathbf{X}^\top(Y - \mathbf{X}\beta(t-1)), \tag{8}$$

which is a continuous-time ordinary differential equation over time $t \geq 0$ with an initial condition $\beta(0) = \mathbf{0}$. We can see that by setting $\beta(t) = \beta_k$ at time $t = k\eta$, the left-hand side of Eq. (7) could be viewed as the discrete derivative of $\beta(t)$ at time t, which approaches its continuous-time derivative as $\eta \to 0$. To make it clear, $\beta(t)$ denotes the continuous-time view, and β_k the discrete-time view.

Lemma 1 *With fixed predictor matrix* $\mathbf{X}$ *and fixed response vector* Y, *the gradient flow problem in Eq.* (8), *subject to* $\beta(0) = \mathbf{0}$, *admits the following exact solution* (Ali et al. 2019)

$$\hat{\beta}_{\text{gf}}(t) = (\mathbf{X}^\top\mathbf{X})^+(\mathbf{I} - \exp(-t\mathbf{X}^\top\mathbf{X}/n))\mathbf{X}^\top Y, \tag{9}$$

for all $t \geq 0$. *Here* A^+ *is the Moore-Penrose generalized inverse of a matrix* A, *and* $\exp(A) = I + A + A^2/2! + A^3/3! + \cdots$ *is the matrix exponential.*

In continuous-time, ℓ_2-regularization corresponds to taking the estimator $\hat{\beta}_{\text{gf}}(t)$ in Eq. (9) for any finite value of $t \geq 0$, where smaller t corresponds to greater regularization. Specifically, the time t of gradient flow and the tuning parameter λ of ridge regression are related by $\lambda = 1/t$.

3 The proposed `AgFlow` algorithm

In this section, we first formulate the research problem, then present the design of proposed algorithm with a brief algorithm analysis.

3.1 Problem definition for ℓ_2-penalized PCA model selection

We formulate the model selection problem as selecting the empirically-best ℓ_2-Penalized PCA for the given dataset with respect to a performance evaluator.

- $\lambda \in \Lambda \subseteq \mathbb{R}^+$—the tuning parameters and the set of possible tuning parameters (which is a subset of positive reals);
- $\mathbf{X}_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times d}$ and $\mathbf{X}_{\text{val}} \in \mathbb{R}^{n_{\text{val}} \times d}$—the training data matrix and the validation data matrix, with n_{train} samples and n_{val} samples respectively;
- $\mathbf{Y}^j$—a given $n \times 1$ vector referring to the estimate of the j^{th} principal subspace;
- $\hat{\beta}^j(\lambda)$—the j^{th} projection vector, or the corresponding loading vector of the j^{th} PC, $\hat{\beta}^j(\lambda) = \left(\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{Y}^j$ solution of Eq. (4);
- $\hat{\boldsymbol{\beta}}(\lambda) = [\hat{\beta}^1(\lambda), \hat{\beta}^2(\lambda), \cdots, \hat{\beta}^{d'}(\lambda)] \in \mathbb{R}^{d \times d'}$—the projection matrix based on ℓ_2-penalized PCA with the tuning parameter λ, where each column $\hat{\beta}^j(\lambda)$ is the corresponding loadings of the j^{th} principal component;
- $\mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}(\lambda)$ and $\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}(\lambda)$—the dimension-reduced training data matrix and the dimension-reduced validation data matrix, respectively;
- $\texttt{model}(\cdot) : \mathbb{R}^{n_{\text{train}} \times d'} \rightarrow \mathcal{H}$—the target learner for performance tuning that outputs a model $h \in \mathcal{H}$ using the dimension-reduced training data $\mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}(\lambda)$.
- $\texttt{evaluator}(\cdot) : \mathbb{R}^{n_{\text{val}} \times d'} \times \mathcal{H} \rightarrow \mathbb{R}$—the evaluator that outputs the reward (a real scalar) of the input model h based on the dimension-reduced validation data $\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}(\lambda)$.

Then the model selection problem can be defined as follows.

$$\begin{aligned} \underset{\lambda \in \Lambda}{\text{maximize}} \quad & \texttt{evaluator}\left(\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}(\lambda),\ h(\lambda)\right), \\ \text{subject to} \quad & h(\lambda) = \texttt{model}\left(\mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}(\lambda)\right). \end{aligned} \tag{10}$$

Where

$$\begin{aligned} \bar{\beta}^j(\lambda) &= \underset{\beta \in \mathbb{R}^d}{\arg\min} \left\{ \frac{1}{2n} \left\| \mathbf{Y}^j - \mathbf{X}\beta \right\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right\}, \quad \text{for } j = 1, \ldots, d', \\ \hat{\beta}^j(\lambda) &= \frac{\bar{\beta}^j(\lambda)}{\|\bar{\beta}^j(\lambda)\|_2}. \end{aligned} \tag{11}$$

Note that $\texttt{model}(\cdot)$ can be any arbitrary target learner in the learning task and $\texttt{evaluator}(\cdot)$ can be any evaluation function of validation metrics. To make it clear, we take a classification problem as an example, thus the target learner $\texttt{model}(\cdot)$ can be the support vector machine (SVM) or random forest (RF), and the evaluation function $\texttt{evaluator}(\cdot)$ can be the classification error. To solve the above problem for arbitrary learning tasks $\texttt{model}(\cdot)$ under various validation metrics $\texttt{evaluator}(\cdot)$, there are at least two technical challenges needing to be addressed,

1. *Complexity* - For any given and fixed λ, the time complexity to solve the ℓ_2-penalized PCA (for dimension reduction to d') based on the Ridge-regression is $O(d' \cdot d^3)$, as it requests to solve the Ridge regression (to get the Ridge estimator) d' rounds to obtain the corresponding loadings of the top-d' principal components and the complexity to calculate the Ridge estimator in one round is $O(d^3)$.
2. *Size of* Λ - The performance of the model selection relies on the evaluation of models over a wide range of λ, while the overall complexity to solve the problem should be $O(|\Lambda| \cdot d' \cdot d^3)$. Thus, we need to obtain a well-sampled set of tuning parameters Λ that can balance the cost and the quality of model selection.

Algorithm 1 `AgFlow` Algorithm for Model Selection.

```
Input data:
    Training data matrix $\mathbf{X}_{\text{train}} = [\mathbf{x}_1, \cdots, \mathbf{x}_{n_{\text{train}}}]^\top$
    Validation data matrix $\mathbf{X}_{\text{val}} = [\mathbf{x}_1, \cdots, \mathbf{x}_{n_{\text{val}}}]^\top$
Parameters:
    Iterations $K$, step size $\eta$, batch size $m$, the reduced dimension $d'$.

%Approximate $\hat{\boldsymbol{\beta}}(\lambda)$ with $\hat{\boldsymbol{\beta}}_k$ using Approximated Gradient Flow %
for $j = 1, \cdots, d'$ do
    Get the $j$th principal subspace: $\mathbf{Y}^j \leftarrow$ QuasiPS($\mathbf{X}$, j).
    $\beta_0^j \leftarrow \mathbf{0}_d$.                                                    %Initialization%
    for $k = 1, \cdots, K$ do
        Sample a subset of index $I_k \subseteq \{1, \cdots, n_{\text{train}}\}$ with batch size $|I_k| = m$.
        $\beta_k^j \leftarrow \beta_{k-1}^j + \frac{\eta}{m}\sum_{i \in I_k}(\mathbf{Y}_i^j - \mathbf{x}_i^\top \beta_{k-1}^j)\mathbf{x}_i$.
        $\hat{\beta}_k^j \leftarrow \beta_k^j / \|\beta_k^j\|_2$                                  %Normalization%
    end for
end for

%The Projection Matrix based on the Approximated Gradient Flow%
The projection matrix: $\hat{\boldsymbol{\beta}}_k = [\hat{\beta}_k^1, \hat{\beta}_k^2, \cdots, \hat{\beta}_k^{d'}] \in \mathbb{R}^{d \times d'}$ for $k = 1, \cdots, K$.
(Where each $\hat{\boldsymbol{\beta}}_k$ corresponds to some $\hat{\boldsymbol{\beta}}(\lambda)$.)

%The Dimension-reduced Data Matrix Flow%
The dimension-reduced data matrix flow:
    $\tilde{\mathbf{X}}_{\text{train}}(k) = \mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}_k$, for $k = 1, \cdots, K$.
    $\tilde{\mathbf{X}}_{\text{val}}(k) = \mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}_k$, for $k = 1, \cdots, K$.

%Model selection based on the AgFlow matrix%
for $k = 1, \cdots, K$ do
    Fit the target learner $h(k) =$ model $\left(\mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}_k\right)$ using the dimension-reduced data.
    Get the evaluation function Eval$_k$ = evaluator $\left(\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}_k, h(k)\right)$.
end for

return $\hat{\boldsymbol{\beta}}^* = \arg\max_{\hat{\boldsymbol{\beta}}_k}$ evaluator $\left(\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}_k, h(k)\right)$, for $k = 1, \cdots, K$, as well as the optimal $k^*$ which
corresponds to the optimal $\lambda^*$ under $\lambda \propto \frac{1}{k\sqrt{\eta}}$.
```

3.2 Model selection for ℓ_2-penalized PCA over approximated gradient flow

In this section, we present the design of `AgFlow` algorithm (**Algorithm 1**) for obtaining the whole path of the loadings corresponding to each principal component for ℓ_2-penalized PCA. Consider the j^{th} principal component. Let $\mathbf{Y}^j$ be the j^{th} principal subspace, which can be approximated by the Quasi-Principal Subspace Estimation Algorithm (QuasiPS) through the call of QuasiPS($\mathbf{X}$, j) (**Algorithm 2**). The path of ℓ_2-penalized PCA should be the solution path of Ridge regression in Eq. (4) with varying λ from $0 \to \infty$.

With the implicit regularization of Stochastic Gradient Descent (SGD) for Ordinary Least Squares (OLS) (Ali et al. 2020), the solution path is equivalent to the optimization path of the following OLS estimator using SGD with zero initialization, such that

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2n} \|\mathbf{Y}^j - \mathbf{X}\beta\|_2^2 . \tag{12}$$

More specifically, with a constant step size $\eta > 0$, an initialization $\beta_0^j = \mathbf{0}_d$, and mini-batch size m, every SGD iteration updates the estimation as follows,

$$\beta_k^j = \beta_{k-1}^j + \frac{\eta}{m} \cdot \sum_{i \in I_k} (\mathbf{Y}_i^j - \mathbf{X}_i^\top \beta_{k-1}^j)\mathbf{x}_i , \tag{13}$$

for $k = 1, 2, \ldots, K$, and thus the solutions on the (stochastic) gradient flow path for the j^{th} principal component can be obtained. According to (Ali et al. 2020), the relationship of the explicit regularization λ and the implicit regularization effects introduced by SGD is $\lambda \propto \frac{1}{k\sqrt{\eta}}$. Thus, with the total number of iteration steps K large enough, the proposed algorithm could compete the path of penalized PCA for a full range of λ, but with a much lower computation cost.

Since in the problem of model selection of ℓ_2-penalized PCA based on Ridge estimator, we need to select the optimal $\hat{\beta}(\lambda^*)$ corresponding to the the optimal λ^*. Here we deal with the same model selection problem but with an alternative algorithm which uses the AgFlow algorithm instead of using matrix inverse in Ridge estimator. Therefore, we need to select the optimal $\hat{\beta}^*$ corresponding to the the optimal k^* with $k^* \propto \frac{1}{\lambda^* \sqrt{\eta}}$. To obtain the optimal λ^*, k-fold cross-validation is usually applied on a searching grid of λ-s in the model selection. As an analog of obtaining the optimal k^*, the proposed AgFlow algorithm is firstly used to get the iterated projection vector $\hat{\beta}_k$ of the given training data, which corresponds to some $\hat{\beta}(\lambda^*)$ with $k \propto \frac{1}{\lambda\sqrt{\eta}}$, then to select the optimal β^* based on the performance on the validation data.

Finally, **Algorithm 1** outputs the best projection matrix $\hat{\boldsymbol{\beta}}_{k^*} \in \mathbb{R}^{d \times d'}$, which maximizes the evaluator $\texttt{Eval}_k = \texttt{evaluator}\big(\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}_k, h(k)\big)$, for $k = 1, \ldots, K$. Where the index $k \propto \frac{1}{\lambda\sqrt{\eta}}$, and each column of the projection matrix $\hat{\boldsymbol{\beta}}_k$ is a normalized projection vector $\|\hat{\beta}_k^j\|_2 = 1$. Note that, as discussed in the preliminaries in Sect. 2, when the sample covariance matrix $1/n\mathbf{X}^\top\mathbf{X}$ is non-singular (when $n \gg d$), there is no need to place any penalty here, i.e., $\lambda \to 0$, and $k \to \infty$, as the normalization would remove the effect of the ℓ_2-regularization (Zou et al. 2006) considering Karush-Kuhn-Tucker conditions. However, when $d \gg n$, the sample covariance matrix $1/n\mathbf{X}^\top\mathbf{X}$ becomes singular, and the Ridge-liked estimator starts to shrink the covariance matrix as in Eq. (3), i.e., $\lambda \neq 0$, and k is some finite integer but not ∞, making the sample covariance matrix invertible and the results penalized in a covariance-regularization fashion (Witten et al. 2009). Even though the normalization would rescale the vectors to a ℓ_2-ball, the regularization effect still remains.

3.3 Near-optimal initialization for quasi-principal subspace

The goal of the QuasiPS algorithm is to approximate the principal subspace of PCA with given data matrix with extremely low cost, and AgFlow would fine-tune the rough quasi-principal projection estimation (i.e. the loadings) and obtain the complete path of the ℓ_2-penalized PCA accordingly. While there are various low-complexity algorithms in this area , such as (Hardt and Price 2014; Shamir et al. 2015; De Sa et al. 2015; Balsubramani et al. 2013), we derive the Quasi-Principal Subspace (`QuasiPS`) estimator (in **Algorithm** 2) using the stochastic algorithms proposed in (Shamir et al. 2015). More specifically, **Algorithm** 2 first pursues a rough estimation of the j^{th} principal component projection (denoted as $\tilde{w}_L$ after L iterations) using the stochastic approximation (Shamir et al. 2015), then obtains the quasi-principal subspace $\mathbf{Y}^j$ through projection $\mathbf{Y}^j = \mathbf{X}\tilde{w}_L$.

Note that $\tilde{w}_L$ is not a precise estimation of the loadings corresponding to its principal component (compared to our algorithm and (Oja and Karhunen 1985) etc.), however it can provide a close solution in an extremely low cost. In this way, we consider $\mathbf{Y}^j = \mathbf{X}\tilde{w}_L$ as a reasonable estimate of the principal subspace. With a random unit initialization $\tilde{w}_0$, $\tilde{w}_L$ converges to the true principal projection $\mathbf{v}^*$ in a fast rate under mild conditions, even when $\tilde{w}_0^\top \mathbf{v}^* \geq \frac{1}{\sqrt{2}}$ (Shamir et al. 2015). Thus, our setting should be non-trivial.

Algorithm 2 `QuasiPS`$(\mathbf{X}, \mathtt{j})$ – Quasi-Principal Subspace Approximation.

```
Input:
  Data matrix $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, index $j$.
Parameters:
  Step size $\eta_2$, epoch length $M$, iterations $L$.
Output: The $j^{th}$ Quasi-PS, $\mathbf{Y}^j$.

Initialization: $\check{w}_0 \sim \mathcal{N}(0, \mathbf{I}_d)$ and $\tilde{w}_0 \leftarrow \frac{\check{w}_0}{\|\check{w}_0\|_2}$.          %Random Unit Initialization.%
for $l = 1, \cdots, L$ do
  Let $\tilde{u} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i(\mathbf{x}_i^\top \tilde{w}_{l-1})$.
  $w_0 = \tilde{w}_{l-1}$.
  for $s = 1, 2, \cdots, M$ do
    Pick $i_s \in \{1, \cdots, n\}$ uniformly at random.
    $w_s^{'} = w_{s-1} + \eta_2[\mathbf{x}_{i_s}(\mathbf{x}_{i_s}^\top w_{s-1} - \mathbf{x}_{i_s}^\top \tilde{w}_{l-1}) + \tilde{u}]$.
    $w_s = \frac{w_s^{'}}{\|w_s^{'}\|_2}$
  end for
  $\tilde{w}_l = w_M$.
end for

$\mathbf{Y}^j \leftarrow \mathbf{X}\tilde{w}_L$          %Quasi Principal Subspace%
return $\mathbf{Y}^j$.
```

3.4 Algorithm analysis

In this section, we analyze the proposed algorithm from perspectives of statistical performance and its computational complexity.

3.4.1 Statistical performance

`AgFlow` algorithm consists of two steps: Quasi-PS initialization and solution path retrieval. As the goal of our research is fast model selection on the complete solution path for ℓ_2-penalized PCA over varying penalties, the performance analysis of the proposed algorithm can be decomposed into two parts.

Approximation of Quasi-PS to the true principal subspace. In `Algorithm 2`, the `QuasiPS` algorithm first obtains a Quasi-PS projection $\tilde{w}_L$ using L epochs of low-complexity stochastic approximation, then it projects the sample $\mathbf{x}$ to get the j^{th} principal subspace $\mathbf{Y}^j$ via $\mathbf{x}\tilde{w}_L$.

Lemma 2 *Under some mild conditions as in* Shamir et al. (2015) *and given the true principal projection w^*, with probability at least $1 - \log_2(1/\epsilon)\delta$, the the distance between w^* and $\tilde{w}_L$ holds that*

$$\|\tilde{w}_L - w^*\|_2^2 \leq 2 - 2\sqrt{1-\epsilon}, \tag{14}$$

provided that $L = \log(1/\epsilon)/\log(2/\delta)$.

It can be easily derived from (Theorem 1. in Shamir et al. (2015)). When $\epsilon \to 0$, $\|\tilde{w}_L - w^*\|_2^2 \to 0$ and the error bound becomes tight. Suppose samples in $\mathbf{X}$ are i.i.d. realizations from the random variable X and $\mathbb{E}XX^\top = \Sigma^*$ denote the true covariance.

Theorem 1 *Under some mild conditions as in* Shamir et al. (2015)*, the distance between the Quasi-PS and the true principal subspace holds that*

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathbf{x}\sim X}\|\mathbf{x}\tilde{w}_L - \mathbf{x}w^*\|_2^2 &= (\tilde{w}_L - w^*)^\top\Sigma^*(\tilde{w}_L - w^*)\\ &\leq \lambda_{\max}(\Sigma^*)\|\tilde{w}_L - w^*\|_2^2\\ &= (2 - 2\sqrt{1-\epsilon})\cdot\lambda_{\max}(\Sigma^*)\,,\end{aligned} \tag{15}$$

where $\lambda_{\max}(\cdot)$ refers to the largest eigenvalue of a matrix.

When considering the largest eigenvalue $\lambda_{\max}(\Sigma^*)$ as a constant, Quasi-PS is believed to achieve exponential coverage rate for principal subspace approximation for every sample. Thus, the statistical performance of Quasi-PS can be guaranteed.

Approximation of approximated stochastic gradient flow to the solution path of ridge. In (Ali et al. 2019, 2020), the authors have demonstrated that when the learning rate $\eta \to 0$, the discrete-time SGD and GD algorithms would diffuse to two continuous-time dynamics over (stochastic) gradient flows, i.e., $\hat{\beta}_{\text{sgf}}(t)$ and $\hat{\beta}_{\text{gf}}(t)$ over continuous time $t > 0$. According to Theorem 1 in Ali et al. (2019), the statistical risk between Ridge and continuous-time gradient flow is bounded by

$$\text{Risk}(\hat{\beta}_{\text{gf}}(t), \beta^*) \leq 1.6862 \cdot \text{Risk}(\hat{\beta}_{\text{ridge}}(1/t), \beta^*) \tag{16}$$

where $\text{Risk}(\beta_1, \beta_2) = \mathbb{E}\|\beta_1 - \beta_2\|_2^2$, β^* refers to the true estimator, and $\lambda = 1/t$ for Ridge. While the stochastic gradient flow enjoys a faster convergence but with slightly larger statistical risk, such that

$$\text{Risk}(\hat{\beta}_{\text{sgf}}(t), \beta^*) \leq \text{Risk}(\hat{\beta}_{\text{gf}}(t), \beta^*) + o\left(\frac{n}{m}\right)\,, \tag{17}$$

where m refers to the batch size and $o\left(\frac{n}{m}\right)$ is an error term caused by the stochastic gradient noises. Under mild conditions, with discretization ($\mathbf{d}t = \sqrt{\eta}$), we consider the k^{th} iteration of SGD for Ordinary Least Squares, denoted as β_k, which tightly approximates to $\hat{\beta}_{\text{sgf}}(t)$ in a $o(\sqrt{\eta})$-approximation with $t = k\sqrt{\eta}$. In this way, given the learning rate η and the total number of iterations K, the implicit Ridge-like AgFlow screens the ℓ_2-penalized PCA with varying λ in the range of

$$\frac{1}{K\sqrt{\eta}} \leq \lambda \leq \frac{1}{\sqrt{\eta}}\ , \tag{18}$$

with bounded error in both statistics and approximation.

In this way, we could conclude that under mild conditions, QuasiPS can well approximate the true principal subspace ($d' \ll n$) while AgFlow retrieves a tight approximation of the Ridge solution path..

3.4.2 Computational complexity

The proposed algorithm consists of two steps: the initialization of the quasi-principal subspace and the path retrieval. To obtain a fine estimate of Quasi-PS and hit the error in Eq. (14), one should run Shamir's algorithm (Shamir et al. 2015) with

$$O\big((\mathrm{rank}(\Sigma^*)/\mathrm{eigengap}(\Sigma^*))^2 \log(1/\epsilon)\big)$$

iterations, where rank$(\cdot)$ refers to the matrix rank, eigengap$(\cdot)$ refers to the gap between the first and second eigenvalues, and ϵ has been defined in Lemma 2 referring to the error of principal subspace estimation.

Furthermore, to get the loadings corresponding to the j^{th} principal subspace, AgFlow uses K iterations for OLS to obtain the estimate of K models for ℓ_2-penalized PCA, where each iteration only consumes $O(m \cdot d^2)$ complexity with batch size m, which gets total $O(K \cdot m \cdot d^2)$ for K models, and total $O(d' \cdot K \cdot m \cdot d^2)$ with the reduced-dimension d'. Moreover, we also propose to run AgFlow with full-batch size $m = n$ using gradient descent per iteration, which only consumes $O(d^2)$ per iteration with lazy evaluation of $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{X}^\top\mathbf{Y}$, with total $O(K \cdot d^2)$ for K models, which gets $O(d' \cdot K \cdot \cdot d^2)$ with the reduced-dimension d'.

To further improve AgFlow without incorporating higher-order complexity, we carry out the experiments by running a mini-batch AgFlow, and a full-batch AgFlow (i.e., $m = n$) with lazy evaluation of $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{X}^\top\mathbf{Y}$ in parallel for model selection.

4 Experiments

In this section, we show some experiments on real-world datasets with a significantly large number of features; that fits well in the natural High Dimension Low Sample Size (HDLSS) settings. Since cancer classification has remained a great challenge to researchers in microarray technology, we try to adopt our new algorithm on these gene expression datasets. In particular, except for three publicly available gene expression datasets (Zhu et al. 2007), the well-known FACES dataset (Huang et al. 2008) in machine learning is also considered in our study. A brief overview of these four datasets is summarized in Table 1.

4.1 Experiment setups

Evaluation procedure of the AgFlow algorithm. There are two regimes to demonstrate the performance of the proposed model selection method; the first is to evaluate the accuracy of the `AgFlow` algorithm based on k-fold cross-validation, which we call it evaluation-based model selection; the second is to do prediction based on the given training-validation-testing set which consists of three steps, i.e., model selection, model evaluation and prediction, which we call it prediction-based model selection. Usually, in the real-world applications, the prediction-based model selection is used, where the testing set is unseen in advance. The proceeding step would be to split the raw data into training-validation set for further cross-validation in evaluation-based model selection and training-validation-testing set for prediction-based model selection. There are two main steps, first is to get the the projection matrix of the the training data using the `AgFlow` algorithm; second is to apply the projection matrix to the validation/testing set.

Here we take the prediction-based model selection as an example. To do model selection using the `AgFlow` algorithm, *firstly* we need to get the projection matrix flow of the given training set by running the `AgFlow` algorithm, e.g. $\hat{\boldsymbol{\beta}}_k \in \mathbb{R}^{d\times d'}$ for $k = 1, \dots, K$. Then the dimension-reduced training-validation-testing data matrix flow can be obtained by matrix multiplication, e.g. $\tilde{\mathbf{X}}_{\text{train}}(k) = \mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}_k$, where $\hat{\boldsymbol{\beta}}_k = [\hat{\beta}_k^1, \hat{\beta}_k^2, \dots, \hat{\beta}_k^{d'}] \in \mathbb{R}^{d\times d'}$. Each column $\hat{\beta}_k^j$ is the j^{th} projection vector, i.e., the j^{th} loadings corresponding to the j^{th} principal component, which approximates the ℓ_2-penalized PCA with the tuning parameter λ, under the calibration $\lambda \propto 1/(k\sqrt{\eta})$. *Then* the dimension-reduced training data matrix flow is fed into the target learner $h(k) = \mathtt{model}(\mathbf{X}_{\text{train}}\hat{\boldsymbol{\beta}}_k)$ for performance tuning which outputs models $h(k)$ for $k = 1, \dots, K$. *Lastly*, the dimension-reduced validation data matrix flow is used to choose the optimal model with best performance according to the evaluator $\mathtt{evaluator}(\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}_k, h(k))$ for $k = 1, \dots, K$, which gives the optimal $\hat{\boldsymbol{\beta}}^* = \arg\max_{\hat{\beta}_k} \mathtt{evaluator}(\mathbf{X}_{\text{val}}\hat{\boldsymbol{\beta}}_k, h(k))$ and the optimal k^*. Note that each data flow matrix possesses some implicit regularization introduced by the `AgFlow` algorithm, which corresponds to an explicit penalty in Ridge. Under the calibration $\lambda \propto 1/(k\sqrt{\eta})$, we have $\hat{\boldsymbol{\beta}}_k \approx \hat{\boldsymbol{\beta}}(\lambda)$, $\hat{\boldsymbol{\beta}}^* \approx \hat{\boldsymbol{\beta}}(\lambda^*)$, with $\lambda^* \propto 1/(k^*\sqrt{\eta})$, thus we can do model selection using results based on `AgFlow` algorithm.

Settings of the `AgFlow` ***algorithm.***

- *Construction of training-validation-testing set.* For the above four datasets, we randomly split the raw data samples into training-validation-testing set with a fixed split ratio of $60\% - 20\% - 20\%$ within each class. Then the sample size for the training-validation-testing set is (240, 80, 80), (37, 12, 13), (43, 14, 15), (35, 11, 14) for FACES, Colon Tumor, ALL-AML Leukemia, Central Nervous System data, respectively. Thus dimension/sample size ratio *d*/*n* of the training set is 17.1, 54.1, 165.8, 203.7 accordingly.
- *Settings of default parameters.* For the default parameters in `AgFlow`, the number of iterations is set $K = 5000$, the step size $\eta = 0.5 \times 10^{-4}$, the batch size $\min(100, n/2)$, and the reduced dimension $d' = 30$.

Table 1 Description of the FACES dataset and Three Microarray Datasets (i.e. gene expression dataset of Colon Tumor, ALL-AML Leukemia, Central Nervous System)

Dataset	#Total Features (d)	#Samples (n)	#Classes
FACES	4096	400	10
Colon Tumor	2000	62	2
ALL-AML	7129	72	2
Central Nervous System	7129	60	2

For the values of explicit regularization of λ in `Ridge`, the ℓ_2-penalized PCA, we take 100 values in the log-scale ranging from 10^{-4} to 10^4 as the searching grid. For the default parameters in `QuasiPS`$(\mathbf{X}, j)$, we take the same default values as those specified in the original paper Shamir et al. (2015), where the step size $\eta_2 = \frac{1}{\bar{r}n}$, and $\bar{r} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i\|_2^2$, the epoch length $M = n$, and the number of iterations $L = 100$.

Baseline PCA Algorithms. To demonstrate the performance of the `AgFlow` algorithm, we compare the results with some other comparable methods, such as Oja's method (Oja and Karhunen 1985), Power iteration (Golub and Loan 2013), Shamir's Variance Reduction method (Shamir et al. 2015), vanilla PCA (Jolliffe 1986), and Ridge-based PCA (Zou et al. 2006) (two variants: the closed-form ridge estimator in Eq. (3), Ridge_C, and that based on scikit-learn solvers, Ridge_S).

4.2 Overall comparisons of model selection

In this section, we evaluate the performance of the proposed `AgFlow` algorithm and compare it with other baseline algorithms (especially in the performance comparisons with Ridge-based estimator) using FACES data, and three gene expression data of Colon Tumor, ALL-AML Leukemia, and Central Nervous System, respectively. In all these experiments, the training datasets have a limited number of samples and a significantly large number of features in the dimension reduction problem. For example, d/n ranges from 10 to 120, which is significantly larger than one in the four datasets. The common learning problem becomes ill-posed and models are all over-fit to the small training datasets. Model selection with the validation set becomes a crucial issue to improve the performance.

Figure 2 presents the overall performance comparisons on the dimension reduction problem between `AgFlow` and other baseline algorithms using FACES dataset, where the classification accuracy with dimension-reduced data is used as the metric. As only `AgFlow` and Ridge are capable of estimating penalized PCA models for model selection, in Fig. 2, we select the best models of both `AgFlow` and Ridge in terms of validation accuracy. For a fair comparison, we compare `AgFlow` with Ridge for model selection in a similar range of penalties (λ) using a similar budget of computation time, while we make sure that the time spent by `AgFlow` algorithm is much shorter than Ridge (Please refer Table 2 for the time consumption comparisons between `AgFlow` and Ridge.).

Under such critical HDLSS settings, usually all algorithms work poorly while `AgFlow` outperforms all these algorithms in most cases. Furthermore, Shamir's (Shamir et al. 2015) method, Oja's method (Oja and Karhunen 1985), Power iteration method and the vanilla PCA based on SVD, all achieve the similar performance in these settings, it seems these algorithms beat the best performance achievable for the unbiased PCA estimator without any regularization under ill-posed and HDLSS settings. The comparison between `AgFlow` and unbiased PCA estimators demonstrates the performance improvement contributed by the implicit regularization effects (Ali et al. 2020) and the potentials of model selection with validation accuracy. Furthermore, the comparison between `AgFlow` and Ridge indicates that the implicit regularization effect of SGD provides the model estimator with higher stability than Ridge in estimating penalized PCA under HDLSS settings, as the matrix inverse used in Ridge is unstable when the model is ill-posed (Eldad et al. 2008). Furthermore, the continuous trace of SGD provides model selector with more flexibility than Ridge in screening massive models under varying penalties with fine-grained granularity.

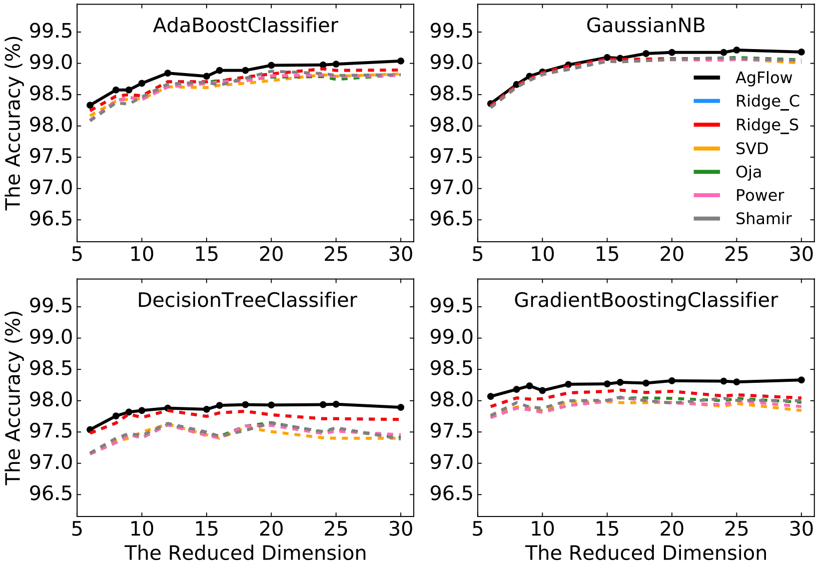
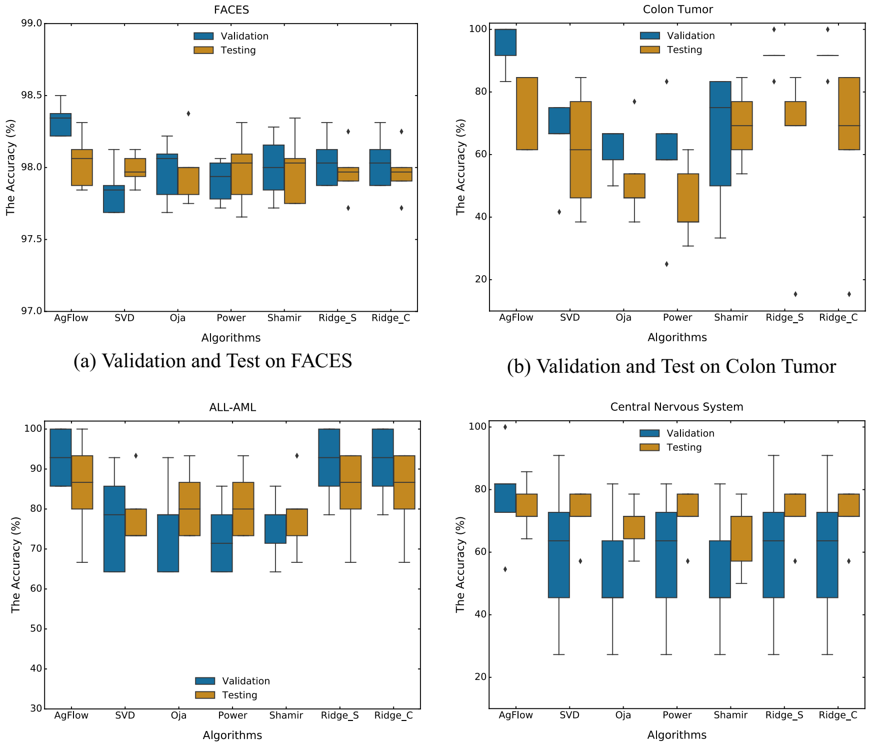


Fig. 2 Performance Comparisons on Dimension Reduction between `AgFlow` and Other Baseline Algorithms based on the Validation Accuracy of Adaptive Boosting Classifier, Gaussian Naive Bayes, Decision Tree Classifier, Gradient Boosting Classifier on FACES Dataset, respectively. SVD: Vanilla PCA based on SVD (Jolliffe 1986), Oja: Oja's stochastic PCA method (Oja and Karhunen 1985), Power: Power Iteration method (Golub and Loan 2013), Shamir: Shamir's Variance Reduction method (Shamir et al. 2015) and Ridge: Ridge-based PCA (Zou et al. 2006). (*Ridge_C stands for the ridge estimator based on the closed-form as in Eq.* (3)*,* Ridge_S stands for the ridge estimator based on scikit-learn solvers)

Figure 3 gives the performance comparison of validation and testing accuracy of different dimension reduction methods on different datasets, including `AgFlow` and other baseline algorithms such as vanilla PCA based on SVD (Jolliffe 1986), Oja's Stochastic PCA method (Oja and Karhunen 1985), Power Iteration method (Golub and Loan 2013), Shamir's Variance Reduction method (Shamir et al. 2015), and Ridge: Ridge-based PCA (Zou et al. 2006). Figure 3 shows that for the gene expression dataset of the Colon Tumor and Central Nervous System, the `AgFlow` algorithm outperforms other baseline algorithms with an overwhelming improvement with respect to the validation accuracy as well as the testing accuracy. For the FACES dataset, not much advantage of `AgFlow` is gained because all the algorithms achieve an accuracy above 90%, thus the improvement is less than 5%. For the ALL-AML dataset, the performance of all the algorithms varies a lot, our `AgFlow` is still the best one with respect to the validation accuracy, however, it is not the case when applied to the testing accuracy. The reason may be that, with one shot of training-validation-testing splitting, there is some variability in the data splitting and as the sample size is not that large that makes this uncertainty worse, which also explains that the testing accuracy is somewhat larger than the validation accuracy for some algorithms.

In this way, based on the comparisons of different dimension reduction methods using the same data with a given classifier function as in Fig. 2 and the the comparisons using different datasets Fig. 3, we can conclude that `AgFlow` is more effective than Ridge for estimating massive models and selecting the best models for penalized PCA, with the same or even stricter budget conditions. We also present the comparison



(a) Validation and Test on FACES

(b) Validation and Test on Colon Tumor

(c) Validation and Test on ALL-AML

(d) Validation and Test on Central Nervous System

Fig. 3 Performance Comparisons of Validation and Testing Accuracy of Different Dimension Reduction Methods, `AgFlow` and Other Baseline Algorithms, on FACES (with $d' = 30$ and Gradient Boosting Classifier), Colon Tumor (with $d' = 30$ and Quadratic Discriminant Analysis), ALL-AML (with $d' = 30$ and Random Forest Classifier), Central Nervous System Dataset (with $d' = 24$ and Gradient Boosting Classifier). SVD: vanilla PCA based on SVD (Jolliffe 1986), Oja: Oja's Stochastic PCA method (Oja and Karhunen 1985), Power: Power Iteration method (Golub and Loan 2013), Shamir: Shamir's Variance Reduction method (Shamir et al. 2015) and Ridge: Ridge-based PCA (Zou et al. 2006). (*Ridge_C stands for the ridge estimator based on the closed-form as in Eq..* (3)*, Ridge_S stands for the ridge estimator based on scikit-learn solvers*)

results based on different datasets in Fig. 3 using various classifiers. Similar results are obtained: Ridge works well as more samples provided and `AgFlow` outperforms Ridge estimator in most cases.

4.3 Comparisons of time consumption and performance tuning

Table 2 illustrates the time consumption of the `AgFlow` algorithm and Ridge-based algorithms over varying penalties on the four datasets. We can see from the table that the time used in the `AgFlow` algorithm is only a small portion of that of the Ridge_S and Ridge_C which are two versions of Ridge-based algorithms. When the sample size and the number of predictors are both small, as in the Colon Tumor dataset with $(n, d) = (62, 2000)$, the time consumption is acceptable for both `AgFlow` and Ridge-based algorithms. However, when the number of the dimension becomes extremely large as in the ALL-AML dataset with $(n, d) = (72, 7129)$ or the Central Nervous System data with $(n, d) = (60, 7129)$, the time consumption of Ridge_S and Ridge_C becomes dramatically large. For example, when $d' = 30$ for the ALL-AML dataset, Ridge_S requires more than 12.39 hours, which is unacceptable in practice application, whereas the the `AgFlow` algorithm requires 14 minutes, which has dramatically reduced the computation time.

More specifically, when considering the time consumption of `AgFlow` and Ridge-Path for the above performance tuning procedure, we can find `AgFlow` is much more efficient. Table 2 shows that `AgFlow` only consumes 246 seconds to obtain the estimates of 10,000 penalized PCA models when $d' = 30$ for the Colon Tumor data with $d = 2000$ genes and 851 seconds for the ALL-AML Leukemia data with $d = 7129$ genes, while Ridge-Path needs 1226 seconds/1276 seconds to obtain only 100 penalized PCA models for the Colon Tumor data and 44, 606 seconds/43, 268 seconds for the ALL-AML Leukemia data, whether using closed-form Ridge estimators or solver-based ones.

Figure 4 illustrates the examples of performance tuning using Ridge-Path and `AgFlow` over varying penalties with Random Forest classifiers. While `AgFlow` estimates the ℓ_2-penalized PCA with varying penalty by stopping the SGD optimizer with different number of iterations, Ridge-Path needs to shrink the sample covariance matrix with varying λ and estimate ℓ_2-penalized PCA through the time-consuming matrix inverse. It is obvious that both `AgFlow` and Ridge-Path have certain capacity to screen models with different penalties.

In conclusion, `AgFlow` demonstrates both efficiency and effectiveness in model selection for penalized PCA, in comparisons with a wide range of classic and newly-fashioned algorithms (Zou et al. 2006; Shamir et al. 2015; Oja and Karhunen 1985; Golub and Loan 2013). Note that, the classification accuracy of some tasks here might not be as good as those reported in Zhu et al. (2007). While our goal is to compare the performance of ℓ_2 -penalized PCA model selection with classification accuracy as the selection objective, the work (Zhu et al. 2007) focus on selecting a discriminative set of features for classification.

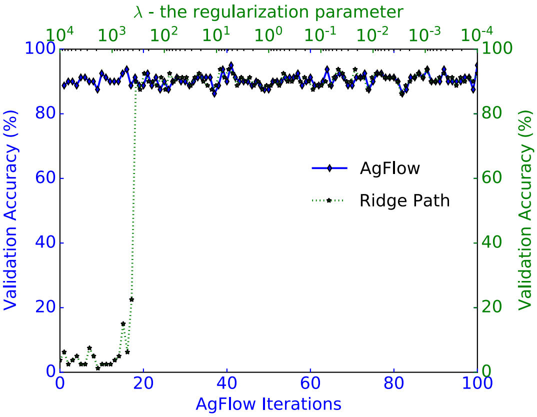
5 Conclusions

Since PCA has been widely used for data processing, feature extraction and dimension reduction in unsupervised data analysis, we have proposed `AgFlow` algorithm to do fast model selection with a much lower complexity in ℓ_2-penalized PCA where the regularization is usually incorporated to deal with the multicolinearity and singularity issues encountered under HDLSS settings. Experiments show that our `AgFlow` algorithm beats the existing methods with an overwhelming improvement with respect to the accuracy and computational complexity, especially, when compared with the ridge-based estimator which is implemented as a time-consuming model estimation and selection procedure among a wide range of penalties with matrix inverse. Meanwhile, the proposed `AgFlow` algorithm naturally retrieves the complete solution path of each principal component, which shows an implicit regularization and can help us do the model estimation and selection simultaneously. Thus we can identify the best model from an end-to-end optimization procedure using low computational complexity. In addition, except for the advantage of the accuracy and computational complexity, the `AgFlow` enlarges the capacities of performance tuning in a more intuitive and easily way. The observations backup our claims.

Table 2 Comparison of Time Consumption/(#Models) in Seconds using FACES dataset, and gene expression data for Colon Tumor, ALL-AML Leukemia, Central Nervous System. (*Ridge_C stands for the ridge estimator based on the closed-form as in Eq.* (3), *Ridge_S stands for the ridge estimator based on scikit-learn solvers*)

	FACES $(n, d) = (400, 4096)$			Colon Tumor $(n, d) = (62, 2000)$			ALL-AML $(n, d) = (72, 7129)$			Central Nervous System $(n, d) = (60, 7129)$		
d'	`AgFlow` (10,000 Models)	Ridge_S (100 Models)	Ridge_C (100 Models)	`AgFlow` (10,000 Models)	Ridge_S (100 Models)	Ridge_C (100 Models)	`AgFlow` (10,000 Models)	Ridge_S (100 Models)	Ridge_C (100 Models)	`AgFlow` (10,000 Models)	Ridge_S (100 Models)	Ridge_C (100 Models)
6	1135	2346	2601	100	281	316	167	8713	8841	165	8531	9136
8	1284	3011	3348	106	361	403	194	11,645	11,722	186	11,350	12,114
9	1369	3341	3714	110	400	443	212	13,099	13,158	203	12,758	13,602
10	1440	3670	4067	114	439	483	227	14,550	14,601	220	14,167	15,053
12	1598	4332	4781	124	526	562	267	17,468	17,475	262	16,992	17,973
15	1830	5316	5800	138	637	682	335	21,920	21,776	330	21,217	22,287
16	1915	5646	6141	143	676	722	361	23,428	23,209	354	22,628	23,716
18	2086	6303	6827	155	754	801	417	26,455	26,074	407	25,446	26,648
20	2246	6956	7501	167	833	880	478	29,463	28,931	465	28,280	29,510
24	2600	8275	8854	195	991	1039	610	35,516	34,661	597	34,090	35,210
25	2679	8597	9190	204	1030	1079	647	37,041	36,093	634	35,554	36,641
30	3127	10,243	10,866	246	1226	1276	851	44,606	43,268	836	42,891	43,788

Fig. 4 An Example of Parameter Tuning on FACES dataset with Random Forest



6 Future work

Though the `AgFlow` algorithm naturally retrieves the complete solution path of each principal component and can do model selection under the implicit ℓ_2-norm regularization effect, the linear combination of all the original variables is often not friendly to interpret the results. New methods with implicit or explicit ℓ_1-norm regularization (lasso penalty) are in great demand, where ℓ_1-norm regularization produces sparse solutions and we can do variable estimation and selection simultaneously.

In addition to the Approximated Gradient Flow, we are also interested in the implicit regularization introduced by other (stochastic) optimizers, such as Adam and/or Nesterov's momentum methods, with potential new applications to Markov Chain Monte Carlo or other statistical computations. Furthermore, the implicit regularization of the `AgFlow` running nonlinear models for statistical inference would be interesting too.

References

Ali, A., Dobriban, E., & Tibshirani, R. J. (2020). The implicit regularization of stochastic gradient flow for least squares. In *International conference on machine learning* (pp. 233–244). PMLR.

Ali, A., Kolter, J. Z., & Tibshirani, R. J. (2019). A continuous-time view of early stopping for least squares regression. In *The 22nd international conference on artificial intelligence and statistics* (pp 1370–1378).

Arora, R., Cotter, A., Livescu, K., & Srebro, N. (2012). Stochastic optimization for PCA and PLS. In *2012 50th annual allerton conference on communication, control, and computing (allerton)* (pp. 861–868). IEEE.

Balsubramani, A., Dasgupta, S., & Freund, Y. (2013). The fast convergence of incremental PCA. In F. Bach, & D. Blei (Ed.), *Advances in neural information processing systems* (pp. 3174–3182).

Candes, E., Tao, T., et al. (2007). The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics, 35*(6), 2313–2351.

De Sa, C., Re, C., & Olukotun, K. (2015). Global convergence of stochastic gradient descent for some nonconvex matrix problems. In *International conference on machine learning* (pp. 2332–2341).

Dutta, A., Hanzely, F., & Richtárik, P. (2019). A nonconvex projection method for robust PCA. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 1468–1476).

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association, 96*(456), 1348–1360.

Friedman, J., & Popescu, B. E. (2003). Gradient directed regularization for linear regression and classification. Technical report, Technical Report, Statistics Department, Stanford University.

Friedman, J., & Popescu, B. E. (2004). Gradient directed regularization. Unpublished manuscript. http://www-stat.stanford.edu/hf/ftp/pathlite.pdf. Accessed 24 June 2021.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software, 33*(1), 1.

Gaynanova, I., Booth, J. G., & Wells, M. T. (2017). Penalized versus constrained generalized eigenvalue problems. *Journal of Computational and Graphical Statistics, 26*(2), 379–387.

Golub, G., & Loan, C. (2013). *Matrix computations* (4th ed.). Baltimore: Johns Hopkins University Press.

Haber, E., Horesh, L., & Tenorio, L. (2008). Numerical methods for experimental design of large-scale linear ill-posed inverse problems. *Inverse Problems, 24*(5), 055012.

Hardt, M., & Price, E. (2014). The noisy power method: a meta algorithm with applications. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 2861–2869). MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Berlin: Springer.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(1), 55–67.

Hoerl, A. E., Kannard, R. W., & Baldwin, K. F. (1975). Ridge regression: Some simulations. *Communications in Statistics-Theory and Methods, 4*(2), 105–123.

Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

Jolliffe, I. T. (1986). Principal components in regression analysis. In I. T. Jolliffe (Ed.), *Principal component analysis* (pp. 129–155). Springer.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., & Simard, P. (1995). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks* (Vol. 60, pp. 53–60). Australia: Perth.

Lee, Y. K., Lee, E. R., & Park, B. U. (2012). Principal component analysis in very high-dimensional spaces. *Statistica Sinica*, *22*(3), 933–956.

Mitliagkas, I., Caramanis, C., & Jain, P. (2013). Memory limited, streaming PCA. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani & K.Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 2886–2894).

Mohammed, A. A., Minhas, R., Jonathan Wu, Q. M., & Sid-Ahmed, M. A. (2011). Human face recognition based on multidimensional PCA and extreme learning machine. *Pattern Recognition, 44*(10–11), 2588–2597.

Oja, E., & Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications, 106*(1), 69–84.

Shamir, O. (2015). A stochastic PCA and SVD algorithm with an exponential convergence rate. In *International conference on machine learning* (pp. 144–152). PMLR.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288.

Witten, D. M., & Tibshirani, R. (2009). Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71*(3), 615–636.

Witten, D. M., Tibshirani, R., & Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics, 10*(3), 515–534.

Yeung, K. Y., & Ruzzo, W. L. (2001). Principal component analysis for clustering gene expression data. *Bioinformatics, 17*(9), 763–774.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68*(1), 49–67.

Zhu, Z., Ong, Y.-S., & Dash, M. (2007). Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition, 40*(11), 3236–3248.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(2), 301–320.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics, 15*(2), 265–286.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.