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

Kvantuminformatika és gépi tanulás

Szakdolgozat

Készítette
Szabó Dániel

Konzulensek
Dr. Friedl Katalin
Dr. Daróczy Bálint Zoltán

2018. december 6.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Szabó Dániel*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2018. december 6.

Szabó Dániel
hallgató

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 két terület ötvözése rendkívül ígéretesnek tűnik.

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.

A kanadai D-Wave Systems vállalat volt az első a világon, amely kvantumszámítógépet adott el, a legutóbbi számítógépük 2048 kvantumbites. Bár megoszlanak a vélemények arról, hogy valóban kvantumos elven működő számítógépekről van-e szó, ettől függetlenül érdemes foglalkozni azzal a számítási modellel, amely a gépek mögött van. A modell alapja a Kiméra gráf (Chimera graph), ebbe kell beágyazni a problémákat.

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 adatokat 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.

A dolgozat fő célja egyrészt a különböző kvantuminformatikai számítási modellek ismertetése és néhány NP-teljes gráfelméleti probléma D-Wave kvantumszámítógépen történő megoldásának bemutatása. Másrészt felvázoljuk bizonyos gépi tanulási módszerek kvantum változatát, és egy új bináris osztályozó algoritmust is terveztünk, amely megalkotásánál szempont volt a kvantumos megvalósíthatóság is. A téma jelentősége abban áll, hogy a bemutatott problémák hatékonyabb megoldása válhat lehetővé egy jövőbeli kvantumszámítógépen.

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 a growth in the importance of quantum algorithms in the near future. Machine learning is also a very popular area, and it is used in various fields, e.g. chess playing machines and self-driving cars. The mixture of these fields seems extremely promising.

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 all the states 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.

The Canadian D-Wave Systems was the first company in the world to sell quantum computers. Their latest model is a 2048-quantum bit system. Although it is controversial that these computers really work by quantum principles, the computational model that they provide is worth to investigate. This model is based on the Chimera graph, in which we should embed our problems.

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.

The main goal of this paper on the one hand is to review the different models of quantum computing and to show how to solve some NP-complete problems on a D-Wave quantum computer. On the other hand we present the quantum versions of some machine learning techniques and we also designed a new binary classifier algorithm, keeping an eye on a possible realization as a quantum algorithm. The significance of this field is that in the future it may be possible to solve the presented problems more efficiently on a quantum computer.

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 [Nor17]. Ennek az a jelentősége, hogy a kulcsszétosztás a szimmetrikus kulcsú kriptográfia alapvető lépése, és kvantumkriptográfiával matematikailag bizonyítottan feltörhetetlen 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 kvantumbitek állapotát maximum 100 mikroszekundum ideig tudják fenntartani, és a zajra érzékenység miatt nehéz megmondani, hogy valóban jó eredményt kaptunk-e [GK18, Kni17].

A kvantuminformatika egy másik modelljén (lehűtéses modell) alapulnak a D-Wave Systems cég kvantumszámítógépei. Legutóbbi, 2017-es gépük már 2048 qubites, ez azonban nem vetendő össze az előző bekezdésben említett cégek számítógépeivel, mivel más modellre épülnek. Ez a modell optimalizálási és mintavételezési problémák megoldására alkalmas. A dolgozatban néhány NP-teljes gráfelméleti problémát oldunk meg hatékonyan a segítségével.

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 [VJ01]) 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 kutatók, 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 I. részében a két érintett tudományterület alapismereteit foglaljuk össze. Az 1. fejezetben egy rövid kvantuminformatikai ismertető olvasható, a 2. fejezet pedig a gépi tanulás területét mutatja be – természetesen közel sem kimerítően. Ezután kezdődik a II. rész, amely a lehűtéses modellről szól. A 3. fejezet azokat a beágyazási módszereket ismerteti, amelyeket felhasználtam a konkrét problémák beágyazására. Ezek a gyakorlati alkalmazásokban is előforduló alapvető feladatok kerülnek sorra a 4. fejezetben: maximális klikket, maximális teljes páros részgráfot, és minimális domináns halmazt kerestem tetszőleges gráfokban. Ezek mellett egy általam írt programról is szó lesz, amellyel ellenőrizhetjük a beágyazások helyességét.

A III. rész néhány gépi tanuló módszerről és azok kvantumos megvalósításáról szól. Ebben megvizsgálunk bizonyos konkrét gépi tanulási módszereket, és ezek kvantumos változatait: az 5. fejezet a döntési fákkal, a 6. fejezet pedig az ensemble módszerekkel foglalkozik mind klasszikus, mind kvantumos értelemben. Az 5.3. szekcióban néhány új, kvantum döntési fákkal kapcsolatos eredményemet közlöm. A 7. fejezet arról az új algoritmusról szól, amelyet az addig bemutatott eredmények felhasználásával terveztünk meg és implementáltunk. Néhány adathalmazon kipróbáltam több algoritmust – köztük az újat is – és összevetettem az eredményeket. Az új algoritmussal kapcsolatban is szóba kerül a kvantumos megvalósítás.

A dolgozat tartalma főleg a 2017. és a 2018. évi TDK dolgozataimból áll [Sza17, Sza18].

I. rész

A kvantuminformatika és a gépi tanulás alapjai

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 [Fey82]), 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 [Wikg].

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 interferenciakép 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.

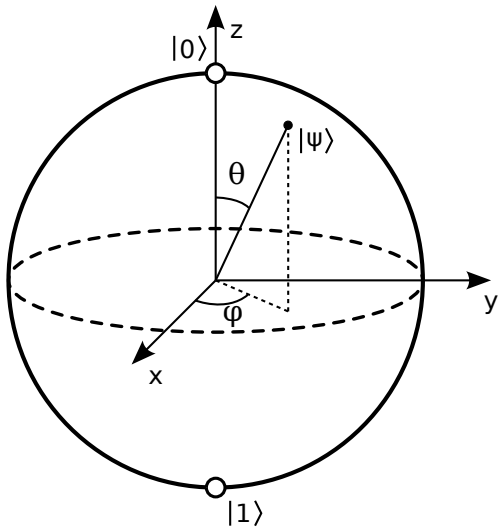
I. posztulátum

Zárt fizikai rendszer állapota egy $\mathcal{H}$ Hilbert-térbeli egységhosszú vektorral írható le (az-az 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 kétdimenziós $\mathbb{C}^2$ 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. Egy kvantumbit a $\mathbb{C}^2$ egy eleme, ezért alapvetően négy független valós paraméterrel írható le. Azonban tudjuk, hogy a normája 1; valamint hogy az ún. globális fázis nem számít (azaz $|\psi\rangle$ és $e^{i\theta}\,|\psi\rangle$ – ahol $\theta \in \mathbb{R}$ – azonos állapotot ír le), kettő paraméter is elég. A Bloch-gömb két átellenes pontja jelképezi a két standard bázisvektort, a felületén helyezkednek el a kvantumbitként interpretálható tiszta állapotok (amelyek nem állnak elő más állapotok konvex kombinációjaként), a belsejében pedig a kevert állapotok (a nem tiszta állapotok, amelyek qubitek konvex kombinációjaként állnak elő, és sűrűségoperátorokkal írhatók le). A továbbiakban tiszta állapotokkal (kvantumbitekkel) foglalkozunk.



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 egységhosszúak

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 áramkörökhöz 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 $\mathrm{POVM}(\mathcal{X}, \mathcal{H}) = \{M = \{M_x\}_{x\in\mathcal{X}} : M_x \geq 0,\ \#\{x : M_x \neq 0\} < \infty,\ \sum_{x\in\mathcal{X}} M_x = I\}$.

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

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

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 qubit 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. Ebből 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.

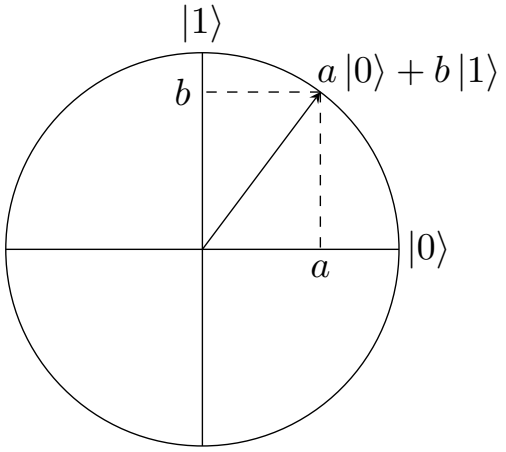
Ezeknek az érdekessége az, hogy ha a kvantumbitpár egyik tagját megmérjük, akkor ennek hatására a másik qubit is egy klasszikus állapotba kerül, mégpedig az első qubit mérésének eredménye egyértelműen meghatározza, hogy melyik állapotba. A fenti esetben például biztos, hogy mérés hatására mindkét kvantumbit 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 egységhosszú 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 qubitet 0-nak mérjük, a^2; annak pedig, hogy 1-nek mérjük, $1 - a^2 = b^2$.

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



1.2. ábra. *Kvantumbit egyszerűsített ábrázolása*

végzünk, akkor már nem alakul ki interferenciakép, csak azon a résen halad át a részecske, amelyiknek a detektora jelez.

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

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 [Sho94]. 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 [Gro96].

1.3. Számítási modellek

Több modell alapján is megvalósíthatók kvantumszámítógépek. Először az áramköri modell alakult ki Deutsch 1989-es cikke [Deu89] nyomán. Ez vált a standard modellé.

1.3.1. Áramköri modell

Kvantum logikai kapuk sorozatából áll elő a számítás, alapvetően erről szólt a fejezet eddigi része. A kvantumbitek működését nehéz összehangolni, és a qubitek számának növekedésével azok állapotának stabilitása csökken, ezért egyelőre csak kevés qubites 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.

1.3.2. Lehűtéses modell

A lehűtéses számítási modell optimalizálási és mintavételezési problémák megoldására alkalmas. Azt használja ki, hogy a fizikai rendszerek mindig az energiaminimumra törekednek. Itt nem mi befolyásoljuk a működést, hanem a probléma betáplálása után hagyjuk, hogy a fizikai törvényszerűségeknek megfelelő változások történjenek. A D-Wave Systems cég gépei ezt a modellt követik, és nem univerzálisak [D-Wd].

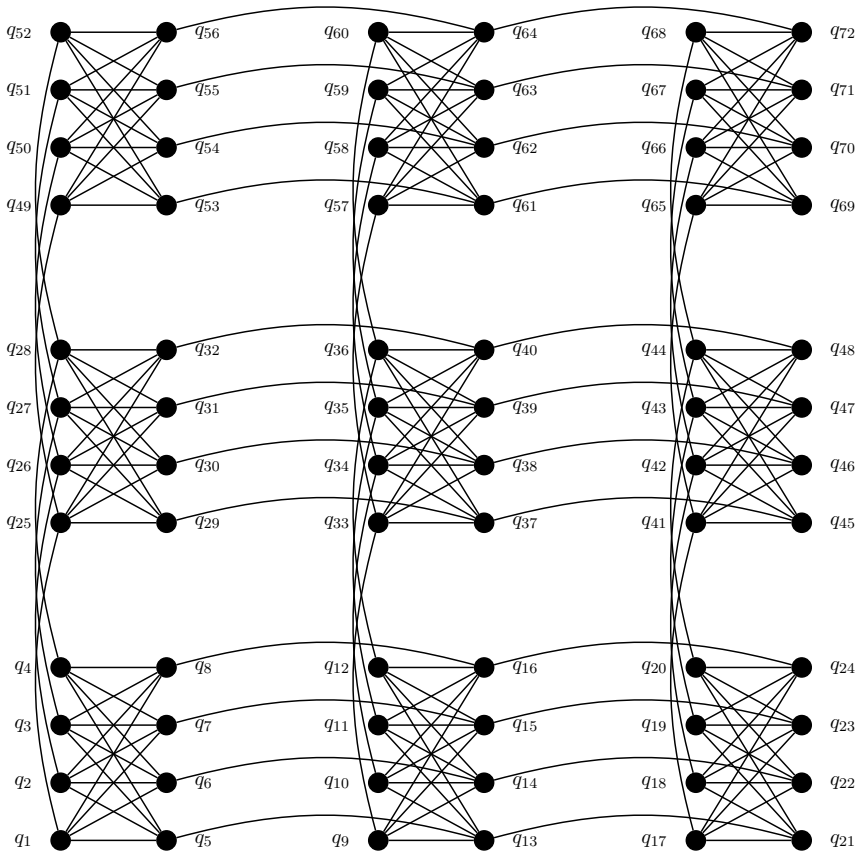
A *Hamilton-függvény* olyan függvény, amelybe, ha beírjuk a kvantumbitek értékeit, akkor az eredménye ezen állapot energiaszintje lesz. Kezdetben például minden qubit azonos valószínűséggel 0 és 1, ez az állapot könnyen előállítható Hadamard-transzformációval. Ekkor a paramétereket (csúcs- és élsúlyok) figyelmen kívül hagyó kezdeti Hamilton-függvény számít, ebben az esetben pedig ez a minimális energiájú állapot. Végül a paramétereket figyelembe vevő végső Hamilton-függvény értéke lesz a mérvadó. Az így előálló legalacsonyabb energiaszintű állapot hordozza magában a megoldást. Ekkor már klasszikus bitekről beszélhetünk, ugyanis már nincsenek szuperpozícióban [D-Wc].

D-Wave

A D-Wave egy kanadai cég, amelyet 1999-ben alapítottak, és ez lett a világ első olyan vállalata, ami kvantumjelenségeket kihasználó számítógépeket árult. 2004-ben döntöttek úgy, hogy kvantumszámítógépet építenek. Eleinte magas hőmérsékletű szupravezetőkkel próbáltak megvalósítani kvantumbitet – ezek egyik fatáját hívják d-wave-nek: innen származik a cég neve. 2010-ben adták ki az első kereskedelmi gépüket, a D-Wave One-t, ami 128 kvantumbites volt. A legutóbbi számítógépük a 2017-es D-Wave 2000Q, ami 2048 qubites. A számítógépeiket használó szervezetek között olyanokat találunk, mint a Volkswagen, a Google vagy a NASA [Wika, D-Wb].

A D-Wave gépei lehűtéses kvantumszámítógépek. A számítási modelljük alapja a Kiméra[1] gráf (Chimera graph), amely 4+4 csúcsú teljes páros gráfok ($K_{4,4}$) összekötéséből és rácsba rendezéséből áll (1.3 ábra). A kvantumbitek megfeleltethetők a csúcsoknak, az élek pedig az összefonódással állnak kapcsolatban. A qubitek fizikai megvalósítása szupravezetőből (nióbium) készült körökkel, és az ezekben folyó áramok által indukált kis mágneses

[1]Kiméra (Khimaira) a görög mitológia alakja, többek között Szphinx, Kerberosz és a lernéi Hüdra testvére. Jellegzetessége, hogy több állat különböző testrészeiből áll: oroszlán, aminek hátából kecskefej nő ki, a farka helyén pedig kígyó leng. A gráf elnevezésének oka az lehet, hogy különálló $K_{4,4}$ gráfokból áll össze. [Forrás: Wikipédia. Chimera (mythology) (2018. 11. 24.). `https://en.wikipedia.org/wiki/Chimera_(mythology)`.]



1.3. ábra. *Kiméra gráf*

mezőkkel történik [D-Wa].

A csúcsokhoz és az élekhez is rendelhetők súlyok, amelyek minden lehetséges esethez meghatározzák az energiaszintet, és így azt, hogy mit tekintünk a legjobb megoldásnak. A problémák beágyazása egy ilyen gépbe tulajdonképpen ezen súlyok értékeinek meghatározását jelenti. A megoldást valószínűségi alapon kapjuk: a legalacsonyabb energiaszintű állapot előállásának a legnagyobb a valószínűsége. Ezért érdemes többször futtatni, és a kapott eredmények eloszlását figyelni.

Már a D-Wave One megjelenésétől vannak, akik szerint ez valójában nem is igazi kvantumszámítógép, ugyanis nem látják bizonyítottnak, hogy tényleg kihasznál kvantummechanikai jelenségeket a megoldás meghatározásához. A cég publikált bizonyítékokat, de azok csak közvetve mutatták ki az összefonódás jelenlétét [L$^+$14]. A kétkedés mértéke 2007 óta csökkent, de nem halt el teljesen [Wika].

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 [Alt06].

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 para-

métereket 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, ami meghatározza, hogy adott bemenethez milyen kimenetet rendel. Ez a tanulás iterációi során 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 az $y_i = f(\underline{x}_i; \underline{w})$ 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 problémáról 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 $g(\underline{x})$ 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 [Vap95]. 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 (overfitting). 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 [BB08].

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.

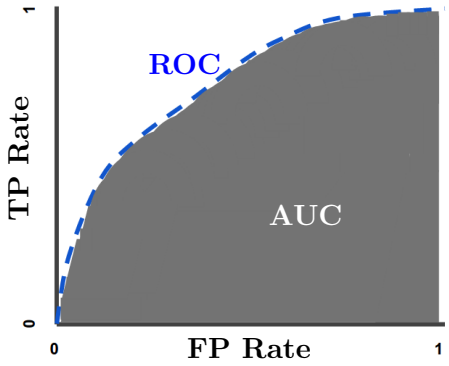
AUC

Amikor egy bináris osztályozó algoritmust minősítünk, kézenfekvőnek tűnik azt figyelni, hogy a tesztkészlet mintáinak mekkora hányadát osztályozta jól. Azonban az osztályozás valamilyen küszöbérték alapján történik: ha a vizsgált érték kisebb nála, akkor az egyik, egyébként a másik osztályba sorolódik a minta. Így előfordulhat, hogy a küszöbérték kis mértékű módosítása jelentős mennyiségű minta átsorolódását eredményezi. Ezért célszerűbb lehet egy küszöbértéktől független jellemzőt választani a minősítésre. Ilyenkor csak azt figyeljük, hogy a vizsgált érték alapján sorbarendezve a mintákat mennyire kerülnek egymás mellé az egyes osztályok mintái.

Ilyen jellemző például az AUC, ami a ROC-görbe (Receiver Operating Characteristic curve) alatti terület nagysága (2.1. ábra). A ROC-görbe azt mutatja meg, hogy a küszöbérték változásával hogyan változik a helyesen pozitívnak jelzett minták aránya az összes pozitív (TPR – True Positive Rate), valamint a tévesen pozitívnak jelzett minták aránya az összes negatív minta között (FPR – False Positive Rate). Ezért a ROC-görbe monoton nő; 0-ból indul, mert kezdetben nagyon magas a küszöb, és így minden mintát negatívnak jelez az osztályozó; végül pedig 1-be érkezik, mivel ekkor az alacsony küszöb miatt már az összes mintát pozitívnak prediktálja.

Az AUC (Area Under the ROC Curve) tehát a ROC-görbe alatti terület nagysága. Legrosszabb esetben $1/2$ az értéke, azaz a ROC-görbe nagyjából az $y = x$ függvénynek feleltethető meg. Ekkor ugyanis ahogy a küszöb csökkentésével egy-egy új minta kerül a

pozitívan osztályozottak közé, az azonos eséllyel (felváltva) lesz valójában pozitív illetve negatív. A legjobb esetben az AUC értéke 1 (vagy 0: ekkor a két osztályt felcserélve már 1-et kapunk), ami annak felel meg, hogy a TPR már FPR = 0-nál 1-re ugrik. Azaz ahogy egyre több elemet osztályozunk pozitívnak, azok mind valóban pozitívak egészen addig, amíg el nem érjük a pozitív és a negatív minták közötti küszöböt. Azonban a TPR ezután is 1 marad, az FPR pedig 1-re ugrik.



2.1. ábra. *A ROC-görbe és az AUC (az eredeti kép forrása [Goo])*

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 az 5. 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 [Alt06] 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.

II. rész

A kvantuminformatika lehűtéses modellje

3. fejezet

Beágyazási módszerek

Ebben a fejezetben az 1.3.2. alszekcióban leírt lehűtéses modell néhány általános beágyazási módszerének ismertetése következik, amelyeket felhasználtam a konkrét problémákhoz. Ezeket néhol példák is illusztrálják, amelyek segíthetik a megértést.

A D-Wave kvantumszámítógépeinek programozása a hagyományos programozástól jelentősen eltér. Például egy ilyen gépnek nincs memóriája, és így állapota sincs, valamint a működése valószínűségi alapú, nem pedig determinisztikus [Dah13]. Olyan módon kell beletáplálni a problémát, hogy meg tudja oldani azt az energiaminimalizálásra törekedve. Az energiaszintet leíró Hamilton-függvény [RVO+15]:

$$H(s) = A(s)H_I + B(s)H_P = A(s)H_I + B(s)\left(-\sum_{i=1}^{N} a_i q_i + \sum_{(i,j)\in E_C} b_{ij} q_i q_j\right).$$

Az egyenletben az $s \in [0,1]$ az idő múlását jelzi. H_I a kezdeti, H_P a végső Hamilton-függvény. Az $A(s)$, $B(s)$ együtthatók határozzák meg H_I illetve H_P súlyát a pillanatnyi energiaszintben, $A(0) = B(1) = 1$ és $A(1) = B(0) = 0$. A kvantumbitek (avagy a Kiméra gráf csúcsainak) száma N, ezek közül az i-ediket q_i jelöli, aminek a súlya a_i. A Kiméra gráf éleinek halmaza E_C; az i-edik és a j-edik qubit közötti él (i,j), aminek a súlya b_{ij}.

Amikor egy potenciális megoldást kapunk, akkor a kvantumbitek már klasszikusnak tekinthetők: bebillentek a 0 vagy az 1 állapotba. Ezért $q_i \in \{0,1\}$ teljesülni fog, azaz összesen 2^N darab lehetőség van. A cél, hogy ezek közül a „jók" esetén kisebb legyen H_P értéke, mint a kevésbé jók esetén.

3.1. Közvetlen beágyazás

Közvetlen beágyazás alatt azt értjük, hogy a csúcs- és élsúlyokat direkt módon, a megoldandó problémától függően határozzuk meg. A lehűtési folyamat végén kapott eredményekből (tehát a minimális energiaszinthez tartozó bitértékekből) valahogy ki kell derülnie annak, hogy mi is a megoldás, így ezt kell kódolnunk a qubitekbe. A megoldandó probléma meghatároz bizonyos kényszereket, amelyeknek mindenképp teljesülni kell egy megoldásban. Ezeket is be kell táplálnunk a gépbe, erre használjuk a súlyokat.

A későbbiekben érdemes különbséget tenni a logikai és a fizikai kvantumbitek és élek

között. A fizikai qubitek és élek azok, amik a Kiméra gráfban megjelennek mint csúcsok, illetve élek. H_P képletében fizikai kvantumbitekről van szó. Sok esetben nem elegendők a Kiméra gráf struktúrájából következő fokszámok a qubitekhez, ezért vezetjük be a logikai kvantumbitek fogalmát. Egy logikai kvantumbit több fizikaiból állhat, amelyek értékeinek meg kell egyezni. Ennek biztosításához szükséges, hogy az egy logikai qubitet alkotó csúcsok összefüggő részgráfot alkossanak a Kiméra gráfban. A logikai qubitek között futnak a logikai élek. Egy logikai él megvalósításához elég, ha a két logikai qubit egy-egy fizikai kvantumbitje között fut egy fizikai él.

A következő példa forrása a D-Wave egyik hivatalos kiadványa [Dah13].

A térképszínezés problémáról lesz szó. A feladat az, hogy az egyes területeket kiszínezzük a lehetséges C db szín egyikével úgy, hogy azok a területek, amelyeknek van közös határszakaszuk (nem csak különálló pontokban érintkeznek), különböző színt kapjanak. A beágyazás több lépésből fog állni: először a területek színét kell elkódolnunk, majd a szomszédos területekre vonatkozó feltétel teljesülését biztosítjuk.

A színek elkódolása

Minden területhez tartozzon C db logikai kvantumbit, amelyeket megfeleltetünk a színeknek. (Azért nem lehet fizikai qubiteket használni, mert a kényszerek biztosításához szeretnénk, ha az egyes színek kvantumbitjei teljes gráfot alkotnának. Ez fizikai qubitekkel a Kiméra gráf struktúrája miatt már $C = 3$ esetben sem lehetséges.) Az a logikai qubit legyen 1, ami a terület színének felel meg, a többi legyen 0. Ezzel olyan kényszert kaptunk, hogy néhány kvantumbit közül egyszerre pontosan egy legyen 1. Elsőként a legkönnyebben megérthető $C = 2$ és $C = 3$ eseteket nézzük, utána pedig az ezekből szerzett tapasztalatok segítségével egy általános megoldást is adunk.

Két qubit (azaz két szín: $C = 2$) esetén H_P-nek a most vizsgált kényszerből következő, egy területre vonatkozó része $E = -(a_1q_1+a_2q_2)+b_{12}q_1q_2$ alakú lehet. Ebben az a_1, a_2, b_{12} súlyokat úgy kell megválasztani, hogy az E függvény értéke kisebb legyen $q_1 \neq q_2$ esetén, mint egyébként. E értékei esetekre bontva:

- ha mindkét qubit 0, akkor $E = 0$;
- ha az i-edik qubit ($i \in \{1, 2\}$) lesz 1, a másik 0, akkor $E = -a_i$;
- ha mindkettő 1, akkor $E = -a_1 - a_2 + b_{12}$.

Például az $a_1 = a_2 = 1$; $b_{12} = 2$ választás esetén a „jó” megoldások energiaszintje -1, a „rosszaké” 0, tehát ez a paraméterezés megfelelő.

A továbbiak egyszerűsítése kedvéért már most is megfogalmazhatjuk egy igényünket. Több kvantumbit esetén is az számít jó megoldásnak, ha pontosan egy olyan i létezik, amelyre $q_i = 1$, a többi qubit értéke 0. Azt szeretnénk, hogy a különböző jó megoldások azonos eséllyel szerepeljenek. Mivel az esélyt az esetnek megfelelő energiaszint határozza meg, ami a jó megoldások esetén $-a_i$ lesz, ezért feltesszük, hogy $a_1 = a_2 = \cdots = a$. Hasonlóan, szimmetriaokok miatt (két kvantumbit felcserélésével ne változzon meg az eredmény) legyen minden élsúly is azonos: $b_{12} = b_{13} = \cdots = b$.

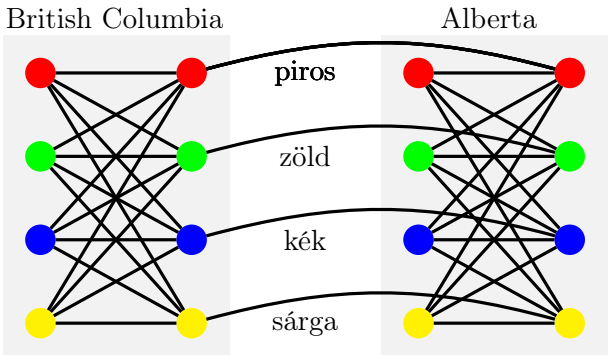
$C = 3$ esetén az idetartozó energia: $E = -a(q_1 + q_2 + q_2) + b(q_1q_2 + q_1q_3 + q_2q_3)$.

- Ha minden qubit 0, akkor $E = 0$;
- ha pontosan egy qubit 1, a többi 0, akkor $E = -a$;
- ha pontosan két qubit 1, a harmadik 0, akkor $E = -2a + b$;
- ha mindhárom kvantumbit 1, akkor $E = -3a + 3b$.

Itt is megfelelő paraméterezés az $a = 1$; $b = 2$: a jó esetekre -1, a rosszakra 0 vagy 3 az energiaszint. Könnyen látszik, hogy a színek számának növelésével, azaz a $C \geq 4$ esetben is ugyanígy eljárva jó megoldást kapunk az $a = 1$; $b = 2$ választással.

Most már beágyazhatjuk a Kiméra gráfba a területek színeit: minden egyes terület feleljen meg a Kiméra gráf egy $K_{4,4}$ teljes páros részgráfjának. A $K_{4,4}$ egy-egy sora (két-két fizikai kvantumbit) feleljen meg az egyes színeknek. (A négyszín-tétel miatt négy szín mindenképp elég egy térkép megfelelő színezéséhez.) Tehát itt több fizikai kvantumbit alkot egy logikait, ezért biztosítanunk kell, hogy ezek a fizikai qubitek ne lehessenek eltérő értékűek. A fenti, „1 a 2-ből” mintájára azt látjuk, hogy az $a_1 = a_2 = -1$; $b_{12} = -2$ választás esetén, ha azonos a két qubit, akkor 0, ha különböző, akkor 1 a végső Hamilton-függvény értéke. Ez alapján legyen egy-egy szín logikai kvantumbitjén belül a fizikai qubitek súlya -1, az ezeket összekötő él súlya -2.

Másrészt szükséges, hogy egy-egy terület $K_{4,4}$ részgráfján belül a négyből csak egy szín legyen kiválasztva. A korábbiak alapján az egyes színek logikai kvantumbitjeinek súlya 1, a köztük futó logikai éleké 2. Ezeket egyenlő arányban osztjuk szét a fizikai megfelelőik között. A logikai qubitekhez két fizikai tartozik, ezért a fizikai kvantumbitek 1/2 súlyt kapnak; és mivel két logikai qubit között mindig két fizikai él van, ezek súlya 1 lesz. Ha egy fizikai qubit vagy él több ok miatt is kap különböző súlyokat, akkor a Hamilton-függvény linearitása miatt ezek összegét kell venni.



3.1. ábra. *Szomszédos területek leképezése*

A szomszédsági kényszer

Az egyes területek színeit már beágyaztuk, most a szomszédsági kényszert (a szomszédos területek nem lehetnek azonos színűek) fogjuk elkódolni. Legyen két szomszédos területnek

megfeleltetett $K_{4,4}$ az A és a B. Ha például A-ban és B-ben a piros szín kvantumbitje egyszerre 1, akkor az energiaszintnek emelkednie kell. Ha azonban mindkét helyen 0, vagy az egyiken 0, a másikon 1 a qubit értéke, akkor ne változzon a H_P értéke. A korábbi, „1 a 2-ből” mintájára látjuk, hogy $a_1 = a_2 = 0$; $b_{12} = 1$ jó paraméterezés: a mindkét qubit 0 és a pontosan egy qubit 0 esetben is 0 az ehhez a kényszerhez tartozó energiaszint, de a mindkét qubit 1 esetet büntetjük, ekkor 1-gyel nő a H_P értéke.

Ennek megvalósításához két szomszédos terület azonos színt jelképező logikai kvantumbitjeit összekötő élre van szükség. A 3.1. ábrán ívvel jelölve láthatók ezek. Elegendő tehát ezek súlyát 1-re állítani az azonos színű szomszédok büntetéséhez.

Még egy problémával szembesülhetünk: a fent leírt leképezéssel a Kiméra gráf struktúrája miatt minden területnek legfeljebb 4 szomszédja lehet, és három terület nem lehet páronként szomszédos egymással. Ebben segít a területek „klónozása”: egy területnek több $K_{4,4}$-et feleltetünk meg. Ez a több fizikai kvantumbitből álló logikai qubitekkel teljesen analóg módon működik, annak megfelelően kell beállítani a súlyokat. Példaként lássuk Kanada térképét (3.2. ábra), és annak leképezését (3.3. ábra). Az utóbbin egy cella egy $K_{4,4}$-nek felel meg, és a területek nevét a postai kódokkal rövidítették.

3.2. ábra. *Kanada 13 területegysége (forrás: [Wikb])*

	0	1	2	3	4
0	NL	ON	MB	SK	AB
1	PE	QC	NU	NT	AB
2		NB	NS	NT	BC
3				YT	BC

3.3. ábra. *Kanada térképének leképezése (forrás: [Dah13] Figure 5)*

Ezek alapján a 3.1. ábrának megfelelő (tehát klónozás nélküli) esetben a súlyok alakulása:

- a fizikai kvantumbitek esetén a súlyok (kényszerek szerinti bontásban):
  - egy szín logikai qubitjén belül megegyező értékű kvantumbitek legyenek: -1;
  - egy terület $K_{4,4}$-én belül csak egy szín legyen aktív: 1/2;
  - a szomszédsági kényszer miatt: 0;
  - $\rightarrow$ a Hamilton-függvény linearitása miatt ezek összegét kell venni, ami -1/2;
- a fizikai élek esetén a súlyok (az élek helyzete szerinti bontásban):
  - egy szín két fizikai qubitjét összekötő él: -2;
  - egy $K_{4,4}$-en belül különböző színek között futó él: 1;
  - két szomszédos terület azonos színt reprezentáló logikai qubitjei közötti él: 1.

A D-Wave-nél az 512 kvantumbites számítógépbe ágyazták a problémát. A $C = 2, 3, 4$ értékekre futtatták, mindegyikre háromszor, és mindig ezer mintával (azaz ezer darab – nem feltétlenül különböző – kvantumbit-konfiguráció állt elő a futás végére). Az eredményeket mutatja a 3.1. táblázat. Az értékpárok első tagja a kényszereknek megfelelő jó megoldások száma, a második pedig a kapott különböző eredmények száma. Látható, hogy Kanada térképét két színnel még nem, de hárommal vagy többel már sikerült kiszínezni. Ez megfelel a valóságnak, legkevesebb három szín kell a színezéshez.

A színek száma (C)	Megfelelő színezések/különböző minták
2	0/37, 0/45, 0/34
3	252/417, 228/371, 258/393
4	837/978, 812/947, 801/955

3.1. táblázat. *A futtatások eredménye (az adatok forrása: [Dah13] Table 3)*

3.2. QUBO

A QUBO probléma (Quadratic Unconstrained Binary Optimization problem) a másodfokú, kényszer nélküli, bináris optimalizálási probléma angol rövidítése. Lényege, hogy olyan másodfokú kifejezést akarunk minimalizálni, aminek a változói a $\{0, 1\}$ halmazból vehetnek fel értéket. A QUBO egy NP-nehéz probléma [Wikd]. Láttuk, hogy a Hamilton-függvény éppen a (végső soron bináris) kvantumbitek másodfokú függvénye, és az értékét minimalizálni szeretnénk. Tehát az energiaszint minimalizálása, és így a kvantumszámítógépnek adott probléma megoldása QUBO problémaként kezelhető. A fejezet további részének a fő forrása [RVO+15].

Az ilyen módon történő beágyazásnak két lépése van: a probléma leképezése QUBO-ra (mapping); és a beágyazás a hardverbe (embedding). A QUBO-ból történő beágyazáshoz

használható a D-Wave heurisztikus beágyazó szoftvere, így ezzel a lépéssel itt nem foglalkozunk.

A leképezésre több módszer létezik. Vannak általánosan használhatók (pl. időosztásos, CNF megközelítés), ezek azonban erőforrásigényesek, és így a jelenleg elérhető gépeken csak kis méretű problémák beágyazására alkalmasak. Másik lehetőség, hogy minden problémára külön-külön készítünk QUBO-t, ezt nevezzük közvetlen leképezésnek.

3.2.1. Időosztásos módszer

Lépésekre osztjuk a kezdeti- és a végső állapot közötti intervallumot, és így egyszerre csak egy-egy kisebb változás történik. Valamennyire tudjuk, minek kellene történni, és ami ezzel ellentétes, azt szankcionáljuk. Ezek leírására állapotváltozókat és akciókat használunk, amelyeket a kvantumbitekbe kódolunk. A büntetések összege adja az energiaszintet.

Például a kezdeti- vagy a végállapotban ismerünk bizonyos feltételeket, amelyeknek teljesülni kell az állapotváltozókra, és ha a qubitek között van, ami ennek ellentmond, akkor büntetésként növeljük az energiaszintet. Hasonlóképpen, ha egy akció végrehajtása utáni állapotban egy kvantumbit értéke eltér az akció előttitől, pedig az akcióból nem következett volna a megváltozása, azt is büntetjük. Valamint az akcióknak lehetnek előfeltételei, és kerülendő, hogy úgy hajtsuk végre azt, hogy ezek nem teljesülnek.

3.2.2. CNF megközelítés

A CNF, azaz konjunktív normálforma olyan logikai formula, amelyben a bináris változók és negáltjaik *vagy* kapcsolatai szerepelnek zárójelekben, és ezek között *és* kapcsolat van. Ha minden zárójelen belül k db változó van, akkor k-CNF kifejezésről beszélünk. Elérhetők olyan szoftverek (pl. SATPLAN), amelyek elvégzik problémák CNF-ként való megfogalmazását, így abból indulunk ki, hogy a megoldandó probléma CNF-ben adott.

A CNF-ből először PUBO-t gyártunk, ami ugyanaz, mint a QUBO, kivéve, hogy tetszőleges fokú lehet (a P betű a polinom szóra utal). Ezt úgy érjük el, hogy a CNF minden zárójelben lévő kifejezésének egy szorzat felel meg. Ebben a ponált változók helyett 1-nek és a megfelelő kvantumbit értékének a különbsége szerepel, a negált változók helyett pedig a megfelelő qubit értéke. Így például $a \vee \neg b \vee \neg c \vee d$ megfelelője $(1-a)bc(1-d)$ lesz. Látható, hogy a kapott kifejezés pontosan akkor lesz 0, ha a neki megfelelő logikai kifejezés igaz, egyébként pozitív az értéke.

Ahhoz, hogy az így előálló PUBO-ból QUBO-t kapjunk, még a fokszámot kell csökkenteni. Az erre használható egyik algoritmus lényege, hogy a szorzatokban leggyakrabban előforduló qubitpárokat egyetlen, új kvantumbittel helyettesítjük. Szükséges bevezetni egy büntető tagot, ami azt biztosítja, hogy az új qubit értéke pontosan akkor 1, amikor az eredeti kifejezés is. Ezt addig ismételgetjük, amíg a kapott kifejezés már csak másodfokú.

Erre egy lehetőség a következő: tegyük fel, hogy egy yzw tagban az yz tényezőt kicseréljük az új, x változóra. Ekkor elvárjuk, hogy x pontosan akkor legyen 1, ha y és z is 1, egyébként 0 legyen. Ehhez büntető tagnak megfelel az $x(3-2y-2z)+yz$, amit értéktáblázattal (3.2. táblázat) ellenőrizhetünk.

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
$x = yz$	1	1	1	0	0	0	0	1
$x(3 - 2y - 2z) + yz$	0	0	0	1	3	1	1	0

3.2. táblázat. *A büntető tag megfelelőségének ellenőrzése*

Láthatjuk, hogy a büntető tag pontosan akkor 0, amikor $x = yz$, egyébként pozitív, azaz valóban biztosítja, hogy az új változó értéke ugyanaz legyen, mint az eredeti két változó szorzata. Tehát az yzw harmadfokú tag helyett írhatjuk az $xw + x(3 - 2y - 2z) + yz$ másodfokú kifejezést.

Magasabb fokú tagok esetén ugyanezt a lépést többször kell elvégezni, egyszerre mindig csak eggyel csökken a fokszám. Tehát egy n-edfokú tag esetén $n-2$ lépésre – és ugyanennyi új változóra – van szükség, hogy az eredmény másodfokú legyen.

3.2.3. Közvetlen leképezés

A közvetlen leképezés nem általános módszer, minden problémára más és más, ezért csak példán keresztül mutatható be. Ez a példa a Hamilton-út keresés lesz (irányítatlan gráfban).

Egy n csúcsú gráfhoz vezessünk be n^2 db változót: $x_{ij} = 1$ azt jelenti, hogy az i jelű csúcs a Hamilton-úton lévő csúcsok sorában a j-edik pozícióban szerepel (úgy is fogalmazhatunk, hogy a csúcsokat sorban látogatva az i csúcs j-edikként kerül sorra); $i, j \in \{1, 2, \ldots, n\}$. Az x_{ij}-k logikai kvantumbitek, ugyanis a (szoftverrel történő) beágyazás után sokan lehetnek hatással egymásra, és így a fizikai qubitek közötti élek kevésnek bizonyulhatnak.

Három kényszer van, amelyek megszegését büntetni kell.

1. Minden csúcs pontosan egyszer van a Hamilton-úton.

   Minden i csúcsra összegezzük, hogy hányszor látogattuk meg. Ha ez nem egy, büntetünk:

   $$\sum_{i=1}^{n} \left(\left(\sum_{j=1}^{n} x_{i,j} \right) - 1 \right)^2 .$$

2. Egy pozícióban pontosan egy csúcs van (avagy egyszerre nem látogatunk meg egynél több vagy kevesebb csúcsot).

   Minden j pozícióra összegezzük, hogy hány csúcsot látogattunk meg ekkor. Ha ez nem egy, akkor büntetünk:

   $$\sum_{j=1}^{n} \left(\left(\sum_{i=1}^{n} x_{i,j} \right) - 1 \right)^2 .$$

3. Két, szomszédos pozícióban lévő (egymás után látogatott) csúcs között fut él.

   Minden $j < n$ pillanatra összegezzük azon esetek számát, hogy a k csúcs következik

az i csúcs után (előbbit a $(j+1)$-edik, utóbbit a j. pozícióban látogattuk meg), de nincs közötük él a gráfban:

$$\sum_{j=1}^{n-1} \sum_{i,k:(i,k)\notin E} x_{i,j}x_{k,j+1}.$$

Látjuk, hogy valóban másodfokú (QUBO) kifejezéseket kaptunk. Ennek a három kifejezésnek az összege lesz a QUBO probléma, amit a megfelelő szoftverrel már be tudunk ágyazni a D-Wave kvantumszámítógépeibe.

3.3. Ütemezési típusú problémák

A szekció forrása [RVO+15]. Az ún. ütemezési problémák lényege az, hogy az egyes részfeladatok között szétosztjuk a felhasználandó erőforrásokat, és időszeleteket rendelünk hozzájuk. Azon feladatok, amelyek ugyanazt az erőforrást használják, nem futhatnak egyazon időszeletben. Mindeközben teljesülnie kell bizonyos megszorító feltételeknek. Ezek alapján az ütemezési problémák egy része megfeleltethető gráfszínezési problémának. A gráf csúcsai a feladatok, az azonos erőforrást használó feladatok között él fut, eltérő színek jelölik a különböző időszeleteket. Így a kromatikus szám lesz az összes feladat elvégzéséhez minimálisan szükséges időszeletek száma.

Például a gráfszínezési probléma a következőképpen fogalmazható meg ütemezési problémaként. Adott az irányítatlan, n csúcsú G gráf, amelynek csúcsait k db színnel szeretnénk kiszínezni. Feleltessünk meg minden v csúcsnak néhány logikai kvantumbitet:

- k db akció, amelyeket a_v^c-vel jelölünk, jelentése: a v csúcsot c színűre színezzük;
- egy s_v^g állapotváltozó, ami azt jelöli, hogy kiszíneztük-e már v-t;
- k db s_v^c állapotváltozó, ami azt jelöli, hogy a v csúcs c színű-e.

Jelöljük $C(v)$-vel azon csúcsok halmazát, amelyek a bemeneti gráfban szomszédosak v vel. Minden a_v^c akciónak $|C(v)|+1$ előfeltétele van: egyrészt a v csúcs még ne legyen kiszínezve, azaz s_v^g legyen hamis; másrészt v szomszédai között ne legyen c színű, azaz $\forall v_i \in C(v)$ esetén $s_{v_i}^c$ legyen hamis. Minden akciónak van két következménye is: s_v^g és s_v^c igaz lesz.

A kezdeti állapotban minden v-re és minden c-re s_v^g és s_v^c is hamis. A végső állapotban az összes csúcs ki kell legyen színezve, azaz minden v-re s_v^g igaz. Egy végrehajtás – azaz a bemeneti gráf kiszínezése – n db akció sorozatából áll, ahol minden akció egy-egy csúcs kiszínezését jelenti.

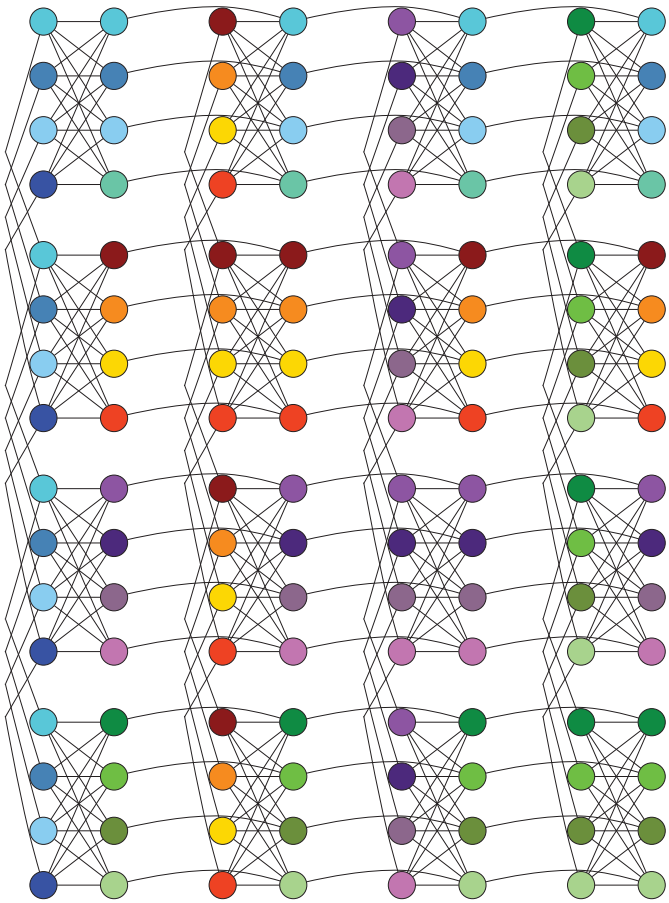
Az általános módszerekkel viszonylag könnyen elvégezhető az ütemezési problémák leképezése QUBO-ra, ezért, ha egy feladatot meg tudunk fogalmazni ütemezési problémaként, akkor már tekinthetjük úgy, hogy sikerült leképezni – és így a D-Wave megfelelő szoftverének segítségével beágyazni is.

3.4. Teljes gráf beágyazása

Sok esetben nagyon kényelmessé teszi a beágyazást, ha bármely két logikai csúcs között van él, azaz ha mindegyik qubit értéke hatással lehet a többire. Ez a Kiméra gráfban a fizikai qubitek között sajnos nem teljesül, de ha a megfelelő fizikai kvantumbitekkel valósítjuk meg a logikaiakat, akkor már elérhetjük a célt. A szekció forrása [KSH12].

A Kiméra gráffal azonos szerkezetű, de általánosabb gráfba fogunk ágyazni. Legyen ez a gráf $K_{4,4}$-ek helyett $K_{c,c}$ gráfokból álló rács. Egy $m \times m$-es rácsba az itt következő módszerrel be tudunk ágyazni egy $cm+1$ csúcsú teljes gráfot (azaz K_{cm+1}-et). A fejezet további részében a teljes gráf csúcsaira használjuk a csúcs elnevezést, míg a Kiméra gráf csúcsaira qubitekként hivatkozunk – ezek ebben az esetben a fizikai qubitek, míg a logikai kvantumbitek megfelelnek a teljes gráf csúcsainak.

Az 1×1-es rácsba ($m = 1$ eset), azaz az egyetlen $K_{c,c}$-ből álló gráfba ágyazáshoz legyenek a logikai qubitek a $K_{c,c}$ sorai (mint a közvetlen beágyazásnál a színek kódolásánál láttuk), kivéve az utolsó sort, ahol mindkét qubit feleljen megy egy-egy új logikai kvantumbitnek (azaz csúcsnak). Így $c+1$ db csúcsot kaptunk, és valóban bármely kettő között fut él, azaz a K_{c+1} beágyazását végeztük el.



3.4. ábra. *K_{17} beágyazása 4×4-es rácsba*
(forrás: [KSH12] Fig. 6.(b))

Ha $c+1$-nél több csúcsú teljes gráfot szeretnénk beágyazni, akkor a további lépésekben eggyel növeljük a rács méretét mindkét irányban, mondjuk jobbra és lefelé. Ekkor az új jobb alsó sarokban lévő $K_{c,c}$ egyes sorainak qubitpárjai azonos logikai qubithez fognak tartozni, de minden sor másikhoz, és mindegyik a beágyazandó teljes gráf olyan csúcsához, amely az előző méretű rácsban még nem szerepelt (így c db új csúcsot ágyazunk be). A bővítéssel kapott többi $K_{c,c}$ fizikai kvantumbitjei annak a beágyazandó csúcsnak a logikai qubitjéhez fognak tartozni, amelyikhez közvetlen él köti őket. Így néggyel több csúcsú teljes gráf beágyazásához egy új sor és oszlop szükséges, azaz a szükséges qubitek száma négyzetesen nő a beágyazandó teljes gráf méretével. Mindezt szemlélteti a 3.4 ábra, ahol a teljes gráf egyes csúcsainak megfeleltetett fizikai kvantumbitek (azaz az egyazon logikai qubithez tartozó fizikai qubitek) kaptak azonos színt.

4. fejezet

Néhány probléma beágyazása

Ebben a részben kerül sor az általam kigondolt beágyazások ismertetésére. Maximális méretű teljes részgráf, maximális méretű teljes páros részgráf és minimális méretű domináns halmaz keresésével foglalkoztam. Ezen kívül írtam egy egyszerű programot, amellyel a közvetlen beágyazást lehet szimulálni – erről is lesz szó.

4.1. MAXKLIKK

A MAXKLIKK annak a gráfelméleti problémának a jelölése, amelyben egy bemeneti G gráfban a legnagyobb előforduló teljes részgráf (klikk) mérete a kérdés. Ennek az eldöntési változatában G mellett egy k paraméter is adott, és az a kérdés, hogy van-e a gráfban k méretű klikk. A probléma közismerten NP-teljes. Bár a beágyazást már korábban megoldották, én ettől függetlenül jutottam az alábbiakra.

Közvetlen beágyazással

Egy n csúcsú G gráf esetén az n csúcsú teljes gráfot ágyazzuk be (a 3.4. szekcióban tárgyalt módon) – de csak G komplementerének ($\bar{G}$) éleit fogjuk használni. G csúcsainak súlyai viszonylag kis abszolút értékű pozitív számok legyenek, pl. mindegyik legyen 1. (Mivel pozitívak, a Hamilton-függvényhez negatív értéket fognak adni, hiszen abban a csúcsokra vonatkozó súlyösszeg előtt negatív előjel szerepel.) $\bar{G}$ éleinek súlya legyen valamivel nagyobb abszolút értékű pozitív szám, pl. 3; G éleinek súlya pedig 0.

A gráf csúcsainak illetve éleinek súlyai rendre logikai qubitek és logikai élek súlyai lesznek. Ezeket a következő módon osztjuk szét a fizikai megfelelők között: egy logikai qubit súlyát a fizikai kvantumbitjei között egyenlően osztjuk szét; egy logikai él fizikai megfelelői közül egy megkapja a teljes logikai súlyt, a többi 0 súlyú lesz. (Az eredményen nem változtatna, ha a logikai élek súlyát is szétosztanánk a hozzá tartozó fizikaiak között.) A fizikai kvantumbitek értéke 1, ha bevesszük a maximális klikkbe a gráf megfelelő csúcsát, és 0, ha nem. Azt, hogy egy logikai qubiten belül a fizikai kvantumbitek azonos értékűek legyenek, a 3.1. szekcióban látottak alapján biztosíthatjuk.

Ez valóban megfelelő beágyazás, mert egyrészt ha nem klikket választunk ki, az nem lehet optimális. Ugyanis ha van olyan csúcs a kiválasztottak között, ami nincs összekötve

az összes többi kiválasztottal, akkor ennek a csúcsnak az elhagyásával alacsonyabb energiaszintű állapotba jutnánk. Hiszen a legalább egy darab $\bar{G}$-beli él törlése miatt legalább 3-mal csökken az energia szintje, a csúcs törlése miatt pedig csak 1-gyel nő, más olyan változás pedig nem történik, ami változtatna az energiaszinten.

Másrészt ha egy kisebb klikket választanánk ki, mint a maximális, akkor kevesebb érték összege lesz a kiválasztott csúcssúlyok összege. A kiválasztott csúcsok között pedig ebben az esetben csak G-beli élek vannak, amelyek súlya 0, így ezek nem változtatják az energiát. Mivel a csúcssúlyok összege negatív előjellel számít, nagyobb energiaszintű állapotot eredményez, ha a maximális klikk helyett egy kisebbet választunk ki.

A felhasznált fizikai qubitek száma a teljes gráf beágyazás miatt a bemeneti gráf csúcsszámával négyzetesen nő. A kvantumszámítógépen való futtatáskor azt figyeljük, hogy a kimeneti minták között van-e olyan, amelyikben a vizsgált gráfnak az 1 értékű fizikai qubitekhez tartozó csúcsai legalább k méretű teljes gráfot alkotnak.

Ütemezési problémaként

A G gráf minden egyes csúcsához tartozik két állapotváltozó: bevesszük-e a klikkbe (x_i), bevehető-e oda (y_i); és egy akció is: hogy bevesszük a klikkbe. Kezdetben minden i-re $x_i = 0$, és $y_i = 1$. Ahhoz, hogy az i. csúcsot bevegyük, két feltételnek kell teljesülnie: még ne legyen bevéve ($x_i = 0$), és legyen bevehető ($y_i = 1$). A bevétel következménye, hogy az i. csúcs bekerült a klikkbe, és már nem vehető be ($x_i = 1$; $y_i = 0$). Továbbá azok a csúcsok, amelyek nem szomszédosak az i. csúccsal, már nem vehetők be, formálisan: ha $(i, j) \in E(\bar{G})$, akkor mostantól $y_j = 0$ kell legyen.

Így valóban mindig teljes gráfot alkotnak a bevett csúcsok és a köztük futó élek. A maximális klikk pedig nyilván elérhető, ha az ehhez tartozó csúcsokat vesszük be. A logikai kvantumbitek (állapotváltozók és akciók) száma lineárisan nő a probléma méretével, de az ezek közötti kapcsolatok biztosítása érdekében a fizikai qubitekre képezéshez szükség lehet teljes gráf beágyazására, és így a fizikai kvantumbitek száma már négyzetesen változhat. (De ha nincs szükség sok kapcsolatra a logikai qubitek között, akkor kevesebbet igényelhet.)

QUBO közvetlen leképezéssel

Két, össze nem kötött csúcs egyszerre nem lehet egy klikkben, és ne maradjon bevehető csúcs a végére. Az ezeknek ellentmondó eseteket büntetjük az energiaszint növelésével. A (4.1) képletben az x_i-k és az y_i-k ugyanazt jelölik, mint az ütemezési problémaként történő megfogalmazásnál.

$$\sum_{(i,j)\in E(\bar{G})} x_i x_j + \sum_{i \in V(G)} y_i \tag{4.1}$$

Az ütemezési megfogalmazáshoz hasonlóan itt is négyzetesen függhet a fizikai kvantumbitek száma a bemeneti gráf méretétől.

4.2. Maximális teljes páros részgráf

Teljes páros gráfon, avagy biklikken olyan páros gráfot értük, amelynek minden olyan csúcspárja között fut él, amelynek tagjai közül az egyik csúcs az egyik, a másik pedig a másik csúcsosztályból való. Most azt a problémát vizsgáljuk, hogy egy bemenetként adott tetszőleges G gráfban mekkora a legnagyobb méretű feszített teljes páros részgráf. Azaz olyan A és B diszjunkt részhalmazait keressük a csúcsoknak, amelyekre teljesül, hogy minden A-beli csúcs össze van kötve bármely B-belivel, de semelyik két A-beli, illetve semelyik két B-beli csúcs között nincs él. Ennek a méretén a kiválasztott csúcsoknak az $|A| + |B|$ elemszámát értjük. Ennek is az eldöntési változatát tekintjük: van-e G-nek olyan feszített teljes páros részgráfja, amelynek a mérete legalább k. A probléma NP-teljes [Yan78].

Ütemezési problémaként

A tetszőleges G gráf minden (i-edik) csúcsához tartozik négy állapotváltozó: benne van-e a páros részgráf egyik csúcshalmazában, A ban: x_i^A, benne van-e a másikban, B ben: x_i^B, bevehető-e A ba: y_i^A, bevehető-e B be: y_i^B. Két akció is tartozik a csúcsokhoz: bevétel A ba: a_i^A, bevétel B-be: a_i^B.

Kezdetben minden i-re $x_i^A = 0$; $x_i^B = 0$; $y_i^A = 1$; $y_i^B = 1$. Az a_i^A előfeltétele, hogy az i. csúcs bevehető legyen A-ba ($y_i^A = 1$). Következményei, hogy már be van véve ($x_i^A = 1$), már nem vehető be ($y_i^A = 0$), valamint az összes, még semelyik halmazba be nem vett, de valamelyikbe még bevehető csúcsra ellenőrizni kell, hogy még mindig bevehető-e oda.

Egy csúcs (legyen ez a j.) pontosan akkor marad bevehető B-be az a_i^A akció után, ha szomszédos az i. csúccsal, formálisan: ha $(x_i, x_j) \in E(\bar{G})$, akkor mostantól $y_j^B = 0$ kell legyen. Egy csúcs pontosan akkor marad bevehető A-ba, ha nem szomszédos az i. csúccsal, formálisan: ha $(x_i, x_j) \in E(G)$, akkor ezentúl $y_j^A = 0$ kell legyen. Az a_i^B akciókkal természetesen hasonló a helyzet, csak az A-kat és a B-ket kell felcserélni egymással.

Így valóban csak olyan csúcs kerülhet be valamelyik halmazba, amely az adott halmazon belül lévő csúcsok közül egyikkel sem szomszédos, és a másik halmaz csúcsai mind szomszédosak vele, azaz végig biklikket alkotnak a bevett csúcsok. Az is könnyen látszik, hogy ezzel a módszerrel mindenképp elérhető a maximális méretű teljes páros részgráf az ebben benne lévő csúcsok bevételével. A korábbiakhoz hasonlóan itt is igaz, hogy a logikai qubitek száma lineárisan, a fizikaiaké négyzetesen változik a probléma méretével.

QUBO közvetlen leképezéssel

Ha két szomszédos csúcs egyazon csúcshalmazba került, vagy ha két nemszomszédos csúcs különböző halmazba került, azt büntetni kell. Valamint ne maradjon egyik halmazhoz hozzáadható csúcs sem. Rendre ezt a három feltételt írja le a (4.2) kifejezés három tagja, ahol a változók az ütemezési feladatként való megoldásban ismertetetteknek felelnek meg.

$$\sum_{(i,j)\in E(G)} \left(x_i^A x_j^A + x_i^B x_j^B\right) + \sum_{(i,j)\in E(\bar{G})} \left(x_i^A x_j^B + x_i^B x_j^A\right) + \sum_{i\in V(G)} \left(y_i^A + y_i^B\right) \tag{4.2}$$

Ha nem feszített részgráf a cél, akkor majdnem ugyanez az eredmény, csupán az A illetve a B halmazon belül futó élek büntetését (azaz az első tagot) kell elhagyni. Az korábbiakhoz hasonlóan itt is teljesül, hogy a logikai qubitek száma lineárisan, a fizikaiaké négyzetesen változik a probléma méretével.

4.3. Domináns halmaz

A domináns halmaz egy G gráf csúcsainak olyan D részhalmaza, amelyre igaz, hogy minden olyan csúcs, ami nincs benne D-ben, legalább egy D-beli csúcsnak szomszédja. Az ebből származó optimalizálási probléma az, hogy határozzuk meg a legkisebb elemszámú ilyen halmazt. Eldöntési feladatként megfogalmazva: van-e legfeljebb k méretű domináns halmaz G-ben. Ez is NP-teljes probléma [GJ79].

Ütemezési problémaként

Az eddigiekhez képest fordítva járunk el: kezdetben a G gráf minden csúcsát vegyük be a domináns halmazba, és majd kiveszünk belőle néhányat. Minden csúcshoz tartozik két állapotváltozó: benne van-e a domináns halmazban (x_i), kivehető-e onnan (y_i); és egy akció is: kivétel a halmazból.

Kezdetben tehát az összes csúcs eleme a domináns halmaznak, és kivehető onnan: minden i-re $x_i = 1$, és $y_i = 1$. Az i. csúcs kivételének előfeltétele, hogy kivehető legyen ($y_i = 1$). Következménye, hogy az i. csúcs már nincs bevéve a domináns halmazba ($x_i = 0$), és már nem is vehető ki ($y_i = 0$). Ezen kívül minden, ekkor még kivehető (j.) csúcsra ellenőrizzük, hogy még mindig kivehető-e: ha $\sum_{(x_k,x_j)\in E(G)} x_k = 0$ (azaz a j. csúcs egyik szomszédja sincs már bevéve a halmazba), akkor már nem vehető ki, egyébként kivehető marad.

Így valóban csak olyan csúcsok kerülhetnek ki a halmazból, amelyeknek még van szomszédja a bennmaradók között, azaz a bennmaradók mindvégig domináns halmazt alkotnak. A legkisebb méretű domináns halmazba nem tartozó csúcsok kivételével pedig nyilván elérhető a minimális méretű domináns halmaz. A korábbiakhoz hasonlóan itt is igaz, hogy a logikai qubitek száma lineárisan, a fizikaiaké négyzetesen változik a probléma méretével.

QUBO a közvetlen leképezéssel kapott PUBO-ból

Azt kell büntetni, ha van olyan csúcs, ami nincs bevéve a domináns halmazba, és ugyanez igaz az összes szomszédjára is. Ezen kívül a végére ne maradjon kivehető csúcs. A (4.3) képletben az x_i-k és az y_i-k jelentése az ütemezési megoldásnál leírtaknak felel meg.

$$\sum_{i\in V(G)} \left((1 - x_i) \prod_{(i,j)\in E(G)} (1 - x_j) \right) + \sum_{i\in V(G)} y_i \tag{4.3}$$

Ez a kifejezés nem másodfokú, az első tagban ugyanis eggyel több kvantumbit szorzata szerepel, mint ahány szomszédja van az i-edik csúcsnak – ez pedig kettőnél jóval több is lehet. Tehát egy PUBO-t sikerült felírnunk. A CNF megközelítés (3.2.2.) alszekcióban írtaknak megfelelően ez már QUBO-vá alakítható.

Megismételjük, hogy a módszer lényege az, hogy egy-egy bitpárt egy új változóval helyettesítünk, és egy (másodfokú) büntető taggal biztosítjuk, hogy az új változó értéke valóban az eredeti kettő szorzata legyen. Ezt addig ismételjük, amíg a kapott kifejezés másodfokú nem lesz.

Nézzük például az $x_1x_2x_3 \dots x_k$ tagot. Először az x_1x_2 helyett vezessük be az y_1 tényezőt. Így az $y_1x_3 \dots x_k + y_1(3 - 2x_1 - 2x_2) + x_1x_2$ kifejezést kapjuk. A következő lépésben az $y_2 = y_1x_3$ tényezőt használjuk a fokszám további csökkentésére. A kapott kifejezés: $y_2x_4 \dots x_k + y_1(3 - 2x_1 - 2x_2) + x_1x_2 + y_2(3 - 2y_1 - 2x_3) + y_1x_3$.

Ezt addig folytatjuk, amíg végül az alábbi másodfokú kifejezést nem kapjuk:

$$y_{k-2}x_k + y_1 (3 - 2x_1 - 2x_2) + x_1x_2 + \sum_{i=2}^{k-2} (y_i (3 - 2y_{i-1} - 2x_{i+1}) + y_{i-1}x_{i+1}) .$$

Ezt minden, kettőnél nagyobb fokszámú tagra elvégezzük, és így az egész PUBO-t átalakíthatjuk QUBO-ra. A szükséges logikai qubitek száma függ a PUBO fokszámától, a fizikaiak száma pedig legfeljebb ezek négyzetével lehet arányos.

4.4. Beágyazást szimuláló program

A Java nyelven írt szoftver[1] működése annyiból áll, hogy a bitek összes lehetséges értékére megnézi az energiaértékeket, és ezek közül a minimálisakhoz tartozó konfigurációkat kiírja. Épp ezért nagyobb bemenetekre rendkívül lassúvá válik – éppen az a kvantumszámítógépek jelentősége, hogy azok nem exponenciális futásidejűek az ilyen problémák esetén, szemben a klasszikus társaikkal.

A program bemenetként a beágyazáshoz használt Kiméra gráf méreteit és a súlyokat várja el. Ezért leginkább a közvetlen beágyazáshoz használható segédeszközként. Nem csak a D-Wave-nél használt Kiméra gráfot lehet használni, tetszőleges méretű teljes páros gráfokból állhat, és ezek is tetszőleges méretű rácsot alkothatnak.

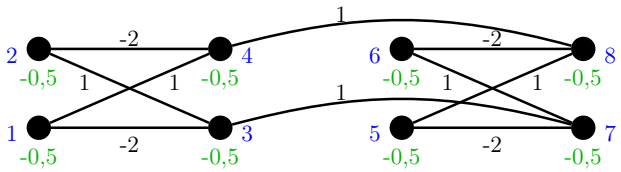
A programot kipróbáltam mindkét, a dolgozatban szereplő közvetlen beágyazással megoldott problémára (térképszínezés és MAXKLIKK).

4.4.1. Térképszínezés

Az első kísérlet két szomszédos terület két színnel színezése volt. Ehhez egy 1×2-es, $K_{2,2}$ részgráfokból álló rácsba kell ágyazni. A közvetlen beágyazásról szóló (3.1.) szekcióban írtaknak megfelelően minden csúcs súlya -0,5; a $K_{2,2}$-n belül az egyes sorok két-két qubitjét összekötő élek súlya -2; az összes többi él súlya pedig 1. A 4.1. ábrán az éleken azok súlya feketével, a csúcsok mellett azok sorszáma kékkel, a súlyuk pedig alattuk zölddel szerepel.

A futás eredménye: akkor lesz minimális az energiaszint, ha a 2, 4, 5 és 7 sorszámú, vagy akkor, ha az 1, 3, 6 és 8 sorszámú bitek értéke 1. Ez megfelel annak, amit várunk: akkor és csak akkor a legalacsonyabb az energia szintje, ha a két terület különböző színt kap.

[1] A program kódja használati útmutatóval és példabemenetekkel elérhető a következő linken: `https://drive.google.com/open?id=0BwoN5CdvjrZKaWJ1TjNsTE5mc0U`



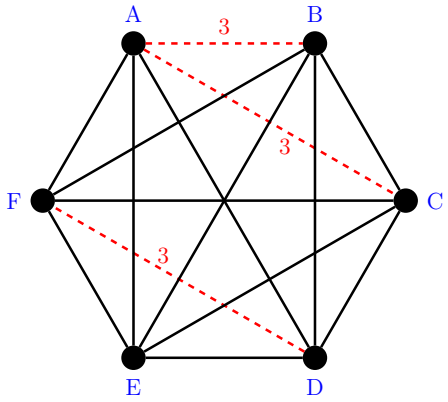
4.1. ábra. *Térképszínezés 2 területtel, 2 színnel*

Három színre is kipróbáltam ugyanezt, ekkor hat esetben kapunk minimális energiaszintet. Valóban, az első terület bármilyen színű lehet a 3 közül, és ekkor a második színe már csak a maradék két szín egyike lehet. A kapott esetek tényleg a jó megoldásokat írják le.

Olyan esetet is néztem, ahol nincs megoldás (három, páronként szomszédos terület színezése két színnel). Ekkor rengeteg (192 db) olyan kimenet volt, ahol minimális az energiaszint. Ha egyről ellenőrizzük, hogy az nem megoldás, és feltételezzük, hogy az összes minimális energiaszintű állapot jó megoldás (ha van ilyen), akkor láthatjuk, hogy a feladatnak nincs megoldása.

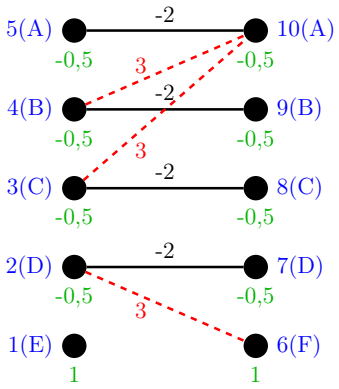
4.4.2. MAXKLIKK

A MAXKLIKK probléma esetén először egy hatcsúcsú gráffal próbálkoztam (4.2. ábra). Az ábrán a gráf élei a fekete színű élek, a piros szaggatottak a gráf komplementerének élei. A MAXKLIKK közvetlen beágyazással történő megoldásánál (35. oldal) írtaknak megfelelően a piros szaggatottal jelölt élek súlya 3, a többié 0; a csúcsok súlya pedig 1.



4.2. ábra. *A MAXKLIKK próbagráfja*

A beágyazáshoz teljes gráf beágyazást (3.4. szekció) használunk. A D-Wave Kiméra gráfjából hat csúcsú teljes gráf beágyazásához a korábban kifejtett módszerrel 2×2-es rácsra lenne szükség. De én kihasználtam, hogy a programban megadható a Kiméra gráf teljes páros részgráfjainak mérete. Így egyetlen $K_{5,5}$-be ágyaztam a gráfot: minden sor megfelel egy-egy csúcsnak (logikai qubit), kivéve az alsó sort: itt mindkét fizikai kvantumbit egy önálló logikai qubit, azaz a bemeneti gráf egy csúcsa. A 4.3. ábrán a qubitek mellett kékkel a sorszámuk és zárójelben az eredeti, hatcsúcsú gráf (4.2. ábra) neki megfelelő csúcsának betűjele szerepel; alattuk zölddel pedig a qubitek súlya. A nulla súlyú élek nincsenek feltüntetve az ábrán, a nem nulla súlyú élekre azok súlya van írva.



4.3. ábra. *MAXKLIKK beágyazása*

A qubitek súlya egyrészt a MAXLIKK probléma közvetlen beágyazásából ered: a legtöbb fizikai qubit ebből kap 0,5 súlyt, mivel a logikai kvantumbitek (csúcsok) súlya 1, és ez két fizikai kvantumbitből áll. Kivétel az 1. és a 6. qubit, ezek 1-1 súlyt kapnak. A súlyok másik forrása annak a kényszernek a biztosítása, hogy az egy logikai kvantumbithez tartozó fizikai qubitek értéke azonos legyen. Ez a közvetlen beágyazásról szóló (3.1.) szekcióban írtak alapján minden fizikai kvantumbitre, amelyik egy másikkal alkot egy logikai qubitet, -1. Így az 1. és a 6. qubit súlya marad 1, a többié pedig $0,5 - 1 = -0,5$ lesz.

Az élek közül a fekete folytonosak súlya az egy logikai qubiten belüli fizikai qubitek konzisztenciájából adódik: a 3.1. szekcióban írtak alapján ez a súly -2. A piros szaggatott élek a 4.2. ábra piros éleinek felelnek meg. A beágyazás megtervezésénél (35. oldal) ugyanis azt írtuk, hogy a logikai kvantumbitek között futó fizikai élek közül egyetlen, tetszőleges él kapja meg a teljes súlyt, a többi súlya 0 lesz. Az eredményen nem változtatna, ha az élsúlyt is szétosztanánk a fizikai élek között (pl. a 4. és 10. qubit közötti 3 súlyú él helyett egy 4. és 10. közötti 1,5, és egy 5. és 9. közötti 1,5 súlyú él lenne).

A program futásának eredménye: az 1, 3, 4, 6, 8 és 9 vagy az 1, 2, 3, 4, 7, 8 és 9 bitek legyenek 1 értékűek. Azaz a $BCEF$ vagy a $BCDE$ a megoldás: valóban ez a két K_4 részgráf van a bemeneti gráfban, ennél nagyobb klikk pedig nincs.

4.4.3. Futásidő

Mivel a szoftver minden lehetséges konfigurációt ellenőriz, a futásidőt legfőképp a kvantumbitek száma határozza meg. Kísérleteim alapján egy 28 qubites térképszínezés még egy perc alatt lefutott, de 32 qubites esetben fél óra sem volt elég a feladat elvégzéséhez. Innen is látszik az exponenciális futásidő, ami miatt ilyen problémák esetén szívesebben fordulunk egy kvantumszámítógéphez – vagy ha ilyen épp nincs kéznél, akkor valamilyen klasszikus heurisztikus algoritmushoz.

III. rész

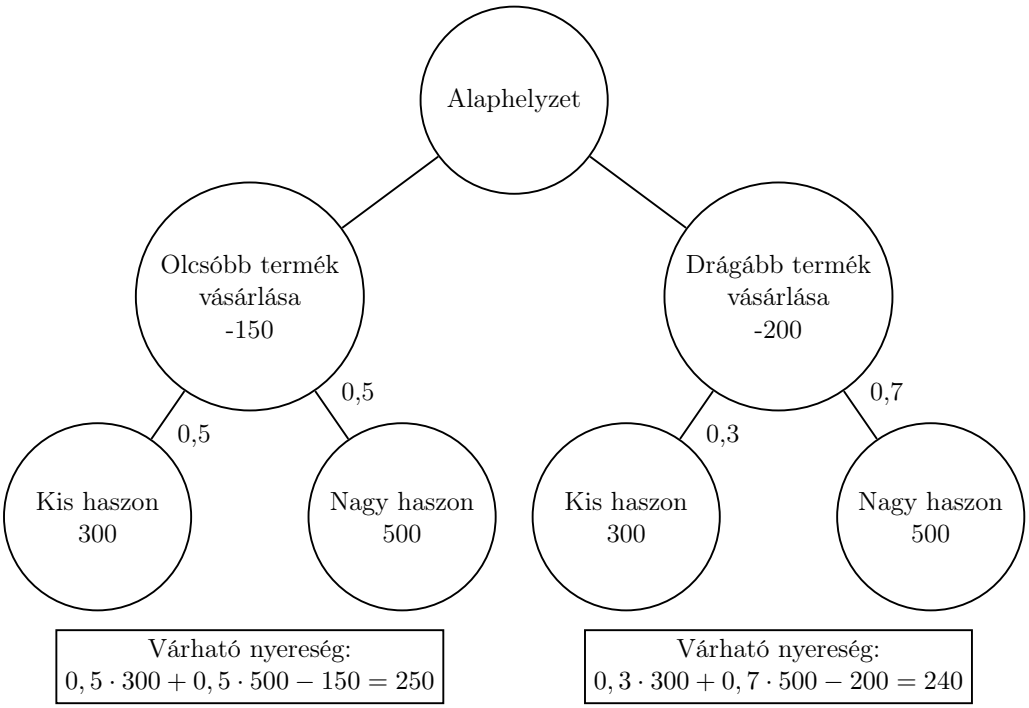
Gépi tanulás áramköri modellel

5. 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 az 5.1. ábrán.

A fejezetben először röviden vázoljuk a klasszikus döntési fákat, majd az ezekhez hasonló módon leírható kvantum döntési fákat egy 2014. évi cikk [LB14] alapján. Az utolsó szekció néhány, ezen kvantum döntési fák használatát megkönnyítő észrevételünket tartalmazza.



5.1. ábra. *Egy egyszerű döntési fa*

5.1. 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 [BFOS83]). 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.

5.2. 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 [LB14].

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 kvantumentrópiájú tulajdonság szerint ágazunk el.

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 a bemenetek az adott tulajdonsághoz tartozó állapotuknak megfelelő ágon mennek tovább. Az összetett mintázatú esetre bevezetünk egy hasonlósági mértéket. Két vektor $(|x_i\rangle, |x_j\rangle)$ hasonlósága: $F(|x_i\rangle, |x_j\rangle) = \operatorname{Tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}}$, 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 vektor, 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 csoportjába (cluster) teszünk. A legnagyobb hasonlóság (és korábban a minimális entrópia) meghatározásához 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.

5.3. Alkalmazást segítő új eredmények

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

Hasonlóság számítása skalárszorzat segítségével

Belátjuk, hogy a fent definiált F hasonlósági mérték valójában (amennyiben tiszta állapotokról van szó) a két bemeneti vektor skalárszorzatának abszolút értékét 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 skalárszorzatának abszolút értéke.

Bizonyítás: Két egységvektorra (azaz tiszta állapotra), $|x_i\rangle$-re és $|x_j\rangle$-re

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

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 skalárszorzatának abszolút értékével: $F(|x_i\rangle, |x_j\rangle) = |\langle x_i|x_j\rangle| = \frac{|a+d|}{\sqrt{2}}$.

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 egy mintabemenet tulajdonságait leíró $|x_i\rangle$ vektorban 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 (lásd: (5.1) képlet). A negyedik elem valami egyéb tulajdonsághoz tartozhat.

$$\text{hőmérséklet} \begin{cases} 0 \\ 1 \\ 0 \\ 0 \end{cases} \text{szárazság} \tag{5.1}$$

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

5.1. 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 5.1. 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.

6. 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 néhány fontosabbat megemlítünk, egy algoritmust pedig részletesen is kifejtünk. A különböző módszerek elnevezésénél – azoknak a magyar nyelvű szakirodalomban való elterjedtsége miatt – az angolszász megnevezést használjuk.

6.1. Klasszikus alapváltozatok

Az alábbi módszerek ismertetéséhez felhasznált irodalom: a bagging esetén az MIT mély tanulási tankönyvének [GBC16] 7.11. fejezete, a dropoutnál [GBC16] 7.12. fejezete, a boostingnál pedig Freund és Schapire cikke az AdaBoost algoritmusról [FS97].

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 mintahalmazból 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 fontosabb (a kime-

net becslésében hasznosabb) tulajdonságok nagyobb valószínűséggel kerülnek kiválasztásra (lásd pl. [Wikf]).

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. Ahhoz 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.

6.2. AdaBoost

Az AdaBoost algoritmus [FS97] 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: a $h_t(\underline{x}_i) = y_i$ egyenletet teljesítő mintapontok összsúlya a lehető legnagyobb. Feltesszük, hogy a **WeakLearn** algoritmus ezt a kiválasztást elvégzi, és visszaadja a kiválasztott gyenge osztályozó hipotézis függvényét. Ennek az algoritmusnak a minták tulajdonságvektorain 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: $(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)$
$\underline{D}$ kezdeti eloszlás a minták felett
a gyenge osztályozókat kezelő 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.

6.3. Egy kvantum osztályozó algoritmus

A most bemutatandó algoritmus [SP18] az egyik első, ami kvantum osztályozókból álló ensemble megalkotásáról szól, és így kihasználja mind a kvantumalgoritmusok, mind az ensemble módszerek előnyeit.

6.3.1. Általános leírás

Az alapgondolat az, hogy az egyes kvantum osztályozókat (gyenge osztályozókat) olyan függvényekként kezeljük, amelyeknek nemcsak bemenete van, hanem paraméterei is, azaz $f(x;\theta)$ 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:

$$\mathcal{A}\,|x\rangle\,|\theta\rangle\,|0\rangle \to |x\rangle\,|\theta\rangle\,|f(x;\theta)\rangle\,. \tag{6.1}$$

Paraméterek (kvantum osztályozók) szuperpozíciójára is elvégezhető a művelet:

$$\mathcal{A}\,|x\rangle\,\frac{1}{\sqrt{E}}\sum_\theta |\theta\rangle\,|0\rangle \to |x\rangle\,\frac{1}{\sqrt{E}}\sum_\theta |\theta\rangle\,|f(x;\theta)\rangle\,. \tag{6.2}$$

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:

$$\mathcal{W}\,|x\rangle\,\frac{1}{\sqrt{E}}\sum_\theta |\theta\rangle\,|0\rangle \to |x\rangle\,\frac{1}{\sqrt{\chi E}}\sum_\theta \sqrt{w_\theta}\,|\theta\rangle\,|0\rangle\,. \tag{6.3}$$

[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 (6.4) képlet.

$$|\tilde{x}\rangle \frac{1}{\sqrt{\chi E}} \sum_\theta \sqrt{w_\theta} \, |\theta\rangle \, |f(\tilde{x};\theta)\rangle \tag{6.4}$$

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 $f(\tilde{x};\theta) = 1$. Az $\mathcal{E}^-$ értelemszerűen ugyanez, de a 0 eredményt adó osztályozókra.

6.3.2. 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 (6.5) 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.)

$$a_\theta = \frac{1}{M} \sum_{m=1}^{M} |f(x_m;\theta) + y_m - 1| \tag{6.5}$$

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.

$$|0\ldots0;0\rangle \otimes |0\ldots0\rangle \otimes |0\rangle \otimes |0\rangle \to \frac{1}{\sqrt{E}} \sum_{i=0}^{2^\tau-1} |0\ldots0;0\rangle \, |i\rangle \, |0\rangle \, \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \tag{6.6}$$

A (6.6) 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 (6.7) lesz, egyébként elvetjük az eredményt, és újrakezdjük az algoritmust.

$$\frac{1}{\sqrt{\chi E}} \sum_{\theta} \sqrt{a_\theta} \, |0 \ldots 0; 0\rangle \, |\theta\rangle \, |0\rangle \, |0\rangle \tag{6.7}$$

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 (6.8) állapotba jut a rendszer, ahol a kimeneti qubit mérési statisztikái adják meg a kívánt eredményt (többszöri méréssel növelhető az eredmény pontonossága).

$$\frac{1}{\sqrt{\chi E}} \sum_{\theta} \sqrt{a_\theta} \, |0 \ldots 0; 0\rangle \, |\theta\rangle \, |f(\tilde{x}; \theta)\rangle \, |0\rangle \tag{6.8}$$

7. 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.

7.1. Működése klasszikusan

Az algoritmusunk alapötlete az AdaBoost algoritmusból származik, de a kvantumos megvalósíthatóság miatt a 6.3. 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.

7.1.1. 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 írja 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

7.1.2. 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 „nehézségét" 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. mintát rosszul osztályozó gyenge osztályozók összsúlya), és $\underline{w}^{(t)} = M'\underline{p}^{(t)}$ (az egyes osztályozók jól osztályozásainak száma a minták nehézségével 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 $\underline{w}^{(1)} = M'\underline{p}^{(1)}$ (pontosabban mindezek normalizálva).

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. [Wikc].)

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 $\lim_{T\to\infty} \underline{w}^{(T)} = \underline{s}_1$, az $M'M$ mátrix domináns sajátértékéhez tartozó sajátvektor. Tehát a mátrixszorzások vagy hatványozás elvégzése helyett elegendő ezt a sajátvektort meghatározni.

7.1.3. 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 AUC (19. oldal) értéket figyeltünk, így ugyanis a konkrét küszöbértéktől független jellemzőt kapunk. Ez azt is jelenti, hogy a 7.1.1. és a 7.1.2. alszekcióban leírt algoritmusokban a h_f végső hipotéziseknél szereplő 0,5 küszöb valójában nem konkrét szám, hanem azt figyeljük, hogy mennyire jól osztja kétfelé a mintákat az adott algoritmus.

A mátrixszorzásos és a sajátvektoros változatban végül nem $(0, 1)$-mátrixokat használtunk, hanem $M_{i,j}$ és $M'_{j,i}$ az i. minta j. gyenge osztályozó által vizsgált tulajdonságának az osztályozó (döntési tönk) 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 milyen lenne a „végső” hipotézis. Így képet kaphattunk a tanulás minőségéről az iterációk során.

Öt algoritmust vetünk össze: az AdaBoostnak az sklearn ensemble csomagjában található implementációját, a 6.3. 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 (7.1. és 7.2. 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

7.1. táblázat. *AUC értékek a különböző adathalmazokon*

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

7.2. táblázat. *Futásidők a különböző adathalmazokon*

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

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 csak a mintavételezéses megoldás az, ami szinte mindig vetekszik az AdaBoosttal, vagy jobb annál.

7.2. Kvantumos megvalósítás

A kvantum osztályozó algoritmus (6.3. szekció) 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.

7.2.1. 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:

$$|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \otimes |0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle\,. \tag{7.1}$$

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

Első lépés: (7.2)→(7.3). A 6.3. szekció kvantum osztályozójának tanításával megegyezően történik.

$$\frac{1}{\sqrt{E}}\sum_{i=0}^{2^\tau-1}|0\ldots0;0\rangle\,|i\rangle\,|0\rangle\,\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \tag{7.2}$$

$$\downarrow$$

$$\frac{1}{\sqrt{\chi_1 E}}\sum_{\theta}|0\ldots0;0\rangle\,\sqrt{a_\theta}\,|\theta\rangle\,|0\rangle\,|0\rangle\otimes|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \tag{7.3}$$

Második lépés: (7.4)→(7.5). 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.

$$\frac{1}{\sqrt{\chi_1 E}} \sum_{\theta} |0\ldots0;0\rangle \sqrt{a_\theta}\,|\theta\rangle\,|0\rangle\,|0\rangle \otimes \frac{1}{\sqrt{N}} \sum_{|x;y\rangle} |x;y\rangle\,|0\ldots0\rangle\,|0\rangle\,\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle) \tag{7.4}$$

$$\downarrow$$

$$|0\ldots0;0\rangle\,|0\ldots0\rangle\,|0\rangle\,|0\rangle \otimes \frac{1}{\sqrt{\chi_2 N}} \sum_{|x;y\rangle} \sqrt{w_{(x;y)}}\,|x;y\rangle\,|0\ldots0\rangle\,|0\rangle\,|1\rangle \tag{7.5}$$

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.

7.2.2. 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 kvantuminformatika és a gépi tanulás is. A részletesebben megvizsgált módszerek segíthetnek való életbeli problémák hatékony megoldásában.

A II. részben bemutatott MAXKLIKK és maximális teljes párosítás probléma például hálózatokban meglévő „közösségek" keresésében segíthet. A domináns halmaz pedig a hálózatoknak a támadásokkal szembeni ellenállóképességének ellenőrzésében válhat hasznossá. (Pl. egy elemű domináns halmaz: SPOF – Single Point Of Failure – egyetlen csomópont kiesése tönkreteheti az egész hálózatot.) Így az általam adott beágyazások ezek és hasonló problémák hatékony megoldását segíthetik.

A dolgozat III. részében a legtöbb szó osztályozási problémák megoldásáról esett. Bár többször is csak bináris osztályozásra tértünk ki, ezek mindig kiterjeszthetők több osztály esetére. A kvantum döntési fákkal kapcsolatos eredményeim gyorsabb, egyszerűbb számítást tesznek lehetővé. Az új osztályozó algoritmusunk klasszikus változata pedig bizonyos problémáknál pontosabb osztályozást érhet el az eddig használt módszereknél, ezt láttuk a futási eredményeknél.

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 a tárgyalt feladatok megoldása – kellő kapacitású és stabilitású kvantumszámítógépek kereskedelmi megjelenésével – gyorsabbá válhat.

Továbblépési lehetőségek

További munka tárgyát képezheti a dolgozat lehűtéses modellt tárgyaló (II.) részében leírt leképezések és beágyazások optimalizálási lehetőségeinek vizsgálata. Ez például konkrétan abban valósulhat meg, hogy minél kevesebb fizikai kvantumbitet használjuk egy-egy logikai qubit reprezentálására. Ezzel ugyanis elérhetjük, hogy adott méretű Kiméra gráfba nagyobb bemeneti gráf is beágyazható legyen, és így nagyobb méretű problémák hatékony megoldására nyílik lehetőség.

A gépi tanulásról szóló (III.) résszel kapcsolatban 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, Kabódi László és Pereszlényi Attila 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 a dolgozat III. részében bemutatott eredmények.

Irodalomjegyzék

[Alt06] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor és Valyon József. *Neurális hálózatok*. Panem Könyvkiadó Kft., 2006.

[BB08] 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.

[BFOS83] Leo M. Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *CART: Classification and Regression Trees*. 01 1983.

[D-Wa] D-Wave Systems. Introduction to the D-Wave quantum hardware (2018. 11. 24.). `https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware`.

[D-Wb] D-Wave Systems. Meet D-Wave (2018. 11. 24.). `https://www.dwavesys.com/our-company/meet-d-wave`.

[D-Wc] D-Wave Systems. Physics of quantum annealing – Hamiltonian and eigenspectrum (2018. 10. 06.). `https://www.youtube.com/watch?v=tnikftltqE0`.

[D-Wd] D-Wave Systems. What is quantum annealing? (2018. 10. 06.). `https://www.youtube.com/watch?v=zvfkXjzzY0o`.

[Dah13] Edward D. Dahl. White paper: Programming with D-Wave: Map coloring problem. Technical report, D-Wave Systems Inc., 2013.

[Deu89] 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.

[Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[FS97] 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.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[GK18] Martin Giles and Will Knight. Google thinks it's close to "quantum supremacy." here's what that really means. *MIT Technology Review*, March 2018.

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

[Gro96] 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.

[Kni17] Will Knight. IBM raises the bar with a 50-qubit quantum computer. *MIT Technology Review*, November 2017.

[KSH12] Christine Klymko, Blair B. Sullivan, and Travis Humble. Adiabatic quantum programming: Minor embedding with hard faults. *Quantum Information Processing*, 13, 10 2012.

[L+14] T. Lanting et al. Entanglement in a quantum annealing processor. *Phys. Rev. X*, 4:021041, May 2014.

[LB14] Songfeng Lu and Samuel L. Braunstein. Quantum decision tree classifier. *Quantum Information Processing*, 13(3):757–770, March 2014.

[Nor17] Amy Nordum. China demonstrates quantum encryption by hosting a video call. *IEEE Spectrum*, October 2017.

[RVO+15] Eleanor G. Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14(1):1–36, Jan 2015.

[Sho94] 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.

[SP18] Maria Schuld and Francesco Petruccione. Quantum ensembles of quantum classifiers. In *Scientific Reports*, volume 8, February 2018.

[Sza17] Szabó Dániel. Gráfelméleti algoritmusok beágyazása D-Wave kvantumszámítógépbe. In *Tudományos Diákköri Konferencia*. BME-VIK, 2017.

[Sza18] Szabó Dániel. Gépi tanulás és kvantuminformatika. In *Tudományos Diákköri Konferencia*. BME-VIK, 2018.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg, 1995.

[VJ01] 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.

[Wika] Wikipedia. D-Wave Systems (2018. 11. 24.). `https://en.wikipedia.org/wiki/D-Wave_Systems`.

[Wikb] Wikipedia. List of regions of Canada (2018. 11. 29.). `https://en.wikipedia.org/wiki/List_of_regions_of_Canada`.

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

[Wikd] Wikipedia. Quadratic unconstrained binary optimization (2018. 11. 24.). `https://en.wikipedia.org/wiki/Quadratic_unconstrained_binary_optimization`.

[Wike] Wikipedia. Qubit (2018. 11. 30.). `https://en.wikipedia.org/wiki/Qubit#Physical_implementations`.

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

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

[Yan78] Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 253–264, New York, NY, USA, 1978. ACM.