Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudományi és Információelméleti Tanszék

Gépi tanulás és kvantuminformatika

TDK DOLGOZAT

Készítette
Szabó Dániel

Konzulens
Dr. Friedl Katalin

2018.

Tartalomjegyzék

Kivonat

Napjainkban egyre gyakrabban hallani híreket a kvantuminformatika világából, például új kvantumszámítógépekről vagy kvantumkommunikációs áttörésekről. Ezekkel összefüggésben egyre fontosabb szerep juthat a közeljövőben a kvantumalgoritmusoknak. Szintén nagyon népszerű terület a gépi tanulás, amit rengeteg különböző célra használnak a sakkozógéptől az önvezető autóig.

A kvantuminformatika a kvantummechanikából ismert jelenségeken alapul, amely szerint a mikrorészecskék mozgása valószínűségi módon írható le. Ilyen részecskével valósítható meg egy kvantumbit: egyszerre van a két, jól megkülönböztethető klasszikus állapot (0 és 1) szuperpozíciójában, az egyikben p, a másikban $1 - p$ valószínűséggel. Ilyen módon egy n kvantumbites rendszer egyszerre 2^n állapotban van, ennek leírására (azaz az egyes állapotok valószínűségének eltárolására) klasszikus esetben 2^n-nel arányos darabszámú bit szükséges. Ettől lehet hatékonyabb bizonyos problémák megoldása kvantumszámítógéppel, mint klasszikusan, pl. prímtényezőkre bontás, rendezetlen halmazban keresés stb. [11, 15].

Gépi tanulást (és általában heurisztikus megoldásokat) akkor érdemes használni, ha nem ismert olyan egzakt algoritmus, ami hatékonyan megold egy problémát. Lényege, hogy a gép, ami megoldja a feladatot, „tanul” a korábban már látott esetekből. Ezt ellenőrzött tanulás esetén úgy érjük el, hogy ismert eredménnyel rendelkező tanító bemeneteket adunk bemenetként, amelyekből a gép egy modellt tud előállítani. Így ha olyan új bemeneteket kap, amelyeknek már nem ismert az eredménye, akkor erre a modell alapján becslést tud mondani. Elterjedt gépi tanulási módszerek például a neurális hálózatok és a döntési fák. Tipikusan gépi tanulással megoldott feladatok például a képfelismerés és a kéretlen levelek kiszűrése [22].

A dolgozat fő célja bizonyos gépi tanulási módszerek (döntési fák, ensemble módszerek/boosting) kvantum változatának bemutatása. Ez azért fontos, mert a széles körben használt gépi tanuló algoritmusok sok adattal dolgozhatnak, és a kvantuminformatikában rejlő párhuzamosítási képesség nagyban gyorsíthatja ezek futását egy jövőbeli kvantumszámítógépen. Ezenkívül az ismertetett megoldásokból kiindulva egy új bináris osztályozó algoritmus is született, amely megalkotásánál szempont volt a kvantumos megvalósíthatóság is. Ez az algoritmus is bemutatásra kerül a dolgozatban.

Abstract

Nowadays quantum computing is getting more and more publicity. For example, we can hear news about new quantum computers or breakthroughs in quantum communication. These may cause quantum algorithms to become more and more important in the near future. Machine learning is also a very popular field, and it is used in various fields, e.g. chess playing machines and self-driving cars.

Quantum computing is based on quantum mechanical effects which describe the movement of microparticles probabilistically. A quantum bit can be realized with a microparticle which can be in a superposition of 0 and 1 at a time with probability p and $1-p$, respectively. An n quantum bit system encodes 2^n states at a time, while in a classical computer the number of bits used to store the probabilities for each state is proportional to 2^n. That is why several problems can be solved more efficiently on a quantum computer than on a classical one. E.g. factorization, searching in unordered sets [11, 15].

Machine learning (and more generally, heuristic algorithms) are used when no efficient exact algorithm is known for solving a problem. We can talk about machine learning if the machine which solves the problem 'learns' from the cases already seen. In supervised learning we give samples as input, for which we know the result so that the machine can create a model. If we give new inputs without the results, the machine can predict the results based on this model. Popular machine learning methods include neural networks and decision trees. Some typical problems which are solved using machine learning are image recognition and spam filtering [22].

The main goal of this paper is to present the quantum version of some machine learning techniques (decision trees, ensemble methods/boosting). Motivation comes from the fact that the widespread machine learning algorithms may work with a huge amount of data, and the quantum parallelism can greatly increase the speed of these algorithms on a potential future quantum computer. A new binary classifier algorithm was also designed, keeping an eye on a possible realization as a quantum algorithm. This algorithm will also be presented in the paper.

Bevezető

A kvantuminformatika területe egyre népszerűbbé válik, ahogyan születnek az áttörésnek számító eredmények ezen a területen. 2017 nyarán például arról hallhattunk, hogy Kínában kvantumkommunikációs műholddal kísérleteztek sikeresen, nem sokkal később pedig megvalósult a világ első kvantumkulcsszétosztással titkosított videohívása [16]. Ennek az a jelentősége, hogy a kulcsszétosztás a szimmetrikus kulcsú kriptográfia alapvető lépése, és kvantumkriptográfiával matematikailag bizonyítottan biztonságos kulcsokat lehet előállítani a biztonságos kommunikációhoz.

Tavaly ősszel az IBM jelentette be, hogy 50 kvantumbites (qubites) kvantumszámítógépet építettek, idén tavasszal pedig a Google számolt be az új, 72 qubites chipjéről. Ilyen értékeknél már felmerül az ún. kvantumfölény elérése, hiszen eddig legfeljebb 49 qubites rendszert sikerült klasszikus számítógépen szimulálni az IBM kutatóinak. Azonban a qubitek stabilitásának hiánya miatt még nem beszélhetünk kvantumfölényről, mivel a qubitek állapotát maximum körülbelül 100 mikroszekundum ideig tudják fenntartani, és a zajra érzékenység miatt nehéz megmondani, hogy valóban jó eredményt kaptunk-e [7, 12].

A gépi tanulás szintén módfelett felkapott területnek számít, amelynek fő oka, hogy így sok olyan problémát lehet már az embereknél is hatékonyabban megoldani klasszikus gépekkel, amelyeknél ez korábban elképzelhetetlennek tűnt. Ehhez nagyban hozzájárult a számítási kapacitásnak az utóbbi évtizedekben tapasztalt hatalmas növekedése, mivel vannak olyan feladatok is, amelyeket elméletben már régebben megoldottak, azonban az akkori gépek használhatatlanul lassan adtak eredményt (pl. sakkozógépek). Mára lehetővé vált olyan komplex feladatok megoldása (pl. képek és más adatok gyors feldolgozása), amelyek szükségesek voltak például az önvezető autók megjelenéséhez.

Döntési fákkal nagyon egyszerűen modellezhetők választási lehetőségekből és azok következményeiből álló folyamatok. Ezért népszerű a döntéstámogatásban, a legjobb stratégia kiválasztásában, de gépi tanulásra is gyakran használják.

Az ensemble módszerek lényege, hogy több gépi tanuló algoritmust használnak egyszerre, ezáltal érve el nagyobb hibatűrést, mint az egyes algoritmusok külön-külön. Ilyen algoritmus például az AdaBoost, amely „gyenge” osztályozó algoritmusok felhasználásával azoknál jobb eredményt ér el. Az algoritmus egy változata (Viola-Jones [21]) elterjedten használt kamerákban arcdetekcióra.

A fentiekből is látható, hogy a gépi tanulás és a kvantuminformatika együttes alkalmazása hatalmas lehetőségeket rejt magában. Bár a kvantumszámítógépek megbízható használata még várat magára, kvantumalgoritmusokat már tömegével terveznek a kuta-

tók, hiszen – mint az a gépi tanulás esetében is történt – könnyen lehet, hogy csak idő kérdése, hogy ezeket a gyakorlatban is kipróbálhassuk. Ezen kvantumalgoritmusok között szép számmal szerepelnek a gépi tanulásból származó modelleket kvantumszámítógépre átültető megoldások is.

A dolgozat 1. fejezetében egy rövid kvantuminformatikai ismertető olvasható. A 2. fejezet a gépi tanulás területét mutatja be – természetesen közel sem kimerítően. Ezek után kerülnek sorra bizonyos konkrét gépi tanulási módszerek, és ezek kvantumos változatai: a 3. fejezet a döntési fákkal, a 4. fejezet pedig az ensemble módszerekkel foglalkozik mind klasszikus, mind kvantumos értelemben. A 3.3. szekcióban néhány új, kvantum döntési fákkal kapcsolatos eredményünket közöljük. Az 5. fejezet arról az új algoritmusról szól, amelyet az addig bemutatott eredmények felhasználásával terveztünk meg és implementáltunk. Néhány adathalmazon kipróbáltunk több algoritmust – köztük az újat is – és összevetettük az eredményeket. Az új algoritmussal kapcsolatban is szóba kerül a kvantumos megvalósítás. A dolgozat összegzéssel zárul.

1. fejezet

Kvantuminformatika

Az 1970-es évektől kezdve már beszélhetünk kvantuminformatikáról, de ekkor még alig voltak eredmények. A tudományág az 1980-as években kezdett igazán fejlődni (Feynman [5]), bár ekkor is még elméleti szinten maradt. Az 1990-es években készültek el az első (még csak pár kvantumbites) kvantumszámítógépek. Fontos kvantumalgoritmusok is születtek ebben az évtizedben: pl. Shor prímfaktorizáló- és Grover keresőalgoritmusa, amelyek jóval hatékonyabbak klasszikus társaiknál [25].

Ennek a bevezető fejezetnek egyes részei a saját tavalyi TDK dolgozatomból [4] valók.

1.1. Kvantummechanikai és -informatikai alapok

A kvantuminformatika elnevezés onnan származik, hogy ez a terület olyan számítógépekkel foglalkozik, amelyek működése kvantummechanikai jelenségeken alapul. Ebből a szempontból a legfontosabb jelenség például a kétréses kísérletben figyelhető meg: egy átlátszatlan lemezen két, párhuzamos rés van. Ennek az egyik oldalán található forrásból egyesével bocsátunk ki részecskéket (pl. fotonokat vagy elektronokat). A lemez túlsó oldalán lévő ernyőn interferencia kép alakul ki, tehát a vizsgált részecske egyszerre mindkét résen átmegy, és a fáziseltolódás miatt hol gyengíti, hol erősíti önmagát. A részecske helyzete valószínűségi módon írható le. Ez nem csak a mikrorészecskék pozíciójára igaz, hanem más jellemzőire is, ezt írják le a Heisenberg-féle határozatlansági relációk. További fontos kvantummechanikai jelenségek az összefonódás és az alagúteffektus.

1.1.1. A kvantummechanika posztulátumai

A kvantummechanika posztulátumaival tudjuk formalizálni, matematikai alapokra helyezni a kvantuminformatikát.

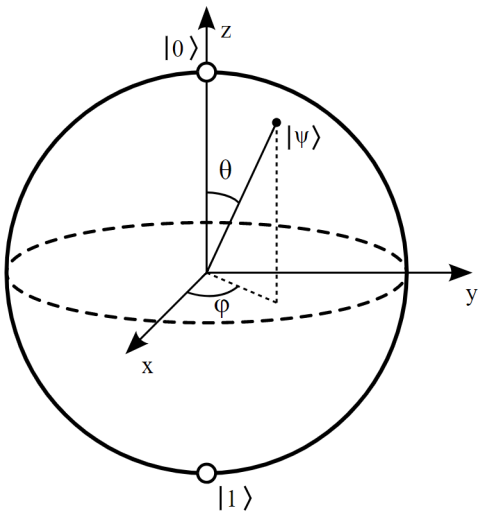
I. posztulátum

Zárt fizikai rendszer állapota egy $\mathcal{H}$ Hilbert-térbeli egység hosszú vektorral írható le (azaz csak az iránya jellemzi, a nagysága nem), ahol $\mathcal{H}$ a rendszer állapottere. A Dirac-formalizmussal ezt „ket” vektorral jelöljük, ami egy oszlopvektornak feleltethető meg: $|\psi\rangle$. A „bra” vektor ennek az adjungáltja, vagyis az a sorvektor, melynek elemei rendre $|\psi\rangle$

elemeinek konjugáltjai: $\langle\psi|$. Ezzel a jelöléssel a skalárszorzatot $\langle\psi|\varphi\rangle$ alakban írhatjuk, amelyre a normáltság miatt $\langle\psi|\psi\rangle = \|\psi\|^2 = 1$ teljesül.

Egy *kvantumbit* (qubit) a két dimenziós $\mathbb{C}^2$ Hilbert-tér egy eleme. Ebben a térben a standard bázisállapotokat $|0\rangle$ és $|1\rangle$ jelöli, így egy $|\psi\rangle \in \mathbb{C}^2$ kvantumbit ezek *szuperpozíció*jában van, és felírható a lineáris kombinációjukként: $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$, ahol $a, b \in \mathbb{C}$, és $|a|^2 + |b|^2 = 1$. Az a és a b ún. *valószínűségi amplitúdó*k, amelyek abszolút érték négyzete adja meg a megfelelő állapotban mérés valószínűségét (erről a III. posztulátumban lesz szó bővebben). Szokásos jelölés szerint ekkor $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$, azaz $|\psi\rangle$ i-edik koordinátája az i-edik bázisállapot valószínűségi amplitúdóját tartalmazza (most $i \in \{1, 2\}$).

Egy kvantumbit szemléltetésére Bloch-gömböt (1.1. ábra) szokás használni. Mivel egy kvantumbit a $\mathbb{C}^2$ egy eleme, alapvetően négy független valós paraméterrel írható le, azonban mivel tudjuk, hogy a normája 1, három paraméter is elég. A Bloch-gömb két átellenes pontja jelképezi a két standard bázisvektort, a felületén helyezkednek el a tiszta állapotok (amelyek nem állnak elő más állapotok konvex kombinációjaként), a belsejében pedig a kevert állapotok (a nem tiszta állapotok). A középpont a maximálisan kevert állapot.



1.1. ábra. *A Bloch-gömb vázlata (forrás: Wikipedia - Bloch sphere)*

II. posztulátum

Zárt rendszer időbeli fejlődése csak a kezdő- és a végállapottól függ, azaz a bemenet és a transzformáció ismeretében tudjuk, mi a kimenet, valamint a kimenet és a transzformáció ismeretében meg tudjuk mondani, hogy mi volt a bemenet. Ez pontosan azt jelenti, hogy az időbeli fejlődés leírható *unitér transzformáció*kkal, azaz olyan operátorokkal, amelyeknek az adjungáltjukkal vett szorzata az identitás (vagyis $UU^* = I$). Ezek a transzformációk ugyanis hossztartók, így normalizált bemenet esetén a kimeneti vektorok is egység hosszúak lesznek. Például a $|\psi\rangle$ állapoton végrehajthatunk egy U unitér transzformációt: $U\,|\psi\rangle = |\varphi\rangle$. Ekkor $\|\psi\| = \|\varphi\| = 1$.

Ezeket a transzformációkat *kvantumkapu*knak is nevezik, és gyakran a klasszikus áram-

körökhöz hasonló ábrákon ábrázolják a kapuk sorozatából előálló algoritmusokat. Híres, gyakran használt kapuk az I (vagy σ_0) identitás kapu mellett a Pauli-féle X, Y és Z (más jelöléssel rendre $\sigma_X, \sigma_Y, \sigma_Z$ vagy $\sigma_1, \sigma_2, \sigma_3$) kapuk, valamint a H Hadamard-kapu.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Ez utóbbinak azért van nagy jelentősége, mert segítségével például a $|0\rangle$ állapotból (ami 1 valószínűséggel 0) olyan állapotot tudunk létrehozni, ami egyenletes eloszlású, azaz 0,5 valószínűséggel 0 és ugyanilyen eséllyel 1:

$$H|0\rangle = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

III. posztulátum

Kvantumrendszer belső állapota nem figyelhető meg, a megfigyeléshez *mérés*sel klasszikus állapottá kell azt alakítani. A rendszer állapota a $\mathcal{H}$ Hilbert-tér elemei közül kerülhet ki, a mérés lehetséges kimeneteinek halmaza $\mathcal{X}$. Az M mérés M_x operátorok halmaza, ahol $x \in \mathcal{X}$, és M_x az x kimenetet eredményező mérés operátora. Ezek összegének az identitást kell adnia, hogy a mérés lehetséges kimeneteinek valószínűségösszege egy legyen. Az M mérésre akkor mondjuk, hogy pozitív operátor értékű mérés (POVM), ha minden $M_x \in M$ pozitív szemidefinit, és véges sok kimenet következhet be. Azaz a POVM-ek halmazára $\mathrm{POVM}(\mathcal{X}, \mathcal{H}) = \{M = \{M_x\}_{x\in\mathcal{X}} : M_x \geq 0,\ \#\{x : M_x \neq 0\} < \infty,\ \sum_{x\in\mathcal{X}} M_x = I\}$.

Például a $|\psi\rangle = a|0\rangle + b|1\rangle$ kvantumbit esetén az $M = \{M_0, M_1\}$ mérés hatására $|a|^2$ valószínűséggel kapjuk a $|0\rangle$, és $|b|^2 = 1 - |a|^2$ valószínűséggel a $|1\rangle$ klasszikus állapotot.

Egy M mérés projektív (PVM), ha POVM, és projekció (azaz $M^2 = M$). Így a PVM-ek halmaza: $\mathrm{PVM}(\mathcal{X}, \mathcal{H}) = \{M = \{M_x\}_{x\in\mathcal{X}} : M \in \mathrm{POVM}(\mathcal{X}, \mathcal{H}),\ M_x^2 = M_x\ \forall x \in \mathcal{X}\}$.

IV. posztulátum

Összetett rendszer állapota a részrendszerek állapotainak tenzorszorzata. A több kvantumbitből álló rendszereket *kvantumregiszter*eknek nevezzük. Például a két kvantumbites, $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$ és $|\varphi\rangle = \begin{bmatrix} c \\ d \end{bmatrix}$ qubitekből álló összetett rendszer állapota $|\psi\rangle \otimes |\varphi\rangle = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$. A regiszter állapota többféleképpen jelölhető: $|\psi\rangle \otimes |\varphi\rangle = |\psi\rangle|\varphi\rangle = |\psi, \varphi\rangle = |\psi\varphi\rangle$.

Látható, hogy egy két qubites rendszer esetén négy állapot különböztethető meg, hiszen mindkét bit mérhető 0-nak vagy 1-nek. Így itt már négyelemű a standard bázis, aminek jelölése $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. Az előző példa ebben a bázisban felírva:

$$|\psi\varphi\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

Egy általános, n kvantumbites rendszer leírásához tehát már 2^n együttható szükséges. Eb-

ből látható a kvantuminformatika erőssége: egy n qubites rendszer szimulálásához klasszikusan 2^n komplex számot kell eltárolnunk.

1.1.2. További alapismeretek

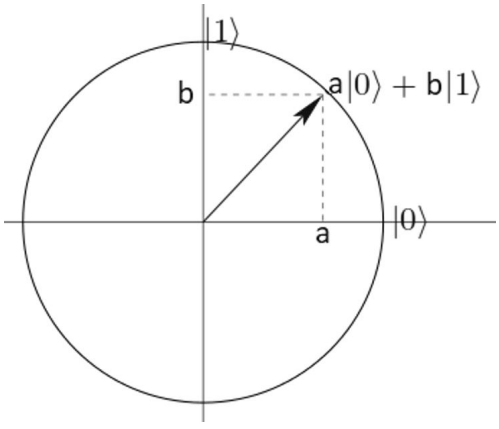
Összefonódás

Egy összetett állapotból általában kikövetkeztethető, hogy milyen alapállapotok összességeként állt elő, pl. $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Azonban akadnak kivételek, például az $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ állapotot nem lehet felbontani két qubit tenzorszorzatára. Az ilyen kvantumbitpárokat nevezzük összefonódott pároknak.

Ezeknek az érdekessége az, hogy ha a kvantumbitpár egyik tagját megmérjük, akkor ennek hatására a másik qubit is egy klasszikus állapotba kerül, mégpedig az első qubit mérésének eredménye egyértelműen meghatározza, hogy melyik állapotba. A fenti esetben például biztos, hogy mérés hatására mindkét kvantumbit azonos állapotba kerül: $1/2$ valószínűséggel mindkettő 0, és szintén $1/2$ valószínűséggel mindkettő 1 lesz. Így ha az egyiket megmérjük, akkor a másik is ugyanabba az állapotba billen be, akkor is, ha a mérés pillanatában nagyon távol van egymástól a két qubit. Ezt a jelenséget sok kvantumkommunikációs eljárásban felhasználják.

Kvantumbit egyszerűbb változata

Egyszerűbben elképzelhető a kvantumbit, ha valós együtthatók esetével foglalkozunk: ekkor egy síkbeli derékszögű koordináta-rendszerben egység hosszú helyvektorokként ábrázolhatjuk a kvantumbiteket (1.2. ábra). Az x tengelyen a $|0\rangle$, az y tengelyen a $|1\rangle$ valószínűségi amplitúdóját (rendre a-t és b-t; $a, b \in [-1, 1]$) olvashatjuk le. Mivel egységvektorról beszélünk, az általános esethez hasonlóan teljesül, hogy $a^2 + b^2 = 1$ a Pitagorasz-tétel miatt. Ekkor annak a valószínűsége, hogy a bitet 0-nak mérjük, a^2; annak pedig, hogy 1-nek mérjük, $1 - a^2 = b^2$.



1.2. ábra. *Kvantumbit egyszerűsített ábrázolása*
(eredeti kép forrása: TrillionByte – What's a Quantum Bit?)

Fizikai megvalósítás

Egy kvantumbit (qubit) realizálására olyan mikrorészecskét használnak, amelynek valamely jellemzője alapján van két, jól megkülönböztethető állapota (pl. foton két, egymásra merőleges síkú polarizációja). Ez a két állapot megfeleltethető a klasszikus informatikában szereplő bit két állapotának, a 0-nak és az 1-nek. A részecske pedig általában ezek szuperpozíciójában van, az egyikben p, a másikban $1 - p$ valószínűséggel, és a mérés az, ami „belekényszeríti" az egyik vagy a másik állapotba (vö. Schrödinger macskája). Pl. a korábban említett kétréses kísérletben, ha a résekhez detektorokat helyezünk, azaz mérést végzünk, akkor már nem alakul ki interferencia kép, csak azon a résen halad át a részecske, amelyik detektora jelez.

A leggyakrabban ioncsapdát, szupravezető mesterséges atomokat vagy szennyezett kristályokat használnak a fizikai megvalósításra.

1.2. Néhány hatékonyabban megoldható probléma

Vannak olyan problémák, amelyek kvantumszámítógéppel hatékonyabban oldhatók meg, mint klasszikusan. Például ismert probléma a számok prímtényezőkre bontása, azaz a prímfaktorizáció. A rendkívül elterjedt RSA nyilvános kulcsú titkosítás sem lenne biztonságos, ha a problémára ismert lenne polinomiális algoritmus. Klasszikusan nem ismert, hogy létezne ilyen, azonban Peter W. Shor 1994-ben megalkotta a hatékony prímfaktorizáló kvantumalgoritmust [18]. Szerencsére azonban egyelőre nem tudunk róla, hogy lenne a jelenleg használt, több száz jegyű prímek szorzatának felbontásához megfelelő kapacitású és kellően stabil kvantumszámítógép. Idővel lehet, hogy lesz, viszont a kvantuminformatika másik ága, a kvantumkommunikáció segíthet új, biztonságos titkosítás elterjesztésében. A bevezetőben említett kvantumkommunikációs hálózat ugyanis olyan szimmetrikus kulcsú titkosítás megvalósításához használható, ami bizonyítottan feltörhetetlen.

Egy másik példa a rendezetlen halmazban keresés. Egy n elemű halmazban keresünk egy x elemet. Ha a halmaz rendezetlen, akkor ehhez (legrosszabb esetben) minden elemet meg kell vizsgálnunk, hogy egyenlő-e x-szel, azaz klasszikusan a lépésszám n-nel arányos. Lov K. Grover 1996-ban írta le azt a kvantumalgoritmust, ami $O(\sqrt{n})$ lépéssel megoldja a keresést [10].

1.3. Számítási modellek

Több modell alapján is megvalósíthatók kvantumszámítógépek. Először az *áramköri modell* alakult ki Deutsch 1989-es cikke [3] nyomán. Ez vált a standard modellé. Ebben kvantum logikai kapuk sorozatából áll elő a számítás. A kvantumbitek működését nehéz összehangolni, és a qubitek számának növekedésével azok állapotának stabilitása csökken, ezért egyelőre csak kevés bites áramköri alapú kvantumszámítógépek vannak. Az áramköri modellre épülnek pl. az IBM és a Google kvantumchipjei. Az ilyen alapokon nyugvó számítógépeket nevezzük univerzális kvantumszámítógépeknek. Több, az áramkörivel ekvivalens megközelítés is létezik még.

A *lehűtéses számítási modell* főleg optimalizálási és mintavételezési problémák megoldására használatos. Azt használja ki, hogy a fizikai rendszerek mindig az energiaminimumra törekednek. Az energiaszint a qubitek állapotának függvényében az ún. Hamilton-függvényből kapható meg. Itt nem mi befolyásoljuk a működést, hanem a probléma betáplálása (súlyok hozzárendelése a kvantumbitekhez és az ezek közti kapcsolatokhoz) után hagyjuk, hogy a fizikai törvényszerűségeknek megfelelő változások történjenek. A végső, minimális energiaszintnek megfelelő állapot magában hordozza a megoldást. A D-Wave Systems cég kvantumszámítógépei (a tavaly megjelent változat már több mint 2000 qubites) ezt a modellt követik, és nem univerzálisak [19]. Vannak akik abban is kételkednek, hogy valóban kvantumos elven működő számítógépekről van-e szó.

2. fejezet

Gépi tanulás

A tanulás fogalmát általában élőlényeken szoktuk értelmezni: a korábbi tapasztalataik alapján módosítani tudják a viselkedésüket. Gépi tanulás esetén is hasonlóról van szó: a tanuló gép a környezetéből szerzett információk segítségével javítja a teljesítőképességét – ennek megvalósítása egy tanuló algoritmus kifejlesztésével történik. A fejezet fő forrása [1].

A gépi tanulásnak több fajtáját különböztetjük meg:

- Ellenőrzött tanulás (felügyelt tanulás, tanulás tanítóval)
- Félig ellenőrzött tanulás
- Nemellenőrzött tanulás (felügyelet nélküli tanulás, tanulás tanító nélkül)
- Analitikus tanulás

Ellenőrzött tanulás esetén a tanuláshoz használt adathalmaz összetartózó bemenet-kimenet párokból áll, és a rendszer feladata ezek alapján a bemenet $\rightarrow$ kimenet függvény minél pontosabb közelítése.

Nemellenőrzött tanulásnál a kimenetek (címkék, kívánt válaszok) nem adottak, a gép feladata a bemeneti adatok közötti kapcsolatok felfedezése (pl. csoportok felismerése).

A kettő közötti a félig ellenőrzött tanulás: az adatok kis része az ellenőrzött tanuláshoz megfelelő, a többi azonban nem címkézett. A sok címkézetlen adat is felhasználható a közelítendő leképezés becslésének javításához.

Analitikus tanulás esetén a kapott adatokból matematikailag számítjuk ki, hogyan kellene működnie a rendszernek. Ez a tanulási típus olyan szempontból kivételnek számít, hogy nem igényli több iteráció futtatását, egyetlen lépésben megkapjuk az eredményt.

A dolgozatban bemutatott módszerek mind az ellenőrzött tanulás kategóriája alá tartoznak, ezért ezt bővebben is kifejtjük.

2.1. Ellenőrzött tanulás

Ellenőrzött tanulás esetén tehát olyan adatokat kap a tanuló rendszer, amelynél a mintapontok összetartozó bemenet-kimenet párok. Ezeket használja arra, hogy olyan paramétereket állítson be minél optimálisabban, amelyek meghatározzák a tanított rendszer működését, vagyis azt, hogy milyen bemeneteket milyen kimenetekre képezzen.

Ezt úgy is felfoghatjuk, hogy van egy rendszer, aminek a működését közelíteni szeretnénk, és ami egy $\underline{x} \mapsto g(\underline{x})$ leképezést valósít meg. Ezt a g függvényt nem ismerjük, csak az $\{\underline{x}_i, d_i\}$ mintapont-halmazt (mintakészletet), ahol $d_i = g(\underline{x}_i)$, $\underline{x}_i$ pedig általában többdimenziós vektor, azaz több tulajdonsága van a bemeneteknek (de egy probléma esetén tipikusan minden i-re azonos a tulajdonságok száma).

A tanuló rendszer leképezését f-fel jelöljük. Tartozik hozzá egy $\underline{w}$ paramétervektor, amivel azt írjuk le, hogy adott bemenethez milyen kimenetet rendel. Ez a tanulás iterációi során mindig változhat. Így a rendszer az $f(\underline{x}; \underline{w})$ leképezést valósítja meg, és az a célunk, hogy $\underline{w}$-t úgy módosítsuk a tanulás során, hogy a kapott $\underline{w}^*$-gal az $f(\underline{x}; \underline{w}^*)$ függvény minél jobb közelítése legyen a modellezendő rendszer $g(\underline{x})$ leképezésének.

Ezt a tanító mintakészletet felhasználva tudjuk elérni: az egyes iterációkban $y_i = f(\underline{x}_i; \underline{w})$ válaszoknak a d_i kívánt válaszoktól való eltérését használjuk arra, hogy eldöntsük, hogyan módosítjuk a $\underline{w}$ paramétervektort a következő iterációra. A tanuló eljárás végére kapjuk a valamilyen szempontból optimális $\underline{w}^*$ paramétervektort.

2.1.1. Osztályozás

Osztályozási feladatról akkor beszélünk, ha a lehetséges kimenetek halmaza véges, egyébként regresszió a feladat. A dolgozatban csak az előbbivel foglalkozunk. Az osztályozás bináris, ha a kimenetek csak kétféle értéket vehetnek fel (általában $\{0, 1\}$ vagy $\{+1, -1\}$).

Tekinthetnénk többdimenziós kimeneteket is, de ha a kimeneti vektornak n koordinátája van, és a koordináták értelmezési tartománya k elemű, akkor egy k^n méretű halmazon értelmezett egyetlen kimeneti változóval is ugyanazok az értékek fejezhetők ki.

2.1.2. Minősítés

Valamilyen módon meg kell határozni, hogy mi alapján tekintünk egy közelítést jónak vagy rossznak. Ehhez egy $L(g(\underline{x}), f(\underline{x}; \underline{w}))$ *költségfüggvény*t definiálunk, ami tehát a tanuló rendszer által adott f közelítéstől és a kívánt d választól is függ. Gyakran például a rendszer által adott és az elvárt válasz különbsége, azaz a közelítés hibája használatos költségfüggvényként: $\varepsilon(\underline{x}, \underline{w}) = |g(\underline{x}) - f(\underline{x}; \underline{w})|$.

Ezt azonban csak a megadott mintapontokra tudjuk kiszámolni, új bemenetekre ez alapján semmit nem mondhatunk. Pedig kardinális kérdés, hogy a rendszer jól tudjon általánosítani, azaz akkor sem lehetünk elégedettek, ha minden tanító mintapontot jól osztályoz; az is szükséges, hogy a minták alapján egy olyan modellt állítson elő, amely a hasonló, új adatokra is jól működik [20]. Hasonló alatt azt értjük, hogy a tanító minták és az új adatok is egyazon valószínűségi eloszlásból való mintavételezéssel származtathatók – csak az új adatoknál nem ismerjük a kimeneti részt.

Ha sok paraméterünk van ($|\underline{w}|$ összemérhető a tanító mintakészlet méretével), akkor az f függvény nagyon sokféle lehet, ezért viszonylag könnyen kivitelezhető, hogy szinte minden tanító mintapontot jól osztályozzon. Azonban ezt úgy is elérheti, hogy a paraméterekben azt tárolja, hogy melyik pontnak mi az osztálya. Így viszont egy új pontot nem fog tudni elég nagy valószínűséggel jól osztályozni. Ezt a jelenséget nevezik *túlilleszkedés*nek (overfitt-

ing). Azt a problémát, hogy egyrészt a mintapontokra elég jól illeszkedjen a függvényünk, másrészt a paraméterek számát tartsuk kordában (ezzel biztosítva az általánosítóképességet), nevezik *torzítás-variancia dilemmá*nak [2].

Segítene a probléma megoldásában, ha olyan „új" bemeneten is kiértékelhetnénk a tanulás eredményeként kapott függvényt, amelyet nem használtunk fel a tanításhoz. Erre az a bevett módszer, hogy a kapott adatokat, azaz a mintapontok halmazát több részre bontjuk:

- Tanító mintakészlet: amit a tanításra közvetlenül felhasználunk
- Validációs (kiértékelő) mintakészlet: ezt még a tanítás folyamata alatt arra használhatjuk, hogy értékeljük, éppen mennyire előrehaladott a tanulás (közvetve része a tanításnak)
- Tesztkészlet (minősítő készlet): amit nem használunk fel a tanítási folyamat során, azt a végső értékelésre, az általánosítóképesség becslésére használjuk

Elég sok mintapont esetén ez nem okoz gondot. Kevés pont esetén a *kereszt-kiértékelés* jelenthet megoldást: a mintapontokat felbontjuk k db diszjunkt részhalmazra, és $k-1$ db részhalmaz pontjait tanításra használjuk, a maradékot pedig validációra. Ezt k-szor végezzük el úgy, hogy minden részhalmaz legyen egyszer validációra használva.

2.2. Néhány gyakori gépi tanuló módszer

Gépi tanulásról leginkább az 1950-es évek óta beszélhetünk, azóta rengeteg, változatos módszer alakult ki a területen. Az egyik a döntési fa, amelyről a következő fejezet szól. A teljesség igénye nélkül röviden említést teszünk még néhány típusról.

Mesterséges neurális hálózatok

A mesterséges neurális hálózatok az elsők között jelentek meg, de napjainkban is a legnépszerűbb változatok közé tartoznak. A biológiai neurális hálózatok adták az alapötletet hozzájuk: sok hasonló, egyszerű építőegységből, neuronból (a biológiában az idegsejteket nevezik így) állnak, amelyek sokféleképpen kapcsolódhatnak egymáshoz. Párhuzamos feldolgozást tesznek lehetővé, és redundancia valamint adaptivitás jellemzi őket. Ezáltal sokféle probléma megoldására alkalmasak, például kitűnően teljesítenek felismerési feladatok esetén (minták alapján).

Mély tanulás

A mély tanulás (deep learning) leggyakrabban a mesterséges neurális hálózatok csoportjába tartozó sokrétegű hálók alkalmazását jelenti, de más többrétegű módszerek is idetartoznak. A neve onnan ered, hogy több, egymással kapcsolatban álló rétegből áll: az egymást követő rétegeknél az egyik kimenete a következő bemeneteként szolgál. Az egyes rétegek különböző feladatokat végezhetnek, például előfeldolgozást, jellemzők kivonását, detekciót stb.

Bayes-hálók

Valószínűségi változókat és ezek feltételes függőségeit irányított aciklikus gráfként (DAG) ábrázolhatjuk. Az így kapott valószínűségi hálót Bayes-hálónak nevezzük, mivel a Bayes-tétel segítségével tudunk benne összefüggéseket meghatározni bizonyos feltételes valószínűségek ismeretében. Ha csak a Bayes-tételt használnánk, és minden kiszámolt feltételes valószínűséget eltárolnánk, akkor exponenciálisan több tárhelyet használnánk, mint ha a háló segítségét igénybe vesszük, ugyanis az utóbbi esetben tudjuk, hogy a valószínűségi változók közül csak bizonyosak függnek egymástól.

Szupport vektor gépek

A szupport vektor gépek (SVM) a kernel gépek csoportjába tartoznak. Alapváltozatuk lineáris szeparálásra képes, de kiterjeszthető nemlineáris szeparálásra és regresszióra is. Előnyük, hogy olyan lineárisan szeparálható feladat esetén, amelynek több jó megoldása is van, nemcsak egy tetszőleges megoldást adnak, hanem a legjobb, maximális margójú megoldást. A megoldás komplexitása korlátos marad, ami az ún. kernel trükk hatása. Az SVM-ek részletes bemutatása nem célja a dolgozatnak, az érdeklődőknek ajánljuk [1] 6. fejezetét.

Genetikus algoritmusok

A genetikus algoritmusok a biológiai evolúciós folyamathoz hasonló módon próbálnak eljutni a legjobb megoldásokhoz. Minden iteráció bemenete egy populáció, azaz egyedek (potenciális megoldások) egy halmaza. Ezekhez egy fitneszfüggvény annak megfelelően rendel értéket, hogy mennyire „jó” az adott megoldás. Ez alapján kiválasztunk az egyedek közül néhányat, akik a következő generáció szülei lesznek. A szülőpárok keresztezik a megoldásaikat, majd véletlenszerűen mutáció történik a kapott megoldásokon. A fitneszfüggvény alapján dől el, hogy ki kerül a következő populációba a régi és az új egyedek közül.

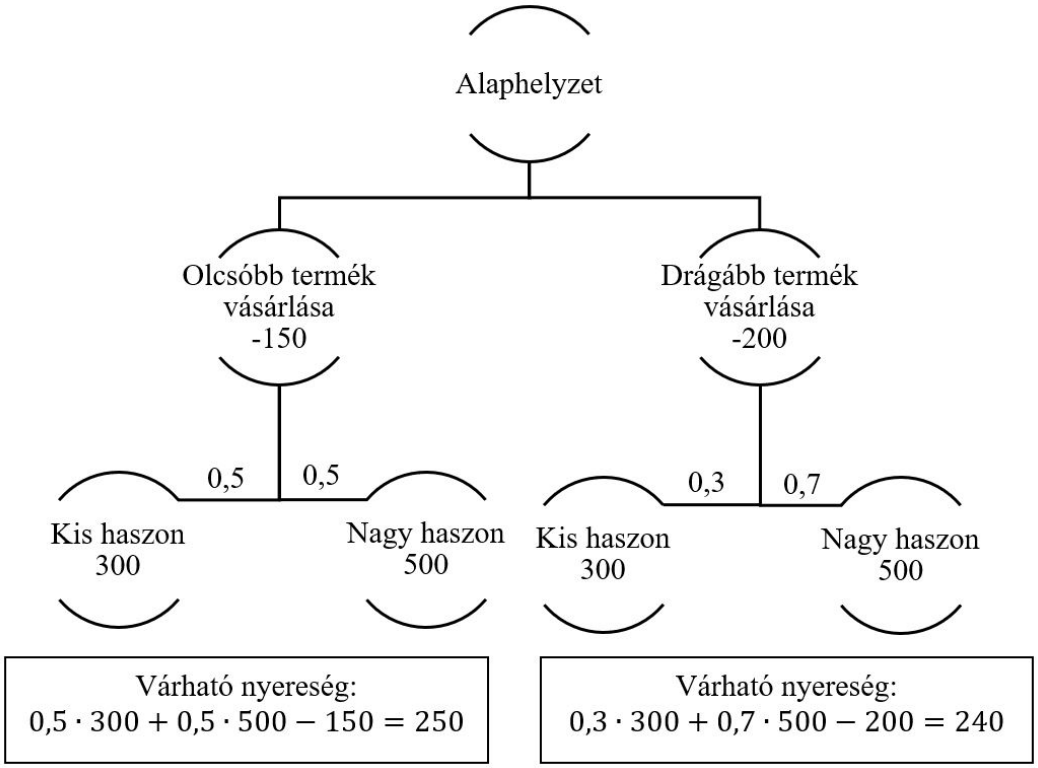
Megjegyezzük, hogy genetikus algoritmusokat általában akkor alkalmazunk, amikor a klasszikus optimalizálás nem vezet eredményre.

3. fejezet

Döntési fák

A döntési fákat sok esetben döntési helyzetek modellezésére, döntéstámogatásra használják. Ekkor a fa csomópontjaiban történik az esetekre bontás: adott helyzetben a lehetséges események vagy döntések szerint ágazik tovább a fa. Az események bekövetkezési valószínűségét is érdemes felvenni a megfelelő elágazáshoz. A levelekben található végkimenetelekhez tudjuk, hogy melyik mennyire hasznos a számunkra, és már azt is kiszámíthatjuk, hogy milyen döntések esetén melyik mekkora eséllyel következik be. Így segíthetjük a döntéshozatalt. Erre látható egy példa a 3.1. ábrán.

A fejezetben először röviden vázoljuk a klasszikus döntési fákat, majd az ezekhez hasonló módon leírható kvantum döntési fákat egy 2014. évi cikk [14] alapján. Az utolsó szekció néhány, ezen kvantum döntési fák használatát megkönnyítő észrevételünket tartalmazza.



3.1. ábra. *Egy egyszerű döntési fa*

3.1. Klasszikus változat

A döntési fák gépi tanulási módszerként is használhatók, ilyenkor osztályozásra (vagy regresszióra, de erre az esetre itt nem térünk ki) használjuk azokat. Adott egy bemeneti minta: tulajdonságok vektora, amelynek egy-egy tulajdonságáról kérdést teszünk fel a csomópontokban, és a választól függő irányban megyünk tovább egy csúcsba a fa következő szintjén. A levelekhez tartozik egy-egy címke (osztály), ami az osztályozás eredménye.

Ha előírjuk, hogy minden csomópontban kétfelé kell ágaznia a fának, akkor bináris döntési fáról beszélünk. A továbbiakban az általánosabb változatról lesz szó, azaz az egyes tulajdonságok ellenőrzésekor tetszőlegesen sok irányba ágazhat a fa.

Fa építés

A kész fa használata egyszerű, a fent írtaknak megfelelően történik. Gépi tanulási szempontból a feladat a döntési fa konstruálása a tanító mintapontok alapján. Ahhoz hogy ezt megtegyük, meg kell határoznunk, hogy az egyes csomópontokban melyik tulajdonság szerint és milyen ágakra bontunk. Ehhez általában az információelméleti *entrópia* fogalmát használják, mi is így fogunk tenni (de pl. a CART algoritmus a Gini-index alapján választ [13]). Az entrópia kiszámítása, ha N lehetőségünk van, amelyek valószínűségei a p_i értékek ($i \in \{1, \ldots, N\}$, és $\sum_{i=1}^{N} p_i = 1$):

$$H(T) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Esetünkben azt a tulajdonságot szeretnénk megtalálni, amely szerint a vizsgált csomópontot szétbontva az információnyereség maximális. Minden választható tulajdonság minden potenciális ágához tartozik egy ilyen H érték, ahol a p_i értékek az adott ágra eljutó minták között az egyes osztályokba tartozók arányát jelentik, N pedig az ezen ágra eljutó minták között előforduló címkék száma. Minden egyes tulajdonságra súlyozottan összegezzük a hozzá tartozó ágak entrópiáit, a súlyozás alapja az adott ágra jutó elemek számaránya. Mivel az információnyereségre vagyunk kíváncsiak, ezt még ki kell vonni a kiinduló állapot (tökéletes osztályozással) entrópiájából. Így minden tulajdonsághoz megkapjuk, hogy az alapján szétbontva a mintákat mekkora információnyereség érhető el, és ezek közül a legnagyobbat érdemes választani.

3.2. Kvantum döntési fák

A feladat és az alapgondolat is azonos a klasszikus változatéval. A bemeneti vektorokhoz szeretnénk a megfelelő osztályozási címkéket rendelni. A tanítóhalmaz minden elemét azonos tulajdonságok írják le, és ezeket mind, valamint a kimeneti címkéket is ismerjük. Egy tulajdonságot tipikusan több kvantumbit ír le, így az i-edik mintapont egy $|x_i\rangle$ vektorként (kvantumregiszterként) jelenik meg; az ehhez tartozó kimeneti címke is lehet több qubites, ezt $|y_i\rangle$-vel jelöljük. A klasszikus változathoz hasonlóan itt is a fa megépítése jelenti a kihívást. A szekció forrása [14].

Melyik tulajdonság szerint ágazzunk el?

A fa építésekor el kell döntenünk, hogy az egyes csomópontokban melyik tulajdonság alapján érdemes szétválasztani a bemeneteket. Klasszikus esetben ehhez az entrópiát használtuk, azonban mivel a bemenet most lehet szuperpozícióban is, ehelyett egy hasonló másik, a kvantum-információelméletben használt mértéket vezetünk be, a *kvantum-* avagy *Neumann-entrópiát*: $S(\rho) = -\operatorname{Tr}(\rho \log \rho)$, ahol $\rho = \sum_{i=1}^{n_{t_j}} p_i^{t_j} \left|y_i^{t_j}\right\rangle \left\langle y_i^{t_j}\right|$. A jelölések feloldása: a t_j csomópontba eljutó tanítóminták között még n_{t_j} db különböző kimeneti osztály szerepel, ρ a különböző $\left|y_i^{t_j}\right\rangle$ osztályok (pontosabban az azok önmagukkal vett külső szorzatából kapott diádok) $p_i^{t_j}$ valószínűségekkel súlyozott átlaga. A $p_i^{t_j}$ szám a t_j csomópontba jutó mintapontok között az i-vel megegyező osztályúak aránya. Ahogy a klasszikus változatnál is láttuk, ha a t csomópontot szeretnénk több ágra bontani, akkor ehhez a potenciális gyerekcsomópontok entrópiáját számítjuk ki. Ezért írtuk fel a képletet a t_j csomópontra, ami alatt a t-nek valamely potenciális gyerekét értjük.

A gyerekcsomópontok Neumann-entrópiájából még ki kell számítani a t csomópontnak a különböző tulajdonságok szerinti felbontásainak várható információnyereségét. Ezt oldja meg a következő fogalom. A *várható kvantumentrópia*, ha a t csúcsban az f tulajdonság szerint ágazik szét a fa $b_{t,f}$ db ágra: $S_e(\rho_f^t) = \sum_{j=1}^{b_{t,f}} p_j S(\rho_{f,j}^t)$, ahol p_j az j. gyerek csomópontba jutás valószínűsége, azaz az ide eljutó tanítóminták száma osztva a t-be eljutó tanítóminták számával; $S(\rho_{f,j}^t)$ pedig a j. gyerek csomóponthoz tartozó Neumann-entrópia. Mindig a legkisebb várható entrópiájú tulajdonság szerint ágazunk el, amit például a Grover-algoritmussal találhatunk meg.

Milyen ágakat hozzunk létre?

Annak eldöntéséhez, hogy adott csúcsban az előbbiek alapján már meghatározott tulajdonság szerint milyen ágakra bontsunk, bevezetünk néhány fogalmat. *Multiplicitási arány* (multiplicity ratio): a t csomópontban az i tulajdonság j. lehetséges értékéhez tartozó állapot $\left|x_{i,j}^{(t)}\right\rangle$. Ennek a multiplicitási aránya a saját előfordulásainak száma osztva a t-be eljutó minták számával. Egy állapotra azt mondjuk, hogy *nagy állapot*, ha ez az érték nagyobb egy előre megadható értéknél. *Egyszerű mintázati arány* (simple pattern ratio): a t csúcsban az i tulajdonság szerinti nagy állapotok száma osztva az összes különböző t csúcsbeli, i szerinti állapot számával. A t csúcsban egy i tulajdonság *egyszerű mintázatú* (simple pattern), ha az előbbi arányszám nagyobb egy előre megadható értéknél, egyébként *összetett mintázatú* (complex pattern).

Egyszerű mintázatú esetben az adott tulajdonság minden különböző állapotának külön ágat hozunk létre, és minden bemenet az adott tulajdonsághoz tartozó állapotának megfelelő ágon megy tovább. Az összetett mintázatú esetre bevezetünk egy hasonlósági mértéket. Két bemeneti $|x_i\rangle, |x_j\rangle$ vektorhoz tartozó hasonlósági mérték: $F(|x_i\rangle, |x_j\rangle) = \operatorname{Tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}}$, ahol $\rho = |x_i\rangle\langle x_i|$, $\sigma = |x_j\rangle\langle x_j|$. Ez az érték 0 és 1 közötti, és minél hasonlóbb a két állapotvektor, annál nagyobb. Először minden lehetséges (az adott csúcsba eljutó mintákból képzett) pár közül kiválasztjuk a legnagyobb hasonlóságúakat: ezek lesznek a *centroid*ok. Ezután minden nem-centroidot a hozzá leghasonlóbb centroid cso-

portjába (cluster) teszünk. A maximális hasonlóság meghatározásához mindkét helyen a Grover-algoritmust használhatjuk.

Keresés a fában

Miután megkonstruáltuk a döntési fát, a kész fában a keresés a következőképpen zajlik. Az új bemenettel a fa gyökerétől indulunk. Minden aktuális csúcsnál összehasonlítjuk a bemenet állapotvektorát minden egyes lehetséges ággal (centroiddal) oly módon, hogy kiszámítjuk a hasonlóságukat (F értékét). A legnagyobb hasonlóságú ág irányában haladunk a fában. Végül levélhez érkezünk, aminek a címkéje meghatározza a kimeneti osztályt.

3.3. Alkalmazást segítő új eredmények

Ebben a szekcióban a kvantum döntési fákkal kapcsolatban néhány saját eredmény kerül bemutatásra.

F valójában skalárszorzat

Belátjuk, hogy a fent definiált hasonlósági mérték valójában a két bemeneti vektor skalárszorzatát adja eredményül. Ennek az eredménynek a felhasználásával egyszerűbb módon számolható a hasonlóság, mint a fentebbi, általános formulával.

Állítás: Az F értéke a két bemeneti egységvektor skalárszorzata.

Bizonyítás: Két egységvektorra, $|x_i\rangle$-re és $|x_j\rangle$-re

$$F(|x_i\rangle, |x_j\rangle) = \mathrm{Tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}} = \mathrm{Tr}\sqrt{(|x_i\rangle\langle x_i|)^{1/2}(|x_j\rangle\langle x_j|)(|x_i\rangle\langle x_i|)^{1/2}} =$$
$$\mathrm{Tr}\sqrt{(|x_i\rangle\langle x_i|)(|x_j\rangle\langle x_j|)(|x_i\rangle\langle x_i|)} = \mathrm{Tr}\sqrt{|x_i\rangle\langle x_i|x_j\rangle\langle x_j|x_i\rangle\langle x_i|} =$$
$$\langle x_i|x_j\rangle \mathrm{Tr}\sqrt{|x_i\rangle\langle x_i|} = \langle x_i|x_j\rangle$$

Az első két lépésben definíciók alapján helyettesítettünk be. Utána azt használtuk ki, hogy egységhosszú $|x_i\rangle$ vektorra $(|x_i\rangle\langle x_i|)^2 = (|x_i\rangle\langle x_i|)(|x_i\rangle\langle x_i|) = |x_i\rangle\langle x_i|x_i\rangle\langle x_i| = |x_i\rangle\langle x_i|$. A negyedik lépésbeli átrendezés után az ötödik lépésben kiemeltük előre a $\sqrt{\langle x_i|x_j\rangle^2} = \langle x_i|x_j\rangle$ skalárt. Végül mivel $|x_i\rangle$ hossza egységnyi, a $|x_i\rangle\langle x_i|$ diád főátlójában pedig a valószínűségi amplitúdók négyzetei szerepelnek (és az imént láttuk, hogy $(|x_i\rangle\langle x_i|)^2 = |x_i\rangle\langle x_i|$), a nyom 1 lesz.

Például két qubitre legyen egy általános centroid: $|x_j\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$; és egy összefonódott pár bemenet: $|x_i\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. A hasonlóság fentebbi definíció szerinti mértéke megegyezik a két vektor skalárszorzatával: $F(|x_i\rangle, |x_j\rangle) = \langle x_i|x_j\rangle = \frac{a+d}{\sqrt{2}}$.

Elég kimeneti halmaz mérete számú centroidhoz hasonlítani

Nem kell a döntési fa minden csomópontjában minden centroiddal összehasonlítani az új bemenetet, ha előállítunk bizonyos szuperpozíciókat. Minden lehetséges kimenethez fog tartozni egy-egy szuperpozíció (tehát bináris osztályozási feladat esetén összesen kettő).

Elegendő az új bemenetet ezekkel összehasonlítani (F kiszámításával), és a legnagyobb hasonlósághoz tartozó osztályba érdemes tenni a bemenetet.

Nézzük meg, hogyan állíthatók elő az említett szuperpozíciók bináris osztályozási feladat esetén. Egy $n+1$ qubites csupa 0 regiszterből ilyen szuperpozíció létrehozható a következő módon (n a mintaadatok tulajdonságokat leíró $|x_i\rangle$ vektorában a qubitek száma):

1. Egyenletes szuperpozíciót állítunk elő az első n kvantumbiten Hadamard-kapuval.
2. Az első n qubiten végrehajtjuk a kérdéses bináris függvényt, az eredmény az $(n+1)$. qubitbe kerül.
3. Az utolsó kvantumbitet megmérjük.

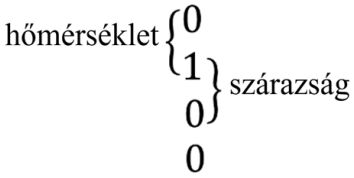
A mérés eredményének függvényében az első n qubit a 0 vagy az 1 eredményt adó állapotok szuperpozíciójába áll. A másik osztályhoz tartozó szuperpozíció előállítása megoldható, ha mindkét kimenet elég gyakran előfordul (aránya legalább 1/polinom), ekkor ugyanis néhány mérés után nagy valószínűséggel sikerül mindkét esetre eredményt kapni.

Így klasszikusan nehéz problémák közül van, ami hatékonyabban megoldható kvantum bináris fa segítségével. Ilyen például a XOR (paritás) és a multiplexer probléma: elegendő az egyes kimeneteknek megfelelő állapotok szuperpozíciójához való hasonlóságot vizsgálni, míg klasszikusan minden bitet végig kell nézni, hogy meghatározzuk az eredményt.

Példa

Nézzünk egy példát arra, hogy kvantum döntési fa építésekor hogyan döntjük el, hogy melyik tulajdonság szerint ágazzon el a fa adott csomópontja.

Területeket osztályozunk éghajlatok szerint. A döntési fa építése közben egy csomópontban két tulajdonság szerint történhet a szétbontás: a hőmérséklet vagy a szárazság szerint. Célunk meghatározni, hogy melyiket érdemes választani. Ezek nem függetlenek egymástól, ami például úgy is megjelenhet a modellben, hogy az ezeket leíró koordináta-halmazok metszete nem üres. Tehát ha két kvantumbit ír le egy bemenetet, akkor a négy lehetséges állapotkonfiguráció valószínűségi amplitúdóit tartalmazó vektor első két eleme a hőmérsékletre, a második és harmadik eleme a szárazságra vonatkozik (3.2. ábra). A negyedik elem valami egyéb tulajdonsághoz tartozhat.



3.2. ábra. *A tulajdonságokat leíró vektor*

A hőmérséklet esetén a $|00\rangle$ a hideg, $|10\rangle$ a kicsit hideg, $|01\rangle$ a kicsit meleg, és $|11\rangle$ a meleg állapotot jelenti; szárazságból $|00\rangle$ a nedves, $|01\rangle$ a kicsit nedves, $|10\rangle$ a kicsit száraz, és $|11\rangle$ a száraz. Így például a nedves és a kicsit nedves területek uniója pontosan ugyanaz, mint a hideg és a kicsit hideg területeké.

Az osztályok tehát az éghajlatok, ebből nyolcat használunk, 1-től 8-ig számozzuk őket. A vizsgált csomópontba eljutó tanítóminták a következők (a cél az volt az adatok előállításánál, hogy a hőmérséklet valamivel jobban meghatározó legyen):

első\utolsó bitek	00	01	10	11
00	1	1, 2	1, 2	2, 3
01	2, 4	4, 5	5	3, 5
10	3, 4	2, 6	4, 6	6
11	7	5, 7	7, 8	3, 8

3.1. táblázat. *A példában használt adatok*

Tehát pl. az $|x_k\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ vektorral leírható állapotokhoz (ami hideg és kicsit nedves területet jelent) a 2 és a 3 sorszámú éghajlat tartozhat. Az egyszerűség kedvéért ahol két éghajlat is szerepel, ott ezek valószínűsége fél-fél.

A számítás menete: például a 3.1. táblázat első sorához, tehát a hőmérséklet tulajdonságon belül a hideg állapothoz tartozó entrópiát számítjuk ki. Ehhez az ő sorában szereplő összes előforduló osztály száma méretű mátrixokkal fogunk számolni (elvileg 8×8-as kellene, de most a 0 sorokat és oszlopokat elhagyjuk). Az előbb említett vektorhoz a 2. és a 3. éghajlat (címke) tartozhat, mégpedig mindkettő 0,5 valószínűséggel, ezért ugyanennek a mintának az osztályokat leíró $|y_k\rangle$ vektorában ezekhez $1/\sqrt{2}$ érték tartozik, az 1-hez pedig 0. Így ennek a vektornak a külső szorzata önmagával a csupa 1/2-ből álló 2×2-es mátrix egy előre beszúrt csupa 0 sorral és oszloppal:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 \end{bmatrix}$$

Hasonló mátrixokat számítunk az első sor többi eleméhez is, majd ezeket súlyozva összeadjuk. Most legyen minden súly (valószínűség) 1/4. Az így kapott mátrixot összeszorozzuk a saját logaritmusával, és vesszük az eredménymátrix nyomának ellentettjét. Ezzel megkaptuk az entrópiát, ami ez esetben $S = 0,7306$.

Ugyanezt a többi hőmérsékleti osztályra is kiszámítjuk, és ezek (ismét 1/4-del) súlyozott összegét vesszük, így kapjuk meg a várható kvantumentrópiát. Majd a teljes műveletsort megismételjük a szárazsági osztályok szerint is. A kapott eredmény: a hőmérséklet várható entrópiája 0,93, a szárazságé 1,13, tehát érdemesebb a hőmérséklet alapján dönteni a vizsgált csomópontban.

4. fejezet

Ensemble módszerek

Az egyes gépi tanuló algoritmusok használata az adatoktól függően eltérő eredményességű lehet. Jobb, és az adatok változására kevésbé érzékeny megoldást kaphatunk, ha több algoritmust együtt használunk, sőt többféle modell használatával a hibatűrés is növelhető. Az ilyen algoritmus-összességet, mint meta-gépi tanulási módszert nevezzük ensemble-nek (sokaságnak). Erre változatos módszerek alakultak ki, amelyek közül a fontosabbakat megemlítjük, egy algoritmust pedig részletesen is kifejtünk. A különböző módszerek elnevezésénél – azoknak a magyar nyelvű szakirodalomban való elterjedtsége miatt – az angolszász megnevezést használjuk.

4.1. Klasszikus alapváltozatok

Az alábbi módszerek ismertetéséhez felhasznált irodalom: a bagging esetén az MIT mély tanulási tankönyvének [8] 7.11. fejezete, a dropoutnál [8] 7.12. fejezete, a boostingnál pedig Freund és Schapire cikke az AdaBoost algoritmusról [6].

Bagging

A bagging szó a bootstrap aggregating kifejezés rövidítése. A módszer a modell átlagolás (model averaging) nevű stratégiák közé tartozik, amelyek alapötlete, hogy több osztályozót tanítunk egymástól elkülönítve. Amikor egy új bemenet osztályáról kell dönteni, akkor ezek szavaznak, és a többségi szavazás eredményének megfelelő címkét rendeljük a bemenethez.

A különböző osztályozókat bagging esetén úgy nyerjük, hogy egy adott algoritmust különböző adatokkal tanítjuk (pl. az előző fejezetben tárgyalt döntési fa építő algoritmus különböző paraméterű fákat eredményezhet, ha más adatokkal tanítjuk). A különböző adathalmazokat úgy nyerjük, hogy a k elemű tanító mintahalmazából visszatevéssel mintavételezünk véletlenszerűen annyiszor k elemet, ahány elemű ensemble-t szeretnénk.

A *random forest* (véletlen erdő) módszer is idesorolható. Az erdő döntési fákból áll, amelyeket tanulás közben épít az algoritmus, és amelyek kimeneteinek módusza lesz a random forest kimenete. A döntési fák hátránya, hogy könnyen túlilleszkednek, ezen segít ez a módszer. Úgy működik, hogy a fákra külön-külön véletlenszerűen választja ki a tulajdonságok egy részhalmazát, amely alapján szétbontja a bemeneteket. Idővel a fontos (a kimenet

becslésében hasznos) tulajdonságok nagyobb valószínűséggel kerülnek kiválasztásra (lásd pl. [24]).

Dropout

A dropoutot főleg regularizációs hatása miatt használják a túlilleszkedés elkerülésére. A motiváció, hogy például nagy méretű neurális hálózatok esetén a baggingnek erős a számítás- és memóriaigénye. Ehhez hasonló eredményt érhetünk el jóval hatékonyabban, ha egy alap neurális hálózat minden részhálóját tanítjuk. Egy részhálón olyan hálózatot értünk, amelyet úgy kapunk meg, hogy az alap hálóból elhagyunk néhány neuront (általában a megfelelő súlyokat nullára állítjuk). Így azonban exponenciálisan sok neurális hálót kell tanítani.

Egy új tanítóminta betöltésekor véletlenszerűen választunk egy maszkot, ami meghatározza, hogy melyik neuronok „esnek ki". Az egyes neuronok kiesési valószínűségét hiperparaméter határozza meg (általában 0,5 körüli érték). Az így kapott részhálót a szokott módon tanítjuk hibavisszaterjesztéses algoritmussal.

Fontos különbség a bagginghez képest, hogy itt nem függetlenek egymástól a tanított modellek, osztoznak a paramétereken. Ez teszi lehetővé, hogy hatékonyan taníthassuk az exponenciálisan sok hálót. Ténylegesen minden lépésben a hálóknak csak egy kis részét tanítjuk, és a közös paraméterek által a többi hálózat is megfelelő paraméterezést kap. Ettől eltekintve azonban a dropout a bagging algoritmust követi.

Boosting

A módszer legrövidebben úgy foglalható össze, hogy gyenge osztályozókból egy erőset állít elő. *Gyenge osztályozó* (weak learner) alatt olyan osztályozó gépi tanulóalgoritmust értünk, aminek a pontossága 1/2-nél (azaz a véletlen osztályozásnál) nem sokkal jobb. Ilyen gyenge osztályozókból áll az ensemble.

A tanítás több (T db) körben történik. Minden kör elején adott a mintákon egy eloszlás: a mintákat a „nehezen tanulhatóságuk" szerint súlyozzuk (azaz aszerint, hogy a korábbi körökben mennyire osztályoztuk rosszul őket). Adott körben olyan gyenge osztályozót használunk, aminek ezzel az eloszlással súlyozott hibája alacsony. Ez alapján a hiba alapján módosítjuk az eloszlást, és ezzel kezdjük az új kört. Az utolsó iteráció után megkapjuk, hogy hogyan kell a T db gyenge osztályozó eredményét kombinálni. A tanítás után egy új bemenetet ennek megfelelően osztályozunk.

4.2. AdaBoost

Az AdaBoost algoritmus [6] a boosting módszerek csoportjába tartozik. Itt csak a bináris osztályozási problémák esetét tekintjük át, de ez kiterjeszthető többosztályos osztályozási, sőt regressziós feladatokra is. Az algoritmus neve az „adaptive boosting algorithm" kifejezés rövidítéséből származik, mivel az algoritmus adaptálódik a gyenge osztályozók hibáihoz. Ez az egyik legsikeresebb ellenőrzött tanulási algoritmus, mert általában jól osztályoz, és kevéssé hajlamos a túltanulásra.

A tanítás T fordulóból áll, a tanítóminták száma N, a mintákat tulajdonságvektor-címke párokként kezeljük: $(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)$, ahol $y_i \in \{0, 1\}\ \forall i \in \{1, \ldots, N\}$. Legyen $X = \{\underline{x}_i\}_{i=1}^N$. A t. fordulóban olyan gyenge osztályozót választunk, amelynek a hipotézise (azaz hogy milyen osztályokba sorolná az elemeket) az aktuális eloszlás figyelembevételével a legpontosabbak közül való: $h_t(\underline{x}_i) = y_i$ a lehető legtöbb $i \in \{1, \ldots, N\}$ esetén. Feltesszük, hogy a **WeakLearn** algoritmus ezt a kiválasztást elvégzi, és visszaadja a kiválasztott gyenge osztályozó hipotézis függvényét. Ennek az algoritmusnak a mintákon (azok y_i címkék nélküli részein) túl bemenete a minták D eloszlása is. gyenge osztályozónak általában kis méretű döntési fákat, például döntési tönköket (decision stump), azaz egyszintű döntési fákat használnak.

A tanítás kezdetén a mintapontok eloszlására általában $D(i) = 1/N\ \forall i \in \{1, \ldots, N\}$, de ha van valamilyen a priori ismeretünk a minták nehézségéről vagy fontosságáról, azt is megadhatjuk. A t. körnek egy $\underline{w}^{(t)} \in \mathbb{R}^N$ súlyvektor a kimenete, ami a mintáknak az addigi fordulókban tapasztalható nehézségétől függő súlyozását tartalmazza. Ennek a normalizálásával kapjuk a következő kör eloszlását. A T forduló végén a T db gyenge osztályozó kimenetének súlyozott összegeként kapjuk a h_f végső hipotézist. Adható felső korlát a h_f végső hipotézis ϵ hibájára: ha a t. iterációban hívott gyenge osztályozó hibája ϵ_t, akkor belátható, hogy $\epsilon \leq 2^T \prod_{t=1}^T \sqrt{\epsilon_t(1-\epsilon_t)}$ teljesül.

Bemenet: az N elemű tanító mintahalmaz elemei: $(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)$
D kezdeti eloszlás a minták felett
gyenge osztályozó algoritmus: **WeakLearn**
$T \in \mathbb{Z}^+$: az iterációk száma
Inicializálás: a súlyvektor elemei: $w_i^{(1)} = D(i)\ \forall i \in \{1, \ldots, N\}$
for $t = 1, 2, \ldots, T$ **do**

1. Az eloszlás legyen $\underline{p}^{(t)} = \dfrac{\underline{w}^{(t)}}{\sum_{i=1}^N w_i^{(t)}}$
2. Hívjuk meg a **WeakLearn** metódust a $\underline{p}^{(t)}$ eloszlással.
   A visszatérési értéke a $h_t : X \to [0, 1]$ hipotézis.
3. Számítsuk ki a h_t hipotézis hibáját: $\epsilon_t = \sum_{i=1}^N p_i^{(t)} |h_t(\underline{x}_i) - y_i|$
4. Legyen $\beta_t = \dfrac{\epsilon_t}{1 - \epsilon_t}$
5. Az új súlyvektor kiszámítása: $w_i^{(t+1)} = w_i^{(t)} \beta_t^{1-|h_t(\underline{x}_i)-y_i|}$

end
Kimenet: a végső hipotézis: $h_f(\underline{x})$
if $\sum_{t=1}^T \left(\log \dfrac{1}{\beta_t}\right) h_t(\underline{x}) \geq \dfrac{1}{2} \sum_{t=1}^T \log \dfrac{1}{\beta_t}$ **then**
  $h_f(\underline{x}) = 1$
else
  $h_f(\underline{x}) = 0$
end

Algoritmus 1: Az AdaBoost algoritmus

A kimenet előállításánál látható, hogy a pontosabb hipotézisek kapnak nagyobb súlyt, hiszen β_t a hibával nő, itt pedig ennek a reciprokának a logaritmusával arányos a súly, azaz a hiba növekedésével csökken. A kimenet pontosan akkor lesz 1, ha a gyenge osztályozók eredményeinek súlyozott összegeként kapott érték nagyobb, mint az az érték, amit akkor kapnánk, ha minden gyenge osztályozó 1/2-et prediktált volna.

Megjegyezzük, hogy bár az AdaBoost e változatában a gyenge osztályozók kimenetei folytonosak, azaz $h_t(\underline{x}) \in [0,1]$, az algoritmusnak egy olyan verziója is létezik, ami az itt bemutatottal nagyrészt azonos működésű, de amelynél ezek a kimenetek binárisak: $h_t(\underline{x}) \in \{0,1\}$ (vagy $\{+1,-1\}$).

Az érdeklődő olvasó ezen a linken[1] találhat egy egyszerű példát az algoritmus futására.

4.3. Egy kvantum osztályozó algoritmus

A most bemutatandó algoritmus [17] az egyik első, ami kvantum osztályozókból álló ensemble megalkotásáról szól, és így kihasználja mind a kvantumalgoritmusok, mind az ensemble módszerek előnyeit.

4.3.1. Általános leírás

Az alapgondolat az, hogy az egyes kvantum osztályozókat (gyenge osztályozókat) olyan függvényekként kezeljük, amelyeknek nemcsak bemenete van, hanem paraméterei is, azaz $f\left(|x\rangle\,;|\theta\rangle\right)$ alakban írhatók fel, ahol $|x\rangle$ a bemeneti tulajdonságokból, $|\theta\rangle$ pedig az osztályozó paramétereiből álló vektor. (Vegyük észre, hogy ez a felírás ekvivalens a 2.1. szekcióban használttal: $f(\underline{x};\underline{w})$.) Ez azért hasznos, mert így akár exponenciálisan sok kvantum osztályozó $\sum_\theta |\theta\rangle$ szuperpozícióját tudjuk képezni, és így egyszerre kiértékelhetjük azokat.

Két transzformációt fogunk bevezetni. Az egyik megadja, hogy adott paraméterekkel rendelkező osztályozó adott bemenetre milyen hipotézist ad. Ezt *kvantum osztályozó*nak nevezzük, és $\mathcal{A}$-val jelöljük:

$$\mathcal{A}\,|x\rangle\,|\theta\rangle\,|0\rangle \to |x\rangle\,|\theta\rangle\,|f(|x\rangle\,;|\theta\rangle)\rangle \tag{4.1}$$

Paraméterek (kvantum osztályozók) szuperpozíciójára is elvégezhető a művelet:

$$\mathcal{A}\,|x\rangle\,\frac{1}{\sqrt{E}}\sum_\theta |\theta\rangle\,|0\rangle \to |x\rangle\,\frac{1}{\sqrt{E}}\sum_\theta |\theta\rangle\,|f(|x\rangle\,;|\theta\rangle)\rangle \tag{4.2}$$

A fenti egyenletben E az ensemble mérete, azaz a kvantum osztályozók száma.

A második transzformáció a súlyozást végzi el, azaz az egyenletes szuperpozíció helyett egy súlyozottat állít elő. Ezt *kvantum ensemble*-nek hívjuk, és $\mathcal{W}$-vel jelöljük:

$$\mathcal{W}\,|x\rangle\,\frac{1}{\sqrt{E}}\sum_\theta |\theta\rangle\,|0\rangle \to |x\rangle\,\frac{1}{\sqrt{\chi E}}\sum_\theta \sqrt{w_\theta}\,|\theta\rangle\,|0\rangle \tag{4.3}$$

[1] `https://mitpress.mit.edu/sites/default/files/titles/content/boosting_foundations_algorithms/chapter001.html#h1-2`

Tehát minden egyes θ osztályozó a w_θ valószínűségnek megfelelő súlyt kap. A χ egy normalizáló tényező, vagyis az értéke éppen annyi, hogy $\sum_\theta \frac{w_\theta}{\chi E} = 1$ legyen.

Új bemenet ($|\tilde{x}\rangle$) osztályozásához először súlyozzuk az osztályozókat a $\mathcal{W}$ transzformációval, majd az $\mathcal{A}$ művelettel egyszerre kiszámítjuk az összes osztályozó predikcióját. Az így kapott állapotot írja le a (4.4) képlet.

$$|\tilde{x}\rangle \frac{1}{\sqrt{\chi E}} \sum_\theta \sqrt{w_\theta} |\theta\rangle |f(|\tilde{x}\rangle ; |\theta\rangle)\rangle \tag{4.4}$$

Az utolsó kvantumbit mérési statisztikái alapján kapjuk az ensemble predikcióját. Ha $\tilde{y}$ a mérés eredménye, akkor $p(\tilde{y} = 0) = \sum_{\theta \in \mathcal{E}^+} \frac{w_\theta}{\chi E}$, és $p(\tilde{y} = 1) = \sum_{\theta \in \mathcal{E}^-} \frac{w_\theta}{\chi E}$, ahol $\mathcal{E}^+$ a kvantum osztályozók azon részhalmaza, amelynek minden $\theta \in \mathcal{E}^+$ elemére igaz, hogy $f(|\tilde{x}\rangle ; |\theta\rangle) = 1$. Az $\mathcal{E}^-$ értelemszerűen ugyanez, de a 0 eredményt adó osztályozókra.

4.3.2. Pontossággal arányos súlyozás

Az eddigiekben a $\mathcal{W}$ fekete dobozként súlyozta az osztályozókat. Az alábbiakban $\mathcal{W}$ működésére egy konkrét lehetőséget mutatunk be: az egyes (θ paramétervektorral meghatározott) osztályozók súlya a (4.5) képletben definiált a_θ pontosságukkal (accuracy) lesz arányos, azaz az általuk jól osztályozott tanítóminták számának és az összes tanítómintának az arányával. A tanítóminták halmaza M elemű: $\{(|x_1\rangle, y_1), \ldots, (|x_M\rangle, y_M)\}$. (A tanítóhalmazon számoljuk a pontosságot, nincs külön validációs- vagy tesztkészlet.)

$$a_\theta = \frac{1}{M} \sum_{m=1}^{M} |f(|x_m\rangle ; |\theta\rangle) + y_m - 1| \tag{4.5}$$

A kiindulási állapot csupa 0 qubitből áll, és négy regiszter tenzorszorzataként áll elő. Az adatregiszter $\delta + 1$ qubites – δ db a minták bemeneti részének, 1 pedig a címkéknek a leírására szolgál. A paraméterregiszter τ kvantumbitből áll, és az osztályozók paramétervektora adható meg itt. A kimeneti- és a pontosságregiszter egy-egy qubitesek, előbbi az osztályozók eredményét tárolja, utóbbit pedig a pontosság meghatározásához használjuk. Első lépésként Hadamard-kapuval egyenletes szuperpozíciót hozunk létre a paraméter- és a pontosságregiszteren.

$$|0\ldots0;0\rangle \otimes |0\ldots0\rangle \otimes |0\rangle \otimes |0\rangle \rightarrow \frac{1}{\sqrt{E}} \sum_{i=0}^{2^\tau - 1} |0\ldots0;0\rangle |i\rangle |0\rangle \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \tag{4.6}$$

A (4.6) lépés végállapotában szükségképpen $E = 2^\tau$; minden $|i\rangle$ pedig egy-egy osztályozó paramétereit írja le.

Következő lépésként egyesével betöltjük a tanítómintákat az adatregiszterbe, az $\mathcal{A}$ transzformációval kiszámítjuk a kimenetet, és attól függően, hogy ez megegyezik-e a tényleges kimenettel vagy nem, csekély mértékben rendre a $|0\rangle$ vagy a $|1\rangle$ irányába forgatjuk a pontosság-qubitet. Mire minden tanítómintát feldolgoztunk, a pontosság-qubit összefonódott a paraméterregiszterrel, és a $\sqrt{a_\theta} |0\rangle + \sqrt{1 - a_\theta} |1\rangle$ állapotban van. Ezt megmérjük, és

ha a $|0\rangle$ állapotot kaptuk, akkor az egész rendszer állapota (4.7) lesz, egyébként elvetjük az eredményt, és újrakezdjük az algoritmust.

$$\frac{1}{\sqrt{\chi E}} \sum_{\theta} \sqrt{a_\theta} \, |0 \ldots 0; 0\rangle \, |\theta\rangle \, |0\rangle \, |0\rangle \tag{4.7}$$

A fenti képletben χ normalizáló tényező, ezért $\chi = \frac{1}{E} \sum_\theta a_\theta$.

Most már a pontossággal arányosan súlyozottak az osztályozók. Ha betöltünk egy új bemenetet ($|\tilde{x}\rangle$), és elvégezzük az $\mathcal{A}$ műveletet, akkor a (4.8) állapotba jut a rendszer, ahol a kimeneti qubit mérési statisztikái adják meg a kívánt eredményt (többszöri méréssel növelhető az eredmény pontonossága).

$$\frac{1}{\sqrt{\chi E}} \sum_{\theta} \sqrt{a_\theta} \, |0 \ldots 0; 0\rangle \, |\theta\rangle \, |f(|\tilde{x}\rangle \, ; |\theta\rangle)\rangle \, |0\rangle \tag{4.8}$$

5. fejezet

Az új algoritmus

A fejezetben szereplő eredmények a Friedl Katalinnal, Daróczy Bálinttal és Kabódi Lászlóval közös, még folyamatban lévő munkánkból származnak. A dolgozatban eddig bemutatott ismereteket felhasználva terveztünk egy olyan osztályozó algoritmust, ami kvantumalgoritmusként is megvalósítható.

5.1. Működése klasszikusan

Az algoritmusunk alapötlete az AdaBoost algoritmusból származik, de a kvantumos megvalósíthatóság miatt a 4.3. szekcióban bemutatott kvantumalgoritmusból indultunk ki.

Az algoritmus működése röviden úgy foglalható össze, hogy felváltva súlyozzuk a mintapontokat és a gyenge osztályozókat. A gyenge osztályozók bemenetül kapják a mintapontokat, és ezek osztályozásában elért pontosságuk alapján súlyozzuk őket. Ezután az előbbi súlyozást is felhasználva súlyozzuk a mintapontokat a tanulhatóságuk nehézsége szerint. A mintapontokon így kapott eloszlásnak megfelelően újra mintavételezünk azokból, és ez az új mintaponthalmaz lesz a gyenge osztályozók új bemenete. Ezt iteratívan megismételjük néhányszor.

Az algoritmus implementálása python nyelven történt, a Jupyter Notebook környezet használatával. A tervezés folyamata során több változat is született az alapötlet megvalósítására. A következőkben ezeket ismertetjük.

5.1.1. Mintavételezést használó megvalósítás

Alapvetően a fentebb írtaknak megfelelően működik ez a változat. A mintapontok száma N, eloszlásuk kezdetben egyenletes. Minden gyenge osztályozó minden mintapontot osztályoz, és ezeknek a hipotéziseknek valamint a valódi kimeneti címkéknek az eltérése alapján kapnak súlyt az osztályozók.

Gyenge osztályozónak egyszintű döntési fákat, döntési tönköket használtunk. Ezeket három paraméter ír le: melyik tulajdonságra (azaz a bemeneti vektor melyik koordinátájára) vonatkozik; mi a küszöbérték; és ettől melyik irányba történő eltérés jelenti a 0 illetve az 1 osztályt (pl. második tulajdonság; 0,3; és a küszöbnél kisebb értékeket becsli 1, a nem kisebbeket 0 osztályúnak). A gyenge osztályozók száma W.

Minden iterációban aggregáljuk a gyenge osztályozók súlyait, hogy a végső hipotézis minden iteráció eredményét felhasználva álljon elő.

A hipotézisek, a valódi kimenetek és az osztályozók súlyainak függvényében kapnak súlyt a mintapontok, majd ebből az eloszlásból N elem mintavételezésével kapjuk a következő iteráció mintapontjait. A t. iterációban az aktuális mintapontok halmaza $\left\{\left(\underline{x}_i^{(t)}, y_i^{(t)}\right)\right\}_{i=1}^{N}$.

A végső hipotézis a gyenge osztályozók kimeneteinek az aggregált súlyukkal képzett súlyozott összege alapján áll elő.

Bemenet: az N elemű tanító mintahalmaz elemei: $(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)$
a gyenge osztályozók: $h_j,\ j \in \{1, \ldots, W\}$
$T \in \mathbb{Z}^+$: az iterációk száma
Inicializálás: a minták súlyvektora: $p_i^{(1)} = \frac{1}{N}\ \forall i \in \{1, \ldots, N\}$
az aktuális mintaponthalmaz elemei: $\left(\underline{x}_i^{(1)}, y_i^{(1)}\right) = (\underline{x}_i, y_i)\ \forall i \in \{1, \ldots, N\}$
az osztályozók aggregált súlyvektora: $\underline{w}_{aggr}^{(1)} = \underline{0}$
for $t = 1, 2, \ldots, T$ **do**

1. Számítsuk ki a gyenge osztályozók hipotéziseit az $\left\{\underline{x}_i^{(t)}\right\}_{i=1}^{N}$-beli mintákra:
   $h_j : \left\{\underline{x}_i^{(t)}\right\}_{i=1}^{N} \to \{0, 1\},\ j \in \{1, \ldots, W\}$
2. A h_j hipotézisek pontossága: $a_j^{(t)} = \sum_{i=1}^{N} 1 - \left|h_j\left(\underline{x}_i^{(t)}\right) - y_i^{(t)}\right|$
   Ebből az oszályozók súlya: $w_j^{(t)} = \frac{a_j^{(t)}}{\sum_{j=1}^{W} a_j^{(t)}}$
3. Az aggregált súly módosítása: $\underline{w}_{aggr}^{(t+1)} = \frac{\underline{w}_{aggr}^{(t)} + \underline{w}^t}{\left\|\underline{w}_{aggr}^{(t)} + \underline{w}^t\right\|}$
4. A mintapontok súlya: $p_i^{(t+1)} = \frac{\sum_{j=1}^{W} w_j^{(t)} \left|h_j\left(\underline{x}_i^{(t)}\right) - y_i^{(t)}\right|}{\sum_{i=1}^{N}\sum_{j=1}^{W} w_j^{(t)} \left|h_j\left(\underline{x}_i^{(t)}\right) - y_i^{(t)}\right|}$
5. Az $\left\{\left(\underline{x}_i^{(t)}, y_i^{(t)}\right)\right\}_{i=1}^{N}$ halmazból a $\underline{p}^{(t+1)}$ eloszlás szerint N elemet véletlenszerűen mintavételezve kapjuk a következő mintaponthalmazt, $\left\{\left(\underline{x}_i^{(t+1)}, y_i^{(t+1)}\right)\right\}_{i=1}^{N}$-et.

end
Kimenet: a végső hipotézis: $h_f(\underline{x})$
if $\sum_{j=1}^{W} h_j(\underline{x}) w_{aggr,j}^{(T+1)} > 0{,}5$ **then**
  $h_f(\underline{x}) = 1$
else
  $h_f(\underline{x}) = 0$
end

Algoritmus 2: Az algoritmus mintavételezést használó változata

5.1.2. Megvalósítás mátrixszorzással

Az előző megvalósításban mindkétféle súlyozáskor azt használtuk fel, hogy melyik osztályozó melyik mintapontot osztályozza jól illetve rosszul. Az osztályozók súlyozásakor a több jól osztályozás jelenti a nagyobb súlyt, míg a mintapontok súlya éppen akkor nagyobb, ha többen osztályozták rosszul. Ha N db mintapontunk és W db gyenge osztályozónk van, akkor a súlyozásokhoz szükséges adatokat két mátrixban foglalhatjuk össze. Az egyik $M \in \mathbb{R}^{N\times W}$, amelynek az elemei $M_{i,j} = 1$, ha az i. mintapontot a j. osztályozó rosszul osztályozza, egyébként 0 ($i, j \in \mathbb{N}, 1 \leq i \leq N, 1 \leq j \leq W$). A másik, $M' \in \mathbb{R}^{W\times N}$ mátrix nagyon hasonló M-hez, úgy kaphatjuk meg belőle, hogy transzponáljuk, és minden elemét kicseréljük az ellenkezőre, azaz $M'_{i,j} = 1 - M_{j,i}$.

Ha a minták elosztását a t. iterációban $\underline{p}^{(t)}$, a gyenge osztályozók súlyát pedig $\underline{w}^{(t)}$ jelöli, akkor $\underline{p}^{(t+1)} = M\underline{w}^{(t)}$ (hiszen az eredményvektor i. koordinátája az i. minta rosszul osztályozásainak az osztályozók súlyával súlyozott száma), és $\underline{w}^{(t)} = M'\underline{p}^{(t)}$ (az egyes osztályozók jól osztályozásainak száma a minták eloszlásával súlyozva). Ezekből $\underline{p}^{(t+1)} = MM'\underline{p}^{(t)}$, vagyis $\underline{p}^{(T)} = (MM')^{T-1}\underline{p}^{(1)}$, ahol $p_i^{(1)} = 1/N,\ \forall i \in \{1, \ldots, N\}$. Vagy hasonlóan az osztályozók súlyaira: $\underline{w}^{(t+1)} = M'M\underline{w}^{(t)}$, vagyis $\underline{w}^{(T)} = (M'M)^{T-1}\underline{w}^{(1)}$, ahol $\underline{w}^{(1)} = M'\underline{p}^{(1)}$.

Így a végső hipotézis: ha $\sum_{j=1}^{W} w_j^{(T)} h_j(\underline{x}) > 0{,}5$, akkor $h_f(\underline{x}) = 1$, egyébként $h_f(\underline{x}) = 0$.

Számítás sajátvektorral

A domináns sajátértékhez tartozó sajátvektor kiszámítására használható a hatványmódszer (más néven hatványiteráció vagy von Mises-eljárás). Eszerint legyen A egy diagonalizálható mátrix, amelynek van a többinél szigorúan nagyobb abszolút értékű sajátértéke; és $\underline{v}_1$ olyan vektor, aminek van nemnulla komponense a keresett sajátvektor irányában. Ekkor ha $\underline{v}_{k+1} = \frac{A\underline{v}_k}{\|A\underline{v}_k\|}$, akkor a $(\underline{v}_k)$ sorozat tart az A domináns sajátértékéhez tartozó sajátvektorához. (Lásd pl. [23].)

Esetünkben $\underline{v}_1 = \underline{w}^{(1)} = M'\underline{p}^{(1)}$, és $\underline{w}^{(T)} = \frac{(M'M)^{T-1}\underline{w}^{(1)}}{\left\|(M'M)^{T-1}\underline{w}^{(1)}\right\|}$. Ezért $\lim\limits_{T\to\infty} \underline{w}^{(T)} = \underline{s}_1$, az $M'M$ mátrix domináns sajátértékéhez tartozó sajátvektor. Tehát a mátrixszorzások vagy hatványozás elvégzése helyett elegendő ezt a sajátvektort meghatározni.

5.1.3. Futási eredmények összehasonlítása

Az osztályozás eredményének minősítésére külön validációs mintakészletet használtunk, ami a teljes adathalmaz véletlenszerűen választott egytized részéből állt (így a maradék 9/10 rész volt a tanító mintakészlet). Ezeken leginkább az ún. AUC értéket figyeltünk, azaz a ROC-görbe alatti területet (0 és 1 közötti érték, a 0,5 a véletlenszerű, az 1 a tökéletes osztályozás eredménye; ezekről a fogalmakról gyors áttekintést nyújt [9]), így ugyanis a konkrét küszöbértéktől független jellemzőt kapunk. Ez azt is jelenti, hogy az 5.1.1-ben és az 5.1.2-ben leírt algoritmusokban a h_f végső hipotéziseknél szereplő 0,5 küszöb valójában nem konkrét szám, azt figyeljük, hogy mennyire jól osztja kétfelé a mintákat az adott algoritmus.

A mátrixszorzásos és a sajátvektoros változat megvalósításánál végül nem 0-1 mátrixokat használtunk, hanem $M_{i,j}$ és $M'_{j,i}$ az i. minta j. gyenge osztályozó által vizsgált tulajdonsá-

gának az osztályozó küszöbértékétől való távolságával arányos. Tehát nem csak azt nézzük, hogy a küszöb megfelelő oldalára esik-e a minta, hanem azt is, hogy mennyire „biztos" a predikcióban az osztályozó.

A mintavételezést és a mátrixszorzást használó változatokban minden iterációban ellenőriztük a validációs mintahalmazon, hogy ha csak addig tartana a tanítás, akkor az úgy kapott súlyozással milyen eredményt adna a „végső" hipotézis. Így képet kaphattunk róla, hogy hogyan alakul a tanulás minősége az iterációk során.

Öt algoritmust vetünk össze: az AdaBoostnak az sklearn ensemble csomagjában található implementációját, a 4.3. szekcióban bemutatott kvantum osztályozó általunk implementált klasszikus változatát, és a saját algoritmusunk három változatát (mintavételezéses, mátrixszorzásos, sajátvektoros). A futásidők számításához $T = 10$ értékkel futtattuk a mintavételezéses és a mátrixszorzásos változatot.

Mivel az adathalmazt véletlenszerűen vágjuk tanító- és validációs készletre (sőt a mintavételezéses megoldásnál a mintavételezés is véletlenszerűen történik), a különböző futtatások eltérő eredményt adhatnak. Ezért minden esetben két jellemző, de valamennyire különböző eredményt sorolunk fel, valamint nagyjából átlagos futásidőket.

Több, nyilvánosan elérhető adathalmazon is futtattuk az algoritmusokat, így összehasonlíthatjuk az eredményeket (5.1. és 5.2. táblázat).

- Cleveland[1]: szívbetegségek felismerése a feladat. Általában jobb eredményt adnak a saját módszereink az AdaBoostnál, akár jelentősen is.
- Banknote[1]: valódi és hamis bankjegyek megkülönböztetése a feladat. Néhány módszer 1 körüli (tökéletes) eredményt ad, de a többi is elég magas értéket.
- Farm[2]: nagyméretű adathalmaz, mely néhány, farmokkal kapcsolatos weboldalon talált reklámról szól. Az AdaBoost teljesít a legjobban, de a saját módszereink közel járnak hozzá. A kvantum osztályozó határozottan rosszabb eredményt ad.

algoritmus\adathalmaz	Cleveland	Banknote	Farm
AdaBoost	0,73; 0,87	0,99; 1,00	0,90; 0,92
Kvantum osztályozó	0,93; 0,89	0,94; 0,92	0,68; 0,69
Mintavételezéses	0,94; 0,91	0,99; 1,00	0,87; 0,85
Mátrixszorzásos	0,90; 0,88	0,92; 0,89	0,83; 0,82
Sajátvektoros	0,90; 0,88	0,91; 0,87	0,83; 0,82

5.1. táblázat. *AUC értékek a különböző adathalmazokon*

Az eredményekből látható, hogy a saját megoldások közül a mintavételezéses osztályoz a legmegbízhatóbban, azonban legtöbbször ez a legmagasabb futásidejű is (bár minden bizonnyal lehetne optimalizálni a kódját). Az is egyértelmű, hogy bizonyos adatokra bármelyik megoldásunk lehet jobb, mint az AdaBoost, azonban általában nem ez a helyzet. A mintavételezéses megoldás viszont szinte mindig vetekszik az AdaBoosttal, vagy jobb annál.

[1] `https://jamesmccaffrey.wordpress.com/2018/03/14/datasets-for-binary-classification/`
[2] `https://archive.ics.uci.edu/ml/datasets/Farm+Ads`

algoritmus\adathalmaz	Cleveland	Banknote	Farm
AdaBoost	0,1 s	0,2 s	440 s
Kvantum osztályozó	0,3 s	0,3 s	330 s
Mintavételezéses	4,1 s	5,1 s	6110 s
Mátrixszorzásos	0,7 s	0,8 s	890 s
Sajátvektoros	0,5 s	5,3 s	1090 s

5.2. táblázat. *Futásidők a különböző adathalmazokon*

5.2. Kvantumos megvalósítás

A kvantum osztályozó algoritmus mintájára terveztünk egy kvantumalgoritmust, aminek működése alapvetően megfelel az előző szekcióban bemutatott klasszikus algoritmusénak. A gyenge osztályozók most lehetnek döntési tönkök helyett például egyszintű kvantum döntési fák.

5.2.1. A működés vázlata

Az algoritmus első fő lépése az osztályozók, a második a mintapontok súlyozása. Az első azonosan működik a kvantum osztályozó algoritmussal, de a második lépést is ehhez hasonlóan végezhetjük el. A két lépést néhányszor iteratívan ismételjük.

A kiinduló állapot a kvantum osztályozónál látottnak a duplikáltja, azaz $\delta + 1$ qubites adatregiszter, τ kvantumbites paraméterregiszter és egy-egy eredmény- illetve pontosság-qubit tartozik mind az első, mind a második lépéshez:

$$|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \otimes |0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \tag{5.1}$$

Az eredmény- és a pontosság-qubit használható közösen, itt csak a könnyebb érthetőség kedvéért bontjuk szét ilyen egyértelműen két részre a kvantumregisztereket.

Első lépés: (5.2)→(5.3). A 4.3. szekció kvantum osztályozójának tanításával megegyezően történik.

$$\frac{1}{\sqrt{E}}\sum_{i=0}^{2^\tau-1}|0\ldots0;0\rangle\,|i\rangle\,|0\rangle\,\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \tag{5.2}$$

$$\downarrow$$

$$\frac{1}{\sqrt{\chi_1 E}}\sum_{\theta}|0\ldots0;0\rangle\,\sqrt{a_\theta}\,|\theta\rangle\,|0\rangle\,|0\rangle\otimes|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \tag{5.3}$$

Második lépés: (5.4)→(5.5). Miután előállítottuk a tanító mintahalmaz N db elemének egyenletes szuperpozícióját az adatregiszterben, egyesével betöltjük a gyenge osztályozókat (azok paramétervektorait) a paraméterregiszterbe, és kiszámítjuk a kimenetüket. Ezeknek és a tényleges címkéknek a különbözőségétől vagy azonosságától függően rendre a $|1\rangle$ vagy a $|0\rangle$ irányába forgatjuk a második lépés pontosság-qubitjét az adott osztályozó súlyának megfelelő mértékben. Végül megmérjük ezt a pontosság-qubitet, és csak a különbözőséghez tartozó állapotát ($|1\rangle$) fogadjuk el. Így a minták nehezen tanulhatóságával arányos

súlyozású szuperpozíció alakul ki az adatregiszterben.

$$\frac{1}{\sqrt{\chi_1 E}} \sum_{\theta} |0 \dots 0; 0\rangle \sqrt{a_\theta} |\theta\rangle |0\rangle |0\rangle \otimes \frac{1}{\sqrt{N}} \sum_{|x;y\rangle} |x; y\rangle |0 \dots 0\rangle |0\rangle \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{5.4}$$

$$\downarrow$$

$$|0 \dots 0; 0\rangle |0 \dots 0\rangle |0\rangle |0\rangle \otimes \frac{1}{\sqrt{\chi_2 N}} \sum_{|x;y\rangle} \sqrt{w_{(x;y)}} |x; y\rangle |0 \dots 0\rangle |0\rangle |1\rangle \tag{5.5}$$

Ezután már újra elvégezhetjük az első lépést a mintákon előálló eloszlásból való mintavételezéssel (mérésekkel) kapott új mintákon. Nincs szükség mérésekre, ha meg tudjuk oldani, hogy a minták súlyával arányos mértékben forgassuk az első lépéshez tartozó pontosság-qubitet.

5.2.2. Felmerülő problémák

A kvantumos megvalósítással kapcsolatos kutatásunk még folyamatban van, ezért jelen pillanatban több problémával is szembesülünk. A problémák azzal kapcsolatosak, hogy hogyan tudunk hozzáférni az egyes lépések végére megkapott súlyokhoz.

- Ha az utolsó lépés végére előálló súlyozás szerint kombináljuk a gyenge osztályozókat, az valószínűleg rosszabb eredményt ad, mintha az összes korábbi súly aggregáltjával számolnánk. Az aggregálás megoldása még kérdéses.
- Hogyan tudjuk a pontosság-qubiteket az előző lépés végére kapott súlyokkal arányosan forgatni?

Összegzés

A tárgyalt területek jelentőségét mutatja, hogy napjainkban a figyelem középpontjába került a gépi tanulás és a kvantuminformatika is. A részletesebben megvizsgált módszerek segíthetnek való életbeli problémák hatékony megoldásában.

A dolgozat jelentős része osztályozási problémák megoldásáról szólt. Bár többször is csak bináris osztályozásról ejtettünk szót, ezek mindig kiterjeszthetők több osztály esetére, legegyszerűbben például úgy, hogy ha k féle címke van, akkor $O(\log_2 k)$ db bináris osztályozási feladatra bontjuk az eredeti problémát – és ezeket már meg tudjuk oldani.

Rengeteg, nagy gyakorlati jelentőséggel bíró probléma megfogalmazható osztályozási feladatként. Például az önvezető járműveknek a beérkező adatok alapján el kell tudniuk dönteni, hogy minden rendben van-e, és ha nem, akkor hogyan kell reagálni az adott helyzetre. Úgy is mondhatjuk, hogy az adatokkal leírt helyzetet osztályozniuk kell aszerint, hogy milyen reakciót igényelnek – és aztán ennek megfelelően avatkoznak be.

A dolgozatban bemutatott kvantumalgoritmusok felhasználásával az ilyen feladatok megoldása – kellő kapacitású és stabilitású kvantumszámítógépek kereskedelmi megjelenésével – gyorsabbá válhat. A kvantum döntési fákkal kapcsolatos eredményeink gyorsabb, egyszerűbb számítást tesznek lehetővé. Az új osztályozó algoritmusunk klasszikus változata pedig bizonyos problémáknál pontosabb osztályozást érhet el az addig használt módszereknél, ezt láttuk a futtatási eredményeknél.

Továbblépési lehetőségek

A saját algoritmusunk megvalósításainak optimalizálása mellett azt is ki szeretnénk deríteni, hogy mitől függ, hogy melyik adathalmazokon melyik módszerek érnek el jobb eredményt. A kvantumos változatnál már megemlítettünk néhány problémát, amelyek a megvalósíthatóság szempontjából fontosak.

Első lépésként azt szeretnénk megvizsgálni, hogy a különböző kvantum ensemble módszerek milyen geometriai alakzatok felismerésére alkalmasak. Azaz pl. egy síkbeli négyzetrács pontjai a minták, amelyek tulajdonságai a vízszintes és a függőleges koordinátájuk, címkéjük pedig 1, ha az alakzatnak része az adott pont, egyébként 0. Kérdés, hogy milyen alakzatok esetén tudja elég jól megtanulni az osztályozó a pontok címkéjét.

Köszönetnyilvánítás

Elsősorban természetesen a konzulensemnek, Friedl Katalinnak tartozom hálával. Rajta kívül Daróczy Bálint és Kabódi László is rengeteget segítettek a gondolataikkal, támogatásukkal. Külön kiemelem Bálint remek ötleteit, amelyek nélkül biztosan nem születhetett volna meg az új osztályozó algoritmus.

Köszönöm az Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutatóintézetének (MTA-SZTAKI), hogy használhattam az infrastruktúrájukat, valamint fontos megemlíteni, hogy a náluk végzett munkámhoz is kötődnek az eredmények.

Ezenkívül az anyagi támogatásért köszönetet jár az Emberi Erőforrások Minisztériuma Új Nemzeti Kiválóság Programjának.

Emberi Erőforrások
Minisztériuma

Az Emberi Erőforrások Minisztériuma ÚNKP-18-1 kódszámú Új Nemzeti Kiválóság Programjának támogatásával készült

Irodalomjegyzék

[1] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor és Valyon József. *Neurális hálózatok*. Panem Könyvkiadó Kft., 2006.

[2] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 161–168. Curran Associates, Inc., 2008.

[3] David Deutsch. Quantum computational networks. In *Proceedings of The Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, volume 425, pages 73–90, 09 1989.

[4] Szabó Dániel. Gráfelméleti algoritmusok beágyazása D-Wave kvantumszámítógépbe. In *Tudományos Diákköri Konferencia*. BME-VIK, 2017.

[5] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[7] Martin Giles and Will Knight. Google thinks it's close to "quantum supremacy." here's what that really means. *MIT Technology Review*, March 2018.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] Google Developers – Machine Learning. Classification: ROC and AUC (2018. 10. 21.). `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc`.

[10] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[11] Mika Hirvensalo. *Quantum Computing*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2004.

[12] Will Knight. IBM raises the bar with a 50-qubit quantum computer. *MIT Technology Review*, November 2017.

[13] Breiman LM, Jerome Friedman, Richard A. Olshen, and Charles J. Stone. *CART: Classification and Regression Trees*. 01 1983.

[14] Songfeng Lu and Samuel L. Braunstein. Quantum decision tree classifier. *Quantum Information Processing*, 13(3):757–770, March 2014.

[15] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[16] Amy Nordum. China demonstrates quantum encryption by hosting a video call. *IEEE Spectrum*, October 2017.

[17] Maria Schuld and Francesco Petruccione. Quantum ensembles of quantum classifiers. In *Scientific Reports*, volume 8, February 2018.

[18] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 124–134, Washington, DC, USA, 1994. IEEE Computer Society.

[19] D-Wave Systems. What is quantum annealing? (2018. 10. 06.). `https://www.youtube.com/watch?v=zvfkXjzzYOo`.

[20] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg, 1995.

[21] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, Dec 2001.

[22] Wikipedia. Machine learning (2018. 09. 28.). `https://en.wikipedia.org/wiki/Machine_learning`.

[23] Wikipedia. Power iteration (2018. 10. 21.). `https://en.wikipedia.org/wiki/Power_iteration`.

[24] Wikipedia. Random forest (2018. 10. 15.). `https://en.wikipedia.org/wiki/Random_forest`.

[25] Wikipedia. Timeline of quantum computing (2018. 10. 06.). `https://en.wikipedia.org/wiki/Timeline_of_quantum_computing`.