Evoluční kódování v Yale

Jan Skalický

***Abstrakt*— Tento článek pojednává o testování evolučního kódování na datech použitých v induktivním modelování neuronovými sítěmi. Předzpracování dat při takovém modelování může zviditelnit souvislosti, které jsou neuronovými sítěmi a jejich gradientním učením obtížně zachytitelné.**

I. Zadání

Cílem práce je otestovat plugin pro prostředí Yale, který realizuje evoluční kódování dat.

II. Úvod

A. Yale

Program Yale je modulární prostředí pro vytěžování poznatků z dat a učení modelů k jejich zpracování. Modularita tohoto prostředí spočívá v definici experimentu, který je hierarchickou aplikací zabudovaných operátorů různých typů na vstupní data. Předmětem jisté diplomové práce katedry počítačů FEL ČVUT byla tvorba pluginu pro toto prostředí, který realizuje evoluční kódování. Jde tedy o sadu nových operátorů pro použití v experimentu.

B. Evoluční kódování

Evoluční kódování je metoda předzpracování dat překódováním vstupních hodnot tak, aby byly lépe přizpůsobené pro některé přístupy jejich zpracování umělou inteligencí. Jedná se vlastně o zobrazení domény atributů (zejm. těch s nominálními hodnotami) do číselné domény a toto zobrazení je šlechtěno evolučním algoritmem. V případě realizace testovaným pluginem pro Yale jde o evoluci užitím genetického algoritmu, kde jako měřítko kvality kódování vystupuje korelace mezi kódovaným atributem a predikovanou třídou jevu (ve skutečnosti je fitness funkce realizována vnořeným operátorem, což umožňuje její volitelnou definici).

III. Cíl práce

Práce si klade za cíl porovnat výsledky klasifikace/predikce dat neuronovou sítí dosažené užitím evolučního kódování s výsledky na nepřizpůsobených datech. Vzorek dat byl součástí zadání práce (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/car/).

IV. Příprava prostředí

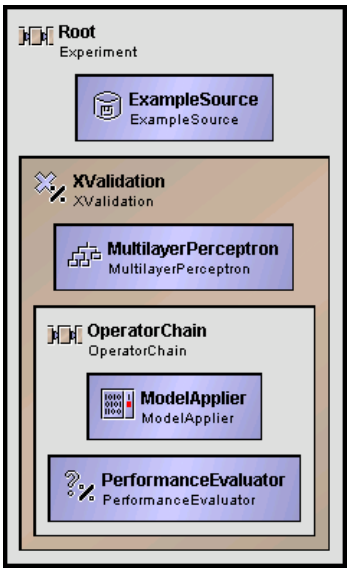
Prostředí Yale bylo od doby tvorby pluginu aktualizováno projektem Rapidminer. Zjistili jsme, že plugin však není kompatibilní s novou verzí, a proto byla jako referenční verze programu zvolena starší distribuce Yale 3.4. Ačkoliv data byla zadána ve formátu C4.5 a Yale obsahuje operátor pro jeho čtení, nepodařilo se nám je úspěšně načíst, a proto jsme vytvořili jejich popis manuálně ve formátu AML pro vstupní operátor ExampleSource, který provádí načtení vzorů. Všem sloupcům jsme deklarovali nominální typ atributu (použitelné pro slovní hodnoty či obecně jinak neuspořádatelné).

V. Model zájmu

Modelem aplikovaným na předzpracovaná data je vícevrstvá perceptronová neuronová síť s dopředným šířením - MLP (Multi-Layer Perceptron). Volba jejích parametrů (počet a mohutnost skrytých vrstev, parametry učení) bude závislá na konkrétním experimentu. Experimentální data byla převzata z databáze dat pro strojové učení a jedná se o klasifikaci automobilů do tříd vhodnosti k nákupu podle základních parametrů (počet míst, provozní náklady, bezpečnost apod.).

VI. Experimenty

V kontextu s původní prací, která provádí experimenty konvenčního i evolučního kódování na neuronové síti a rozhodovacím stromu, bylo zvoleno základní schéma experimentu užívající křížovou validaci na datech. Kritériem pro hodnocení výsledků modelu je úspěšnost klasifikace. Schéma základního experimentu nepoužívajícího žádné předzpracování dat vidíme na obrázku 1. Schéma experimentu používajícího evoluční kódování je na obrázku 2.

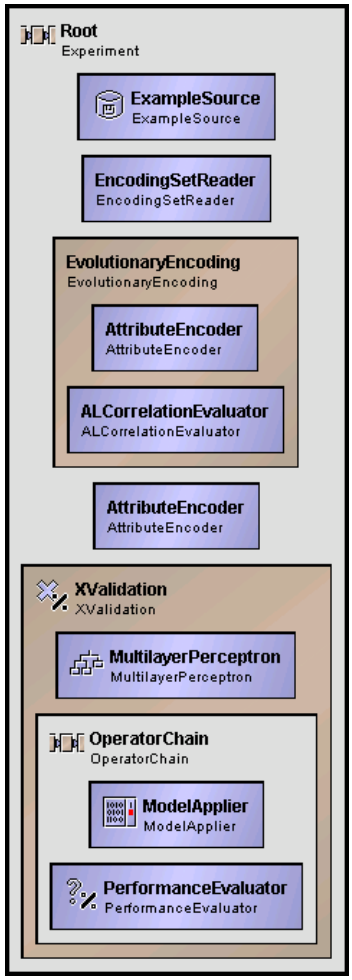


Obr. 1. Základní experiment Yale

A. Data

Zdrojem dat je klasifikace automobilů podle parametrů:

- cena pořizovací
- cena údržby
- počet dveří
- kapacita osob
- kapacita zavazadelníku
- bezpečnost



Obr. 2. Experiment Yale s evolučním kódováním

Všechny atributy jsou nominálního typu (typicky ve tvaru cena vysoká/střední/nízká, kapacita 2/4/více). Počet instancí v souboru s daty je 1728 a pokrývají celý atributový prostor. Klasifikační třídy jsou 4 (nepřijatelné, přijatelné, dobré, výborné) s distribucí cca 70%, 22%, 4%, 4%. Soubor neobsahuje chybějící data. Formát dat je c45, ale ačkoliv má Yale operátor pro načtení tohoto formátu, jejich import se nezdařil, takže byla vytvořena definice těchto dat ve formátu AML. Operátor ExampleSource provádí načtení těchto dat do experimentu.

Křížová validace, použitá pro vyhodnocení výsledků byla nastavena na hodnotu 10 se stratifikovaným vzorkováním. Testovací neuronová síť - vícevrstvý percpetron, byl učen algoritmem backpropagation s koeficientem učení 0.3 a setrvačností 0.2 (výchozí hodnoty). Ukázalo se, že je vhodné experimentovat s počtem epoch učení a vzhledem k tomu, že jednou z věcí, kterou je možno sledovat na výstupu, je rychlost konvergence učení, volili jsme menší hodnoty - 10 a 50 epoch.

Pokud nepoužijeme žádné předzpracování dat, zůstanou domény použitých atributů nominální a je nutné je na vstupu sítě kódovat kódováním 1 z N. To má za následek zvětšení vstupní vrstvy sítě, protože vstupem jsou pak binární hodnoty testující atribut na všechny hodnoty z jeho domény. Tím ovšem dojde ke změně velikosti i v další vrstvě, protože vhodná volba velikosti skryté vrstvy pro 21 vstupů se jistě liší od sítě se 6 vstupy. Implementace perceptronu v Yale implicitně nabízí velikost skryté vrstvy jako $(attributes + classes)/2$. Tato skutečnost poněkud kompiluje srovnávání výsledků základního experimentu a výsledků s kódováním atributů, protože síť pro překódovaná data bude znatelně menší. Můžeme však experimentovat alespoň s velikostí skryté vrstvy, jejíž činnost je ovšem omezena množstvím informace, které do ní přichází, takže v případě 6 překódovaných vstupů nemá cenu nastavovat velké hodnoty. Bez problémů můžeme ale srovnávat efekt simulovanané evoluce na kódování a použití náhodného nebo ručně vyrobeného kódování.

B. Operátory

Operátor EncodingSetReader načítá kódování atributů ze souboru XML. Tak můžeme vyrobit kódování ručně. Všechny atributy byly kódovány do intervalu 0.0 - 1.0, tozn. normalizované reálné číslo (zpočátku bylo experimentováno i s celými čísly). Hlavní operátor evolučního kódování obsahuje operátor pro jeho aplikaci na datech AttributeEncoder a vyhodnocování jeho kvality ALCorrelationEvaluator. Autor implementace metody zvolil jako hodnotící funkci korelaci mezi atributem a labelem instance.

Vlastní kódování provádí genetický algoritmus. Lze nastavit velikost populace, počet generací nebo zastavení příznakem konvergence algoritmus selekce, mutace a pravděpodobnost křížení (postrádám pravděpodobnost mutace). V experimentech jsme použili turnajovou selekci a mutaci Gaussiánem. Vzhledem k tomu, že experiment má odpovědět na otázku, zda-li má evoluční kódování vůbec smysl, snažíme se nastavit parametry evoluce tak, aby dávala co možná nejkvalitnější výsledky bez ohledu na dobu jejího běhu. Výchozí nastavení velikosti populace 5 bylo proto navýšeno na 20, později na 50 jedinců a počet generací na 100. Delší doba běhu nám zde nevadí, cílem je najít nejlepší kódování (vzhledem k použité fitness).

Operátor evolučního kódování umí šlechtit pouze jeden atribut (aplikovat kód pak umí hromadně), proto jsme nejprve všechny atributy postupně zakódovali a poté aplikovali klasifikaci neuronovou sítí. Nejprve jsme však vyrobil kódovaní ručně, ve snaze zlepšit separovatelnost prostoru instancí. Další zakódování bylo vyrobeno dosazením náhodných hodnot (z intervalu 0.0 - 1.0) za původní nominální hodnoty. Máme tak materiál k porovnání s výsledky evoluce.

VII. Naměřené výsledky

Tabulka I ukazuje výsledky klasifikace MLP učené 10 epoch. Sloupce kromě posledního jsou vlastně prvky hlavní diagonály konfuzní matice. Je vidět, že obtížně klasifikovatelné jsou třídy good a vgood, které nejlépe klasifikuje síť s kódováním 1 z N. Evoluční kódování dosahuje lepších výsledků než předdefinované (náhodné nebo ruční). Evoluce kódu tedy separovatelnost těchto, jinak špatně rozpoznatelných tříd, signifikantně zlepšuje. Vzhledem k tomu, že třída unacc se 70% pokrytím v datech nečinila větší problémy žádnému

TABULKA I

ÚSPĚŠNOST KLASIFIKACE PO 10 EPOCHÁCH UČENÍ MLP [%]

třída	unacc	acc	good	vgood	celkem
kódování 1 z N	98.43	94.53	82.61	95.38	96.82
náhodné kódování	90.25	69.01	0.00	0.00	78.53
ruční kódování	95.87	90.36	31.88	29.23	89.58
evoluční kód 1	95.95	90.89	65.22	58.46	92.19
evoluční kód 2	95.79	90.62	56.52	60.00	91.73
1 z N + evol 1	98.84	94.79	86.96	89.23	97.11
1 z N + evol 2	99.01	94.01	85.51	87.69	96.93

TABULKA II

ÚSPĚŠNOST KLASIFIKACE PO 50 EPOCHÁCH UČENÍ MLP [%]

třída	unacc	acc	good	vgood	celkem
kódování 1 z N	99.75	98.18	95.65	98.46	99.19
náhodné kódování	89.09	78.39	4.35	29.23	81.08
ruční kódování	97.69	89.84	73.91	70.77	93.98
evoluční kód 1	97.85	94.01	88.41	93.85	96.47
evoluční kód 2	97.93	94.53	85.51	92.31	96.47
1 z N + evol 1	99.92	98.18	97.10	98.46	99.36
1 z N + evol 2	99.92	97.66	92.75	98.46	99.07

kódování, celková úspěšnost klasifikací je i přes špatné výsledky na některých třídách poměrně vysoká.

Po 50 epochách (tab. II) učení se výrazně zlepšuje úspěšnost evolučního kódování na problematických třídách. Předdefinovaná kódování klasifikují tyto třídy o něco lépe, ale náhodný kód se zlepšil jen nepatrně a poskytuje stále nepoužitelné výsledky. Zdá se tedy, že použitá síť MLP-6-5-4 pro číselné kódování, nemá při špatném zakódování vzhledem ke své velikosti možnost problém dobře separovat ani při delším učení. Takové případy jsou vhodné právě pro nasazení evolučního kódování (nebo jiného inteligentního určení pravidel), které zde dosahuje cca 2x menší celkové chyby než ruční zakódování. Úspěšnost kódování 1 z N se dalším učením nadále vylepšuje, zejm. na třídě good, kde byla úspěšnost pro krátké učení nízká.

Evoluční kód 1 vznikl genetickým algoritmem s menší populací a menším počtem generací než evoluční kód 2. Již při kódování bylo však z průběhů fitness funkce v populaci zřejmé, že výsledná kódování nebudou dávat příliš rozdílné výsledky. Průběhy konvergovali poměrně brzo a někdy se výsledná zdatnost příliš nelišila od hodnoty v počáteční populaci. Předmětem dalšího zkoumání by tak mohla být otázka úspěšnosti nalezení skutečně vhodného kódování, která vzhledem ke generickému pojetí GA zde bude zejména otázkou volby hodnotící funkce. Zde použitá korelace atributu s výstupem má však výhodu v nenáročnosti výpočtu (není třeba šlechtit na vlastní síti). Vzhledem k možnostem parametrizace vlastního GA by však bylo vhodné zařadit stupně volnosti i do nastavení hodnotící funkce, která dle mého názoru sehraje na úspěšnosti nalezení vhodného kódování majoritní vliv.

VIII. DISKUSE

Pozorování na experimentech dává hrubou představu o důsledcích vhodného kódování nominálních atrubitů do intervalu hodnot na schopnost sítě správně klasifikovat. Nabízí se však otázka, je-li takové kódování užitečné vzhledem k

TABULKA III

ÚSPĚŠNOST VŮČI VELIKOSTI SÍTĚ PO 50 EPOCHÁCH [%]

síť	MLP-3	MLP-5	MLP-9	MLP-13	MLP-17
kódování 1 z N	91.26	94.73	98.55	99.19	99.31
náhodné kódování	77.84	81.08	82.58	83.51	83.45
ruční kódování	89.53	93.98	94.79	96.24	96.70
evoluční kód 1	92.13	96.47	96.76	96.82	97.05
evoluční kód 2	92.94	96.47	96.99	96.88	96.93

TABULKA IV

ÚSPĚŠNOST KLASIFIKACE PŘI ZAKÓDOVÁNÍ POUZE 1 ATRIBUTU [%]

třída	unacc	acc	good	vgood	celkem
buying	99.75	97.92	86.96	100.00	98.84
maint	100.00	97.40	92.75	95.38	98.96
doors	99.34	95.31	91.30	98.46	98.09
persons	98.26	92.71	85.81	93.85	96.35
lug_boot	99.59	96.88	91.30	92.31	98.38
safety	99.75	97.66	85.51	95.38	98.55

faktu, že kódování 1 z N nám ve všech případech přineslo lepší úspěšnost klasifikace. Tento problém má několik aspektů. Jednak zde hodnotíme vlastnosti evolučního kódování, které má jako hodnotící funkci korelaci atributu a výstupu, ačkoliv myšlenka evolučního kódování nás v tomto ohledu neomezuje (dokonce ani iterativní heuristikou nemusí být zrovna GA). Druhým závažným aspektem porovnání výsledků evolučního kódování s kódováním 1 z N je struktura klasifikátoru. Pro číselně kódované atributy je totiž počet vstupů roven pouze počtu atributů a nikoliv počtu všech použitých prvků z domén jako u 1 z N. Problém jiné mohutnosti vstupních vrstev se přenese i do skrytých vrstev, protože za vrstvou s 5 neurony bude mít skrytá vstva z 15 neurony nejspíš menší využitelnost, než pokud jich bude 15 i na vstupu. Celkově tak jde o nepoměr vah k nastavování - při kódování 1 z N jich máme více, zde bylo experimentováno se sítěmi MLP-21-13-4 pro 1 z N, což je 342 vah a s MLP-6-5-4 pro číselné kódování, což je pouze 59 vah. Je evidentní, že při kódování 1 z N není zcela využita informačí kapacita kanálu mezi vrstvami, protože např. na vstupu je v rámci 1 atributu aktivní pouze 1 neuron z několika (kódování je vlastně redundantní), ale situace v dalších vrstvách již taková být nemusí a tyto rozdíly (více vah, větší redundance informace na začátku sítě a naopak) se nemusí kompenzovat.

Očekával bych, že výše uvedené znevýhodňuje klasifikátor s číselným kódováním; když pomineme možnost přeučení, tak méně vah je obecně méně podrobná separace tříd. Vzhledem k tomuto aspektu jsme porovnávali ještě úspěšnost kódování při měnících se parametrech struktury sítě, viz tab. III. Číslo za označením sítě je mohutnost její skryté vrstvy (vstup pro kód 1 z N má 21 neuronů, pro číselná kódování 6). Vidíme, že neúspěch náhodného kódování nelze opravit zvětšením sítě. Zajímavé je, že ruční kódování má tendenci konvergovat k výsledkům z evoluce, ale činí tak pomaleji, tozn. pro stejnou úspěšnost vyžaduje větší počet neuronů. Toto chování by odpovídalo představě, že evoluční kódování lépe zviditelňuje souvislosti v datech, které pak dokáže zachytit i menší síť. To je pozitivní zjištění, protože v některých případech může být požadavek na zmenšení sítě zásadní.

Pokud se snažíme přizpůsobovat velikost sítě ve snaze lépe porovnat výsledky evolučního kódování s 1 z N, zjistíme, že při výrazném (3x) zmenšování sítě pro 1 z N začína hůře klasifikovat (o něco hůře než evolučního kódování se stejným počtem neuronů ve skryté vrstvě) a naopak při výrazném zvětšování sítě pro evoluční kódování se úspěšnost klasifikace zlepšuje jen nepatrně (při ztrojnásobení asi jen o 0.5 %). Zdá se tedy, že toto přizpůsobování vede spíš na degeneraci struktury než na stavy s výsledky použitelnými k porovnání. Lze říci, že vhodné číselné kódování klasifikuje dobře i s malou sítí, kterou nemá cenu příliš zvětšovat. Kódování 1 z N vyžaduje více vstupů a potom i více neuronů ve skryté vrstvě. Pokud je jich málo, začne se úspěšnost rychle zhoršovat. Výhodou je lepší úspěšnost klasifikace při vyšším počtu neuronů, avšak je nutné učit výrazně větší počet vah, což se projeví na rychlosti. Je také možné evolučně zakódovat pouze některé atributy a zbytek ponechat v kódu 1 z N. Některé atributy totiž půjdou do intervalu kódovat lépe než jiné, např. uspořádatelné. Výsledky takového expetimentu jsou uvedeny v tab. IV. Ještě by bylo možné použít obojí kódování zároveň, což sice zvýší dimenzionalitu dat, ale může přinést novou informaci. Pohledem na poslední 2 řádky tabulek I a II však nezjišťujeme velký rozdíl při těchto vstupech od použití samotného 1 z N (v jednom případě jsme obdrželi dokonce nepatrně horší výsledek).

IX. Závěr

Otestovali jsme funkčnost evolučního kódování v Yale na umělých datech a učinili několik poznatků. Evoluční kódování dosahuje znatelně lepších výsledků než ruční nebo náhodné kódování do intervalu a rychleji konverguje vzhledem k délce učení, což jej činí použitelným. Nejlepší výsledky s roustoucím počtem epoch dává kódování 1 z N, ale jeho přímé porovnání s evolučním kódováním je problematické. Síť pracující nad číselným zakódováním však může být menší a pokud použijeme stejně malou síť i pro 1 z N, ztrácí tím výhodu v úspěšnosti klasifikace. Otázkou evolučního kódování zůstává ještě volba hodnotící funkce. Celkově se zdá, že v případě, kdy nejsme omezeni velikostí sítě, pak konzervativnější kódování 1 z N dává lepší výsledky. Hypotézy zde zmíněné by však bylo vhodné ověřit v rámci rozsáhlejšího experimentu.

Poděkování

Děkuji Ing. Pavlu Kordíkovi, Phd. za ochotu konzultovat otázky týkající se práce.

Literatura

[1] Petr Zelenka: *Predzpracování dat v programu YALE*, Diplomová práce, 2007.
[2] katedra počítačů FEL ČVUT: *Stránky podpory výuky předmětu X36NAN*, WWW, 2007.