catastrophic interference

Saturday, May 11, 2024 4:06 PM

https://datascientest.com/en/catastrophic-interference-what-is-it-how-to-avoid-it

What is catastrophic interference?

- One important feature distinguishes the human brain from artificial neural networks: plasticity.
- Plasticity is the human capacity for continuous learning.
- When learning a new task, NN tend to forget what they've learned before.
- For example, in a famous experiment conducted by McCloskey and Cohen in 1989, the researchers trained a neural network to solve mathematical problems based on examples containing the number 1.
- They then fed the model another series of problems, this time containing the number 2. As a result, the neural network learned to solve problems containing a 2, but forgot how to solve those with the number 1
- Catastrophic Forgetting, also known as Catastrophic Interference, is a phenomenon that occurs when a neural network or machine learning model "forgets", or dramatically reduces its performance on previously learned tasks after learning a new task. This can occur when training a model on a stream of tasks, rather than training it on all tasks at once.

Why does it happend?

- There are a few different ways that catastrophic forgetting can occur. One way is through the process of "overfitting", where the model is so focused on fitting the training data for the new task that it forgets the information from the previous tasks.
- When you feed it new information, new pathways are formed, sometimes causing the algorithm to "forget" the previous tasks it was trained for. Sometimes, the margin of error increases, but other times, the machine completely forgets the task. This is what's called Catastrophic Forgetting or Catastrophic Interference.

Continuous ML

continual learning is also referred to as incremental learning or lifelong learning in much of the literature, without a strict distinction.

A major challenge is known as catastrophic forgetting, where adaptation to a new distribution generally results in a largely reduced ability to capture the old ones. This dilemma is a facet of the trade-off between learning plasticity and memory stability.

- Learning plasticity refers to the ability of a model or system to adapt and learn from new information or experiences. In neural networks, plasticity is often associated with the ability to update synaptic weights or network parameters in response to changes in the input data or task requirements.
- Memory stability refers to the ability of a model to retain previously learned information or experiences over time, despite exposure to new data or tasks. In the context of neural networks, memory stability is closely related to the concept of retaining useful representations or knowledge in the network's parameters or activations.

Learning senarios: https://www.nature.com/articles/s42256-022-00568-3

Class-Incremental Learning

- **Definition**: The model incrementally learns to discriminate between classes that are observed sequentially.
- **Class**: Refers to a category or label that the model learns to identify or classify (e.g., "cat", "dog").
- **Incremental Observation**: New classes are introduced over time, and the model must learn to recognize these new classes without forgetting previously learned ones.

Task-Incremental Learning

- **Definition**: The model learns to solve a sequence of distinct tasks, each with its own unique set of classes or objectives.
- **Task**: Refers to a distinct learning problem or objective, often with a unique set of classes (e.g., task 1 might involve classifying animals, while task 2 involves classifying vehicles).
- **Task Identities:** The model is aware of which task it is learning or being tested on. Task identities are provided during both training and testing.

Domain-Incremental Learning

- **Definition**: The model learns to solve the same problem across different contexts or domains, which have different input distributions but share the same label space.
- **Domain**: Refers to variations in the input distribution or context in which the problem is solved (e.g., different environments, lighting conditions, or sensor types).
- **Same Problem:** The underlying problem or classification task remains the same, but the context in which the data is presented changes.

Evalution Metrics:

https://arxiv.org/pdf/1706.08840: average accuracy, intransience measure, forgetting measure
https://arxiv.org/abs/1706.08840: backward/forward transfer

- **Overall performance:**
  - Average accuracy: Average accuracy measures the overall accuracy of the model across all tasks. **Purpose**: This metric gives a holistic view of the model's performance across all learned tasks at the end of the training sequence.
  - Average incremental accuracy: https://arxiv.org/pdf/1611.07725
    for class-incremental learning methods, a measure of a model's performance across all tasks. **Purpose**: This metric provides an overview of the model's ability to maintain high performance as it learns new tasks incrementally.

  Memory Stability Metric:
  - Forgetting Measure: quantifies the degree to which a model forgets previously learned tasks as it learns new ones. **Purpose**: This metric evaluates how well the model retains its knowledge over time.
    Example: If the accuracy on a task drops from 90% to 70% after learning subsequent tasks, the forgetting for that task is 20%. The FM is the average forgetting across all tasks.
  - Backward Transfer (BWT): measures the impact of learning new tasks on the performance of previously learned tasks.Example: If the accuracy on a task improves from 70% to 75% after learning a new task, the BWT is +5%. If it decreases from 70% to 65%, the BWT is -5%.
- **Learning Plasticity Metrics:**
  - Intransience Measure: for class-incremental learning method evaluates the model's ability to acquire new knowledge without being impeded by previously learned tasks. **Example**: If a model achieves 85% accuracy on a new task when learned in isolation but only 80% when learned sequentially, the intransience is indicated by the 5% drop.
  - Forward Transfer (FWT): measures the influence of previously learned tasks on the learning of new tasks. Example: If the accuracy on a new task is 80% when learned sequentially but only 75% when learned in isolation, the FWT is +5%.

Prequels to Knowledge Distillation

Soft targets: are the probability distributions over classes produced by a model, typically by applying a softmax function to the model's output logits. Unlike hard targets, which are the one-hot encoded vectors indicating the correct class, soft targets provide richer information by conveying the model's confidence in each class.

For example, if a model outputs logits z=[2.0,1.0,0.1] for a three-class problem, the softmax function can convert these logits to soft targets p=[0.65,0.24,0.11]

Temperature:
is used to control the smoothness of the output probability distribution. Adjusting the temperature can make the output probabilities more or less confident, which is particularly useful for training a student model to better mimic the teacher model's behavior.

https://arxiv.org/abs/2109.00525

Lifelong ML

One of the most significant and challenging open problems in Artificial Intelligence (AI) is the problem of Lifelong Learning. Lifelong Machine Learning considers systems that can continually learn many tasks (from one or more domains) over a lifetime. A lifelong learning system efficiently and effectively:

- retains the knowledge it has learned from different tasks;
- selectively transfers knowledge (from previously learned tasks) to facilitate the learning of new tasks;
- ensures the effective and efficient interaction between (1) and (2).

Lifelong Learning introduces several fundamental challenges in training models that generally do not arise in a single task batch learning setting. This includes problems like catastrophic forgetting and capacity saturation. This workshop aims to explore solutions for these problems in both supervised learning and reinforcement learning settings.

List of Survey

- https://arxiv.org/pdf/2302.03648
- https://arxiv.org/pdf/2302.00487
- https://arxiv.org/pdf/2006.05525

Regularization techniques are methods used in machine learning to prevent overfitting by adding constraints or penalties to the learning process. In the context of catastrophic forgetting, regularization techniques help a model retain previously learned information while learning new tasks. These techniques adjust the training process to balance the learning of new information with the preservation of existing knowledge.

EWC https://arxiv.org/pdf/1612.00796

EWC is a specific regularization technique designed to address catastrophic forgetting.
How EWC Works:

- Identify Important Weights: After training the model on the first task, EWC identifies which parameters (weights) are crucial for that task. This is done by calculating the Fisher Information Matrix, which measures the importance of each parameter. The idea is that parameters with higher Fisher values are more important.
- Penalize Changes to Important Weights: When the model begins training on a new task, EWC adds a penalty to the loss function. This penalty discourages significant changes to the important weights identified from the previous task. Mathematically, this is expressed as an additional term in the loss function.

$$L = L_{\text{new task}} + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta_i^*)^2$$

- $L_{\text{new task}}$ is the loss for the new task.
- λ is a hyperparameter that controls the strength of the regularization.
- F_i is the Fisher Information Matrix for the parameter θ_i.
- θ_i^* are the parameters learned from the previous task.

Synaptic Intelligence (SI) https://arxiv.org/pdf/1703.04200

1. Importance of Synapses:
   - In a neural network, "synapses" refer to the connections between neurons, represented by weights.
   - SI assigns an importance value to each synapse based on its contribution to the loss reduction during training on a specific task. This importance reflects how crucial a particular weight is for the current task.
2. Cumulative Penalty:
   - Instead of just penalizing changes to important weights from the previous task (like in EWC), **SI accumulates the importance of weights over multiple tasks.**
   - When learning a new task, SI adds a penalty term to the loss function that is proportional to the accumulated importance of the weights and the amount of change in those weights.

Calculating Synaptic Importance:
After training on a task, SI calculates the importance of each weight. This is done by summing the product of the weight's gradient and the change in the weight over all training steps.
The importance score Ω for each weight θ is given by

$$\Omega_i = \sum_{t=1}^{T} \frac{\partial L}{\partial \theta_i}(\theta_i^{(t)} - \theta_i^{(t-1)})$$

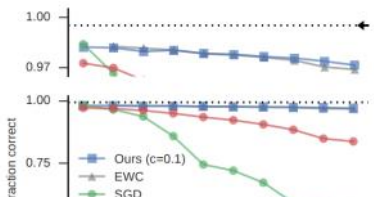
where $\frac{\partial L}{\partial \theta_i}$ is the gradient of the loss with respect to weight θ_i, and $\theta_i^{(t)}$ is the value of the weight at step t.

Penalizing Weight Changes:
When training on a new task, SI adds a regularization term to the loss function to penalize changes to important weights:

$$L = L_{\text{new task}} + \sum_i \Omega_i(\theta_i - \theta_i^*)^2$$

where Ωi is the accumulated importance of weight θi, and θi∗ is the value of the weight after training on the previous task.

Test: chia làm 5 Task (0-1) (2-3)(4-5)(6-7)(8-9). Model là MLP 2 hidden layer of 256 dùng Relu, dùng cross-entropy, batch size 64, 10 epochs, Adam optimizer



Temperature:
is used to control the smoothness of the output probability distribution. Adjusting the temperature can make the output probabilities more or less confident, which is particularly useful for training a student model to better mimic the teacher model's behavior.

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

where $\mathbf{z} = [z_1, z_2, \ldots, z_n]$ are the logits.

When incorporating a temperature parameter T, the softmax function is modified as follows:

$$\sigma_T(\mathbf{z})_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

Higher temperature values help in preserving the teacher model's knowledge by emphasizing the relationships between different classes rather than focusing solely on the correct class.

The Kullback–Leibler (KL) divergence: is a measure from information theory that quantifies the difference between two probability distributions. It is also known as relative entropy. In the context of knowledge distillation, KL divergence is used to measure how one probability distribution (typically the soft targets from the teacher model) diverges from a second probability distribution (the outputs from the student model).

For two discrete probability distributions P and Q, the KL divergence from Q to P is defined as:

$$D_{KL}(P\|Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

Here:

- $P(i)$ is the probability of the i-th outcome according to the distribution P.
- $Q(i)$ is the probability of the i-th outcome according to the distribution Q.

Knowlege Distillation
Main article: https://arxiv.org/abs/1503.02531
Survey : https://arxiv.org/pdf/2006.05525

Knowledge distillation is a technique in machine learning where the knowledge from a larger, more complex model (often referred to as the "teacher" model) is transferred to a smaller, more efficient model (the "student" model). This method aims to retain the performance and knowledge of the teacher model while benefiting from the reduced computational cost and memory footprint of the student model. Knowledge distillation can be particularly useful in scenarios where deploying a large model is impractical due to resource constraints.

The overall loss function used to train the student model is a weighted sum of the cross-entropy loss and the KL divergence:

$$L_{\text{total}} = \alpha L_{\text{CE}} + (1 - \alpha) L_{\text{KL}}$$

Step-by-Step:

- Training the Teacher Model: like mormal
- Soft Targets Generation: After training use the teacher model to predict training data, apply temperature-scaled softmax to the logits produced by the teacher model to obtain the soft target
- Training the Student Model with disillation_loss

- Cross-Entropy Loss:

$$L_{\text{CE}} = -\sum_i y_i \log(p_i)$$

where y_i is the true label and p_i is the predicted probability for class i.

- KL Divergence:

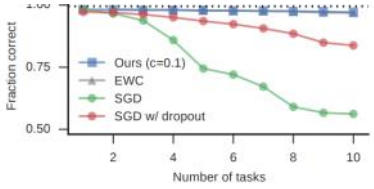
$$L_{\text{KL}} = T^2 \sum_i \sigma_T(\mathbf{z}_t)_i \log\left(\frac{\sigma_T(\mathbf{z}_t)_i}{\sigma_T(\mathbf{z}_s)_i}\right)$$

where $\sigma_T(\mathbf{z}_t)$ and $\sigma_T(\mathbf{z}_s)$ are the soft targets from the teacher and student models, respectively.

- Combined Loss:

$$L_{\text{total}} = \alpha L_{\text{CE}} + (1 - \alpha) L_{\text{KL}}$$

Learning without Forgetting: Combination of Distillation Networks and fine-tunning. State-of-the-art algorithm



Rwalk: combination of SI and EWC https://arxiv.org/pdf/1801.10112
R-EWC: diagonalize the FIM: https://arxiv.org/pdf/1802.02950
AFEC : mechanism that enhances model adaptability and efficiency by actively removing outdated or less relevant data from the model's memory. https://arxiv.org/abs/2110.12187

Replay-Based Approach https://arxiv.org/pdf/2108.06758

Concept: The fundamental idea behind replay-based approaches is to store and replay past experiences to maintain the model's performance on previously learned tasks while learning new ones.

Experience replay: This approach involves saving a subset of data from previously learned tasks in a memory buffer. During training on new tasks, the model intermittently revisits this stored data to reinforce old knowledge. Due to the extremely limited storage space, the key challenges consist of how to construct and how to exploit the memory buffer.

- Sample selection
- Improve storage efficiency

Method:

- iCaRL(Incremental Classifier and Representation Learning): ER + KD https://arxiv.org/abs/1611.07725
- ER-Reservoir(Experience Replay with Reservoir Sampling): https://arxiv.org/pdf/1902.10486
- AGEM(Average Gradient Episodic Memory): https://arxiv.org/abs/1812.00420

Generative Replay: using generative models to recreate past data rather than storing actual data samples. With advantages of: Memory Efficiency, Scalability and Flexibility.
Concept:

- Generative Models: Instead of storing raw data samples from previous tasks, a generative model (e.g., Generative Adversarial Network (GAN), Variational Autoencoder (VAE)) is trained to generate synthetic data that mimics the distribution of the original data.
- Replay Mechanism: During training on new tasks, the model generates synthetic data representing past tasks and interleaves this with the new data to maintain performance on previous tasks.

Method:

- DGR(Deep Generative Replay): https://arxiv.org/abs/1705.08690
- MeRGAN(Memory Replay GANs): https://arxiv.org/abs/1809.02058
- iCaRL with Generative Models: https://arxiv.org/abs/1904.03137

Optimization-Based Method: manipulating the optimization programs.
Architecture-Based Approach: manipulating parameters of the models