

A kriptográfia és gépi tanulás kapcsolódásai

Egyéni kutatómunka 2.

Mikulás Zsófia

Témavezető: Lukács András

1 Bevezetés

A gépi tanulást (ML) és a mélytanulást (DL) egyre több területen alkalmazzák, például számítógépes látásban, természetes nyelvfeldolgozásban, beszédfelismerésben. Az ML modellek jó teljesítménye és hatékonysága általában nagy mennyiségű tanulási adat és erős számítástechnikai erőforrások segítségével érhető el. Azonban a nagymennyiségű adatfelhasználás komoly adatvédelmi aggályokat vet fel, mivel a személyes és érzékeny információk kiszivárgásának kockázata is megnőhet. Ezen kívül a fejlődő jogszabályi környezetek, például a GDPR és a CCPA egyre szigorúbb korlátozásokat alkalmaznak az érzékeny adatok elérhetőségére és felhasználására, ami komoly kihívásokat jelent az ML rendszerek számára.

Ezért kulcsfontosságú a magánélet-védő gépi tanulás (Privacy-Preserving Machine Learning, PPML) kidolgozása, például az anonimizálási mechanizmusok használata és új adatvédelmi módszerek használása. A PPML megoldások célja, hogy biztosítsák az adatvédelmet anélkül, hogy csökkentenék az ML modellek teljesítményét. A különböző PPML megoldások különböző adatvédelmi garanciákat kínálnak, és az alkalmazott megoldások technikai hatékonyságát is fontos figyelembe venni. Az olyan módszerek, mint a homomorfikus titkosítás, a federált tanulás és a differenciális adatvédelem, egyre fontosabb szerepet kapnak ezen megoldásokban.

A kutatómunka során három fontos megoldást használó modellt implementáltam: homomorfikus titkosítást, federált tanulást és differenciális adatvédelmet. Mindezek különböző megközelítéseket kínálnak, és mindegyik más és más módon járul hozzá a privát adatkezeléshez és a modellek hatékonyságához.

2 Gépi tanulásról röviden

A gépi tanulási rendszerek két fázisa általában a következő: tréning (adatgyűjtés, előkészítés, előfeldolgozás, modell tanítása és értékelés) és szolgáltatás (a betanított modell használatára összpontosít, például a modell telepítésére és egy adott adatmintán történő következtetésre).

Formálisan, adott egy mintahalmaz a tréningelésre: $(x_1, y_1), \dots, (x_n, y_n)$, ahol $x_i \in \mathbf{R}^m, y_i \in \mathbf{R}$. Az egyszerűség kedvéért tegyük fel, hogy egy lineáris modellről van szó, tehát a modell tréningezésének célja, hogy megtanulja a megfelelő függvényt, amelyet a következőképpen jelölünk:

$$f_{w,b}(x) = w^T x + b$$

ahol $w \in \mathbf{R}^m$ a modell paramétereinek halmaza, és b az eltolás. A modell tréningezéséhez a cél a következő hibafüggvény minimalizálása:

$$E(w, b) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) + \alpha R(w),$$

ahol L a veszteségfüggvény, $R(w)$ a regularizációs kifejezés, és α egy hyperparaméter, amely szabályozza a regularizáció erősségét. A sztochasztikus gradiens csökkenés (SGD) segítségével frissítjük a paramétereket:

$$w \leftarrow w - \eta \nabla_w E = w - \eta [\alpha \nabla_w R + \nabla_w L]$$

$$b \leftarrow b - \eta \nabla_b E = b - \eta [\alpha \nabla_b R + \nabla_b L],$$

ahol η a tanulási sebesség, amely szabályozza a lépés nagyságát a paraméterek térében.

A szolgáltatás fázisában a tréningelt modellt használjuk új adatpontok előrejelzésére, a következőképpen:

$$\hat{y} = f_{w_{trained}, b_{trained}}(x).$$

A szolgáltatás tehát az SGD tréningelés egy egyszerűsített verziója, mivel nem igényel gradiens frissítéseket és súlyok módosítását. Az adatvédelmet megőrző tréningezés összetettebb feladat, mint a szolgáltatás, mivel több adatvédelmi kihívást jelent a modell tanítása során.

3 PPML

A PPML munkafolyamat az ML fázisainak megfelelően az alábbiakat tartalmazza: adatvédelmet megőrző adat előkészítése, modell tréningezése és értékelése, modell telepítés és modell következtetés.

A legtöbb adatvédelmet megőrző modell generálási megoldás arra helyezi a hangsúlyt, hogy a privát információk ne szivároogjanak ki a tréningadatokból. A kutatások két fő kérdést fogalmazznak meg: (i) hogyan lehet a tréningadatokat úgy szűrni, hogy eltávolítsuk a privát információkat, (ii) hogyan lehet adatvédelmi szempontból megfelelően feldolgozni az adatokat. A meglévő adatvédelmi megoldások különböző technikákat alkalmaznak, mint például k -anonimitás, l -diverzifikáció, a t -diszkrécio és differenciális adatvédelem. Ezek az adatokat úgy kezelik, hogy az egyéneket nem lehet visszamenőlegesen azonosítani, miközben az adatok hasznosak maradnak. Az adatvédelmet megőrző adatkezelési megoldások célja, hogy eltávolítsák az érzékeny információkat és biztosítsák az egyének védelmét

a lekérdezések során. A titkosításon alapuló megoldások, mint a homomorfikus titkosítás és a funkcionális titkosítás, erősebb adatvédelmet biztosítanak, mivel lehetővé teszik az adatok titkosítva történő feldolgozását.

Egy adatbázis D adatsorai akkor **k-anonim**, ha minden adatsornak legalább k másik adatsora van, amely ugyanazzal az értékkel rendelkezik a lehetséges azonosítók (pl. név, cím, születési dátum) tekintetében.

Egy adatbázis D adatsorai akkor **l-diverzifikáltak**, ha minden olyan k -anonim csoportban, ahol k adatsor található, legalább l különböző érzékeny információval rendelkeznek (pl. különböző diagnózisokkal vagy betegségekkel), ezzel biztosítva, hogy az adatok védettek maradjanak.

A **t-diszkréció** azt biztosítja, hogy egy adatmintában minden olyan attribútum, amely érzékeny információkat tartalmaz (pl. betegség, jövedelem, politikai hovatartozás), a lehetséges értékek közül legalább t -féle különböző értéket vegyen fel.

Az adatvédelmi garanciák fő típusai:

- **Adat-orientált adatvédelmi garancia:** célja, hogy megakadályozza a privát információk kiszivárgását közvetlenül az adatbázisból. Hátránya, hogy csökkentheti az adatok hasznosságát.
- **Modell-orientált adatvédelmi garancia:** a privát információk védelmét a betanított modell szintjén biztosítja. Célja, hogy megakadályozza a modell lekérdezései révén történő adatlopást.
- **Munkafolyamat-orientált adatvédelmi garancia:** az egész gépi tanulási munkafolyamat adatvédelmét biztosítja, figyelembe véve az adatfeldolgozási határokat. A bizalom mértéke meghatározza, hogy a helyi vagy harmadik féltől származó erőforrásokat használják-e.
- **Teljes adatvédelmi garancia:** minden lépésben biztosítja az adatvédelmet, beleértve az adatfeldolgozást és a modell generálást. Ez biztosítja, hogy az adatvédelmi szabályok minden fázisban érvényesüljenek.

A PPML megoldások célja, hogy biztosítsák az adatvédelem fenntartását anélkül, hogy kompromisszumot kellene kötni a gépi tanulás teljesítményével.

A gépi tanulás és a kriptográfia hatékony együttműködése érdekében az adverzariális robusztusság, azaz a támadásokkal szembeni ellenálló képesség integrálása is fontos szerepet kap. Az adverzariális neurális hálózatok (GAN-ek) használata lehetővé teszi a titkosítási rendszerek védelmét azáltal, hogy olyan modelleket hoznak létre, amelyek ellenállnak a támadásoknak. Ezáltal a kriptográfiai rendszerek erősebbek és biztonságosabbak lesznek, miközben megőrzik a titkosított adatokat és védelmet biztosítanak a külső támadásokkal szemben.

A gépi tanulás alkalmazása a kriptográfiában nemcsak a titkosítási sémák fejlesztésére korlátozódik, hanem a titkosítási mechanizmusok biztonságának javítására is. A neurális hálózatok segítségével új titkosítási rendszerek, például

szimmetrikus titkosítást alkalmazó modellek jöttek létre. Azonban a hálózati architektúrák biztonsága még mindig kérdéses, különösen akkor, ha az architektúrák nyilvánosságra kerülnek. Az újabb kutatások célja az adatok védelme és a modellek biztonságos titkosítása, hogy azok ne kerülhessenek illetéktelen kezekbe.

4 Homomorfikus titkosítás

A titkosításon alapuló megoldások, mint például a homomorfikus titkosítás, egyre népszerűbbek, mivel erősebb adatvédelmet kínálnak, mint a hagyományos módszerek. A **homomorfikus titkosítás (HE)** egy nyilvános kulcsú kriptoszisztéma, amely lehetővé teszi számítások végrehajtását titkosított adatokon, anélkül, hogy hozzáférnénk a privát titkos kulcshoz. A számítások eredményei továbbra is titkosított formában maradnak, és a visszafejtett eredmény megegyezik az eredeti, titkosítatlan adatokon végrehajtott műveletek eredményével. A HE típusai:

- **Részleges homomorfikus titkosítás (PHE):** Csak bizonyos matematikai műveleteket lehet elvégezni titkosított adatokon.
- **Bizonyos mértékben homomorfikus titkosítás (SHE):** Többféle műveletet is végre lehet hajtani titkosított adatokon, de nem mindent.
- **Teljes homomorfikus titkosítás (FHE):** Lehetővé teszi bármilyen számítás elvégzését titkosított adatokon, miközben megőrzi az adatokat titkosítva.

A HE általános definíciója négy fő algoritmust tartalmaz:

$$(pk, sk) \leftarrow E_{HE}.KGen(1^\lambda)$$

ahol $KGen$ a kulcsgeneráló algoritmus, pk a nyilvános kulcs és sk a privát kulcs.

A titkosított adatokat a következőképpen kódoljuk:

$$CHE \leftarrow \{E_{HE}.Enc_{pk}(m_i)\}_{i \in \{1, \dots, n\}}$$

ahol m_i az üzenetek, amelyek titkosítva lesznek.

A műveletek a titkosított adatokon történnek az értékelési algoritmus segítségével:

$$C_f^{HE} \leftarrow E_{HE}.Eval_{pk}(f, CHE)$$

Ez az algoritmus végrehajtja a f függvényt a titkosított adatokon. A végső eredmény visszafejtése:

$$f(m_1, \dots, m_n) \leftarrow E_{HE}.Dec_{sk}(C_f^{HE})$$

ahol $f(m_1, \dots, m_n)$ az eredeti üzenet, amit visszafejtettek.

4.1 Homomorfikus titkosítás (HE) implementációja

A homomorfikus titkosításhoz a *Tenseal* könyvtárat használtam, amely a CKKS séma alapján biztosít titkosított adatfeldolgozást:

```
1 context = ts.context(ts.SCHEME_TYPE.CKKS,
2 poly_modulus_degree=8192,
3 coeff_mod_bit_sizes=[60, 40, 40, 60])
4 context.generate_galois_keys()
5 encrypted_X_test = [ts.ckks_vector(context,
6 row.tolist()) for row in X_test]
```

A *Logistic Regression* modellek mind titkosított, mind titkosítatlan adatokkal történő tanítása és értékelése megtörtént. A titkosított adatokra történő előrejelzés a következőképpen történik: a titkosított bemenetekkel először számítjuk ki az osztályokhoz tartozó pontszámokat, majd hozzáadjuk a modell torzítóját (intercept).

```
1 encrypted_predictions = []
2 for encrypted_row in encrypted_X_test:
3     encrypted_scores =
4         [encrypted_row.dot(weights[i]) +
5          intercepts[i] for i in
6          range(weights.shape[0])]
7     encrypted_predictions.append(encrypted_scores)
```

A titkosított előrejelzéseket a következő módon dekódoljuk: a titkosított pontszámokat visszafejtjük, és a legmagasabb pontszámú osztály kerül kiválasztásra:

```
1 decrypted_predictions =
2     [np.array([score.decrypt()[0] for score in
3               scores]) for scores in encrypted_predictions]
4 class_predictions = [np.argmax(scores) for scores
5                      in decrypted_predictions]
```

A titkosított adatokkal végzett előrejelzés időigénye az alábbi kódrészlettel mérhető:

```
1 start_time_encrypted_inference = time.time()
2 # Perform inference on encrypted data
3 encrypted_predictions = []
4 for encrypted_row in encrypted_X_test:
5     encrypted_scores =
6         [encrypted_row.dot(weights[i]) +
7          intercepts[i] for i in
8          range(weights.shape[0])]
9     encrypted_predictions.append(encrypted_scores)
10 end_time_encrypted_inference = time.time()
```

```

encrypted_inference_time =
    end_time_encrypted_inference -
    start_time_encrypted_inference

```

Az eredmények azt mutatják, hogy a titkosított adatokkal végzett előrejelzés során a titkosítás, előrejelzés és visszafejtés teljes időigénye jelentősen megnöveli a folyamat időtartamát, azonban a pontosság nem csökkent.

A homomorfikus titkosítás hátránya, hogy a számítási költsége magasabb, és jelentős időbeli késleltetést eredményez, különösen nagy adatkészletek és komplex modellek esetén.

5 Federált Tanulás

A federált tanulás egy elosztott tanulási módszer, amely lehetővé teszi, hogy a modelleket helyi eszközökön képezzük, így az adatok nem hagyják el a felhasználó eszközeit. Ez különösen fontos adatvédelmi szempontból, mivel az érzékeny információk nem kerülnek központi szerverekre. Ezáltal biztosítható, hogy a felhasználók személyes adatai ne kerüljenek ki a helyi eszközökről, miközben a modellek a helyi eszközökön végrehajtott tanulás során javulnak. A federált tanulás egy optimalizációs problémát céloz, amely a helyi eszközökön végzett számításokat egyesíti a központi szerveren, anélkül hogy a nyers adatokat megosztanánk.

A matematikai optimalizációs folyamat az alábbiak szerint működik:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(\theta)$$

Ahol:

- θ a modell paramétereit jelenti,
- η a tanulási ráta (learning rate),
- $L(\theta)$ a veszteségfüggvény, amely a modell előrejelzése és a valós (helyes) kimenet közötti hibát méri.

Mivel a federált tanulás során minden kliens (helyi eszköz) külön tanul, a helyi eszközök a saját paraméterfrissítéseiket küldik vissza a központi szerverre. Az összegyűjtött frissítéseket az alábbi módon egyesítik:

$$\theta_{t+1} = \frac{1}{N} \sum_{i=1}^N \theta_{t+1}^{(i)}$$

Ahol:

- N a kliensek száma,
- $\theta_{t+1}^{(i)}$ az i -edik kliens által frissített paramétereket jelenti.

Federált tanulásban a kommunikációs költségek csökkentése érdekében két fontos módszert alkalmaznak: a strukturált frissítéseket és a skicelt frissítéseket. A strukturált frissítések során a frissítések egy kisebb paramétertömbből kerülnek meghatározásra, például alacsony rangú mátrixok vagy véletlenszerű maszkolás alkalmazásával, míg a skicelt frissítések esetén a teljes frissítést először tömörítik a küldés előtt. Tesztek szerint e módszerek használata két nagyságrenddel csökkentheti a kommunikációval szükséges adatok mennyiségét, miközben csak minimális mértékben csökkenti a konvergencia sebességét és a modellek pontosságát.

5.1 Federált tanulás (FL) implementációja

A *PyTorch*-t és a *MNIST* adatkészletet használtam a federált tanulás (FL) implementációjában. A kód alapvetően a *FedAvg* (Federated Averaging) algoritmuson alapul. Minden kliens egy saját, lokálisan tanuló modellt tartalmaz, amely az aggregált globális modell paramétereit kapja és végzi el a saját adatain tanulást.

```

1     client_updates = []
2     for client in self.clients:
3         update =
4             client.update(copy.deepcopy(self.global_model.state_dict()))
        client_updates.append(update)

```

Két modellt is betanítottam, a *ConvNet* és *SimpleCNN* modellek a különböző klienseknél futnak. A kliensek 5 lokális epochot futtatnak a saját adataikon, majd a globális modell állapotát frissítik a tanulás eredményeként. A globális modell összegzése és frissítése a következőképpen működik:

```

1     aggregated_state =
2         self.aggregate_models(client_updates)
3     self.global_model.load_state_dict(aggregated_state)

```

A modell aggregálásához a *FedAvg* algoritmus van implementálva, amely az összes kliens által küldött frissítéseket átlagolja. Az aggregálás után a globális modell új paraméterekkel frissül, amit aztán az összes kliens számára elérhetővé teszünk.

```

1     aggregated_state = copy.deepcopy(client_states[0])
2     for key in aggregated_state.keys():
3         aggregated_state[key] =
            torch.stack([states[key] for states in
                client_states]).mean(0)

```

A *Server* osztály felelős a kliens frissítések begyűjtéséért és a globális modell értékeléséért minden egyes kerekben. A modell globális értékelése az alábbi kódrészlettel történik:

```

1     for data, labels in self.test_loader:

```

```

2         data, labels = data.to(self.device),
           labels.to(self.device)
3         outputs = self.global_model(data)
4         loss = criterion(outputs, labels)
5         test_loss += loss.item()

```

6 Differenciális Adatvédelem

A differenciális adatvédelem (DP) egy módszer arra, hogy megakadályozza, hogy egy adatfeldolgozó rendszer túl sok információt áruljon el egyes adatok részleteiről. Például, ha egy támadó tudja, hogy egy adott személy szerepel az adatbázisban, a differenciális adatvédelem biztosítja, hogy a támadó ne tudja meg, hogy pontosan milyen adatokat tartalmaz a személyre vonatkozó bejegyzés. A differenciális adatvédelem mértékét az úgynevezett ϵ paraméter adja, amely meghatározza, hogy mennyire védett az adat, és a δ érték biztosítja, hogy az adatok védelme a valószínűség egy bizonyos szintjén maradjon.

Egy $M : D \rightarrow R$ mechanizmus, ahol D a bemeneti tartomány és R a kimeneti tartomány, akkor és csak akkor felel meg a (ϵ, δ) -differenciális adatvédelemnek, ha bármely két szomszédos bemeneti $d, d_0 \in D$ és bármely $S \subseteq R$ kimeneti halmaz esetén teljesül:

$$\Pr[M(d) \in S] \leq e^\epsilon \cdot \Pr[M(d_0) \in S] + \delta.$$

A hozzáadott δ lehetővé teszi, hogy az ϵ -differenciális adatvédelem megsérüljön egy δ valószínűséggel, ahol előnyös, ha δ kisebb, mint $1/|d|$. Ezen kívül a Rényi differenciális adatvédelem (Rényi DP) az alábbi feltételt alkalmazza:

$$D_\alpha(M(d)||M(d_0)) \leq \epsilon,$$

ahol D_α a Rényi divergencia, és $\alpha > 1$ egy paraméter, amely két valószínűségi eloszlás P és Q közötti eltérést mér.

A differenciális adatvédelemben gyakran alkalmazott "hozzáadott zaj mechanizmusok" közé tartozik a **Laplace mechanizmus** és a **Gauss mechanizmus**. Ezek az alábbiak szerint vannak definiálva:

$$M_{\text{Gauss}}(d; f, \epsilon, \delta) = f(d) + N(\mu, \sigma^2),$$

$$M_{\text{Lap}}(d; f, \epsilon) = f(d) + \text{Lap}(\mu, b),$$

ahol $N(\mu, \sigma^2)$ a Gauss-eloszlás, és $\text{Lap}(\mu, b)$ a Laplace-eloszlás, amelyek a mechanizmusok által hozzáadott zajként működnek, hogy biztosítsák a privát információk védelmét.

A differenciális adatvédelem két fő irányban alkalmazható a PPML megoldásokban: (i) a nyers adatbázison közvetlenül alkalmazva a zajmechanizmusokat az adatkiadás során, vagy (ii) az eredeti tréning módszert átalakítva egy differenciálisan privát tréning módszerre, hogy a betanított/kiadott modell ϵ -differenciális adatvédelmi garanciát nyújtson.

A differenciális adatvédelem alkalmazása nemcsak az adatkiadási problémák kezelésére korlátozódik, hanem a szintetikus adatok generálására is alkalmazható, valamint a generatív ellenálló hálózatokban (GAN) is. A különböző megközelítések és algoritmusok lehetővé teszik, hogy a gépi tanulási modellek továbbra is titokban tartsák az adatokat, miközben fenntartják a magas szintű adatvédelmet a különböző alkalmazásokban.

6.1 Differenciális adatvédelem (DP) implementációja

A *PyTorch*-t és az *Opacus* könyvtárat használtam, amely a különböző privát adatok kezelésére szolgál. A 'RDPAccountant' segítségével követhető a privát adatok védelme a tanulási folyamat során. A kód rögzíti a futás során előálló statisztikákat, mint a veszteség, pontosság, és a privát költség (epsilon). Például:

```
1 self.accountant = RDPAccountant()
2 self.get_privacy_spent =
  self.accountant.get_epsilon(delta=self.target_delta)
```

A *SecureMLPModel* egy egyszerű több réteges perceptron (MLP) modellt definiál. A *DPTrainer* osztály felelős a tanulásért, amely a *differenciált privát* technikákat alkalmazza: a gradiens normák vágása és a zaj hozzáadása. A tanulás során *vegyes pontosságú* (mixed-precision) tanulás történik. A *DPTrainer* osztály kódrészlete:

```
1 with autograd():
2     outputs = self.model(data)
3     loss = self.criterion(outputs, target)
```

A kód hozzáadja a zajt a gradienshez a tanulási folyamat során, biztosítva ezzel, hogy a modell ne tanuljon túl sokat egyetlen adatpontról. A *RDPAccountant* nyomon követi a privát költséget, és számolja a ϵ értéket, amely a magánélet védelmét jelzi. A zaj hozzáadása a gradienshez:

```
1 noise =
  self.privacy_engine.generate_noise(param.grad.shape,
  self.device)
2 param.grad += noise / self.batch_size
```

7 Kísérleti beállítás és eredmények

A három magánélet-védő gépi tanulási módszer teljesítményét különböző adatbázisokon is teszteltem.

- **Adat előkészítés:** Az adatokat előkészítettem és titkosítottam a három módszer alkalmazásával.
- **Modell tanítása:** A gépi tanulási modelleket (például logisztikus regressziót, neurális hálózatot) a titkosított adatokkal tanítottam.

- **Értékelés:** A titkosított modellek teljesítményét értékeltem a tesztadatokon.

A kísérletek során azt találtam, hogy a három módszer közül mindegyik sikeresen megoldja a privát adatkezelés problémáját, bár a számítási idő növekedett, különösen a homomorfikus titkosítás esetén. Azonban minden módszer biztosította a magánélet védelmét anélkül, hogy jelentős mértékben csökkentette volna a modellek teljesítményét.

7.1 Összegzés

A jövőbeli kutatások célja, hogy tovább javítsák a PPML rendszerek hatékonyságát és alkalmazhatóságát. A kutatási irányok között szerepel a különböző technikák kombinálása, a még nagyobb adatvédelmi szint elérésének érdekében, miközben a gépi tanulás modellek hatékonysága növekszik. A privát gépi tanulás alkalmazása nemcsak a technológiai fejlődést, hanem az adatkezelés etikai aspektusait is előtérbe helyezi, így hosszú távon segíthet az adatok biztonságosabb és felelősségteljesebb felhasználásában.

References

- [1] Runhua Xu, Nathalie Baracaldo, and James Joshi, *Privacy-Preserving Machine Learning: Methods, Challenges and Directions*, IBM Research - Almaden Research Center, San Jose, CA, United
- [2] Korn Sooksatra and Pablo Rivas, *A Review of Machine Learning and Cryptography Applications*
- [3] Ronald L. Rivest, *Cryptography and Machine Learning*
- [4] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Ananda Theertha Suresh, Dave Bacon, Peter Richtarik *Federated Learning: Strategies for Improving Communication Efficiency*