

TDK-dolgozat

Grósz Tamás

2012

Mély neuronhálók az akusztikus modellezésben

**Szegedi Tudományegyetem
Természettudományi és Informatikai Kar**

*Grósz Tamás
II. évf. programtervező informatikus MSc*

*Témavezető: Dr. Tóth László
Tudományos főmunkatárs
MTA-SZTE Mesterséges Intelligencia Kutatócsoport*

Tartalomjegyzék

1. Bevezetés.....	3
2. Mély neuronhálók.....	5
2.1 Korlátos Boltzmann Gép	6
2.3 Kontrasztív divergencia (CD).....	8
2.4 A CD algoritmus igazolása.....	9
3. Implementáció.....	13
3.1 Előtanítás.....	13
3.2 A backpropagation-algoritmus.....	14
4. Adatbázisok.....	15
4.1 TIMIT adatbázis.....	15
4.2 Híradós adatbázis.....	15
4.3 Hangoskönyv.....	16
4.4 Előfeldolgozás.....	16
5. Eredmények.....	17
5.1 TIMIT.....	17
5.2 Híradós adatbázis.....	18
5.3 Hangoskönyv.....	18
6. Összefoglalás.....	19
Irodalomjegyzék.....	20

1. Bevezetés

Az elmúlt néhány évtizedben a mesterséges neuronhálók számos változatát kipróbálták a beszédfelismerésben, általános elismertséget azonban csak a többrétegű perceptron-hálózatokra (MLP) épülő ún. hibrid HMM/ANN modellnek sikerült elérnie. Bár kisebb felismerési feladatokon a neuronhálós modellek jobb eredményt adnak, mint a sztenderd rejtett Markov-modell (HMM), használatuk mégsem terjedt el általánosan, részben mivel technikailag nehezebb a használatuk, másrészt mivel nagyobb adatbázisokon az előnyük elvész, köszönhetően a HMM-ekhez kifejlesztett trifón modellezési és diszkriminatív tanítási technikáknak. A hibrid modell a felsorolt okok miatt ugyan versenyképes maradt, de igazi áttörést nem hozó alternatív státuszba került.

Mindez megváltozni látszik azonban az ún. mély neuronhálók (deep neural nets) megjelenésével. A mély neuronhálót (és tanítási algoritmusát) 2006-ban publikálták először [1], a kezdeti cikkben képi alakfelismerési tesztek használtak demonstrációként. Az első cikk a mély neuronhálók beszédfelismerési alkalmazásáról 2009-ben jelent meg, melyben sikerült megdönteni az összes korábban elért felismerési pontosságot [2] egy közismert tesztfeladaton, a TIMIT adatbázison. Az azóta eltelt néhány évben tovább javították a modellt és még jobb eredményeket sikerült elérniük [3] a mély neuronhálókkal.

A mély neuronhálók két fontos dologban térnek el a megszokott többrétegű perceptronoktól, mégpedig hálózati struktúrájukban és tanító algoritmusukban. Hagyományos hálózatok esetén egy vagy maximum két rejtett réteget szokás használni, a hálózat osztályozási pontosságát pedig a rejtett neuronok számának növelésével próbáljuk növelni. Ezzel szemben a mély neuronháló nevét onnan kapta, hogy a rejtett rétegek száma jóval nagyobb: általában 4-5, de némely cikkekben akár kilencig is elmennek. A matematikai érvek és az empirikus kísérletek is amellettszólnak, hogy – adott neuronszám mellett – a több réteg komplexebb reprezentációt tesz lehetővé. Ekkor azonban újabb probléma jelentkezik, mégpedig hogy a neuronhálók hagyományos tanítóalgoritmusai a backpropagation algoritmus egyre nehezebben talál optimum-közeli paramétereket. Ebben hozott áttörést Hinton cikke[1], amely lényegében egy

felügyelet nélküli előtanítási algoritmust javasolt, amely megkönnyíti a backpropagation dolgát, így előtanítás után lényegesen jobb eredményt képes produkálni.

Dolgozatomban a mély neuronhálókra épülő akusztikus modelleket beszédhang-felismerési teszteken értékelem ki és vetem össze az eddigi eredményekkel. Részletesen bemutatom a mély neuronhálók tanítási algoritmusát, amely két részből áll: egy felügyelet nélküli előtanításból és egy finomhangolásból.

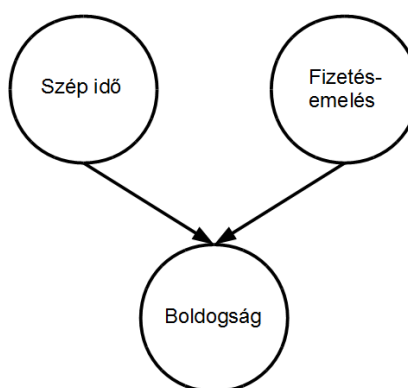
A tanuló adatbázisok nagysága és a tanítási algoritmus nagy időigénye miatt a saját implementációt GPU-n futtathatóvá tettem, ehhez CUDA-t használtam.

Az eredmények fejezetben elsősorban a TIMIT adatbázison elért eredményeimet közlöm, mert a legtöbb cikk is ezt az adatbázist használja, így összevethetőek az eredményeimmel. Továbbá az akusztikus modelleket kiértékelem még két magyar nyelvű adatbázison (egy hangoskönyv és egy híradás adatbázis) is.

2. Mély neuronhálók

Osztályozási feladatok megoldásához régóta ismert és használt technika a neuronhálók használata. A neuronhálók rétegekből épülnek fel, a rétegek pedig neuronokból állnak. A legelső réteget látható rétegnek szokás nevezni (a háló ezen rétegen keresztül kapja a bemenetet) a többi réteget pedig rejtett rétegnek. A rejtett rétegek neuronjai kapcsolatban állnak az alattuk lévő réteg neuronjaival és (a legfelső réteg kivételével) a felettük lévő réteg neuronjaival, a kapcsolatokhoz pedig súlyok tartoznak. A háló kiértékelésekor alulról felfelé kiszámítjuk a neuronok állapotát a súlyok és a neuronokhoz tartozó bias-ok alapján, majd a legfelső réteg alapján meghatározzuk a háló által az inputhoz rendelt címkét. A hagyományos neuronhálók tanítása backpropagation-algoritmussal (ez egy gradiens alapú módszer) történik, amelynek során fentről lefelé propagáljuk a hibát és az alapján számítjuk ki a súlyok gradiensét.

A hagyományos neuronhálók tanítása sok rejtett réteg esetén problémássá válik, mivel nehéz következtetni a posteriori eloszlásra a rejtett változók felett egy megadott adatvektor esetén az ún. megmagyarázás (explaining away) hatás miatt.



ábra 1: explaining away hatás

Az explaining away megértéséhez tekintsük a következő példát: legyen 3 neuronból álló hálózatunk az 1. ábrán látható módon. Ekkor a boldogság okai között szerepel a szép idő és a fizetésemelés is, viszont a szép idő független a fizetésemeléstől. Amennyiben megfigyeljük, hogy boldogok vagyunk, akkor egyenlő eséllyel teljesül, hogy szép az idő vagy fizetésemelést kaptunk. Amennyiben tudjuk, hogy boldogok vagyunk és azt is tudjuk, hogy szép az idő, akkor kevésbé tartjuk

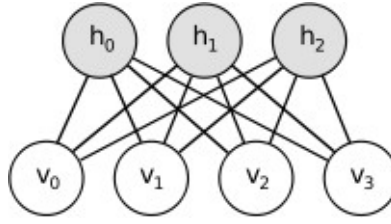
valószínűnek, hogy fizetésemelést kaptunk, lényegében a szép idő „megmagyarázza” a boldogságunkat és csökkenti a fizetésemelés valószínűségét. A fent leírt esetben a két független változó függetlensége már nem áll fent, mivel az egyik együttes megfigyelése az okozattal csökkenti a másik valószínűségét.

A mély neuronhálók (deep belief nets) lényegében olyan speciális neuronhálók, amelyekre a következő fontos tulajdonságok teljesülnek:

- A két felső réteg között szimmetrikus irányítatlan kapcsolat van, így egy asszociatív memóriát alkotnak.
- Létezik egy hatékony, rétegenként haladó tanuló algoritmus, amely meghatározza minden rétegre, hogy hogyan függnek a réteg neuronjai a felette lévő réteg neuronjaitól.
- A legalsó réteg az input. Felügyelt tanulás esetén a legfelső réteghez kell csatolni egy új réteget, amely a címkéket fogja reprezentálni.
- Lehetőség van arra, hogy „megkérdezzük” a hálózatot, hogy milyen adatban „hisz”, ehhez az inverz irányú kiértékelést használhatjuk, tehát egy címkét megadva megtudhatjuk, hogy ahhoz milyen inputot várna a hálózat.

2.1 Korlátos Boltzmann Gép

Egy korlátos Boltzmann gép (RBM) lényegében egy adott típusú Markov véletlen mező (MRF), amely egy sztochasztikus rejtett rétegből és egy sztochasztikus látható rétegből áll. A két rétegben található neuronoknak nem kell Bernoulli véletlen változóknak lenniük, bármilyen exponenciális eloszlást követhetnek. A rejtett réteg és a látható réteg összes neuronja össze van kötve, a háló egy páros gráfnak tekinthető. A korlátos jelző abból származik, hogy a rétegen belüli neuronok nem lehetnek összekötve. Az egyes kapcsolatokhoz tartozó súlyok és a neuronokhoz tartozó bias-ok egy véletlen eloszlást definiálnak a látható réteg neuronjainak állapotait tartalmazó $\mathbf{v}$ vektorok felett, egy energiafüggvény segítségével.



ábra 2: példa RBM-re (3 rejtett és 4 látható egységgel)

Az energiafüggvény $(\mathbf{v}, \mathbf{h})$ együttes előfordulására:

$$E(\mathbf{v}, \mathbf{h}, \theta) = - \sum_{i=1}^V \sum_{j=1}^H w_{ij} v_i h_j - \sum_{i=1}^V b_i v_i - \sum_{j=1}^H a_j h_j, \quad (1)$$

ahol $\theta = (\mathbf{w}, \mathbf{b}, \mathbf{a})$ és w_{ij} reprezentálja az i . látható neuron és j . rejtett neuron szimmetrikus kapcsolatának súlyát, b_i , illetve a_j pedig a bias-okat. V és H a látható és rejtett egységek/neuronok száma. A modell által a $\mathbf{v}$ látható vektorhoz rendelt valószínűség:

$$p(\mathbf{v}, \theta) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}} \sum_{\mathbf{h}} e^{-E(\mathbf{u}, \mathbf{h})}}, \quad (2)$$

ahol $\mathbf{u}$ eleme az input vektoroknak, $\mathbf{h}$ pedig a rejtett réteg állapotvektorainak. Mivel a korlátos Boltzmann gépben nem engedélyezett rejtett-rejtett és látható-látható kapcsolat, ezért a $p(\mathbf{v}|\mathbf{h})$ -t és a $p(\mathbf{h}|\mathbf{v})$ -t a következő módon definiálhatjuk:

$$\begin{aligned} p(h_j=1|\mathbf{v}, \theta) &= \sigma\left(\sum_{i=1}^V w_{ij} v_i + a_j\right) \\ p(v_i=1|\mathbf{h}, \theta) &= \sigma\left(\sum_{j=1}^H w_{ij} h_j + b_i\right), \end{aligned} \quad (3)$$

ahol $\sigma(\mathbf{x})$ a szigmoid függvény¹.

Speciális változata az RBM-eknek az ún Gaussian-Bernoulli RBM, amely esetén a látható réteg neuronjai nem binárisak, hanem valós értékűek. Az energiafüggvényük a következőképpen módosul:

$$E(\mathbf{v}, \mathbf{h}, \theta) = \sum_{i=1}^V \frac{(v_i - b_i)^2}{2} - \sum_{i=1}^V \sum_{j=1}^H w_{ij} v_i h_j - \sum_{j=1}^H a_j h_j, \quad (4)$$

és a modell által a $\mathbf{v}$ látható vektorhoz rendelt valószínűség:

¹ $\sigma(\mathbf{x}) = 1/(1+\exp(-\mathbf{x}))$

$$p(v_i|h,\theta)=N\left(b_i+\sum_{j=1}^H w_{ij}h_j,1\right), \quad (5)$$

ahol $N(\mu, \sigma)$ a μ várható értékű és σ varianciájú gauss eloszlás.

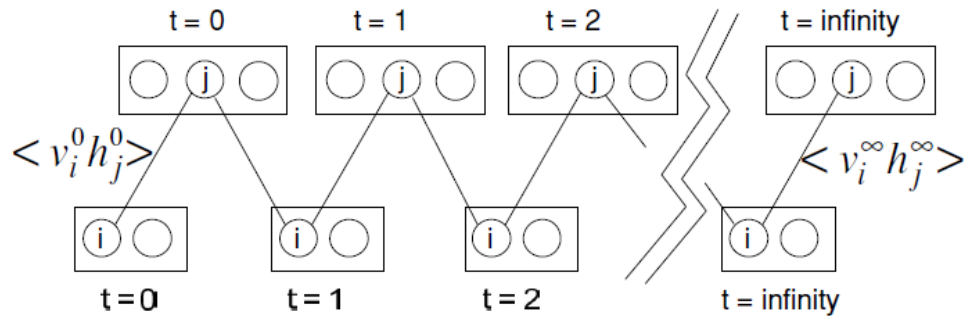
A pontos maximum likelihood tanulás alkalmatlan nagy méretű RBM esetén, ugyanis a derivált számításának időigénye exponenciálisan nő a hálózat méretével. A hatékony megoldást egy közelítő tanítóalgoritmus, az ún. összehasonlító divergencia (Contrastive Divergence CD) [1] biztosítja. Hatékony tanítóalgoritmusának köszönhetően az RBM tökéletesen alkalmas arra, hogy a mély neuronhálók építő eleme legyen.

2.3 Kontrasztív divergencia (CD)

Hinton 2006-os cikkében [1] javasolt egy tanító algoritmust a korlátos Boltzmann gépekhez, amelyet kontrasztív divergenciának (Contrastive Divergence) nevezett el. A javasolt módszer során a súlyok frissítési szabálya:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{\text{input}} - \langle v_i h_j \rangle_{\text{rekonstrukció}} \quad (6)$$

A (6) jobb oldalán található első tag az i . látható és j . rejtett egység korrelációja, bináris esetben annak gyakorisága, hogy mindkét neuron egyszerre aktív, ennek meghatározásához a látható rétegen az adatbázisból vett input vektorok szerepelnek, a rejtett réteg állapotát pedig a (3) alapján számítjuk. A második tag jelentése hasonló, csak ekkor rekonstrukciós állapotokat használunk.



ábra 3: Rekonstrukciós lánc

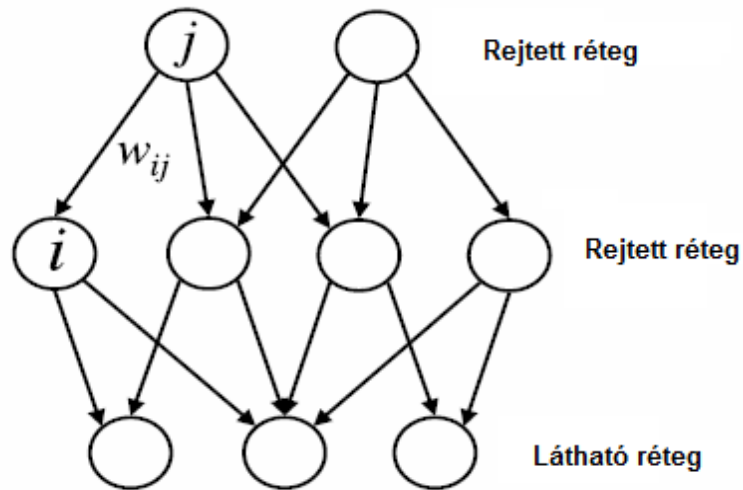
Rekonstrukció alatt a következőt kell érteni: miután az input alapján meghatároztuk a rejtett réteg állapotait, a (3) felhasználásával tudjuk (a rejtett réteg alapján) a látható réteg állapotait kiszámolni, ezután az így kapott látható réteghez generáljuk a rejtett réteget. A rekonstrukciót tetszőleges alkalommal megismételhetjük a 3. ábrán látható módon. Mivel a rekonstrukciós lépések

rendkívül időigényesek, ezért általában csak k db rekonstrukciót végzünk. A CD mohó algoritmus $k = 1$ rekonstrukciót végez, és az alapján tanulja a súlyokat, általánosan ez a módszer terjedt el viszonylag kis időigénye és rendkívül jó teljesítménye miatt. A mohó előtanítás során a súlyok frissítését a következő módon végezzük:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_1 \quad (7)$$

2.4 A CD algoritmus igazolása

Míg irányítatlan esetben könnyen el lehet mondani a CD algoritmust, bizonyítani rendkívül nehéz, irányítatlan esetben pedig ez fordítva igaz, azaz könnyebb bizonyítani, de az algoritmus bonyolultabbá válik.



ábra 4: Egyszerű irányított neuronháló

Tekintsünk egy egyszerű irányított hálót (4. ábra), ekkor a látható réteg generálása egyszerű: a legfelső réteg állapotainak meghatározása után a $(k-1)$ -edik réteg generálása a k . réteg állapotai alapján párhuzamosan történik (3) felhasználásával:

$$p\left(h_i^{(k-1)} = 1 \mid h^{(k)}, W^{(k)}\right) = \sigma\left(b_i^{(k-1)} + \sum_j w_{ij} h_j^{(k)}\right), \quad (8)$$

ahol $\mathbf{h}^{(k)}$ a k -edik réteg állapotvektora, $\mathbf{W}^{(k)}$ a $k-1$ -edik és k -edik réteg kapcsolataihoz tartozó súlymátrix. A látható réteg neuronjainak állapotvektora: $\mathbf{v}$ ($\mathbf{v} = \mathbf{h}^0$).

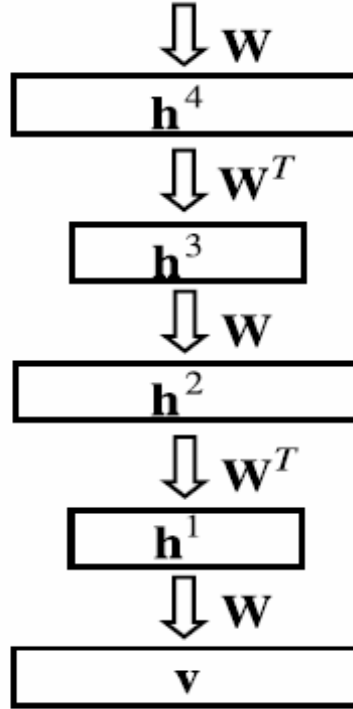
Célunk a kapcsolatokhoz tartozó súlyokat úgy beállítani, hogy a látható rétegen minél nagyobb valószínűséggel legyen a tanító adat. Erre egy Gradiens hegymászó módszert használhatunk ha az összes rejtett állapothoz tartozó

posteriori eloszlásáról tudunk torzítatlan mintát venni az input vektor alapján:

$$\Delta w_{ij} \propto \langle h_j^{(k)} (h_i^{(k-1)} - p(h_i^{(k-1)} = 1 | h^{(k)}, W^{(k)})) \rangle \quad (9)$$

Sajnos torzítatlan mintát csak kis méretű hálók esetében tudunk előállítani az explaining away hatás miatt.

Létezik a neuron hálóknak egy olyan speciális típusa, amely estén a rejtett rétegeken belüli posteriori eloszlás mintavételezése pont annyira könnyű feladat, mint az input generálása a modell alapján.



ábra 5: Végtelen háló első 4 rétege

Vegyünk az 5. ábrán látható neuronhálót, amelynek végtelen sok rétege van és kötött szimmetrikus súlyokkal rendelkezik. Ebben a hálózatban a rejtett egységek függetlenek tetszőleges input esetén. A függetlenség oka visszavezethető arra a tényre, hogy a korrelációt, amit a feljebb lévő rétegekből érkező prior okoz, teljesen kinullázza az antikorreláció, amely a lejjebb lévő rétegek likelihood tagjából származik[8]. A faktoriális posteriort ebben a hálózatban egyszerűen kiszámíthatjuk a látható vektorok és a súlymátrix transzponáltjának szorzataként. Ha a legelső rejtett rétegre elvégeztük a posterior mintavételezését, akkor a következő rétegre ugyanezt a módszert alkalmazhatjuk. A tanulást nehezíti, hogy a

súlymátrix minden másolatának más lesz a deriváltja, bár azt tudjuk, hogy magasan lévő rétegek esetén a derivált várhatóan 0 lesz. Ennek ok: a leírt módszer alkalmazásával lényegében egy Markov láncot kapunk, ami egy stacionárius eloszláshoz fog konvergálni magasabb rétegek esetén, stacionárius eloszlás esetén pedig az aktuális súlyok tökéletesek a minták megmagyarázására, ezért 0 a deriváltjuk.

A kötött súlyok használata a következőt jelenti: ugyanazt az eljárást használjuk $\mathbf{h}^2$ következtetéséhez $\mathbf{h}^1$ -ből mint amivel $\mathbf{v}$ -t generáljuk $\mathbf{h}^1$ -ből. Következésképpen $\mathbf{h}^2$ -re tekinthetünk úgy, mint látható réteg valószínűségeinek egy zajos torzítatlan becslésére, illetve $\mathbf{h}^3$ -ra mint a rejtett réteg valószínűségeinek egy zajos torzítatlan becslésére. Ezen két tulajdonságot kihasználva és a 9. képletet felhasználva a következő tanulás szabályt kapjuk:

$$\begin{aligned}\Delta w_{ij} &\propto \langle h_j^{(1)}(v_i - h_i^{(2)}) + h_i^{(2)}(h_j^{(1)} - h_j^{(3)}) \rangle \\ &\propto \langle v_i h_j^{(1)} \rangle - \langle h_i^{(2)} h_j^{(3)} \rangle\end{aligned}\tag{10}$$

Most, hogy már van egy hatékony tanuló algoritmusunk egy kötött súlyokkal rendelkező végtelen neuron hálóra, a következő lépésben egy olyan általános modellt hozunk létre, amely minden rétegen különböző súlyokat enged meg. Első lépésben tanítsunk kötött súlymátrixokkal, majd a legalsó réteg súlymátrixát feloldjuk és értékeit rögzítjük ($\mathbf{W}^{(1)}$). A többi súlymátrixot ismét kötött módon tanuljuk, annyi különbséggel, hogy a $\mathbf{h}^{(l)}$ -et kezeljük úgy mint az előzőekben $\mathbf{v}$ -t. Természetesen $\mathbf{h}^{(l)}$ -et $(\mathbf{W}^{(l)})^T$ segítségével számítjuk $\mathbf{v}$ -ből.

Az így kapott hálóban már a felsőbb rétegek nem adnak tökéletes torzítatlan mintát $\mathbf{h}^{(l)}$ -re, mivel $\mathbf{W}^{(l)}$ eltérő lesz a magasabb rétegek súlymátrixától, az azok által generált prior nem fogja tökéletesen kinullázni a korrelációt a likelihood tagban. Amiért mégis ezt a módszert használjuk: bizonyítható, hogy ha folytatjuk a $\mathbf{h}^{(l)}$ következtetését úgy, mintha a feljebb lévő súlymátrixok nem változtak volna, akkor a tanítóadat logaritmikus valószínűsége javulni fog[1].

Ezt az eljárást – alsó réteg rögzítése majd a felette lévők tanulása rögzített súlyokkal – alkalmazva tetszőleges számú réteg súlyait megtanulhatjuk. Amennyiben csak k db réteget tanulunk, akkor egy mély neuronhálót kapunk. Ha osztályozási feladatra szeretnénk használni a kapott hálót, akkor a k -nál

magasabb rétegeket² el kell dobni és helyettük egy ún. softmax réteget használni. Az így előálló hálót felügyelt tanító algoritmusokkal (pl backpropagation) finomhangolhatjuk.

Softmax réteg alatt a következőt értjük: n db osztálycímke esetén n db neuronból áll és mindegyik neuron egy címkének felel meg. Legyen $\mathbf{h}^{(k)}$ a legfelső előtanított réteg, l címke valószínűsége ekkor:

$$p(l|\mathbf{h}^{(k)}) = \frac{e^{b_l + \sum_i h_i^{(k)} w_{il}}}{\sum_m e^{b_m + \sum_i h_i^{(k)} w_{im}}}, \quad (11)$$

ez a háló azt a címkét rendeli a bemenethez, amelyre ez a valószínűség a legnagyobb. A felügyelt tanulás során már nem kell új tulajdonságdetektorokat tanulnia a hálónak, mivel azokat már az előtanítás során megtanulta, csupán finomhangolni azokat.

Az ismertetett általános előtanító algoritmus széles körben elterjedt, sikeresen használják nem csupán beszédfelismerési feladatokhoz, hanem dimenziócsökkentő eljárásokhoz, kézírás és 3-D objektumok felismeréséhez.

² Ezen rétegek a K . réteg súlymátrixát és annak transzponáltját használják

3. Implementáció

A tanító algoritmust c++ programozási nyelven valósítottam meg. A megvalósított tanító algoritmusok nagy időigénye miatt és a felhasznált tanító adatbázisok nagy mérete miatt GPU-n futtathatóvá tettem a programomat. A tanításokat nem online módon, hanem ún. kötegeken (batch) végeztem, egyszerre több input vektort felhasználva bemenetként és ezeken párhuzamosan tanítva a neuronhálót. Kötegek használata esetén a háló inputja egy vektor helyett egy mátrix lesz, amelynek sorai a tanulóvektorok, tehát a tanítás igazából mátrix műveletekből áll. A GPU-k nagy előnye a hagyományos CPU-kal szemben, hogy rendkívül jól használhatóak párhuzamos műveletek elvégzésére.

A jelenlegi grafikus kártyák (GPU) nagy teljesítményű párhuzamos rendszerek (sok száz processzorral és akár 1 teraflopnál nagyobb teljesítménnyel), amelyeket nem csupán a képszintézisben, hanem általános célú számításigényes feladatokban is alkalmazhatóak. A GPU processzortömbjét C-szerű programozási nyelveken lehet programozni (Cg, HLSL, CUDA, OpenCL, stb.), én a CUDA-t választottam, azon belül is elsősorban a CUDABLAS programkönyvtárat használtam a mátrixműveletek megvalósításához. Egy Nvidia GeForce GTX 560Ti kártyán futtatva lényeges gyorsulást tapasztaltam a hagyományos CPU-s verzióhoz képest (kb. 20-40-szereset, a pontos érték természetesen függ az aktuális gép paramétereitől).

A program bemenetként egy paraméterfájl nevét kapja, amelyből beolvassa a tanító adatbázis és a címkefájl nevét, a tanítandó hálózat struktúráját tartalmazó fájl nevét, illetve a későbbiekben ismertetésre kerülő további paramétereket.

3.1 Előtanítás

Az előtanítás a 2. fejezetben ismertetett mohó CD algoritmus alapján történt. A háló súlyait 0 várható értékű és 0.1 varianciájú normál eloszlás szerint inicializáltam, a bias-okat pedig mindet 0-ra. A legalsó rétegpáros egy Gaussian-Bernoulli RBM-et alkotott, a legfelső réteg pedig egy softmax réteg, amelynek súlyait előtanítás során nem módosítottam.

A Gaussian-Bernoulli RBM tanítása csupán annyiban különbözött a hagyományos

RBM-étől, hogy a látható réteg rekonstrukciója során nem alkalmaztam a szigmoid függvényt. Mivel a legelső rétegpáros valós bemenettel rendelkezett, szükségesnek bizonyult, hogy más paraméterekkel végezzem az előtanítást, ahogy azt [2]-ben javasolták. A tanítás hatékonyságának méréséhez a következő módszert használtam: az epoch-on belül összegeztem az összes batch-en vett négyzetes hibát (bemenet-rekonstruált bemenet) majd kiszámítottam az egy neuronra eső eltérést³. A [2]-ben javasolt módon én is 128-as mini batch-eket használtam, így gyorsabban konvergált az algoritmus. Általánosan megfigyelhető volt, hogy az első néhány epoch-ban a hiba jelentősen csökkent, majd a későbbiekben már csak minimálisan változott.

3.2 A *backpropagation*-algoritmus

A felügyelt tanításhoz (finomhangoláshoz) a hagyományos backpropagation algoritmust választottam, batch méretnek pedig az előtanítás során is használt 128-at választottam. Az előtanításnak köszönhetően a tanulási rátát viszonylag nagyra vehettem és a tanulás során, ha az eredmény nem javult kellő mértékben, akkor felére csökkentettem. A javulás mértékét a validációs halmazon elért eredmények alapján határoztam meg. A tanítás addig folytatódott, amíg az egymás utáni rossz iterációk száma el nem ért egy paraméterben kapott küszöböt.

³ Négyzetes hiba osztva a neuronok száma*példák számával

4. Adatbázisok

Ebben a fejezetben rövid leírást adok a program teszteléséhez használt adatbázisokról, illetve az előfeldolgozásról.

4.1 TIMIT adatbázis

A TIMIT angol nyelvű adatbázis személyfüggetlen fonetikai beszédfelismerők betanítására és tesztelésére szolgál. Szómodellek felépítésére alkalmatlan, mivel szűkített szótárkészletet használ, fonetikailag gazdag mondatai viszont kiválóan alkalmasak beszédhang modellek létrehozására. A korpusz összesen 630 beszélő által felolvasott, egyenként 10 fonetikailag gazdag mondatból áll. Az adatbázis két fő részre oszlik: egy része betanításra, másik része tesztelésre ad lehetőséget. A tanító rész adott százalékát (dolgozatomban 10%-át) a tanulás során validációra használtam. Több cikkben is a csökkentett címkekészletet használják a felügyelt tanítás során (39 osztály), de az újabb publikációkban már az eredeti 183 osztálycímke használatát javasolják, mivel ezzel jobb eredményt lehet elérni. A 183 címke előáll a korpuszban lévő 61 fonéma mindegyikéhez 3 állapot (kezdő, középső, végső) rendelésével.

4.2 Híradós adatbázis

A híradós adatbázis nyolc tévécsatorna (ATV, Hálózat TV, Hír TV, M1, M2, RTL, Tv2) 70 rögzített híradójából áll. A felvételek néhány mondatos blokkokra vannak felosztva és három kategóriába lettek besorolva minőség szerint: a „tisztá” kategóriába kerültek azok a felvételek, amelyeken szépen artikulált beszéd hallható, és a háttérzaj minimális. Tipikusan ide tartoznak a stúdióban, a műsorvezetőktől elhangzó részletek. „Zajos” besorolást kaptak a tervezett beszédet tartalmazó, de magasabb zajszintű felvételek – jellemzően a külső helyszínen tartózkodó riporterek bejelentkezése. Végezetül, a „spontán” címkét kapták a spontán beszédet tartalmazó blokkok – ezek tipikusan a riportalanyok szájából elhangzó mondatok. Jelen dolgozatban csak a tiszta besorolású felvételeket használtuk fel. A 70 híradót 44-9-17 arányban osztottuk fel betanítási (train), fejlesztési (development) és tesztelő (test) blokkokra, ügyelve arra, hogy a

tévécsonnának mindegyikéből kerüljön mindegyik részhalmazba. Időtartamot tekintve kb. 5 és fél óra – 1 óra – 2 óra arányban oszlanak el a felvételek a blokkok között. A beszédjelek fonetikai címkézése 52 címkével történt, ezek lényegében megfelelnek a magyar nyelv hangkészletének, a TIMIT korpuszhoz hasonlóan ebben az adatbázisban is minden címkét 3 külön címkére bontottunk: a beszédhang kezdete, közepe és vége, így összesen 156 címkét használtunk a tanításhoz.

4.3 Hangoskönyv

A harmadik adatbázis egy magyar nyelvű hangoskönyv: Krúdy Gyula Szinbád kalandjai. A korpusz különlegessége, hogy a teljes hanganyag egy felolvasótól (Gáspár Sándor) származik. A hangoskönyv 212 perces hanganyagot tartalmaz, amely WAV (Microsoft Wave mono 16 bit PCM) formátumba lett konvertálva 16 kHz-es mintavételezéssel. Előfeldolgozás előtt a hanganyag és az eredeti szöveg összevetésre kerül, az esetleges eltérések ki lettek javítva és a fejezetek végén hallható zene el lett távolítva. Tanítás és tesztelés céljából a hanganyag 2 részre lett osztva, a 10 történetből 8-at (186 perc) használtunk tanításra, 2-t (26 perc) pedig tesztelésre.

4.4 Előfeldolgozás

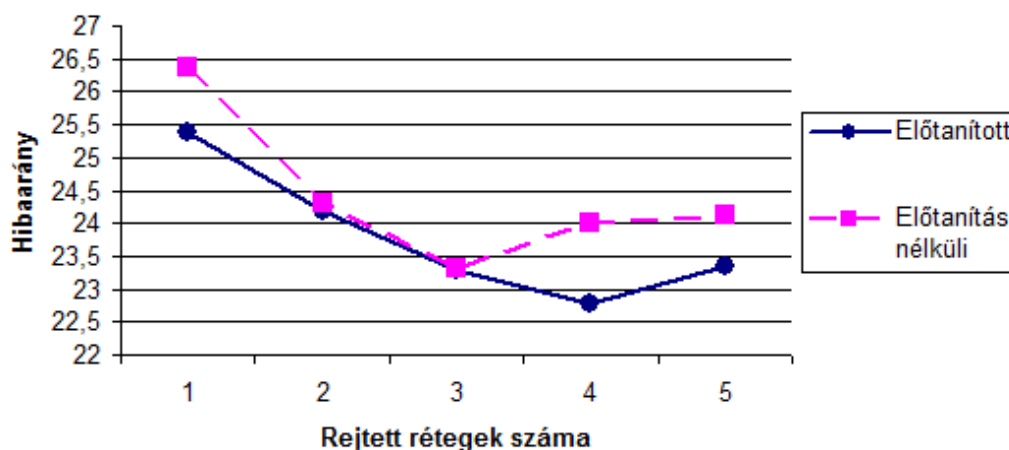
Az előfeldolgozás rendkívül fontos beszédfelismerési feladatok esetén, hiszen közvetlenül a beszédjelen a sok jellemző miatt nehéz tanítást végezni. Mindhárom adatbázis esetén ugyanazt az előfeldolgozási módszert alkalmaztuk: 25-ms Hamming ablakot véve 10-ms átfedéssel az azt megelőző ablakkal. A beszédjelet 12-ed rangú Mel frekvencia kepsztrális együtthatókkal (MFCC) és annak energiájával, illetve első és második rendű deriváltjával reprezentáltuk. A tanítás segítése érdekében az inputot normalizáltuk, 0 várható értékűre és 1 szórásúra. A tesztalmazon ugyanazt a normalizálást alkalmaztuk, mint a tanulóalmazon. A tanítás során nem csupán 1 ablakot vettünk bemenetnek, hanem annak adott számú szomszédját is az eredmény javítása érdekében, ennek magyarázata, hogy az így kapott bemenet nagyjából lefedte egy átlagos fonéma hosszát.

5. Eredmények

Ezen fejezetben a különböző adatbázisokon elért eredményeimet írom le. Mindegyik háló esetében a rejtett rétegek neuronszáma 1024 volt, az előtanítás során a legalsó réteget 50, a többi réteget 20 epoch-on keresztül tanítottam.

5.1 TIMIT

A tanítás során a háló bemenetként 9 beszédkeretet kapott előtanítás során a legelső rétegre 0.002-es, a többi rétegre pedig 0.02 tanuló rátát. 0.9-es momentum használatával gyorsítottam a konvergenciát, a weight cost paramétert pedig 0.0002-re állítottam. Backpropagation algoritmushoz 0.02-es tanuló rátát és 0.9-es momentumot használtam. Az eredményeket a 6. ábra mutatja.



ábra 6: Az előtanítás hatása rétegek számának függvényében

Jól megfigyelhető a 6. ábrán a javuló tendencia a rétegek számának növelésével, egészen a 4 réteges hálóig. Az 5 réteg esetén a kapott modell romlásának oka a túl kicsire inicializált kezdeti súlyok lehetnek, mivel ekkor a finomhangolás során a legalsó rétegekre csak nagyon kis gradienst kapunk így azok súlyait nem tudja megfelelően tanulni a modell. Látható, hogy a legjobb esetben (4 réteg) az előtanítás több mint 1 százalékat javít az előtanítás nélküli esethez képest. A TIMIT-en elért legjobb eredményemet hasonlítja össze az 1. táblázat az irodalomból ismert értékekkel (beszédhang-felismerési hiba a „core” tesztalmonon). Az 1. táblázatban látható, hogy a 2009-es cikkben [2] elért eredményt sikerült megjavítani, bár a későbbi cikkekben [8] annál sokkal jobb

eredményt is jelentettek. Ennek oka azonban az, hogy ebben a cikkben más előfeldolgozási módszert használtak.

Módszer	Hibaarány
Trifón HMM [2]	27,3%
Rekurrens Neuron Háló [2]	26,1%
Monofón HTM [2]	25,5%
Monofón Deep Net [2]	23%
Monofón Deep Net, saját implementáció	22,79%
Monofón Deep Net [8]	21,8%
Monofón Deep Net + MCRBM [3]	20,5%

táblázat 1: TIMIT-en elért eredmény összevetése más módszerekkel

5.2 Híradós adatbázis

A híradós adatbázison való tanuláshoz 15 beszédkeretet vettem bemenetként (7 szomszédság), a többi paramétert a TIMIT-en kitapasztalt értékre állítottam. Az így tanított monofón modellek felismerési pontossága:

Előtanított rétegek száma	Pontosság
3	78,74%
4	79,31%
5	78,84%

táblázat 2: Elért pontosság előtanítással

Itt is jól megfigyelhető, hogy az 5 réteges esetben már nem javult tovább az eredmény, hasonlóan mint a TIMIT-en. A [7]-ben közölt eredményekkel összevetve megállapítható, hogy monofón modellt használva a mély neuronháló lényegesen jobban teljesít mint a hagyományos HMM (67,18%) vagy mint a hibrid módszer 3 állapottal (76,93%). (Ennél ugyan jobb eredmény (83,33%) is található a cikkben, ehhez azonban nem egy szimpla neuronhálót, hanem egy kétfázisú modellt és trifón címkézést használtak)

5.3 Hangoskönyv

A hangoskönyv esetén (a híradóshoz hasonlóan) 15 keretet vettem bemenetnek, és ezen adatbázison is a legjobb eredmény a 4 előtanított réteggel rendelkező háló adta. A 4 réteges háló eredménye ezen adatbázison monofón modellt használva: 88,8%, ami lényegesen jobb mint a [6]-ban közölt legjobb eredmény: hibrid módszer (82,1%). Az adatbázis speciális volta miatt (csak 1 beszélő) célszerű az előtanítás paramétereinek további hangolása a még jobb eredmények elérése érdekében.

6. Összefoglalás

Dolgozatomban általános áttekintést adtam a mély neuronhálókról és tanítási algoritmusukról, illetve a segítségükkel tanított akusztikus modellek eredményeiről nem csak a TIMIT-en, hanem 2 magyar nyelvű adatbázison is.

Az eredmények fejezetben látható, hogy a mély neuronhálókval valóban jobb eredményt lehet elérni, mint a hagyományosakkal, ugyanakkor az is megfigyelhető, hogy előnyük főleg több (>3) rejtett réteg esetén válik szignifikánssá. A saját programommal tanított akusztikus modellek a TIMIT-en és a híradós adatbázison kiemelkedően jó eredményt értek el.

A mély neuronhálókra alapuló akusztikus modellezés biztató eredményeket produkált, ami azt erősíti meg, hogy érdemes a módszerrel mélyrehatóbban foglalkozni. A bemutatott eredmények további javításához több módszert is tervezek a jövőben kipróbálni:

- A rétegek számának növelését, ekkor viszont más inicializálást is kell használnom,
- A rétegekben lévő neuronok számának növelését,
- Monofón helyett trifón címkézés használata,
- [8]-ban javasolt előfeldolgozás használata.

Irodalomjegyzék

- [1] Hinton, G. E, Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets, *Neural Computation*, Vol. 18, pp. 1527-1554, 2006.
- [2] Mohamed, A., Dahl, G., Hinton, G.: Deep belief networks for phone recognition, *Proc. NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.
- [3] Dahl, G., Ranzato, M., Mohamed, A.-R., Hinton, G.: Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine, *Advances in Neural Information Processing Systems*, No. 23, pp. 469-477, 2010.
- [4] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks, *Proc. AISTATS*, pp. 249-256, 2010.
- [5] Toth, L.: A Hierarchical, Context-Dependent Neural Network Architecture for Improved Phone Recognition, *Proc. ICASSP 2011*, pp. 5040-5043.
- [6] Toth L., Tarjan B., Sarosi G., Mihajlik P.: Speech Recognition Experiments with Audiobooks, *Acta Cybernetica* 19 (2010). pp. 695-713.
- [7] Gosztolya, G., Toth, L.: Kulcsszókeresési kísérletek hangzó híryanagyokon beszédhang alapú felismerési technológiákkal, *Proc. MSZNY 2010*, pp. 224-235.
- [8] Mohamed, A., Dahl, G., Hinton, G.: Acoustic Modeling using Deep Belief Networks, *IEEE Trans. on Audio, Speech, and Language Processing*, 20, pp 14-22.