Memory Aware Synapses: Learning what(not) to forget (ECCV2018)

Introduction

Fig. 1: Our continuous learning setup. As common in the LLL literature, tasks are learned in sequence, one after the other. If, in between learning tasks, the agent is active and performs the learned tasks, we can use these unlabeled samples to update importance weights for the model parameters. Data that appears frequently, will have a bigger contribution. This way, the agent learns what is important and should not be forgotten.

- 기존의 Continual learning은 두 가지 task 사이에서 수행되거나 여유로운 capacity, 시간 등이 여유로운 상황에서의 학습이 수행되었다.
- 하지만, real-world에서는 제한된 환경(capacity) 에서 여러가지 task가 계속해서 주어지는 상황을 커버해야한다.
- 따라서, 본 논문에서는 이전 task들에 대해 model이 전부 기억하려고 하기보다는 덜 중요한 정보는 잊고 중요한 정보는 보존하는 방식을 제안한다. 즉, Network의 parameter weight 들의 importance를 계산하고, 계산된 중요도를 기반으로 Regularization term을 통해 중요한 weight의 갱신을 방지한다.

→ LLL(Lifelong learning) 방법은 주로 짧은 sequence에 적용되며, 두개 이하의 작업으로 구성되며, 상대적으로 큰 용량을 갖춘 네트워크를 사용 **하지만,** 끝 없는 작업으로 모델은 용량에 한계에 도달한다.

작업 간의 관련성이 낮거나 현재 작업에 더 중요하지 않은 정보를 forgetting 하여 모델 용량을 확보한다.

Contribution of this paper are threefold:

1. 새로운 LLL(Lifelong learning) 방법인 MAS를 제안.
   network parameter에 대한 중요도 weight를 unsupervised, online manner 방식으로 추정하며, unlabeled data에 적응할수 있게 한다.
2. MAS가 어떻게 local variant가 Hebbian learning scheme 이 연결되어 있는지 보여준다.

Proposed Method

저자들은 핵심 아이디어는 " 모든 parameter를 regularization 하지 말고, 중요한 parameter만 regularization 하는 것 " 이다.

→ MAS 는 Loss에 의존하지 않고, 출력함수의 민감도를 살펴봄으로써 network parameter의 importance weight를 추정함으로써 달성된다.

이때 중요한 parameter는 어떻게 정의할까 ?

저자들은 "parameter 값이 약간 변하더라도 feature값이 크게 변하는 parameter를 중요한 parameter " 라고 정의 합니다.

4.1 Estimating parameter importance

- Importance weight는 network의 parameter 변화에 대한 learned function의 sensitivity
- Data point에 대해 parameter가 small perturbation으로 변했을 때, output function의 변화량을 다음과 같이 근사화 했다.

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

- 좌변은 파라미터 θ가 미세하게 δ 만큼 변했을 때의 feature 값의 차이다. 이는 우변의 gradient로 표현할 수 있다. 따라서 parameter의 중요도는 아래와 같이 gradient의 크기로 표현할 수 있다.

Importance weight

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

- Importance weight는 network의 parameter 변화에 대한 학습된 function의 민감도를 나타낸 것으로 모든 data point에서 parameter에 대한 Output function의 변화량의 평균을 뜻한다.
  즉, network의 parameter가 변할 때 learned function 대비 output function이 크게 변하면 해당 parameter에 높은 중요도를 부여한다.
- 즉, parameter의 gradient가 크면 새로운 task로 학습시킬 때 backproprogation을 적게하고, 반대로 gradient가 작으면 새로운task를 학습시킬 때 정상적으로 backpropagation을 해서 parameter를 변화시키면 된다.

$$g_{ij}(x_k) = \frac{\partial[\ell_2^2(F(x_k;\theta))]}{\partial\theta_{ij}}$$

- Importance weight를 계산할 때, output function이 multi-dimension인 경우 계산해야하는 gradient가 많아진다.
- 이때 문제는 각 차원별로 전부 gradient를 구해 backpropagation을 해줘야한다는 것인데,
- 차원을 줄이기 위해 F를 그대로 쓰지 않고, learned function의 output에 L2 norm을 취하여 n(length of output vector) 배 빠른 계산을 수행하게 한다.

→ learned function의 출력을 L2 norm을 취하는 것은 다차원 출력을 하나의 스칼라 값으로 변환하는 작업이다.

이제 이 출력 값들을 L2 norm을 취하여 하나의 스칼라 값을 얻을 수 있다.

```
l2_norm = sqrt(f1(x)^2 + f2(x)^2+...+fn(x)^2)
```

- 이렇게하면 각 차원의 출력값이 제곱되고 더해진 후 루트를 씌워진 결과가 나오게된다.
- 이 l2_norm 값은 다차원 출력의 크기 정보를 담고 있으면서도 하나의 숫자로 표현된다.
- 이와 같은 접근 방식은 차원 축소와 중요도 부여를 효과적으로 결합한 방법으로, 모델이 다차원 출력을 다루는 동시에 중요한 패턴에 집중하도록 도와줍니다. 하지만 이러한 방식을

사용할 때에도 해당 차원에 대한 gradient 계산과 업데이트 작업이 필요하며, 모델이 architecture와 학습 방법을 조정해가며 실험하여 최적의 결과를 얻을 수 있다.

4.2 Learning a new task

- N 번째 task의 학습에 대해 loss function은 다음과 같이 정의 된다.

$$L(\theta) = L_n(\theta) + \lambda \sum_{i,j} \Omega_{ij}(\theta_{ij} - \theta^*_{ij})^2$$

새로운 task의 loss function

기존 parameter와 새로운 parameter의 차이*
importance weight

- 우변에 regularization 항이 추가된 것을 확인할 수 있는데, 이때 위에 설명한 parameter importance에 해당하며, 모든 parameter에 대해 일관된 규제를 하는 방식이 아닌, 중요도가 큰 parameter에만 집중적으로 변화량에 규제한다고 생각할 수 있다.

⬇️

- important weight = high importance weight → 새로운 parameter가 기존 parameter에서 크게 변하지 못함.
- forgettable weight = low importance weight → 새로운 parameter가 기존 parameter에서 크게 변함
- 높은 중요도를 가진 weight는 기존 parameter에서 크게 변하지 못하게 하고, 낮은 중요도를 가진 weight는 기존 parameter에서 크게 변하는 것을 허용한다.

4.3 Connection to Hebbian learning

Hebbian learning theory는 다음과 같다.

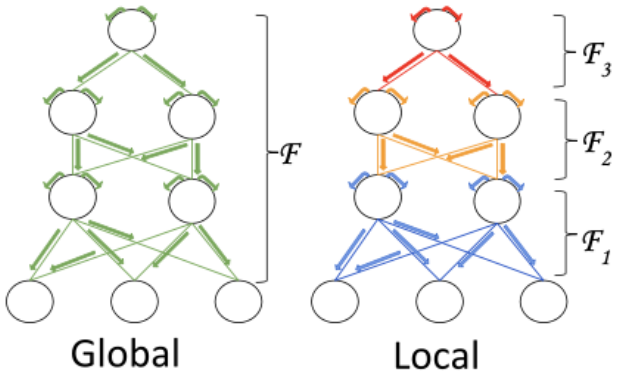
> “하나의 자극에 대해 같이 활성화되는 neuron synapse 들은 연결이 더 강화되어, 그 결과 해당 자극에 대한 성능이 더 좋아진다.”

이를 Deep learning 관점에서 해석하면, Parameter의 importance는 neuron의 activation correlation에 의해 결정된다고 할 수 있다.

이를 수식으로 표현하면 다음과 같다.

$$\Omega_{ij} = \frac{1}{N}\sum_{k=1}^{N} y_i^k * y_j^k$$

위에서 제안한 방법은 모델 전체의 parameter에 영향을 주는 방법이다. 이를 layer마다 나누어 생각해보자.



위 그림의 local 한 방식처럼 layer마다 나눠서 생각해보면, 위에서 구한 gradient는 다음과 같다.

$$\ell_2^2(F_l(y^k;\theta_l+\delta_l)) - \ell_2^2(F_l(y^k;\theta_l)) \approx \sum_{i,j} g_{ij}(x_k)\delta_{ij}$$

이때 gradient는 다음과 같이 표현이 된다.

$$g_{ij}(x_k) = \frac{\partial[\ell_2^2(F(x_k;\theta))]}{\partial\theta_{ij}}$$

이때 activation이 ReLU일 때는 다음과 같이 표현할 수 있다.

$$g_{ij}(x_k) = 2 * y_i^k * y_j^k$$

결국 위에서 본 Hebbian learning을 수식으로 표현한 것은 다음과 같은 식이 되었다. 따라서 저자들은 본인들의 제안 방식은 신경과학적 관점과 동일하다고 주장합니다.

정리하면 제시하는 Memory Aware Synapses (MAS) 는

- 이 방법은 new data point에 적용될 때 network의 activation을 기반으로 해당 값을 계속 update 한다.
- 이 방법은 network의 모든 기능을 보존하는 대신 특정 data point 하위 집합에 적응하고 특화될 수 있다.
- 더 나아가 network를 훈련한 후에 추가 할 수 있으며, 사전에 훈련된 어떤 network 외에도 적용할 수있으며 label을 가져올 필요 없이 어떤 dataset 에서도 중요도를 계산할 수 있다.
- 이는 작업을 parameter의 중요도를 계산하기 위해 loss function에 의존하는 방법과 구별하는 중요한 기준이 된다.

Experiments

5.1 Object Recognition

- ImageNet으로 pretrain한 AlexNet을 기반으로 실험 수행
- Task dataset: MIT scene, Caltech-UCSD, Oxford Flowers

Method	Birds → Scenes		Scenes → Birds		Flower → Birds		Flower → Scenes	
FineTune	45.20 (-8.0)	**57.8**	49.7 (-9.3)	**52.8**	64.87 (-13.2)	**53.8**	70.17 (-7.9)	57.31
LwF [17]	51.65 (-2.0)	55.59	55.89 (-3.1)	49.46	73.97 (-4.1)	53.64	76.20 (-1.9)	58.05
EBLL [28]	52.79 (-0.8)	55.67	56.34 (-2.7)	49.41	75.45 (-2.6)	50.51	76.20 (-1.9)	**58.35**
IMM [16]	51.51 (-2.1)	52.62	54.76 (-4.2)	52.20	75.68 (-2.4)	48.32	76.28 (-1.8)	55.64
EWC [12]	52.19 (-1.4)	55.74	**58.28 (-0.8)**	49.65	76.46 (-1.6)	50.7	77.0 (-1.1)	57.53
SI [39]	52.64 (-1.0)	55.89	57.46 (-1.5)	49.70	75.19 (-2.9)	51.20	76.61 (-1.5)	57.53
MAS (ours)	**53.24 (-0.4)**	55.0	57.61 (-1.4)	49.62	**77.33 (-0.7)**	50.39	**77.24 (-0.8)**	57.38

Table 2: Classification accuracy (%), drop in first task (%) for various sequences of 2 tasks using the object recognition setup.

Method	Ω_{ij} computed. on	Birds → Scenes		Scenes → Birds		Flower → Bird		Flower → Scenes	
MAS	Train	53.24 (-0.4)	55.0	57.61 (-1.4)	49.62	77.33 (-0.7)	50.39	77.24 (-0.8)	57.38
MAS	Test	53.43 (-0.2)	55.07	57.31 (-1.7)	49.01	77.62 (-0.5)	50.29	77.45 (-0.6)	57.45
MAS	Train + Test	53.29 (-0.3)	56.04	57.83 (-1.2)	49.56	77.52 (-0.6)	49.70	77.54 (-0.5)	57.39
l-MAS	Train	51.36 (-2.3)	55.67	57.61 (-1.4)	49.86	73.96 (-4.1)	50.5	76.20 (-1.9)	56.68
l-MAS	Test	51.62 (-2.0)	53.95	55.74 (-3.3)	50.43	74.48 (-3.6)	50.32	76.56 (-1.5)	57.83
l-MAS	Train + Test	52.15 (-1.5)	54.40	56.79 (-2.2)	48.92	73.73 (-4.3)	50.5	76.41 (-1.7)	57.91

Table 3: Classification accuracies (%) for the object recognition setup - comparison between using Train and Test data (unlabeled) to compute the parameter importance Ω_{ij}.

5.3 Longer sequence

- Sequence of 8 tasks 에서 실험 수행
- Stanford Cars, FGVC-Aircraft, VOC Actions, Letters, SVHN 5가지 task dataset 추가 사용
- Sequence: Flower → Scenes → Birds → Cars → Aircraft → Letters → SVHN

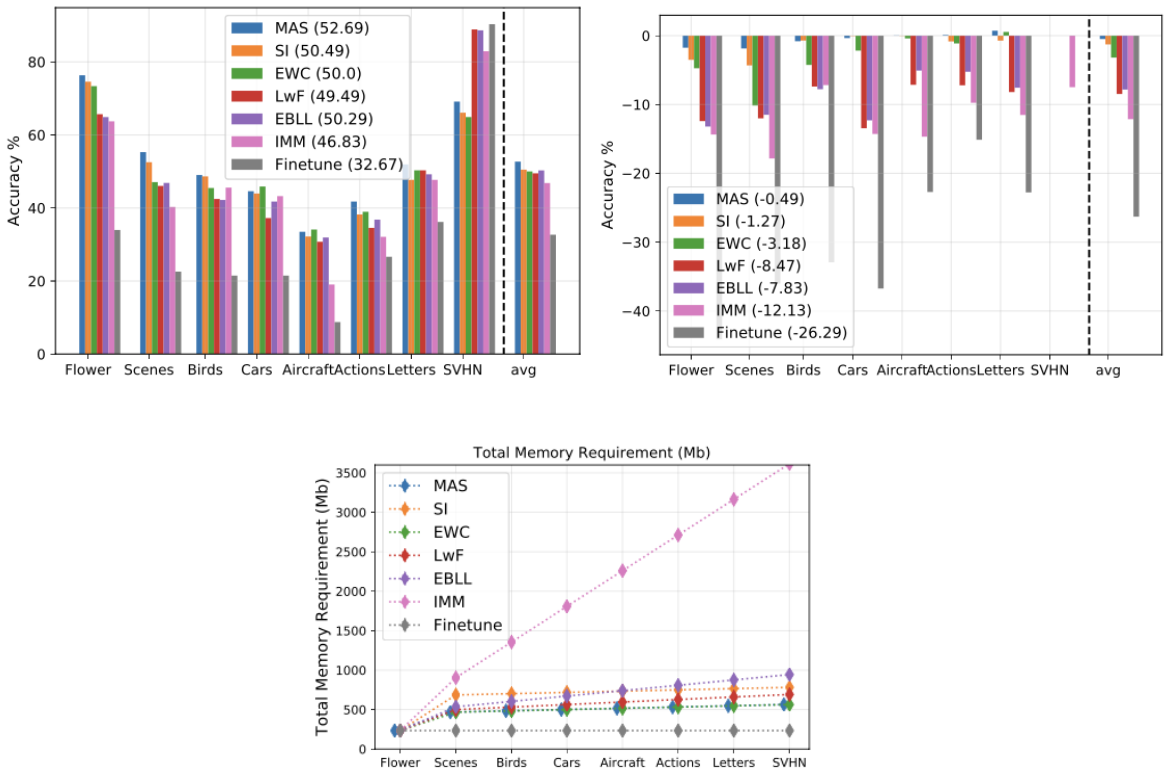


Fig. 4: Overall memory requirement for each method at each step of the sequence.

Conclusion

- 본 논문에서는 유한한 model capacity와 무한히 다양한 발전하는 작업들이 주어진 경우, 모든 이전 지식을 보존하는 것은 불가능한 대신, 무엇을 학습해야 하는지 학습한다.
- Forgetting은 특정 knowledge piece이 사용되는 rate과 관련이 있어야 한다.
- 본 논문에서는 system이 activation이 되는 input data로 부터 network의 parameter의 중요성을 unsupervised method를 학습하는 방법을 제안한다.
- 본 논문은 local variant이 parameter의 importance 를 학습하는데 Hebbian learning theory를 적용한다.