Gráfok a gépi tanulásban

Szakdolgozat

Matematika Bsc, alkalmazott matematikus szakirány

Szögi Evelin

Témavezető: Dr. Grolmusz Vince, egyetemi tanár
Számítógéptudományi Tanszék

Eötvös Loránd Tudományegyetem
2020

Tartalomjegyzék

NYILATKOZAT

Név: Szögi Evelin

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: GN3M24

Szakdolgozat címe:
Gráfok a gépi tanuásban

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2020. 05. 31.

Szögi Evelin

a hallgató aláírása

Köszönetnyilvánítás

Mindenekelőtt szeretném megköszönni a témavezetőmnek, Grolmusz Vince tanár úrnak az érdekes témafelvetést és azt, hogy felkeltette érdeklődésemet a gépi tanulás agygráfokon való alkalmazása iránt. Köszönöm a rendszeres konzultációkat, az iránymutatást, a jó ötleteit és tanácsait. Szeretném megköszönni kutatótársamnak, Keresztes Lászlónak a témához való mindig lelkes és motiváló hozzáállást, a közös kutatásainkba fektetett munkát és a rengeteg segítséget. Végül szeretném megköszönni a családomnak és barátaimnak a támogatásukat és biztatásukat.

1. Bevezető

A természettudományok számos területe foglalkozik olyan objektumokkal, amelyek adatokból és a közöttük fennálló kapcsolatokból állnak. Az ilyen objektumok szerkezete leírható gráfokkal: a gráfok csúcsai kisebb objektumokat jelölnek, a gráf élei pedig az ezek között lévő kapcsolatokat reprezentálják. Sok esetben igaz az, hogy ha két objektumhoz két olyan gráf tartozik, amelyek valamilyen szempontból hasonlóak, akkor a két objektum is hasonlít egymáshoz bizonyos tulajdonságot tekintve, vagy ha két csúcs közel van egymáshoz a gráfban, akkor az általuk reprezentált objektumok között is van kapcsolat. Például a kémiában a molekulák szerkezete leírható gráfokkal, ahol egy molekulában jelen lévő atomok tekinthetők a gráf csúcsainak, az atomok közötti kötések a gráf éleinek. Ha két molekula szerkezete hasonló, akkor bizonyos tulajdonságaikat tekintve a két molekula hasonlóan viselkedik. Egy másik példaként megemlíthetőek az agygráfok, ahol a gráf csúcsai agyterületeket jelölnek, a gráf élei pedig az ezen területek közötti kapcsolatokat. Korábbi kutatások bebizonyították azt, hogy az egyének egyes tulajdonságai összefüggésben vannak az egyén agygráfjával. Az interneten lévő weboldalak is tekinthetők egy gráf csúcsainak, az oldalak közötti hivatkozások a gráf éleinek. Ha két csúcs egymáshoz közel helyezkedik el ebben a gráfban, akkor jó eséllyel az általuk jelölt weboldalak hasonló témájúak. A mérnöki tudományok területén például az objektumfelismerésben az objektum alakjához bizonyos módszerekkel gráfot rendelnek hozzá (skeletonization), ezután két gráf hasonlóságából következtetnek az objektumok alakjának hasonlóságára. A gyakorlatban általában az adatok gráf reprezentációit már ismerjük, és a kihívást az jelenti, hogy az eltárolt hasonlóságra hogyan jövünk rá. Ilyenkor gyakran érdemes a gépi tanulás eszközeihez fordulni, melyeknek köszönhetően tanítóadatok segítségével prediktálhatjuk az újonnan érkező gráf szerkezetű adatok által eltárolt objektumok egyes tulajdonságait.

Általában két esetet különböztethetünk meg: egy gráfon belül a csúcsok közelségéből vagy a csúcsok hasonló szerepéből következtetünk a csúcsok által reprezentált objektumok hasonlóságára, vagy pedig két gráf hasonlóságából történik ez a következtetés. Az első esetben egy gráfon belül kell a gráf csúcsait klasszifikálni, vagy pedig a gráf csúcsaihoz értéket rendelni, vagyis az input egyetlen gráfból áll.

Ezen belül is megkülönböztethetünk még eseteket: ha a feladat az, hogy a gráf csúcsaihoz diszkrét címkéket rendeljünk, ahol az input gráf néhány csúcsának a címkéjét ismerjük, és ezekből a címkékből és a gráf szerkezetéből kell következtetni a többi csúcs címkéjére. Előfordulhatnak olyan esetek is, amikor az input gráf néhány csúcsához ún. feature vektorok vannak hozzárendelve, a feladat pedig a többi csúcshoz tartozó vektorok kiszámítása. Az ilyen jellegű alkalmazások területén, amikor az input egyetlen gráf, rendkívül erős eszközök a gráf konvolúciós hálók és a gráf neurális hálók [1, 2]. Az ilyen esetekben sokszor csak kevés csúcs címkéjének vagy feature vektorának ismerete is elég ahhoz, hogy magas pontosságot érjen el az adott háló a többi csúcshoz tartozó értékek predikciójában. A szakdolgozatomban az utóbbi esetet fogom részletesebben tárgyalni, ezek közül is azt az esetet, amikor klasszifikáció a feladat, vagyis amikor egy gépi tanulási módszer több gráfot kap inputként, és ezeket a gráfokat kell kettő vagy több osztályba sorolni. Ezt a problémát többféleképpen meg lehet közelíteni. Egy lehetséges megközelítés az, hogy a gráfokat beágyazzuk $\mathbb{R}^d$-be, ezután ezekkel a d dimenziós vektorokkal dolgozunk tovább, ugyanis a gépi tanulási módszerek túlnyomórészt valós vektorokat kapnak inputként és ezekkel dolgoznak. A természetes nyelvfeldolgozás (natural language processing - NLP) területén használatos, szavakat vagy akár egész dokumentumokat d dimenziós térbe beágyazó algoritmusok ötletéből kiindulva készültek már algoritmusok, melyek gráfokat ágyaznak be szavak helyett. A szavakat beágyazó algoritmusok mögött az a gondolat áll, hogy ha két szó gyakran hasonló kontextusban szerepel, akkor a jelentésük is hasonló lehet, így az $\mathbb{R}^d$-be való beágyazásaiknak is közel kell lenniük egymáshoz. Ha két dokumentum hasonló szavakból épül fel, akkor a két dokumentum beágyazásával kapott vektorok távolsága legyen kicsi. A gráfbeágyazó algoritmusok fő ötlete az, hogy ahogyan az NLP területén használt algoritmusokkal a szavak beágyazásából megkapható egész dokumentumok beágyazása, úgy egy gráf beágyazása is megkapható a benne előforduló részgráfok beágyazásából [3]. Valós vektorok klasszifikációjára használatos gépi tanulási módszerek között népszerűek a mesterséges neurális hálók (artificial neural network - ANN) és a szupport-vektor gépek módszere (Support-Vector Machine – SVM)[4]. Ha valamilyen módszerrel a gráfokat már beágyaztuk $\mathbb{R}^d$-be,

akkor könnyen felépíthetünk egy neurális hálót, amely a beágyazott vektorokat klasszifikálja. Ha az adatok két osztályba sorolhatók, akkor az SVM módszernél megpróbálunk keresni egy olyan hipersíkot, amely elválasztja $\mathbb{R}^d$-ben a két osztály elemeit. Ezt a feladatot fel lehet írni egy optimalizálási feladatként, melyben az optimalizálandó függvény az ismert címkéjű adatok páronkénti skalárszorzatait tartalmazza. Előfordulhat azonban, hogy nem létezik olyan hipersík, amely elválasztja a két osztályt. Ilyenkor azt a hipersíkot akarjuk megkeresni, amely a lehető legkevésbé klasszifikálja félre a megadott pontokat. Mivel nem létezik minden esetben tökéletesen szeparáló hipersík, megpróbálkozhatunk az ún. kernel trükkel [4, 5]. Ez azt jelenti, hogy nem $\mathbb{R}^d$-ben próbáljuk egy hipersíkkal szétválasztani a két osztályt, hanem egy másik térben. Ilyenkor egy leképezéssel a meglévő adatainkat egy másik skalárszorzatos térbe képezzük, és az optimalizálási feladatban az adatok páronkénti skalárszorzatait lecseréljük az új térben vett páronkénti skalárszorzataikra. Gyakran magát a leképezést, amely az új térbe képezi a meglévő adatainkat, nem kell kiszámolni, anélkül is ki tudjuk számítani az adataink képeinek skalárszorzatát. Kernelfüggvényeknek nevezik azokat a szimmetrikus, kétváltozós függvényeket, melyek az új térben vett skalárszorzatot adják vissza. A kernelfüggvényekre úgy is gondolhatunk, hogy valamilyen szempontból az adatok hasonlóságát fejezik ki egy valós számmal.

A dolgozatomban olyan kernelfüggvényekkel fogok foglalkozni, amelyek gráfokon vannak értelmezve. A 3.2.-es fejezetben szó lesz arról, hogy milyen tulajdonságokkal kell rendelkeznie egy függvénynek, hogy az a valós szám, amelyet két gráfhoz hozzárendel, egy skalárszorzatos térben a két gráf képének skalárszorzata legyen.

A dolgozat 9. fejezetében agygráfok klasszifikációjával fogok foglalkozni, erre kernelizált SVM-et fogok használni. Az agygráfokat röviden a 2. fejezetben mutatom be. Az agygráfokat biológiai nem szerint sorolom két osztályba. Korábbi kutatások rávilágítottak arra, hogy a férfi és a női agygráf számos gráfparaméterben eltér egymástól [6, 7]. A dolgozatban lévő klasszifikáció az agygráfstruktúra (különböző élek megléte/nemléte) alapján történik.

Az ez utáni fejezetekben gráfok alatt mindig véges gráfokat kell érteni. Néha irányított, máskor irányítatlanok lesznek a gráfok, ezt egy adott kernelfüggvény

tárgyalása előtt mindig megemlítem. Ha irányított gráfokról van szó egy kernel esetén, de az adataink irányítatlan gráfokat tartalmaznak, akkor egy irányítatlan él megduplázásával és kétféle megirányításával készíthető irányított gráf az irányítatlan gráfból. Sok esetben a gráf csúcsai vagy élei meg lesznek címkézve, ugyanis az alkalmazásokban is többször előfordul, hogy a gráfok csúcsai vagy élei különböző csoportokba oszthatók. Például a kémiai alkalmazásokban, ahol a gráfok molekulákat reprezentálnak, a csúcscímkék jelölhetnek különböző atomokat, az élcímkék pedig a kötések fajtáit. Az élek rendelkezhetnek még egyéb számszerű attribútumokkal is, például hosszúságokkal vagy más élsúlyokkal. Ha egy kernelfüggvény tárgyalásánál van egy olyan feltétel, hogy a csúcsok és az élek rendelkeznek címkékkel, viszont a mi adatainkban, amelyekre a kernelt alkalmazni szeretnénk, nincsenek megcímkézve a gráfok csúcsai vagy élei, akkor minden csúcshoz hozzárendelhetjük ugyanazt a címkét (például egy '0' címkét), ugyanígy az élekhez is hozzárendelhetünk egyforma címkéket. Ha szerepel olyan feltevés, hogy az élek súlyozva vannak, viszont az adataink nem rendelkeznek élsúlyokkal, akkor minden él vehető 1-es súllyal.

2. Az agygráfok

Az agygráf tekinthető az emberi agy egy makroszkopikus modelljének, ahol a gráf csúcsai agyterületeket jelölnek, a gráf élei pedig ezen agyterületek közötti kapcsolatokat. Az agygráfok diffúziós MRI felvételek alapján készülnek, a traktográfia nevű módszer alapján a FreeSurfer program segítségével. A diffúziós MRI arra épül, hogy az agyban lévő vízmolekulák áramlásából feltérképezzük az agy szürkeállományának területei között húzódó axonkötegeket. Az agy szürkeállományában a vízmolekulák nem rendelkeznek konkrét áramlási iránnyal, viszont könnyebben áramlanak a fehérállományt alkotó axonok mentén, ami lehetővé teszi az axonkötegek feltérképezését a vízmolekulák áramlásának megfigyeléséből. Tulajdonképpen az agygráfok esetén nem lényeges az, hogy ezek a kötegek hogyan helyezkednek el a fehérállományban, minket csupán az érdekel, hogy a szürkeállomány mely területei vannak összekötve. Így egy él két csúcs között megfeleltethető annak, hogy a két agyterület között létezik-e közvetlen információcsere, míg az axonkötegek száma megmutatja, hány ilyen különböző közvetlen kapcsolat van két agyterület között, ami egyszersmind a kapcsolat erősségét is jelzi.

Az általam használt adathalmaz 1064 egyén agygráfját tartalmazza, ebből 489 férfi és 575 női agygráf. Minden egyén MRI felvételéből 5 különböző felbontású agygráf is készült, ezen 5 felbontás esetén a gráf csúcsszámai rendre 83, 129, 234, 463, 1015. Ezek mind az agy szürkeállományának valamekkora felbontását jelentik, pl. 83 csúcs esetén 83, nagyjából azonos méretű agyterületre bontjuk a szürkeállományt. Az agygráfban a csúcsoknak ezek az agyterületek felelnek meg, míg két csúcs között pontosan akkor megy él, ha a két csúcsnak megfelelő agyterület között halad idegrostköteg.

Minden felbontásban a csúcsok a következő attribútumokkal rendelkeznek: (x, y, z) térbeli koordináták; a régió neve, amelyet jelöl; a csúcs kéregbeli vagy kéregalatti. Természetesen különböző gráfok azonos indexű csúcsai ugyanazt az agyi régiót jelölik. Minden felbontás esetén az éleknek a következő attribútumai vannak: idegrostok száma; idegrostok átlagos hossza; az átlagos fractional anisotropy (FA), mely a diffúzió irányban haladását méri.

3. Szupport-vektor gépek (SVM)

3.1. A lineáris SVM

A gépi tanulás témakörében az SVM egy népszerű, gyakran alkalmazott tanulási módszer. A lineáris SVM alapfeladata: Adott n pont $\mathbb{R}^d$-ben, ezek x_1, x_2, ... x_n, valamint adott y_1, y_2, ... y_n, ahol minden i esetén $(i = 1...n)$ $y_i = 1$ vagy $y_i = -1$. A feladat az, hogy találjuk meg $\mathbb{R}^d$-ben azt a hipersíkot, amely a lehető legjobban elválasztja az x_i $(y_i = 1)$ pontokat az x_j $(y_j = -1)$ pontoktól. Két esetet különböztetünk meg: létezik olyan hipersík, mely elválasztja a két pontosztályt, vagy nem létezik szeparáló hipersík. Ezen esetek tárgyalásakor a [4]-ban bemutatott módszert követtem.

3.1.1. 1. eset: létezik szeparáló hipersík

Ha létezik olyan hipersík, amely szeparálja a két pontosztályt, akkor a feladatunk az, hogy azt a hipersíkot keressük meg, amely esetén a hipersíkhoz legközelebb eső pontok távolsága maximális. Ezt a távolságot margónak szokták nevezni, jelöljük a margót m-mel. Mivel a két pontosztály szeparálható, világos, hogy az optimális szeparálás mellett mindkét osztály azon pontjai, melyek legközelebb vannak a hipersíkhoz, $m/2$ távolságra lesznek a szeparáló hipersíktól. A cél az, hogy az m margót maximalizáljuk. Ez a feladat optimalizálási feladatként írható fel. Jelöljük w-vel egy adott szeparáló hipersík esetén a sík normálvektorát, b-vel azt az értéket, amelyre a $w \cdot x + b = 0$ egyenlet a hipersík egyenletét adja. Mivel a hipersík elválasztja a két osztály pontjait, teljesül a következő:

$$w \cdot x_i + b \geq a > 0,$$

ha $y_i = 1$ $(i = 1...n)$, és

$$w \cdot x_j + b \leq -a < 0,$$

ha $y_j = -1$ $(j = 1...n)$. A hipersík w és b paramétereit skálázhatóak úgy, hogy a-t 1-nek tudjuk választani.

Tudjuk, hogy egy x^* pont $w \cdot x + b = 0$ egyenletű hipersíktól való távolsága $\frac{|w \cdot x^* + b|}{||w||}$. Ha most az optimális szeparáló hipersík paramétereit w^* és b^* jelöli, akkor a skálázás miatt a hipersíkhoz legközelebb eső pontokat ebbe behelyettesítve

$$\frac{|w^* \cdot x + b^*|}{||w^*||} = \frac{1}{||w^*||},$$

vagyis a margó hossza $m = \frac{2}{||w^*||}$. Vagyis a szeparáló hipersík euklideszi normájának minimalizálásával tudjuk maximalizálni m értékét.

Tehát a feladatunk az, hogy minimalizáljuk az $m(w, b) = \frac{1}{2} w \cdot w$ célfüggvényt, ahol a feltételeink azok, hogy az egyik pontosztály a hipersík egyik oldalára, míg a másik pontosztály a hipersík másik oldalára essen, azaz (i, j = 1,2,...n)

$$w \cdot x_i + b \geq 1$$

minden olyan i indexre, ahol $y_i = 1$, és

$$w \cdot (-x_j) - b \geq 1$$

minden olyan j indexre, ahol $y_j = -1$. Egyszerűbben írva:

$$y_l(w \cdot x_l + b) - 1 \geq 0$$

minden l = 1,2,...n indexre. Így megkaptunk egy kvadratikus optimalizálási feladatot. Azokat az x_i vektorokat, melyek az optimális hipersíktól vett távolsága minimális, szupport-vektoroknak nevezzük.
A feladat Lagrange-duálisa:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_i^n \alpha_i \left(y_i(w \cdot x_i + b) - 1\right),$$

ahol $\alpha_i \geq 0$. A Karush-Kuhn-Tucker-tételt és a dualitástételt felhasználva a felírt optimalizálási feladat helyett nézhetjük a következő optimalizálási feladatot:

$$\max_{\alpha}(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$$

A korlátozó feltételek pedig:

$$\sum_{i=1}^n \alpha_i y_i = 0$$

$$\alpha_i \geq 0, i = 1...n.$$

Az SVM módszer a duális feladatot oldja meg, hogy megtalálja a maximális margójú szeparáló hipersíkot.

Előfordulhat olyan eset, hogy a kiindulási pontrendszerünk nem szeparálható lineárisan egy hipersíkkal. Ekkor a következőképpen járunk el:

3.1.2. 2.eset: nem létezik szeparáló hipersík

Új változókat, ún. slack változókat vezetünk be (z_i, $i = 1...n$), melyekkel a félreklasszifikálást mérjük. Megadunk egy C súlyt is, amivel büntetjük azt, ha nem jól klasszifikálunk egy-egy pontot. A minimalizálandó célfüggvény:

$$\min_{w,z,b}(\tfrac{1}{2}w \cdot w + C\sum_{i=1}^{n} z_i)$$

A korlátozó feltételek ($i = 1, 2...n$):

$$y_i(wx_i - b) \geq 1 - z_i$$

$$z_i \geq 0$$

A feladat Lagrange-duálisánál az optimalizálandó célfüggvény:

$$\max_{\alpha}(\sum_{i=1}^{n} \alpha_i - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j x_i \cdot x_j)$$

A korlátok:

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

és $i = 1, 2...n$-re

$$C \geq \alpha_i \geq 0$$

3.2. Kernelizált SVM

Ha az adataink nem szeparálhatók lineárisan, akkor felmerülhet az a kérdés, hogy ha egy ϕ függvénnyel egy másik térbe képeznénk őket, akkor ott el tudnánk-e választani a két pontosztályt. Az előzőekben láttuk, hogy ha egy szeparáló hipersíkot szeretnénk találni, akkor egy olyan optimalizálási feladatot kell megoldanunk, amely felírásában az adataink (x_i-k) egymással vett skalárszorzatai szerepelnek. Ha egy ϕ leképezéssel egy $\mathcal{F}$ Hilbert-térbe képeznénk az adatainkat, akkor megpróbálkozhatnánk egy $\mathcal{F}$-beli szeparáló hipersík megkeresésével. Ehhez elég a már említett

$$\max_{\alpha}(\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i \cdot x_j)$$

optimalizálási feladatban az $x_i \cdot x_j$ skalárszorzatokat lecserélni a $\phi(x_i)$ és $\phi(x_j)$ vektorok $\mathcal{F}$-beli skalárszorzatára. Vegyük észre, hogy itt a konkrét ϕ leképezést nem kell kiszámolnunk, elég ismerni minden i, j párra a $\phi(x_i)$ és $\phi(x_j)$ vektorok skalárszorzatát.

A [5] irodalomban található pár definíció és állítás, amelyek segítségével egyszerűen meg lehet mondani, hogy milyen tulajdonságokat kell teljesítenie egy szimmetrikus, kétváltozós k függvénynek ahhoz, hogy $k(x, y)$ egy valamilyen Hilbert-térben lévő skalárszorzatot jelentse. Legyen χ nemüres halmaz, legyen $\mathcal{F}$ Hilbert-tér.

3.1. Definíció. *A $\phi : \chi \to \mathcal{F}$ típusú leképezéseket feature függvényeknek (jellemzőfüggvényeknek) nevezzük, az $\mathcal{F}$ tér neve feature tér (jellemzőtér).*

3.2. Definíció (Gram-mátrix)**.** *Legyen adott egy szimmetrikus $k : \chi^2 \to R$ függvény, és $x_1, ...x_m \in \chi$. Ekkor az $m \times m$-es K mátrixot, ahol $K_{ij} := k(x_i, x_j)$, az $x_1, ...x_m$ pontok k függvényre vonatkozó Gram-mátrixának nevezzük.*

3.3. Definíció (Pozitív szemidefinit mátrix)**.** *Az $m \times m$-es valós és szimmetrikus K mátrix pozitív szemidefinit, ha $\sum_{i,j} c_i c_j K_{ij} \geq 0$ bármely c_i, c_j valós együtthatók esetén.*

3.4. Definíció ((Pozitív szemidefinit) kernel)**.** *Egy szimmetrikus $k : \chi \times \chi \to R$ függvényt kernelfüggvénynek nevezünk, ha minden $m \in N$ és $x_1, ...x_m \in \chi$ esetén a K Gram-mátrix pozitív szemidefinit.*

3.5. Állítás. *Tegyük fel, hogy $k : \chi \times \chi \to \mathbb{R}$ kernelfüggvény. Ekkor létezik olyan $\mathcal{F}$ Hilbert-tér (feature tér) és $\phi : \chi \to \mathcal{F}$ leképezés, hogy minden $x_i, x_j \in \chi$ esetén $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{F}}$, ahol $\langle ., . \rangle_{\mathcal{F}}$ az $\mathcal{F}$-beli skalárszorzatot jelöli. Fordítva is igaz: tetszőleges $\mathcal{F}$ Hilbert-tér és $\phi : \chi \to \mathcal{F}$ leképezés esetén a $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{F}}$ mátrix pozitív szemidefinit, vagyis a $k : \chi \times \chi \to R$, $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{F}}$ függvény kernelfüggvény.*

3.6. Megjegyzés. *Kernel trükknek nevezzük azt, amikor egy algoritmus egy k kernelfüggvényt használ, és létrehozunk egy másik algoritmust úgy, hogy az első algoritmusban a k függvényt lecseréljük egy másik k' kernelfüggvényre.*

A kernelizált SVM-ben is ezt a kernel trükköt alkalmazzuk, ugyanis a közönséges skalárszorzatot cseréljük le egy másik k kernelre. A következő fejezetekben lesz szó ezekről a k függvényekről.

4. Címkepárokon és címkesorozatokon alapuló gráfkernelek

Ebben a fejezetben a [8] cikkben ismertetett kerneleket mutatom be. Ebben a részben gráf alatt címkézett, irányított, párhuzamos élek nélküli gráfot értsünk, élsúlyokkal most ne foglalkozzunk. Egy adott gráf esetén az A mátrix adjacenciamátrixot jelöl. Tegyük fel, hogy a gráfok csúcscímkéi az $\mathcal{L} = \{l_1, l_2, ...l_{|\mathcal{L}|}\}$ véges halmazból valók. egy adott $G = (V, E)$ gráf esetén az L egy mátrix, mérete $|\mathcal{L}| \times |V|$ és $L_{ij} = 1$ pontosan akkor, ha a v_j csúcs címkéje l_i, más esetben $L_{ij} = 0$. Könnyen ellenőrizhető, hogy az LL^T mátrix átlójában az i-edik elem azt mutatja, hogy az l_i címke az adott gráf hány csúcsának a címkéje. Ismert, hogy egy gráf adjacenciamátrixának hatványai a gráf csúcsai között menő, adott hosszúságú séták számát tárolják, azaz A^n_{ij} elem a v_i-ből v_j-be menő n hosszú séták száma. Ebből könnyen adódik, hogy az $(LA^nL^T)_{ij}$ elem azt mondja meg, hogy hány olyan n hosszú séta van a gráfban, melynek egyik végpontja l_i címkéjű, a másik l_j címkéjű.

4.1. Címkepárokon alapuló gráfkernel

Tegyük fel, hogy a gráfokban csak csúcscímkék vannak, élcímkék nincsenek.

Egy G gráf esetén jelölje $W_n(G)$ az összes G-ben előforduló n élű sétát. Legyenek a $\lambda_1, \lambda_2, \lambda_3...$ számok nemnegatív valós számok. Ha $w \in W_n(G)$, akkor $l_1(w)$ jelölje a w séta első csúcsának címkéjét, $l_{n+1}(w)$ pedig az utolsó csúcs címkéjét. Leképezhetjük a gráfokat egy ϕ leképezéssel egy $|\mathcal{L}|^2$ dimenziós $\mathcal{F}$ feature térbe a következőképpen: minden (l_i, l_j) címkepárhoz tartozzon egy $|\mathcal{L}|^2$ dimenziós $v_{(l_i,l_j)}$ egységvektor úgy, hogy a $v_{(l_i,l_j)}$ vektor egy koordinátája 1-es, a többi 0, és különböző (l_i, l_j) párokhoz különböző egységvektor tartozzon. Minden G gráf esetén $\phi(G)$-t ezen vektorok lineáris kombinációjaként írjuk fel. Ebben a lineáris kombinációban a $v_{(l_i,l_j)}$ vektor együtthatója legyen

$$\sum_{n=1}^{\infty} \lambda_n |\{w \in W_n(G) : l_1(w) = l_i \wedge l_{n+1}(w) = l_j\}|.$$

A $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ skalárszorzat legyen a szokásos skalárszorzat. Ekkor egy gráfkernel definiálható a következő módon:

4.1. Definíció (Címkepárokon alapuló gráfkernel)**.** *A fenti jelölésekkel a G_1 és G_2 gráf esetén a címkepárokon alapuló k gráfkernel: $k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle_{\mathcal{F}} = \langle L_1(\sum_{i=1}^{\infty} \lambda_i A_1^i) L_1^T, L_2(\sum_{j=1}^{\infty} \lambda_j A_2^j) L_2^T \rangle$, ahol a második $\langle ., . \rangle$ skalárszorzat alatt azt kell érteni, hogy az azonos méretű mátrixok megfelelő elemeit összeszorozzuk, majd a kapott szorzatokat összeadjuk.*

Ha itt a λ_i nemnegatív valós számokat ügyesen választjuk meg, akkor a $\sum_{i=1}^{\infty} \lambda_i A^i$ sor konvergens lesz és az összeget könnyen ki lehet számítani. Ha például $\lambda_i = 1/i!$, akkor ismeretes, hogy az említett összeg e^A alakban írható. Ha viszont megfelelő λ választással minden i-re $\lambda_i = \lambda$, akkor az összeg $(I - \lambda A)^{-1}$.

Ha az előforduló címkék száma (vagyis $|\mathcal{L}|$) kicsi, akkor $\mathcal{F}$ dimenziója is kicsi, így nem biztos, hogy elég hasznos lesz az alkalmazásokban ez a gráfkernel. A következőként említett gráfkernel esetében már végtelen dimenziós lesz az $\mathcal{F}$ feature tér.

4.2. Címkesorozatokon alapuló gráfkernel

Ebben a részben tegyük fel, hogy a csúcsok és az élek is címkézve vannak. Ha a gráfadathalmazunkban nincsenek csúcscímkék, akkor minden csúcshoz rendeljük hozzá a $'0'$ címkét, ha nincsenek élcímkék, akkor minden élhez rendeljük hozzá az $'1'$ címkét.

Jelölje S_n az összes lehetséges címkesorozat halmazát, amely egy n élű séta során előfordulhat. Ha $s \in S_n$, akkor s_1 a séta első csúcsának címkéje, s_2 az ezt követő él címkéje, s_3 a második csúcs címkéje, ... s_{2n+1} a séta utolsó csúcsának címkéje. Egy G gráf esetén $W_n(G)$ jelentse ugyanazt, mint az előbb. Ha $w \in W_n(G)$, akkor w-re úgy tekintünk, mint a séta $n+1$ csúcsának és n élének váltakozó felsorolása. Az $\mathcal{F}$ feature tér végtelen dimenziós lesz, minden $n \in \mathbb{N}$-re S_n minden eleméhez különböző, 0-1 koordinátájú egységvektorok fognak tartozni. Legyen $\lambda_i \geq 0$ $(i \in \mathbb{N})$. Ha $s \in S_n$, $s = s_1 s_2 ... s_{2n+1}$, akkor egy G gráfra a következőképpen definiáljuk az s-hez tartozó feature értéket (vagyis az s-hez tartozó egységvektor együtthatóját):

$$\phi_s(G) = \sqrt{\lambda_n} |\{w \in W_n(G), \forall i : l_i(w) = s_i\}|,$$

ahol $l_i(w)$ a w-ben előforduló i. címke.

Ha $G_1 = (V_1, E_1)$ és $G_2 = (V_2, E_2)$ két gráf, akkor többféleképpen is definiálhatjuk a direkt szorzatukat. Ebben a részben érdemes a következő definíciót nézni: a direkt szorzatuk egy $G_\times = (V_\times, E_\times)$ gráf, ahol $V_\times = \{(v_1, v_2) \in V_1 \times V_2 : l(v_1) = l(v_2)\}$ és $E_\times = \{((u_1, v_1), (u_2, v_2)) \in V_\times^2 : (u_1, v_1) \in E_1 \wedge (u_2, v_2) \in E_2 \wedge (l(u_1, v_1) = l(u_2, v_2))\}$, ahol $l(v)$ a v csúcs címkéje, $l(u, v)$ pedig az (u, v) él címkéje. A direkt szorzat gráf is címkézett: $G_\times$ egy $v = v_1 \times v_2$ csúcsának címkéje az őt definiáló v_1 és v_2 címkéje (a definíció miatt a kettő ugyanaz). Hasonlóan $G_\times$ egy élének címkéje az őt definiáló két él címkéje.

Nem nehéz látni, hogy ha teszünk egy n élből álló w sétát $G_\times$-ben, akkor tudunk tenni egy olyan n élből álló sétát G_1-ben és G_2-ben is, mely esetén a csúcsokon és az éleken váltakozva előforduló címkék pontosan ugyanazok, mint amelyek w-ben fordulnak elő, és ugyanez fordítva is elmondható. Az is könnyen adódik, hogy $s \in S_n$ esetén

$$|\{w \in W_n(G_\times), \forall i : l_i(w) = s_i\}| = |\{w \in W_n(G_1), \forall i : l_i(w) = s_i\}||\{w \in W_n(G_2), \forall i : l_i(w) = s_i\}|.$$

Most értelemszerűen G_1 és G_2 esetén a $\langle G_1, G_2 \rangle_\mathcal{F}$ skalárszorzatot úgy kapjuk, hogy G_1 és G_2 megfelelő feature értékeit összeszorozzuk minden lehetséges s címkesorozat esetén, és a kapott szorzatokat összeadjuk. Ekkor a következő gráfkernelt kapjuk:

4.2. Definíció (Címkesorozatokon alapuló gráfkernel)**.** *A fenti jelölésekkel a G_1 és G_2 gráf esetén a címkesorozatokon alapuló k gráfkernel: $k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle_\mathcal{F} = \sum_{i,j=1}^{|V_\times|} [\sum_{n=0}^{\infty} \lambda_n A_\times^n]_{ij}$, ahol $A_\times$ a direkt szorzat gráf adjacenciamátrixát jelöli.*

Megfelelő λ_i együtthatók megválasztásával a fenti $\sum_{n=0}^{\infty} \lambda_n A_\times^n$ sor konvergens lesz és a sorösszeg egyszerűen kiszámolható.

5. Véletlen sétákon alapuló gráfkernel

Ebben a részben a [9] cikkben ismertetett kernelekről lesz szó.

5.1. Jelölések

Ebben a részben irányított gráfokkal foglalkozunk, feltesszük még, hogy a gráfok erősen összefüggőek. $V = \{v_1, v_2, ...v_n\}$ véges halmaz jelöli egy adott gráf csúcshalmazát, $E \subset V \times V$ jelöli a gráf irányított éleinek halmazát, $(v_i, v_j) \in E$ pontosan akkor, ha a gráfban vezet él v_i-ből v_j-be. Feltesszük még, hogy a gráfokban nincsenek hurokélek vagy párhuzamos élek. Ha irányítatlan gráfokkal is szeretnénk dolgozni, akkor egy élet vegyünk be két különböző irányítással, vagyis $(v_i, v_j) \in E$ pontosan akkor, ha $(v_j, v_i) \in E$. Gyakran olyan gráfokkal dolgozunk, amelyek élei súlyozva vannak, $w_{ij} > 0$ jelölje a (v_i, v_j) él súlyát. Ha nincs (v_i, v_j) él, akkor legyen $w_{ij} = 0$. Ha egy gráfban nincsenek élsúlyok, akkor az $\tilde{A}$ mátrix az adjacenciamátrix transzponáltját jelöli, vagyis $\tilde{A}_{ij} = 1 \Leftrightarrow (v_j, v_i) \in E$, egyébként $\tilde{A}_{ij} = 0$. Ha a gráf élei súlyozottak, akkor 1-esek helyett a megfelelő súlyt (w_{ji}) írjuk $\tilde{A}_{ij}$-be. Az A mátrix jelölje az $\tilde{A}$ mátrix olyan normalizált változatát, melyre A minden oszlopában 1 lesz az elemek összege. $A := \tilde{A}D^{-1}$, ahol D egy diagonális mátrix, melyben $D_{jj} = \sum_i \tilde{A}_{ij}$. Ezentúl a w_{ij} számok már ennek a normalizált mátrixnak az elemeit jelölik.

5.2. Véletlen séták

Az A normalizált mátrix elemeire tekinthetünk úgy, mint valószínűségekre. Nézzünk véletlen sétákat a gráfokon. Kezdetben adott egy p_0 n magas oszlopvektor (n az adott gráf csúcsainak száma), melynek i-edik eleme annak a valószínűségét tartalmazza, hogy a véletlen séta a v_i csúcsból indul. Ha nincsenek előzetes ismereteink p_0-ról, akkor minden koordinátája legyen $1/n$. Tegyük fel, hogy a véletlen sétánk első k csúcsa a $v_{i_1}, v_{i_2}, ...v_{i_k}$ csúcsok. Ekkor $w_{i_{k+1}i_k}$ mondja meg annak a valószínűségét, hogy a $v_{i_{k+1}}$ csúcs lesz a séta következő csúcsa. A $p_t := A^t p_0$ vektor azt mondja meg, hogy egy t élű séta esetén mekkora a valószínűsége annak, hogy az egyes csúcsok lesznek a séta $(t+1)$-edik pontjai, vagyis a p_t i-edik koordinátája adja annak a

valószínűségét, hogy egy t élű séta utolsó pontja v_i. A gráfkernel definiálásakor minden csúcshoz fog tartozni egy megállási valószínűség is. Jelölje q azt az n magas oszlopvektort, mely az egyes csúcsokhoz tartozó megállási valószínűségeket tartalmazza, vagyis a j-edik koordináta annak a valószínűsége, hogy ha a véletlen séta a v_j ponthoz ért, akkor ott befejeződik. Előzetes ismeretek nélkül itt is, mint p_0-nál, lehet minden koordináta $1/n$. Ekkor $q^T p_t$ lesz annak a valószínűsége, hogy egy véletlen séta t lépés után megáll. Ha q minden koordinátáját $1/n$-nek választjuk, akkor ez az érték minden t esetén ugyanaz lenne, ezért ilyenkor a kernelértékek kiszámításakor az $\tilde{A}$ mátrix D^{-1}-zel való normálását nem végezzük el.

5.3. Gráfkernel definiálása

Ebben a részben címkézett gráfok alatt csak élcímkével rendelkező gráfokat értsünk, csúcscímkékkel most ne foglalkozzunk. Legyen $\mathcal{L}$ az élek címkéinek halmaza. Minden n csúcsú gráfhoz tartozik egy $n \times n$-es L címkemátrix, ahol $L_{ij} \in \mathcal{L}$ és L_{ij} a (v_j, v_i) él címkéje. Feltehető, hogy van $\mathcal{L}$-nek egy speciális eleme, ξ, és ha a gráfban nincs (v_j, v_i) él, akkor $L_{ij} = \xi$.

Többféleképpen is definiálható két gráf direkt szorzata. A [9] cikk szerzői egy olyan definíciót használnak, ahol nem veszik figyelembe a gráfok élcímkéit. Ha $G_1 = (V_1, E_1)$ és $G_2 = (V_2, E_2)$ két gráf, akkor a direkt szorzatuk egy $G_\times = (V_\times, E_\times)$ gráf, ahol $V_\times = V_1 \times V_2$ és $E_\times = \{((u_1, v_1), (u_2, v_2)) \in V_\times^2 : (u_1, v_1) \in E_1 \wedge (u_2, v_2) \in E_2\}$. Ha $\tilde{A}_1$ jelöli a G_1 gráf adjacenciamátrixának transzponáltját, $\tilde{A}_2$ pedig a G_2 gráfét, akkor a direkt szorzat gráf adjecenciamátrixának transzponáltja a következőképpen írható fel: $\tilde{A_\times} = \tilde{A}_1 \otimes \tilde{A}_2$. Ha A egy $a_1 \times a_2$-es mátrix, B egy $b_1 \times b_2$-es, akkor az $A \otimes B$ mátrix egy $a_1 b_1 \times a_2 b_2$-es mátrix lesz, és

$$A \otimes B := \begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1a_2}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{a_1 1}B & A_{a_1 2}B & \dots & A_{a_1 a_2}B \end{bmatrix}.$$

Az $A \otimes B$ mátrixot az A és B mátrixok Kronecker-szorzatának nevezik. Ugyanez igaz a normált mátrixokra is: ha A_1 az $\tilde{A}_1$ normáltja, A_2 az $\tilde{A}_2$ normáltja, akkor a $G_\times$ gráfhoz tartozó $A_\times$ így írható fel: $A_1 \otimes A_2$.

A cél egy olyan kernel definiálása, mely két gráf hasonlóságát a rajtuk végrehajtott véletlen séták hasonlóságával méri. Ezt meg lehet úgy tenni, hogy G_1-en és G_2-n is véletlen sétákat generálunk, és ezután megszámoljuk, hogy hány séta van, amely mindkét gráf esetén előjött. Ha adottak a p_{01} és p_{02} kezdeti valószínűségek, melyek azt mutatják, hogy egy véletlen séta G_1-ben és G_2-ben mekkora valószínűséggel indul az egyes csúcsokból, akkor $G_\times$ gráfra ez a $p_\times$ valószínűség a következőképpen írható: $p_\times = p_{01} \otimes p_{02}$. Ugyanez elmondható a megállási valószínűségekre is: $q_\times = q_1 \otimes q_2$.

Ha van egy pozitív szemidefinit k kernel, mely az $\mathcal{L} \times \mathcal{L}$ halmazban lévő címkepárokhoz rendel valós számokat, akkor jelölje $\mathcal{F}$ a kernelhez tartozó feature teret, $\phi : \mathcal{L} \to \mathcal{F}$ pedig a feature leképezést. Feltesszük, hogy a ϕ leképezés az $\mathcal{L}$ halmaz speciális ξ eleméhez $\mathcal{F}$ nullelemét rendeli. Ha adott egy G gráf és a hozzá tartozó L címkemátrix, akkor a $\phi(L)$ mátrixot G feature mátrixának nevezzük, ahol $[\phi(L)]_{ij}$ az az $\mathcal{F}$-beli elem, ahova az L_{ij} elemet képezi a ϕ leképezés.

Legyen adott két élcímkézett gráf: G_1 és G_2, a hozzájuk tartozó címkemátrixok L_1 és L_2. A $G_\times$ direkt szorzat gráfhoz definiálható egy súlymátrix:

$$W_\times = \phi(L_1) \otimes \phi(L_2),$$

ahol a $\phi(L_1)$ és $\phi(L_2)$ mátrixok elemei $\mathcal{F}$-beli elemek, és a Kronecker-szorzat definíciójában a mátrixok elemeinek szorzásakor az $\mathcal{F}$-beli skalárszorzást használjuk. Ha nincsenek élcímkék, akkor nézhetjük azt az egyszerű esetet, amikor $\phi(L) = \tilde{A}$. Ekkor $W_\times = \tilde{A_\times}$. Ha pedig $\phi(L) = A$, akkor $W_\times = A_\times$. Ha vannak élcímkék, $\mathcal{L} = \{l_1, l_2, ...l_r\}$, akkor a ϕ leképezés definiálható úgy, hogy az l_i címkét az r magas e_i egységvektorba viszi. Ilyenkor egy adott G gráf esetén a $\phi(L)$ mátrixot érdemes normálni is, vagyis ha $L_{ij} = l_k$, akkor $\phi(L_{ij}) = \frac{1}{d_i} e_k$, ahol d_i jelöli a már definiált D diagonális mátrix i-edik diagonálelemét. A $\phi(L)$ mátrix többi eleme legyen 0. Így a $W_\times$ mátrix egy eleme akkor nem lesz 0, ha mindkét gráfban létezik a megfelelő él és ugyanaz a címkéjük. Bevezethető egy újfajta jelölés: minden $l = 1, 2, ...r$ esetén jelölje $A^{(k)}$ azt a mátrixot, mely i-edik sorának j-edik eleme A_{ij} pontosan akkor, ha $L_{ij} = l_k$, egyébként pedig 0. Ekkor a $W_\times$ mátrix így is felírható: $\sum_{i=1}^{r} A_1^{(i)} \otimes A_2^{(i)}$.

Nem nehéz látni, hogy ha két gráf direkt szorzat gráfjában teszünk egy véletlen sétát, akkor az a séta megfelel egy-egy sétának mindkét gráfban. Felhasználva azt, hogy $A_\times = A_1 \otimes A_2$, $A_\times^k$ minden eleme egy olyan szám, amely azt mutatja, hogy mekkora a valószínűsége annak, hogy egyszerre tehetünk meg egy-egy k élű véletlen sétát G_1-ben és G_2-ben, ahol az egyik séta G_1 két rögzített csúcsa között megy, a másik pedig G_2 két rögzített csúcsa között.

$W_\times$-re és a hatványaira tekinthetünk úgy, hogy a G_1 és G_2 gráfok hasonlóságát a bennük végrehajtható, különböző hosszúságú véletlen sétákban mérik. Ha a $p_\times$ és $q_\times$ kezdeti és megállási valószínűségek adottak, akkor egy gráfkernelt definiálhatunk a következőképpen:

5.1. Definíció (Véletlen sétákon alapuló gráfkernel)**.** $k(G_1, G_2) := \sum_{k=0}^{\infty} \mu(k) q_\times^T W_\times^k p_\times$, *ahol minden* k*-ra a* $\mu(k) \geq 0$ *együtthatókat úgy kell megválasztani, hogy a sor konvergens legyen.*

Ha a definícióban szereplő sor konvergens, akkor egy pozitív szemidefinit kernelt kapunk. Ennek belátásához először érdemes bevezetni a $vec(.)$ jelölést: ha A egy $p \times q$-as mátrix, akkor $vec(A)$ egy pq magas oszlopvektor lesz, melyet úgy kapunk, hogy A első oszlopával kezdve egymás alá pakoljuk A oszlopait, vagyis

$$vec(A) := \begin{bmatrix} A_{*1} \\ \vdots \\ A_{*q} \end{bmatrix},$$

ahol A_{*j} az A mátrix j-edik oszlopa. Be lehet látni, hogy ha A, B, C mátrixok, méreteik olyanok, hogy az ABC szorzás értelmes, akkor

$$vec(ABC) = (C^T \otimes A) vec(B).$$

Teljes indukcióval belátható a következő:

$$W_\times^k p_\times = vec[\phi(L_1)^k p_1 (\phi(L_2)^k p_2)^T].$$

Nézzük ezt: $q_\times^T W_\times^k p_\times$. Ha $q_\times$ helyett $q_1 \otimes q_2$-t írunk, $W_\times^k p_\times$-re alkalmazzuk a második tulajdonságot, akkor ezt kapjuk:

$$q_\times^T W_\times^k p_\times = (q_1 \otimes q_2)^T vec[\phi(L_1)^k p_1 (\phi(L_2)^k p_2)^T].$$

Ismert, hogy $(X \otimes Y)^T = X^T \otimes Y^T$. Ezt a tulajdonságot és a fenti első tulajdonságot alkalmazva

$$q_\times^T W_\times^k p_\times = (q_1 \otimes q_2)^T vec[\phi(L_1)^k p_1 (\phi(L_2)^k p_2)^T] = vec[q_2^T \phi(L_2)^k p_2 (\phi(L_1)^k p_1)^T q_1] = (q_1^T \phi(L_1)^k p_1)^T (q_2^T \phi(L_2)^k p_2).$$

Ha most $\rho_k(G) := q^T \phi(L) p$, akkor a kapott kifejezés $\rho_k(G_1)^T \rho_k(G_2)$-vel egyenlő, ez pedig pozitív szemidefinit. Felhasználva azt, hogy pozitív szemidefinit kernelek nemnegatív lineáris kombinációi és pontonkénti limeszei is pozitív szemidefinit kernelt eredményeznek, a $k(G_1, G_2) := \sum_{k=0}^{\infty} \mu(k) q_\times W_\times^k p_\times$ kernel valóban pozitív szemidefinit, ha a sor konvergens.(hivatkozás)

Ha a $\mu(k)$ együtthatókat úgy választjuk meg, hogy $\mu(k) = \lambda^k$ olyan $\lambda > 0$-ra, amely esetén a sor konvergens, akkor a fenti kernel így írható:

$$k(G_1, G_2) = q_\times^T (I - \lambda W_\times)^{-1} p_\times.$$

A [8] cikkben bemutatott kernel is megkapható alkalmas $\mu(k)$ együtthatók mellett. Ha $|V(G_1)| = n_1$, $|V(G_2)| = n_2$, az említett kernel a következő alakban írható:

$$k(G_1, G_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=0}^{\infty} \lambda_k [A_\times^k]_{ij}.$$

Ha most $\mu(k) = \lambda_k$, p_1 és p_2 megfelelő méretű oszlopvektorok, melyek minden koordinátája $1/n_1$, illetve $1/n_2$ (ekkor $p_\times$ és $q_\times$ is olyan $n_1 n_2$ magas oszlopvektorok, melyek minden koordinátája $1/(n_1 n_2)$), valamint a ϕ leképezést a fent említett módon úgy választjuk, hogy $W_\times = \tilde{A}_\times$, akkor az ebben a részben definiált kernel a már említett címkesorozatokon alapuló gráfkernel $\frac{1}{n_1^2 n_2^2}$-szerese. Itt is ugyanúgy, mint a címkesorozatokon alapuló gráfkerneleknél, megfelelően választva a $\mu(k)$ együtthatókat, megkapható a geometriai vagy az exponenciális sorösszeg is.

A konkrét megvalósításban, ha az 5.1.-ben definiált kernelt szeretnénk kiszámolni, és feltesszük, hogy mindkét gráf n csúcsú, akkor a $W_\times$ mátrix már $n^2 \times n^2$-es lesz, és az invertálás miatt $O(n^6)$ időbe telik kiszámolni a $k(G_1, G_2)$ értéket (ha nem számolunk azzal, hogy egy $n \times n$-es mátrix invertálására ismert $O(n^c)$ idejű algoritmus, ahol $c < 3$). A [9] cikkben bemutatnak néhány ötletet ennek hatékony kiszámítására. Egyik ötlet az, hogy felhasználhatjuk az $X = SXT + X_0$ alakú egyenlet megoldására szolgáló hatékony algoritmusokat. Itt az S, T, X_0 mátrixok

adott $n \times n$-es valós mátrixok, az X $n \times n$-es valós mátrixot keressük. Sok programcsomag rendelkezik olyan metódussal, amely ezt az egyenletet $O(n^3)$ időben meg tudja oldani. A cikkben levezették, hogy a fenti 5.1.-ben látott kifejezés felírható így:

$$q_\times^T vec(X) = q_\times^T (I - \lambda W_\times)^{-1} p_\times,$$

ahol X egy olyan $n \times n$-es mátrix, mely egy $X = SXT + X_0$ alakú egyenlet megoldása, $vec(x)$ pedig azt jelenti, hogy az X mátrixból elkészítünk egy n^2 magas oszlopvektort úgy, hogy X oszlopait egymásra pakoljuk. Így tulajdonképpen $O(n^6)$ idő helyett $O(n^3)$-ben ki lehet számolni a gráfkernelt.

Egy másik lehetséges megoldás az, hogy olyan programcsomagot használunk, mely az $Mx = b$ egyenletrendszert gyorsan meg tudja oldani, ha M teljesít pár feltételt. Ilyenek az ún. *conjugate gradient* metódusok (CG methods), amelyek gyorsan megoldják a fenti egyenletrendszert, ha az M mátrixnak kevés különböző sajátértéke van, vagy ha az M mátrix ritka. Egy ilyen metódussal először megoldjuk az $(I - \lambda W_\times)x = p_\times$ egyenletrendszert, majd az eredményt balról megszorozzuk a $q_\times^T$ sorvektorral.

Ha nem törekszünk arra, hogy a kernel pontos értékét számoljuk ki, megelégszünk a kernelérték egy approximációjával is, akkor fixpont-iterációs módszerrel közelíthetjük az értéket. Ebben az esetben az $x = p_\times + \lambda W_\times x$ megoldását akarjuk közelíteni. Legyen $x_0 = p_\times$ és $x_{t+1} = p_\times + \lambda W_\times x_t$. Ha λ-t kellően kicsinek választjuk, akkor x_t konvergálni fog a megoldáshoz.

6. A Weisfeiler-Lehman-kernel

Ebben a fejezetben a [10] cikkben bemutatott ötletet fogom ismertetni. Egy olyan gráfkernelről lesz szó, amely az egydimenziós Weisfeiler-Lehman-algoritmuson alapszik. Ez az algoritmus azt próbálja eldönteni, hogy két adott gráf, G_1 és G_2, izomorfak-e. A gráfizomorfia eldöntése általános gráfok esetén nem egyszerű a feladat. A probléma NP-ben van, de nem bizonyított, hogy NP-teljes lenne és polinomiális algoritmus sem ismert a megoldására [10]. Az egydimenziós Weisfeiler-Lehman-algoritmus két gráf izomorfiáját a gráfok csúcsainak kiszínezésével (megcímkézéséevel) szeretné eldönteni. Ez több iterációban történik, minden iterációban frissítjük az összes csúcs címkéjét, viszont előfordulhat, hogy egy csúcs címkéje egy iteráció után nem változik meg. Ha a két gráfban előforduló címkék halmaza nem egyezik meg valamelyik iteráció után, akkor az algoritmus megáll. Ekkor ugyanis biztos az, hogy a két gráf nem izomorf. Ha ez nem történik meg, akkor az algoritmust egy előre meghatározott számú iterációig futtatjuk (ez gyakran n, ahol n a gráf csúcsainak száma - feltehető, hogy a két gráf csúcsainak száma azonos, hiszen ha ez nem teljesül, akkor biztosan nem izomorf a két gráf), vagy pedig az iterációkat addig ismételjük, amíg a csúcsok partíciója nem változik (két csúcs akkor kerül egy osztályba, ha ugyanaz a címkéjük). Ha az algoritmus végén a két gráfban előforduló két címkehalmaz nem azonos, akkor nem izomorfak a gráfok. Ha ezek a halmazok megegyeznek, akkor vagy izomorf a két gráf, vagy pedig az egydimenziós Weisfeiler-Lehman-algoritmus nem tudta megkülönböztetni őket. A [10] szerint legtöbb esetben ez az algoritmus jó teszt az izomorfia eldöntésére. Vannak azonban gráfok, ahol ezzel az algoritmussal nem lehet eldönteni két gráfról, hogy izomorfak-e.

Ebben a részben tegyük fel, hogy irányítatlanok a gráfjaink és van egy l függvényünk, amely a gráf minden csúcsához egy véges halmazból hozzárendel egy címkét. Feltehető, hogy ez a halmaz egymást követő természetes számokból áll. Ha címkézetlen gráfokkal szeretnénk dolgozni, akkor ez az l függvény rendelje hozzá minden csúcshoz az $'1'$-est. A gráfot gyakran a (V, E, l) hármassal jelöljük, ahol V és E a gráf csúcs- és élhalmaza.

6.1. A Weisfeiler-Lehman-algoritmus egy iterációja

Kezdetben mindkét gráf csúcsaihoz hozzárendeljük azt a címkét, amelyet az l függvény hozzárendel. Egy iterációban az történik, hogy minden csúcshoz hozzárendelünk egy multihalmazt. Ez a multihalmaz az adott csúcs szomszédainak címkéit tartalmazza. Ezután minden multihalmazban növekvő sorrendben rendezzük az elemeket, majd ezt egy stringgé fűzzük össze. Ezután minden csúcs esetén a csúcshoz tartozó multihalmazból képzett string elé beszúrjuk az adott csúcs címkéjét. Most szükségünk van egy f függvényre, amely minden csúcs esetén tömöríti a csúcshoz tartozó stringet és elkészíti a csúcsok új címkéit. Ennek az f függvénynek injektívnek kell lennie, vagyis két csúcs csak akkor kapjon megegyező új címkét, ha a hozzájuk tartozó stringek is megegyeztek. Ilyen f-et könnyű konstruálni: először a csúcsokhoz tartozó multihalmazokból képezett stringeket rendezzük lexikografikus sorrendben. Legyen egy számlálónk, amelyben azt számoljuk, hogy f hány különböző stringet tömörített már. Ha egy iterációban f-nek egy olyan stringet kell tömörítenie, amilyet már korábban tömörített, akkor ezt a stringet f tömörítse úgy, hogy a számláló aktuális értékét rendeli hozzá. Ha egy olyan string jön, amelyet korábban még nem tömörítettünk, akkor a számlálót növeljük meg 1-gyel és ez az érték legyen a string tömörítése. Mivel a stringek rendezve vannak, így az azonos stringek egymás után jönnek, így az f ugyanazt az értéket rendeli hozzájuk, viszont két különböző string nem kapja ugyanazt a tömörítést, hiszen a sorrendben később következőhöz egy nagyobb számot rendel f. Ha f-fel minden stringet tömörítettünk, akkor a csúcsok új címkéi legyenek ezek a tömörítések.

Algorithm 1 One iteration of the 1-dim. Weisfeiler-Lehman test of graph isomorphism

1: Multiset-label determination
- For $i = 0$, set $M_i(v) := l_0(v) = \ell(v)$. [2]
- For $i > 0$, assign a multiset-label $M_i(v)$ to each node v in G and G' which consists of the multiset $\{l_{i-1}(u) | u \in \mathcal{N}(v)\}$.

2: Sorting each multiset
- Sort elements in $M_i(v)$ in ascending order and concatenate them into a string $s_i(v)$.
- Add $l_{i-1}(v)$ as a prefix to $s_i(v)$ and call the resulting string $s_i(v)$.

3: Label compression
- Sort all of the strings $s_i(v)$ for all v from G and G' in ascending order.
- Map each string $s_i(v)$ to a new compressed label, using a function $f : \Sigma^* \to \Sigma$ such that $f(s_i(v)) = f(s_i(w))$ if and only if $s_i(v) = s_i(w)$.

4: Relabeling
- Set $l_i(v) := f(s_i(v))$ for all nodes in G and G'.

A Weisfeiler-Lehman-algoritmus egy iterációja [10].

Jelölje l_i azt a csúcscímkéző függvényt, amely egy gráf csúcsaihoz hozzárendeli azokat a címkéket, amelyekkel a Weisfeiler-Lehman-algoritmus i-edik iterációja után rendelkeznek. l_0 legyen a kezdetben is adott l függvény.

6.1. Definíció. *Egy $G = (V, E, l)$ gráf i mélységű(magasságú) Weisfeiler-Lehman-gráfjának nevezzük a $G_i = (V, E, l_i)$ gráfot. Egy G gráf h-mélységű Weisfeiler-Lehman-gráfsorozatának nevezzük a $G_0, G_1, ...G_h$ sorozatot, ahol $G_0 = G$.*

6.2. Definíció. *Legyen k tetszőleges gráfkernel (ez lesz az alapkernel). A h iterációs Weisfeiler-Lehman-kernelt a következőképpen definiáljuk:*

$$k_{WL}^{(h)}(G, G') = k(G_0, G_0') + k(G_1, G_1') + ... + k(G_h, G_h'),$$

ahol $G_0, G_1, ...G_h$ és $G_0', G_1', ...G_h'$ a G-hez és G'-höz tartozó Weisfeiler-Lehman-gráfsorozatok.

Ha a k alapkernel pozitív szemidefinit, akkor tetszőleges h-ra a $k_{WL}^{(h)}$ gráfkernel pozitív szemidefinit. Ugyanis ha r jelöli azt a függvényt, amelyre $G_{i+1} = r(G_i)$, vagyis r jelöli az algoritmus egy iterációjában történő csúcsátcímkézést, és ϕ jelöli a k kernelhez tartozó feature-leképezést, akkor $k(G_i, G_i^{'}) = \langle \phi(G_i), \phi(G_i^{'}) \rangle = \langle \phi(r^i(G)), \phi(r^i(G^{'})) \rangle$. Ha $\phi^{'} = \phi \circ r^i$, akkor $k(G_i, G_i^{'}) = \langle \phi^{'}(G), \phi^{'}(G^{'}) \rangle$, vagyis $k(G_i, G_i^{'})$ pozitív szemidefinit. Mivel $k_{WL}^{(h)}(G, G^{'})$ ilyenek összegéből adódik, ezért $k_{WL}^{(h)}$ is pozitív szemidefinit. Érdemes megjegyezni, hogy $k_{WL}^{(h)}$ definíciójában $k(G_j, G_j^{'})$-k tetszőleges nemnegatív lineáris kombinációit is nézhetnénk. Így kicsit általánosabb pozitív szemidefinit kernelt kapnánk.

Jelölje Σ_i a Weisfeiler-Lehman-algoritmus i-edik iterációja után G-ben és $G^{'}$-ben előforduló csúcscímkék halmazát, Σ_0 jelölje a kezdetben előforduló címkékét. Feltehető, hogy ezek a Σ_i halmazok páronként diszjunktak. Feltehető még, hogy a Σ_i halmazok rendezve vannak, vagyis $\Sigma_i = \{\sigma_{i1}, \sigma_{i2}, ...\sigma_{i|\Sigma_i|}\}$. Legyen $c_i : \{G, G^{'}\} \times \Sigma_i \to \mathbb{N}$ az a függvény, melyre $c_i(G, \sigma_{ij})$ a σ_{ij} címke előfordulásainak a száma a G gráfban. Definiálható egy kernel, amely megszámolja G és $G^{'}$ esetén a gráfokban előforduló közös címkéket, miután h iterációt elvégeztünk a Weisfeiler-Lehman-algoritmussal. A következő kernel pontosan ezt teszi:

6.3. Definíció (Számláló Weisfeiler-Lehman-kernel)**.** *A korábbi jelöléseket használva a G és $G^{'}$ gráfok esetén a gráfkernel*

$$k_{WLcount}^{(h)}(G, G^{'}) = \langle \phi_{WLcount}^{(h)}(G), \phi_{WLcount}^{(h)}(G^{'}) \rangle,$$

ahol

$$\phi_{WLcount}^{(h)}(G) = (c_0(G, \sigma_{01}), ..., c_0(G, \sigma_{0|\Sigma_0|}), ..., c_h(G, \sigma_{h1}), ..., c_h(G, \sigma_{h|\Sigma_h|}))$$

és

$$\phi_{WLcount}^{(h)}(G^{'}) = (c_0(G^{'}, \sigma_{01}), ..., c_0(G^{'}, \sigma_{0|\Sigma_0|}), ..., c_h(G^{'}, \sigma_{h1}), ..., c_h(G^{'}, \sigma_{h|\Sigma_h|})).$$

7. A Tanimoto- és a minmax-kernel

Ebben a fejezetben a [11] cikkben ismertetett kernelekről lesz szó.

Irányítatlan gráfokkal foglalkozzunk csak, és tegyük fel, hogy a csúcsok és az élek címkézve vannak. Az $A = \{l_1^a, l_2^a, ...l_p^a\}$ halmaz jelölje a csúcscímkék halmazát, a $B = \{l_1^b, l_2^b, ...l_q^b\}$ halmaz pedig az élcímkék halmazát. Jelölje $P(d)$ az összes olyan címkesorozatot, amely olyan utat kódolhat, amely legfeljebb d élű, vagyis $P(d)$ elemei olyan legfeljebb $2d + 1$ hosszú sorozatok, melyek hossza páratlan, minden páratlanadik tagja A-beli elem, minden párosadik tagja B-beli.

Legyen d rögzített. Legyen G gráf, $p \in P(d)$ esetén legyen $\phi_p(G)$ az a függvény, amely 1-et vesz fel, ha van G-ben legalább egy olyan legfeljebb d élű út, amely címkézése a p címkesorozatnak felel meg, egyébként 0-t. A ϕ leképezés rendeljen minden G gráfhoz egy vektort a következőképpen:

$$\phi(G) = (\phi_p(G))_{p \in P(d)}.$$

Hasonlóképpen definiálható egy ϕ leképezés is, amely megszámolja, hogy $P(d)$ egyes elemeit hányszor kapjuk meg G-beli címkézett utakként. A $\tilde{\phi_p}(G)$ függvény legyen az a függvény, amely egy G gráf esetén k-t vesz fel, ha pontosan k különböző, legfeljebb d élű út van G-ben, amely megcímkézése a $p \in P(d)$ címkesorozatot állítja elő. A $\tilde{\phi}$ leképezés a következő vektort rendeli hozzá egy G gráfhoz:

$$\tilde{\phi}(G) = (\tilde{\phi_p}(G))_{p \in P(d)}.$$

Ezeket a leképezéseket felhasználva megadható két gráfkernel a következőképpen:

$$k(G, G') = \langle \phi(G), \phi(G') \rangle = \sum_{p \in P(d)} \phi_p(G)\phi_p(G'),$$

$$\tilde{k}(G, G') = \langle \tilde{\phi}(G), \tilde{\phi}(G') \rangle = \sum_{p \in P(d)} \tilde{\phi}_p(G)\tilde{\phi}_p(G').$$

Könnyen látható, hogy ezek pozitív szemidefinit kernelek. A k és $\tilde{k}$ kernelek segítségével definiálható még a következő két kernel:

7.1. Definíció (Tanimoto-kernel)**.** *: A fenti k kernel segítségével a G és G' gráfok esetén a Tanimoto-kernel értéke a következő:*
$k_{tanimoto}(G, G') = \frac{k(G,G')}{k(G,G)+k(G',G')-k(G,G')}$.

A Tanimoto-kernel számlálójában azon $P(d)$-beli elemek száma áll, amelyek G-ben és $G^{'}$-ben is előfordulnak legfeljebb d hosszú utak címkesorozataként, a nevezőben pedig azon elemek száma áll, amelyek G és $G^{'}$ közül legalább az egyikben előfordulnak.

7.2. Definíció (Minmax-kernel)**.** *A fenti $\tilde{\phi}_p$ leképezés segítségével a G és $G^{'}$ gráfok esetén a minmax-kernel értéke a következő:* $k_{minmax}(G, G^{'}) = \frac{\sum_{p \in P(d)} min(\tilde{\phi}_p(G), \tilde{\phi}_p(G^{'}))}{\sum_{p \in P(d)} max(\tilde{\phi}_p(G), \tilde{\phi}_p(G^{'}))}$.

A [11] cikk szerint a Tanimoto-kernel és a minmax-kernel pozitív szemidefinit kernelek.

8. Legrövidebb utak kernele

Ebben a részben a [12] cikkben bemutatott ötletet fogom ismertetni. Most is irányítatlan és összefüggő gráfokat tekintsünk. A gráfok élei rendelkezhetnek súlyokkal, azonban ezek a súlyok most valamilyen távolságot fejezzenek ki, ezért legyenek pozitívak. Ha a gráfokban nincsenek élsúlyok, akkor minden él súlya legyen 1. A gráfok csúcsai rendelkezhetnek valamilyen véges halmazból származó címkékkel.

Legyen adott két gráf, G_1 és G_2. Jelölje $P(G_1)$ a G_1 gráfban előforduló olyan séták halmazát, amely esetén egy sétában nem ismétlődnek élek, $P(G_2)$ jelölje ugyanezt a G_2 gráf esetén. Tegyük fel, hogy már definiáltunk egy csúcskernelt és egy élkernelt, melyek két csúcshoz, illetve két élhez rendelnek egy értéket. Legyen k_p egy kétváltozós pozitív szemidefinit függvény a $P(G_1)$ és $P(G_2)$ elemein, melyre $p_1 \in P(G_1)$ és $p_2 \in P(G_2)$ esetén a $k_p(p_1, p_2)$ értéket a p_1-ben és p_2-ben előforduló csúcsokon és éleken kiszámolt csúcskernel, illetve élkernel értékeinek szorzataként definiáljuk. Ezután definiálható a következő gráfkernel:

$$k(G_1, G_2) = \sum_{p_1 \in P(G_1)} \sum_{p_2 \in P(G_2)} k_p(p_1, p_2),$$

vagyis a G_1 és G_2 gráfokban előforduló, citeélismétlődést nem tartalmazó sétákra páronként ki kell számítani a k_p értékét, majd ezt összegezni.
Be lehet bizonyítani, hogy ez a kernel pozitív szemidefinit. A bizonyítás ötlete az, hogy a fenti k kernel felírható úgy is, mint egy úgynevezett pozitív szemidefinit R-konvolúciós kernel [13]. A dolgozatban az R-konvolúciós kernelekről nem lesz szó, ezért a bizonyítás részleteit sem írnám le most.
Bár a fent definiált kernel pozitív szemidefinit, a kiszámításához meg kell határozni az összes utat G_1-ben és G_2-ben, az pedig ismert, hogy egy NP-nehéz probléma. Emiatt érdemes lehet a fenti kerneldefiníciót módosítani úgy, hogy ne az összes élismétlődést nem tartalmazó sétára menjen az összegzés. Mivel feltettük, hogy pozitívak az élsúlyok, a Floyd-Warshall-algoritmussal polinom időben ($O(n^3)$ időben, ahol n az adott gráf csúcsainak száma) meghatározható egy gráfban az összes csúcspár között menő legrövidebb út hossza. Így felvetődhet az az ötlet, hogy módosítsuk a fenti k kerneldefiníciót úgy, hogy csak a legrövidebb utakra menjen az összegzés. Ehhez először érdemes kiszámítani minden G gráfhoz a legrövidebb utak

gráfját, mely egy teljes gráf, csúcsszáma megegyezik G csúcsszámával és minden él súlyozva van, mégpedig egy uv él súlya a G gráfban lévő legrövidebb uv út hossza.

8.1. Definíció (Legrövidebb utak kernele)**.** *Legyen G_1 és G_2 két gráf, $S_1 = (V_1, E_1)$ és $S_2 = (V_2, S_2)$ pedig ezen gráfokból képezett legrövidebb utak gráfjai. Ekkor a legrövidebb utak kernele G_1 és G_2 esetén a következő értéket veszi fel:*

$$k_{\text{legrövidebb utak}}(G_1, G_2) = \sum_{e1 \in E_1} \sum_{e_2 \in E_2} k_e(e_1, e_2),$$

ahol k_e egy pozitív szemidefinit kernel az éleken.

A fenti $k_{\text{legrövidebb utak}}$ kernel pozitív szemidefinit. A bizonyítás ötlete ebben az esetben is az, hogy erre a kernelre tekinthetünk úgy, mint egy R-konvolúciós kernelre.

Az alkalmazásokban gyakran definiálják a k_e kernelt a következőképpen: ha e_1 és e_2 hossza nem egyezik meg, akkor $k_e(e_1, e_2)$ értéke legyen 0. Ha a két hosszúság megegyezik, és a gráfok csúcsai nincsenek címkézve, akkor a $k_e(e_1, e_2)$ érték legyen 1. Ha a csúcsok rendelkeznek címkékkel, akkor a $k_e(e_1, e_2)$ érték akkor legyen 1, ha e_1 két végpontjának címkéi megegyeznek e_2 két végpontjának címkéivel, különben ez érték legyen 0. Vagyis ekkor címkézetlen esetben az ugyanolyan hosszú legrövidebb útpárok számát határozzuk meg G_1-ben és G_2-ben, címkézett esetben pedig az ugyanolyan típusú csúcspárok között menő, azonos hosszúságú legrövidebb útpárok számát.

9. Eredmények

A bemutatott kerneleket a gyakorlatban is kipróbáltam az agygráfokon. A 83 és 129 csúcsú felbontású adatokkal dolgoztam. Számoltam kerneleket az egész gráfokra és redukált gráfokra is. A redukálás úgy történt, hogy négy agyi lebeny közül (homloki lebeny - frontalis; falcsonti lebeny - parietalis; halántéki lebeny - temporalis; nyakszirti lebeny - occipitalis) az összes lehetséges módon kiválasztottam kettőt, és az agygráfok azon éleit tartottam meg, amelyek mindkét végpontja a kiválasztott két lebeny egyike. A 129-es felbontásban olyan redukálást is elvégeztem, amelyekben csak egy lebeny csúcsait és a lebenyen belül haladó éleket hagytam meg.

A kernelizált SVM-et kétosztályú klasszifikációra használtam, az alanyok neme szerint osztályoztam az agygráfokat. A kerneleket 240 gráfra számoltam ki, ezekből 120 férfi és 120 női agygráf. A kiszámolt kernelek közül egy kernel (a véletlen sétákon alapuló gráfkernel) figyelembe veszi az élsúlyokat, ha vannak. Ennél a kernelnél a gráfok élattribútumai közül a *number of fibers* nevű súlyt, vagyis az idegrostok számát használtam. A csúcscímkéket figyelembe vevő kernelek esetében az agygráfok azonos agyterületet jelölő csúcsainak címkéi megegyeztek, a különböző agyterületet jelölő csúcsok címkéi is különböztek.

Minden esetben két részre osztottam az adatokat: tanítóadatokra és tesztadatokra. A tanítóadatokon betanítottam egy kernelizált SVM modellt, majd a tesztadatokon megnéztem a predikciós képességét, vagyis megnéztem, hogy a tesztadatok valódi osztályai és a betanított modell által prediktált osztályok hány százalékban egyeznek meg. Minden esetben 10 teszt pontosságának átlaga adta a végső predikciós pontosságot. Mindig 60 adatot használtam tesztelésre (30 női, 30 férfi), többi adatot tanításra használtam. Minden esetben a 60 tesztadatot véletlenszerűen választottam ki az összes adatból. A kernelizált SVM-nek van egy C hiperparamétere. Ezt a hiperparamétert mind a 10 esetben a tanítóadatok segítségével, 10-es cross validation használatával határoztam meg.

A kernelizált SVM-en kívül megnéztem az egyszerű lineáris SVM predikciós képességét is. Itt minden agygráfot egy d-dimenziós vektorral reprezentáltam, ahol d a 240 agygráfban előforduló különböző élek száma. Két esetet vizsgáltam: súlyozott és súlyozatlan esetet. A súlyozott esetben a d-dimenziós vektorok koordinátái a

megfelelő élek *number of fibers* értékei, a súlyozatlan esetben minden koordináta 0 vagy 1, annak megfelelően, hogy az adott él jelen van-e a vektor által reprezentált agygráfban.

A kernelizált és lineáris SVM-hez a Python Scikit-learn programcsomagját és a [14]-ben feltüntetett Github repository-t használtam.

9.1. A kernelizált és lineáris SVM összehasonlítása

A következő négy táblázat a lineáris SVM és a kernelizált SVM által elért predikciós pontosságokat tartalmazza a két felbontásban. A *Data* nevű oszlop a két lebeny latin nevének kezdőbetűit tartalmazza (homloki lebeny - frontalis (F); falcsonti lebeny - parietalis (P); halántéki lebeny - temporalis (T); nyakszirti lebeny - occipitalis (O)). Az *all nodes* nevű mező azt jelöli, hogy az agygráfokat nem redukáltam. A kernelizált SVM eredményeit tartalmazó táblázat érték nélküli mezői arra utalnak, hogy az adott kernel kiszámítási ideje nagyon hosszú volt az adott esetben, vagy pedig a kernelizált SVM módszer nem kezdett konvergálni 15 percen belül. A táblázatban található kernelek: *random walk* - véletlen sétákon alapuló gráfkernel; *WL* - Weisfeiler-Lehman-kernel; *minmax* - minmax-kernel; *sp* - legrövidebb utak kernele; *lab.seq* - címkesorozatokon alapuló gráfkernel; *tanimoto* - Tanimoto-kernel.

9.1.1. 83-as felbontás

A következő két táblázatból kiderül, hogy ennél a felbontásnál az élsúlyokat használó lineáris SVM és az élsúlyokat nem használó kernelek közül a kiválasztott lebenyektől függően hol az egyik módszer, hol a másik módszer teljesít jobban, viszont néhány esetben ezek a módszerek sokkal jobbak az élsúlyokat nem használó lineáris SVM-nél és az élsúlyokat használó, véletlen sétákon alapuló kernelnél.

Data	Number of features	Weighted acc.	Unweighted acc.
FT	312	0.733	0.583
FP	315	0.781	0.653
FO	221	0.726	0.628
OP	141	0.691	0.618
TO	179	0.730	0.630
TP	224	0.753	0.585
all nodes	1749	0.802	0.747

1. táblázat. Lineáris SVM a 83-as felbontású agygráfokon

Data	random walk	WL	minmax	sp	lab.seq	tanimoto
FT	0.526	0.721	0.693		0.743	0.678
FP	0.598	0.720	0.708		0.728	0.72
FO	0.53	0.738	0.738		0.763	0.748
OP	0.651	0.651	0.658		0.672	0.665
TO	0.588	0.643	0.628		0.658	0.617
TP	0.623	0.648	0.63		0.678	0.631
all nodes		0.751		0.748		

2. táblázat. Kernelizált SVM a 83-as felbontású agygráfokon

9.1.2. 129-es felbontás

Ebben a felbontásban az élsúlyokat használó kernelizált SVM szerepelt a leggyengébben, a többi három közül pedig a lebenyek választásától függ, hogy melyik módszer pontosabb.

Data	Number of features	Weighted acc.	Unweighted acc.
F	446	0.748	0.692
T	199	0.705	0.731
O	57	0.610	0.521
P	251	0.726	0.650
FT	737	0.758	0.733
FO	579	0.780	0.710
FP	1050	0.821	0.740
TO	432	0.746	0.745
TP	679	0.748	0.736
OP	497	0.773	0.673
all nodes	3622	0.822	0.763

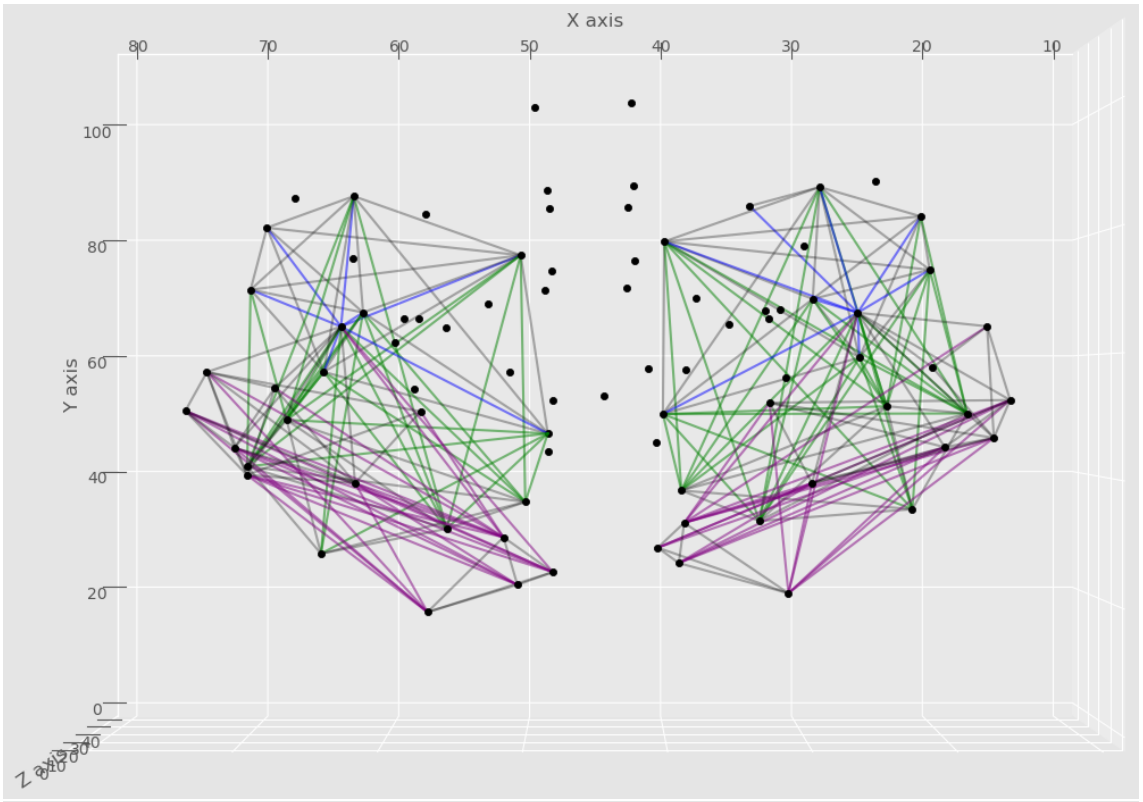
3. táblázat. Lineáris SVM a 129-es felbontású agygráfokon

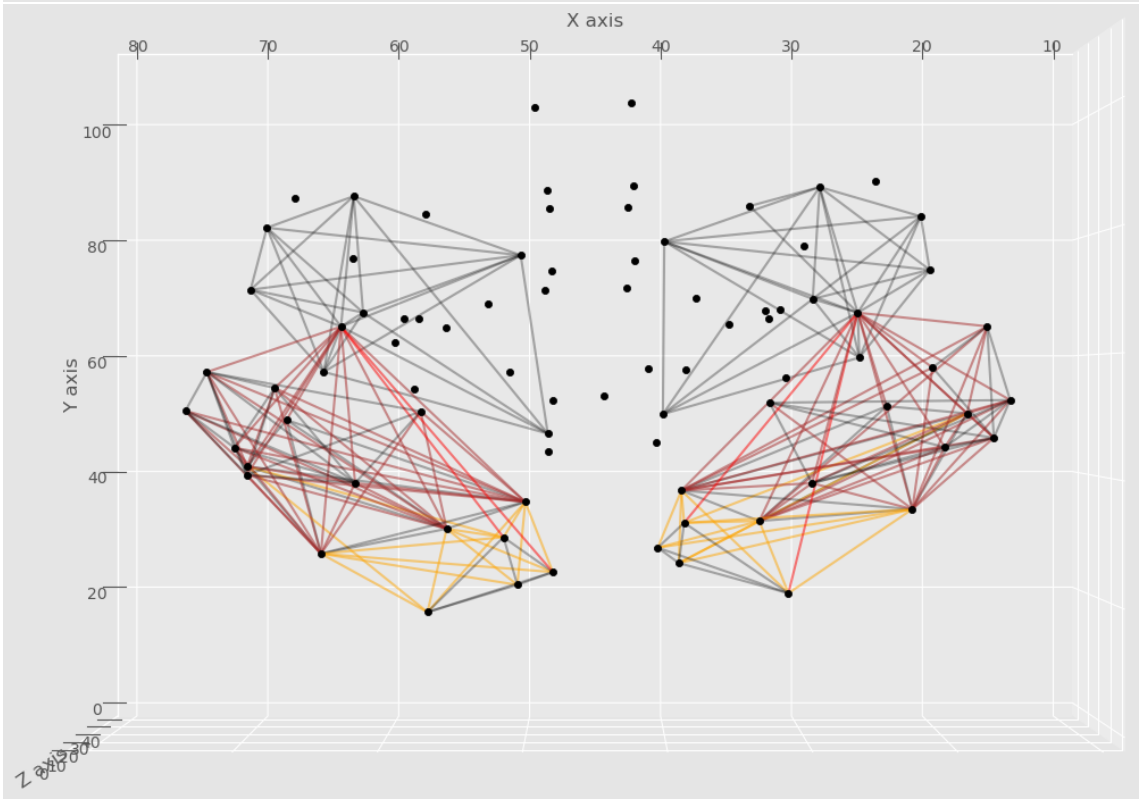
Data	random walk	WL	minmax	sp	lab.seq
F	0.545	0.723	0.728	0.659	0.741
T	0.586	0.741	0.721	0.679	0.713
O	0.546		0.497		
P	0.573	0.751	0.69	0.616	0.697
FT		0.753	0.755	0.695	0.755
FO	0.65	0.751	0.751	0.723	0.74
FP		0.667	0.736	0.678	0.66
TO	0.627	0.741	0.763	0.77	0.75
TP	0.655	0.738	0.726	0.681	0.717
OP	0.667	0.673	0.696	0.613	0.685
all nodes		0.713		0.782	

4. táblázat. Kernelizált SVM a 129-es felbontású agygráfokon

9.1.3. Ábrák a 83-as felbontásról

A következő két ábra egy alany redukált agygráfját ábrázolja, különböző színekkel jelölve a lebenypárok által feszített részgráfokat. Mindkét ábrán szürkével vannak jelölve azok az élek, amelyek az egyes lebenyeken belül haladnak, a színes élek pedig olyan élek, amelyek végpontjai különböző lebenyekben vannak. A színek jelentése a következő: kék - FT; zöld - FP; piros - FO; barna - TP; lila - TO; sárga - OP.


X axis
80
70
60
50
40
30
20
10
Y axis
100
80
60
40
20
0


X axis
80
70
60
50
40
30
20
10
Y axis
100
80
60
40
20
0

10. Összefoglalás

A dolgozat végén agygráfok nem szerinti klasszifikációját végeztem el, erre csupán a gráfok struktúráját használtam. A gépi tanulás egy eszközével, a kernelizált SVM-mel mértem meg azt, hogy tanítóadatok segítségével mekkora pontossággal lehet a korábban még nem látott agygráfok nemét prediktálni. A következő gráfkernelekre történt a kiértékelés: címkesorozatokon alapuló gráfkernel, véletlen sétákon alapuló kernel, Weisfeiler-Lehman-kernel, Tanimoto- és minmax-kernel, illetve a legrövidebb utak kernele. A 83 csúcsú és a 129 csúcsú agygráfokon, illetve az ezekből készített redukált gráfokon végeztem el a kísérleteket, és a kapott eredményeket összehasonlítottam az egyszerű lineáris SVM-mel elért eredményekkel. Az előző oldalakon látott táblázatokból kiolvasható, hogy a 83-as felbontás esetén a legjobb predikciót az élsúlyokat használó lineáris SVM adta, ez körülbelül 80%-os pontosságot jelent, viszont vannak lebenypárok, amelyek kernelizált SVM-mel vagy lineáris SVM-mel 72%-os pontosságnál jobb eredményt értek el, vagyis már a lebenypárok szintjén felfedezhető különbség a két nem agygráfjainak struktúrájában. A nagyobb, 129-es felbontásnál több lebeny vagy lebenypár pontosabban tudja prediktálni a biológiai nemet, mint a kisebb felbontásnál. Egyik esetben egy kicsivel jobbnak bizonyul a súlyozott lineáris SVM, másik esetben a kernelizált SVM.

Az élsúlyokat figyelembe nem vevő gráfkernelek eredményei minden esetben jobbak, mint az élsúlyokat figyelembe vevő gráfkernelé, ezért a kernelizált SVM-nek egy előnye az, hogy elég csupán a gráfokban az egyes élek meglétét vagy nemlétét ismerni, az élsúlyokkal nem kell foglalkozni. A nagyobb felbontásban az élsúlyokat nem használó kernelizált SVM jobb eredményeket ér el (néha akár 10%-kal is jobbat), mint az élsúlyokat nem használó lineáris SVM.

Az élsúlyokat nem használó kernelizált SVM és az élsúlyokat használó lineáris SVM pontosságai között nincs igazán különbség. Alkalmazásbeli különbség azonban van: a kernelizált SVM alkalmazása jóval bonyolultabb, mint egy (súlyozott) lineáris SVM-é, ugyanis a gráfok páronkénti kernelértékeinek kiszámítása sok esetben igen sok időt vagy memóriát vesz igénybe, illetve az újonnan érkező adatokhoz is sok kernelértéket kell kiszámolni, hogy el tudjuk végezni rajtuk a klasszifikációt. Ezzel

szemben az egyszerű lineáris SVM-hez képest átlagosan nem ér el jobb eredményeket nem szerinti osztályozásban. A lineáris SVM egy óriási előnye, hogy könnyen és gyorsan alkalmazható, elég az optimális hipersík normálvektorának koordinátáit és a b értéket ismerni ahhoz, hogy az új adatokra is prediktálni tudjuk a biológiai nemet.

A dolgozat rávilágított arra, hogy az agy bármely két lebenye között van eltérés a férfi és a női agygráfok között. A súlyozott lineáris SVM eredményeiből következtethetünk arra, hogy ez a lebenypáronkénti eltérés jobban megnyilvánul az idegrostok számának eltérésében, mint az élek elhelyezkedésének eltérésében. A kernelizált SVM (nagyobb felbontásban elért) eredményei pedig arra utalhatnak, hogy sok esetben a lebenyek vagy lebenypárok izomorfiája, a csúcspárok között haladó súlyozatlan utak vagy séták szerkezetei is eltérnek a két nem agygráfjai között.

Hivatkozások

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *arXiv e-prints*, page arXiv:1901.00596, 2019.

[2] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[3] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning Distributed Representations of Graphs. *arXiv e-prints*, page arXiv:1707.05005, 2017.

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

[5] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2001.

[6] Balázs Szalkai, Bálint Varga, and Vince Grolmusz. Graph theoretical analysis reveals: Women's brains are better connected than men's. *PLOS ONE*, 10(7):e0130045, July 2015.

[7] Balázs Szalkai, Bálint Varga, and Vince Grolmusz. Brain size bias compensated graph-theoretical parameters are also better in women's structural connectomes. *Brain Imaging and Behavior*, 12(3):663–673, April 2017.

[8] Peter Flach Thomas Gärtner and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. *Learning Theory and Kernel Machines*, pages 129–143, 2003.

[9] Schraudolph N.N. Kondor R. Borgwardt K.M. Vishwanathan, S.V.N. Graph kernels. *Journal of Machine Learning Research 11*, page 1201–1242, 2010.

[10] Schweitzer P. Leeuwen E.J.v. Mehlhorn K. Borgwardt K.M. Shervashidze, N. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research 12*, page 2539–2561, 2011.

[11] Hiroto Saigo Pierre Baldi Liva Ralaivola, Sanjay J. Swamidass. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

[12] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. *In Proceedings of the International Conference on Data Mining*, pages 74–81, 2005.

[13] David Haussler. Convolution kernels on discrete structures. Technical report, 1999.

[14] Linlin Jia, Benoit Gaüzère, and Paul Honeine. graphkit-learn. https://github.com/jajupmochi/graphkit-learn/tree/master.