A Comprehensive Survey of Continual Learning: Theory, Method and Application

인공지능과 유기체의 근본적 차이

- 인공지능은 일반적인 유기체들처럼 학습 프레임워크와 인지건축 프레임워크가 통합된 구조가 아님.

Continual learning이 시사하는 문제점 그리고 딜레마

- continual learning의 대표적인 문제는 "catastrophic forgetting".
- learning plasticity and memory stability 딜레마.
  - plasticity: 받을 적응력

continual learning의 방법론들

- continual learning은 usuable resource를 최대로 활용하는 선에서 노력함. 따라서 continual learning이 추구하는 이상이 있는 방법은 오직 새로운 training sample만 가지고 학습하는 방법.
- continual learning은 딥러닝 학습 프레임워크에서 어느 부분에서 초점을 맞추냐에 따라 방법론을 크게 분류할 수 있음.
  - * replay: 기존 task의 샘플을 다시 가져다서 최적화에 활용.
  - * architecture: 학습하는 네트워크 구조를 점점히 변형하여 필요한 부분만 학습.
  - * representation: 변화하는 상황에 대해서 robust한 representation을 학습.
  - * optimization: 기존 representation에 drift가 크지 않는 방향으로 최적화 진행.
  - * regularization: 학습 시 정규화를 통해 학습 안정성 ↑.

Continual learning의 다양한 시나리오들

Instance-Incremental Learning (IIL)

- 모든 학습 sample은 같은 task에 속하며, 배치 단위로 들어옴.
- 이는 흔 일반적인 딥러닝 학습 알고리즘이나의 setting과 동일한 상황에서 input dataset만 continual하게 들어오는 상황을 가정하며, 단일 task를 가정하기 때문에 task specific identification 요구X.

Domain-Incremental Learning (DIL)

- IIL과 다르게 단일 task가 아닌 여러 tasks가 존재하는 상황을 전제로 함.
- 모든 task들은 동일한 data label space를 가지고 input distribution만 다름.
- 단일 task가 아님에도 불구하고 task specific identification 요구X.

Task-Incremental Learning (TIL)

- 각 task들 간 disjoint한 data label space를 가짐.
- task specific identity에 대한 정보가 training 및 testing에서 제공됨.

Class-Incremental Learning (CIL)

- 각 task들 간 disjoint한 data label space를 가짐.
- 인지건축 관점에서 task specific identity 제한.

Task-Free Continual Learning (TFCL)

- 각 task는 서로 disjoint한 data label space를 가짐.
- task specific identity 학습 단계에서 제한, 인퍼런스 단계에서 optionally available.

Online Continual Learning (OCL)

- 각 task는 서로 disjoint한 data label space를 가짐.
- 학습 시 각 task의 모든 샘플은 one-pass로 최적화에 사용됨 (한번 최적화에 사용된 데이터 배치는 다시 input으로 사용되지 않음). 따라서 굳이 task를 구분하지 않더라도 모든 배치는 single-stream framework를 따르기 때문에 task에 무관한 학습이 됨.
- task specific identity 인퍼런스 단계에서 optionally available.

Blurred Boundary Continual Learning (BBCL)

- task의 경계선이 애매(blurry)하고, data label space가 일부 겹침.
- 인퍼런스 단계에서 task specific identity 제한.

Continual Pre-training (CPT)

- 사전 학습되는 data가 continual (sequence)로 도착하는 상황.
- 궁극적 목적은 downstream task의 성능을 올리는 것.
- pre-train 단계를 continual한 상황으로 가정했기 때문에 testing 단계에서는 단일 downstream task를 기준으로 삼게 됨. 따라서 downstream task의 task label은 굳이 요구되지 않음.

Continual Learning에서의 Evaluation metric

- continual learning이 가지는 목적은 '이전 representation을 얼마나 잘 유지하면서 현재 task의 성능을 잘 올릴 수 있는가?'로 정리할 수 있음.

① 기존(old) task들의 성능 평가 : Overall performance

- 기존 task들의 성능 평가는 보통 average accuracy(AA)나 average incremental average (AIA)로 나타냄.
  * AA : 여러 태스크들을 학습한 후, 각 태스크에 대한 평균 정확도를 계산.
  * AIA : 새로운 태스크를 학습할 때마다, 이전에 학습된 모든 태스크들에 대한 평균 정확도를 계산.
- AA와 AIA를 계산할 때, 모델이 각 태스크에 대해 예측을 수행할 때 사용하는 output space를 고려해야함. (CIL의 경우, 모델은 현재까지 학습된 모든 클래스에 대해 예측 수행. TIL의 경우, 모델은 각 태스크에 대해 별도의 output space를 가질 수 있음.

② 기존(old) task에 대한 memory stability : Memory stability

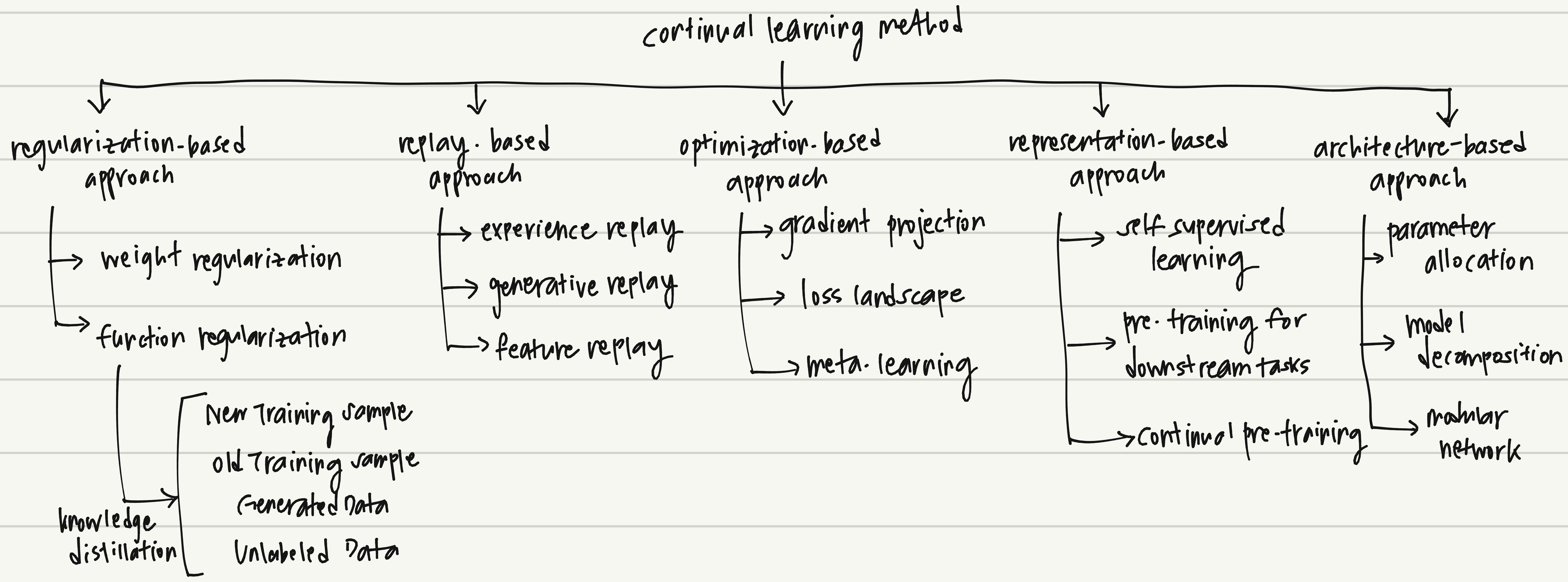
- 기존 task를 얼마나 잘 기억하고 있는지 측정하는 기표로는 보통 forgetting measure (FM)이나 backward transfer (BWT)를 사용.
  * FM : 각 task에 대한 forgetting 정도를 과거에 획득한 성능 중 최대치와 현재 성과의 차이로 계산.
  * BWT : k번째 task가 이전 k-1번째 task까지의 성능에 주는 영향을 평균내어 계산.
- overall performance는 성능 자체를, memory stability는 성능 변화를 기록.

③ 새로운(New) task의 성능 평가 : Learning plasticity

- 새로운 task를 얼마나 빠르게 학습하는지에 대한 지표가 되는 learning plasticity는 일반적으로 instransience measure (IM) 그리고 forward transfer (FWT)로 측정.
  * IM : 네트워크가 새로운 task를 잘 배우지 못하는 정도를 의미. joint training performance와 continual learning performance의 차이.
    • joint training performance : 기준이 되는 모델을 k번째 task까지의 데이터로 jointly(한꺼번에) 학습시켰을 때의 성능.
  * FWT : 새로운 k번째 태스크를 학습할 때 이전에 학습된 태스크들이 새로운 태스크에 미치는 긍정적인 영향을 측정하는 방법. k번째 태스크를 학습한 후의 성능과 기존 성능 사이의 차이를 계산하여 측정.

Continual Learning Methods

- 결국 continual learning의 성능을 재는 방식은 k번째 task 자체에 대한 performance와, performance 변화 그리고 실제 continual learning 방식의 효과성에 대한 입증으로 구성됨.



Regularization based approach

- 일반적으로 생각해볼 수 있는 방법은 바로 old task와 new task 간의 균형을 위해 explicit한 regularization term을 더해주는 것.
- 정규화가 consistency하게 보려고 하는 위치에 따라 weight regularization과 function regularization으로 구분됨.
  * 정규화를 위해서 old task에 대해 학습된 frozen model을 reference로 가지고 있어야 한다는 점은 같음.

[Weight Regularization]

- 네트워크 파라미터의 변화를 선택적으로 정규화하는 방법.
- 일반적인 구현은 loss function에 각 parameter의 변화에 대한 quadratic penalty를 추가한다던지, 그게 아니라면 각 parameter의 중요도 (contribution / importance)에 따라 penalty를 주는 방법이 있음.
- 고차원의 뉴럴 네트워크를 다룰 때 ground truth가 되는 각 latent modality의 distribution을 직접 알 수 없다는 점이 real-world에 대한 근본적인 어려움으로 작용함. 하지만 만약 log-likelihood를 전체 dimension에 대해 구하지 않고 라플라스 근사를 하게 되면, 적어도 해당 분포의 local point에서의 분포 형태를 가우시안 분포로 구할 수 있음. 그렇게 되면 결국 log-likelihood를 가우시안 근사 했을 때 분포의 curvature 정보를 얻게 되는데, 이를 토대로 중요한 parameter와 중요하지 않은 parameter를 구분할 수 있는 척도가 됨. (EWC | overcoming catastrophic forgetting in neural networks 논문)

- Online EWC (Progress& Compress: A scalable framework for continual learning) 에서는 다수의 태스크를 학습하는 과정에서 각 태스크에 대한 중요도를 누적하고, 이를 통해 모델이 가중치을 업데이트 할 때 중요한 가중치를 최소화으로 보존함. (task agnostic하게 연속적으로 데이터 학습)

  * EWC vs Online EWC: 기존 EWC는 오로지 순차적으로 구분된 태스크들 사이의 연속 학습에 초점을 맞춤.

- Syraptic Intelligence (Continual Learning Through Syraptic Intelligence): 각 파라미터 단위로 얼마나 중요도는 loss function에 대해 parameter의 기여도를 구한 값을 catastrophic forgetting을 방지하기 위해 parameter 변화의 trajectory로 정규화 함.

- 이러한 방법들로 모두 weight parameter에 대해 quadratic penalty term을 간접적으로 사용한다는 점에서 공통점을 가짐.

- penalty term을 사용하지 않고 factorized rotation을 기반으로 parameter space를 FIM(Fisher Information Matrix)에 diagonalize 하는 방법이나 (Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting 논문), Batch Normalization이 포함된 구조에 유리한 Kronecker-Factored Approximate Curvature (Continual Learning with Extended kronecker-factored Approximate Curvature 논문)이 제안되기도 했음. 하지만 이전 방법들은 모두 "이전에 학습된 old task parameter"를 기준으로 한다는 점에서 parameter 변화를 막는다는 공통적인 constraint를 가지고, 이는 둘 사이로는 task에 adaptation이 적용되는 과정에서 보수적인 효과를 가져옴.

- 이러한 문제점을 해결하고자 expansion 및 renormalization 방법이 제안되기도 했고, 이는 new task solution을 독립적으로 obtain한 뒤에, 이를 old model에 재배치하는 형태로 구현됨.

- IMM (Incremental Moment Matching / Overcoming Catastrophic Forgetting by Incremental Moment Matching 논문)이 초기 approach에 있는 논문인데, old task와 new task 간의 moment matching을 통해 점진적으로 새로운 task의 representation을 추가해나가는 관점을 취함.

  * moment matching: 확률 분포간의 유사성을 측정하고 조정하는데 사용됨.

- 이후 논문으로는 ResCL (IMM에 추가로 combination coefficient를 학습 가능하게 바꾼것 Residual Continual Learning 논문), Online EWC가 있음. 추가 network에 학습된 task를 기존 network에 distillation 하는 과정인데, 이때 weight consolidation을 formulation 하는 과정에서 online EWC를 사용함.

  * combination coefficient: 모델이 여러 태스크나 데이터셋을 학습하면서 각각에 대한 중요도를 개발하는데, 이러한 다양한 중요도를 균형있게나 통합하는데 사용되는 계수.

  * weight consolidation을 formulation 하는 것: 신경망의 파라미터를 조정하고 고정하여, 새로운 태스크를 학습하는동안 이전 태스크들의 지식을 유지하는 것.

- AFEC (Active Forgetting of Negative Transfer in Continual Learning 논문)의 경우는 forgetting rate(모델이 새로운 task를 받아들일 때 이전 정보와 conflict되는 정도를 active하게 조정)를 제안함. 이는 새로운 knowledge가 transfer되는 과정에서 잠재적으로 기존 representation과 충돌했을때 발생하는 negative transfer 문제를 다루고자 함.

  * negative transfer: 새로운 task를 학습하는 과정에서 이전에 학습된 지식이 오히려 방해가 되어 학습 성능이 저하되는 현상.

- Towards Better Plasticity-Stability Trade-off in Incremental Learning - A simple Linear Connector 논문 : plasticity 및 stability 간의 trade-off (old task와 new task loss 사이의 trade-off) 균형을 위해 low-error path 간의 linear connector를 구성한 방법.

  * low-error path : old task와 new task 각각에 대해 낮은 loss를 보이는 신경망의 파라미터 집합.

  * linear connector : 두 태스크 간의 가중치 공간에서 직선 경로를 형성하며, 이 경로를 따라 이동하면서 새로운 태스크를 학습하고 이전 태스크의 성능을 유지할 수 있음.

- Overcoming Catastrophic Forgetting by Neuron-level Plasticity Control 논문: parameter의 변화 자체를 규정하기보다 learning rate를 줄이는 방법 제안.

- penalty를 통해 간접적으로 weight update를 막는 방법 대신 important neuron의 학습을 멈춤으로써 (freeze) hard regularization을 채택한 방법들도 존재함.

[function regularization]

- prediction function의 최종 output 혹은 그 중간의 output을 기준으로 정규화를 진행하는 방법. 모델의 출력이 일관성(consistency)를 유지해야함.

- knowledge distillation 방법으로 훈련하는데, teacher 모델은 기존의 지식을 바탕으로 일관된 출력을 제공하고, student 모델은 이를 참고해 새로운 태스크를 학습함.

- 대표적인 논문은 Learning without Forgetting.

Replay based approach

- 새로운 task를 학습할 때 old data distribution을 recover하고 approximate 하는 방법들도 생각해 볼 수 있는데, 각각 replay 하는 주체에 따라 3가지로 구분됨.

[experience replay]

- 작은 memory buffer를 가지고 여기에 약간의 이전 task에 대한 training sample을 저장하는 형태로 구현됨.
  * reservoir sampling: 가장 간단하게 sample selection 과정마다 고정된 갯수만큼 old sample을 유지하고 나머지를 랜덤하게 대체하는 방식.
  * ring buffer sampling: reservoir sampling을 발전시켜서 old training sample을 새로이 저장하는 과정마다 class마다 같은 샘플 수가 유지되도록.
  * mean-of-feature sampling: 각 class의 특징자 벡터의 평균을 일종의 prototype으로 간주하여 이와 가장 closest(유사한) 샘플을 저장하는 방식을 사용.
  * gradient based sample selection: gradient 기준으로 다양성을 최대화하는 방법.
  * Coresets via Bilevel Optimization for Continual Learning and Streaming 논문: entity 간의 관계 조건인 cardinality constraint를 task performance와 연관짓는 방법
    - (entity: 데이터 샘플)
    - (cardinality constraint: 샘플들 간의 관계 또는 상호 작용을 나타내는 제약 조건)
    - 샘플의 선정이 모델의 전반적인 성능, 특히 새로운 태스크를 학습할 때 이전 태스크의 정보를 얼마나 잘 보존하는지에 대한 영향을 고려함.
  * batch 단위로 유사성을 보는 방법.
  * 최적화가 가능한 방법으로 latent decision boundary를 조정하는 등 고정되지 않은 알고리즘으로서 정의된 approach가 발전하기 시작.
  * 동시에 샘플 저장 효율성과 관련된 방법도 동시에 발전하기 시작했는데, vector quantize 방법을 포함하여 sample point를 압축하여 저장하기도 했으며, augmentation을 사용하여 한정된 샘플에 다양성을 부여하기도 함.
    - vector quantization: 연속적인 입력 데이터를 유한한 수의 벡터로 표현.
  * 의도적으로 forgetting boundary (잊기 쉬운 샘플들)을 adversarial하게 합성해 replay 효과를 높이는 방법도 존재.

=> 굳이 샘플을 저장하지 말고 feature를 저장하면 안되는가?

[feature replay]

- experience replay에 비해 용량 측면에서 이점이 있으나. feature extractor가 업데이트되면서 같은 샘플이 매번 다른 representation이 갖게되는 문제 발생. 이는 곧 feature level에서의 replay가 catastrophic forgetting 위험성을 내포한다는 뜻.

- 문제 해결방법:

  * old model과 new model 간의 feature distillation.

  * experience replay를 기준으로 feature representation (포컨이나 포즈변화 등과 같은 통계)을 복구하는 방법.

  * old sample의 representation을 저장해두고 이를 업데이트하면서 distribution shift를 예측하는 방법. (REP 논문)

  * 초기 layer를 고정시켜 중간 feature을 추출하여 이를 해당 layer를 업데이트하는 과정의 replay sample로 사용하는 방법.

[generative replay]

- old task의 학습 데이터를 생성 모델을 사용하여 replay에 사용하는 전략.

  * ex) 새로운 task에 대해 생성된 데이터와 old task에 대해 생성된 데이터 간의 consistency를 사용하는 방법.

optimization based approach

- regularization 방법이나 replay를 통해 기존에 학습된 데이터를 implicit하게 혹은 explicit하게 활용할 수 있는 방법.

- 여러 "기준" 아래를 키워드에서 보면서 explicit하게 최적화 구조를 바꾸거나 고정하는 방법이 소개되었으며, 이를 곧 optimization based approach라는 큰 틀로 묶을 수 있음.

- 이 접근법을 모델의 파라미터 업데이트 과정을 조정하여 새로운 태스크의 학습이 이전 태스크의 기억에 부정적인 영향을 최소화하도록 함.

- 가장 흔한 아이디어는 gradient projection. (새로운 태스크를 학습할 때 발생하는 그래디언트 업데이트가 이전 태스크의 학습 결과를 해치지 않도록 하는 전략. 파라미터 업데이트 방향을 조정해 기존 태스크의 기억이 유지되도록 함)

  * replay-based approach의 변형: 일부 replay-based approach는 experience replay의 업데이트 방향을 기준으로 파라미터 업데이트를 조정.

  * OWM (Orthogonal Weight Modification): 파라미터 업데이트를 이전 input space에 orthogonal한 방향으로 수행. (기존 태스크의 입력 공간 유지)

  * OGD (Orthogonal Gradient Descent): 이전 태스크에서의 파라미터 업데이트를 보존하고, 새로운 태스크의 그래디언트 최적화 방향을 이전 태스크의 그래디언트와 수직인 방향으로 조정. (그래디언트 공간 유지 전략)

  * Bayesian weight regularization과 gradient projection의 결합 : Bayesian 정규화는 모델이 가중치의 불확실성을 고려하면서 데이터에 대해 일반화를 잘 할 수 있도록 돕고. 그래디언트 사영은 새로운 태스크의 학습이 이전 태스크의 성능에 부정적인 영향을 미치지 않도록 함.

- 여러 task에 대해 robust한 optimization이라고 하면 meta-learning도 떠올릴 수 있음.

  * meta learning : 학습 알고리즘 자체를 최적화. "학습하는 방법을 학습하는 것"

  * OML (Online Meta Learning): 메타러닝 학습 프레임워크를 제안함으로써 연속적으로 제공되는 샘플 input에 대해 학습 과정에서의 간섭을 줄이면서 online update에 도움이 되는 방법을 제안.

  * ANML (A Neuromodulated Meta-Learning Algorithm): 점진적으로 증가하는 task에 대해, context-dependent 정보를 활용하여 특정 뉴런을 활성화하는 방식으로 학습 진행.

- generalization 관점 : loss landscape을 안정적으로 만드는 것. task 및 domain 관점에서 loss landscape가 안정적인 형태(curvature가 낮은 . flat 한 local minima)를 가질수록 adaptation에 도움이 된다는 관점에서 출발.

 *이러한 loss landscape을 고려하는 방법들은 대부분 optimizer에 관한 연구로 구성되며 SGD나 Adam optimizer에 추가적인 제공함.

representation based approach

- generalization을 위해 meta learning, self-supervised learning, large scale pre-training과 같은 방법을 사용하여 continual learing에서 각 task에 특정하지 않고 넓은 범위의 학습 분포를 커버하는 robust한 representation을 개발하는 전략.

- SSL을 사용한 방법들 :

 * SSL approach를 사용한 논문들의 공통점은 "SSL로 학습된 representation" 일수록 catastrophic forgetting에 강인하다는 점.

 * LUMP : old task와 new task 간의 representation이 연속이라는 가정 하에 interpolation을 진행하는 방식 사용.

 * MindRed : 저장된 old training sample을 사용하여 replay experience를 기존 representation으로부터 decorrelate 시키는 방법 사용.

 * self. supervised loss를 distillation 방법론과 결합 : 현재의 representation을 이전 state로 mapping하거나 모델 간의 mapping을 하는 전략.

- pre-training을 도입한 방법들 :

 * 주로 continual learning 과정에서 얻는 representation의 아닌 사전 학습된 representation을 활용하여 여러 downstream continual setting에서 좋은 성능을 내고자 하는 것이 목적.

 * 대규모의 데이터셋에 대해 학습된다든가, 보다 큰 파라미터 수를 가지는 larger model에 대해 학습된다든가, 아니면 contrastive loss 등등 SSL 전략들과 함께 사용되었을 때 좋은 성과를 보임.

  · contrastive loss : 모델이 데이터 샘플 간의 유사성을 학습하도록 하는 손실 함수.

 *이러한 방법론에서 중요한건 사전 학습된 representation을 continual 환경에서 어떻게 잘 전달하느냐.

  ① 사전학습된 backbone 파라미터가 고정된 경우

   · Side-Tuning : parameter-efficient tuning 전략. 기존 parameter와 parallel 하게 학습될 수 있는 방법.

   · TwF (Transfer without Forgetting) : backbone의 knowledge를 레이어별로 distillation 하는 전략.

  ② 사전학습된 backbone 파라미터가 고정되지 않은 경우

   · 생성 모델의 representation의 각 layer로부터 task-specific 파라미터를 학습하는 구조. (고정될 수도 있고, 안될 수도 있음)

   · Adapter를 기반으로 사전 학습된 transformer를 tuning하는 방법.

    기존의 대규모 사전 학습된 모델에 소규모의 추가적인 네트워크나 레이어를 삽입하여 특정 태스크에 대해 모델을 fine-tuning 하는 방법.

architecture based approach

- 위의 방법들은 대부분 공유된 parameter(단일 model을 하나의 파라미터 단위로 생각했을때)를 여러 task에서 잘 활용하는 방법에 대한 해결책.

- 하지만 이렇게 단일 파라미터를 여러 task에서 employ 할 경우 근본적으로 interference 문제를 해결하기 어렵다는 단점이 있음.

- 기존 방법들과 달리 task-specific parameter를 공유되는 parameter와 독립적으로 학습함으로써 간섭 문제를 해결하고자 한 방법이 바로 architecture based approach.

[parameter allocation]

- 네트워크 전반에 걸쳐 각 task에 기여하는 parameter 부분집합 (subspace)을 분리. 이때 architecture는 고정되어있는듯 있고, dynamic하게 변하는듯 있음.

- 고정된 네트워크 구조를 차용한 경우 학습하고자 하는 뉴런이나 파라미터를 각 task마다 선택해주는 binary mask를 explicit하게 학습하는 형태.

  * binary mask: 네트워크의 각 파라미터에 대해 '0' 또는 '1'의 값을 가지는 마스크.

    '0': 사용 X. '1': 특정 태스크에 사용 O.

- 중요한 뉴런이나 파라미터 가지를 identify하는 방법들이 사용되기도 함. (iterative pruning, activation value, uncertainty estimation 등)

- 네트워크의 용량 한계로 인해 발생하는 성능 저하 문제를 해결하기 위해 네트워크 아키텍처를 dynamic하게 확장하는 방법들이 제안되었는데, 이는 특히 새로운 태스크를 효과적으로 학습하기 위해 기존 네트워크 파라미터의 용량이 부족할 때 유용한 전략.

[model decomposition]

- model을 task-sharing과 task-specific 성분으로 구분하는 approach.

- task-specific component를 학습 가능한 network로 가정하는 게 일반적임.

- task-specific components를 구성하는 구조:

  * ACL: parallel branches

  * GVCL: adaptive layer

    * 각각 feature map에 대한 mask를 생성하는 generator를 고안하는 방법.

[modular network]

- parallel한 sub-network나 sub-module을 기본 네트워크 구조 상에 제안한 형태.

- progressive Network: 각 task마다 별도의 sub-network를 추가하는 방식 제안. adaptor connection으로부터 새로운 sub-network까지의 knowledge transfer를 수행하는 방법 제안.

- 여러 parallel branch를 통해 candidate path를 설계하고 최적의 경로를 선택하는 방법도 있음.

- 여러 sub-network 간의 앙상블을 사용하기도 함.