Hw14
Lifelong Learning
ML TAs
ntu-ml-2021spring-ta@googlegroups.com
Outline

- Introduction
- Dataset
- Sample Code
- COOL Quiz
- Grading
- Submission
Introduction - LifeLong Learning

Goal: A model can beat all task!
Introduction - LifeLong Learning

Condition: Model Sequentially Learn Different Task! (In Training Time)
KeyPoint: Avoid Catastrophic Forgetting

Catastrophic Forgetting

Avoid Catastrophic Forgetting
Introduction

In Sample Code:

Continual Learning Methods

Replay-based methods
- Rehearsal
  - iCaRL [122]
  - ER [126]
  - SER [63]
  - TEM [26]
- Pseudo Rehearsal
  - DGR [139]
  - PR [13]
  - CCLUGM [81]
  - LGM [118]
- Constrained
  - GEM [92]
  - AGEM [25]
  - GSS [9]

Regularization-based methods
- Prior-focused
  - EWC [69]
  - IMM [83]
  - SI [168]
  - RWC [90]
  - MAS [5]
  - Remanian Walk [24]
- Data-focused
  - LWF [85]
  - LFL [68]
  - EBLL [120]
  - DMC [169]

Parameter isolation methods
- Fixed Network
- Dynamic Architectures
  - PackNet [96]
  - PathNet [38]
  - Piggyback [95]
  - HAT [136]
  - Progressive Networks [134]
  - ExpertGate [6]
  - RCL [160]
  - DAN [128]
Dataset

Permuted MNIST
Sample Code - Training Detail

- 5 task / Each task has 10 epochs for training.
- Each method cost ~20 minutes for training model.
- CoLab Link (copy to your drive first! Don’t simply run colab.)
Sample Code - Methods

- Baseline
- EWC
- MAS
- SI
- RWalk
- SCP
Sample Code - Guideline

- Utility
- Visualization
- Methods
Sample Code

- Utility
  - Permutation
  - Dataloader and Training Argument
  - Model
  - train
  - evaluate
  - evaluate metric

- Visualization

- Methods
Sample Code

- Utility
  - Permutation
  - Dataloader and Training Argument
  - Model
  - train
  - evaluate
  - evaluate metric

- Visualization

- Methods

5 PERMUTATION = 5 TASK
Sample Code

- Utility
  - Permutation
  - Dataloader and Train
    - Model
  - Evaluate
  - Evaluate metric

- Visualization

- Methods

```python
Model(
  (fc1): Linear(in_features=784, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=256, bias=True)
  (fc4): Linear(in_features=256, out_features=10, bias=True)
  (relu): ReLU()
)
```

Fixed model size!
Sample Code

- Utility
  - Permutation
  - Dataloader and Training Argument
  - Model
  - train
  - evaluate
  - evaluate metric

- Visualization
- Methods

In k th task: \[ \text{Average Accuracy}_k = \frac{1}{k} \sum_{j=1}^{k} a_{k,j} \]
Sample Code

- Utility
- Visualization
- Methods
Sample Code - Guideline

- Utility
- Visualization
- Methods
  - Baseline
  - EWC
  - MAS
  - SI
  - RWalk
  - SCP
Baseline

Sequentially Train

Training Pipeline:

Task Data

Lifelong Class (baseline, ewc, mas, si, rwalk, scp ..)

Model

Each Task

train

Evaluate

Change Task
Lifelong learning class

class baseline(object):
    ""
    baseline technique: do nothing in regularization term [initialize and all weight is zero]
    ""
    def __init__(self, model, dataloaders, device):
        self.model = model
        self.dataloaders = dataloaders
        self.device = device
        self.params = (n: p for n, p in self.model.named_parameters() if p.requires_grad)  # extract all parameters in models
        self.p_old = ()  # store current parameters
        self.precision_matrices = self.calculate_importance()  # generate weight matrix
        for n, p in self.params.items():
            self.p_old[n] = p.clone().detach()  # keep the old parameter in self.p_old
    
    def calculate_importance(self):
        precision_matrices = ()
        for n, p in self.params.items():  # initialize weight matrix (fill zero)
            precision_matrices[n] = p.clone().detach().fill_(0)
        return precision_matrices
    
    def penalty(self, model: nn.Module):
        loss = 0
        for n, p in model.named_parameters():
            _loss = self.precision_matrices[n] * (p - self.p_old[n]) ** 2
            loss += _loss.sum()
        return loss
    
    def update(self, model):
        # do nothing
        return

Lifelong Class
(baseline, ewc, mas, si, rwalk, scp ..)
Lifelong learning class

```python
6 class baseline(object):
    
    baseline technique: do nothing in regularization term [initialize and all weight is zero]
    
    def __init__(self, model, dataloaders, device):
        
        self.model = model
        self.dataloaders = dataloaders
        self.device = device
        
        self.params = {n: p for n, p in self.model.named_parameters() if p.requires_grad} # extract all parameters in models
        self.p_old = {} # store current parameters
        self.precision_matrices = self._calculate_importance() # generate weight matrix
        
        for n, p in self.params.items():
            self.p_old[n] = p.clone().detach() # keep the old parameter in self.p_old

    def _calculate_importance(self):
        precision_matrices = {}
        for n, p in self.params.items(): # initialize weight matrix (fill zero)
            precision_matrices[n] = p.clone().detach().fill_(0)
        return precision_matrices

    def penalty(self, model: nn.Module):
        loss = 0
        for n, p in model.named_parameters():
            loss = self.precision_matrices[n] * (p - self.p_old[n]) ** 2
            loss += _loss.sum()
        return loss

    def update(self, model):
        # do nothing
        return
```
EWC - Elastic Weight Consolidation

1. You need to know how to generate Guardiance weight from EWC!
2. Do this method need to use label?
3. Hint: (trace the class ewc and its calculate_importance function)

You need to know how to generate Guardiance weight from MAS!
Do this method need to use label?
Hint: (trace the class mas and its calculate_importance function)

Paper Link: https://arxiv.org/abs/1711.09601
1. You need to know how to generate Guardiance weight from SI!
2. Do this method need to use label?
3. Hint: (Accumulated loss change in each update step)

SI - Main Idea

$$L(\theta) = L_2(\theta) + c \sum_i \Omega_i (\theta_i - \theta_{1,i}^*)^2$$

From learning trajectory

Parameter importance on-line from learning trajectory!

Picture comes from: Talk Slide
Leveraging per-parameter importance for continual learning

\[ L(\theta) = L_2(\theta) + c \sum_i \Omega_i \left( \theta_i - \theta_{1,i}^* \right)^2 \]

\[ \Omega_i \equiv \frac{\omega_i}{(\Delta_i)^2 + \epsilon} \]

Picture comes from: Talk Slide
SI - Method

Total change in loss is given by the path integral over the gradient field

\[
\int_{C} g(\theta(t))d\theta = \int_{t_0}^{t_1} g(\theta(t)) \cdot \theta'(t)dt = L(t_1) - L(t_0)
\]

\[
= \sum_{k} \int_{t_0}^{t_1} g_k(t)\theta'_k(t)dt \equiv -\sum_{k} \omega_k
\]

- Is a parameter-specific quantity
- Can be computed on-line during training (running sum)

\[g: \text{Gradient} \]
\[\theta: \text{Parameters} \]
\[\theta': \text{Updates} \]

Natural way of assigning credit for a global change to local parameters

\[L(t_1) - L(t_0) = -\sum_{k} \omega_k^{t_t} \]

Picture comes from: Talk Slide
RWalk - Remanian Walk

1. Trace class rwalk and its update function!
2. Do this method need to use label?
3. Hint: (The code is similar to two method which mentioned in sample code)

Paper Link: https://arxiv.org/abs/1801.10112
SCP - Sliced Cramer Preservation

1. Paper Link: https://openreview.net/pdf?id=BJge3TNKwH

2. Do this method need to use label?
SCP - Main Idea

- Propose Distributed-based Distance to prevent fast intransigence and avoid overestimate the importance of parameter.

Model do not want to learn new task, and it just keep old task performance.
SCP - Sliced Cramer Preservation (Hint)

1.

Paper Link: https://openreview.net/pdf?id=BJge3TNKwH
COOL Quiz

- 25 multiple choice questions
- Basic Concept & Dataset: 4 Questions
- Sample Code: 15 Questions
  - EWC: 3 Questions
  - MAS: 3 Questions
  - SI: 3 Questions
  - Remanian Walk: 3 Questions
  - Sliced Cramer Preservation: 3 Questions
- Other Methods & scenario: 6 Questions
  - ICaRL, LwF, GEM, DGR
  - Three Scenario
Grading

- All Questions (0.4pt)
- You have to choose ALL the correct answers for each question
- Warning: The answers in NTU Cool are not correct, NTU grading score before deadline is not the final grading score.
- Please do not select the choice depends on the grading score.
Submission

- No late submission!
- We can only pick the last submission!
- Deadline: 2021/07/02 23:59

- Warning: The answers in NTU Cool are not correct, NTU grading score before deadline is not the final grading score.
- Please do not select the choice depends on the grading score.
Links

- CoLab Link
- NTU Cool Multiple Choice Question
- Warning: The answers in NTU Cool are not correct, NTU grading score before deadline is not the real grading score.
- Please do not select the choice depends on the grading score.
If any questions, you can ask us via...

- **NTU COOL (recommended)**
  - [https://cool.ntu.edu.tw/courses/4793](https://cool.ntu.edu.tw/courses/4793)

- **Email**
  - [ntu-ml-2021spring-ta@googlegroups.com](mailto:ntu-ml-2021spring-ta@googlegroups.com)
  - The title must begin with “[HW14]”

- **TA hours**
  - Each Monday 19:00~21:00 線上
  - Each Friday 13:30~14:20 線上
  - Each Friday During Class