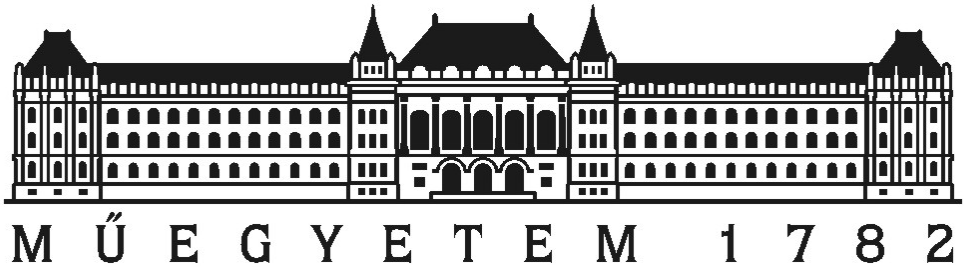


GPU-ra implementált Multiple Kernel Learning és genomikai alkalmazásai

Bolgár Bence, SE-ÁOK VI.

Konzulens:

Dr. Antal Péter, egyetemi docens, BME MIT

Budapesti Műszaki- és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

Budapest, 2011.

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

Szimbólumok

$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$	Lagrange-multiplikátorok vektorai.
b, ρ	A szeparáló hipersík egyenletéhez tartozó bias tag.
C	Cost (költség) paraméter az SVM soft-margin formalizációjában, amely a komplexitás és az általánosító képesség között egyensúlyoz.
$\boldsymbol{d}$	A kernel súlyok vektora.
Δ	Lépés a duál célfüggvény optimalizálása során.
K_k	A k-adik kernel mátrix.
λ	MKL paraméter, amely a kernel súlyok nagyságát korlátozza az Lp-regularizáció kényszerítésével.
ν	Az egyosztályos SVM paramétere, amely a szupportvektorok arányát (komplexitást) szabályozza.
p	Az Lp regularizáció paramétere.
ϕ_k	A k-adik kernel függvény.
Q_k	A k-adik címkézett kernel mátrix.
$\boldsymbol{w}$	A szeparáló hipersík normálvektora.
$\boldsymbol{x_i}$	Adatvektor.
$\boldsymbol{\xi}$	Slack változók vektora az SVM soft-margin formalizációjában.

Rövidítések

AMD Advanced Micro Devices Inc..

CNN Cellular Nonlinear Network.

DMA Direct Memory Access, CPU-tól független memóriahozzáférés.

GPGPU General Purpose GPU (általános célú GPU programozás).

GPU Grafikus processzor.

JIT Just-in-time compilation, itt: a host alkalmazás futás közben fordítja a binárisokat.

LRU Least Recently Used (gyorsítótárazási stratégia).

MACCS Kémiai leíró (fingerprint), amely bizonyos kémiai jegyek meglétét/hiányát kódolja.

MedDRA Medical Dictionary for Regulatory Activities, ontológia.

MKL Multple Kernel Learning.

PCI Peripherial Component Interconnect, a CPU és az eszközök közti kommunikáció útja.

QP Quadratic Programming, matematikai optimalizálási feladat.

RBF Radiális bázisfüggvény.

SIMD Single Instruction, Multiple Data [1].

SMO Sequential Minimal Optimization [2].

SSE Streaming SIMD Extensions, CPU utasításkészlet-bővítmény (vektorműveletek).

SVM Support Vector Machine, gépi tanulási algoritmus.

UCI University of California, Irvine.

UMLS Unified Medical Language System, ontológia.

VLIW Very Long Instruction Word, utasításszintű párhuzamosságot kihasználó architektúra [1].

WSS Working Set Selection [3].

1. Bevezetés

E munka egy bioinformatikai kutatási vonalat mutat be, amelynek célja a genomikai kísérlettervezés, adatelemzés és értelmezés segítése. A dolgozat az alábbi fejezetekre oszlik:

- 1. fejezet, ahol röviden áttekintjük a korszerű biológiai kutatások során felmerülő problémákat és összefoglaljuk az eddigi módszereket.
- 2. fejezet, ahol a kutatási probléma és a fejlesztett rendszer matematikai hátterének részleteit és kiterjesztését ismertetjük.
- 3. fejezet, ahol az olvasó megismerkedhet a grafikus kártyán történő implementáció részleteivel.
- 4. fejezet, ahol valós bioinformatikai feladatokra alkalmazzuk az eszközt.

A dolgozatban összefoglalt munka főbb eredményei az alábbiak:

- az SMO-MKL megközelítés eredeti levezetése alapján kidolgoztunk egy CPU-ra és GPU-ra implementálható pszeudokódot és formulákat (lásd 2.2 alfejezet),
- az SMO-MKL megközelítést a prioritizálási feladatokban fontos egyosztályos feladatra is átfogalmaztuk, az implementációhoz szükséges formulákat ezen kiterjesztés esetében is származtattuk (lásd 2.4 alfejezet),
- elkészítettünk egy GPU alapú implementációt, amely egy általános referencia megoldó rendszerhez képest két nagyságrend, az eredeti SMO-MKL megközelítéshez képest egy nagyságrend gyorsulást eredményez (lásd 3. fejezet),
- elkészítettünk egy CPU alapú implementációt is, amely grid alapú futtatásokra is alkalmas (lásd 3.6 alfejezet),
- több valós bioinformatikai – génprioritizálási és gyógyszerkutatási – feladatban is alkalmaztuk, amelyek eredményeinek publikálása folyamatban van.

1.1. A genomika és a bioinformatika

Ahhoz, hogy megértsük a jelen munka létjogosultságát, célját és helyét a modern biológiai kutatásokban, először a genomika tudományáról kell szólnunk néhány szót. Genomnak nevezzük egy élőlény DNS-állományát, amely – legalábbis jelenlegi tudásunk szerint – az örökölhető információ legnagyobb részét hordozza; a genomika pedig a biológia rohamosan fejlődő új területe, amely a genomot egészében, nem csak az egyes gének szintjén vizsgálja. Nem lehet olyan természettudományos diszciplínát említeni, ahonnan ne kölcsönözne eszközöket és módszereket,

ideértve a fizikát, matematikát, informatikát, kémiát, stb. A tudományág történetének ismertetésétől most eltekintünk (említhetnénk például a Human Genome Projectet), helyette a jelenre összpontosítva az utóbbi egy-két évtizedben felmerülő problémákat ismertetjük.

A XX. század utolsó negyedében számos biológiai eljárás látott napvilágot, amelyek mára hatalmas mennyiségű információ felhalmozódásához vezettek. A korszerű szekvenálási, genotipizálási, génexpressziós, stb. módszerek az évek során egyre olcsóbbá és olcsóbbá váltak, az általuk „termelt" adatok pedig ember számára kezelhetetlen méreteket öltöttek. Ez a tendencia hívta életre a bioinformatika tudományát, amelynek célja a kutatás támogatása az adatok elemzésével, adatbázisok fenntartásával, új módszerek kifejlesztésével.

1.2. Nagy áteresztőképességű módszerek: a génhalászat

Az elektronika területén megfogalmazott Moore-törvény az egy chip-en elhelyezhető tranzisztorok számára vonatkozott, azonban a számítás és adattárolás más területein is hasonló törvényeket fogalmaztak meg, amelyek a csíkszélesség csökkenéséhez legfeljebb igen áttételesen kapcsolódnak. Az információfeldolgozás és adattárolás fejlődését kihasználva szinte minden tudományterületen a méréstechnika is hasonló ütemben fejlődött, az asztrofizikától, nukleáris fizikától, időjáráselőrejelzés és a klímaváltozás kutatásától kiindulva a molekuláris biológiáig. A molekuláris biológiai méréstechnikák esetében a Carlson-törvények a biológiai szekvenciák meghatározására, generálására, de akár fehérjék térszerkezetének a meghatározására is exponenciális jelleget állapítottak meg. A méréstechnikák, adattárolási és elemzési módszerek gyors fejlődése az orvosbiológia területén egy tudományelméleti szempontból is forradalmi megközelítést tett lehetővé, a hipotézismentes kutatási paradigmát. A hipotézismentes kutatási paradigma az omikai megközelítésen alapul, amely egy adott terület entitásainak (genomika-gének, proteomika-fehérjék, stb.) együttes vizsgálatát teszi lehetővé, eltérően a hipotézisek alapján generált kísérlettervezéstől és adatgyűjtéstől. Az új megközelítéshez kapcsolódó, nagy áteresztőképességű biológiai eljárásokra vonatkozik a népszerű szakszleng kifejezés, a „génhalászat".

A hipotézismentes kutatási megközelítés legfőbb kihívása az adatelemzés eredményeinek értelmezése maradt. Az új paradgima és az omikai megközelítés mellett egy harmadik, tudományelméleti szempontból is ismét unikális hozzájárulása az orvosbiológia kutatásoknak a kapcsolt omikai szinteknek a vizsgálata. A genetikai, epigenetikai, transzkriptomikai, proteomikai, metabolomikai, lipidomikai, fenomikai (fenotípusok) szintek egymással kapcsolt értelmezése egy új adat- és tudásintegrációs kihívást jelent.

1.3. Fúziós módszerek az orvosbiológiai kutatásokban

A normatív adat- és tudásintegráció kérdése a helyes következtetés, és azon belül az indukció évezredes problémaköreihez tartozik. A Jacob Bernoulli-tól, Thomas Bayes-től eredeztett szubjektív értelmezésű valószínűségi megközelítés az a priori hiedelmek és a megfigyelések normatív

kombinálására adott megoldást. Ez a fúziós módszer a bayesi döntéselmélet és a bayes statisztika alapját is jelenti, azonban az elméleti egyszerűsége ellenére komplex modellek esetében analitikusan nem kezelhető megoldásokhoz vezet, amelyek nagy számításigényű szimulációkat igényelnek. Ennek megfelelően a bayesi tudásintegrációs megközelítések csupán az utóbbi két évtizedben lettek egyre népszerűbbek. Alapjukat az elérhető háttértudás a priori modellosztályok feletti eloszlásokban való reprezentálása jelenti, amelyet az adatok segítségével a posteriori eloszlások konstruálására vagy akár közvetlen módon maximális hasznosságú akciók meghatározására használunk fel. A bayesi tudásfúzió fő kihívása az informális a priori ismeretek transzformálása formális logikai (kvalitatív) szintre, ami modellosztályokat határoz meg és formális kvantitatív szintre, ami egyes modellosztályokon belül a priori paramétereloszlásokat határoz meg. Az a priori információk felhasználása különösen fontosnak tűnt a komplex modellek, például a génszabályozási hálózatok területén. A transzformáció nehézsége miatt azonban az egyes orvosbiológiai területeken speciális, ad hoc fúziós megközelítések alakultak ki.

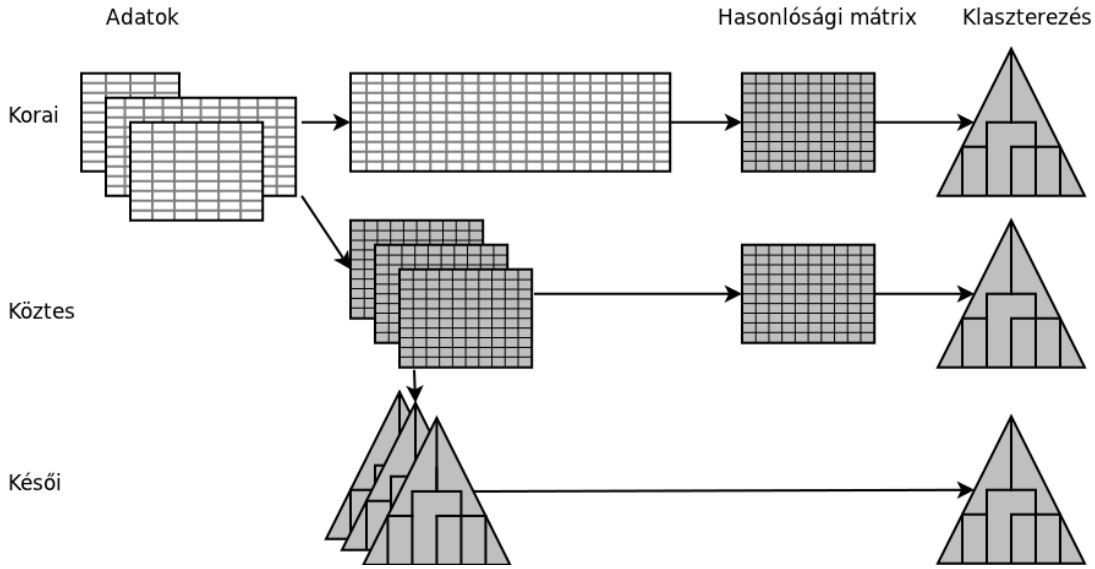
A dolgozatban követett módszerhez nagyon hasonló formalizációra vezetett az expressziós adatok klaszterezésének vizsgálata heterogén információforrások felhasználásával. Mivel a klaszterezési algoritmusok nagy része csak a hasonlósági mátrixokon alapul, ezért háromféle fúziót különböztethetünk meg (1.ábra):

1. *„korai" vagy adatalapú*, amikor az adatvektorokat konkatenáljuk,
2. *„köztes" vagy hasonlósági mátrix alapú*, amikor hasonlósági mátrixokat kombinálunk,
3. *„késői" vagy klaszteralapú*, amikor az egyes információforrások alapján létrejövő klaszterezési eredményeket kombináljuk.

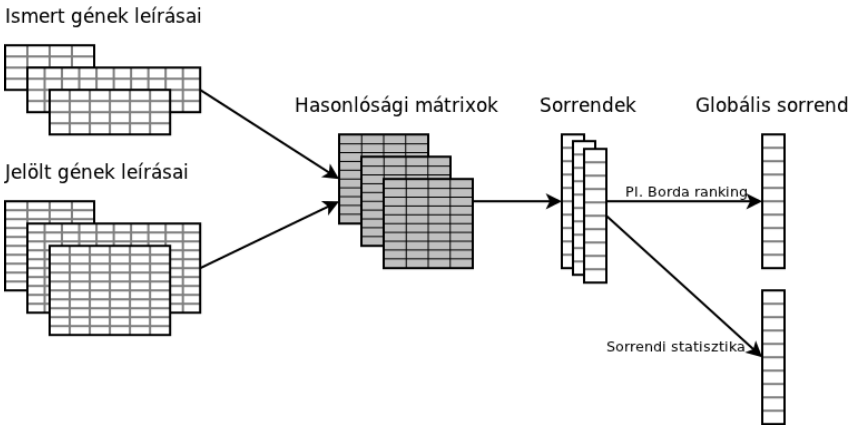
A klaszterezésben megfogalmazott hasonlósági mátrixokon alapuló standardizálás és az erre épülő fúziós kutatások alapján a prediktív vonalon is egy nagyon hasonló standardizálási és formalizálási javaslat fogalmazódott meg 2004-ben, ami a szupportvektor-gépeken alapult [4]. A kernel módszerek használata a prediktív vonalon is lehetővé tette a háromféle fúziót: az adatvektorok, a kernelmátrixok és az eredményül kiadódó prediktorok kombinálásával, ami egyfajta modellátlagolási vagy akár „Mixture of Experts" keretben is formalizálható.

Az adat- és tudásfúziós kutatásokban egy következő korszakot egy új feladat, a génprioritizálás megjelenése hozta, ami a korábbi hálózati és klaszterezési megközelítéseknek egy jelentős egyszerűsítése. A génprioritizálás egy sorrendi tanulási feladat, amit nagyban motivált a szabadszöveges keresőgépek működésének egyszerűsége (pl. PageRank). Leegyszerűsítve a feladat adott „pozitív" mintákhoz megkeresni a leginkább kapcsolódókat a felcímkézetlenek közül, koherensen felhasználva az összes lehetséges adatforrást. Az egyes információforrások alapján történő sorrendezések naív fúziója mellett egy elméleti megközelítés a sorrendek statisztikáján alapult (2. ábra) [5].

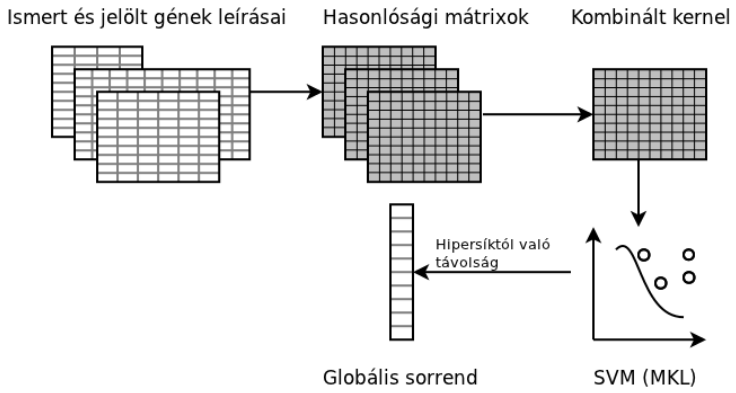
A komolyabb áttörést jelentő, a klaszterezési, majd a prediktív megközelítésben is sikeres



1. ábra. A klaszterezés területén megfogalmazott adatfúziós eljárások. A korai módszer a konkatenált adatvektorok hasonlósági mátrixa alapján klaszterez, a köztes az egyes információforrások adatvektorainak hasonlósági mátrixait kombinálja. A késői módszer lényege a források külön-külön történő klaszterezése utáni fúzió.



2. ábra. A sorrendi fúzió, amelynek lényege a források alapján kiszámított sorrendek egyesítése naív vagy statisztikai módszerekkel.



3. ábra. Kernel fúzió, ahol hasonlósági mátrixokat kombinálunk, majd az SVM-probléma megoldása után hipersíktól való távolság alapján prioritizálunk.

hasonlósági mátrix-alapú fúziót a prioritizálási feladatra is sikerrel adaptálták [6]. Ennek vázlatát a 3. ábra mutatja.

A kernel fúzió – vagy Multiple Kernel Learning – legfőbb előnye a heterogén adatok standardizálása, illetve egy jelenleg még nem kellően felismert tulajdonsága, hogy adott kérdés alapján, a kérdés információtartalmára támaszkodva képes elvégezni a fúziót, azaz a globális fúzió helyett választhatunk egy „intelligens", kisebb tanuló adathalmazt. Ez a lekérdezés specifikus fúzió lehetővé teszi, hogy a megadott minták akár ismeretlen összetartozásának módja kiemeljen egyes, az adott lekérdezés szemponjtából fontos információforrásokat, amire a 4. fejezetben mutatunk példát.

1.4. A GPU, mint lehetőség

A számítástechnikai iparban régóta ismert, hogy a jelenlegi gyártástechnológia igencsak közeledik fizikai korlátaihoz, azaz a Moore-törvény érvényességének is lassan a határára jutunk. A tranzisztorok számának növelése, a csíkszélesség csökkentése egyre nagyobb és nagyobb nehézségekbe ütközik, ahogy az elméleti határhoz közeledünk. A hardvergyártók a növekvő nyomásra válaszolva számos stratégiát fogalmaztak meg. Kézenfekvő a processzormagok számának növelése, amely jelenleg is az egyik uralkodó trend a piacon. Ettől eltérő megközelítés a specializált feldolgozóegységek használata; ide sorolhatók a különböző vektorprocesszorok, vagy akár a még gyerekcipőben járó, emberi neuronhálózat elvén működő CNN-chipek. Napjaink egyik leginten-zívebben kutatott területe az ún. heterogén feldolgozás, amelynek jelentőségét az utóbbi néhány évben ismerték fel. Ennek alapja, hogy adott platformon egyszerre működhetnek általános és speciális feldolgozóegységek, így a megfelelő feladatokra rendelkezésre áll az összehasonlíthatatlanul gyorsabb eszköz is.

A fenti paradigmát alkalmazzuk a GPU általános programozása (GPGPU) során. Nagy előnye, hogy a hagyományos CPU-khoz képest nagyságrendekkel gyorsabb (megfelelő feladatokra

akár százszoros sebességnövekedést is elérhetünk); a számítógépes játékok széleskörű elterjedése pedig erős piacot és folyamatos intenzív fejlesztést biztosít. Hátrányai közé tartozik, hogy a hatékonyan végrehajtható számítások köre korlátozott – nagy mértékben(!) párhuzamosítható algoritmusok szükségesek –, valamint az eszközzel folytatott kommunikáció nagyrészt a lassú PCI-buszon zajlik. E dolgozat alapját az a felismerés adja, hogy az SVM (MKL) módszer során alkalmazott vektorműveletek igen jól párhuzamosíthatók, így lehetséges a kernel fúziós génprioritizálás GPU-ra való implementációja.

1.5. Célkitűzés

A fentiek ismeretében megfogalmazhatjuk a jelen munka alapvető célját, amely az általános Multiple Kernel Learning módszer megvalósítása GPU-ra, valamint genomikai, farmakológiai alkalmazhatóságának vizsgálata.

2. A kernel fúzió és az SMO algoritmus

A heterogén információforrások fúziójának egyik modern megközelítése a Multiple Kernel Learning, amely az utóbbi 5-6 év egyik intenzíven kutatott kérdése. A módszer a gépi tanulás területén népszerű szupportvektor-gépek (SVM) egy kiterjesztése, amely arra keresi a választ, hogy hogyan lehet több kernel mátrix alapján, egyetlen feladat megoldásával megállapítani az optimális szeparáló hipersíkot és az egyes kernelekhez tartozó súlyokat. Erre számos formalizáció született, ám a legmodernebbeknek is közös gyenge pontja, hogy a duál célfüggvény nem differenciálható, így a hagyományos SVM-nél bevált algoritmusok egyáltalán nem voltak használhatók [7, 8], vagy csak a számításigényes Moreau-Yosida regularizáció alkalmazása után [9]. Erre a problémára talált megoldást egy új módszer [10], amelynek gyakorlati alkalmazását egy új területen, az egyosztályos prioritizálási feladatban vizsgáljuk meg. A kernel módszereket és a hozzájuk kapcsolódó általános matematikai fogalmakat ismertnek tételezzük fel, amelyek több áttekintésben is megtalálhatók [11, 12].

A 2.1 alfejezetben elsőként részletesen ismertetjük az SMO-MKL levezetését, a később ismertetett implementációkhoz átdolgozva az eredeti levezetést [10]. A 2.2 alfejezetben leírjuk a probléma újszerű megfogalmazása által lehetségessé vált SMO megközelítés alkalmazását. A 2.3 alfejezetben a bizonytalan heterogén információforrások fúziója kapcsán optimálisan teljesítő speciális Lp, $p = 2$ esetet foglaljuk össze, amely analitikusan kezelhető. A 2.4 alfejezetben a prioritizálási problémákban fontos, egyosztályos feladatra történő alkalmazást közöljük. Végül a 2.5 alfejezetben a prioritizálásban történő alkalmazás képleteit összegezzük.

2.1. Lp regularizált MKL

Legyen adott k darab kernel mátrix a megfelelő Φ_k kernel függvényekkel és d_k súlyokkal. A Multiple Kernel Learning (MKL) probléma megfelel egy, a konkatenált $\sqrt{d_k}\Phi_k(\boldsymbol{x_i})$ vektorok által meghatározott feature térben megoldandó SVM problémának. Ennélfogva a primál feladat így írható:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{d}} \quad & \frac{1}{2}\sum_k \boldsymbol{w_k}^T\boldsymbol{w_k} + C\sum_i \xi_i + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} \\ s.t. \quad & y_i\left(\sum_k \sqrt{d_k}\boldsymbol{w_k}^T\phi_k(\boldsymbol{x_i}) + b\right) \geq 1 - \xi_i \\ & \boldsymbol{\xi} \geq 0, \quad \boldsymbol{d} \geq 0, \end{aligned} \tag{1}$$

ahol – a korábbi módszerekkel szemben – a regularizáló faktor része a célfüggvénynek, λ pedig az egyensúlyt reprezentálja a kernel súlyok és a regularizáció között. A célfüggvény konvexszé

tehető, ha $\boldsymbol{w_k}$ helyére $\sqrt{d_k}\boldsymbol{w_k}$-t helyettesítünk.

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{d}} \quad & \frac{1}{2}\sum_k \frac{\boldsymbol{w_k}^T\boldsymbol{w_k}}{d_k} + C\sum_i \xi_i + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} \\ s.t. \quad & y_i\left(\sum_k \boldsymbol{w_k}^T\phi_k(\boldsymbol{x_i}) + b\right) \geq 1 - \xi_i \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{d} \geq 0. \end{aligned} \tag{2}$$

A primál feladat Lagrange-függvénye a következő:

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \frac{1}{2}\sum_k \frac{\boldsymbol{w_k}^T\boldsymbol{w_k}}{d_k} + \sum_i (C - \beta_i)\xi_i + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} \\ & \quad - \sum_i \alpha_i\left[y_i\left(\sum_k \boldsymbol{w_k}^T\phi_k(\boldsymbol{x_i}) + b\right) - 1 + \xi_i\right]. \end{aligned} \tag{3}$$

A (3) függvényt $\boldsymbol{w}$, b és ξ_i szerint deriválva az alábbi összefüggésekhez jutunk:

$$\begin{aligned} \nabla_{\boldsymbol{w}}\mathcal{L} &= \sum_k \frac{\boldsymbol{w_k}}{d_k} - \sum_i \alpha_i y_i \sum_k \Phi_k(\boldsymbol{x_i}) = 0 \\ & \Rightarrow \sum_k \frac{\boldsymbol{w_k}}{d_k} = \sum_i \alpha_i y_i \sum_k \Phi_k(\boldsymbol{x_i}) \end{aligned} \tag{4}$$

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial b} &= -\sum_i \alpha_i y_i = 0 \\ & \Rightarrow \sum_i \alpha_i y_i = 0 \end{aligned} \tag{5}$$

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial \xi_i} &= C - \beta_i - \alpha_i = 0 \\ & \Rightarrow 0 \leq \boldsymbol{\alpha} \leq C. \end{aligned} \tag{6}$$

Jelölje Q_k a k-adik kernel mátrix címkézett változatát, azaz

$$\begin{aligned} K_k(i,j) &= (\Phi_k(\boldsymbol{x_i}))^T(\Phi_k(\boldsymbol{x_j})) \\ Q_k(i,j) &= y_i y_j K_k(i,j). \end{aligned} \tag{7}$$

A (4), (5), (6) és (7) összefüggéseket a célfüggvénybe visszahelyettesítve az alábbi nyeregpont-problémát kapjuk:

$$\begin{aligned} \min_{\boldsymbol{d}}\max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_k d_k\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} + \mathbf{1}^T\boldsymbol{\alpha} + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} \\ s.t. \quad & \boldsymbol{d} \geq 0, \quad 0 \leq \boldsymbol{\alpha} \leq C, \quad \boldsymbol{y}^T\boldsymbol{\alpha} = 0. \end{aligned} \tag{8}$$

A kernel súlyok eliminálásához ismét vesszük a feladat Lagrange-függvényét:

$$\mathcal{L}(\boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = -\frac{1}{2}\sum_k d_k\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} + \mathbf{1}^T\boldsymbol{\alpha} + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} - \sum_k \gamma_k d_k. \tag{9}$$

d_k szerint deriválva:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial d_k} &= -\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} - \gamma_k + \frac{\lambda}{2}\cdot\frac{2}{p}\cdot\left(\sum_k d_k^p\right)^{\frac{2}{p}-1}\cdot p\cdot d_k^{p-1} = 0 \\ &\Rightarrow \left(\sum_k d_k^p\right)^{\frac{2}{p}} = \frac{1}{\lambda}\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right). \qquad (10)\end{aligned}$$

Célunk, hogy a (10) egyenlet jobb oldalán lévő d_k tagot eliminálják. Legyen

$$\frac{1}{p} + \frac{1}{q} = 1. \qquad (11)$$

Így (10) az alábbi formában írható:

$$\begin{aligned}\left(\sum_k d_k^p\right)^{\frac{1}{p}} = \frac{1}{\lambda} &\cdot \left[\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\right]^{\frac{1}{p}} \\ &\cdot \left[\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\right]^{\frac{1}{q}} \cdot \left(\sum_k d_k^p\right)^{-\frac{1}{p}}.\end{aligned}$$

A jobb oldalon minden tag bevihető az $1/p$ hatványkitevő alá, így

$$\begin{aligned}\left(\sum_k d_k^p\right)^{\frac{1}{p}} &= \left[\frac{1}{\lambda^p}\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\right. \\ &\quad \left.\cdot\left(\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\right)^{\frac{p}{q}}\left(\sum_k d_k^p\right)^{-1}\right]^{\frac{1}{p}},\end{aligned}$$

melyből $d_k^p = d_k d_k^{p-1}$ felhasználásával:

$$d_k^{p-1} = \frac{1}{\lambda^p}\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\left(\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\right)^{\frac{p}{q}}\left(\sum_k d_k^p\right)^{-1}.$$

Mindent az $1/p$ hatványra emelve és a (11) egyenlőséget is felhasználva, valamint a jobb oldal utolsó tagjával leosztva:

$$d_k{}^{\frac{1}{q}}\left(\sum_k d_k^p\right)^{\frac{1}{p}} = \frac{1}{\lambda}\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)^{\frac{1}{p}}\left(\sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)\right)^{\frac{1}{q}}.$$

Melyből $d_k^{1/q}$ -val való osztással és a tagok összevonásával:

$$\left(\sum_k d_k^p\right)^{\frac{1}{p}} = \frac{1}{\lambda}\left(\sum_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)^{1+\frac{q}{p}}\right)^{\frac{1}{q}}.$$

Végül (11)-t alkalmazva és négyzetre emelve megkapjuk a regularizáló tagra vonatkozó összefüggést:

$$\left(\sum_k d_k^p\right)^{\frac{2}{p}} = \frac{1}{\lambda^2}\left(\sum_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} + \gamma_k\right)^q\right)^{\frac{2}{q}}. \qquad (12)$$

A (9) Lagrange-függvény tovább alakítható az alábbi módon:

$$\mathcal{L}(\boldsymbol{d},\boldsymbol{\alpha},\boldsymbol{\gamma}) = \mathbf{1}^T\boldsymbol{\alpha} - \sum_k d_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} + \gamma_k\right) + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}}. \tag{13}$$

Melyből a (10) és (12) egyenletek alapján

$$\begin{aligned}\mathcal{L}(\boldsymbol{d},\boldsymbol{\alpha},\boldsymbol{\gamma}) &= \mathbf{1}^T\boldsymbol{\alpha} - \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} \\ &= \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2\lambda}\left(\sum_k\left(\frac{1}{2}\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} + \gamma_k\right)^q\right)^{\frac{2}{q}}.\end{aligned} \tag{14}$$

Mivel $\gamma_k \geq 0, \quad \boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} \geq 0 \quad \Rightarrow \gamma_k = 0$, azaz γ_k értéke akkor optimális, ha egyenlő nullával. Így végül eljutottunk a duál feladathoz:

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{8\lambda}\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}} \\ s.t. \quad & 0 \leq \boldsymbol{\alpha} \leq C, \quad \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \quad \frac{1}{p} + \frac{1}{q} = 1.\end{aligned} \tag{15}$$

A kernel súlyok számolásához felhasználjuk (10) és (12) ekvivalenciáját.

$$\left(\sum_k d_k^p\right)^{\frac{2}{p}} = \frac{1}{2\lambda}\sum_k d_k\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} = \frac{1}{4\lambda^2}\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}}. \tag{16}$$

A (11) összefüggés alapján:

$$\begin{aligned}\sum_k d_k\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha} &= \frac{1}{2\lambda}\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{-\frac{1}{p}}\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{\frac{1}{q}} \\ d_k &= \frac{1}{2\lambda}\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^{q-1}\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{-\frac{1}{p}}\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{\frac{1}{q}} \\ d_k &= \frac{1}{2\lambda}\left(\sum_k\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^q\right)^{\frac{1}{q}-\frac{1}{p}}\left(\boldsymbol{\alpha}^T Q_k\boldsymbol{\alpha}\right)^{\frac{q}{p}}.\end{aligned} \tag{17}$$

2.2. Az SMO algoritmus

Az SMO (Sequential Minimal Optimization) az ún. coordinate ascent algoritmusok közé tartozik. Elsőként Platt javasolta a memória- és számításigényes numerikus módszerek alternatívájaként [2]. Lényege, hogy iteratív módon, egyszerre mindig csak két Lagrange-multiplikátort optimalizál (míg a többit fixen tartja), azaz a eredeti problémát minimális méretű, analitikusan megoldható szub-problémákra bontja szét, kikerülve ezzel a QP-solver alkalmazását (megjegyzendő, hogy ez SVM-re vonatkozik; jelen esetben az analitikus megoldás csak L2 esetében lehetséges, egyébként a Newton-Raphson módszer vált be). A szub-problémák minimális mérete

a kényszerfeltételekből ered, legkevesebb két multiplikátor szükséges ugyanis ahhoz, hogy ezeket tiszteletben tartsuk a megoldás során. Ismét felírjuk a duál problémát:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{8\lambda}\left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}} \\ s.t. \quad & 0 \le \boldsymbol{\alpha} \le C, \quad \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \quad \frac{1}{p} + \frac{1}{q} = 1. \end{aligned}$$

Legyen a két Lagrange-multiplikátor α_i és α_j, a többi indexet jelölje N, azaz $M \equiv \{1, 2, ..., l\}, N \equiv M \setminus \{i, j\}$. Ekkor a duál feladat így írható:

$$\begin{aligned} \max_{\alpha_i,\alpha_j} \quad & \alpha_i + \alpha_j - \frac{1}{8\lambda}\Bigg[\sum_k \Bigg(\begin{bmatrix} \alpha_i & \alpha_j \end{bmatrix} \begin{bmatrix} Q_{kii} & Q_{kij} \\ Q_{kij} & Q_{kjj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ & + \boldsymbol{Q_{kiN}}\boldsymbol{\alpha_N}\alpha_i + \boldsymbol{Q_{kjN}}\boldsymbol{\alpha_N}\alpha_j \Bigg)^q \Bigg]^{\frac{2}{q}} \\ s.t. \quad & 0 \le \alpha_i, \alpha_j \le C, \quad y_i\alpha_i + y_j\alpha_j = c, \quad \frac{1}{p} + \frac{1}{q} = 1. \end{aligned}$$

Legyenek az új értékek $\alpha_i \Leftarrow \alpha_i + \Delta, \quad \alpha_j \Leftarrow \alpha_j + s\Delta$, ahol $s = -y_i y_j$. Ekkor Δ^* így számolható:

$$\begin{aligned} \Delta^* \quad = \quad & \arg\max_{\Delta} \Delta + s\Delta - \frac{1}{8\lambda}\Bigg[\sum_k \Bigg(\begin{bmatrix} \alpha_i + \Delta & \alpha_j + s\Delta \end{bmatrix} \begin{bmatrix} Q_{kii} & Q_{kij} \\ Q_{kij} & Q_{kjj} \end{bmatrix} \begin{bmatrix} \alpha_i + \Delta \\ \alpha_j + s\Delta \end{bmatrix} \\ & + \boldsymbol{Q_{kiN}}\boldsymbol{\alpha_N}(\alpha_i + \Delta) + \boldsymbol{Q_{kjN}}\boldsymbol{\alpha_N}(\alpha_j + s\Delta) \Bigg)^q \Bigg]^{\frac{2}{q}} \\ s.t. \quad & L \le \Delta \le U, \quad \frac{1}{p} + \frac{1}{q} = 1, \end{aligned}$$

azaz:

$$\begin{aligned} \Delta^* = \arg\max_{\Delta} \quad & (1+s)\Delta - \frac{1}{8\lambda}\Bigg[\sum_k \Big(a_k\Delta^2 + 2b_k\Delta + c_k\Big)^q\Bigg]^{\frac{2}{q}} \\ & a_k = Q_{kii} + Q_{kjj} + 2sQ_{kij} \\ & b_k = \boldsymbol{\alpha}^T(\boldsymbol{Q_{kiM}} + s\boldsymbol{Q_{kjM}}) \\ & c_k = \boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} \\ s.t. \quad & L \le \Delta \le U, \quad \frac{1}{p} + \frac{1}{q} = 1. \end{aligned} \tag{18}$$

Δ lehetséges értékeinek halmaza alulról és felülről korlátos a $0 \le \boldsymbol{\alpha} \le C$ és $\boldsymbol{y}^T\boldsymbol{\alpha} = 0$ kényszerfeltételekből következően. Előbbi miatt a két multiplikátor egy „dobozban" (box constraint), utóbbi miatt egy egyenesen (line constraint) helyezkedik el. A határok így alakulnak:

$$L = \begin{cases} max(-\alpha_i, -\alpha_j), & \text{ha } s = 1 \\ max(-\alpha_i, \alpha_j - C), & \text{ha } s = -1 \end{cases}$$

$$U = \begin{cases} min(C - \alpha_i, C - \alpha_j), & \text{ha } s = 1 \\ min(C - \alpha_i, \alpha_j), & \text{ha } s = -1 \end{cases}$$

Δ tartományon kívülre eső értékei esetében az algoritmus vágást alkalmaz.

A hatékonyság szempontjából kritikus, hogy minden iterációban olyan multiplikátorokat válasszunk, amelyekkel a legnagyobbat léphetünk a megoldás felé (working set selection). Az egyik legelterjedtebb módszer Lin WSS2 algoritmusa [13], amelyhez szükséges a célfüggvény gradiensének és Hesse-mátrixának kiszámítása. Felhasználjuk, hogy a (16) egyenlőség ismeretében a célfüggvényt így is írhatjuk:

$$\mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\sum_k d_k \boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}.$$

A gradiens így számolható:

$$\nabla_{\boldsymbol{\alpha}}\mathcal{D} = 1 - \sum_k d_k Q_k \boldsymbol{\alpha}.$$

A Hesse-mátrix:

$$\begin{aligned}
\nabla^2_{\boldsymbol{\alpha}}\mathcal{D} &= -\sum_k \left(d_k Q_k + \nabla_{\boldsymbol{\alpha}} d_k \cdot Q_k\boldsymbol{\alpha}\right) \\
&= -Q - \sum_k \nabla_{\boldsymbol{\alpha}} \left[\frac{1}{2\lambda}\left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}-1} \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^{q-1}\right] Q_k\boldsymbol{\alpha} \\
&= -Q - \frac{1}{2\lambda}\sum_k \left[\left(\frac{2}{q}-1\right)\left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}-2} \cdot q \cdot \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^{q-1}\right. \\
&\quad \cdot 2 \cdot Q_k\boldsymbol{\alpha} \cdot \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^{q-1} \cdot Q_k\boldsymbol{\alpha} + (q-1)\left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^{q-2} \cdot 2 \cdot Q_k\boldsymbol{\alpha} \\
&\quad \left.\cdot \left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}-1} \cdot Q_k\boldsymbol{\alpha}\right] \\
&= -Q - \frac{1}{\lambda}\sum_k \left[(2-q)\left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^{2q-2}\left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}-2}\right. \\
&\quad \left. + (q-1)\left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^{q-2}\left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}-1}\right] (Q_k\boldsymbol{\alpha})^T (Q_k\boldsymbol{\alpha}),
\end{aligned}$$

ahol ismét felhasználtuk a (11) összefüggést. Látható, hogy a kernelekből egyszerre mindössze két oszlop, illetve a diagonális tárolása szükséges, így a memóriaigény töredékére csökken.

2.3. Kiterjesztés biológiai problémákra

Általánosságban elmondható, hogy minél alacsonyabbnak választjuk p értékét, annál ritkább kernelkombinációhoz jutunk, azaz pl. $p = 1$ esetén kevés súly kap magas értéket, míg a többi nullát vagy közel nullát. Ez hasznos, ha nagyszámú kernellel dolgozunk, amelyek közül a „legjobbakat" szeretnénk kiválasztani, előnytelen viszont biológiai problémák esetében; itt rendszerint kevés, gondosan kiválasztott kernelről van szó, melyek mindegyike fontos információt hordoz. [14] azt találta, hogy az L2-MKL szignifikánsan jobban teljesített a sparse módszereknél mind

hatékonyság, mind predikciós teljesítmény tekintetében. Ennek megfelőlen a duál probléma így módosul:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{8\lambda}\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^2 \\ s.t. \quad & 0 \leq \boldsymbol{\alpha} \leq C, \quad \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \quad \frac{1}{p} + \frac{1}{q} = 1, \end{aligned} \tag{19}$$

a (17) egyenlet alapján a kernel súlyok:

$$d_k = \frac{1}{2\lambda}\sum_k \boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} \tag{20}$$

(18) pedig:

$$\begin{aligned} \Delta^* = \arg\max_{\Delta} \quad & (1+s)\Delta - \frac{1}{8\lambda}\sum_k \left(a_k\Delta^2 + 2b_k\Delta + c_k\right)^2 \\ & a_k = Q_{kii} + Q_{kjj} + 2sQ_{kij} \\ & b_k = \boldsymbol{\alpha}^T(\boldsymbol{Q_{kiM}} + s\boldsymbol{Q_{kjM}}) \\ & c_k = \boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha} \\ s.t. \quad & L \leq \Delta \leq U, \quad \frac{1}{p} + \frac{1}{q} = 1. \end{aligned} \tag{21}$$

A gradiens változatlanul így számolható:

$$\nabla_{\boldsymbol{\alpha}}\mathcal{D} = 1 - \sum_k d_k Q_k \boldsymbol{\alpha}, \tag{22}$$

míg a Hesse-mátrix:

$$\nabla^2_{\boldsymbol{\alpha}}\mathcal{D} = -Q - \frac{1}{\lambda}\sum_k (Q_k\boldsymbol{\alpha})^T(Q_k\boldsymbol{\alpha}). \tag{23}$$

Látható, hogy (21) analitikusan megoldható, ugyanis deriválás valamint a négyzetes tag kifejtése után az alábbi egyenletet kapjuk:

$$1 + s - \frac{1}{8\lambda}\sum_k \left(4a_k^2\Delta^3 + 3\cdot 4a_k b_k \Delta^2 + 2\cdot(2a_k c_k + 4b_k^2)\Delta + 4b_k c_k\right) = 0,$$

ahol a konstans c_k^2 tag kiesett, Δ^* pedig a harmadfokú egyenletek megoldóképletével megtalálható. Δ^0 együtthatója itt:

$$\begin{aligned} 1 + s - \frac{1}{2\lambda}\sum_k b_k c_k &= 1 - \frac{1}{2\lambda}\sum_k d_k \boldsymbol{\alpha}^T\boldsymbol{Q_{kiM}} + s - \frac{1}{2\lambda}\sum_k d_k \cdot s \cdot \boldsymbol{\alpha}^T\boldsymbol{Q_{kjM}} \\ &= \boldsymbol{\nabla_{\alpha}\mathcal{D}}_i + s\boldsymbol{\nabla_{\alpha}\mathcal{D}}_j, \end{aligned}$$

ahol felhasználtuk a (20) és (22) összefüggéseket.

2.4. Kiterjesztés egyosztályos prioritizációra

Az eddigiekben leírtak a kétosztályos (illetve kevés módosítással a többosztályos) MKL-re vonatkoztak, ennek működéséhez „pozitív" és „negatív" tanítóminták egyaránt szükségesek. Biológiai problémák esetében ez a megoldás gyakran nem célravezető. Képzeljük el például, hogy adott gének – pl. az asthma patogenezisében szerepet játszó jelátviteli útvonalak génjei – ismeretében próbálunk további, eddig ismeretlen, ide kapcsolódó genetikai tényezőket felderíteni. Ezeket a tényezőket nem sorolhatjuk egyöntetűen az $y = -1$ osztályba, mivel mindegyik a maga egyéni módján „negatív".

A fenti feladat megoldására alkalmas az egyosztályos (one-class) SVM, melyet először Schölkopf javasolt [15]. A levezetés nem különbözik az eddigiekben leírtaktól, így ettől eltekintünk, és csak a végeredményt közöljük. MKL esetén a primál probléma:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{d},\rho} \quad & \frac{1}{2}\sum_k \frac{\boldsymbol{w_k}^T\boldsymbol{w_k}}{d_k} - \rho + \frac{1}{\nu l}\sum_i \xi_i + \frac{\lambda}{2}\left(\sum_k d_k^p\right)^{\frac{2}{p}} \\ s.t. \quad & \sum_k \boldsymbol{w_k}^T\phi_k(\boldsymbol{x_i}) \geq \rho - \xi_i \\ & \boldsymbol{\xi} \geq 0, \boldsymbol{d} \geq 0, \quad i = 1, 2, ..., l. \end{aligned}$$

A duál:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{8\lambda}\left(\sum_k \left(\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}\right)^q\right)^{\frac{2}{q}} \\ s.t. \quad & 0 \leq \boldsymbol{\alpha} \leq 1, \quad \mathbf{1}^T\boldsymbol{\alpha} = \nu l, \quad \frac{1}{p} + \frac{1}{q} = 1. \end{aligned}$$

Itt – a korábbiaktól eltérően – az osztályokról nincs információnk, csak egyetlen tanítóhalmazt szükséges megadni.

2.5. Kernel alapú prioritizáció

Az optimális szeparáló hipersík kiszámítása után a további adatpontok (pl. gének) sorrendezhetők a hipersíktól való távolságuk alapján. (4)-ből tudjuk, hogy

$$\sum_k \boldsymbol{w_k} = \sum_i \alpha_i y_i \sum_k d_k \Phi_k(\boldsymbol{x_i}),$$

így a $\boldsymbol{z}$ adatpont score-ja a következőképpen számolható:

$$f(\boldsymbol{z}) = \frac{\sum_k \boldsymbol{w_k}^T\phi_k(\boldsymbol{z}) + b}{\|\boldsymbol{w}\|} = \frac{\sum_i \alpha_i y_i \sum_k d_k K_k(\boldsymbol{x_i}, \boldsymbol{z}) + b}{\sqrt{\sum_k d_k \boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}}},$$

illetve egyosztályos esetben:

$$f(\boldsymbol{z}) = \frac{\sum_i \alpha_i \sum_k d_k K_k(\boldsymbol{x_i}, \boldsymbol{z}) - \rho}{\sqrt{\sum_k d_k \boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}}},$$

ahol $K_k(\boldsymbol{x_i}, \boldsymbol{z}) = (\phi_k(\boldsymbol{x_i}))^T(\phi_k(\boldsymbol{z}))$.

3. Az SMO-MKL alapú fúzió GPU-s implementációja

3.1. Az OpenCL keretrendszer

Az OpenCL eredetileg egy Apple által kifejlesztett nyílt szabvány, amelyet a Khronos group tart karban, csakúgy, mint a jól ismert OpenGL-t [16]. Célja, hogy a heterogén platformokon zajló, általános célú párhuzamos programozás számára egy egységes keretrendszert nyújtson, amely megkönnyíti a – főleg, de nem kizárólag – teljesítmény-orientált alkalmazások fejlesztését. Az 1.1-es specifikáció 2010. június 14-ére készült el, számos nagy- és kisebb vállalat támogatásával (AMD, NVIDIA, Intel, stb.). Az alábbiakban egy rövid áttekintés következik az OpenCL architektúráról, elsősorban AMD szemszögből, mivel a jelen munka AMD grafikus processzorok felhasználásával készült. Az architektúra az alábbi négy modellel írható le:

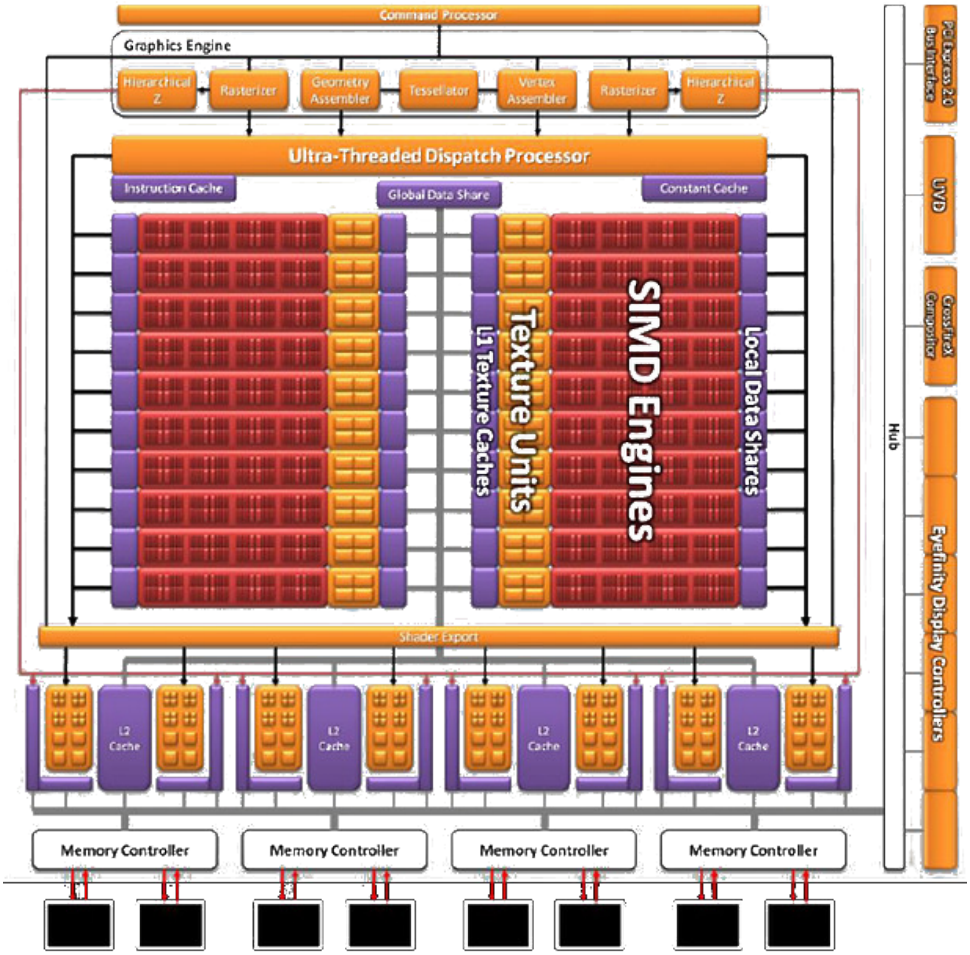
- Platform modell
- Végrehajtási (execution) modell
- Memória modell
- Programozási modell

3.1.1. A platform modell és az AMD (ATI) GPU-k architektúrája

A modell alapját a host és a hozzá kapcsolódó eszközök (devices) képzik. A host-on fut az OpenCL alkalmazás, amely parancsokat (command) ad az eszköznek – jelen esetben a GPU-nak, illetve magának a CPU-nak. Az eszközben számítási egységek (compute unit) találhatók, amelyek GPU esetében megfelelnek a SIMD egységeknek, CPU esetén az egyes magoknak. A SIMD egységek (Single Instruction, Multiple Data) tovább bonthatók ún. stream magokra (stream core), amelyek mindegyike egy VLIW-5 (very long instruction word) architektúrára épülő feldolgozóegységet rejt. A 4. ábrán látható az ATI Radeon HD 5870 felépítése, az általunk használt eszközök adatai a következő alfejezetben megtalálhatók.

3.1.2. A végrehajtási modell

Két részre osztható: a host alkalmazásra, és az eszközön futó, ún. kernelekre. A nevek szerencsétlen ütközése miatt ebben a munkában a kernel kifejezést továbbra is a matematikai értelemben használjuk, az eszközön futó függvényekre az OpenCL függvény megnevezést fogjuk alkalmazni. Mikor a host alkalmazás egy OpenCL függvényt kíván futtatni, elküldi az eszköznek a megfelelő parancsot (függvény, paraméterek, stb.), kiegészítve a programozó által megadott index térre vonatkozó információkkal, azaz meghatározza, hány példányban és milyen elrendezésben fog futni a függvény. A függvény egy futó példányát nevezzük munkaegységnek (work item), adott mennyiségű, azonos számítási egységen futó munkaegységeket munkacsoportnak



4. ábra. A Cypress architektúra (HD5850 és HD5870) [1]

(work group, általában 64 vagy 256 egység). Egy munkaegység egy stream magon fut, ami a párhuzamos feldolgozás két szintjét jelenti: egyrészt több stream magon folyik egyszerre a számítás, másrészt a VLIW5 architektúra eredményeképp ún. utasításszintű párhuzamosság is jelen van (4 lebegőpontos + 1 speciális művelet, pl. sin, exp). Az egyszerre végrehajtódó munkaegységek blokkját wavefront-nak nevezik, azonos wavefront munkaegységei közül 4 darab sorakozik fel (pipeline) egy stream magra. Ez a memória latenciájának elrejtését szolgálja; amíg egy munkaegység az adatokra vár, egy másik használhatja a feldolgozóegységet helyette.

3.1.3. A memória modell

Négy különböző memóriatípust különítenek el: privát, lokális, globális, konstans. A privát memória a leggyorsabb, a munkaegység sajátja, gyakorlatban egy regiszter. A lokális memória a SIMD egység on-chip memóirája, a munkacsoport tagjai által megosztott. Leglassabb, viszont legnagyobb méretű a globális memória, amely minden munkaegység által elérhető, illetve a host alkalmazás is ide tud írni. A konstans memória a kis méretű, nem változó adatok gyorsítótárazott

tárolására szolgál. Speciális az ún. textúra (kép) memória, amelyről most nem beszélünk, az érdeklődő olvasót az AMD leírásához irányítjuk [1].

3.1.4. A programozási modell

Az OpenCL programozási modell támogatja az adatpárhuzamos (SIMD) és feladatpárhuzamos végrehajtást; utóbbiról itt nem beszélünk. A környezetet, amelyben az OpenCL függvények és memória-operációk futnak, kontextusnak (context) nevezzük, amely magában foglalja a hostot, az eszközöket és ezek memóriáját és a parancssort (command queue). Az OpenCL függvények létrehozásához először az OpenCL programokat kell megírni, amelyek tartalmazhatnak hagyományos függvényeket is. Mindehhez egy kiterjesztett C nyelvet lehet használni, majd a programot a host alkalmazás futása közben lefordítja a megadott eszközre (JIT, just-in-time compilation). A kész binárisból hozhatók létre az OpenCL függvények, majd ezek felsorakoztathatók a parancssorba, csakúgy, mint a különböző memória-operációk.

3.2. Felhasznált eszközök

A fejlesztést Ubuntu 11.04 és Windows 7 operációs rendszereken végeztük. A rendszer az alábbi fordítókkal (biztosan) kompatibilis: gcc 4.4.5, msvc10. A fejlesztés során az AMD APP SDK 2.4-es verzióját használtuk. A felhasznált eszközök adatai az alábbi táblázatban láthatók:

Típus	ATI Radeon HD5470	ATI Radeon HD5850	Intel Core i5 430M
Órajel	750 MHz	725 MHz	2,26 GHz
SIMD egységek	2	18	4 (HyperThreading)
Stream magok	16	288	-
Munkaegységek	80	1440	-
Maximális GFLOPS	120	2088	18,08
Globális memória méret	1 GB	2 GB	3 GB (host memória)
Globális memória sávszél	25,6 GB/s	128 GB/s	8 GB/s (host memória)
Lokális memória/SIMD	32 kB	32 kB	-

2. táblázat. Hardveradatok

3.3. A rendszer felépítése

A tervezés során alapvető szempont volt a hatékonyság és a megfelelő teljesítmény biztosítása; nem elhanyagolható azonban a karbantartás és a kiterjeszthetőség kérdése sem. A korszerű C++ irányzatoknak megfelelően az ún. policy-alapú tervezés mellett döntöttünk, amelyet először Alexandrescu javasolt [17]. Ez azt jelenti, hogy az osztályhierarchiákat, virtuális függvényhívásokat használó objektumorientált programozás helyett a generikus programozás elveit követve

kis „policy” osztályokat hozunk létre, majd ezeket fordítási időben, az igényeknek megfelelően összekapcsolva alakul ki a végleges kód. Ez egyrészt az újrafelhasználást és az egyszerű bővítést segíti, másrészt a gyakorlatban gyorsabbnak bizonyult a klasszikus objektumorientált megoldásoknál [17]. További előnye a letisztult, felhasználóbarát felület, amelynek segítségével egy-egy tanulási feladat 8-10 sornyi kóddal megvalósítható. Hátrányai között említhető a hosszabb fordítási idő, hiszen ekkor jönnek létre a konkrét osztályok a különböző policy-k összefűzésével, és a metaprogramok is ekkor futnak le. További hátrány, hogy a bináris mérete növekszik, illetve egyes fordítók (pl. clang 2.9) még nem képesek értelmezni az így megírt kódot.

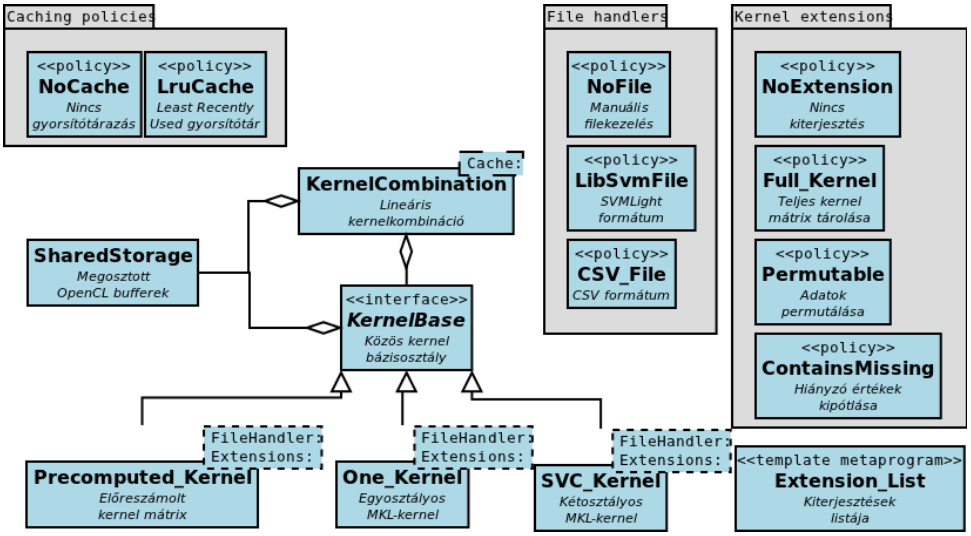
Az OpenCL eszközökkel való kommunikációt az Engine osztály végzi. Feladata a platformok, eszközök és a parancssor kezelése, az eszközön futó binárisok létrehozása, sikertelen fordítás esetén a napló lekérése, a hibakódok lefordítása, illetve az OpenCL munkacsoportok számának és méretének automatikus beállítása (lásd Optimalizációk). Az eszköz memóriájának elérése a Buffer sablonosztályon keresztül lehetséges, amely számos kényelmi funkciót (fill, host-ra másolás, stb.) is nyújt. A tipikus felhasználás során a fent említettek nem szükségesek, de biztosítjuk a lehetőséget az eszköz közvetlen kezelésére (bővítés, hibaelhárítás, stb.).

A rendszer jelenleg az egy- és kétosztályos (illetve többosztályos) MKL-t támogatja. Az SMO algoritmusnak köszönhetően a kernelekből egyszerre mindössze két oszlop, illetve a diagonális tárolása szükséges. Ezeket számolhatjuk a GPU-n, illetve megadhatunk előre kiszámolt mátrixokat is (pl. gén-gén interakciós mátrixok, ahol a vektoriális leírás nem létezik). Előbbi esetben az alábbi hasonlóságmértékek/függvények támogatottak:

- RBF (Radial Basis Function): $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\gamma \|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}$
- Polinomiális: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (p \cdot \boldsymbol{x_i}^T \boldsymbol{x_j} + q)^r$
- Szigmoid: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = tanh(p \cdot \boldsymbol{x_i}^T \boldsymbol{x_j} + q)$
- Koszinusz: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i}^T \boldsymbol{x_j}) / (\|\boldsymbol{x_i}\| \, \|\boldsymbol{x_j}\|)$
- Tanimoto: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i}^T \boldsymbol{x_j}) / (\|\boldsymbol{x_i}\|^2 + \|\boldsymbol{x_j}\|^2 - \boldsymbol{x_i}^T \boldsymbol{x_j})$

A testreszabás, illetve a kiterjesztés egyik módja bővítmények alkalmazása, ezeket egy ún. „template metaprogram” kapcsolja a kernelt reprezentáló osztályokhoz. A jelenleg rendelkezésre álló bővítmények:

- **Adatok permutálása.** Adott entitás-feature mátrix esetén minden feature-nek megfelelő oszlopot megpermutálunk, és az így kapott vektorok alapján számítjuk a kernelt. Mivel az OpenCL keretrendszerben nincs „gyári” pszeudorandom-generátor, erre a célra a statisztikai szimulációkban gyakran alkalmazott Mersenne Twistert valósítottuk meg, melyet a host oldalon generált pszeudorandom számmal hajtunk meg [18].



5. ábra. Kernel osztálydiagram. A Kernel objektumok a mátrix oszlopait számolják, a KernelCombination osztály az összegzést és a gyorsítótárazást végzi. A kernelek bővíthetők tetszőleges számú bővítménnyel, ez az Extension_List template metaprogramon keresztül lehetséges. Szükséges még egy filekezelő megadása is.

- **Hiányos adatok kezelése.** A hiányzó értékeket pótolhatjuk egy átlagos hasonlósággal, ehhez azonban először az összes érték kiszámítása szükséges, amely hatékonysági kérdéseket vet fel.
- **Teljes kernel kiszámítása.** Kis adatmennyiség esetén előnyös lehet a teljes mátrix kiszámítása; határt jelenthet a GPU-n rendelkezésre álló memória.

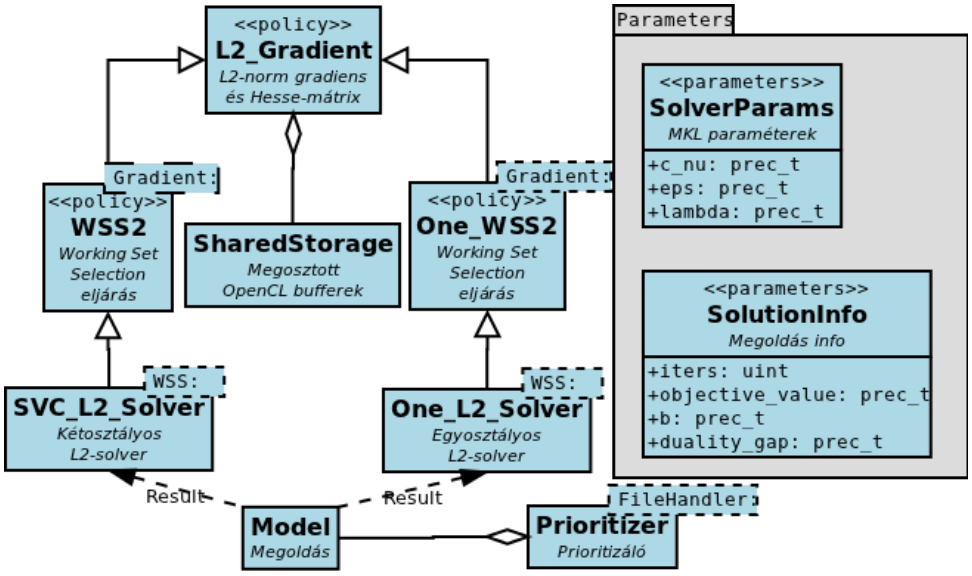
A policy-alapú tervezésnek köszönhetően igen könnyű további bővítményeket írni, illetve a kernel objektumok könnyedén testreszabhatók az igényeknek megfelelő filekezelő és bővítmények megadásával.

A kerneleket a KernelCombination osztály fogja össze, amely a lineáris kombinációt reprezentálja. Ugyancsak ez az osztály kezeli a gyorsítótárat, amelyről részletesebben az optimalizációk fejezetben szólunk. Összefoglalásként a 5. ábrán látható az idevágó osztálydiagram.

A megoldást az L2_Solver osztályok szolgáltatják, ehhez meg kell adnunk az alkalmazott „Working Set Selection” eljárást és a szükséges paramétereket, név szerint:

- C vagy ν: mint az egyszerű kétosztályos vagy egyosztályos SVM-nél,
- ε: Konvergencia küszöb,
- λ: Lásd (1) egyenlet és leírás.

A kimenet a Model osztály egy példánya, amely tartalmazza a megoldáshoz tartozó Lagrange-multiplikátorokat (pontosabban azok címkével vett szorzatát), az optimális súlyozást, a „bias



6. ábra. Solver osztálydiagram. A solver osztályok számítják ki az MKL-modellt, ehhez tetszőleges WSS eljárás választható. A Gradient osztály feladata a gradiens tárolása, frissítése és a Hesse-mátrix kiszámítása. A prioritizálás bemenete a kiszámolt MKL-modell.

term" értékét, a szupportvektorok számát és a kernelekre vonatkozó adatokat (függvény, paraméterezés, a nem nulla multiplikátorokhoz tartozó training vektorok). Visszaad továbbá egy SolutionInfo objektumot, amely a megoldás egyéb adatairól tájékoztat (iterációk, célfüggvény értéke, „duality gap"). A klasszifikációhoz illetve prioritizációhoz szükséges hipersíktól való távolságokat a Prioritizer osztály számolja.

3.4. Optimalizáció

3.4.1. A GPU-ra vonatkozó optimalizáció

A GPU-ra történő optimalizáció más elvek alapján történik, mint CPU esetében. Jelen munkában az előbbire törekedtünk, így a kód lassabban fut CPU-n, mint a natív szekvenciális kód. Néhány alapelv és alkalmazásuk:

- **Általában szűk keresztmetszet az adatok átvitele a grafikus kártyára**, ugyanis ehhez a lassú PCI-buszt kell használni. A probléma kiküszöbölésére megoldásunk mindent a GPU memóriájában tárol, ami az esetek legnagyobb részében elegendő. Extrém méretű training adat (milliós nagyságrendű adatpont ezres dimenziószámmal) mostani implementációnkkal még nem kezelhető, csak a csúcsmodell HD6990 VGA segítségével.
- **Sokkal nagyobb a számítási teljesítmény, mint a CPU esetében.** Gyakran gyorsabb ismét kiszámítani egy részeredményt, mint eltárolni.

- **Másképp működnek az elágazások.** Ha egy munkacsoporton belül az elágazások divergálnak, akkor ezen munkacsoport egységei kétszer futtatják le a függvényt oly módon, hogy először az egyik, majd a másik ág hajtódik végre. Ennek következményeként érdemes kerülni az if-eket. Ha mégis erre van szükség, akkor célszerű olyan elrendezést választani, hogy egy munkacsoporton belül csak az egyik ág fusson (hiszen elég egyetlen „kakukktojás", és máris kétszeresére nyúlik a futásidő). Rendszerünk ezt a beépített select függvénnyel oldja meg, amely nem hoz létre elágazást, hanem a kapott logikai érték alapján, utasításszinten választja ki a megfelelőt két érték közül. Hagyományos if-es ciklusra csak a MapReduce implementációnknál volt szükség.

- **A globális memóriára vonatkozó elvek:** Ajánlott a 128 bites igazítás (alignment) alkalmazása, ez gyakorlatban az adatsorok végének kitöltését (padding) jelenti. A sávszélesség kihasználása akkor optimális, ha a szomszédos munkaegységek szomszédos memóriacímekhez férnek hozzá. Ellenkező esetben előfordulhat, hogy sok munkaegység ugyanazon a bank-on lévő címet próbál elérni, ekkor a hardver szerializálja a hozzáférést. Ugyancsak akkor optimális a kihasználás, ha egyszerre több értéket (pl. float4 vektorokat) érünk el. Az általunk implementált fájlkezelők a tanuló adatokat automatikusan oszlopos (coloumn major) elrendezésbe olvassák és 128 bitre igazítják. A függvények float4 és int4 vektorokat használnak, az oszlopos elrendezés lehetővé teszi a szomszédos („coalesced") hozzáférést.

- **Az on-chip lokális memória használata.** Ennek elérése sokkal gyorsabb, mint a globális memóriáé, így érdemes a minden munkaegység által használt közös adatokat először ide betölteni. A munkacsoporthoz tartozó lokális memória mérete azonban egy ponton túl az egyszerre futó csoportok számát korlátozza. Figyelni kell itt is a bank-ütközésekre.

- **Megfelelő mennyiségű munkát kell adni a munkaegységeknek**, így kevesebb példányban kell futtatni a függvényt, ami csökkenti az indításnál fellépő időveszteséget. **VISZONT egyúttal megfelelő mennyiségű munkacsoportot kell indítani**, mivel a memóriaelérések (4-600 órajelciklus!) alatt a várakozó munkacsoport helyett egy másik tud futni az adott stream magon, így elegendőt felsorakoztatva („pipeline") a memória latenciája elrejthető. Az egyensúly megtalálása döntő kérdés. A munkacsoportok mérete ideálisan a 64 többszöröse. Mindebből látszik, hogy kritikus a munkacsoportok méretének és számának jó megválasztása. Mi az alábbi heurisztikára hagyatkozunk: legyen a munkacsoportok száma annyi, hogy minden SIMD egységre 4 darab jusson. Ez épp elegendő, hogy elrejtse a latenciát, azaz az egységet mindig „lefoglalja". A méret legyen a lehető legnagyobb (pl. HD5470: 64, HD5850: 256), a munkát pedig egyenletesen osszuk el a létrejövő munkaegységek között. CPU esetében legyen mindkét tényező 1.

- **Minél kevesebb regiszter használata.** A használt regiszterek száma csökkenti az egyszerre futó munkaegységek számát, sőt egy ponton túl az értékek a globális memóriába

kerülnek, ami hatalmas esést jelent a hatékonyságban.

- **Vektorműveletek használata.** Mivel a munkaegységek alatt valójában VLIW5/VLIW4 architektúrájú feldolgozóegység húzódik meg, az utasításszintű párhuzamosság kihasználásához egyszerre több elemmel, gyakorlatban gentype4 (float4, int4, stb.) vektorokkal ajánlott dolgozni. Ebből a CPU is profitál, a fordító itt képes a műveleteket SSE-utasításokba csomagolni. Mivel a fordító nem tud ciklusokon keresztül összevonni műveleteket, ezért ajánlott a "loop unrolling", azaz egy ciklusban több művelet elvégzése, kevesebb cikluson át.

- **További optimalizációk.** Minden egyes kernel objektum egyedi OpenCL binárist használ, amelybe belefordít egyes paramétereket, pl. a tanuló adatmátrix méreteit, így a fordító további optimalizációkat tud kivitelezni. Ez gondot jelenthet nagyszámú kernel esetén, azonban az L2-normalizációnál feltételeztük, hogy keveset fogunk használni. A jelenleg még nem támogatott L1-normalizációnál is célszerű a méreteket hasonlóképp kezelni, hiszen ott jelemzően arról van szó, hogy ugyanarra a tanuló adatmátrixra alkalmazunk igen sok kernel függvényt.

3.4.2. Az algoritmusra vonatkozó optimalizáció

Öt különböző gyorsítótárat alkalmazunk az algoritmusban, melyek közül elsőként a két kernel oszlop gyorsítótárazását mutatjuk be. Jelenleg az ún. LRU (Least Recently Used) módszert alkalmazzuk, ugyanis [19] megmutatta, hogy nagyobb eséllyel választunk ki ismét egy multiplikátort (és ezzel együtt egy oszlopot), ha azt nemrég kiválasztottuk. A GPU memóriájának legnagyobb része erre a célra van fordítva, de természetesen megadhatunk más méretet is. Valójában nem a kombinált kernel oszlopait tároljuk, hanem az egyes kernelekét, ezekre ugyanis külön-külön is szükség van (pl. Δ együtthatóinak számolásakor). Megjegyzendő, hogy a teljesítmény növekedése nem minden feladatnál ugyanakkora; elsősorban az adatoktól függ, hogy mennyit nyerünk ezzel az optimalizációval.

A többi gyorsítótár a diagonálist (hiszen ez nem változik), a gradienst, a $Q_k\boldsymbol{\alpha}$, és az $\boldsymbol{\alpha}^T Q_k \boldsymbol{\alpha}$ szorzatokat tartalmazza. Utóbbi három α_i és α_j új értékeinek ismeretében hatékonyan frissíthető. [10] azt találta, hogy a Hesse-mátrix tárolása túl költséges, így ezt mi sem alkalmazzuk.

A megfelelő multiplikátorok kiválasztásához Lin WSS2 módszerét használjuk, amelynek első lépésében egy sorozat maximumát, másodikban egynek minimumát kell kiválasztani [3]. Ehhez a Google által kifejlesztett MapReduce algoritmust implementáltuk GPU-ra [20].

3.5. Teljesítményadatok

A tesztelésnél referenciaként a [10] cikk szerzőinek kódját (a továbbiakban SMO-MKL), valamint a Shogun toolbox 1.0.0 verzióját használtuk [21]. A teljesítményt az UCI Adult adatbázisán

Tábla	Tanuló adatpontok	Teszt adatpontok	Dimenziószám
a1a	1605	30956	124
a2a	2265	30296	124
a3a	3185	29376	124
a4a	4781	27780	124
a5a	6414	26147	124
a6a	11220	21341	124
a7a	16100	16461	124
a8a	22696	9865	124
a9a	32561	16281	124

3. táblázat. Tanuló és teszt adatmátrixok mérete

mértük, amely 9 különböző méretű táblát tartalmaz [22]. A 3. táblázatban láthatók a tanuló- és tesztadatok méretére vonatkozó információk.

A mérést először – a más közleményekkel való összevethetőség érdekében –kétosztályos tanulási feladattal és bináris klasszifikációval végeztük. Az alkalmazott kernel függvények és paramétereik:

- RBF ($\gamma = 0.1$)
- RBF ($\gamma = 0.5$)
- Lineáris
- Polinomiális ($p = 2$)

A választott MKL-paraméterek:

- $C = 100$
- $\varepsilon = 0.001$
- $\lambda = 0.1$ (csak SMO-MKL, HD5470 és HD5850)

Megjegyzendő, hogy a jelen fejezetben elsősorban a teljesítményt vizsgáljuk, tehát nem törekedtünk a lehető legnagyobb klasszifikációs pontosság elérésére. Megfelelő kernelekkel és alaposan átgondolt paraméterezéssel nagyobb pontosságot lehet elérni; mi itt beérjük a tapasztalt 82-84%-kal. A λ paraméter megválasztásáról később külön szólunk, itt előzetes ismertek hiányában önkényesen 0.1-nek választottuk.

A 4. táblázatban találhatók a mérési eredmények és az SMO-MKL algoritmushoz képest tapasztalt gyorsulás. A Shogun rendszer az utolsó két tábla esetében nem adott eredményt belátható időn belül. Látható, hogy az igen gyenge, alsókategóriás HD5470 megfelelő méretű

Méret	SMO-MKL (s)	HD5470 (s)	HD5850 (s)	Shogun (s)	HD5470 gyorsulás	HD5850 gyorsulás
1605	2.01	6.60	3.51	20.06	0.30x	0.57x
2265	3.99	10.38	5.00	41.73	0.38x	0.79x
3185	7.67	17.53	7.59	51.90	0.43x	1.01x
4781	17.08	31.72	11.51	121.5	0.53x	1.48x
6414	35.03	49.65	15.44	218.33	0.70x	2.26x
11220	259.15	105.35	27.19	704.87	2.45x	9.53x
16100	713.35	209.49	46.98	2235.78	3.40x	15.18x
22696	1631.55	375.54	71.80	-	4.34x	22.72x
32561	3300.47	667.61	111.10	-	4.94x	29.70x

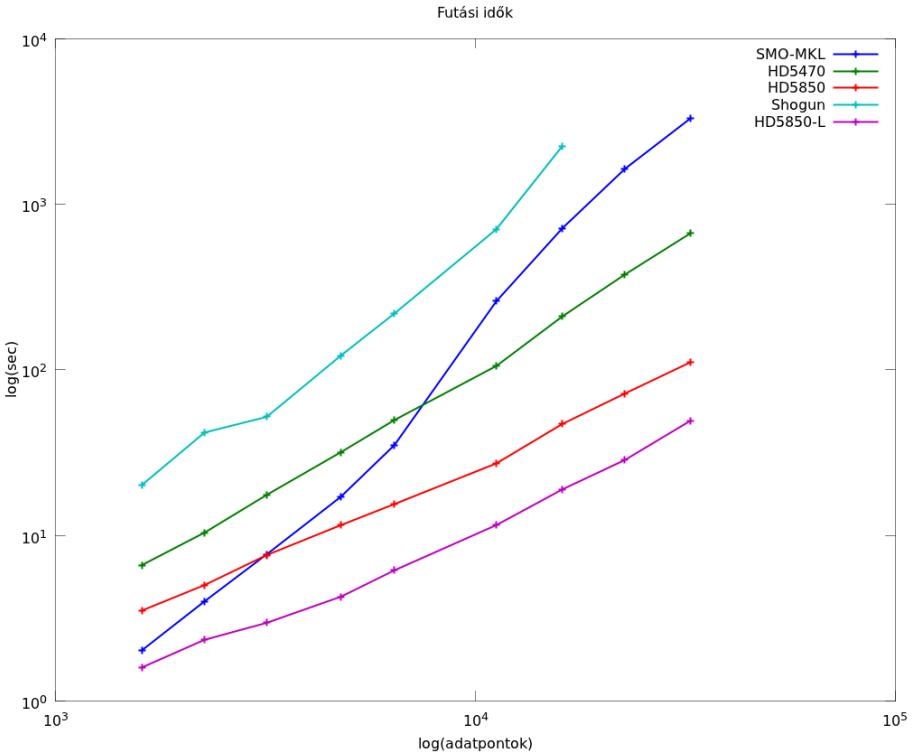
4. táblázat. Mérési eredmények

Adatbázis	Shogun	HD5850	HD5850 (s)	HD5850-λ	HD5850-λ (s)
a1a	84.00%	83.12%	3.51	84.03%	1.59
a2a	84.06%	82.63%	5.00	83.99%	2.34
a3a	83.41%	82.51%	7.59	84.03%	2.96
a4a	83.58%	82.95%	11.51	84.37%	4.26
a5a	83.49%	82.75%	15.44	84.54%	6.15
a6a	83.23%	82.69%	27.19	84.66%	11.53
a7a	83.65%	82.74%	46.98	84.51%	18.92
a8a	-	83.00%	71.80	85.10%	28.52
a9a	-	83.06%	111.10	84.83%	49.30

5. táblázat. Tanulási idő és klasszifikációs pontosság, változásuk a λ paraméter módosításával.

adatokra ötször gyorsabb volt, mint az SMO-MKL, a középkategóriát képviselő HD5850 harmincszoros gyorsulást hozott. E három módszer ugyanahhoz a megoldáshoz konvergált, míg a Shogun kissé eltérően osztotta ki a kernel súlyokat. A legnagyobb súlyt mindenhol a 2. RBF kernel kapta, legalacsonyabbat a lineáris. A Shogun az 1. RBF kernelnek közel akkora súlyt adott, mint a 2.-nak, míg a többi módszer a polinomiálist részesítette előnyben. A klasszifikáció pontosságát a 5. táblázatban írjuk le.

A pontosság csökkenésének oka nagy valószínűséggel a λ paraméter önkényes megválasztása, ahogy azzal az SMO-MKL alkotói is szembesültek [10]. Mivel ezzel kapcsolatban semmilyen információ nem áll rendelkezésre, itt javasolunk egy tapasztalati úton származtatott összefüggést a paraméter „belövésére”. Hangsúlyozzuk, hogy a probléma még közel sincs megoldva, mindenképp



7. ábra. Tanulási idők az adatpontok számának függvényében

további vizsgálatok szükségesek. A javasolt összefüggés:

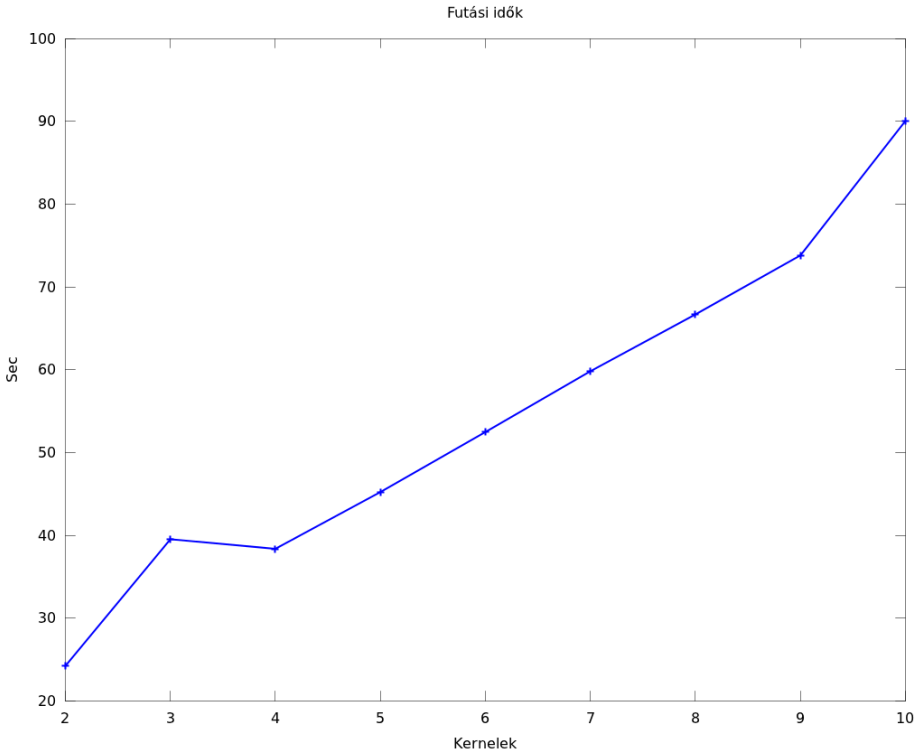
$$\lambda = \frac{e^{k^2}}{l^{ln(k)}}, \tag{24}$$

ahol k a kernelek száma, l pedig a kernelek mérete. Az így számolt paraméterrel nem csak a pontosság emelkedett, hanem az algoritmus is lényegesen gyorsabban konvergált (5. táblázat, 7. ábra).

Egyosztályos MKL esetén hasonló gyorsulást és pontosságot mértünk, így ezeket az adatokat külön nem közöljük. Fontos azonban megvizsgálni, hogyan skálázódik a módszer a kernelek számának emelésével. Ennek méréséhez a legnagyobb méretű tanuló adathalmazt és 2-10 különböző típusú és paraméterezésű kernelt használtunk. A 8. ábrán látszik, hogy a közleményeknek megfelelően a kernelek száma és a futási idők között lineáris az összefüggés [10].

3.6. CPU alapú implementáció

Bár az OpenCL keretrendszer lehetővé teszi, hogy a kódot CPU-n futtassuk, a GPU-ra történő optimalizáció miatt ez lassabb, mint a natív szekvenciális kód. Így a kifejezetten kis méretű tanuló adatmátrixokhoz, illetve a nagy számításigényű permutációs tesztekhez fejlesztettünk egy CPU-s implementációt is. Előbbi esetben a GPU inicializálása dominálná a számítási időt, ami



8. ábra. Skálázódás a kernelek számával

lassú, és az erőforrások felesleges pazarlásával járna; utóbbi esetben célunk a grid-alapú futtatás volt. Mivel az egyes permutációk függetlenek egymástól, az algoritmust lehet nagy példányszámban, párhuzamosan futtatni grid rendszereken.

3.7. Kitekintés, fejlesztési irányok

Rövid távú céljaink közé tartozik a módszer hatékonyságának további fokozása, elsősorban a GPU jobb kihasználásán, valamint a lambda paraméter szerepének mélyebb megértésén keresztül. A profiladatok szerint a futási idő legnagyobb részét a kernel oszlopok kiszámítása képviseli, azon belül is az adatok memóriában való elérése a szűk keresztmetszet. Egy-egy oszlop kiszámításához átlagosan 0,18 ms szükséges a HD5850 típusú kártyán, az legnagyobb Adult táblát használva. Ezalatt összesen $4 \cdot (124 + 32561 \cdot 124 + 32561)$ byte adatot kell átvinni, az effektív sávszélesség tehát 84,24 GB/s-nak adódik. Ez 65,8%-a az elméleti maximum 128 GB/s-nak – világos, hogy e téren van még hova fejlődni. Jelenleg kísérletek folynak az OpenCL függvények további optimalizálására. Előreszámolt kernelek esetén tervezzük az aszinkron memóriatranszferek beépítését (DMA), amint az AMD OpenCL implementációja erre lehetőséget ad.

Hosszabb távú célunk a rendszer kiterjesztése Lp-normalizációra tetszőleges p-érték mellett, amellyel ritka és sűrű kernelkombinációkat egyaránt lehet tanulni; ez alkalmas lehet a kernelek előszűrésére, sok hasonló közül a legmegfelelőbb kiválasztására, illetve hierarchiák építésére.

4. Példák a módszer alkalmazására

Implementációnk lehetséges alkalmazásainak száma igen nagy, a spamszűréstől kezdve az beszéd- és arcfelismerésig, a játékok mesterséges intelligenciájától az orvosi döntéstámogatáson át a pénzügyi-gazdasági problémákig. A fejezet két lehetséges felhasználást mutat be a genomika és a farmakológia területéről. Mindkét esetben a következő kérdésre keressük a választ: hogyan lehet heterogén információforrások alapján, ismert entitásokhoz további relevánsakat találni, illetve az egyes források mekkora szerepet játszanak ebben? Azaz például: hogyan lehet egy adott indikációhoz olyan gyógyszereket találni, amelyeket eddig ott nem alkalmaztak, de más területen már beváltak? A hatóanyagok melyik leírása alapján gondolhatjuk, hogy sikeresek lehetnek az új indikációban is? Azt találtam, hogy a kutatott gén szerepet játszik a leukémia patogenezisében, milyen további géneket érdemes megvizsgálni? Világos, hogy ha a fenti kérdésekre válaszokat kapunk, azzal rengeteg időt és pénzt spórolhatunk meg; a gyógyszeripar jelenlegi helyzetét tekintve az első esetben 4-500 millió dollár, és csaknem 5 évnyi toxikológiai-biztonságossági vizsgálat elkerülése a tét [23].

4.1. Adatbázisok, kernelek származtatása

Egy korábbi munkánkban már foglalkoztunk a gyógyszer-újrapozicionálás, azaz az új indikációban való alkalmazás lehetőségének kérdéskörével [24, 25]. Ebben minden hatóanyaghoz előállítottunk három leírást (kémiai, mellékhatásprofil, szövegbányászati), majd egy közös logisztikus regressziós modell alapján próbáltuk megjósolni, hogy van-e a gyógyszereknek közös célfehérjéje. Most az ettől eltérő MKL megközelítést vizsgáljuk. Az információforrások, és a kernelek számítása során felhasznált hasonlóságmértékek:

- Kémiai kernel:
  - MACCS leírók és Tanimoto-hasonlóság.
- Mellékhatásprofil alapú kernelek:
  - Mellékhatás frekvencia adatok az online elérhető SIDER adatbázisból [26], cos-hasonlóság.
  - A Dailymed betegtájékoztatóiból kivonatolt placebo-kontrollált mellékhatásvizsgálatok [27] alapján készült leírás és cos-hasonlóság.
  - Mellékhatások bináris előfordulása a betegtájékoztatók megfelelő szekciójában, tf-idf súlyozás, cos-hasonlóság.
- PubMed ko-okkurencia alapú kernel:
  - Mellékhatások és hatóanyagok említése PubMed absztraktokban, kölcsönös információk származtatása és cos-hasonlóság.

A mellékhatásokat jelölő kifejezések listáját a MedDRA és UMLS ontológiák alapján állítottuk elő [28, 29]. A genomikai vonalon a Leuveni Katolikus Egyetemen fejlesztett Endavour rendszer kerneleit használhattuk [5]. A rendelkezésünkre bocsátott kernelek – gén-gén hasonlóságok – forrásai:

- DNS- és protein-szekvencia adatok: EnsEMBL EST (cDNS) [30], Motif (protein) [31]
- Funkció: Gene Ontology (GO) [32]
- Biológiai útvonalak: Kegg [33]
- Génexpressziós adatok: Son [34], Su [35]
- Szövegbányászati adatok

4.2. Néhány eredmény

A farmakológiai mérések során tanuló halmazként (a továbbiakban: lekérdezés) ismert gyógyszercsoportokat használtunk, majd megvizsgáltuk, hogy a kapott sorrendben vajon előre kerülnek-e olyan hatóanyagok, amelyek jelenlegi tudásunk szerint nem tartoznak a csoportba. Ezen szereknél ugyanis felmerül a lehetőség, hogy újrapozicionálhatók, azaz alkalmazhatók a lekérdezéshez tartozó indikációban. Ha találtunk ilyen szert, akkor irodalmi adatokat kerestünk a feltételezés megerősítésére. Azt találtuk, hogy a PubMed ko-okkurencia alapú kernel nehezíti ezen hatóanyagok felderítését, mivel egyes mellékhatások (pl. benignus prostata hyperplasia – phenylephrine) önálló betegségként, indikációként is szerepelhetnek az absztraktokban. Így a kernel az adott indikációban már alkalmazott, lekérdezésben nem szereplő gyógyszereket sorolja előre, kiszorítva az újrapozicionálhatókat; ezen megfontolásból a PubMed kernelt végül nem használtuk fel.

Elsőként három kalcium-csatorna blokkoló vérnyomáscsökkentőt alkalmaztunk lekérdezésként, név szerint: amlodipine, felodipine, isradipine. A kapott eredményeket a 4.2. táblázat ismerteti.

Röviden magyarázzuk a sorrendet. Az első három helyet nyilvánvalóan a lekérdezés foglalja el, őket két másik vérnyomáscsökkentő hatású szer, a doxazosin és a bisoprolol követi. Megjegyzendő, hogy ezek más hatásmechanizmussal működnek; előbbi α1-receptor, utóbbi β1-receptor blokkoló (lényegében az adrenalin/noradrenalin vérnyomásemelő hatását gátolják – többek között). A nifedipin szintén kalcium-csatorna gátló. A rákövetkező három találat sokkal érdekesebb. A thalidomide (Contergan) tragikus története széles körben ismert, az azonban kevésbé, hogy a szer számos más indikációban megállta a helyét (lepra, egyes rosszindulatú daganatok, myeloma multiplex). Értágító és lehetséges kalcium-csatorna blokkoló aktivitását egy 2010. októberi közlemény írja le [36]. Az imatinib a krónikus myeloid leukémia egy gyakori fajtájának „csodaszere", amely eredeti hatásmechanizmusától független módon gátolja a kalcium-áramlást [37]. A vardenafil a sildenafil (Viagra) rokona; értágító és kalcium-csatorna blokkoló hatása 2008-ban

Kernel súlyok (1-re normálva)					
MACCS	0.299497	MACCS	0.273561	MACCS	0.365815
SIDER	0.204384	SIDER	0.257818	SIDER	0.172973
DailyMed	0.211092	DailyMed	0.167731	DailyMed	0.204612
Tf-idf	0.285027	Tf-idf	0.300889	Tf-idf	0.2566
Sorrend					
Amlodipine	0.452086	Didanosine	0.477122	Cefalexin	0.388889
Felodipine	0.452058	Lamivudine	0.477122	Cefadroxil	0.388846
Isradipine	0.452058	Abacavir	0.477099	Cefazolin	0.388846
Doxazosin	0.295288	Stavudine	0.37893	Cefprozil	0.251269
Bisoprolol	0.290679	Zidovudine	0.351709	Cefotetan	0.23571
Nifedipine	0.283446	Emtricitabine	0.326662	Cefoxitin	0.232604
Thalidomide	0.264182	Amprenavir	0.322583	Ceftizoxime	0.23082
Imatinib	0.262527	Clofarabine	0.31842	...	...
Vardenafil	0.262093	Tipranavir	0.313632	Oxacillin	0.184188
Fosinopril	0.26201	Gemcitabine	0.31104	Dicloxacillin	0.178902
Meloxicam	0.261677	Erlotinib	0.299769	Piperacillin	0.178123
Temazepam	0.261436	Aciclovir	0.293244	Amoxicillin	0.177984
Nimodipine	0.261159	Posaconazole	0.281143	Benzylpenicillin	0.174308

6. táblázat. Prioritizálási eredmények. Az első oszlopban a kalciumcsatorna gátlók, a másodikban az antivirális hatóanyagok, a harmadikban a cephalosporinok láthatók. Fent szerepelnek a kernel súlyok, alul a kiszámított sorrend, ahol az első három hatóanyag alkotta a lekérdezést.

vált ismertté [38]. A közlemény külön kiemeli, hogy a sildenafil és a tadalafil nem mutatja ezt a tulajdonságot, ami összhangban van az eredményeinkkel. A sort a szintén vérnyomáscsökkentő, ACE-gátló fosinopril folytatja, majd két nehezebben magyarázható találat következik. A meloxicam nem-szteroid gyulladásgátló vegyület, amelynek a kalcium-háztartással való viszonya nem tisztázott, de egyes esetekben képes volt az áramlást blokkolni [39]. A benzodiazepin szedatívumokhoz tartozó temazepam vélhetően szintén vérnyomáscsökkentő hatása miatt került ide. A sort a lekérdezési csoporthoz tartozó nimodipin zárja. Itt jegyezzük meg, hogy utóbbi kifejezetten az agyi erekre hat, ez eredményezheti a lekérdezéstől kissé eltérő mellékhatásprofilt, és ennélfogva az alacsonyabb rangot.

Második példánkban a lekérdezést (retro-)vírusellenes gyógyszerek alkották, így a listában is főleg ilyeneket látunk. Bár nem antivirális, nem meglepő az antitumor ágens clofarabine jelenléte, ugyanis nukleozid-analógként számos antivirális szer szerkezeti rokona, csakúgy mint a gemcitabine, amely viszont bizonyítottan mutatja a tulajdonságot [40]. Igazi meglepetés azonban az erlotinib (és néhány hellyel később az imatinib), amelyek teljesen eltérő hatásmechanizmusú antitumor hatóanyagok; egy 2010-es közlemény mégis azt találta, hogy rendelkeznek vírusellenes hatással is [41]. Az utolsóként említett posaconazole gombaölő szer, antivirális aktivitása nem ismert.

Harmadik példánkkal azt demonstráljuk, hogyan lehet a források súlyozása alapján következtetéseket levonni, egyúttal példát mutatunk a korábban említett lekérdezés-specifikus, „intelligens" fúzióra is. Ha lekérdezésként szándékosan igen hasonló, béta-laktám szerkezetű antibiotikumokat (cephalosporinokat) adunk meg, akkor a legnagyobb súlyt a kémiai kernel kapja, a listában pedig előre kerülnek a cephalosporinok, majd a penicillinek, carbapenemek és az aztreonam – azaz az összes béta-laktám csoport. Ilyen eredményt látva megkockáztathatjuk, hogy a magasabb rangok hátterében közös célfehérje/kötőhely áll. A mellékhatások magas súlya esetén valószínűbb lenne a közös biológiai útvonal, amelynek különböző pontjain való beavatkozás hasonló mellékhatást eredményez.

A génprioritizálás bemutatására egy protoonkogénekből és tumorszuppresszor génekből álló tanulóhalmazt választottunk. E gének közös tulajdonsága, hogy egészséges sejtekben az osztódással kapcsolatos funkciót töltenek be, károsodásuk – pl. fokozott expresszió, mutációk, kromoszóma-transzlokációk – esetén a korlátlan osztódás tumorképződéshez vezet. A lekérdezés génjei:

- **TP53.** Igen intenzíven kutatott protoonkogén. Számos rosszindulatú daganat kialakulásában, öröklött daganatszindrómában kulcsszerepet tölt be.
- **CDK4.** A sejtciklus fontos szabályozója, sok daganattípussal kapcsolatba hozták.
- **KRAS.** Protoonkogén, elsősorban gyomor-, vastagbél- és tüdőrákban találták meg károsodását.

Kernel	Súly
EST	0.13
GO	0.07
Kegg	0.35
Motif	0.07
Son	0.15
Su	0.12
Text	0.10

7. táblázat. Kernel súlyok

- **NF1.** Tumorszuppresszor gén, károsodása a neurofibromatosis nevű öröklött daganatszindróma oka.
- **RB1.** Tumorszuppresszor gén, károsodása a gyermekkorban előforduló retinoblastoma oka.

A prioritizálás során a 7. táblázatban látható kernel súlyokat kaptuk. Az előre sorolt gének az alábbi kategóriákra oszthatók (a teljesség igénye nélkül):

- Protoonkogének: pl. BRAF, HRAS, E2F1, MDM2
- Tumorszuppresszor gének: pl. E2F3, CDKN1A, CDKN2A, SOS1
- Apoptózist (programozott sejthalált), túlélést befolyásoló gének: pl. AKT1-3, PIK3R2,3,5, IKBKG
- Osztódási szignált indító receptorok/adapterek: pl. EGFR, GRB2, PDGFRA
- Osztódási kaszkádban résztvevő Ser/Thr kinázok: pl. MAPK1-3
- Növekedési faktorok: pl. EGF, TGFB1-3, PDGFB
- Transzkripciós faktorok: pl. NFKB, RELA, SMAD4

Látható, hogy a rendszer igen sok, a sejtnövekedést, osztódást és sejthalált szabályozó gént talált, és ehhez elsősorban az útvonal-információkra és az expressziós adatokra támaszkodott.

4.3. További lehetőségek

A hardverfejlődési trendek alapján a GPU alapú implementáció által jelenleg tapasztalt gyorsulás minden bizonnyal tovább fog növekedni. Ez lehetőséget ad mind a dimenzióban, kernelszámban, és kimeneti osztályok számában vagy azok kombinációjában a felskálázásra. A dimenziószám növelésére példa a prioritizálás „high-throughput screening” jellegű felhasználása a kemoinformatikában, amikor egy potenciális hatóanyagkönyvtárt vizsgálunk át de novo gyógyszerkutatásban. A felskálázás egy másik rendkívül fontos aspektusa a lekérdezések számának

növelése, például robosztusság vizsgálatokban. Egy GPU klaszter felhasználásával pedig jelenleg elérhetetlennek tűnő alkalmazások is megjelenhetnek, mint például a kernelfúzió használata tanulási algoritmusokba integráltan. Ezek eddig klaszterezésben jelentek meg [42], viszont a sebesség növekedésével a kernel alapú fúzió komplex, strukturált modellek tanulását is segítheti mind informatív, lokális a priori eloszlások futás közbeni számításával, illetve informatív hasznosságfüggvények konstruálásával. Ezen négy kutatási irány vizsgálata a BME MIT Bioinformatikai munkacsoport, az SE SzVI, és a leueni egyetem (KUL) ESAT Bioinformatikai csoport együttműködésében meg is kezdődött a TDK-ban ismertett implementációkra is támaszkodva.

4.4. Összefoglalás

E munkában elsődleges célunk a Multiple Kernel Learning megközelítés GPU-ra történő megvalósítása, valamint bioinformatikai problémákra való alkalmazhatóságának vizsgálata volt. Ennek során az alábbi eredményeket értük el:

- Implementáltunk és optimalizáltunk egy új matematikai megközelítésen alapuló algoritmust, valamint kidolgoztunk hozzá egy könnyen kezelhető és bővíthető felhasználóbarát felületet.
- Megvizsgáltuk az implementáció hatékonyságát és összehasonlítottuk a korábbi eszközökkel; itt csaknem harmincszoros gyorsulást mértünk a referencia algoritmushoz képest, és legalább két nagyságrend gyorsulást az egyik legelterjedtebb eszközhöz képest.
- Valós genomikai és farmakológiai feladatokban értékeltük ki a módszert, ahol irodalmilag megerősített eredményeket kaptunk.

Megállapítjuk, hogy bár az eljárás még több ponton nyitott, bíztató eredményeket szolgáltat, így a jövőben további bővítését és kiterjesztését tervezzük. A fent felsoroltak teljes egészében a szerző munkájának eredményei, a farmakológiai adatok egy most futó párhuzamos kutatásból, a genomikai adatok Yves Moreau professzor leuveni kutatócsoportjától származnak.

5. Köszönetnyilvánítás

A munkát a NKTH TECH 08-A1/2-2008-0120 (Genagrid) támogatta.

A szerző köszönetet mond a következő személyeknek, akik nélkül e dolgozat nem jöhetett volna létre:

- Dr. Antal Péternek a gépi tanulás, adatfúziós módszerek területén nyújtott segítségéért, támogatásáért,
- Arany Ádámnak a szerves kémia területén nyújtott segítségéért és hasznos észrevételeiért,
- Prof. Dr. Mátyus Péternek és Dr. Balogh Balázsnak a farmakológiai adatok biztosításáért,
- Yves Moreau professzornak és Léon-Charles Tranchevent-nek a genetikai adatok biztosításáért,
- S. V. N. Vishwanathan professzornak az algoritmus egyes részleteinek tisztázásáért.

Külön köszönet illeti Prof. Dr. Szél Ágostont és Prof. Dr. Sándor Józsefet támogatásukért.

Hivatkozások

[1] Advanced Micro Devices Inc. *AMD Accelerated Parallel Processing OpenCL Programming Guide*, 2010.

[2] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[3] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, 6:1889–1918, December 2005.

[4] G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing*, 2004.

[5] S. Aerts, D. Lambrechts, S. Maity, P. Van Loo, B. Coessens, F. De Smet, L. C. Tranchevent, B. De Moor, P. Marynen, B. Hassan, P. Carmeliet, and Y. Moreau. Gene prioritization through genomic data fusion. *Nat. Biotechnol.*, 24:537–544, May 2006.

[6] T. De Bie, L. C. Tranchevent, L. M. van Oeffelen, and Y. Moreau. Kernel-based data fusion for gene prioritization. *Bioinformatics*, 23:i125–132, Jul 2007.

[7] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, November 2008.

[8] Marius Kloft, Ulf Brefeld, Soeren Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and Accurate Lp-Norm Multiple Kernel Learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 997–1005. 2009.

[9] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 6–, New York, NY, USA, 2004. ACM.

[10] S. V. N. Vishwanathan, Z. Sun, N. Theera-Ampornpunt, and M. Varma. Multiple kernel learning and the SMO algorithm. December 2010.

[11] Márta Altrichter, Gábor Horváth, Béla Pataki, György Strausz, Gábor Takács, and József Valyon. *Neurális hálózatok*. Panem, 2006. (in Hungarian).

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[14] S. Yu, T. Falck, A. Daemen, L. C. Tranchevent, J. A. Suykens, B. De Moor, and Y. Moreau. L2-norm multiple kernel learning and its application to biomedical data fusion. *BMC Bioinformatics*, 11:309, 2010.

[15] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13:1443–1471, July 2001.

[16] Khronos Group. *The OpenCL Specification*, September 2010.

[17] Andrei Alexandrescu. *Modern C++ design: generic programming and design patterns applied*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[18] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30, January 1998.

[19] Zhen-Dong Zhao, Lei Yuan, Yu-Xuan Wang, F. S. Bao, Shun-Yi Zhang, and Yan-Fei Sun. A Novel Model of Working Set Selection for SMO Decomposition Methods. *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, 2:283–290, 2007.

[20] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[21] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, June 2010.

[22] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[23] Christopher P. Adams and Van V. Brantner. Estimating the cost of new drug development: is it really 802 million dollars? *Health affairs (Project Hope)*, 25(2):420–428, March 2006.

[24] Ádám Arany and Bence Bolgár. Gyógyszerhatóanyagok mellékhatásprofiljának és kémiai szerkezetének együttes vizsgálata. *BME TDK*, 2010.

[25] M. Campillos, M. Kuhn, A. C. Gavin, L. J. Jensen, and P. Bork. Drug target identification using side-effect similarity. *Science*, 321:263–266, Jul 2008.

[26] M. Kuhn, M. Campillos, I. Letunic, L. J. Jensen, and P. Bork. A side effect resource to capture phenotypic effects of drugs. *Mol. Syst. Biol.*, 6:343, 2010.

[27] DailyMed. http://dailymed.nlm.nih.gov/dailymed/about.cfm.

[28] Medical Dictionary for Regulatory Activities. http://www.meddramsso.com.

[29] Olivier Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl 1):D267–D270, January 2004.

[30] P. Flicek, M. R. Amode, D. Barrell, K. Beal, S. Brent, Y. Chen, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gordon, M. Hendrix, T. Hourlier, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, P. Larsson, I. Longden, W. McLaren, B. Overduin, B. Pritchard, H. S. Riat, D. Rios, G. R. Ritchie, M. Ruffier, M. Schuster, D. Sobral, G. Spudich, Y. A. Tang, S. Trevanion, J. Vandrovcova, A. J. Vilella, S. White, S. P. Wilder, A. Zadissa, J. Zamora, B. L. Aken, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, T. J. Hubbard, A. Parker, G. Proctor, J. Vogel, and S. M. Searle. Ensembl 2011. *Nucleic Acids Res.*, 39:D800–806, Jan 2011.

[31] http://www.genome.jp/tools/motif/.

[32] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.*, 25:25–29, May 2000.

[33] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Res.*, 38:D355–360, Jan 2010.

[34] C. G. Son, S. Bilke, S. Davis, B. T. Greer, J. S. Wei, C. C. Whiteford, Q. R. Chen, N. Cenacchi, and J. Khan. Database of mRNA gene expression profiles of multiple human organs. *Genome Res.*, 15:443–450, Mar 2005.

[35] A. I. Su, M. P. Cooke, K. A. Ching, Y. Hakak, J. R. Walker, T. Wiltshire, A. P. Orth, R. G. Vega, L. M. Sapinoso, A. Moqrich, A. Patapoutian, G. M. Hampton, P. G. Schultz, and J. B. Hogenesch. Large-scale analysis of the human and mouse transcriptomes. *Proc. Natl. Acad. Sci. U.S.A.*, 99:4465–4470, Apr 2002.

[36] S. W. Seto, S. Bexis, P. A. McCormick, and J. R. Docherty. Actions of thalidomide in producing vascular relaxations. *Eur. J. Pharmacol.*, 644:113–119, Oct 2010.

[37] M. Cataldi, A. Gaudino, V. Lariccia, M. Russo, S. Amoroso, G. di Renzo, and L. Annunziato. Imatinib-mesylate blocks recombinant T-type calcium channels expressed in human embryonic kidney-293 cells by a protein tyrosine kinase-independent mechanism. *J. Pharmacol. Exp. Ther.*, 309:208–215, Apr 2004.

[38] H. A. Toque, C. E. Teixeira, F. B. Priviero, R. P. Morganti, E. Antunes, and G. De Nucci. Vardenafil, but not sildenafil or tadalafil, has calcium-channel blocking activity in rabbit isolated pulmonary artery and human washed platelets. *Br. J. Pharmacol.*, 154:787–796, Jun 2008.

[39] A. Romare and C. E. Lundholm. Cadmium-induced calcium release and prostaglandin E2 production in neonatal mouse calvaria are dependent on cox-2 induction and protein kinase C activation. *Arch. Toxicol.*, 73:223–228, 1999.

[40] C. L. Clouser, S. E. Patterson, and L. M. Mansky. Exploiting drug repositioning for discovery of a novel HIV combination therapy. *J. Virol.*, 84:9301–9309, Sep 2010.

[41] P. S. Randhawa, N. A. Farasati, Y. Huang, M. Y. Mapara, and R. Shapiro. Viral drug sensitivity testing using quantitative PCR: effect of tyrosine kinase inhibitors on polyomavirus BK replication. *Am. J. Clin. Pathol.*, 134:916–920, Dec 2010.

[42] S. Yu, X. Liu, L. C. Tranchevent, W. Glanzel, J. A. Suykens, B. De Moor, and Y. Moreau. Optimized data fusion for K-means Laplacian clustering. *Bioinformatics*, 27:118–126, Jan 2011.