CMSC5743 2021F Homework 1

Due: Oct. 14, 2021

Solutions

All solutions should be submitted to the blackboard in the format of **PDF/MS Word**.

Q1 (12%)

(a) (4%) We provide a very simple neural network as shown in Figure 1, please calculate the result in the blank neuron.

(b) (4%) If we choose to prune one weight, which weight do you choose to achieve the best result? What's your evaluation metric?

(c) (4%) If you have a chance to prune any weights, what's your pruning plan to make a better tradeoff between accuracy and the number of weights?

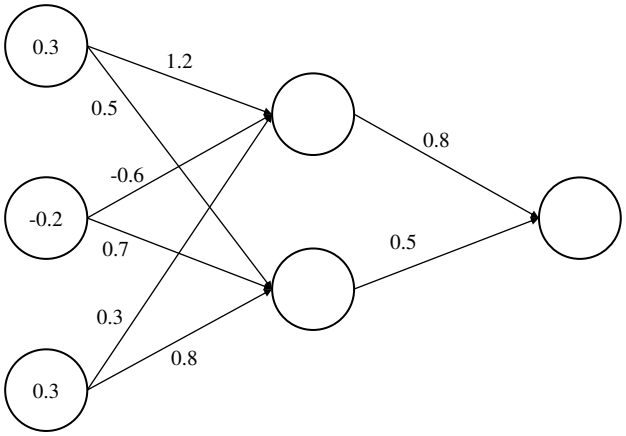

Figure 1: A simple 2-layer neural network

A Answer:

(a) 0.57, 0.25, 0.581

(b) Prune weight 0.3 (between neuron 0.5 and neuron 0.57) Evaluation Metric: The final output error.

(c) Prune weight 0.7 (between neuron 0.3 and neuron 0.25) and prune weight 0.5 (between neuron 0.3 and neuron 0.25).

Q2 (13%)

(a) (5%) Some people may argue that why we do not simply train a smaller neural network instead of pruning a neural network with huge amount of parameters. Have you thought about this problem? What are the advantages of network pruning over training a smaller network? Please list two points and provide as much support as possible.

(b) (4%) If someone want to apply structured pruning of fixed proportion for each layer, is it necessary?

(c) (4%) If someone want to apply unstructured pruning of unfixed proportion for each layer, is it necessary?

A Answer:

- Training a smaller network has to specify the network structure in advance, while pruning can search a suitable network structure under the given parameter amount condition
- Training a smaller network is always relatively difficult to achieve as good result as pruning since the latter one can be fine-tuned from a pre-trained big network while the former one has to be trained from scratch.
- Pruning can push the parameter to low-bit under some conditions which is beneficial for hardware deployment.
- Unnecessary
- Necessary

Q3 (12%) **Regularization**. In this question, you are going to solve a toy problem. Consider a function $J(x) = (x-2)^2, x \in \mathbb{R}$.

(a) (3%) Find the global minimum of function $J(x) + 6x^2$. Justify your answer.

(b) (3%) Find the global minimum of function $J(x) + 6|x|$. Justify your answer.

(c) (3%) Consider the following optimization problem.

$$\min_{x \in \mathbb{R}} J(\alpha; x) = (x-2)^2 + \alpha|x|.$$

How should we determine α so that the minimizer is at $x = 0$?

(d) (3%) How do you get inspired from the above questions about ℓ_1, ℓ_2 and sparsity? Please explain *briefly*.

A Answer.

(a) Let $J'(x) = 14x - 4 = 0$, we have $x^* = \frac{2}{7}$. Then $J^* = \frac{24}{7}$.

(b) $J(x) = x^2 + 2x + 4$ when $x \geq 0$, while $J(x) = x^2 - 10x + 4$ when $x < 0$. Therefore, $x^* = 0$ and $J^* = 4$.

(c) Similar to (b), the answer is $\alpha \geq 4$.

(d) Compared to ℓ_2, ℓ_1 regularization (or LASSO, or anything similar) leads to sparsity.

Q4 (13%) ℓ_0**-norm**. Consider the p-norm (or ℓ_p-norm) of a vector $\boldsymbol{x} = [x_1, \cdots, x_n]^\top$

$$\|\boldsymbol{x}\|_p = \sqrt[p]{|x_1|^p + |x_2|^p + \cdots + |x_n|^p}, \tag{1}$$

where integer $p \geq 1$ and the dimension n is fixed.

(a) (4%) If we have a vector $\boldsymbol{x}$ whose ℓ_2-norm $\|\boldsymbol{x}\|_2 \leq 1$, will its ℓ_1-norm $\|\boldsymbol{x}\|_1$ be bounded? Justify your answer.

(b) (4%) Generalize (1) by letting p be any positive number. Show that the following limit exists, and give the result.

$$\lim_{p \to 0^+} |x_1|^p + |x_2|^p + \cdots + |x_n|^p.$$

(c) (5%) A vector norm function $f(\boldsymbol{x})$ must satisfy *absolute homogeneity*, that is, for any scalar α and vector $\boldsymbol{x}$, we must have $f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$. Can the result of (b) be a proper vector norm? Justify your answer.

A Answer.

(a) Common Cauchy's inequality,

$$n(x_1^2 + \cdots + x_n^2) \geq (|x_1| + \cdots + |x_n|)^2.$$

Therefore, $\|\boldsymbol{x}\|_1 \leq \sqrt{n}$.

(b) For some j such that $1 \leq j \leq n$, if $x_j \neq 0$, then

$$\lim_{p \to 0^+} |x_j|^p = |x_j|^0 = 1;$$

otherwise $|x_j|^p = 0$. Therefore, the result is the total number of non-zero entries of $\boldsymbol{x}$. (In other words, the answer is $\|\boldsymbol{x}\|_0$).

(c) **No**. That is because $\|\alpha \boldsymbol{x}\|_0 = \|\boldsymbol{x}\|_0$ for any $\alpha \neq 0$.

Q5 (12%)

(a) (3%) A concrete formulation is shown as follows.

$$\min_{\beta_1, \beta_2, \beta_3} \left\| \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \right\|_2^2 + 3(|\beta_1| + |\beta_2| + |\beta_3|) + 8(\beta_1^2 + \beta_2^2 + \beta_3^2).$$

Let $\beta_1' = 3\beta_1$, $\beta_2' = 3\beta_2$, $\beta_3' = 3\beta_3$, please transfer the above formulation as an equivalent formulation with respect to β_1', β_2' and β_3'.

(b) (6%) Considering a more general formulation as follows.

$$\min_{\boldsymbol{\beta}} \quad \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2.$$

Try to explain or prove how this formulation can be converted into an equivalent LASSO problem with λ_1 and λ_2 positive numbers.

(c) (3%) Compare the formulation in (b) and typical LASSO formulation by discussing the advantages and disadvantages.

A (a)

$$\min_{\beta_1',\beta_2',\beta_3'} \left\| \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \\ \sqrt{8} & 0 & 0 \\ 0 & \sqrt{8} & 0 \\ 0 & 0 & \sqrt{8} \end{bmatrix} \begin{bmatrix} \beta_1' \\ \beta_2' \\ \beta_3' \end{bmatrix} \right\|_2^2 + (\beta_1'| + |\beta_2'| + |\beta_3'|)$$

(b) & (c) This kind of method is so-called elastic net. The objective function of the elastic net problem

$$\begin{aligned} L_{\text{enet}} &= \min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right) \\ &= \min_{\boldsymbol{\beta}} \left(\left\| \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix} - \begin{pmatrix} \boldsymbol{X} \\ \sqrt{\lambda_2}\boldsymbol{I}_p \end{pmatrix} \boldsymbol{\beta} \right\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right), \end{aligned}$$

is equivalent to the objective function of the lasso problem with augmented data $\boldsymbol{X}' = (1+\lambda_2)^{-1/2} \left(\boldsymbol{X}^T, \sqrt{\lambda_2}\boldsymbol{I}_p \right)^T$ and $\boldsymbol{y}' = \left(\boldsymbol{y}^T, \boldsymbol{0}^T \right)^T$.

So it can be seen as

$$L_{\text{lasso}} = \min_{\boldsymbol{\beta}'} \left(\|\boldsymbol{y}' - \boldsymbol{X}'\boldsymbol{\beta}'\|_2^2 + \gamma \|\boldsymbol{\beta}'\|_1 \right),$$

where $\gamma = \lambda_1/\sqrt{1+\lambda_2}$ and $\boldsymbol{\beta}' = \sqrt{1+\lambda_2}\boldsymbol{\beta}$. The minimizer of L_{lasso} (i.e., $\hat{\boldsymbol{\beta}}'$) can be transformed back to the minimizer of L_{enet} via $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}'/\sqrt{1+\lambda_2}$. Note that we divide design matrix by $\lambda_1/\sqrt{1+\lambda_2}$ to guarantee that each column of $\boldsymbol{X}'$ has sum of squares equalling to 1.

More details can be referred to the following paper: Zou+, "Regularization and Variable Selection via the Elastic Net", Journal of the Royal Statistical Society, 2005.

The LASSO is often exploited to perform both variable selection and regularization. However, due to some limitations, LASSO is not always suitable for variable selection. For example, if $p > N$, the LASSO selects at most N variables. Even for $N > p$ case, if high correlations are observed among columns in $\boldsymbol{X}$, LASSO tends to select one variable and ignore the others. Therefore, to overcome the disadvantages of LASSO, some researchers proposed a new objective function with adding more regularization on $\boldsymbol{\beta}$.

$$\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right),$$

where λ_1 and λ_2 is non-negative. Note that $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^p |\beta_j|$ and $\|\boldsymbol{\beta}\|_2^2 = \sum_{j=1}^p {\beta_j}^2$.

Q6 (13%)

(a) (3%) Consider the typical LASSO formulation

$$\min_{\beta_1,\beta_2,\beta_3} \left\| \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \right\|_2^2 + 3(|\beta_1| + |\beta_2| + |\beta_3|).$$

Let $\beta_1' = 2\beta_1$, $\beta_2' = 3\beta_2$ and $\beta_3' = 4\beta_3$, please transfer the above formulation as an equivalent formulation with respect to β_1', β_2' and β_3'.

(b) (7%) Consider a formulation as follows.

$$\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{i=1}^{p} w_i \left|\beta_i\right| \right),$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_p]^\top$ and $w_i > 0$. We expect to apply same algorithms for solving LASSO problems to handle above object function. Considering that, try to convert the above formulation into the standard LASSO problem (i.e., $\min_{\boldsymbol{\beta}} (\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1)$) under a general assumption that $w_k \neq w_j$, if $k \neq j$.

(c) (3%) Compared with the typical LASSO formulation, what are advantages for the formulation in (b)?

A (a)

$$\min_{\beta_1', \beta_2', \beta_3'} \left\| \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}^{-1} \begin{bmatrix} \beta_1' \\ \beta_2' \\ \beta_3' \end{bmatrix} \right\|_2^2 + 3 \left(\frac{1}{2} \left|\beta_1'\right| + \frac{1}{3} \left|\beta_2'\right| + \frac{1}{4} \left|\beta_3'\right| \right)$$

(b) & (c) Via LASSO, we can obtain the sparse solution. Thus, the prediction error is reduced. However, this is a double-edged sword. The bias of $\boldsymbol{\beta}$ increase simultaneously. Some researchers came up with a solution that add some weights in the regularization term in LASSO problem. The formulation for LASSO becomes that

$$\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} w_j \left|\beta_j\right| \right), \tag{2}$$

where $w_j > 0$. Assume a diagonal matrix $\boldsymbol{W} = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_p \end{bmatrix}$, and denote $\beta_j^* = w_j \beta_j$, and $\boldsymbol{\beta}^* = \boldsymbol{W}\boldsymbol{\beta}$. Then

$$\begin{aligned} \boldsymbol{y} &\approx \boldsymbol{X}\boldsymbol{\beta} \\ &\approx \boldsymbol{X}\boldsymbol{W}^{-1}\boldsymbol{W}\boldsymbol{\beta} \\ &\approx \boldsymbol{X}^*\boldsymbol{\beta}^*, \end{aligned} \tag{3}$$

where $\boldsymbol{X}^* = \boldsymbol{X}\boldsymbol{W}^{-1}$. Hence, we can obtain

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} w_j \left|\beta_j\right| = \|\boldsymbol{y} - \boldsymbol{X}^*\boldsymbol{\beta}^*\|_2^2 + \lambda \|\boldsymbol{\beta}^*\|_1 . \tag{4}$$

When considering the minimizer of the objective function (2), if $\hat{\boldsymbol{\beta}}^*$ is the optimal solution to (4), then $\hat{\boldsymbol{\beta}} = \boldsymbol{W}^{-1}\hat{\boldsymbol{\beta}}^*$. More details can be referred to the following paper: Zou, "The adaptive lasso and its oracle properties", Journal of the American Statistical Association, 2006.

Q7 (10%) Convolution is the most important operation in CNN. As shown in Figure 2, the input activation tensor is $\mathcal{X} \in \mathbb{R}^{H \times W \times C}$. Weight tensor is $\mathcal{W} \in \mathbb{R}^{R \times S \times C \times K}$. The output activation tensor is $\mathcal{Y} \in \mathbb{R}^{P \times Q \times K}$. Here we set $H = W = 5$, $C = 8$, $R = S = 3$, $K = 6$ and $P = Q = 3$. Besides, the stride number is 1 and the padding number is 0.

(a) (2%) Write down direct convolution by C++ language style.

(b) (4%) The loop unrolling is one of loop optimization techniques to make full use of the hardware on-chip storage resources. Write down the loop unrolling at input channel level and output channel level, respectively, by C++ language style.

(c) (4%) Sketch the corresponding computing hardware architectures for the two loop unrolling strategies in (b), respectively.

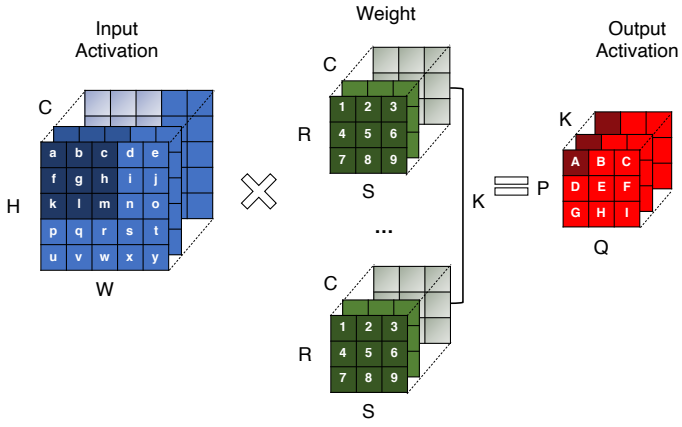

Figure 2: Convolution.

A (a) answer is shown in Figure 3: (b) answer is shown in Figure 4 and Figure 5:

```
for (int co = 0; co < 6; co++)
    for (int ci = 0; ci < 8; ci++)
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                for (int ki = 0; ki < 3; ki++)
                    for (int kj = 0; kj < 3; kj++)
                        Y[i][j][co] += W[ki][kj][ci][co] * X[3 * i + ki][3 * j + kj][ci];
```

Figure 3: direct convolution

```
for (int co = 0; co < 6; co++)
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            for (int ki = 0; ki < 3; ki++)
                for (int kj = 0; kj < 3; kj++)
                    Y[i][j][co] += W[ki][kj][0][co] * X[3 * i + ki][3 * j + kj][0];
                    Y[i][j][co] += W[ki][kj][1][co] * X[3 * i + ki][3 * j + kj][1];
                    Y[i][j][co] += W[ki][kj][2][co] * X[3 * i + ki][3 * j + kj][2];
                    Y[i][j][co] += W[ki][kj][3][co] * X[3 * i + ki][3 * j + kj][3];
                    Y[i][j][co] += W[ki][kj][4][co] * X[3 * i + ki][3 * j + kj][4];
                    Y[i][j][co] += W[ki][kj][5][co] * X[3 * i + ki][3 * j + kj][5];
                    Y[i][j][co] += W[ki][kj][6][co] * X[3 * i + ki][3 * j + kj][6];
                    Y[i][j][co] += W[ki][kj][7][co] * X[3 * i + ki][3 * j + kj][7];
```

Figure 4: unroll input channel level

(c) answer is shown in Figure 6 and Figure 7:

```
for (int ci = 0; ci < 8; ci++)
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            for (int ki = 0; ki < 3; ki++)
                for (int kj = 0; kj < 3; kj++)
                    Y[i][j][0] += W[ki][kj][ci][0] * X[3 * i + ki][3 * j + kj][ci];
                    Y[i][j][1] += W[ki][kj][ci][1] * X[3 * i + ki][3 * j + kj][ci];
                    Y[i][j][2] += W[ki][kj][ci][2] * X[3 * i + ki][3 * j + kj][ci];
                    Y[i][j][3] += W[ki][kj][ci][3] * X[3 * i + ki][3 * j + kj][ci];
                    Y[i][j][4] += W[ki][kj][ci][4] * X[3 * i + ki][3 * j + kj][ci];
                    Y[i][j][5] += W[ki][kj][ci][5] * X[3 * i + ki][3 * j + kj][ci];
```

Figure 5: unroll output channel level

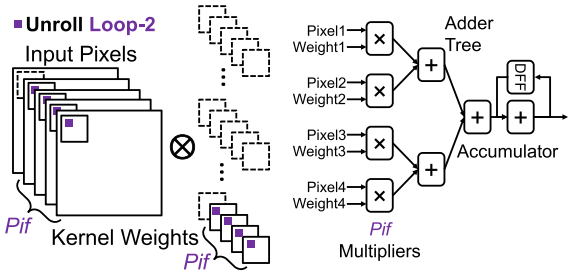

Figure 6: computing hardware architectures for unroll input channel level

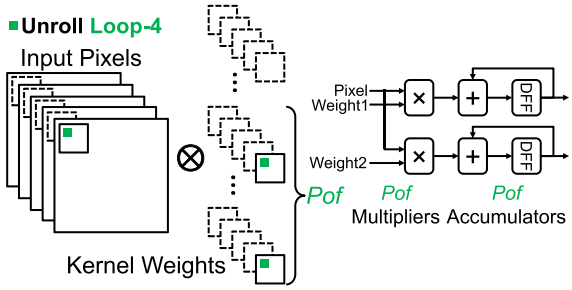

Figure 7: computing hardware architectures for unroll output channel level

Q8 (15%) Convolution can be equivalently represented as matrix matrix multiplication. Here we consider a special case: $\boldsymbol{Y} = \boldsymbol{X} \cdot \boldsymbol{W} + \boldsymbol{V}$, where $\boldsymbol{Y} \in \mathbb{R}^{N \times K}$ and $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ are known input and output matrices. $\boldsymbol{V} \in \mathbb{R}^{N \times K}$ is an unknown model error matrix. $\boldsymbol{W} \in \mathbb{R}^{M \times K}$ is an unknown model coefficient matrix. In particular, indexes of nonzero elements of each row in $\boldsymbol{W}$ is identical to achieve structure sparsity. Let

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix}, \boldsymbol{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}.$$

(a) (7%) Write down the coordinate descent method to handle the formulation as follows.

$$\min_{\boldsymbol{W}} \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix} \cdot \boldsymbol{W} \right\|_2^2 + \sum_{i=1}^{3} \lambda_i \left\| \boldsymbol{w}_{i,\cdot} \right\|_2,$$

where $\boldsymbol{w}_{i,\cdot}$ denotes the i-th row in $\boldsymbol{W}$. $\lambda_1 = 1$, $\lambda_2 = 100$ and $\lambda_3 = 1$. The initial matrix $\boldsymbol{W}^{(0)} = \boldsymbol{O}$. The stopping criterion is set to 2 iterations. Please show the final numerical result.

(b) (8%) Obtaining this structure sparse model coefficient matrix can be formulated as

$$\min_{\boldsymbol{W}} \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix} \cdot \boldsymbol{W} \right\|_2^2$$
$$\text{s.t.} \quad \sum_{i=1}^{3} \mathcal{I}\left[\left\| \boldsymbol{w}_{i,\cdot} \right\| > 0 \right] \leq 2,$$

where $\mathcal{I}[\cdot]$ denotes the indicator function and $\|\cdot\|$ is an any vector norm. In fact, $\sum_{i=1}^{3} \mathcal{I}\left[\left\| \boldsymbol{w}_{i,\cdot} \right\| > 0 \right]$ denotes the number of nonzero rows in the matrix $\boldsymbol{W}$. In the constraint, 2 is given to determine the number of nonzero rows in the matrix $\boldsymbol{W}$. Orthogonal matching pursuit, as a heuristics method, is widely used in one-dimension sparse vector reconstruction. Please extend the typical orthogonal matching pursuit to handle the formulation and show the final numerical result.

A 4(a) answer is shown in Figure 8. The answer is not unique since it relies on the variable ordering and multivalued solution.

In first iteration, $W = \begin{bmatrix} w_{11} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, we fix W except w_{11}, w_{11} is the only variable.

$$L = \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix} \cdot W_2^2 \right\| + \sum_{i=1}^{3} \lambda_i \| \boldsymbol{w}_i, \cdot \|_2$$

$$x_1^{(k)} = \arg\min_{x_1} f(x_1, x_2^{(k-1)}, x_3^{(k-1)}, \dots x_n^{(k-1)})$$
$$x_2^{(k)} = \arg\min_{x_2} f(x_1^{(k)}, x_2, x_3^{(k-1)}, \dots x_n^{(k-1)})$$
$$x_3^{(k)} = \arg\min_{x_3} f(x_1^{(k)}, x_2^{(k)}, x_3, \dots x_n^{(k-1)})$$
$$\dots$$
$$x_n^{(k)} = \arg\min_{x_n} f(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \dots x_n)$$

Figure 8: Coordinate descent

$$L = \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix} \cdot \begin{bmatrix} w_{11} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right\|_2^2 + \lambda_1 \sqrt{(|w_{11}|^2 + |w_{12}|^2)} + \lambda_2 \sqrt{(|w_{21}|^2 + |w_{22}|^2)}$$
$$+ \lambda_3 \sqrt{(|w_{31}|^2 + |w_{32}|^2)}$$

$$L = \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9w_{11} & 0 \\ 10w_{11} & 0 \\ 11w_{11} & 0 \\ 12w_{11} & 0 \end{bmatrix} \right\|_2^2 + (1)\sqrt{(|w_{11}|^2 + |0|^2)} + (100)\sqrt{(|0|^2 + |0|^2)} +$$
$$(1)\sqrt{(|0|^2 + |0|^2)}$$

$$L = \left\| \begin{bmatrix} 1 - 9w_{11} & 2 \\ 3 - 10w_{11} & 4 \\ 5 - 11w_{11} & 6 \\ 7 - 12w_{11} & 8 \end{bmatrix} \right\|_2^2 + (1)\sqrt{(|w_{11}|^2 + |0|^2)}$$

$$L = \left\| \begin{bmatrix} 1 - 9w_{11} & 2 \\ 3 - 10w_{11} & 4 \\ 5 - 11w_{11} & 6 \\ 7 - 12w_{11} & 8 \end{bmatrix} \right\|_2^2 + w_{11}$$
$$L = (1 - 9w_{11})^2 + 2^2 + (3 - 10w_{11})^2 + 4^2 + (5 - 11w_{11})^2 + 6^2 + (7 - 12w_{11})^2 + 8^2 + w_{11}$$
$$L = 204 - 355w_{11} + 446w_{11}^2$$

We are finding the w_{11}, subject to $\frac{dL}{dw_{11}} = 0$. Then $w_{11} = 0.389$.
In second iteration,

$$W = \begin{bmatrix} w_{11}^{(1)} & w_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.389 & w_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

we fix $\boldsymbol{W}$, except w_{12}
w_{12} is the only variable.

$$L = \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix} \cdot W_2^2 \right\| + \sum_{i=1}^{3} \lambda_i \|w_i, \cdot\|_2$$

$$L = \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} 9 & 13 & 17 \\ 10 & 14 & 18 \\ 11 & 15 & 19 \\ 12 & 16 & 20 \end{bmatrix} \cdot \begin{bmatrix} 0.389 & w_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right\|^2 + \lambda_1 \sqrt{\left(|0.389|^2 + |w_{12}|^2\right)} + \lambda_2 \sqrt{\left(|w_{21}|^2 + |w_{22}|^2\right)} + \lambda_3 \sqrt{\left(|w_{31}|^2 + |w_{32}|^2\right)}$$

$$L = \left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} - \begin{bmatrix} -0.389 * 9 & 9w_{12} \\ -0.389 * 10 & 10w_{12} \\ -0.389 * 11 & 11w_{12} \\ -0.389 * 12 & 12w_{12} \end{bmatrix} \right\|_2^2 + (1) \sqrt{\left(|-0.389|^2 + |w_{12}|^2\right)} + (100) \sqrt{\left(|0|^2 + |0|^2\right)} + (1) \sqrt{\left(|0|^2 + |0|^2\right)}$$

$$L = \left\| \begin{bmatrix} -2.501 & 2 - 9w_{12} \\ -0.89 & 4 - 10w_{12} \\ 0.721 & 6 - 11w_{12} \\ 2.332 & 8 - 12w_{12} \end{bmatrix} \right\|_2^2 + (1) \sqrt{\left(|0.389|^2 + |w_{12}|^2\right)}$$

$$L = (-2.501)^2 + (2 - 9w_{12})^2 + (-0.89)^2 + (4 - 10w_{12})^2 + 0.721^2 + (6 - 11w_{12})^2 + 2.332^2 + (8 - 12w_{12})^2 + \sqrt{(0.389^2 + w_{12}^2)}$$

We are finding x_{12}, subject to $\frac{dL}{dw_{12}} = 0$. Then $w_{12} = 0.2942$. The final result is

$$\boldsymbol{W} = \begin{bmatrix} 0.389 & 0.2942 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

4(b) answer is shown in Figure 9. The final result is

ALGORITHM 1:
Simulatenous orthogonal matching pursuit (SOMP)

Require: $\boldsymbol{Y} \in \mathbb{R}^{m \times K}$, $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$, $s \geq 1$
1: Initialization: $\boldsymbol{R}^{(0)} \leftarrow \boldsymbol{Y}$ and $S_0 \leftarrow \emptyset$
2: $t \leftarrow 0$
3: **while** $t < s$ **do**
4: Determine the atom of $\boldsymbol{\Phi}$ to be included in the support:
$j_t \leftarrow \text{argmax}_j(\|(\boldsymbol{R}^{(t)})^{\mathrm{T}} \boldsymbol{\phi}_j\|_1)$
5: Update the support : $S_{t+1} \leftarrow S_t \cup \{j_t\}$
6: Projection of each measurement vector onto span($\boldsymbol{\Phi}_{S_{t+1}}$):
$\boldsymbol{Y}^{(t+1)} \leftarrow \boldsymbol{\Phi}_{S_{t+1}} \boldsymbol{\Phi}^{+}_{S_{t+1}} \boldsymbol{Y}$
7: Projection of each measurement vector onto span($\boldsymbol{\Phi}_{S_{t+1}}$)$^{\perp}$:
$\boldsymbol{R}^{(t+1)} \leftarrow \boldsymbol{Y} - \boldsymbol{Y}^{(t+1)}$
8: $t \leftarrow t + 1$
9: **end while**
10: **return** S_s {Support at last step}

Figure 9: SOMP

$$\boldsymbol{W} = \begin{bmatrix} 4.125 & 4 \\ 0 & 0 \\ -2.125 & -2 \end{bmatrix}$$

Tropp+, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit", JSP'06.