Exact Incremental and Decremental Learning for LS-SVM

Wei-Han Lee[1], Bong Jun Ko[1], Shiqiang Wang[1], Changchang Liu[1], Kin K. Leung[2]

[1] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
[2] Imperial College London, UK

A day in the office of a machine learning (ML) engineer

The ML Engineer

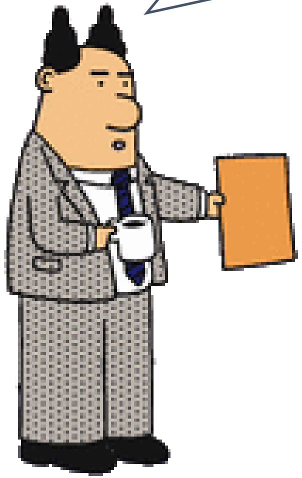
A day in the office of a machine learning (ML) engineer

The Manager

The ML Engineer

A day in the office of a machine learning (ML) engineer

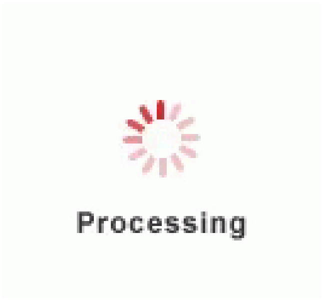
Here's the dataset you've been waiting for.



The Manager

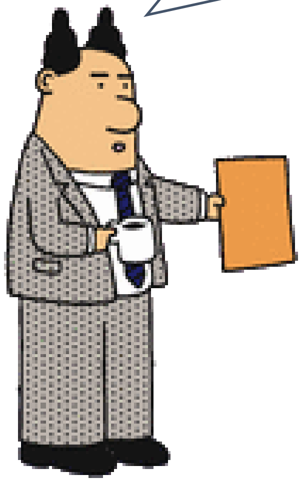
Great! Now I can build a model.

The ML Engineer



A day in the office of a machine learning (ML) engineer

Here’s the dataset you’ve been waiting for.



The Manager

Done!

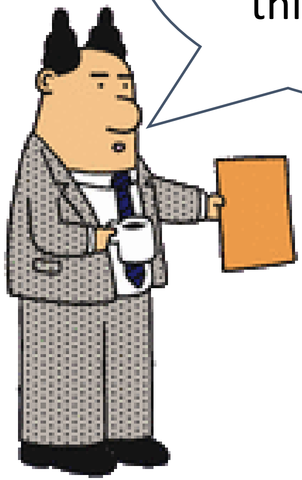
The ML Engineer

The next day

The ML Engineer

The next day

Here's another dataset we just acquired. We need this reflected in the model too.

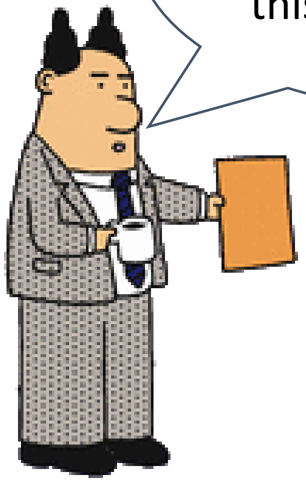


The Manager

The ML Engineer

The next day

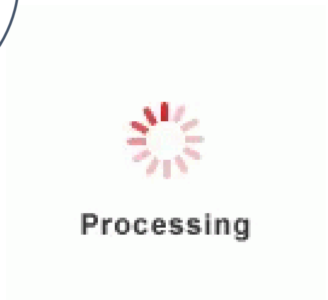
Here's another dataset we just acquired. We need this reflected in the model too.



The Manager

Ok, no problem. I'll just need to add it to the database and run the training job again, so the model doesn't forget the old dataset.

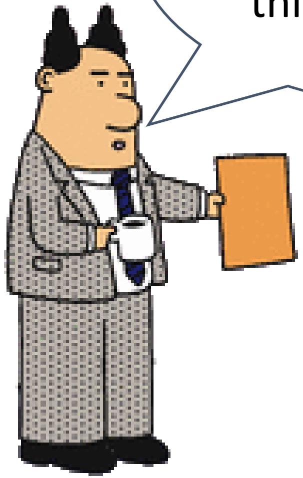
The ML Engineer



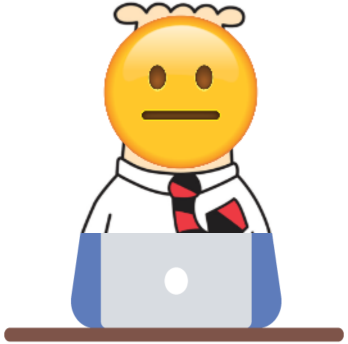
The next day

Here's another dataset we just acquired. We need this reflected in the model too.

Done! It took a little longer, but not a big deal...



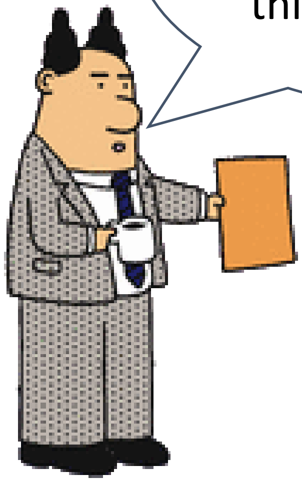
The Manager



The ML Engineer

And so it goes on…

Here’s **yet another** dataset we just acquired. We need this reflected in the model too.

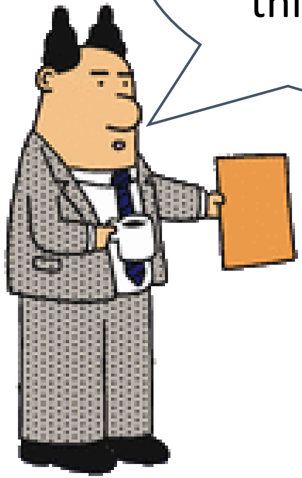


The Manager

The ML Engineer

And so it goes on…

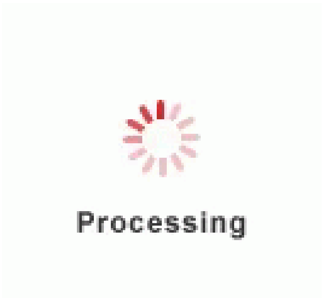
Here’s **yet another** dataset we just acquired. We need this reflected in the model too.



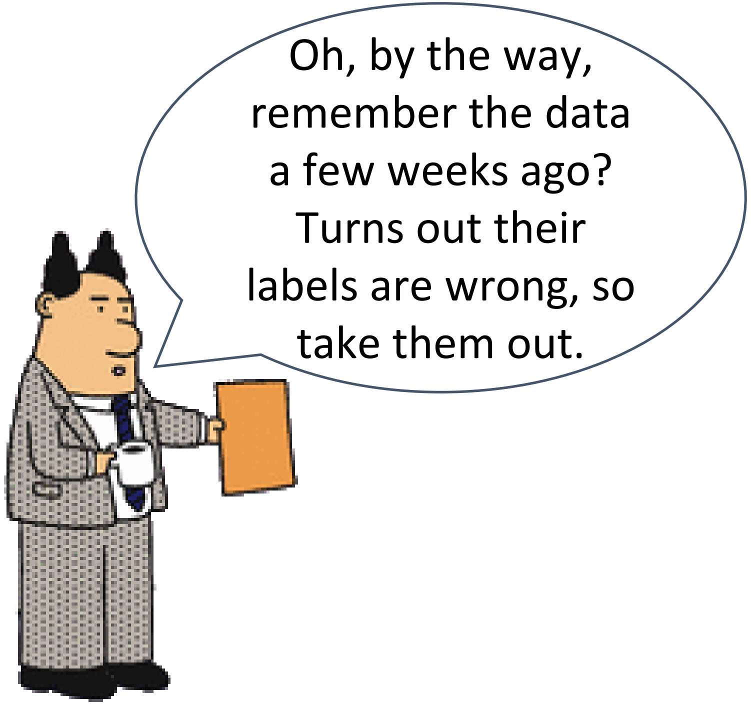
The Manager

Aarggggh! Not again…

The ML Engineer



And yet, even worse…

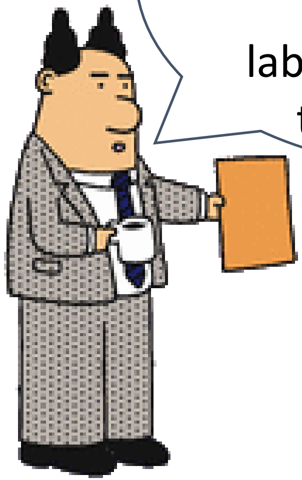


The Manager

The ML Engineer

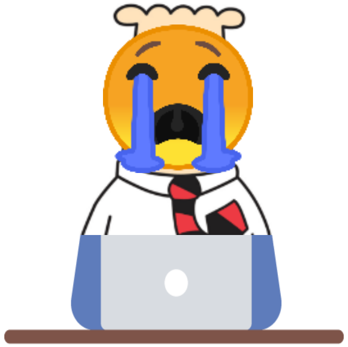
And yet, even worse…

Oh, by the way,
remember the data
a few weeks ago?
Turns out their
labels are wrong, so
take them out.

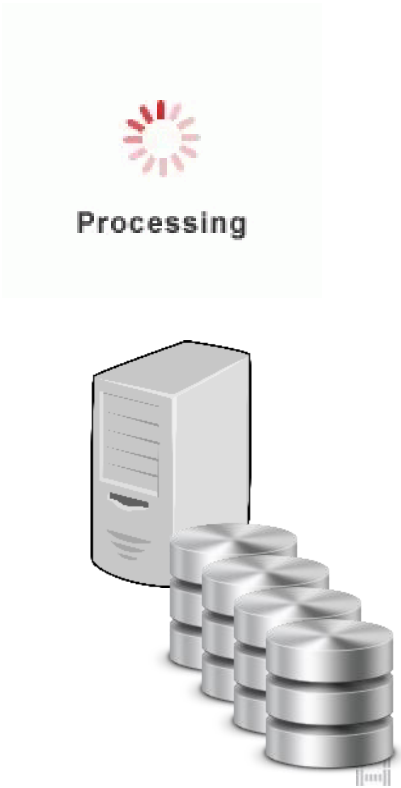


The Manager

What???
Nooooooooooooo!!



The ML Engineer



How to update the model after adding, correcting, or removing data, without having to retrain the entire model?

- Add data
  - Incremental learning
- Remove data
  - Decremental learning
  - Why? – e.g., erroneous data, per user request (GDPR regulation)
- Correction to data
  - Both incremental and decremental learning

State of the Art

- Incremental learning has been considered in various contexts
  - Heuristics for deep neural networks to avoid catastrophic forgetting
  - Exact update methods for shallow models such as support vector machine (SVM)
- A few existing approaches on decremental learning
  - Not for neural networks
  - For SVM but requires old data
- We propose an *exact* (provably optimal) method for *joint* incremental and decremental learning for least squares SVM (LS-SVM) that does *not* require old data

State of the Art (Details)

	Model	Exact?	Requires old data?	Decremental learning?
[1], [2], [3]	SVM	No	Yes	No
[4]	SVM	Yes	Yes	No
[5]	SVM	Yes	Yes	Yes
[6], [7]	SVM	No	No	No
[8], [9]	LS-SVM	Yes	Yes	No
[10], [11]	LS-SVM	No	No	No
Our work	**LS-SVM**	**Yes**	**No**	**Yes**

[1] Syed, N., Liu, H., and Sung, K. Incremental learning with support vector machines. In International Joint Conference on Artificial Intelligence, 1999.

[2] Ruping, S. Incremental learning with support vector machines. In IEEE International Conference on Data Mining, 2001.

[3] Carlotta Domeniconi and Dimitrios Gunopulos. Incremental support vector machine construction, in IEEE International Conference on Data Mining, 2001.

[4] Diehl, C. P. and Cauwenberghs, G. SVM incremental learning, adaptation and optimization. In International Joint Conference on Neural Networks, 2003

[5] Cauwenberghs, G. and Poggio, T. Incremental and decremental support vector machine learning. In Advances in neural information processing systems, 2001.

[6] Yi-Min Wen and Bao-Liang Lu. Incremental learning of support vector machines by classifier combining, in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2007.

[7] Youlu Xing, Furao Shen, Chaomin Luo, and Jinxi Zhao. L3-SVM: a lifelong learning method for SVM. In IEEE IJCNN, 2015

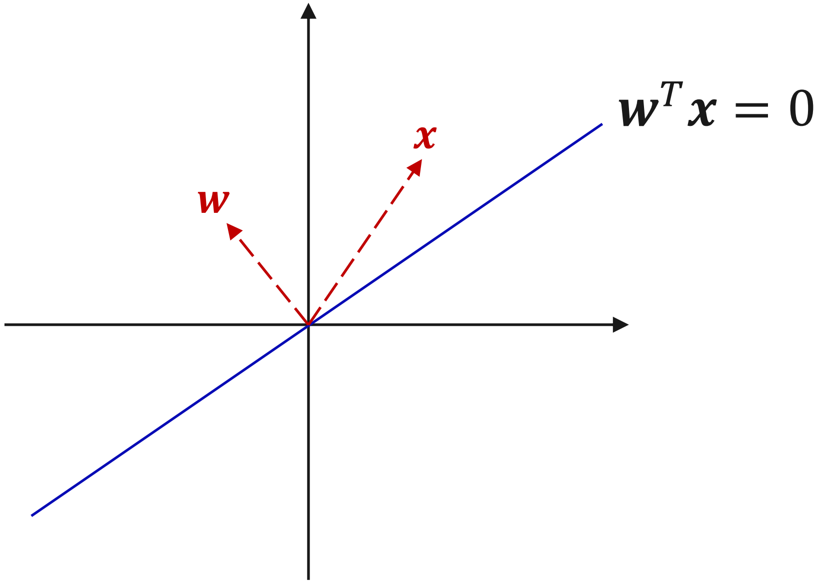
[8] Hoi-Ming Chi and Okan K Ersoy. Recursive update algorithm for least squares support vector machines. Neural Processing Letters,2003

[9] Zhifeng Hao, Shu Yu, Xiaowei Yang, Feng Zhao, Rong Hu, and Yanchun Liang. Online LS-SVM learning for classification problems based on incremental chunk. In International Symposium on Neural Networks, 2004.

[10] Yaakov Engel, Shie Mannor, and Ron Meir. The kernel recursive least squares algorithm. IEEE Trans. On Signal Processing, 2004.

[11] Ling Jian, Shuqian Shen, Jundong Li, Xijun Liang, andLei Li. Budget online learning algorithm for least squares SVM. IEEE Trans. on Neural Networks and Learning Systems, 2017.

Support Vector Machine (SVM)



- If $\boldsymbol{w}^T \boldsymbol{x} > 0$, classify as label $+1$
- If $\boldsymbol{w}^T \boldsymbol{x} < 0$, classify as label -1

- Binary classifier
- Can be converted into multi-class classifier using one-versus-all or one-versus-one ensemble approaches

Least-Squares Support Vector Machine (LS-SVM)

- Optimal solution for model parameter

Regularization constant

Data sample transformed into feature space

Ground-truth label

$$\boldsymbol{w}^* = \mathrm{argmin}_{\boldsymbol{w}\in\mathbb{R}^J} \underbrace{\rho\|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} (\boldsymbol{w}^T \vec{\phi}(\boldsymbol{x}_n) - y_n)^2}_{\text{Learning loss function}}$$

$$= \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{y} = [\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T]^{-1}\boldsymbol{\Phi}\boldsymbol{y}$$

(Analytical optimal solution)

Main Result for Incremental/Decremental Learning

Theorem 1. *For a given model*

$$\boldsymbol{w} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{y} \tag{3}$$

and auxiliary matrix

$$\boldsymbol{C} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{\Phi}^T, \tag{4}$$

when adding new training data $(\boldsymbol{X}_a, \boldsymbol{y}_a)$ *and removing existing training data* $(\boldsymbol{X}_r, \boldsymbol{y}_r)$, *we can compute the new values of* $\boldsymbol{w}$ *and* $\boldsymbol{C}$ *using*

Model updating equations, only involving added/removed data samples and auxiliary matrix $\boldsymbol{C}$

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1}\left(\boldsymbol{\Phi}_c'^T\boldsymbol{w} - \boldsymbol{y}_c\right) \tag{5}$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1}\boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J) \tag{6}$$

where we define $\boldsymbol{\Phi}_c = (\boldsymbol{\Phi}_a, \boldsymbol{\Phi}_r)$, $\boldsymbol{\Phi}_c' = (\boldsymbol{\Phi}_a, -\boldsymbol{\Phi}_r)$, *and* $\boldsymbol{y}_c = (\boldsymbol{y}_a, -\boldsymbol{y}_r)$.

Algorithm

- Initial model training
  - Compute $\boldsymbol{w} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho \boldsymbol{I}_N]^{-1}\boldsymbol{y}$ and $\boldsymbol{C} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho \boldsymbol{I}_N]^{-1}\boldsymbol{\Phi}^T$
- Incremental/decremental learning using new/removed data
  - Compute

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c \left(\rho \boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1} \left(\boldsymbol{\Phi}_c'^T \boldsymbol{w} - \boldsymbol{y}_c\right)$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c \left(\rho \boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1} \boldsymbol{\Phi}_c'^T (\boldsymbol{C} - \boldsymbol{I}_J)$$

- Complexity: $O(L^3 + JL^2 + J^2L + J^3)$

Number of added/removed data samples in a “batch”

Dimension of feature vector

Algorithm

- Initial model training
  - Compute $\boldsymbol{w} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{y}$ and $\boldsymbol{C} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{\Phi}^T$
- Incremental/decremental learning using new/removed data
  - Compute

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1}\left(\boldsymbol{\Phi}_c'^T\boldsymbol{w} - \boldsymbol{y}_c\right)$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1}\boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)$$

- Complexity: $O(L^3 + JL^2 + J^2L + J^3)$

Number of added/removed data samples in a “batch”

Dimension of feature vector

Multiple data sample updates can be done either in one or multiple batches

- When using one sample per batch
  - Complexity is **linear** in the number updated samples
  - May not be best in practice though due to efficient implementations of matrix multiplication (details in experiments)

Special Cases

- One data sample in a batch, either incremental or decremental

Corollary 1 (Incremental learning of a single data sample). *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to include the influence of a new training data sample $(\boldsymbol{x}_{N+1}, y_{N+1})$ using*

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x}_{N+1})(\vec{\phi}(\boldsymbol{x}_{N+1})^T\boldsymbol{w} - y_{N+1})}{\vec{\phi}(\boldsymbol{x}_{N+1})^T\vec{\phi}(\boldsymbol{x}_{N+1}) + \rho - \vec{\phi}(\boldsymbol{x}_{N+1})^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x}_{N+1})}$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x}_{N+1})\vec{\phi}(\boldsymbol{x}_{N+1})^T(\boldsymbol{C} - \boldsymbol{I}_J)}{\vec{\phi}(\boldsymbol{x}_{N+1})^T\vec{\phi}(\boldsymbol{x}_{N+1}) + \rho - \vec{\phi}(\boldsymbol{x}_{N+1})^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x}_{N+1})}.$$

Corollary 2 (Decremental learning of a single data sample). *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to remove the influence of an existing training data sample $(\boldsymbol{x}_r, y_r)$ using*

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} - \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x}_r)\left(\vec{\phi}(\boldsymbol{x}_r)^T\boldsymbol{w} - y_r\right)}{-\vec{\phi}(\boldsymbol{x}_r)^T\vec{\phi}(\boldsymbol{x}_r) + \rho + \vec{\phi}(\boldsymbol{x}_r)^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x}_r)}$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} - \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x}_r)\vec{\phi}(\boldsymbol{x}_r)^T(\boldsymbol{C} - \boldsymbol{I}_J)}{-\vec{\phi}(\boldsymbol{x}_r)^T\vec{\phi}(\boldsymbol{x}_r) + \rho + \vec{\phi}(\boldsymbol{x}_r)^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x}_r)}.$$

Special Cases

- Multiple data samples in a batch, either incremental or decremental

Corollary 3 (Incremental learning of a batch of data samples)**.** *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to include the influence of a batch of new training data samples $(\boldsymbol{X}_a, \boldsymbol{y}_a)$ using*

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a \left(\rho \boldsymbol{I} - \boldsymbol{\Phi}_a^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a\right)^{-1} \left(\boldsymbol{\Phi}_a^T \boldsymbol{w} - \boldsymbol{y}_a\right)$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a \left(\rho \boldsymbol{I} - \boldsymbol{\Phi}_a^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a\right)^{-1} \boldsymbol{\Phi}_a^T (\boldsymbol{C} - \boldsymbol{I}_J)$$

where $\boldsymbol{\Phi}_a = \boldsymbol{\Phi}(\boldsymbol{X}_a)$.

Corollary 4 (Decremental learning of a batch of data samples)**.** *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to remove the influence of a batch of existing training data samples $(\boldsymbol{X}_r, \boldsymbol{y}_r)$ using*

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} - (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r \left(\rho \boldsymbol{I} + \boldsymbol{\Phi}_r^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r\right)^{-1} \left(\boldsymbol{\Phi}_r^T \boldsymbol{w} - \boldsymbol{y}_r\right)$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} - (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r \left(\rho \boldsymbol{I} + \boldsymbol{\Phi}_r^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r\right)^{-1} \boldsymbol{\Phi}_r^T (\boldsymbol{C} - \boldsymbol{I}_J)$$

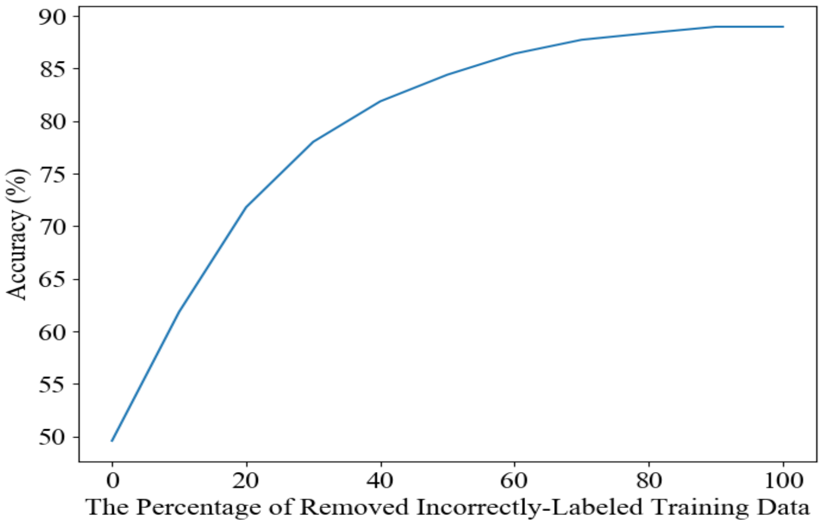
where $\boldsymbol{\Phi}_r = \boldsymbol{\Phi}(\boldsymbol{X}_r)$.

Experiments

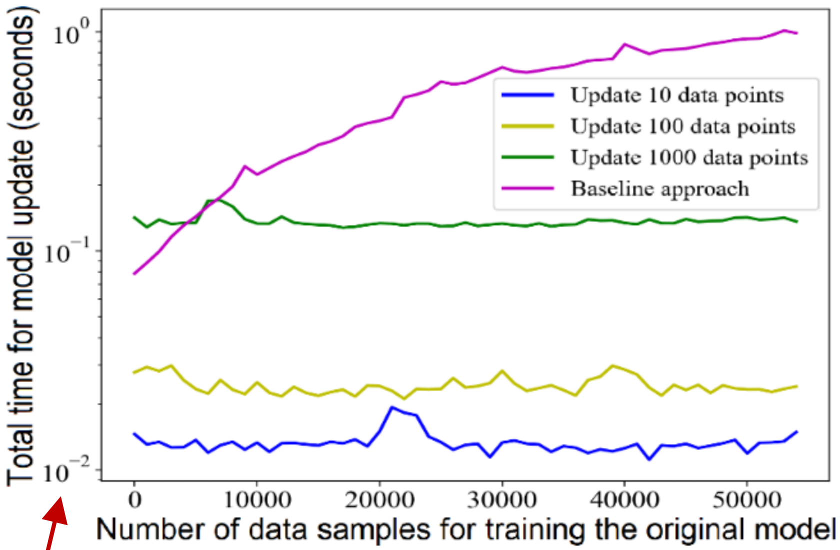
- MNIST dataset of handwritten digits
- Classify even/odd digits using LS-SVM
- 60,000 training data samples
- 10,000 test data samples

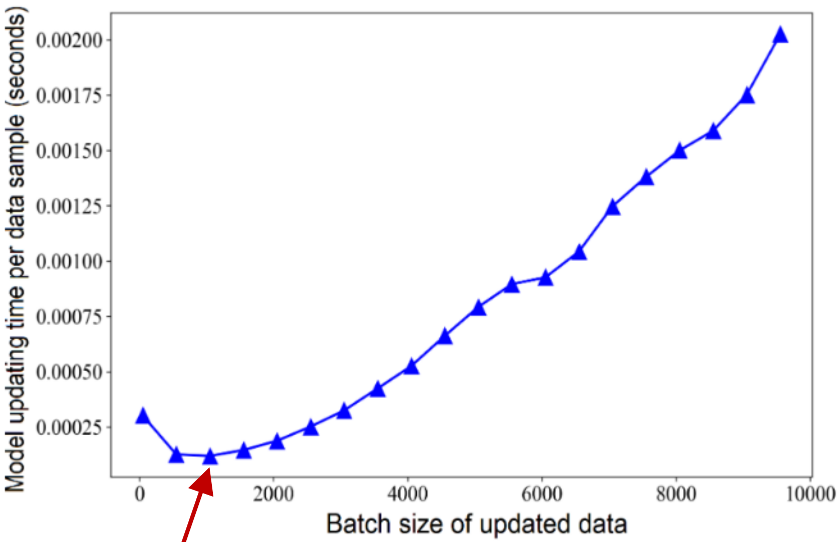
Training with Incorrectly Labeled Data

- Initially, 50% of data samples are mislabeled
- Removing 10% of mislabeled data increases accuracy by 12%
  - Retraining the entire model takes 0.81 seconds (on a personal laptop)
  - Updating using our proposed approach takes 0.26 seconds



Update Time with Different Batch Sizes



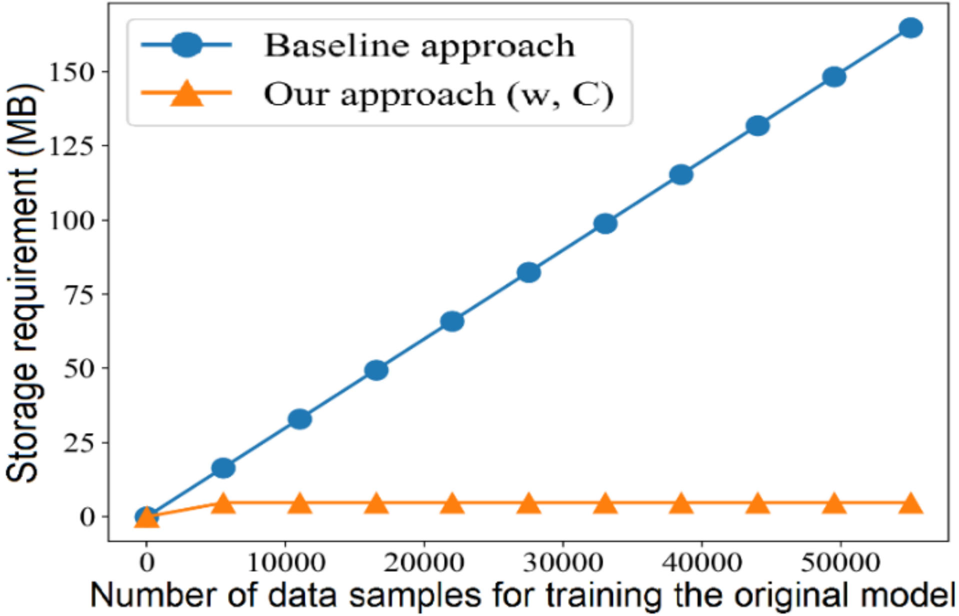


Log-scale

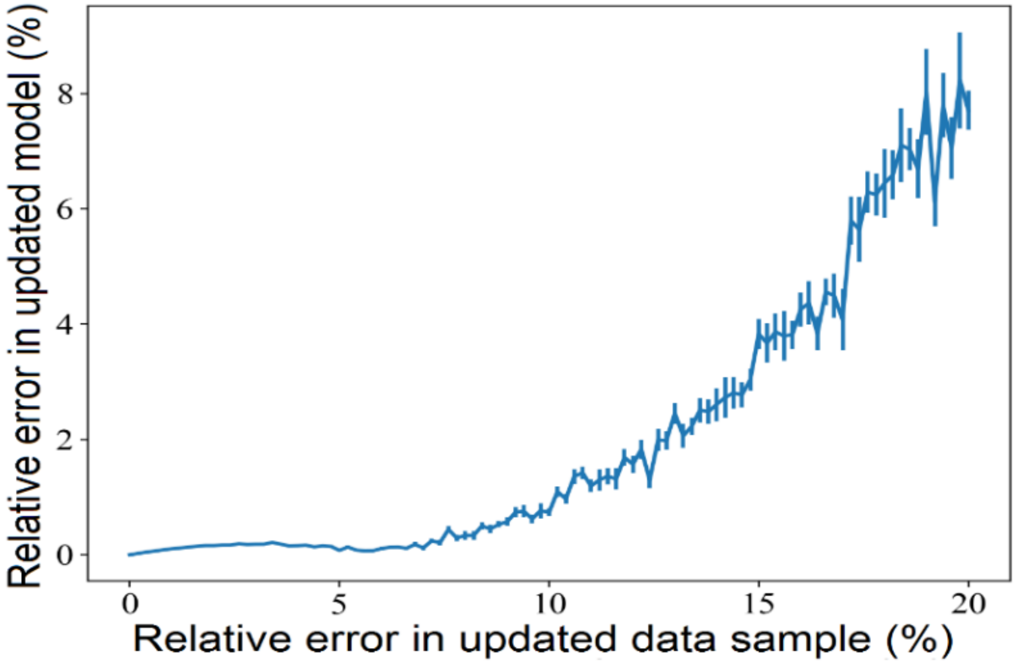
Optimal batch size from experimental (empirical) evaluation

Storage Saving

- We do not need to store original data



When we don’t exactly know the data samples to be removed…



(Removing one data sample)

Summary

- What we have done…
  - Provably optimal incremental and decremental learning for LS-SVM
  - Does not require old data
  - Only requires an auxiliary matrix and data samples that are added/removed
  - Reduces model update time and storage, compared to retraining
  - Preserves privacy of training data when sharing updatable models

- What remains to be done…
  - Incremental and decremental learning (particularly decremental learning) for generic models such as deep neural networks
    - How to properly define decremental learning?
    - Algorithms for joint incremental and decremental learning
    - Anything provable?
    - Any intuitive insights?

Thank you

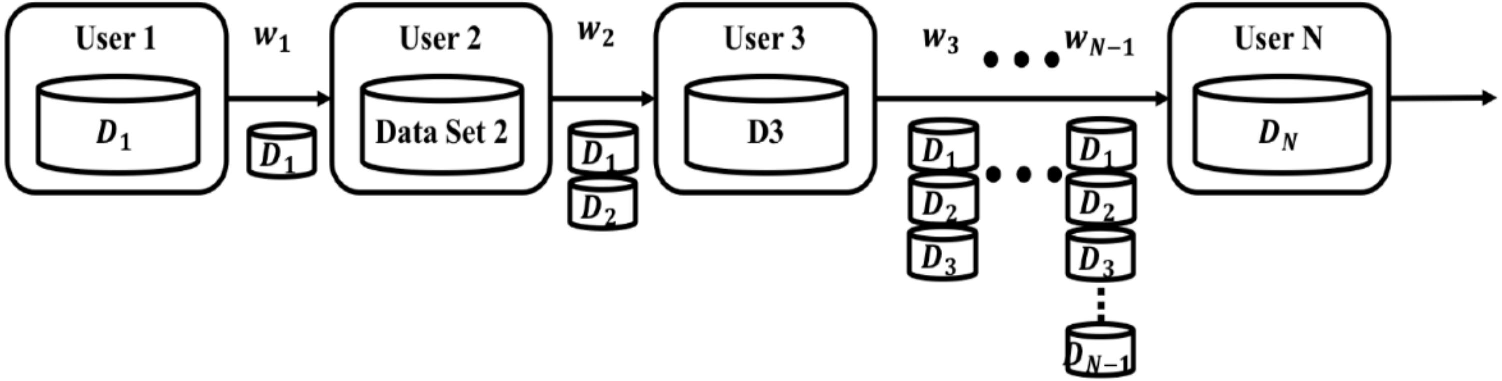
Q & A

Backup slides

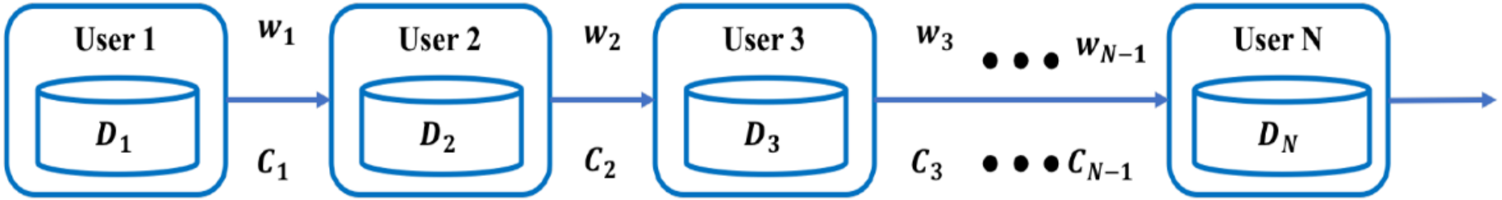
Application

- Decentralized learning

Data Sharing

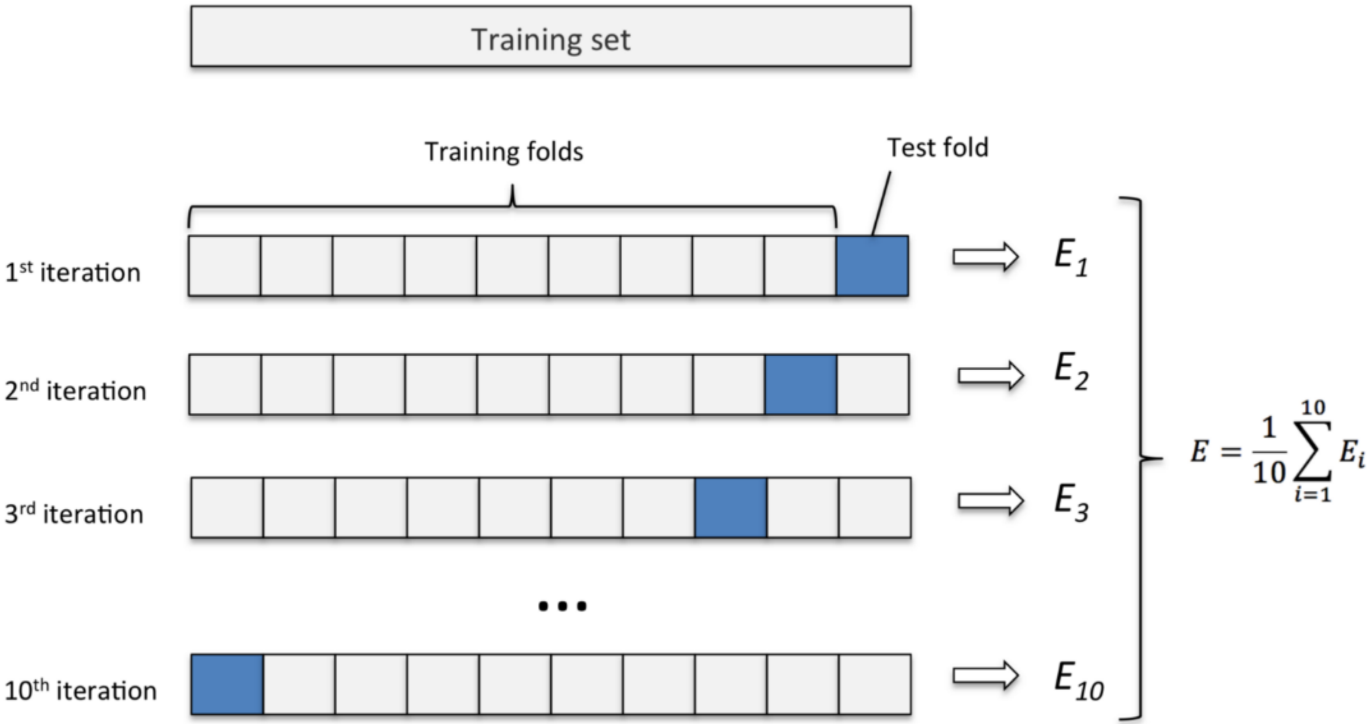


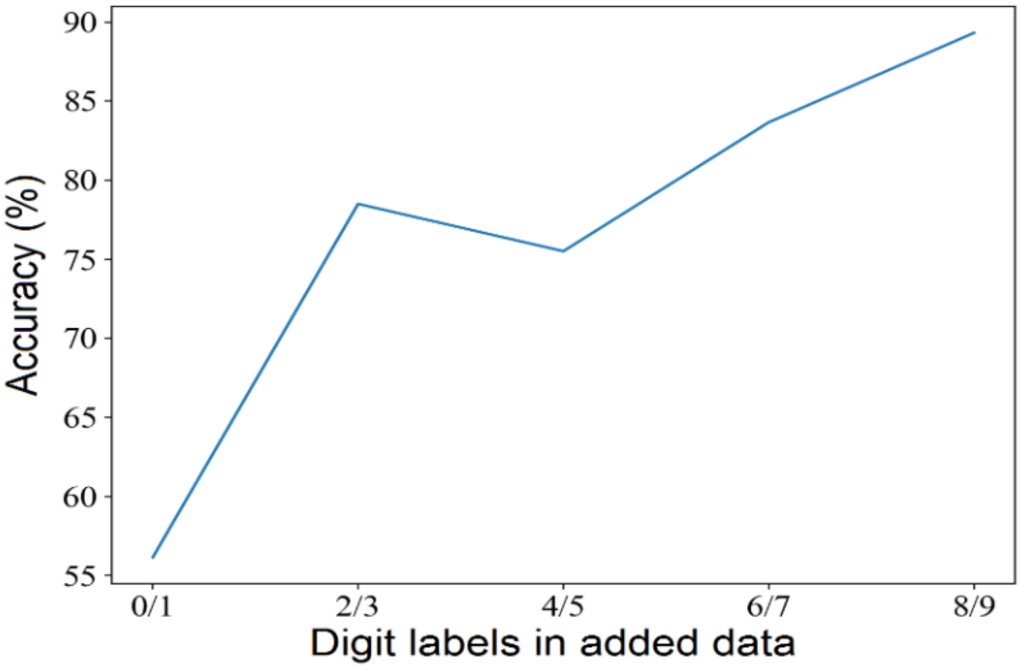
Our Method



Application

- k-fold cross validation (incrementally replace one fold)




Accuracy (%)
90
85
80
75
70
65
60
55
0/1
2/3
4/5
6/7
8/9
Digit labels in added data

Proof. For the updated training dataset, we get $\boldsymbol{\Phi}_{\text{new}} = (\boldsymbol{\Phi}, \boldsymbol{\Phi}_a | \boldsymbol{\Phi}_r)$ and $\boldsymbol{y}_{\text{new}} = (\boldsymbol{y}, \boldsymbol{y}_a | \boldsymbol{y}_r)$, where $(\boldsymbol{Z}_1 | \boldsymbol{Z}_2)$ denotes removing the columns (of a matrix) or elements (of a vector) in $\boldsymbol{Z}_2$ from $\boldsymbol{Z}_1$. Using (2) and (3), we can compute the new model $\boldsymbol{w}_{\text{new}}$ as

$$\begin{aligned}
\boldsymbol{w}_{\text{new}} &= \boldsymbol{\Phi}_{\text{new}}[\boldsymbol{\Phi}_{\text{new}}^T \boldsymbol{\Phi}_{\text{new}} + \rho \boldsymbol{I}_{N+N_a-N_r}]^{-1} \boldsymbol{y}_{\text{new}} \\
&= [\rho \boldsymbol{I}_J + \boldsymbol{\Phi}_{\text{new}} \boldsymbol{\Phi}_{\text{new}}^T]^{-1} \boldsymbol{\Phi}_{\text{new}} \boldsymbol{y}_{\text{new}} \\
&= \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_a \boldsymbol{\Phi}_a^T - \boldsymbol{\Phi}_r \boldsymbol{\Phi}_r^T\right]^{-1} [\boldsymbol{\Phi}\boldsymbol{y} + \boldsymbol{\Phi}_a \boldsymbol{y}_a - \boldsymbol{\Phi}_r \boldsymbol{y}_r] \\
&= \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_c \boldsymbol{\Phi}_c'^T\right]^{-1} [\boldsymbol{\Phi}\boldsymbol{y} + \boldsymbol{\Phi}_c \boldsymbol{y}_c] \qquad (7)
\end{aligned}$$

where we note that $\boldsymbol{\Phi}_c = (\boldsymbol{\Phi}_a, \boldsymbol{\Phi}_r)$, $\boldsymbol{\Phi}_c' = (\boldsymbol{\Phi}_a, -\boldsymbol{\Phi}_r)$ and $\boldsymbol{y}_c = (\boldsymbol{y}_a, -\boldsymbol{y}_r)$ by definition.

According to the Woodbury matrix identity [24], we have $(\boldsymbol{A}+\boldsymbol{U}\boldsymbol{B}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}\left(\boldsymbol{I}+\boldsymbol{B}\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}\right)\boldsymbol{B}\boldsymbol{V}\boldsymbol{A}^{-1}$. We define $\boldsymbol{A} = \rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T$, $\boldsymbol{B} = 1$, $\boldsymbol{U} = \boldsymbol{\Phi}_c$, $\boldsymbol{V} = \boldsymbol{\Phi}_c'^T$, and $\boldsymbol{\Psi} = \boldsymbol{I} + \boldsymbol{\Phi}_c'^T \boldsymbol{A}^{-1} \boldsymbol{\Phi}_c$. From (7), we can obtain the following updating process for $\boldsymbol{w}$:

$$\begin{aligned}
\boldsymbol{w}_{\text{new}} &= \left(\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{A}^{-1}\right) [\boldsymbol{\Phi}\boldsymbol{y} + \boldsymbol{\Phi}_c \boldsymbol{y}_c] \\
&= \boldsymbol{w} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{y}_c - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{w} \\
&\quad - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{A}^{-1} \boldsymbol{\Phi}_c \boldsymbol{y}_c \\
&= \boldsymbol{w} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{y}_c - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c^T \boldsymbol{w} \\
&= \boldsymbol{w} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \left(\boldsymbol{\Phi}_c'^T \boldsymbol{w} - \boldsymbol{y}_c\right) \\
&= \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c \left(\rho \boldsymbol{I} + \boldsymbol{\Phi}_c'^T (\boldsymbol{C} - \boldsymbol{I}_J) \boldsymbol{\Phi}_c\right)^{-1} \left(\boldsymbol{\Phi}_c'^T \boldsymbol{w} - \boldsymbol{y}_c\right).
\end{aligned}$$

The last equality holds because

$$\begin{aligned}
\rho \boldsymbol{A}^{-1} + \boldsymbol{C} &= \rho \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1} + \boldsymbol{\Phi}\left[\rho \boldsymbol{I}_N + \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right]^{-1} \boldsymbol{\Phi}^T \\
&= \rho \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1} + \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1} \boldsymbol{\Phi}\boldsymbol{\Phi}^T \\
&= \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1} \left[\rho \boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right] = \boldsymbol{I}.
\end{aligned}$$

Similarly, we can compute $\boldsymbol{C}_{\text{new}}$ as

$$\begin{aligned}
\boldsymbol{C}_{\text{new}} &= \boldsymbol{\Phi}_{\text{new}}[\boldsymbol{\Phi}_{\text{new}}^T \boldsymbol{\Phi}_{\text{new}} + \rho \boldsymbol{I}_{N+N_a-N_r}]^{-1} \boldsymbol{\Phi}_{\text{new}}^T \\
&= [\rho \boldsymbol{I}_J + \boldsymbol{\Phi}_{\text{new}} \boldsymbol{\Phi}_{\text{new}}^T]^{-1} \boldsymbol{\Phi}_{\text{new}} \boldsymbol{\Phi}_{\text{new}}^T \\
&= [\boldsymbol{A} + \boldsymbol{\Phi}_a \boldsymbol{\Phi}_a^T - \boldsymbol{\Phi}_r \boldsymbol{\Phi}_r^T]^{-1} [\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_a \boldsymbol{\Phi}_a^T - \boldsymbol{\Phi}_r \boldsymbol{\Phi}_r^T] \\
&= [\boldsymbol{A} + \boldsymbol{\Phi}_c \boldsymbol{\Phi}_c'^T]^{-1} [\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_c \boldsymbol{\Phi}_c'^T] \\
&= \left(\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{A}^{-1}\right) [\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_c \boldsymbol{\Phi}_c'^T] \\
&= \boldsymbol{C} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Phi}_c'^T - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{C} \\
&\quad - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{A}^{-1} \boldsymbol{\Phi}_c \boldsymbol{\Phi}_c'^T \\
&= \boldsymbol{C} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T \boldsymbol{C} \\
&= \boldsymbol{C} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c \boldsymbol{\Psi}^{-1} \boldsymbol{\Phi}_c'^T (\boldsymbol{C} - \boldsymbol{I}_J) \\
&= \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c \left(\rho \boldsymbol{I} + \boldsymbol{\Phi}_c'^T (\boldsymbol{C} - \boldsymbol{I}_J) \boldsymbol{\Phi}_c\right)^{-1} \boldsymbol{\Phi}_c'^T (\boldsymbol{C} - \boldsymbol{I}_J).
\end{aligned}$$