Deep Unfolded IRLS-ADMM Network for Classification and Sparse Feature Selection

Xian Yang and Yike Guo

***Abstract*—Two main branches of machine learning methods are model-based methods and deep neural networks. Model-based methods can explicitly include prior knowledge into the model at expense of difficulties in inference, while neural networks are featured in their strong predictive power and straightforward inference approach with the lack of model interpretability. To construct models which are entitled with the advantages of both methods and overcome their problems, the deep unfolding strategy has been developed recently. This paper adopts the idea of deep unfolding to construct a classification and feature selection method. The proposed method is based on the sparse classification; and the iterative inference process of the sparse classification is unfolded into a layer-wise structure analogous to a neural network. Thus, the architecture of our network is fully motivated by the sparse classification method. Different from other neural networks which are essentially black-box methods, our deep unfolded network acts as white-box that features selected in the predictive model can be explicitly returned. Experimental results show the both predictive power and feature selection ability of our methods.**

***Index Terms*—Deep unfolded neural network, IRLS-ADMM net, sparse classification, feature selection, white-box method.**

I. Introduction

Deep unfolding is a recently proposed strategy to construct neural-network-like machine learning models that take the advantage of both model-based methods and neural networks [1]. Model-based methods can easily incorporate prior knowledge, such as sparsity, conditional dependency and latent variable structure, into the model. However, the inference of model-based methods are usually not straightforward and can be both computationally and mathematically intractable that approximation methods such as variational approximation and Monte Carlo sampling are required to iteratively infer variables of interest. Neural networks, on the other hand, organise inference into layers which are typically executed in sequence. Although neural networks become the state of art in many applications, they have the problem of model interpretability. Typical neural methods are black-box methods acting closer to mechanisms than problem-level formulations. Therefore, it comes to the idea of bringing the model-based approach to the task of design the architecture of neural networks, which is called deep unfolding.

The basic idea of deep unfolding is with a given model-based method unfolding the iterations of inference into a layer-wise structure, which forms into a neural network. To increase the flexibility of the model and the ease of training, model parameters across layers are de-coupled. The resulting formula combines the predictive power of neural network with problem-level formulation of model-based methods. The inference is performed with fixed number of layers and back propagation is used to estimate model parameters. In [1], this deep unfolding strategy is used in the domain of speech enhancement, constructing network based on non-negative matrix factorization [2]. The domain knowledge that signals mix linearly is embodied in the model. Deep unfolding has also been applied to multichannel source separation [3]. A multichannel Gaussian mixture model based neural network has been proved to achieve improved performance of separating signals from simultaneous speakers. In [4], deep unfolded network is applied for supervised topic model. It improves the classification accuracy by replacing the maximisation of lower bound of marginal likelihood in variational Bayesian with the minimisation of cross entropy. Another example is, in [5], a Gaussian conditional random field (GCRF) based neural network is constructed for image denoising. The proposed network can explicitly model the input noise variance and handle a range of noise levels. Along with those models, there are many attempts on constructing neural network based on model-based methods that impose sparse constraints. For example, the work in [6] unfolds the iterations of linear sparse regression method into variants of long short-term memory (LSTM) cells with the results showing that the unfolded network has reduced computational burden in inference. The sparse coding based neural network has also been applied to reconstruct sparse signals from noisy and compressive measurements in [7]. The work in [8] design a neural network based on sparse coding to reconstruct high-quality MR images from under-sampled data, which is effective both in reconstruction accuracy and speed.

In this paper, we focus on constructing a model-based network for both classification and feature selection. The model-based method investigated in this paper is sparse Bayesian classification (SBC) [9], [10]. One of the distinguishing characteristics of SBC is that it does not only build a classification model but also returns a set of features with non-zero weights. In the literature, several classification and feature selection methods are based on SBC [11]-[13]. In our previous work [14], we derive an optimisation-centric classification and feature selection method from SBC. The model is constructed through an iterative convex optimisation procedure instead of a one-step closed form calculation. The cost function is cast using hierarchical Bayesian model, where parameters and hyperparameters are

Manuscript received March 22, 2018; revised May 6, 2018.

The authors are with the Data Science Institute, Imperial College London, SW7 2AZ, UK (e-mail: xian.yang08@impeial.ac.uk, y.guo@impeial.ac.uk).

doi: 10.18178/ijmlc.2018.8.3.694

inferred via the convex optimisation procedure [10], [15]. In this paper, we construct neural network based on this method: the parameters in the model-based method are regarded as hidden nodes in the network, while hyperparameters are learned by training the model through the back-propagation method. We adopt iterative reweighted least squares (IRLS) [16] and alternating direction method of multipliers (ADMM) [8] to get the iterative inference steps. IRLS is used to formulate the problem of finding the step direction for Newton's method as a weighted ordinary least squares problem; AMDD is applied to solve the convex optimisation problem by breaking them into smaller pieces, which has proven to be an efficient variable slitting algorithm with convergence guarantee. After we derived formula for iterative updates, we then design deep architecture consisting of multiple stages, each of which corresponds to an iterative inference step. The weight parameters in the model-based classification model are regarded as nodes in the network, while hyperparameters are learned via stochastic gradient descend (SGD) [17]. The following parts of our paper will first introduce the method in detail. Then the experimental results part will show that the proposed deep unfolded network works well in the aspect of both classification power and feature selection.

II. Depp IRLS-ADMM Net for Classification and Feature Selection

A. The Sparse Bayesian Classification Based Optimisation Method

Suppose we get a set of input vectors $\{\boldsymbol{x}_n\}_{n=1}^N$ along with corresponding targets $\{y_n\}_{n=1}^N$. We wish to learn the underlying functional mapping which is defined by a parameterised function $f(\boldsymbol{x};\boldsymbol{\beta}) = \sum_{i=1}^{\aleph} \beta_i\, \phi_i(\boldsymbol{x})$, where the output is the linear weighted sum of $\aleph$ basis functions and $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_{\aleph}]^\top$ contains the parameter. Let $\boldsymbol{\Phi}$ be the $N \times \aleph$ design matrix with $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \boldsymbol{\phi}(\boldsymbol{x}_2), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N)]^\top$, wherein $\boldsymbol{\phi}(\boldsymbol{x}_n) = [\phi_1(\boldsymbol{x}_n), \phi_2(\boldsymbol{x}_n), \ldots, \phi_{\aleph}(\boldsymbol{x}_n)]$. Then we can express the mapping function as $f(\boldsymbol{x};\boldsymbol{\beta}) = \boldsymbol{\Phi}\boldsymbol{\beta}$. The prior distribution of the weights is assumed to follow a zero-mean isotropic Gaussian:

$$\begin{aligned} \mathcal{P}(\boldsymbol{\beta}|\boldsymbol{\gamma}) &= \prod_{i=1}^{\aleph} \mathcal{N}(\beta_i|0,\gamma_i) \\ &= \prod_{i=1}^{\aleph} (2\pi\gamma_i)^{-\frac{1}{2}} \exp\{-\frac{\beta_i^2}{2\gamma_i}\}, \end{aligned} \tag{1}$$

where

$$\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_{\aleph}] \in \mathbb{R}^{\aleph}, \boldsymbol{\Gamma} = \operatorname{diag}[\boldsymbol{\gamma}]. \tag{2}$$

The likelihood function $\mathcal{P}(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{x})$ is expressed in the form of the logistic regression model [18]. According to the Bayes' rule, maximisation of posterior is equivalent to finding the maximum over $\boldsymbol{\beta}$ of $\log \mathcal{P}(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{x})\mathcal{P}(\boldsymbol{\beta}|\boldsymbol{\gamma})$. The hyperparameter $\boldsymbol{\gamma}$ is updated by maximising the marginal likelihood, which is equivalent to $argmin_{\boldsymbol{\gamma}} -\mathcal{P}(\boldsymbol{y}|\boldsymbol{\gamma},\boldsymbol{x}) = argmin_{\boldsymbol{\gamma}} -\int \mathcal{P}(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{x})\mathcal{P}(\boldsymbol{\beta}|\boldsymbol{\gamma})d\boldsymbol{\beta}$. In [14], we propose a method to jointly estimate $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ through a common objective function:

$$\begin{aligned} argmin_{\boldsymbol{\beta},\boldsymbol{\gamma}} \quad & \textstyle\sum_{n=1}^{N} \log[1 + \exp\{-y_n \boldsymbol{\beta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n)\}] \\ & + \boldsymbol{\beta}^\top \boldsymbol{\Gamma}^{-1} \boldsymbol{\beta} + \log|\boldsymbol{\Gamma}| + \tfrac{1}{2}\log|\boldsymbol{H}(\boldsymbol{\beta}^*)|, \end{aligned} \tag{3}$$

where $|\boldsymbol{H}(\boldsymbol{\beta}^*)|$ is the Hessian matrix calculated at mode $\boldsymbol{\beta}^*$, which is assumed to be obtained through the minimisation step of $\boldsymbol{\beta}$ in the iterative optimisation process. This objective function can be expressed in the convex-concave form [19] that the standard iterative optimisation procedure is evoked to get its solution.

B. Deep IRLS-ADMM Net

The approach in [14] carries out an iterative process to update $\boldsymbol{\beta}^{k+1}$ and hyper parameters. The target function of obtaining $\boldsymbol{\beta}$ can be easily formulated in the ℓ_1 regularization form as:

$$\begin{aligned} \boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} & \sum_{n=1}^{N} \log[1 + \exp\{-y_n \boldsymbol{\beta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n)\}] \\ & + \sum_{i=1}^{\aleph} \| w_i^k \cdot \beta_i \|_{\ell_1}, \end{aligned} \tag{4}$$

where $\boldsymbol{w}^k$ is related to hyper-parameters. Rather than considering the iterations as an algorithm, we consider unfolding it as a sequence of layers in a neural network-like architecture. The iteration index is now interpreted as an index to the neural network [1]. Our approach tries to optimise $\boldsymbol{\beta}^{k+1}$ by treating them as nodes of layers in a neural network parameterised by $\boldsymbol{w}^k$. We optimize the parameter $\boldsymbol{w}^k$ via the stochastic gradient descent method by minimising the loss $\sum_{n=1}^{N} D(y_n, y_n *)$, where $y_n *$ is the estimated target and D is the loss function.

1) The IRLS-ADMM algorithm

To construct the network, we need to obtain the updating steps of $\boldsymbol{\beta}^{k+1}$ in the form of:

$$\boldsymbol{\beta}^{k+1} = f_{\boldsymbol{w}^k}(\boldsymbol{X}, \boldsymbol{\beta}^k). \tag{5}$$

There are a lot of work on using model-based inspiration of novel neural networks, especially on sparse coding problems. However, in sparse coding, the target optimisation function is generally in the form of $\|A\boldsymbol{x} - y\|_{\ell_2} + g(\boldsymbol{x})$, where $g(\boldsymbol{x})$ is the penalty term. Optimisation methods, such as ADMM [8] and proximal gradient method [7], can be used to inspire novel neural networks. For the sparse coding problem, the iterative process for updating $\boldsymbol{x}$ can be expressed in the closed form. In our case, however, there is no straightforward way to drive function $f_{\boldsymbol{w}^k}$ due to the complex form of the first term in Equation 4.

Our idea is to reformulate the target optimisation function in the form of least squares that typical optimisation methods can be used to find the expression of $f_{\boldsymbol{w}^k}$. We adopt the idea in [16] based on the IRLS formulation of logistic regression. IRLS reformulates the problem of finding the step direction for Newton's method as a weighted ordinary least squares problem. In every iteration, Newton's method first finds a step direction by approximating the objective function by the second order Taylor expression at the current point, and optimising the quadratic approximation in closed-from. The step direction in the kth iteration is:

$$\boldsymbol{\Omega}^k = \boldsymbol{\beta}^k - \boldsymbol{H}^{-1}(\boldsymbol{\beta}^k)\boldsymbol{g}(\boldsymbol{\beta}^k), \tag{6}$$

where $\boldsymbol{g}$ and $\boldsymbol{H}$ are gradient and Hessian matrices. Once the step direction is computed, Newton's method computes the next iteration

$$\boldsymbol{\beta}^{k+1} = (1-t)\boldsymbol{\beta}^{k} + t\boldsymbol{\Omega}^{k} \tag{7}$$

by a linear search over the step size parameter t.

For the optimisation function in the form of unregularized logistic regression, where the penalty term in Equation 4 is omitted, the gradient and Hessian matrices are in the form of:

$$\boldsymbol{g}(\boldsymbol{\beta}^{k}) = -\sum_{n=1}^{N} y_n \boldsymbol{\phi}(\boldsymbol{x}_n)[1-\sigma\{y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}] \tag{8}$$

and

$$H(\beta^{k}) = \Phi^{\top}\, diag\,[y]\, \Lambda(\beta^{k})\, diag\,[y]\, \Phi, \tag{9}$$

where

$$\begin{aligned}\boldsymbol{\Lambda}(\boldsymbol{\beta}^{k}) = \text{diag}\,(\{&\sigma\{y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}\\ &[1-\sigma\{y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}]\}_{n=1}^{N})\end{aligned} \tag{10}$$

and

$$\sigma\{f\} = \frac{1}{(1+e^{-f})}. \tag{11}$$

Let $\bar{\boldsymbol{\Phi}} = \text{diag}\,[\boldsymbol{y}]\,\boldsymbol{\Phi}$, then we get

$$\boldsymbol{H}(\boldsymbol{\beta}^{k}) = \bar{\boldsymbol{\Phi}}^{\top}\boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})\bar{\boldsymbol{\Phi}}. \tag{12}$$

If we define a vector $\boldsymbol{v}(\boldsymbol{\beta}^{k})$, whose nth element is

$$v_n(\beta^{k}) = y_n \phi(x_n)^{\top}\beta^{k} + \frac{1-\sigma\left\{y_n \beta^{k\top}\phi(x_n)\right\}}{\Lambda(\beta^{k})_{n,n}}, \tag{13}$$

then $\boldsymbol{g}(\boldsymbol{\beta}^{k})$ can be defined in the form of

$$g(\beta^{k}) = -\bar{\Phi}^{\top}\Lambda(\beta^{k})(v(\beta^{k}) - \bar{\Phi}\beta^{k}). \tag{14}$$

The step direction in Equation 6 can be expressed as:

$$\boldsymbol{\Omega}^{k} = (\bar{\boldsymbol{\Phi}}^{\top}\boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})\bar{\boldsymbol{\Phi}})^{-1}\bar{\boldsymbol{\Phi}}^{\top}\boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})\boldsymbol{v}(\boldsymbol{\beta}^{k}), \tag{15}$$

which can be regarded as the solution to the following weighted least squares problem:

$$\boldsymbol{\Omega}^{k} = \underset{\boldsymbol{\Omega}}{\text{argmin}} \,\|\, (\boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})^{\frac{1}{2}}\bar{\boldsymbol{\Phi}})\boldsymbol{\Omega} - \boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})^{\frac{1}{2}}\boldsymbol{v}(\boldsymbol{\beta}^{k})\|_{\ell_2}. \tag{16}$$

Therefore, the Newton step direction can be computed by solving the least squares problem defined in Equation 16. The term IRLS refers to the fact that in every iteration the least squares problem has a different diagonal weighting matrix $\Lambda(\boldsymbol{\beta}^{k})$ and $\boldsymbol{v}(\boldsymbol{\beta}^{k})$. For the case of ℓ_1 regularized logistic regression as in Equation 4, we can get our IRLS formulation as

$$\begin{aligned}\Omega^{k} = \underset{\Omega}{argmin}\, &\frac{1}{2}\|(\Lambda(\beta^{k})^{\frac{1}{2}}\bar{\Phi})\Omega - \Lambda(\beta^{k})^{\frac{1}{2}}v(\beta^{k})\|_{\ell_2}\\ &+\lambda\|F^{k}\Omega\|_{\ell_1}.\end{aligned} \tag{17}$$

Please note that $\boldsymbol{F}^{k}$ in Equation 17 is a diagonal matrix giving weights to elements of $\boldsymbol{\Omega}$.

Now, we would like to adopt the ADMM method to get $\boldsymbol{\Omega}^{k}$ by solving Equation 17. Suppose

$$\boldsymbol{A}^{k} = \boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})^{\frac{1}{2}}\bar{\boldsymbol{\Phi}} \tag{18}$$

and

$$\boldsymbol{b}^{k} = \boldsymbol{\Lambda}(\boldsymbol{\beta}^{k})^{\frac{1}{2}}\boldsymbol{v}(\boldsymbol{\beta}^{k}). \tag{19}$$

In ADMM form, the problem in Equation 17 can be written as

$$\begin{aligned}&minimize\ \ \frac{1}{2}\|\boldsymbol{A}^{k}\boldsymbol{\Omega} - \boldsymbol{b}^{k}\|_{\ell_2} + \lambda\|\boldsymbol{z}\|_{\ell_1}\\ &subject\ to\ \ \boldsymbol{F}^{k}\boldsymbol{\Omega} - \boldsymbol{z} = 0,\end{aligned} \tag{20}$$

which yields the ADMM algorithm in the ith iteration [20]:

$$\begin{aligned}\boldsymbol{\Omega}^{k,i+1} &:= (\boldsymbol{A}^{k\top}\boldsymbol{A}^{k} + \rho\boldsymbol{F}^{k\top}\boldsymbol{F}^{k})^{-1}\\ &\quad (\boldsymbol{A}^{k\top}\boldsymbol{b}^{k} + \rho\boldsymbol{F}^{k\top}(\boldsymbol{z}^{k,i} - \boldsymbol{u}^{k,i}))\\ \boldsymbol{z}^{k,i+1} &:= S_{\lambda/\rho}(\boldsymbol{F}^{k}\boldsymbol{\Omega}^{k,i+1} + \boldsymbol{u}^{k,i})\\ \boldsymbol{u}^{k,i+1} &:= \boldsymbol{u}^{k,i} + \boldsymbol{F}^{k}\Omega^{k,i+1} - \boldsymbol{z}^{k,i+1},\end{aligned} \tag{21}$$

where S is the soft shareholding operator defined as

$$S_{\kappa}(\alpha) = \begin{cases}\alpha-\kappa, & \alpha > \kappa\\ 0, & |\alpha| \le \kappa\\ \alpha+\kappa, & \alpha < -\kappa.\end{cases} \tag{22}$$

The whole process is summarized in Algorithm 1.

Algorithm 1 IRLS-ADMM regularized logistic regression
1: Set $\boldsymbol{\beta}^{1} = \boldsymbol{0}$
2: **for** $k = 1, \ldots, k_{max}$ **do**
3: Compute $\boldsymbol{A}^{k}$ and $\boldsymbol{b}^{k}$ using Equation 18 and 19.
4: Set $\boldsymbol{z}^{k,1}$, and $\boldsymbol{u}^{k,1}$ to $\boldsymbol{0}$
5: **for** $i = 1, \ldots, i_{max}$ **do**
6: Update $\boldsymbol{\Omega}^{k,i}$, $\boldsymbol{z}^{k,i}$ and $\boldsymbol{u}^{k,i}$ using Equation 21.
7: **end for**
8: Update $\boldsymbol{\beta}_{k}$ using Equation 7, where t is found using a backtracking line-search that minimizes the objective function.
9: **end for**

2) Deep network design

We follow the idea in [8] to map the iterative process in Algorithm 1 to a data flow graph. In the graph, there are 9 types of nodes described as follows:

Weighting matrix Λ^{k} construction layer: Given $\boldsymbol{\beta}^{k}$, the output of this layer is obtained through:

$$\Lambda^{k}_{n,n} = \sigma\{y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}[1-\sigma\{y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)], \tag{23}$$

where $\sigma\{f\} = \frac{1}{(1+e^{-f})}$. In the initial stage, $\boldsymbol{\beta}^{0} = \boldsymbol{0}$.

Weighting vector $\boldsymbol{v}^{k}$ construction layer: Given $\boldsymbol{\beta}^{k}$, the output of this layer is obtained through:

$$\begin{aligned}v^{k}_{n} &= y_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\top}\boldsymbol{\beta}^{k} + \frac{1-\sigma\{y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}}{\Lambda^{k}_{n,n}}\\ &= y_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\top}\boldsymbol{\beta}^{k} + 1 + e^{-y_n \boldsymbol{\beta}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}_n)}.\end{aligned} \tag{24}$$

In the initial stage, $\boldsymbol{\beta}^{0} = \boldsymbol{0}$.

Weighting matrix $\boldsymbol{A}^{k}$ construction layer: Given $\boldsymbol{\Lambda}^{k}$, the output of this layer is obtained through:

$$\boldsymbol{A}^{k} = (\boldsymbol{\Lambda}^{k})^{\frac{1}{2}}\bar{\boldsymbol{\Phi}}. \tag{25}$$

Weighting vector $\boldsymbol{b}^{k}$ construction layer: Given $\boldsymbol{\Lambda}^{k}$ and $\boldsymbol{v}^{k}$, the output of this layer is obtained through:

$$(\Lambda^{k})^{\frac{1}{2}}(v^{k}). \tag{26}$$

Feature $\Omega^{k,i}$ construction layer: Given $\boldsymbol{z}^{k,i-1}$, $\boldsymbol{u}^{k,i-1}$, $\boldsymbol{b}^{k}$ and $\boldsymbol{A}^{k}$, the output of this layer is obtained through:

$$\boldsymbol{\Omega}^{k,i} = (\boldsymbol{A}^{k-1\top}\boldsymbol{A}^{k-1} + \rho^{k,i}\boldsymbol{F}^{k,i\top}\boldsymbol{F}^{k,i})^{-1} (\boldsymbol{A}^{k-1\top}\boldsymbol{b}^{k-1} + \rho^{k,i}\boldsymbol{F}^{k,i\top}(\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})), \quad (27)$$

where $\rho^{k,i}$ and $\boldsymbol{F}^{k,i}$ are parameters. When $i = 1$, $\boldsymbol{z}^{k,0}$ and $\boldsymbol{u}^{k,0}$ are initialized to zeros.

Linear transform layer $\boldsymbol{C}^{k,i}$: Given $\boldsymbol{\Omega}^{k,i}$,the output of this layer is:

$$\boldsymbol{C}^{k,i} = \boldsymbol{F}^{k,i}\boldsymbol{\Omega}^{k,i}, \quad (28)$$

where $\boldsymbol{F}^{k,i}$ is the parameter.

Nonlinear transform layer $\boldsymbol{z}^{k,i}$: Given $\boldsymbol{C}^{k,i}$ and $\boldsymbol{u}^{k,i-1}$,the output of this layer is:

$$\boldsymbol{z}^{k,i} = S_{\lambda/\rho^{k,i}}(\boldsymbol{C}^{k,i} + \boldsymbol{u}^{k,i-1}), \quad (29)$$

where $\rho^{k,i}$ is the parameter. When $i = 1$, $\boldsymbol{u}^{k,0}$ is initialized to zeros.

Linear transform layer $\boldsymbol{u}^{k,i}$: Given $\boldsymbol{C}^{k,i}$, $\boldsymbol{u}^{k,i-1}$ and $\boldsymbol{z}^{k,i}$, the output of this layer is:

$$\boldsymbol{u}^{k,i} = \boldsymbol{u}^{k,i-1} + \boldsymbol{C}^{k,i} - \boldsymbol{z}^{k,i}. \quad (30)$$

When $i = 1$, $\boldsymbol{u}^{k,0}$ is initialized to zeros.

Linear transform layer $\boldsymbol{\beta}^{k}$: Given $\boldsymbol{\Omega}^{k-1,i_{max}}$ and $\boldsymbol{\beta}^{k-1}$, the output of this layer is defined as

$$\boldsymbol{\beta}^{k} = (1 - t^{k})\boldsymbol{\beta}^{k-1} + t^{k}\boldsymbol{\Omega}^{k-1,i_{max}}, \quad (31)$$

where t^{k} is the parameter.

An example deep network with $k_{max} = 3$ and $i_{max} = 4$ are shown in Fig. 1.

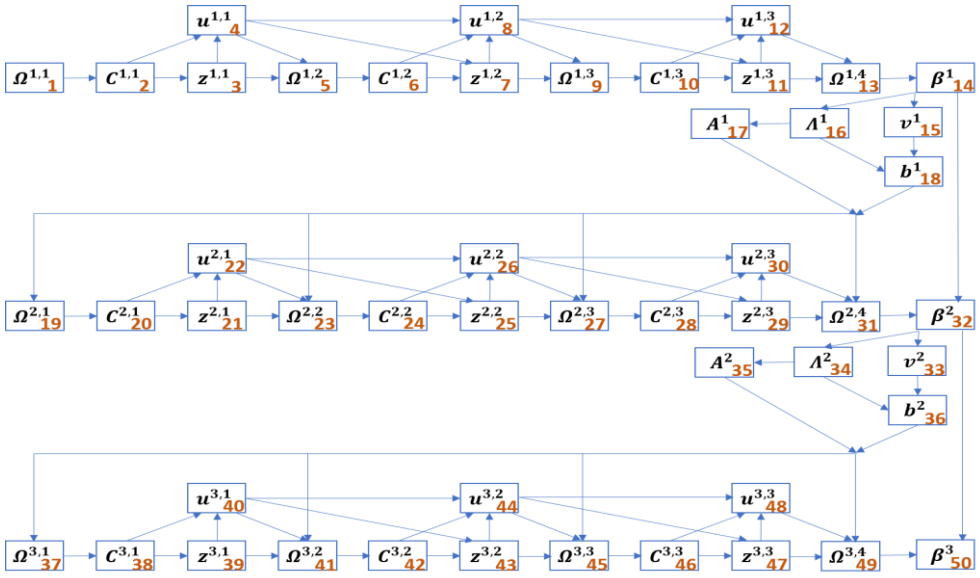


Fig. 1. The architecture of deep IRLS-ADMM net with $\boldsymbol{k_{max}} = \mathbf{3}$ and $\boldsymbol{i_{max}} = \mathbf{4}$.

III. Network Training

A. Network Loss

The loss function we choose for network training is:

$$E = \sum_{n=1}^{N} \log[1 + \exp\{-y_n\boldsymbol{\beta}^{\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}], \quad (32)$$

where $\boldsymbol{\beta}$ is the network output. We learn the parameters $\Theta = \{\rho^{k,i}, \boldsymbol{F}^{k,i}, t^{k}\}$ by minimising the loss w.r.t. them using the stochastic gradient descent method.

B. Gradient Computation by Back-Propagation over Data Flow Graph

Fig. 2 shows 19 types of nodes, most of which have multiple inputs and outputs. The parameters we need to learn are $\Theta = \{\rho^{k,i}, \boldsymbol{F}^{k,i}, t^{k}\}$ for $k \in \{1,2,3\}$ and $i \in \{1,2,3,4\}$. Please refer to the Appendix A for gradients computation for each node.

IV. Experiments

A. Simulation Results

We generate a data matrix with 500 samples and 50 features from the normal distribution. To check the ability of finding correlated features, we split the features with non-zero weights into two sets, S^1 and S^2, each of which contains 4 true features. Then, we initialise a design matrix $\boldsymbol{\Phi} = \boldsymbol{x}$ with 500 samples and 50 features from the normal distribution. Let $\boldsymbol{\phi}_i(\boldsymbol{x})$ be the ith column of $\boldsymbol{\Phi}$. We set $\boldsymbol{\phi}_{S_m^2}(\boldsymbol{x}) = \boldsymbol{\phi}_{S_m^1}(\boldsymbol{x}) + \mathcal{N}(0,0.1)$, where S_m^1 and S_m^2 denote the mth element in each feature set. In this way, we get data generated from correlated features. The target variable vector $\{y_n\}_{n=1}^{500}$ is generated by the linear model using $\boldsymbol{\beta}$ and $\boldsymbol{\Phi}$ with additive independent identical distributed Gaussian noises, where the standard deviation of noise varies ranging from $\{0,0.1,0.5,1\}$. In our method, $\rho^{k,i}$ and t^{k} are initialized to be 1. Each element in the diagonal matrix $\boldsymbol{F}^{k,i}$ is initialized to be 10.

We will first use the whole dataset with no additive noise to construct the predictive model, from which we compare the estimated weights $\boldsymbol{\beta}$ from our method with the regularized logistic regression method (ADMM). It is to investigate the ability of selecting real features using our method. Moreover, we would like to compare the scaled magnitudes of estimated weights $\boldsymbol{\beta}$ with real weights and estimated weights from the regularized logistic regression method. We scale the estimated results to make the first real nonzero feature have the same value. In this way, we can intuitively see whether the estimation has maintained the

ratio in magnitudes of different weights. Thus the rank of the importance of features, which is usually determined by the magnitudes, can be obtained correctly. This is quite important when we use our method as a feature selection approach to select top relevant features in a cross validation process. The results are shown in Fig. 3. Fig. 3 shows the weights from our method and ADMM, from which we can see that there are 50 features, out of which 8 features are nonzero. Our method has successfully detected these nonzero ones with magnitudes significantly larger than the others, while the ADMM method fails to detect 3 nonzero feature. These observations are quite encouraging, showing that our method can be used a good tool to select relevant features. Most classification methods are not guaranteed to have this characteristic.

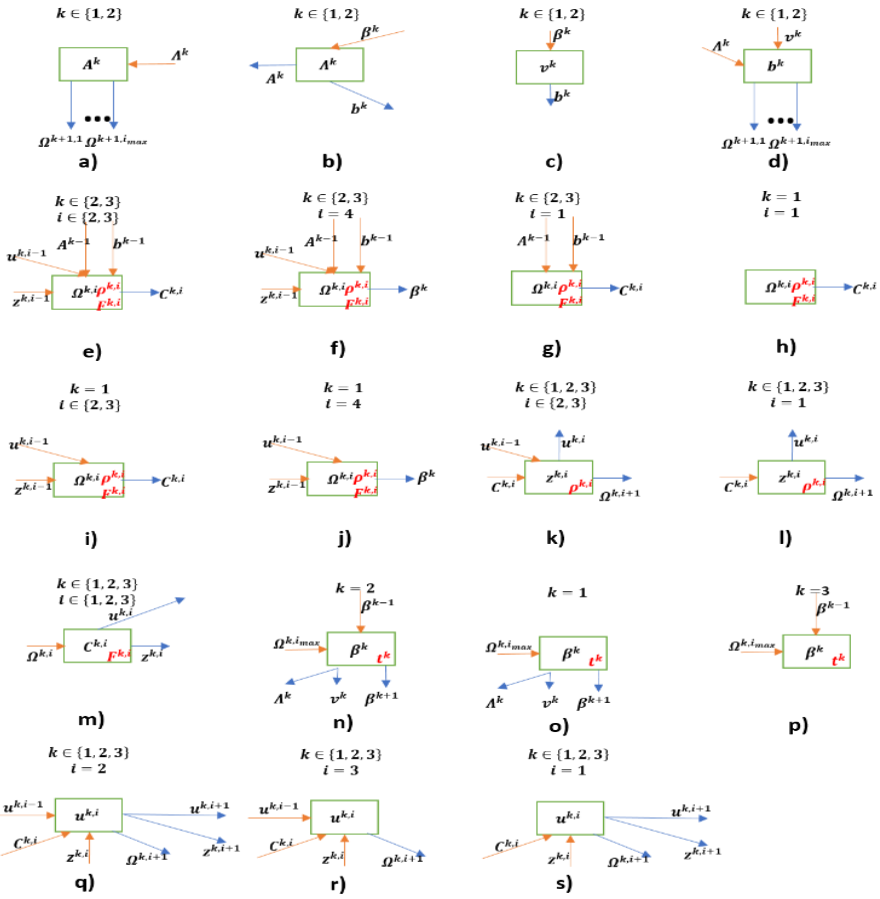


Fig. 2. Illustration of 19 types of nodes. All solid arrows indicate data flow in forward pass.

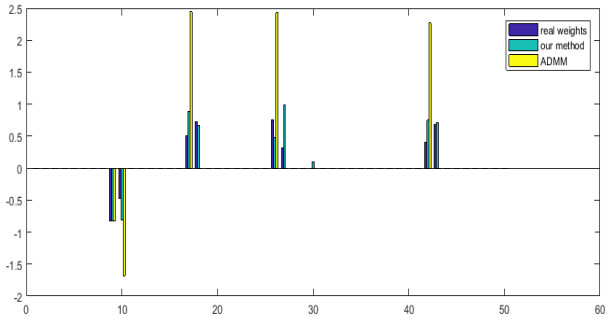


Fig. 3. Comparison of weights estimated from our method and ADMM.

We then investigate the performance of our method with different levels of noise. The performance, including prediction accuracy, false positive (FP) and false negative (FN) rates, is evaluated by the 5-fold cross validation (CV). To calculate FP and FN, we need to compare the selected features across CV with real non-zero features. As in the cross validation process, different training dataset is used for feature selection and predictive model construction in each fold, selected feature sets from all folds may vary due to the variation of training datasets. To select a feature set which is stable with small fluctuations of the input dataset and also has good predictive accuracy, we use the method from [21]. In the kth fold of cross validation, the whole dataset $\boldsymbol{D}$ is split into two subsets: CV training dataset $\boldsymbol{D_k}$ and CV testing dataset $\boldsymbol{D_{\backslash k}}$. Our method can work as a feature selection method on the training dataset $\boldsymbol{D_k}$to rank and select the top q features, labelled as $\boldsymbol{V_{q,k}}$. After features have been selected, our method then constructs a predictive model for classification using $\boldsymbol{V_{q,k}}$. The prediction results at this CV fold are recorded for later evaluation. To get the complete prediction results, we repeat the above steps for all folds of CV. The method presented in [21] returns an optimal feature set $\widetilde{\boldsymbol{V_q}}$ with an associated performance score $\widetilde{P_q}$ under each value of q. The score $\widetilde{P_q}$ is calculated according to the 6th strategy proposed in [22] to assess the prediction accuracy and stability of features (the details of calculating this score can be found in [22]). By checking the maximum value of $\widetilde{P_q}$, we can determine the optimal value of q and the corresponding optimal feature set $\widetilde{\boldsymbol{V_q}}$. The detected optimal set can then be used to construct a predictive model for future prediction.

Fig. 4 shows the change of scores $\widetilde{P}_q$ under different settings of feature size q and noise level. We can see that the scores of our method under different noise levels peak when the value of q is close to the number of real nonzero features. In contrast to our method, the optimal value of q for the ADMM method is around 4, which is different from the real number of nonzero features. From Fig. 4, we can expect that our method works better than the ADMM method in the aspect of feature selection. In the following experiments, we would like to check the performance of these methods averaged over 20 randomly sampled datasets. The results are shown in Table I. The feature set size q is chosen to be the optimal value detected from Fig. 4. To show the ability of detecting real positives (i.e. real nonzero features), we also present the results of false positive (FP) and false negative (FN) rates in Table I. We can see that under different noise levels, the accuracy achieved by different methods are all maintained at high levels. However, the false negative rates from ADMM are much higher than the rates from our method. This is because, the ADMM method can detect correlated features that some real features are ignored.

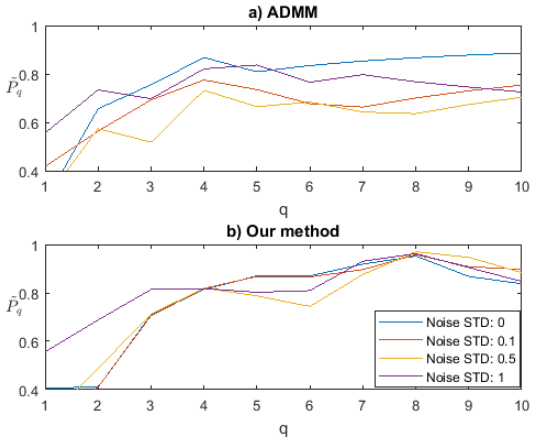


Fig. 4. The scores achieved by different methods with the size of feature set varying from 1 to 8 and standard deviation of noise chosen from 0, 0.1, 0.5 and 1.

TABLE I: THE RESULTS OF OUR METHOD AND ADMM UNDER DIFFERENT NOISE LEVELS. THE CLASSIFICATION ACCURACY, FALSE POSITIVE AND FALSE NEGATIVE RATES ARE COMPARED

Noise STD	Method	Accuracy	FP	FN
0	Our method	0.97	0.02	0
	ADMM	0.98	0.09	0.45
0.1	Our method	0.97	0.02	0
	ADMM	0.98	0.09	0.46
0.5	Our method	0.97	0.02	0
	ADMM	0.96	0.08	0.42
1	Our method	0.96	0.02	0
	ADMM	0.98	0.08	0.44

B. Real Examples

1) Cervical cancer datasets

To demonstrate the applicability of our method in real world problems, we first use the public dataset to test our performance, which can be downloaded from https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29. This dataset focuses on the prediction of diagnosis of cervical cancer. There are 4 indicators (Hinselmann, Schiller, Citology and Biopsy) of disease status. We generate a consensus indicator of disease state that if more than 2 indicators saying the patient has the cancer then we say the patient has the cancer. The dataset comprises of demographic information, habits and historic medical records of patients. We choose records of 737 patients with no missing values and 21 features with at least 10 nonzero records across all patients. The detailed descriptions of features in this risk factor dataset can be found in [23].

We first use the whole dataset to select features used in predictive model construction. The estimated weights are shown in Fig. 5. From Fig. 5, we can see that 5 out of 21 features are mainly used for predicting diagnosis of cervical cancer, which are 'Number of sexual partners', 'First sexual intercourse (age)', 'Number of pregnancies', 'Smokes' and 'Hormonal Contraceptives'. Among these features, the 'First sexual intercourse (age)' feature has the largest absolute magnitude of weight. Its negative value shows that people have their first sexual intercourse at early age are more likely to have cervical cancer. These findings can help researchers to get a proper list of risk factors for cervical cancer prediction. After investigating the ability of feature selection, we then also carry out a 5-fold CV process to evaluate the prediction accuracy of our method. The mean accuracy of our method is as high as 0.95.

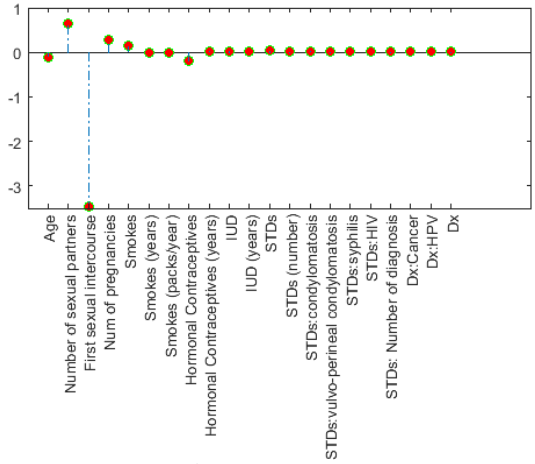


Fig. 5. Weights estimated from our method in the predictive model for cervical cancer risk factors dataset.

2) Embryonal tumour gene expression data analysis

We also use a public available gene expression dataset of the central nervous system embryonal tumours from the study in [24]. The raw data can be downloaded from http://archive.broadinstitute.org/mpr/CNS. We selected 10 CNS medulloblastomas (MD) samples and 10 non-neuronal origin malignant gliomas (Mglio) samples to show the performance of our method in classifying two tumour types. The samples were hybridised on Affymetrix HuGeneFL GeneChip arrays. We first preprocessed the raw data using GCRMA with empirical Bayes estimate [25]. Then we filtered out probe sets which are either not annotated or have little variability across samples. We select probes whose p value from t test for sample type comparison is smaller than 0.05 and fold change is larger than 2. Probes for 350 genes were remained after preprocessing.

Our method can find differences between two tumour types at molecular level. We construct a classifier using the

selected 20 samples with the accuracy of tumour type prediction approaching to 100%. The beauty of our method is that it does not only have strong predictive power, but also selects relevant features that could be candidates of disease biomarkers. Fig. 6a) shows the weights of features in our classification model, where 21 features are mainly used for classification. Features with nonzero weights can be regarded as molecular features distinguishing tumours. By looking at Fig. 6b), we can see that many of these features are highly correlated, telling that our method does not discard features from correlated ones.

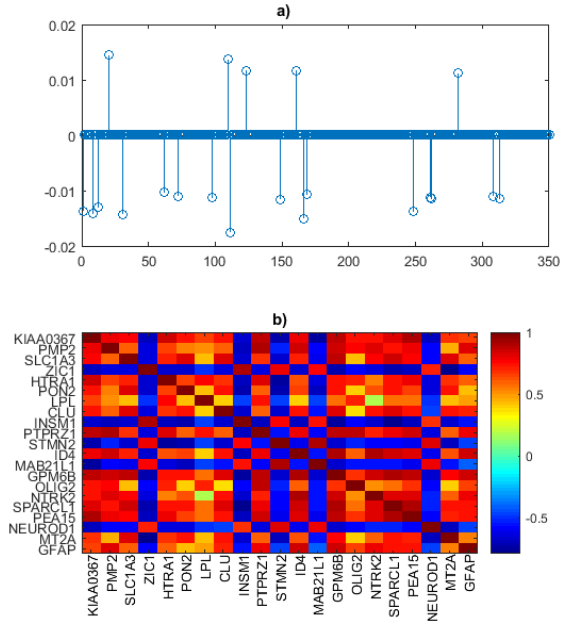


Fig. 6. Results of gene expression analysis: a) weights of features from the classification models using our method; b) Heatmap of the correlation matrix for 21 genes selected by our classifier.

V. Conclusion

This paper adopts the idea of deep unfolding to design a neural network based on IRLS-ADMM for both classification and feature selection. The proposed IRLS-ADMM net is a novel deep neural network. Different from conventional neural networks, which are essentially black-box methods, our method works as a white-box that the network structure are designed according to the iterative updating steps in the model-based model. One advantage of our method is that the network can return a set of features that are used for prediction while maintain good prediction accuracy. Unlike other neural networks, the proposed method can explicitly select features which can be used as a feature selection tool in many applications, such as clinical variable selection and biomarker discovery. Our experimental results have shown that our method has taken the advantages of model-based methods for explicitly incorporating sparsity into model construction and also neural networks with their strong predictive power and straightforward parameter inference process.

Appendix A: Gradient Computation for Deep IRLS-ADMM Net

Node a): As shown in Fig. 2a), this node has one input: $\boldsymbol{\Lambda}^k$ for $k \in \{1,2\}$. Its output $\boldsymbol{A}^k$ is the input to compute $\boldsymbol{\Omega}^{k,i}$ for all $i \in \{1,2,3,4\}$. This layer has no parameters. The loss w.r.t. output is

$$\frac{\partial E}{\partial \boldsymbol{A}^k} = \sum_{i=1}^{4} \frac{\partial E}{\partial \boldsymbol{\Omega}^{k+1,i}} \frac{\partial \boldsymbol{\Omega}^{k+1,i}}{\partial \boldsymbol{A}^k}. \tag{33}$$

The operation in this layer is:

$$\boldsymbol{A}^k = (\boldsymbol{\Lambda}^k)^{\frac{1}{2}} \bar{\boldsymbol{\Phi}}. \tag{34}$$

We compute the gradients of the output in this layer w.r.t its input

$$\frac{\partial \boldsymbol{A}^k}{\partial \Lambda^k_{n,n}} = \frac{1}{2} (\boldsymbol{\Lambda}^k)^{-\frac{1}{2}} \bar{\boldsymbol{I}}^k_{n,n} \bar{\boldsymbol{\Phi}}, \tag{35}$$

where $\bar{\boldsymbol{I}}^k_{n,n}$ is the matrix whose entries are zero except the nth element on the diagonal equalling to 1.

Node b): As shown in Fig. 2b), this node has one input: $\boldsymbol{\beta}^k$. Its output $\boldsymbol{\Lambda}^k$ is the input to compute $\boldsymbol{A}^k$ and $\boldsymbol{b}^k$ for $k \in \{1,2\}$. This layer has no parameters. The loss w.r.t. output is

$$\frac{\partial E}{\partial \boldsymbol{\Lambda}^k} = \frac{\partial E}{\partial \boldsymbol{A}^k} \frac{\partial \boldsymbol{A}^k}{\partial \boldsymbol{\Lambda}^k} + \frac{\partial E}{\partial \boldsymbol{b}^k} \frac{\partial \boldsymbol{b}^k}{\partial \boldsymbol{\Lambda}^k}. \tag{36}$$

The operation in this layer is:

$$\Lambda^k_{n,n} = \sigma\{y_n \beta^{k\top} \phi(x_n)\}[1 - \sigma\{y_n \beta^{k\top} \phi(x_n)\}], \tag{37}$$

where $\sigma\{f\} = \frac{1}{(1+e^{-f})}$. We compute the gradients of the output in this layer w.r.t its input, that is $\frac{\partial \boldsymbol{\Lambda}^k}{\partial \beta^k_m}$ for all m. $\frac{\partial \boldsymbol{\Lambda}^k}{\partial \beta^k_m}$ is a diagonal matrix, whose nth entry on the diagonal is:

$$\frac{\partial \Lambda^k_{n,n}}{\partial \beta^k_m} = y_n \left(\frac{e^{y_n \beta^{k\top} \phi(x_n)}}{(1 + e^{y_n \beta^{k\top} \phi(x_n)})^3} - \frac{e^{-y_n \beta^{k\top} \phi(x_n)}}{(1 + e^{-y_n \beta^{k\top} \phi(x_n)})^3} \right) \phi_m(x_n), \tag{38}$$

where $\boldsymbol{\phi}_m(\boldsymbol{x}_n)$ is the mth element of $\boldsymbol{\phi}(\boldsymbol{x}_n)$.

Node c): As shown in Fig. 2c), this node has one input: $\boldsymbol{\beta}_k$. Its output $\boldsymbol{v}^k$ is the input to compute $\boldsymbol{b}^k$ for $k \in \{1,2\}$. This layer has no parameters. The loss w.r.t. output is

$$\frac{\partial E}{\partial \boldsymbol{v}^k} = \frac{\partial E}{\partial \boldsymbol{b}^k} \frac{\partial \boldsymbol{b}^k}{\partial \boldsymbol{v}^k}. \tag{39}$$

The operation in this layer is:

$$v^k_n = y_n \phi(x_n)^\top \beta^k + 1 + e^{-y_n \beta^{k\top} \phi(x_n)}. \tag{40}$$

We compute the gradients of the output in this layer w.r.t its input, that is $\frac{\partial \boldsymbol{v}^k}{\partial \boldsymbol{\beta}^k}$, whose entry in the nth row and mth column is

$$\frac{\partial v^k_n}{\partial \beta^k_m} = \frac{\partial y_n \phi(x_n)^\top \beta^k + 1 + e^{-y_n \beta^{k\top} \phi(x_n)}}{\partial \beta^k_m} = y_n \phi_m(x_n)(1 - e^{-y_n \beta^{k\top} \phi(x_n)}), \tag{41}$$

where $\boldsymbol{\phi}_m(\boldsymbol{x}_n)$ is the mth element of $\boldsymbol{\phi}(\boldsymbol{x}_n)$.

Node d): As shown in Fig. 2d), this node has two sets of inputs: $\boldsymbol{\Lambda}^k$ and $\boldsymbol{v}^k$. Its output $\boldsymbol{b}^k$ is the input to compute $\boldsymbol{\Omega}^{k,i}$ for all $i \in \{1,2,3,4\}$ and $k \in \{1,2\}$. This layer has no parameters. The loss w.r.t. output is

$$\frac{\partial E}{\partial \boldsymbol{b}^k} = \sum_{i=1}^{4} \frac{\partial E}{\partial \boldsymbol{\Omega}^{k+1,i}} \frac{\partial \boldsymbol{\Omega}^{k+1,i}}{\partial \boldsymbol{b}^k}. \tag{42}$$

The operation in this layer is:

$$\boldsymbol{b}^k = (\boldsymbol{\Lambda}^k)^{\frac{1}{2}}(\boldsymbol{v}^k). \tag{43}$$

We compute the gradients of the output in this layer w.r.t its input

$$\frac{\partial \boldsymbol{b}^k}{\partial \Lambda^k_{n,n}} = \frac{1}{2}(\boldsymbol{\Lambda}^k)^{-\frac{1}{2}} \bar{\boldsymbol{I}}^k_{n,n} \boldsymbol{v}^k \tag{44}$$

and

$$\frac{\partial \boldsymbol{b}^k}{\partial \boldsymbol{v}^k} = (\boldsymbol{\Lambda}^k)^{\frac{1}{2}}, \tag{45}$$

where $\bar{\boldsymbol{I}}^k_{n,n}$ is the matrix whose entries are zero except the nth element on the diagonal equalling to 1.

Node e): As shown in Fig. 2e), this node has four sets of inputs: $\boldsymbol{b}^{k-1}$, $\boldsymbol{A}^{k-1}$, $\boldsymbol{z}^{k,i-1}$ and $\boldsymbol{u}^{k,i-1}$ for $k \in \{2,3\}$ and $i \in \{2,3\}$. Its output $\boldsymbol{\Omega}^{k,i}$ is the input to compute $\boldsymbol{C}^{k,i}$. Parameters of this layer are $\rho^{k,i}$ and $\boldsymbol{F}^{k,i}$. Please note that these two parameters are shared among different layers. Here we just calculate its gradient via the node in this layer. To get the complete the gradient, we need to sum all gradients shared among layers. We will calculate this later. The gradients of loss w.r.t. the parameters can be computed as

$$\frac{\partial E}{\partial \rho^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \rho^{k,i}} \tag{46}$$

and

$$\frac{\partial E}{\partial \boldsymbol{F}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{F}^{k,i}}, \tag{47}$$

where

$$\frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{C}^{k,i}} \frac{\partial \boldsymbol{C}^{k,i}}{\partial \boldsymbol{\Omega}^{k,i}}. \tag{48}$$

To get a complete representation of gradients, we need to do the following calculations. The operation in this layer is:

$$\begin{aligned} \Omega^{k,i} &= (A^{k-1^\top} A^{k-1} + \rho^{k,i} F^{k,i^\top} F^{k,i})^{-1} \\ &\quad (A^{k-1^\top} b^{k-1} + \rho^{k,i} F^{k,i^\top} (z^{k,i-1} - u^{k,i-1})). \end{aligned} \tag{49}$$

Assume

$$Q = (\boldsymbol{A}^{k-1^\top} \boldsymbol{A}^{k-1} + \rho^{k,i} \boldsymbol{F}^{k,i^\top} \boldsymbol{F}^{k,i})^{-1}. \tag{50}$$

The gradients of output in this layer w.r.t. parameters are:

$$\begin{aligned} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \rho^{k,i}} &= -Q^2 \boldsymbol{F}^{k,i^\top} \boldsymbol{F}^{k,i} (\boldsymbol{A}^{k-1^\top} \boldsymbol{b}^{k-1} \\ &\quad + \rho^{k,i} \boldsymbol{F}^{k,i^\top} (\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})) \\ &\quad + Q \boldsymbol{F}^{k,i^\top} (\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1}) \end{aligned} \tag{51}$$

and

$$\begin{aligned} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial F^{k,i}_{m,m}} &= -2\rho^{k,i} Q^2 \bar{\boldsymbol{F}}^{k,i}_{m,m} (\boldsymbol{A}^{k-1^\top} \boldsymbol{b}^{k-1} \\ &\quad + \rho^{k,i} \boldsymbol{F}^{k,i^\top} (\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})) \\ &\quad + \rho^{k,i} Q \bar{\boldsymbol{I}}^{k,i}_{m,m} (\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1}), \end{aligned} \tag{52}$$

where $F^{k,i}_{m,m}$ is the mth entry on diagonal of $\boldsymbol{F}^{k,i}$, $\bar{\boldsymbol{F}}^{k,i}_{m,m}$ and $\bar{\boldsymbol{I}}^{k,i}_{m,m}$ are matrices whose entries are zero except the mth element on the diagonal equalling to $F^{k,i}_{m,m}$ and 1, respectively. We also compute the output in this layer w.r.t. its inputs as follows.

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{b}^{k-1}} = Q \boldsymbol{A}^{k-1^\top} \tag{53}$$

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{z}^{k,i-1}} = \rho^{k,i} Q \boldsymbol{F}^{k,i^\top} \tag{54}$$

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{u}^{k,i-1}} = -\rho^{k,i} Q \boldsymbol{F}^{k,i^\top} \tag{55}$$

$$\begin{aligned} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial A^{k-1}_{n_x,n_y}} &= -Q^2 \bar{\boldsymbol{A}}^{k-1}_{n_x,n_y} (\boldsymbol{A}^{k-1^\top} \boldsymbol{b}^{k-1} \\ &\quad + \rho^{k,i} \boldsymbol{F}^{k,i^\top} (\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})) \\ &\quad + Q \bar{\boldsymbol{T}}^{k-1}_{n_x,n_y} \boldsymbol{b}^{k-1}. \end{aligned} \tag{56}$$

We have

$$\bar{\boldsymbol{A}}^{k-1}_{n_x,n_y} = \tilde{\boldsymbol{A}}^{k-1}_{n_x,n_y} + \tilde{\boldsymbol{A}}^{k-1^\top}_{n_x,n_y}, \tag{57}$$

where $\tilde{\boldsymbol{A}}^{k-1}_{n_x,n_y}$ is a matrix containing zeros except the n_yth row equals to the n_xth row of $\boldsymbol{A}^{k-1}$. And $\bar{\boldsymbol{T}}^{k,i}_{n_x,n_y}$ is a matrix containing zeros except the entry in the n_xth row and n_yth column equals to 1.

Node f): As shown in Fig. 2f), this node is nearly same as the node in Fig. 2e). The only difference is that its output is the input to compute $\boldsymbol{\beta}_k$.

$$\frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\beta}_k} \frac{\partial \boldsymbol{\beta}_k}{\partial \boldsymbol{\Omega}^{k,i}} \tag{58}$$

for $k \in \{2,3\}$ and $i = 4$.

Node g): As shown in Fig. 2g), this node is similar with the node in Fig. 2e). The only difference is that it only have two inputs: $\boldsymbol{b}^{k-1}$ and $\boldsymbol{A}^{k-1}$ for $k \in \{2,3\}$ and $i = 1$. The operation in this layer is:

$$\boldsymbol{\Omega}^{k,i} = (\boldsymbol{A}^{k-1^\top} \boldsymbol{A}^{k-1} + \rho^{k,i} \boldsymbol{F}^{k,i^\top} \boldsymbol{F}^{k,i})^{-1} \boldsymbol{A}^{k-1^\top} \boldsymbol{b}^{k-1}. \tag{59}$$

The gradients of output in this layer w.r.t. parameters are:

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \rho^{k,i}} = -Q^2 \boldsymbol{F}^{k,i^\top} \boldsymbol{F}^{k,i} \boldsymbol{A}^{k-1^\top} \boldsymbol{b}^{k-1} \tag{60}$$

and

$$\frac{\partial \Omega^{k,i}}{\partial F^{k,i}_{m,m}} = -2\rho^{k,i} Q^2 \bar{F}^{k,i}_{m,m} A^{k-1^\top} b^{k-1}. \tag{61}$$

The output in this layer w.r.t. its inputs as follows.

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{b}^{k-1}} = Q \boldsymbol{A}^{k-1^\top} \tag{62}$$

and

$$\begin{aligned} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial A^{k-1}_{n_x,n_y}} &= -Q^2 \bar{\boldsymbol{A}}^{k-1}_{n_x,n_y} \boldsymbol{A}^{k-1^\top} \boldsymbol{b}^{k-1} \\ &\quad + Q \bar{\boldsymbol{T}}^{k-1}_{n_x,n_y} \boldsymbol{b}^{k-1}. \end{aligned} \tag{63}$$

Node h): As shown in Fig. 2h), this node has no input. Its output $\Omega^{k,i}$ is the input to compute $\boldsymbol{C}^{k,i}$ for $k = 1$ and $i = 1$. The operation in this layer is:

$$\boldsymbol{\Omega}^{1,1} = (\boldsymbol{A}^{0\top}\boldsymbol{A}^{0} + \rho^{1,1}\boldsymbol{F}^{1,1\top}\boldsymbol{F}^{1,1})^{-1}\boldsymbol{A}^{0\top}\boldsymbol{b}^{0}. \quad (64)$$

where

$$\boldsymbol{A}^{0} = (\boldsymbol{\Lambda}^{0})^{\frac{1}{2}}\bar{\boldsymbol{\Phi}} \quad (65)$$

$$\boldsymbol{b}^{0} = (\boldsymbol{\Lambda}^{0})^{\frac{1}{2}}\boldsymbol{v}^{0}. \quad (66)$$

As $\boldsymbol{\beta}^{0} = \boldsymbol{0}$, we get $\boldsymbol{\Lambda}^{0} = \frac{1}{4}$ and $\boldsymbol{v}^{0} = 2$. Then we have

$$\boldsymbol{A}^{0} = \frac{1}{2}\bar{\boldsymbol{\Phi}} \quad (67)$$

$$\boldsymbol{b}^{0} = \vec{1}. \quad (68)$$

Parameters of this layer are $\rho^{1,1}$ and $\boldsymbol{F}^{1,1}$. The gradients of loss w.r.t. the parameters can be computed as

$$\frac{\partial E}{\partial \rho^{1,1}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{1,1}}\frac{\partial \boldsymbol{\Omega}^{1,1}}{\partial \rho^{1,1}} \quad (69)$$

and

$$\frac{\partial E}{\partial \boldsymbol{F}^{1,1}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{1,1}}\frac{\partial \boldsymbol{\Omega}^{1,1}}{\partial \boldsymbol{F}^{\boldsymbol{1,1}}}, \quad (70)$$

where

$$\frac{\partial E}{\partial \boldsymbol{\Omega}^{1,1}} = \frac{\partial E}{\partial \boldsymbol{C}^{1,1}}\frac{\partial \boldsymbol{C}^{1,1}}{\partial \boldsymbol{\Omega}^{1,1}}. \quad (71)$$

The gradients of output in this layer w.r.t. parameters are:

$$\frac{\partial \boldsymbol{\Omega}^{1,1}}{\partial \rho^{1,1}} = -(\boldsymbol{A}^{0\top}\boldsymbol{A}^{0} + \rho^{1,1}\boldsymbol{F}^{1,1\top}\boldsymbol{F}^{1,1})^{-2} \boldsymbol{F}^{1,1\top}\boldsymbol{F}^{1,1}\boldsymbol{A}^{0\top}\boldsymbol{b}^{0} \quad (72)$$

and

$$\frac{\partial \boldsymbol{\Omega}^{1,1}}{\partial F^{1,1}_{m,m}} = -2\rho^{1,1}(\boldsymbol{A}^{0\top}\boldsymbol{A}^{0} + \rho^{1,1}\boldsymbol{F}^{1,1\top}\boldsymbol{F}^{1,1})^{-2} \bar{\boldsymbol{F}}^{k,i}_{m,m}\boldsymbol{A}^{0\top}\boldsymbol{b}^{0}. \quad (73)$$

Node i) As shown in Fig. 2i), this node is similar with the node in Fig. 2e). The only difference is that it only have two inputs: $\boldsymbol{z}^{k,i-1}$ and $\boldsymbol{u}^{k,i-1}$ for $k = 1$ and $i \in \{2,3\}$. The operation in this layer is:

$$\boldsymbol{\Omega}^{k,i} = (\boldsymbol{A}^{0\top}\boldsymbol{A}^{0} + \rho^{k,i}\boldsymbol{F}^{k,i\top}\boldsymbol{F}^{k,i})^{-1}(\boldsymbol{A}^{0\top}\boldsymbol{b}^{0} + \rho^{k,i}\boldsymbol{F}^{k,i\top}(\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})) \quad (74)$$

where

$$\boldsymbol{A}^{0} = \frac{1}{2}\bar{\boldsymbol{\Phi}} \quad (75)$$

$$\boldsymbol{b}^{0} = \vec{\boldsymbol{1}}. \quad (76)$$

The gradients of output in this layer w.r.t. parameters are:

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \rho^{k,i}} = -Q^{2}\boldsymbol{F}^{k,i\top}\boldsymbol{F}^{k,i}(\boldsymbol{A}^{0\top}\boldsymbol{b}^{0} + \rho^{k,i}\boldsymbol{F}^{k,i\top}(\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})) + Q\boldsymbol{F}^{k,i\top}(\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1}) \quad (77)$$

and

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial F^{k,i}_{m,m}} = -2\rho^{k,i}Q^{2}\bar{\boldsymbol{F}}^{k,i}_{m,m}(\boldsymbol{A}^{0\top}b^{\boldsymbol{0}} + \rho^{k,i}\boldsymbol{F}^{k,i\top}(\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1})) + \rho^{k,i}Q\bar{\boldsymbol{I}}^{k,i}_{m,m}(\boldsymbol{z}^{k,i-1} - \boldsymbol{u}^{k,i-1}). \quad (78)$$

The output in this layer w.r.t. its inputs as follows.

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{z}^{k,i-1}} = \rho^{k,i}Q\boldsymbol{F}^{k,i\top} \quad (79)$$

$$\frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{u}^{k,i-1}} = -\rho^{k,i}Q\boldsymbol{F}^{k,i\top}. \quad (80)$$

Node j) As shown in Fig. 2j), this node is similar with the node in Fig. 2i). The only difference is that its output is the input to compute $\boldsymbol{\beta}_{k}$.

$$\frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\beta}_{k}}\frac{\partial \boldsymbol{\beta}_{k}}{\partial \boldsymbol{\Omega}^{k,i}} \quad (81)$$

for $k = 1$ and $i = 4$.

Node k): As shown in Fig. 2k), this node has two sets of inputs: $\boldsymbol{u}^{k,i-1}$ and $\boldsymbol{C}^{k,i}$. Its output $\boldsymbol{z}^{k,i}$ is the input to compute $\boldsymbol{u}^{k,i}$ and $\boldsymbol{\Omega}^{k,i+1}$ for $k \in \{1,2,3\}$ and $i \in \{2,3\}$. Parameter of this layer is $\rho^{k,i}$. The operation in this layer is

$$\boldsymbol{z}^{k,i} = S_{\lambda/\rho^{k,i}}(\boldsymbol{C}^{k,i} + \boldsymbol{u}^{k,i-1}), \quad (82)$$

where λ is 1, whose nth element is

$$z^{k,i}_{m} = \begin{cases} C^{k,i}_{m} + u^{k,i-1}_{m} - 1/\rho^{k,i}, & C^{k,i}_{m} + u^{k,i-1}_{m} > 1/\rho^{k,i} \\ 0, & |C^{k,i}_{m} + u^{k,i-1}_{m}| \le 1/\rho^{k,i} \\ C^{k,i}_{m} + u^{k,i-1}_{m} + 1/\rho^{k,i}, & C^{k,i}_{m} + u^{k,i-1}_{m} < -1/\rho^{k,i}. \end{cases} \quad (83)$$

The gradients of the loss w.r.t. parameters are

$$\frac{\partial E}{\partial \rho^{k,i}} = \frac{\partial E}{\partial \boldsymbol{z}^{k,i}}\frac{\partial \boldsymbol{z}^{k,i}}{\partial \rho^{k,i}}, \quad (84)$$

where

$$\frac{\partial E}{\partial \boldsymbol{z}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{u}^{k,i}}\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{z}^{k,i}} + \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i+1}}\frac{\partial \boldsymbol{\Omega}^{k,i+1}}{\partial \boldsymbol{z}^{k,i}}. \quad (85)$$

We compute the gradients of the output in this layer w.r.t. parameter as

$$\frac{\partial z^{k,i}_{m}}{\partial \rho^{k,i}} = \begin{cases} \rho^{k,i\,-2}, & C^{k,i}_{m} + u^{k,i-1}_{m} > 1/\rho^{k,i} \\ 0, & |C^{k,i}_{m} + u^{k,i-1}_{m}| \le 1/\rho^{k,i} \\ -\rho^{k,i\,-2}, & C^{k,i}_{m} + u^{k,i-1}_{m} < -1/\rho^{k,i} \end{cases} \quad (86)$$

and the gradients of the output in this layer w.r.t. the input as

$$\frac{\partial z^{k,i}_{m}}{\partial C^{k,i}_{m}} = \frac{\partial z^{k,i}_{m}}{\partial u^{k,i-1}_{m}} = \begin{cases} 1, & C^{k,i}_{m} + u^{k,i-1}_{m} > 1/\rho^{k,i} \\ 0, & |C^{k,i}_{m} + u^{k,i-1}_{m}| \le 1/\rho^{k,i} \\ 1, & C^{k,i}_{m} + u^{k,i-1}_{m} < -1/\rho^{k,i}. \end{cases} \quad (87)$$

Node l): As shown in Fig. 2l), this node is similar with the node in Fig. 2k). The only difference is that it only has one input: $\boldsymbol{C}^{k,i}$ for $k \in \{1,2,3\}$ and $i = 1$. Parameter of this layer is $\rho^{k,i}$. The operation in this layer is

$$\boldsymbol{z}^{k,i} = S_{\lambda/\rho^{k,i}}(\boldsymbol{C}^{k,i}), \quad (88)$$

where λ is 1, whose nth element is

$$z^{k,i}_{m} = \begin{cases} C^{k,i}_{m} - 1/\rho^{k,i}, & C^{k,i}_{m} > 1/\rho^{k,i} \\ 0, & |C^{k,i}_{m}| \le 1/\rho^{k,i} \\ C^{k,i}_{m} + 1/\rho^{k,i}, & C^{k,i}_{m} < -1/\rho^{k,i}, \end{cases} \quad (89)$$

The gradients of the loss w.r.t. parameters are

$$\frac{\partial E}{\partial \rho^{k,i}} = \frac{\partial E}{\partial \boldsymbol{z}^{k,i}}\frac{\partial \boldsymbol{z}^{k,i}}{\partial \rho^{k,i}}, \quad (90)$$

where

$$\frac{\partial E}{\partial \boldsymbol{z}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{u}^{k,i}}\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{z}^{k,i}} + \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i+1}}\frac{\partial \boldsymbol{\Omega}^{k,i+1}}{\partial \boldsymbol{z}^{k,i}}. \quad (91)$$

We compute the gradients of the output in this layer w.r.t. parameter as

$$\frac{\partial z_m^{k,i}}{\partial \rho^{k,i}} = \begin{cases} {\rho^{k,i}}^{-2}, & C_m^{k,i} > 1/\rho^{k,i} \\ 0, & |C_m^{k,i}| \le 1/\rho^{k,i} \\ -{\rho^{k,i}}^{-2}, & C_m^{k,i} < -1/\rho^{k,i} \end{cases} \quad (92)$$

and the gradients of the output in this layer w.r.t. the input as

$$\frac{\partial z_m^{k,i}}{\partial C_m^{k,i}} = \begin{cases} 1, & C_m^{k,i} > 1/\rho^{k,i} \\ 0, & |C_m^{k,i}| \le 1/\rho^{k,i} \\ 1, & C_m^{k,i} < -1/\rho^{k,i}. \end{cases} \quad (93)$$

Node m) As shown in Fig. 2m), this node has one input: $\Omega^{k,i}$ for $k \in \{1,2,3\}$ and $i \in \{1,2,3\}$. Its output $\boldsymbol{C}^{k,i}$ is used to compute $\boldsymbol{u}^{k,i}$ and $\boldsymbol{z}^{k,i}$. Parameter of this layer is $\boldsymbol{F}^{k,i}$. The gradients of loss w.r.t the parameter can be calculated as

$$\frac{\partial E}{\partial \boldsymbol{F}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{C}^{k,i}}\frac{\partial \boldsymbol{C}^{k,i}}{\partial \boldsymbol{F}^{k,i}}, \quad (94)$$

where

$$\frac{\partial E}{\partial \boldsymbol{C}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{u}^{k,i}}\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{C}^{k,i}} + \frac{\partial E}{\partial \boldsymbol{z}^{k,i}}\frac{\partial \boldsymbol{z}^{k,i}}{\partial \boldsymbol{C}^{k,i}}. \quad (95)$$

The operation in this layer is:

$$\boldsymbol{C}^{k,i} = \boldsymbol{F}^{k,i}\boldsymbol{\Omega}^{k,i}. \quad (96)$$

We compute the gradients of the output in this layer w.r.t parameters

$$\frac{\partial \boldsymbol{C}^{k,i}}{\partial F_{m,m}^{k,i}} = \bar{\boldsymbol{I}}_{m,m}^{k,i}\boldsymbol{\Omega}^{k,i}. \quad (97)$$

The output in this layer w.r.t its inputs are

$$\frac{\partial \boldsymbol{C}^{k,i}}{\partial \boldsymbol{\Omega}^{k,i}} = \boldsymbol{F}^{k,i}. \quad (98)$$

Node n): As shown in Fig. 2n), this node has two sets of inputs: $\boldsymbol{\beta}^{k-1}$ and $\Omega^{k,i_{max}}$. Its output $\boldsymbol{\beta}^{k}$ is the input to compute Λ^k, $\boldsymbol{v}^k$ and $\boldsymbol{\beta}^{k+1}$ for $k = 2$. Parameters of this layer is t^k. The operation in this layer is

$$\boldsymbol{\beta}^{k} = (1 - t^k)\boldsymbol{\beta}^{k-1} + t^k\boldsymbol{\Omega}^{k,i_{max}}. \quad (99)$$

The gradients of loss w.r.t. the parameter can be calculated as

$$\frac{\partial E}{\partial t^k} = \frac{\partial E}{\partial \boldsymbol{\beta}^k}\frac{\partial \boldsymbol{\beta}^k}{\partial t^k}, \quad (100)$$

where

$$\frac{\partial E}{\partial \boldsymbol{\beta}^k} = \frac{\partial E}{\partial \boldsymbol{\Lambda}^k}\frac{\partial \boldsymbol{\Lambda}^k}{\partial \boldsymbol{\beta}^k} + \frac{\partial E}{\partial \boldsymbol{v}^k}\frac{\partial \boldsymbol{v}^k}{\partial \boldsymbol{\beta}^k} + \frac{\partial E}{\partial \boldsymbol{\beta}^{k+1}}\frac{\partial \boldsymbol{\beta}^{k+1}}{\partial \boldsymbol{\beta}^k}. \quad (101)$$

We compute the gradients of the output in this layer w.r.t. parameter as

$$\frac{\partial \boldsymbol{\beta}^k}{\partial t^k} = -\boldsymbol{\beta}^{k-1} + \boldsymbol{\Omega}^{k,i_{max}} \quad (102)$$

and the gradients of the output in this layer w.r.t. the input as

$$\frac{\partial \boldsymbol{\beta}^k}{\partial \boldsymbol{\beta}^{k-1}} = (1 - t^k)\boldsymbol{I}_{M,M} \quad (103)$$

$$\frac{\partial \boldsymbol{\beta}^k}{\partial \boldsymbol{\Omega}^{k,i_{max}}} = t^k\boldsymbol{I}_{M,M}. \quad (104)$$

Node o): As shown in Fig. 2o), this node is similar with the node in Fig. 2n). The only difference is that it only has one input: $\boldsymbol{\Omega}^{k,i_{max}}$ for $k = 1$. Parameters of this layer is t^k. The operation in this layer is

$$\boldsymbol{\beta}^{k} = t^k\boldsymbol{\Omega}^{k,i_{max}}. \quad (105)$$

The gradients of the output in this layer w.r.t. parameter as

$$\frac{\partial \boldsymbol{\beta}^k}{\partial t^k} = \boldsymbol{\Omega}^{k,i_{max}} \quad (106)$$

and the gradients of the output in this layer w.r.t. the input as

$$\frac{\partial \boldsymbol{\beta}^k}{\partial \boldsymbol{\Omega}^{k,i_{max}}} = t^k\boldsymbol{I}_{M,M}. \quad (107)$$

Node p) As shown in Fig. 2p), this node is similar with the node in Fig. 2n). The only difference is that its output is not the input of any other nodes for $k = 3$. That is $\frac{\partial E}{\partial \boldsymbol{\beta}^k}$ can be calculated directly:

$$\frac{\partial E}{\partial \boldsymbol{\beta}^k} = \frac{\partial \sum_{n=1}^{N} log\,[1 + exp\,\{-y_n\boldsymbol{\beta}^{\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}]}{\partial \boldsymbol{\beta}^k} = -\sum_{n=1}^{N} y_n \frac{exp\,\{-y_n\boldsymbol{\beta}^{\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}}{1 + exp\,\{-y_n\boldsymbol{\beta}^{\top}\boldsymbol{\phi}(\boldsymbol{x}_n)\}}\boldsymbol{\phi}(\boldsymbol{x}_n)^{\top}. \quad (108)$$

Node q): As shown in Fig. 2q), this node has three sets of inputs: $\boldsymbol{u}^{k,i-1}$, $\boldsymbol{C}^{k,i}$ and $\boldsymbol{z}^{k,i}$. Its output $\boldsymbol{u}^{k,i}$ is the input to compute $\boldsymbol{u}^{k,i+1}$, $\boldsymbol{z}^{k,i+1}$ and $\boldsymbol{\Omega}^{k,i+1}$ for $k \in \{1,2,3\}$ and $i = 2$. This layer has no parameters. The loss w.r.t. output is

$$\frac{\partial E}{\partial \boldsymbol{u}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{u}^{k,i+1}}\frac{\partial \boldsymbol{u}^{k,i+1}}{\partial \boldsymbol{u}^{k,i}} + \frac{\partial E}{\partial \boldsymbol{z}^{k,i+1}}\frac{\partial \boldsymbol{z}^{k,i+1}}{\partial \boldsymbol{u}^{k,i}} + \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i+1}}\frac{\partial \boldsymbol{\Omega}^{k,i+1}}{\partial \boldsymbol{u}^{k,i}}. \quad (109)$$

The operation in this layer is:

$$\boldsymbol{u}^{k,i} = \boldsymbol{u}^{k,i-1} + \boldsymbol{C}^{k,i} - \boldsymbol{z}^{k,i}. \quad (110)$$

The output in this layer w.r.t its inputs are

$$\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{u}^{k,i-1}} = \boldsymbol{I}_{M,M} \quad (111)$$

$$\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{C}^{k,i}} = \boldsymbol{I}_{M,M} \quad (112)$$

$$\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{z}^{k,i}} = -\boldsymbol{I}_{M,M}, \quad (113)$$

where M is the number of elements in $\boldsymbol{\beta}_k$ and $\boldsymbol{I}_{M,M}$ is the identity matrix with the size of $M \times M$.

Node r) As shown in Fig. 2r), this node is similar with the node in Fig. 2q). The only difference is that its output $\boldsymbol{u}^{k,i}$ is the input to compute $\boldsymbol{\Omega}^{k,i+1}$ for $k \in \{1,2,3\}$ and $i = 3$. The loss w.r.t. output is

$$\frac{\partial E}{\partial \boldsymbol{u}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i+1}}\frac{\partial \boldsymbol{\Omega}^{k,i+1}}{\partial \boldsymbol{u}^{k,i}}. \quad (114)$$

Node s) As shown in Fig. 2t), this node is similar with the node in Fig. 2q). The only difference is that it has only two inputs: $\boldsymbol{C}^{k,i}$ and $\boldsymbol{z}^{k,i}$ for $k \in 1,2,3$ and $i = 1$. The operation in this layer is:

$$\boldsymbol{u}^{k,i} = \boldsymbol{C}^{k,i} - \boldsymbol{z}^{k,i}. \tag{115}$$

The output in this layer w.r.t its inputs are

$$\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{C}^{k,i}} = \boldsymbol{I}_{M,M} \tag{116}$$

$$\frac{\partial \boldsymbol{u}^{k,i}}{\partial \boldsymbol{z}^{k,i}} = -\boldsymbol{I}_{M,M}. \tag{117}$$

Gradients of parameters shared among layers): As we have discussed in node e), parameters may share among different layers. In the above calculations, we only calculate the gradients of parameters dependent on the output of this node. To get a complete expression of gradients, we need to sum them together. Thus, we have

$$\frac{\partial E}{\partial \rho^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \rho^{k,i}} + \frac{\partial E}{\partial \boldsymbol{z}^{k,i}} \frac{\partial \boldsymbol{z}^{k,i}}{\partial \rho^{k,i}} \tag{118}$$

$$\frac{\partial E}{\partial \boldsymbol{F}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{F}^{k,i}} + \frac{\partial E}{\partial \boldsymbol{C}^{k,i}} \frac{\partial \boldsymbol{C}^{k,i}}{\partial \boldsymbol{F}^{k,i}} \tag{119}$$

for $i \in \{1,2,3\}$, and

$$\frac{\partial E}{\partial \rho^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \rho^{k,i}} \tag{120}$$

$$\frac{\partial E}{\partial \boldsymbol{F}^{k,i}} = \frac{\partial E}{\partial \boldsymbol{\Omega}^{k,i}} \frac{\partial \boldsymbol{\Omega}^{k,i}}{\partial \boldsymbol{F}^{k,i}} \tag{121}$$

for $i = 4$.

REFERENCES

[1] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint arXiv:1409.2574*, 2014.

[2] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, 2001, pp. 556–562.

[3] S. Wisdom, J. Hershey, J. Le Roux, and S. Watanabe, "Deep unfolding for multichannel source separation," in *Proc. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 121–125.

[4] C.-H. Lee and J.-T. Chien, "Deep unfolding inference for supervised topic model," in *Proc. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE*, 2016, pp. 2279–2283.

[5] R. Vemulapalli, O. Tuzel, and M.-Y. Liu, "Deep gaussian conditional random field network: A model-based deep network for discriminative denoising," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4801–4809.

[6] H. He, B. Xin, and D. Wipf, "From sparse bayesian learning to deep recurrent nets," *NIPS*, 2017

[7] D. Mahapatra, S. Mukherjee, and C. S. Seelamantula, "Deep sparse coding using optimized linear expansion of thresholds," *arXiv preprint arXiv:1705.07290*, 2017.

[8] J. Sun, H. Li, Z. Xu *et al*., "Deep admm-net for compressive sensing mri," *Advances in Neural Information Processing Systems*, pp. 10–18, 2016.

[9] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[10] W. Pan, "Bayesian learning for nonlinear system identification," Ph.D. dissertation, Imperial College London, 2015.

[11] Y. Li, C. Campbell, and M. Tipping, "Bayesian automatic relevance determination algorithms for classifying gene expression data," *Bioinformatics*, vol. 18, no. 10, pp. 1332–1339, 2002.

[12] B. Krishnapuram, L. Carin, and A. J. Hartemink, "Joint classifier and feature optimization for comprehensive cancer diagnosis using gene expression data," *Journal of Computational Biology*, vol. 11, no. 2-3, pp. 227–242, 2004.

[13] G. C. Cawley and N. L. Talbot, "Gene selection in cancer classification using sparse logistic regression with bayesian regularization," *Bioinformatics*, vol. 22, no. 19, pp. 2348–2355, 2006.

[14] X. Yang, W. Pan, and Y. Guo, "Sparse bayesian classification and feature selection for biological expression data with high correlations," *PLOS One*, vol. 12, no. 12, 2017.

[15] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.

[16] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng, "Efficient l1 regularized logistic regression," in *AAAI*, vol. 6, 2006, pp. 401–408.

[17] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[18] T. P. Minka. (2003). A comparison of numerical optimizers for logistic regression. [Online]. Available: http://research.microsoft.com/∼minka/papers/logreg

[19] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optimization and Engineering*, vol. 17, no. 2, pp. 263–287, 2016.

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and TrendsR in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

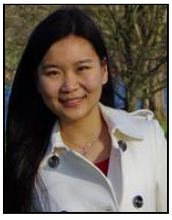
[21] S. Yan, X. Yang, C. Wu, Z. Zheng, and Y. Guo, "Balancing the stability and predictive performance for multivariate voxel selection in fmri study," in *Proc. International Conference on Brain Informatics and Health*. Springer, 2014, pp. 90–99.

[22] P. Kirk, A. Witkover, C. R. Bangham, S. Richardson, A. M. Lewin, and M. P. Stumpf, "Balancing the robustness and predictive performance of biomarkers," *Journal of Computational Biology*, vol. 20, no. 12, pp. 979–989, 2013.

[23] K. Fernandes, J. S. Cardoso, and J. Fernandes, "Transfer learning with partial observability applied to cervical cancer screening," in *Proc. Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2017, pp. 243–250.

[24] S. L. Pomeroy, P. Tamayo *et al*., "Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, no. 6870, pp. 436–442, 2002.

[25] Z. Wu, R. A. Irizarry, R. Gentleman, F. Martinez-Murillo, and F. Spencer, "A model-based background adjustment for oligonucleotide expression arrays," *Journal of the American statistical Association*, vol. 99, no. 468, pp. 909–917, 2004.



Xian Yang received her PhD degree from the Department of Computing, Imperial College London in 2016. She has been working as a research assistant at Imperial College London since 2012 and became a research associate in 2016. Her research interests include machine learning, neuroimaging, bioinformatics, system biology, data mining, statistics and health informatics. She has taken part in various projects, such as the UBIOPRED project on the severe asthma study, the iHealth project on clinical pathway management, and the eTRIKS project on knowledge management for precision medicine research.

Yike Guo is a professor of computing science in the Department of Computing at Imperial College London. He is the founding director of the Data Science Institute at Imperial College, as well as leading the Discovery Science Group in the department. He has been working on technology and platforms for scientific data analysis, where his research focuses on knowledge discovery, data mining and large-scale data management. Professor Guo also holds the position of CTO of the tranSMART Foundation, a global open source community using and developing data sharing and analytics technology for translational medicine, and the position of CIO of IDBS, a world's leading company in developing innovative data management and analytics solutions.