1 Pro uživatele

1.1 Úvod

Program umí zanalyzovat kurzovní data z minulosti a nazákladě nich napovídat, zda je výdělečné prodat měnu, nebo ne. Projekt má dvě části - Po-poradce.exe a Po-prodavač.dll.
Po-poradce.exe přijme kurzovní data ve jednoduchém formátu (např. EUR/USD kurz vzorkovaný každých 5 minut). Po-poradce.exe v jednom režimu zanalyzuje kruzovní data a vypíše specifické poznatky o nich. V druhém režimu natrénuje neuronovou síť na nějak dlouhém úseku ze začátku dat a uloží ji do souboru TEMP-saved-net.txt. Prodavač.dll je schopný tuto síť nahrát v reálném čase přijímat aktuální data a pomocí neuronové sítě zkoušet předvídat zda graf (kurzovní data) poroste nebo ne.

1.2 Po-Poradce.exe

Data se kterými program pracuje jsou cena/čas, přitom mezi dvěma cenami je vždy stejný odstup (např. 5 minut). Příklady vstupních souborů se nacházejí ve složce Menovy-poradce-source > text-files. Po-poradce.exe pracuje se změnami v datech.

```
Pro data
1.27994
1.27983
1.27943
1.27951
jsou odpovídající změny
-0.00011
-0.00040
+0.00008
```

Po-poradce.exe dělá dvě věci - podle toho v jakém je spuštěn režimu, přičemž režimy jsou dva: "-s"a "-n".

1.2.1 Režim -s

V režimu "-s"program nalezne 15 jevů, které s nejvyšší experimentální pravděpodobností předcházejí růstu grafu (tedy pozitivním změnám). Počet jevů je volitelný a v úvahu se berou jen ty, které nastaly s dostatečně velkou frekvencí - defaultně alespoň 5% (také volitelné). V datech se hledají následující tři třídy jevů:

A) průměr z K po sobě jdoucích změn padne do nějakého intervalu - např. (-1,0],(0,1],...,atd.

B) K po sobě jdoucích změn má malý/velký rozptyl. Tedy rozptyl padne do nějakého intervalu - např. [0, 0.1], (0.1, 0.2], ...atd., kde akorát namísto rozptylu se použije vzoreček avg(X - avg(X)), kde X je vektor K po sobě jdoucích změn.

C) N z K změn je pozitivních, kde N = 0,1,...,K.

Na slednování spojitosti mezi jevy a růstem grafu je potřeba dodat ve vstupním souboru dostatečně rozsáhlá data, aby experimentální pravděpodobnost měla nějaký význam - řekněme alespoň data za jeden rok. Po-poradce.exe pro každý jev spočítá kolikrát nastal a jaká je experimentální pravděpodobnost, že po něm následuje růst grafu.
Příklad výstupu:

increase prob.	occurence probability	window size	trait	parameter	additiona info.
0.777778	0.061503	30	A	(-0.000078, -0.000039)	IW: 3.88889e-005 #I: 20
0.741935	0.070615	25	A	(-0.000078, -0.000039)	IW: 3.88889e-005 #I: 20
0.733333	0.068337	28	A	(-0.000078, -0.000039)	IW: 3.88889e-005 #I: 20
0.730769	0.059226	25	C	10	
0.714286	0.063781	29	A	(-0.000078, -0.000039)	IW: 3.88889e-005 #I: 20
...					

- increase prob. = pravděpodobnost, že graf poroste.
- occurence probability = frekvence jak často jev v datech nastal.
- window size = počet po sobě jdoucích změn, nad kterými se hledají jevy.
- trait = do jaké z výše uvedených tříd padne tento jev.
- parameter = může to být interval, do kterého padne průměr (pro třídu A) nebo rozptyl (pro třídu B) nebo počet pozitivních změn (pro třídu C).
- additional info. = pro třídu A i B je to stejné - IW = šířka jednoho intervalu (do, kt. má padnout průměr nebo rozptyl), #I = počet intervalů kolem nuly (pro jev A) nebo od nuly nahoru po kladné ose (pro jev B), přičemž krajní intervaly jdou do plus/minus nekonečna a jsou nekonečně široké (aby byla pokrytá celá číselná osa).

1.2.2 Režim -n

Dále Po-poradce.exe umí na datech natrénovat neuronovou síť a uložit ji do souboru TEMP-saved-net.txt (volitelně může síť natrénovat na nějakých datech a zkoumat, zda na následujících datech dobře předvídá). Více viz knofigurační soubor. Natrénovaná síť lze posléze použít knihovnou Po-prodavač.dll, která ji umí nahrát a v reálném čase použít na předvídání růstu grafu. Několik parametrů neuronové síťě lze nastavit v konfiguračmím souboru. Neuronová síť používá back propagation, sigmoidní symetrickou aktivační funkci a plné propojení.

1.3 Konfigurační soubor

Všechny zmíněné nastavitelné paramtery a další lze nastavit v konfiguračním souboru Menovy-poradce-source > Po-poradce > Release > config.txt, který se při spuštění musí nacházet ve stejné složce jako program.

1.4 PRODAVAČ.DLL

Po-prodavač.dll je dynamicky linkovaná knihovna, která umí načíst neuronovou síť a v reálném čase odpovídat na otázku, zda graf poroste nebo ne. Neuronová síť umí přijmout K po sobě jdoucích změn v datech a na základě nich se pokusí předpovědět, zda graf poroste. To jestli síť bude předvídat dobře nebo špatně záleží na konfiguraci sítě a na datech.

1.5 JAK SE POUŽÍVÁ PO-PORADCE.EXE

Program k běhu používá fanndoubled.dll a config.txt - oba soubory musejí být před spuštěním ve stejné složce jako Po-poradce.exe. Při spuštnění musejí být nastavené nasledující command line arguemnty: *-s path* nebo *-n path*. Kde *-s* a *-n* určují režim a *path* je cesta k souboru s kurzovými daty (příkaldy dat ve spávném formátu jsou ve složce Menovy-poradce-source 〉 text-files).

1.6 JAK SE POUŽÍVÁ PO-PRODAVAC.DLL

Knihovna k běhu používá fanndoubled.dll a TEMP-saved-net.txt - oba soubory musejí být před spuštěním ve stejné složce jako Po-prodavač.dll. Jaké knihovna nabízí API lze nejlépe zjistit z kódu, více viz Prodavac.h.

1.7 JAK SE POUŽÍVÁ TESTER.EXE

Tester.exe je jednoduchý program demonstrující používání knihovny Po-prodavač.dll. Spuští se jendnoduše bez command line argumentu. Stačí si pohlídat, aby u Po-tester.exe byla ve stejné složce knihovna Po-prodavac.dll a knihovna fanndoubled.dll, kterou ona používá a také neuronová síť v souboru TEMP-saved-net.txt. Po-tester.exe demonstruje funkčnost .dll knihovny tak, že vezme datový soubor mensi.txt a síť natrénovanou na několika prvních změnách souboru mensi.txt, čte data, používá síť na predikci a následně hned porovná predikci s realitou. Jelikož síť byla trénovaná na řekněme prvních 100 změnách z mensi.txt, tak bude na těchto změnách mít 100% účinnost, na ostatních ovšemu už ne. Více viz kometáře kódu.

2 KOMPILACE

Stačí pomocí Visual Studia 2013 zkompilovat solutiony. (ostatní platformy než Windows analogicky dle solutionu)

2.1 KOMPILACE FANNDOUBLED.DLL/.LIB

Ve složce FANN-2.0.0-Source > VS2010 se nachází fann.sln. Tento solution stačí vybuildovat a objeví se složka FANN-2.0.0-Source > bin, kde se budou nacházet (mimo jiné) soubory fanndoubled.dll a fandoubled.lib. Soubor fandoubled.lib je potřeba přesunout do složky Menovy-poradce-source > includes-fann a soubor fanndoubled.dll je pak potřeba přiložit ke každému *.exe* nebo *.dll*, který jej používá (to jsou Po-poradce.exe a Po-prodavac.dll) .. alternativně lze fanndoubled.dll presunout do systémové složky Windows > system32, odkud by jej měla umět použít libovolná aplikace (která má přístup přes fandoubled.lib). Takže například je potřeba vložit fandoubled.dll do složek Menovy-poradce-source > Po-poradce > Release .. atd.

2.2 KOMPILACE PORADCE.EXE

Stačí vybuildovat solution a před spuštěním k umístění Po-poradce.exe přikopírovat knihovnu fanndoubled.dll, kterou používá.

2.3 KOMPILACE PRODAVAC.DLL/.LIB

Stačí vybuildovat solution. Výsledkem budou ve složce Menovy-poradce-source > Po-prodavac > Release soubory (mimo jiné) Po-prodavac.dll a Po-prodavac.lib.

2.4 KOMPILACE TESTER.EXE

Tento program testuje funkčnost knihovny Po-prodavac.dll. Před jeho kompilací se tedy musí Po-prodavac.lib zkopírovat do složky Menovy-poradce-source > includes-prodavac. Pak se vybuilduje solution a před spuštením Po-tester.exe se do stejné složky jšetě musí zkopírovat Po-prodavac.dll, fanndoubled.dll a také uložená neuronova síť natrénovaná na datech ze souboru mensi.txt, kterou bude Po-prodavac.dll používat - ta musí být v souboru s názvem TEMP-saved-net.txt. Pokud se program spouští z Visual Studia, je potřeba neuronovou síť umístit do složky Menovy-poradce-source > Po-tester > Po-tester (working directory).

3 Programatorska dokumentace

3.1 Po-poradce.exe

Po-poradce.exe má 2 režimy běhu -n a -s.

3.1.1 Režim -s

Režim -s vypíše řekněme 15 nejleších jevů tak, že nejdříve do temporary souborů vypíše všechny traity s jejich increase-probability a occurence-probability - jedna třída jevů v jednom temporary souboru. Dále z každého souboru jedním průchodem vybere 15 nejlepších jevů a pak je slije dohromady.
Nové jevy lze přidat jako potomky třídy Trait nyní jsou nadefinované jevy A,B,C takže další v pořadí bude D, ale to není vše, ještě je potřeba přidat nové jevy do seznamu jevů třídy Statistic a pak adekvátně přidat kód do funkce get_trait. Také je potřeba do File_manager přidat navý TEMP soubor pro nový Trait.
Pro výpočet jevů jsou důležité tři parametry - window-size, gap-size a view-size. Window-size je počet změn, nad kterými se jev počítá, view-size je počet změn, ze kterých se usuzuje increase (increase může být definován různě - např. první nasledující změna je pozitivní, 10 změn je pozitivních, na libovolném počtu změn se graf zvedne on 0.01, atd. - proto je potřeba volitelná view-size). Gap-size je velikost mezery mezi window a view. Jakým způsobem se měří increase rozhoduje funkce Inc_tester 〉 test_increase - ta lze přeimplementovat nebo případně kdyby bylo více vhodných funkcí, tak by se do konfiguračního souboru měla přidat možnost výběru.
Výpočet všech jevů v jedné tríde probíhá inkrementálně přes window-size, přičemž třída Support emuluje všechny window-sizes zároveň, takže stačí jeden průchod vstupním souborem.

3.1.2 Režim -n

Tento režim používá dynamicky linkovanou knihovnu FANN k natrénování sítě na datech. Všechny potřebné hlavičkové soubory jsou ve složce FANN-2.2.0-Source, kde se nachází i zdrojáky potřebné pro kompilaci fanndoubled.dll a fanndoubled.lib. Program natrénuje síť a pak ji buď vyzkouší na následujících datech nebo uloží do souboru, nebo otestuje jestli se natrénovala správně - což měla. Síť přijme N změn a vrátí předpověď zda bude nebo nebude increase, počet vstupních neuronů je tedy stejný jako window-size a výstupní jsou dva. Nejdříve se vytvoří hloupá síť - její tvar závisí na window-size a na počtu vrstev, program na základě nich automaticky vytvoří síť vajíčkovitého tvaru. Pak se vytvoří tréninkový soubor (study material), nastaví se několik parametrů sítě a pak se na tréninkovém souboru nechá netrénovat. Síť lze jednoduše konfigurovat přes konfigurační souboru, do něj je také možno přidat další položky.
Otázka je, zda vajíčkovitý tvar je optimální. Z technických důvodů auto-generoání sítě podporuje nej tři až pět vrstev - jiný počet lze zařídit mechanickým připsáním kódu do default_layers, ale není to hezké - problém je v tom, že variadická funkce create_standard knihovny FANN neumí přijímat va_list.

3.1.3 CONFIGURATION

Knofigurace je zapsaná v souboru `config.txt`. Všechny podrobnosti k tomuto souboru jsou v `config-explained.txt`. Třída `Configuration` má jednu globální instanci `CONFIG` ta si načte data z `config.txt` v konstruktoru a liobvolná část kódu ji tak může použít. Přidání nových položek do `config.txt` vyžaduje přidání kódu do `Configuration`.

3.1.4 FILE-MANAGER

Temporary soubory spravuje třída `File_Manager`, která má jednu globální instanci `FM` a libovolná část kódu ji tak může použít. Nové temporary soubory do ní lze jednonuše přidat, dokonce seznam souborů má připravená volná místa pro tento účel.

3.2 PO-PRODAVAC.DLL

Předvídání zajišťuje globální neuronová síť, se kterou se komunikuje skrze API funkce.

4 POZNÁMKY

Zdá se, že defaultní styl buildování - nastavený od výrobců knihovny FANN je Debug build - přenastavením na Release build by se mohlo dosáhnout lepšího vykonu (zatím zbytečné). Statistice (režim -s) trvá 40s výpočet nad daty za 1 rok (5-minutové vzorky) a neuronová síť je schopná se naučit až 1000 po sobě jdoucích změn.

5 PROBLÉMY

Statistika trvá lineární I/O čas v délce vstupního souboru pro každý trait (ty jsou 3, ale je možné nějaké přidat jako potomky třídy Trait) - to by šlo vylepšit, kdyby to z nějakého důvodu bylo potřeba, ale je to zatím jen zbytečně složité).
Jelikož je složité optimalizovat tvar a parametry neuronové sítě, program tuto funkcionalitu neobsahuje (jeden z problémů je, že trénik sítě závisí na náhodne inicializaci vah na hranách sítě). Obecně problematika neuronových sítí a úpěšného forcastingu se zdá moc složitá na to aby taková funkcionalita byla implementovaná v tomto díle.

6 WARNINGY

Mnoho signed/unsigned mismatches vyvstane z toho, že vector.size() je typu *size-type*, který je definován v core knihovnách a jeho definice je platformně dependentní, na Windowsech to vypadá na něco unsigned (asi unsigned int). Dále jsou tam double-to-float konverze (hlavně při používání FANN funkcí).

7 REFLEXE

Projekt mě naučil programovat C++ stylem (hodně hlavičkových a .cpp souborů, makra) i trochu C stylem (interakce s knihovnou), také používat a vytvářet *.dllka* (fanndoubled.dll, Po-prodavac.dll), také základy neuronových sítí a uvědomit si, jaký je rozdíl mezi Release buildem a Debug buildem. Dosavadní verze sice nemá vyhlídky na nějaké reálné použití, ale program lze rozšířit o různé komponenty a posílit jeho funkcionalitu.