Neural Contextual Bandits with UCB-based Exploration

1. Introduction

1) 문제 정의

- 매 라운드마다 **여러 개의 선택지(행동)** 가 주어짐.

  예: "어떤 광고를 보여줄까?", "어떤 영화를 추천할까?"

- 각 선택지에는 **특징 정보(Feature vector)** 가 붙어 있음.

  예: "이 광고는 20대 여성 타겟", "이 영화는 로맨스+코미디 장르"

- 선택하면, 그에 따른 **보상(Reward)** 을 받음.

  예: 광고 클릭 여부(0/1), 추천 영화 시청 시간 등

목표: 앞으로 계속 선택하면서 **누적 보상(Reward)을 최대화**하는 것!

2) 기존 접근 : “선형모델” 가정

지금까지는 대부분 **보상이 ‘특징 벡터의 선형 결합’**이라고 가정했었음.

$$\text{보상} = \theta^T x$$

즉, 각 특징이 일정한 가중치로 보상에 기여한다는 것.

예를 들자면

"코미디 장르면 +0.3점, 로맨스면 +0.5점, 액션이면 -0.2점" 이런 식으로 단순히 더하는 모델.

이런 선형 가정은 수학적으로 다루기 쉽고, 실제로도 어느 정도 잘 맞음.

3) 문제점: 현실은 선형이 아니다

- 실제 사람의 취향이나 행동 패턴은 단순 선형으로 설명되지 않음.

  예를 들어,

  - "코미디도 좋아하고 로맨스도 좋아하지만, **코미디+로맨스 조합**일 때는 더 좋아한다" → **비선형**
  - "액션은 평소엔 싫어하지만, 특정 상황(밤+주말)에서는 좋아한다" → **복잡한 조건부 패턴**

👉 즉, 현실의 보상 함수는 훨씬 더 **복잡하고 비선형적임.**

4) 기존의 비선형 연구들

연구자들이 선형 가정의 한계를 넘으려고 여러 방법을 시도했는데, 다 조건이 붙음.

- **GLM (일반화 선형모델)**: 보상 함수가 로지스틱 같은 특정 함수꼴을 따른다고 가정.
- **Lipschitz 연속성**: "입력(feature)이 조금 변하면, 보상도 조금만 변한다"라는 매끄러움 가정.
- **RKHS (커널 공간)**: 보상 함수가 특정 수학적 함수 공간 안에 있다고 가정.

👉 요약하면, "비선형을 허용하긴 하지만, 그래도 보상 함수에 강한 제약을 둬야 가능하다"는 한계가 있었음.

5) 이 논문의 출발점

- "현실에서 보상은 복잡하다.
- 기존 비선형 방법들은 너무 많은 수학적 제약을 둔다.
- 그럼 딥러닝의 **표현력**을 이용하면 제약 없이 더 일반적으로 풀 수 있지 않을까?"

→ 그래서 **NeuralUCB** 라는 딥러닝 기반 밴딧 알고리즘을 제안하게 된 것.

지금까지 연구:

- "선형 모델 = 단순하지만 현실에 안 맞음"
- "비선형 모델 = 현실에 맞지만 수학적 제약이 많음"

이 논문:

- "딥러닝 + UCB = 제약은 줄이고, 현실성은 높이자!"

6) Neural Contextual Bandits

- **딥러닝(DNN)** 을 사용해서 **보상 함수를 학습하거나 특징을 추출**하는 밴딧 알고리즘.
- 기존 선형 가정의 한계를 극복하고 **비선형 보상 함수**에도 잘 작동하게 만드는 것이 목적
- 이런 방법들을 묶어서 **Neural Contextual Bandit 알고리즘**이라고 부름.

Neural-Linear Bandits

- Neural Contextual Bandit 알고리즘 중 하나
- 방식:
  1. DNN의 **마지막 레이어 전까지**를 사용해, 원래 입력(Context)을 **더 나은 표현 (Feature space)** 으로 변환.

     → 즉, "딥러닝을 고급 특징 추출기"처럼 사용.
  2. 변환된 특징 위에 **마지막 레이어는 선형 모델**을 얹어, **탐색(Exploration) 정책**을 학습.

     → 마지막 레이어는 자주 업데이트, 앞쪽 레이어는 덜 자주 업데이트.
- 장점:
  - 표현력이 좋아서 실제 실험에서 성과가 뛰어났음.
- 한계:
  - **이론적 보장(후회 보장)** 이 없음.
  - 즉, 실험은 잘 되지만 "항상 좋은 성능을 낸다"는 증명이 없었음.

▼ 이론적 보장(후회보장)

$$\text{후회} = (\text{최적행동보상}) - (\text{내가선택한보상})$$

- 좋은 알고리즘은 시간이 지날수록 점점 학습을 잘 해서 **후회가 너무 크게 쌓이지 않도록** 해야 함.
- 연구자들은 보통 "이 알고리즘을 썼을 때, 후회가 얼마나 커질 수 있는지 상한선(=bound)을 수학적으로 보장"함.
  - 예: 선형 UCB의 경우 $\text{후회} = O(\sqrt{T})$라는 걸 증명할 수 있음.
  - 이 말은, 라운드 T가 아무리 커져도, 후회는 최대 $\sqrt{T}$ 정도 수준으로만 늘어난다는 뜻.
  - 즉, 시간이 지날수록 "기회 손실 비율"이 점점 줄어든다는 것을 **이론적으로 보장**해 줌.

→ **이론적 보장까지 제공하는** 신경망 기반 밴딧 알고리즘을 제안 : NeuralUCB

7) Neural UCB

NeuralUCB의 핵심 아이디어

1. **보상 함수 학습**: DNN을 이용해 보상 함수를 근사.
2. **탐색 전략**: Upper Confidence Bound (UCB) 기반으로 행동 선택.

   → “기대 보상 + 불확실성”을 고려해 탐색/활용 균형을 잡음.
3. **핵심:**
   - DNN 기반 **Random Feature Mapping**을 사용해 **UCB를 계산**.
   - 즉, 신경망이 만들어낸 특징 공간에서 "보상의 불확실성 구간"을 계산하는 새로운 방식.

이론적 분석

- 최근 DNN 이론 발전(Neural Tangent Kernel, Optimization, Generalization theory 등)을 기반으로 **후회 분석(Regret Analysis)** 를 진행.
- 결과: 보상 함수에 대해 **어떠한 모형 가정도 하지 않음** (bounded만 요구).
- 즉, "보상이 특정 모델에 속한다" 같은 제약 없이, 일반적으로 작동하는 이론 보장을 제시.

> 기존 Neural-Linear Bandit은 **성능은 좋았지만 수학적 보장이 없었음**.
>
> NeuralUCB는 DNN과 UCB를 결합해, **비선형 보상 함수에서도 작동하고, 후회 보장까지 제공하는 최초의 알고리즘**이라는 점이 차별점.

8) main contributions

1. 새로운 알고리즘 제안

- **NeuralUCB**라는 **신경망 기반(Contextual) 밴딧 알고리즘**을 제안
- 이 알고리즘은 기존의 **(일반화된) 선형 밴딧 알고리즘**을 확장한 버전
  - 기존 선형 밴딧은 "보상 함수가 선형"이라는 가정을 했음.
  - 하지만 NeuralUCB는 **보상 함수가 어떤 형태든 상관없이(단, bounded → 값이 무한히 커지지 않음)** 적용 가능.

👉 즉, **선형 밴딧의 장점을 가져오되, 보상 함수가 선형일 필요는 없다**는 점이 핵심.

2. 이론적 보장 (Regret Bound)

- **이 알고리즘이 얼마나 효율적으로 학습하는지**를 **후회(bound)** 로 분석함.
- 후회가 $\sqrt{T}$정도로만 늘어난다 → **시간이 지나도 학습 효율이 떨어지지 않고 안정적으로 탐색/활용을 해낸다**.

3. 실험적 검증

- 단순히 이론만 제시한 게 아니라,
- **합성 데이터(시뮬레이션)** 와 **실제 벤치마크 데이터**에서 실험도 해봤음.
- 결과: NeuralUCB가 여러 대표적인 알고리즘들보다 좋은 성능을 보였음.

2. Problem Setting

1) 문제 정의: Stochastic K-armed Contextual Bandit

- 매 라운드 $t \in [T]$마다:
  - 에이전트는 **K개의 행동**후보를 봄.
  - 각 행동 $a \in [K]$마다 **특징 벡터(feature vector)** $x_{t,a} \in \mathbb{R}^d$가 주어짐.

    → 즉, "각 행동이 어떤 성격인지"를 숫자로 표현한 것.
  - 에이전트는 그 중 하나 a_t를 선택하고, 보상 r_{t,a_t}를 얻음.

👉 목표: 전체 T번 라운드 동안 **누적 보상**을 최대화하기.

이를 위해 **후회(regret)** 라는 척도를 정의함.

2) 후회(Regret)의 정의

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T}\left(r_{t,a_t^*} - r_{t,a_t}\right)\right]$$

- $a_t^* = \arg\max_{a\in[K]} \mathbb{E}[r_{t,a}]$: 라운드 t에서 가장 기대 보상이 큰 "최적 행동".
- 실제 내가 고른 a_t와 비교해서, **얼마나 손해를 봤는지(기회 손실)** 를 합산한 값.

👉 목표는 이 **후회**를 최대한 줄이는 것.

3) 보상 생성 모델

보상은 이렇게 생성된다고 가정함:

$$tr_{t,a_t} = h(x_{t,a_t}) + \xi_t$$

- $h(\cdot)$: **알 수 없는 함수** (reward function).
  - 단, 범위는 제한: 0≤h(x)≤10(bounded assumption).
  - 즉, 보상이 무한히 커지거나 음수가 될 수는 없음.
- ξ_t: 잡음(noise).
  - 조건부로 평균은 0 (E[ξt]=0).
  - $\nu - sub - Gaussian$ → 잡음이 확률적으로 너무 크게 튀지 않는다는 조건.
  - 예: ±1 사이에서만 나오는 bounded noise라면 충분히 만족.

즉, "보상은 **알 수 없는 함수 h + 잡음** 으로 모델링.

4) NeuralUCB에서의 보상 함수 학습

- 보상 함수 h(x) 는 **모르지만 bounded**라고만 가정했음.
- 이를 근사하기 위해 **깊은 신경망(DNN)** 을 사용함.

신경망 구조:

$$f(x;\theta) = \sqrt{m}W_L\sigma(W_{L-1}\sigma(\cdots\sigma(W_1x)))$$

- **Fully Connected Neural Network (FCNN)**, 깊이 L≥2
- 각 hidden layer의 너비는 m으로 동일.
- 활성화 함수는 **ReLU** (σ(x)=max(0,x).
- 파라미터 전체는 θ, 차원은 p
- 👉 즉, "보상 함수 h(x)≈ 신경망 f(x;θ)로 학습.

5) Gradient 특징 (NTK 연결고리)

$$g(x;\theta) = \nabla_\theta f(x;\theta) \in \mathbb{R}^p$$

→ 입력 x에 대해 신경망 출력이 파라미터 θ에 얼마나 민감한지를 나타내는 벡터.

이 g(x;θ)가 바로 뒤에서 **랜덤 피처(random feature mapping)** 로 쓰여서,

"이 입력을 본 적이 얼마나 있는가 → 불확실성 크기" 를 계산하는 데 활용됨.

3. The NeuralUCB Algorithm

1) 기본 아이디어

NeuralUCB는 두 가지를 합친 알고리즘:

1. **신경망** → "이 행동을 선택하면 보상이 얼마일까?" 예측
2. **UCB(Upper Confidence Bound)** → "이 예측이 얼마나 확실한가?" 계산

👉 이렇게 해서, **예측값 + 불확실성 보너스**를 점수로 만들어 가장 좋은 행동을 선택함.

2) 알고리즘 1 (NeuralUCB)

(1) 초기화

- 신경망을 랜덤으로 설정.
- 불확실성을 계산하는 데 필요한 값들도 초기화.

(2) 라운드마다 반복

1. **상황(컨텍스트) 관찰**
    - 예: "사용자에게 추천할 영화 후보 3개 있음" → 각 영화의 정보(장르, 배우, 시간 등)를 특징 벡터로 받음.
2. **모든 행동에 점수 매기기**
    - 신경망이 각 영화의 "예상 보상"을 예측.
    - 동시에 "이 영화에 대한 예측이 얼마나 확실한가?"를 계산.
    - 최종 점수 = **예상 보상 + 불확실성 보너스**
3. **행동 선택**
    - 점수가 가장 높은 영화를 선택해서 추천.
    - 이렇게 하면 이미 잘 아는 장르(확실히 보상이 큰 것)도 고르고,

      아직 잘 모르는 장르(불확실성이 큰 것)도 가끔 시도하게 됨.

4. **보상 관찰**
    - 실제 사용자가 영화를 봤는지(=보상 1), 안 봤는지(=보상 0)를 확인.
5. **학습 업데이트**
    - 지금까지 얻은 데이터를 가지고 신경망을 조금 더 훈련시킴.
    - 불확실성 계산도 갱신해서, 다음 라운드에서 더 똑똑하게 결정할 수 있도록 함.

Algorithm 1 NeuralUCB

Input: Number of rounds T, regularization parameter λ, exploration parameter ν, confidence parameter δ, norm parameter S, step size η, number of gradient descent steps J, network width m, network depth L.
Initialization: Randomly initialize $\boldsymbol{\theta}_0$ as described in the text
Initialize $\mathbf{Z}_0 = \lambda \mathbf{I}$
for $t = 1, \ldots, T$ **do**
  Observe $\{\mathbf{x}_{t,a}\}_{a=1}^{K}$
  for $a = 1, \ldots, K$ **do**
    Compute $U_{t,a} = f(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1}) + \gamma_{t-1} \sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})/m}$
    Let $a_t = \operatorname{argmax}_{a \in [K]} U_{t,a}$
  end for
  Play a_t and observe reward r_{t,a_t}
  Compute $\mathbf{Z}_t = \mathbf{Z}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1}) \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})^\top / m$
  Let $\boldsymbol{\theta}_t = \text{TrainNN}(\lambda, \eta, J, m, \{\mathbf{x}_{i,a_i}\}_{i=1}^{t}, \{r_{i,a_i}\}_{i=1}^{t}, \boldsymbol{\theta}_0)$
  Compute

$$\gamma_t = \sqrt{1 + C_1 m^{-1/6} \sqrt{\log m} L^4 t^{7/6} \lambda^{-7/6}} \cdot \left(\nu \sqrt{\log \frac{\det \mathbf{Z}_t}{\det \lambda \mathbf{I}} + C_2 m^{-1/6} \sqrt{\log m} L^4 t^{5/3} \lambda^{-1/6} - 2 \log \delta} + \sqrt{\lambda} S \right)$$
$$+ (\lambda + C_3 t L) \left[(1 - \eta m \lambda)^{J/2} \sqrt{t/\lambda} + m^{-1/6} \sqrt{\log m} L^{7/2} t^{5/3} \lambda^{-5/3} (1 + \sqrt{t/\lambda}) \right].$$

end for

3) 알고리즘 2 (TrainNN)

- NeuralUCB는 매 라운드마다 신경망이 예측한 보상을 업데이트해야 합니다.
- 그때 쓰이는 과정이 **TrainNN**
- 역할: 신경망 학습 (보상 함수 근사)
- 손실 함수:

$$\mathcal{L}(\theta) = \sum_{i=1}^{t} \left(f(x_{i,a_i}; \theta) - r_{i,a_i} \right)^2 / 2 + m\lambda \|\theta - \theta^{(0)}\|_2^2 / 2$$

- 첫 항: 예측 보상과 실제 보상 차이 최소화.

- 둘째 항: 정규화(regularization) → 과적합 방지.
- 방법: Gradient Descent (경사하강법)으로 J번 반복해서 θ 업데이트.
- 결과: 새로운 신경망 파라미터

4) 비교

- **NeuralBandit (예전 방법)**
  - 행동마다 신경망 하나씩 둠 → 계산이 무겁고 탐색은 단순히 무작위(ε-greedy).
- **NeuralLinear**
  - 신경망 앞부분은 특징 뽑기 전용, 마지막은 선형 모델 → 이론적 보장은 부족.
- **NeuralUCB**
  - 신경망 전체를 활용해서 예측,
  - UCB로 불확실성까지 계산,
  - 네트워크도 하나만 쓰니까 효율적이고,
  - 게다가 후회(regret)에 대한 이론적 보장까지 확보.

예시) 영화 추천예시

1. **초기화**: 랜덤한 "취향 예측 신경망"을 하나 세팅.
2. **추천 후보**: 사용자가 들어오면 "액션, 로맨스, 코미디" 영화 벡터가 들어옴.
3. **점수 계산**:
   - 액션: "예상 보상 0.6 + 불확실성 보너스 0.2 = 0.8"
   - 로맨스: "예상 보상 0.7 + 불확실성 0.1 = 0.8"
   - 코미디: "예상 보상 0.5 + 불확실성 0.3 = 0.8"

     → 셋 다 0.8이면, 세 장르를 균형 있게 시도해가며 학습.
4. **추천 후 보상 받기**: 예를 들어 로맨스를 추천했는데, 실제 클릭=1.
5. **업데이트**: 신경망과 불확실성 행렬 갱신

5. Proof of Main Result

- NeuralUCB의 누적 후회(Regret):

$$R_T = \tilde{O}\left(\sqrt{T \cdot d_e}\right)$$

- 여기서 d_e(effective dimension)은 데이터 구조(컨텍스트의 다양성, NTK 행렬의 랭크)에 따라 결정되는 값.
- 즉, 신경망 파라미터 수와 무관하게, **데이터의 '진짜 난이도'만큼만 후회가 커진다**는 결론.
- **T가 커져도** 후회가 **선형으로** 늘지 않고 **천천히(√T)** 늘어요. → 장기적으로 점점 최적 행동에 가까워짐.
- 복잡도는 **신경망 파라미터 수(p)** 가 아니라 **ded_ede** 에만 달림. → "모델이 크다"는 이유만으로 후회가 나빠지지 않음.
- NeuralUCB는 **복잡한 DNN을 사용하면서도, 선형 UCB에 가까운 후회 성능 보장**을 제공.

▼ d_e(effective dimension)

데이터가 보여주는 **실질 패턴의 개수**.

- 비슷한 컨텍스트가 반복되면 **d_e 작음**(배울 게 적다).
- 매번 완전히 다른 컨텍스트가 오면 **d_e 큼**(배울 게 많다).

NTK(Neural Tangent Kernel) 행렬의 고유값 분포에서 결정됨.

- **상위 몇 개 고유값만 크고 나머지는 빠르게 작아진다면** "중요 패턴이 소수" → d_e 작음.
- **고유값이 널찍하게 퍼져 있으면** 많은 패턴을 따로 학습해야 함 → d_e 큼.

6. Related Work

1) 선형 밴딧 (Linear Bandits)

- **아이디어**: 보상이 **특징벡터와 선형관계**라고 가정.
- **탐색 방법**: 대부분 **UCB (Upper Confidence Bound)** 기반 → "예상 보상 + 불확실성"으로 행동 선택.
- **성과**: 이론적으로 최적에 가까운 후회(√T 규모)를 달성.
- NeuralUCB도 UCB 방식을 쓰기 때문에, 선형 보상이라는 특수 케이스에선 기존 선형 밴딧과 같은 결과로 귀결됨.

2) 비선형 보상 관련

기존 연구들은 선형 가정을 벗어나기 위해 여러 접근을 시도

(a) Generalized Linear Bandits

- 보상 = 선형함수 + 비선형 링크함수(예: 로지스틱 함수).
- NeuralUCB는 이보다 더 일반적 (보상에 특정 함수 형태 가정 X).

(b) Expert Learning 접근

- 각 전문가(가설)를 행동 후보로 두고 학습.
- 하지만 전문가 수가 많으면 계산량이 **지수적으로 커짐** → 실용적 한계.

(c) Supervised Learning으로 환원

- 예: Epoch-Greedy (Langford & Zhang, 2008).
- 지도학습 문제로 바꿔 처리하지만, 후회가 O(T^2/3)으로 아쉽게 큼.
- 개선된 방법들(Agarwal et al., 2014)은 더 나은 후회 달성했지만, **특정 오라클 가정** 필요.

(d) 비모수적 접근 (Non-parametric)

- 퍼셉트론, 랜덤포레스트, Gaussian Process, 커널 기반 방법.
- 특히 Valko et al. (2013): **RKHS 내 보상함수** 가정을 두고 UCB 사용 → $\tilde{O}(\sqrt{Td_e})$ 보장.
- 단점: 어떤 커널/메트릭을 고를지가 매우 중요하고, 계산량이 커질 수 있음.

👉 NeuralUCB는 커널을 직접 고를 필요 없이 **DNN이 자동으로 좋은 표현을 학습**하므로 확장성과 효율성 ↑.

3) 최신 연구

- Foster & Rakhlin (2020): 회귀 오라클을 이용한 밴딧 → $O(T^{3/4})$ 후회 (시간 T에 더 불리).
- NeuralUCB: $\tilde{O}(d_e\sqrt{T})$후회로, **시간에 대한 스케일링은 더 낫다**.

4) 딥러닝 이론과 연결

- 최근 DNN의 표현력 연구 (깊이·너비가 어떻게 표현력을 키우는지).
- 최적화 측면: SGD/gradient descent가 **전역 최적점(global minima)** 을 찾을 수 있음을 수학적으로 증명한 연구들.
- 일반화 측면: DNN의 학습 결과가 **NTK(Neural Tangent Kernel) 공간**의 함수로 근사 가능하다는 결과.

NeuralUCB는 이런 최신 DNN 이론을 바탕으로 후회 보장을 증명한 첫 사례 중 하나.

7. Experiments

1) 목적

- 제안한 **NeuralUCB**의 성능을 **다양한 기존 알고리즘**과 비교.
- 비교 기준: **누적 후회 (Cumulative Regret)** → 작을수록 좋음.

2) 비교대상

- **LinUCB**
  - 선형 보상 가정 + UCB 탐색.
  - 전통적인 밴딧 알고리즘.
- **GLMUCB** (Filippi et al., 2010)
  - 보상을 “선형 함수 + 비선형 링크함수” 형태로 가정.
  - Generalized Linear Model(GLM) 기반.
- **KernelUCB** (Valko et al., 2013)
  - 커널 함수를 정의해, 비선형 보상을 다룸.
  - 단점: 커널을 잘 골라야 하고, 계산량 ↑.

- **BootstrappedNN** (Riquelme et al., 2018)
  - 여러 개의 NN을 **부트스트랩 샘플**로 동시에 학습.
  - 매 라운드, 무작위로 하나의 모델을 선택해 행동 결정.
- **Neural ε-Greedy**
  - NeuralUCB에서 UCB 탐색 대신 **ε-greedy**를 사용.
  - 즉, 확률 ε만큼은 랜덤 탐색, 나머지는 현재 최적 선택.
- **NeuralUCB0**
  - NeuralUCB의 단순화 버전.
  - 신경망을 선형 근사(Neural Tangent Kernel)로 고정, Online Ridge Regression으로 업데이트.
- **Neural ε-Greedy0**
  - NeuralUCB0에서 탐색 전략만 **ε-greedy**로 교체한 버전.

7-1. Synthetic Datasets

- 단순화된 인공 데이터 환경에서 **NeuralUCB와 기존 알고리즘**을 비교.
- 인위적으로 만든 **비선형 보상 함수**에서 누가 더 잘 학습/탐색하는지 확인

1) 실험 환경

- **행동(arms)**: 4개 (예: 추천할 아이템이 4개라고 생각).
- **컨텍스트(context)**: 각 행동은 20차원짜리 특징 벡터로 표현됨 (예: 사용자의 상태나 아이템의 속성).
- **라운드 수**: 10,000번 → 매 라운드마다 어떤 행동을 선택할지 정하고 보상을 관찰.

2) 보상 함수 (Reward Function)

보상은 “이 행동이 얼마나 좋은지”를 나타냄.

일부러 **비선형 함수** 3개를 만듦:

1. **이차식 (quadratic)**

$$h_1(x) = 10(x^\top a)^2$$

   → 어떤 방향 a로 잘 정렬된 행동일수록 보상이 커짐.

2. **행렬 기반 이차식**

$$h_2(x) = x^\top A^\top A x$$

   → 특정 패턴이 있는 입력일수록 높은 보상.

3. **코사인 함수 (비선형, 주기적)**

$$h_3(x) = \cos(3x^\top a)$$

   → 입력 방향에 따라 보상이 주기적으로 변함.

즉, **선형이 아닌 보상 함수**를 일부러 설정해서, **비선형 표현력이 있는 알고리즘**이 더 잘하는지 테스트.

3) 비교할 알고리즘

NeuralUCB 성능을 **7가지 대표 알고리즘**과 비교

- **LinUCB** → 보상을 선형으로 가정 (한계: 실제 보상은 비선형)
- **GLMUCB** → 선형 + 시그모이드 link function
- **KernelUCB** → 커널 방법으로 비선형 처리 가능, 하지만 계산량이 큼
- **BootstrappedNN** → 여러 신경망을 학습시키고 무작위 선택으로 탐색
- **Neural ε-Greedy** → NeuralUCB 구조 + ε-탐욕적 탐색 (무작위성만으로 탐색)
- **NeuralUCB0** → NeuralUCB의 단순화 버전 (Neural Tangent Kernel 활용)
- **NeuralUCB** (제안 알고리즘) → 전체 DNN을 사용 + UCB 기반 탐색

4) 실험 방법

- 각 알고리즘이 같은 환경에서 10,000번 선택을 반복.
- 결과 비교 기준: **누적 후회(cumulative regret)** → 최적 행동과 비교했을 때 잃은 보상.
  - 값이 낮을수록 잘한 것.

7-2. Real-world Datasets

- **사용한 데이터셋**
  - UCI 데이터셋: **covertype, magic, statlog**
  - 컴퓨터 비전 데이터셋: **MNIST**
  - 각 데이터셋의 클래스 수(K)를 그대로 밴딧의 arms로 사용
  - (예: covertype은 7개 클래스 → 7개의 행동)
- **분류 문제 → 밴딧 문제 변환 방법**
  - 원래는 "분류 정확도"를 측정하는 문제.
  - 이를 밴딧 문제로 바꿔서:
    - **정답 클래스(action)를 고르면 regret=0**
    - **틀린 action을 고르면 regret=1**
  - 즉, 분류 성능이 밴딧 누적 후회(cumulative regret)와 직결됨.
- **실험 조건**
  - 라운드 수: **T = 15,000**
  - 각 데이터셋마다 컨텍스트 순서를 섞어서(random shuffle) 10번 반복 → 평균 결과 보고
- **알고리즘 비교**
  - **기존 방법:** LinUCB, GLMUCB, KernelUCB
  - **신경망 기반 방법:** BootstrappedNN, NeuralUCB, NeuralUCB0, Neural ε-Greedy, Neural ε-Greedy0

- **구체적 세팅**
  - NeuralUCB / NeuralUCB0 → 2층 신경망, 너비 m=100
  - Zt 행렬 계산은 **전체가 아니라 대각선(diagonal)만 근사** (계산량 줄이기 위해)
  - 학습 업데이트:
    - NeuralUCB, Neural ε-Greedy, BootstrappedNN → 2000 라운드 이후부터 100 라운드마다 한 번씩 업데이트
  - 최적화: SGD, batch size=500, step 수 J=1000
  - 파라미터들은 Section 7.1과 동일하게 grid search로 최적값 선택

7.3 Results

1) 문제 의식

- **Contextual Bandit** 문제에서 보상 함수가 비선형일 경우,

  기존 **선형/커널 기반 방법**(LinUCB, KernelUCB 등)은 표현력 한계 때문에 잘 작동하지 않음.
- 기존 Neural Contextual Bandit 연구들(NeuralLinear, BootstrappedNN, Neural ε-Greedy 등)은

  성능은 좋지만 이론적 보장(Regret Bound)이 없거나 불완전했음.

2) Neural UCB

- DNN(Deep Neural Network)으로 보상 함수를 근사.
- **UCB(Upper Confidence Bound) 탐색 전략**을 사용 → "예상 보상 + 불확실성"을 고려해 행동 선택.
- **핵심 아이디어**:
  - DNN의 gradient를 활용한 **Random Feature Mapping**을 통해 **불확실성 추정(UCB 계산)**.
  - 단순한 ε-greedy보다 더 효율적인 탐색 보장.
  - 전체 DNN을 활용하여 NTK 기반의 단순화 모델보다 표현력이 뛰어남

- y축 = **누적 후회(regret)**, 낮을수록 좋음.
- x축 = **라운드 수**.
- 기울기(상승 속도)가 완만할수록 장기적으로 더 잘 배우는 알고리즘.

3) Synthetic Data 결론

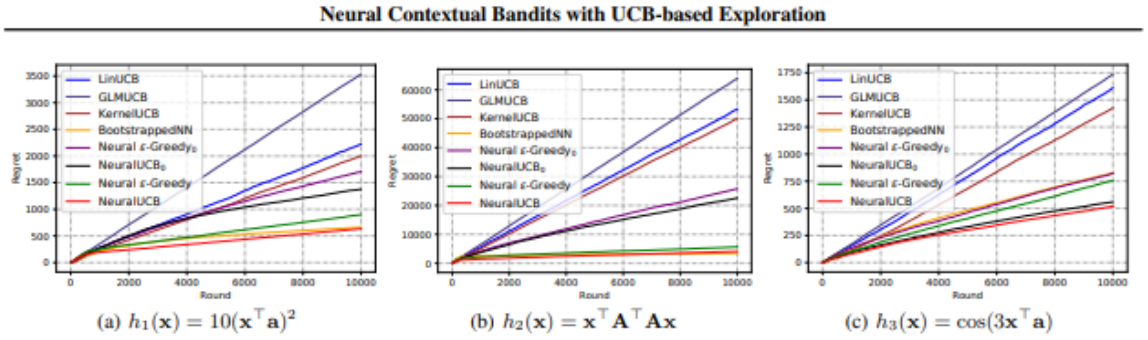

Figure 1. Comparison of NeuralUCB and baseline algorithms on synthetic datasets.

세 가지 비선형 보상 h1,h2,h3h_1, h_2, h_3h1,h2,h3 모두에서:

- **NeuralUCB**: 가장 낮은 후회(가장 평평한 곡선).
  - DNN이 비선형 보상을 잘 근사하고, **UCB**가 "불확실한데 유망한" 영역만 골라서 **낭비 없는 탐색**을 하기 때문.
- **Neural ε-Greedy**: NeuralUCB보다 위(성능↓).
  - ε-greedy는 **무작위 탐색**이라 이미 나쁜 행동에도 종종 샷을 낭비한다. UCB처럼 **표적 탐색**이 아님.
- **NeuralUCB0 / Neural ε-Greedy0**: 두 모델보다 한 단계 더 위(성능 더↓).
  - **NTK 고정 특징**만 쓰면 표현력이 **초기 주변**에 묶여서, 데이터가 요구하는 **적응적 표현 학습**이 부족하다.
- **KernelUCB**: RBF 커널로 비선형을 잡긴 하지만 NeuralUCB보단 높음.
  - 이유: 커널이 **사전 고정 설계**라 유연성/표현력이 DNN보다 제한됨(커널 미스펙 가능).

- **GLMUCB**: 간단한 링크함수(GLM)만으로는 복잡한 비선형을 못 따라가서 제한적.
- **LinUCB**: 보상이 비선형이라 거의 모든 경우에 **가장 나쁨**.

4) Real-world Data 결론

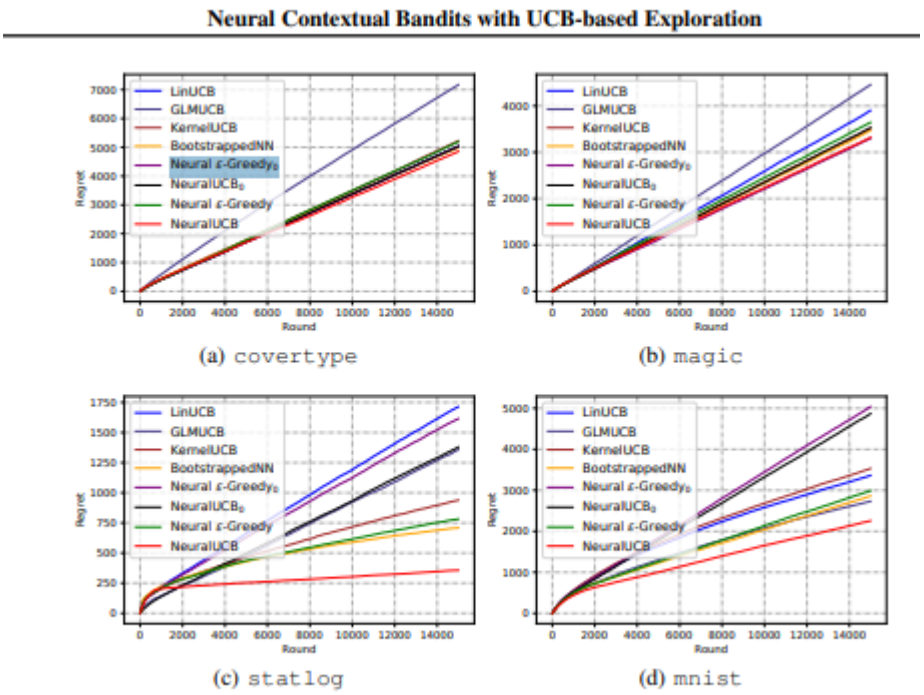

Figure 2. Comparison of NeuralUCB and baseline algorithms on real-world datasets.

(covertype, magic, statlog, mnist 모든 데이터셋 공통)

- **NeuralUCB**가 일관적으로 **최저 후회**.
- **Neural ε-Greedy**가 그다음—표현력은 있지만 탐색이 비효율적이라 차이가 남.
- **KernelUCB**는 데이터에 따라 괜찮을 때도 있으나 전반적으로 NeuralUCB에 뒤짐(커널 한계).
- **BootstrappedNN**은 일부 데이터셋에서 **NeuralUCB에 근접**하기도 함.
  - 대신 **여러 네트워크를 동시에 학습/유지**해야 해서 **계산비용이 큼**(대규모 문제에서 부담).
- **NeuralUCB0/ε-Greedy0**는 고정 특징 한계로 뒤처짐.

- **LinUCB/GLMUCB**는 비선형/고차원 데이터에서 급격히 불리.

요약: **현실 데이터에서도 NeuralUCB가 가장 안정적이고 강함**(낮은 기울기).

- **NeuralUCB가 전반적으로 가장 낮은 후회**를 보임.
- **UCB 탐색 > ε-greedy 탐색**.
- **딥러닝 전체 사용 > NTK 고정 특징**.
- **신경망 > 단순 커널**.
- **BootstrappedNN도 강력하지만 비용 큼**.

8.conclusion

1) 제안

- 본 논문에서는 **NeuralUCB**라는 새로운 알고리즘을 제안.
- 이는 **신경망 + UCB(Upper Confidence Bound) 탐색**을 결합한 **stochastic contextual bandit 알고리즘**임.

2) 이론적 결과

- 최근 DNN의 **최적화(optimization)** 및 **일반화(generalization)** 이론을 바탕으로 분석 수행.
- 임의의 **bounded reward function**에 대해

  → **$\tilde{O}(d_e\sqrt{T})$ regret bound** 달성함을 증명.

  (여기서 d_e는 NTK 기반 효과적 차원, T는 라운드 수)

3) 실험적 결과

- **Synthetic 데이터**와 **Real-world 데이터**에서 실험 수행.
- 결과: 이론적 보장과 일관되며, **실제 적용에서도 유망한 성능**을 보여줌.

4) 향후 연구방향

- 이번 연구는 **UCB 기반 탐색**에 초점을 맞췄음.
- 하지만 실무에서 효과적인 **무작위(randomized) 기반 탐색 전략**에 대한 **이론적 보장 연구**는 아직 부족.

  → 특히, 기존 이론은 얕은 모델(linear, generalized linear)에 국한.

  → 이를 **DNN으로 확장하는 것**이 중요한 연구 과제.
- 또한, 본 연구는 **Neural Tangent Kernel(NTK) 이론**에 의존해 분석.
  - NTK는 유용하지만 한계가 존재.
  - 따라서 **NTK를 넘어선 NeuralUCB 분석**이 남은 과제임.