Lifelong Learning:
Problem Statement & Approach to Solutions

Abhishek Aich
aaich@ece.ucr.edu

Department of Electrical and Computer Engineering

Overview

1. Problem Statement

Overview

1. Problem Statement

2. Approach to Solutions

Overview

1. Problem Statement
2. Approach to Solutions
   - Approach #1: Overcoming Catastrophic Forgetting in Neural Networks
   - Approach #2: Memory Aware Synapses: Learning what (not) to forget

Problem Statement

Define 'Forgetting':

- While learning a new task, neural networks have the tendencies to overwrite the parameters necessary to perform well at a previously trained task.

[1] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

Problem Statement

Define ‘Forgetting’:

- While learning a new task, neural networks have the tendencies to overwrite the parameters necessary to perform well at a previously trained task.

  e.g. a neural network trained to add 1 to a digit, and then trained to add 2 to a digit, would be unable to add 1 to a digit[1].

[1] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

Problem Statement

What are the assumptions?

- Tasks are in particular sequence as well as disjoint.
- Tasks may correspond to
  - different datasets, or
  - different splits of a dataset

  without overlap in category labels.
- When training a task, only the data related to that task is accessible.

Problem Statement

More Formally · · ·

- According to Kirkpatrick et al.[2]:
  - Set of tasks from the same dataset,
    e.g. classifying digits from the MNIST dataset.
  - Fixed sequence of tasks.
  - Offline storage of Fisher Information Matrix for each task and previous tasks model.

[2] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* (2017), p. 201611835.

[3] Rahaf Aljundi et al. "Memory Aware Synapses: Learning what (not) to forget". In: *arXiv preprint arXiv:1711.09601* (2017).

Problem Statement

More Formally ···

- According to Aljundi et al.[3]:
  - Set of tasks from different datasets,
    e.g. From datasets MIT *Scenes* for indoor scene classification and Caltech-UCSD *Birds* for fine-grained bird classification.
  - Fixed sequence of tasks.
  - Offline storage needed for importance weights and model parameters.

[2] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* (2017), p. 201611835.

[3] Rahaf Aljundi et al. "Memory Aware Synapses: Learning what (not) to forget". In: *arXiv preprint arXiv:1711.09601* (2017).

Approach to Solutions

Approach #1: Overcoming Catastrophic Forgetting in Neural Networks[2]

Let the set of weights and biases of a task γ, be denoted as θ_γ, and the estimated set as θ^*_γ.

- **Goal**: Target 'forgetting' in learning sequence of tasks by constraining important parameters to stay close to their old values.
- **Key**: There are many configurations of θ_A that will result in the same performance[1].

[2] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* (2017), p. 201611835.

[1] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

Approach to Solutions

Approach #1: Overcoming Catastrophic Forgetting in Neural Networks

- **Key**: There are many configurations of θ_A that will result in the same performance[1].

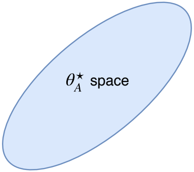


Figure 1: A space representing all possible configurations of θ_A

[1] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

- **Assumption**: There is a solution for task B, θ_B^*, that is close to the previously found solution for task A, θ_A^*.

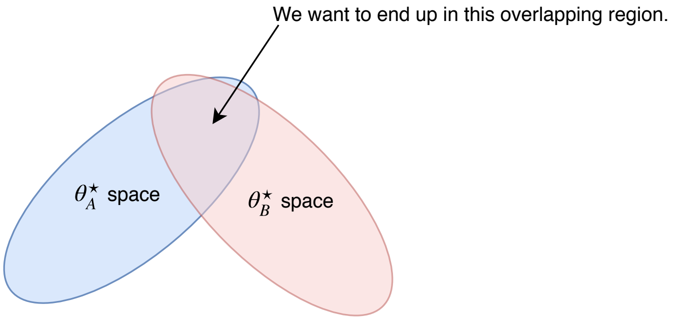


Figure 2: Solution space of θ_A and θ_B

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

- **Assumption**: There is a solution for task B, θ^*_B, that is close to the previously found solution for task A, θ^*_A.

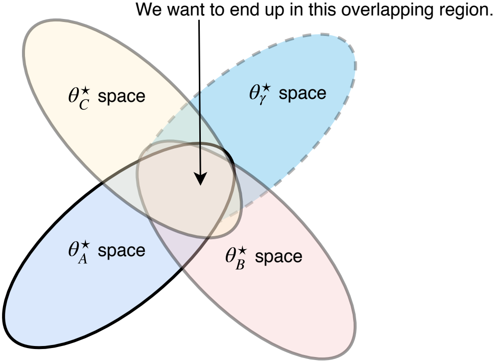
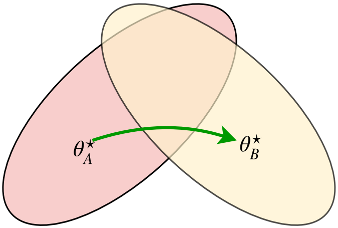


Figure 3: Solution space of θ_γ's for all γ tasks

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

Towards a solution $\cdots$



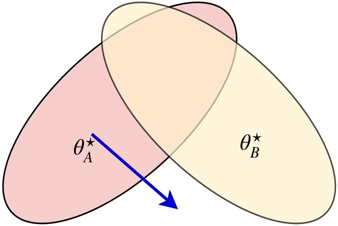
The parameters get updated according to the new dataset.

Figure 4: Train the network as it is: results in 'Forgetting'

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

Towards a solution $\cdots$



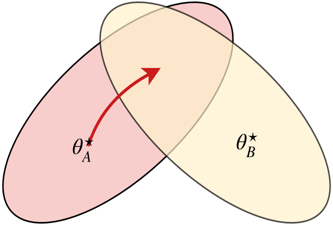
Say the parameters are made rigid and treated equally, then the network performs poorly for both the tasks.

Figure 5: Make no change in the parameters of previous tasks

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

Towards a solution ···



Say the parameters are made flexible and treated according to their importance, then the network performs well for both the tasks.

Figure 6: Make changes in the parameters of the previous tasks depending on their importance

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

What is the "importance" decided?

Suppose we have K tasks to be trained. Kirkpatrick et al. derive the following loss equation:

$$\mathcal{L}(\theta_{1:K}) = \mathcal{L}(\theta_K) + \frac{1}{2}\sum_i \lambda_K \left(\mathbf{F}_{1:K-1}\right)_{ii}\left(\theta_i - \theta^*_{1:K-1,i}\right)^2$$

- $\mathcal{L}(\theta_{1:K})$ = Current total loss to be minimized
- $\mathcal{L}(\theta_K)$ = Loss for the current task only
- $\theta^*_{1:K-1}$ = Optimal parameters for previous $K-1$ tasks
- λ = Hyperparameter that decides the influence of the importance of previous tasks
- $\mathbf{F}_{1:K-1}$ = Fisher information matrix (Indicates the importance)

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

FIM holds the answer $\cdots$

- Once a network is trained to a configuration θ^*_γ, $\mathbf{F}_{\theta^*_\gamma}$ indicates how prone each dimension in the parameter space is to causing forgetting when gradient descent updates the model to learn a new task.
- Preferable to move along the directions with low Fisher information.
- This approach uses $\mathbf{F}_{\theta^*_\gamma}$ in the regularization term to penalize moving in directions with higher Fisher information (more likely to result in forgetting of already-learned tasks).

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

FIM holds the answer $\cdots$

Assume the given network is already in trained in Task A. Then,

$$\theta_A^* = \arg\min_\theta \{-\log p(\theta|\mathcal{D}_A)\}$$

The gradient of $-\log p(\theta|\mathcal{D}_A)$ with respect to θ is 0 at θ_A^* , therefore $-\log p(\theta|\mathcal{D}_A)$ can be locally approximated as the following quadratic form (2nd order Taylor series around θ_A^*):

$$-\log p(\theta|\mathcal{D}_A) \approx \frac{1}{2}(\theta - \theta_A^*)\mathbf{H}(\theta_A^*)(\theta - \theta_A^*)$$

where $\mathbf{H}(\theta_A^*) =$ Hessian of $-\log p(\theta|\mathcal{D}_A)$ w.r.t. θ, evaluated at θ_A^*. Further, $\mathbf{H}(\theta_A^*) \succeq 0$ as θ_A^* is assumed to be a local minimum.

Approach to Solutions

Approach #1: **Overcoming Catastrophic Forgetting in Neural Networks**

FIM holds the answer $\cdots$

- Now, assuming that θ_A^* achieves near-perfect predictions on Task A, we can write

$$\mathbf{H}(\theta_A^*) \approx N_A \cdot \mathbf{F}(\theta_A^*)$$

  where N_A is the number of IID observations in $\mathcal{D}_A$, $\mathbf{F}(\theta_A^*)$ is the empirical Fisher information matrix on Task A.
- As the parameter space is high dimensional, EWC makes a further diagonal approximation of $\mathbf{F}(\theta_A^*)$, treating its off-diagonal entries as 0.

Approach to Solutions

Approach #2: **Memory Aware Synapses: Learning what (not) to forget[3]**

- This approach estimates an importance weight for each parameter in the network.
- Importance weights approximate the sensitivity of the learned function to a parameter change rather than a measure of the (inverse of) parameter uncertainty as in Approach #1.

[3] Rahaf Aljundi et al. "Memory Aware Synapses: Learning what (not) to forget". In: *arXiv preprint arXiv:1711.09601* (2017).

Approach to Solutions

Approach #2: **Memory Aware Synapses: Learning what (not) to forget**

- In a learning sequence, we start with task T_1, training the model to minimize the task loss $\mathcal{L}_1$ on the training data $(X_1, \hat{Y}_1)$.
- After convergence, the model has learned a function F that maps input X_1 to output Y_1.
- **Goal:** Preserve this mapping while learning additional tasks.

Approach to Solutions

Approach #2: **Memory Aware Synapses: Learning what (not) to forget**

- In a learning sequence, we start with task T_1, training the model to minimize the task loss $\mathcal{L}_1$ on the training data $(X_1, \hat{Y}_1)$.

- After convergence, the model has learned a function F that maps input X_1 to output Y_1.

- **Goal:** Preserve this mapping while learning additional tasks.

- **Key:** Measure sensitivity of the parameters!

Approach to Solutions

Approach #2: **Memory Aware Synapses: Learning what (not) to forget**

Towards a solution $\cdots$

For a given data point x_k, the output of the network is $F(x_k; \theta)$. Suppose we introduce a small perturbation in the parameters θ, this results in a change given by

$$F(x_k; \theta + \delta) - F(x_k; \theta) \approx \sum_{i,j} g_{ij}(x_k) \cdot \delta_{ij}$$

where $g_{ij}(x_k) = \dfrac{\partial F(x_k; \theta)}{\partial \theta_{ij}}$ = gradient of the learned function w.r.t. the parameter θ_{ij} evaluated at x_k.

Approach to Solutions

Approach #2: **Memory Aware Synapses: Learning what (not) to forget**

Towards a solution $\cdots$

Finally, the "importance" is measured by the magnitude of gradient $g_{ij}(x_k)$. Accumulate the gradients overall given data points to obtain

$$\text{Importance weight, } \Omega_{ij} = \frac{1}{N}\sum_{k=1}^{N} \|g_{ij}(x_k)\|$$

- Parameters with
  - small Ω_{ij} do not affect the output much $\rightarrow$ should be changed to minimize the loss for subsequent tasks.
  - large Ω_{ij} affect the output much $\rightarrow$ should be left unchanged to minimize the loss for subsequent tasks.

Approach to Solutions

Approach #2: **Memory Aware Synapses: Learning what (not) to forget**

Similar loss equation but different method for importance ⋯

When a new task T_K is to be learned, we have a regularizer that penalizes changes to parameters as per their importance in addition to the new task loss $\mathcal{L}(\theta_K)$:

$$\mathcal{L}(\theta_{1:K}) = \mathcal{L}(\theta_K) + \frac{1}{2}\sum_{i,j}\lambda_K \left(\Omega_{1:K-1}\right)_{ij}\left(\theta_{i,j} - (\theta_{1:K-1})^*_{i,j}\right)^2$$

Thank You!