Strojové učenie

Rozpoznávanie veľkých písmen anglickej abecedy

Katarína Fabianová

4. januára 2018

Abstrakt

Cieľom tohto projektu je multi klasifikácia, pomocou ktorej budem rozpoznávať veľké písmena anglickej abecedy. Dáta sú vytvorené z obrázkov jednotlivých písmen z rôznych fontov. Dáta sú štatistické atribúty písmen. Na multi klasifikáciu používam nasledujúce metódy: Support Vector Classifier, Random Forest Classifier, Decision Tree Classifier a Logistic Regression. Po porovnaní metód som zistila, že najlepšia metóda na multi klasifikáciu na tieto dáta je Support Vector Classifier, ešte celkom dobrou je Random Forest Classifier, potom nasleduje Decision Tree Classifier a úplne najhoršou je Logistic Regression.

1 Úvod

Zadaním projektu je rozpoznať veľké písmená anglickej abecedy pomocou multi klasifikácie. Multi klasifikácia sa dá spraviť viacerými metódami. Vybrala som si štyri a to Support Vector Classifier, Random Forest Classifier, Decision Tree Classifier a Logistic Regression. Natrénujem trénovacie dáta so všetkými metódami a otestujem pomocou testovacích dát a porovnám výsledky testovacích chýb všetkých metód.

Dáta sú získané z tejto stránky: `https://archive.ics.uci.edu/ml/datasets/Letter+Recognition`. Dáta sú numerické, každý riadok obsahuje kategóriu písmena a 16 atribútov, ktoré ho popisujú. Dáta boli vytvorené tak, že sa vytvorili obrázky písmen na základe 20 rôznych fontov. Obrázky pozostávali z bielych a čiernych pixelov. Čierne pixely tvoria písmeno. Každé písmeno bolo náhodne deformované. Každú deformáciu popisuje 16 numerických atribútov, ktoré boli škálované na interval $\langle 0, 15 \rangle$. Atribúty sú štatistické, napríklad šírka, dĺžka obrázka, počet čiernych pixelov, priemer x-ovej súradníce z x-ových súradníc čiernych pixelov, priemer y-ovej súradnice zo všetých y-ových súradnic čiernych pixelov a podobne. Súbor s dátami obsahuje 20 000 riadkov, ktoré popisujú všetky vytvorené deformácie písmen. Každé písmeno má cca 700-800 deformácií v súbore.

Obr. 1: Príklady obrázkov písmen

Dáta som rozdelila na trénovaciu a testovaciu množinu v pomere 3:1. Vyskúšala som aj iné rozdelenie v pomere 4:1. Obidve rozdelenia dosahujú dobré výsledky, ale druhé rozdelenie dosahuje trošku lepšie výsledky, čo je celkom logické, keďže trénovacia množina je väčšia.

2 Kontext

Multi klasifikácia na rozpoznávanie veľkých znakov je celkom známa a bola už viac-krát implementovaná. Konkrétne s týmito dátami som našla dve implementácie, jedna je v programovacom jazyku R a druhá je v Pythone.

1. Implementácia v R: `https://github.com/davidgasquez/letter-recognition/blob/master/letter-recognition.R`

2. Implementácia v Pythone: `https://github.com/mriganktiwari/Letter_recognition_UCI/blob/master/ML1_final_report_Group-17.pdf`

Na tomto datasete dosahuje multi klasifikácia veľmi dobré výsledky, skóre má cez 90%.

3 Použité metódy

V projekte som použila štyri metódy. Rozdelila som si upravené dáta na trénovaciu a testovaciu množinu. Každá metóda je natrénovaná na trénovacích dátach a predikuje y hodnoty na testovacej množine. V každej metóde sa ráta skóre a cross validačné skóre úspešnosti predikovania správnych výsledkov na testovacej množine.

1. **Support Vector Classifier**
   Túto metódu som si zvolila ako prvú, lebo je najvhodnejšia na zvolený dataset. *SVC* metóde som nastavila *kernel="rbf"*, pretože chcem dosiahnuť klasifikáciu do viacerých tried. Skúšala som nastaviť aj iné kernely, konkrétne lineárny a polynomiálny. Pri lineárnom bolo skóre 83%, pri polynomiálnom 94%. Cross validačné skóre bolo ale ešte nižšie, 81% a 89%. Pri oboch kerneloch trvala klasifikácia dlhšie ako pri rbf, čo je radial basis function, čiže nejaký Gaussian, ktorý vie najlepšie klasifikovať dáta do viacerých tried. Ďalší parameter, ktorý som nastavila je *C=50.0*, čo určuje cenu za chybu. Pre *kernel="rbf"* a *C=50.0* som dostala skóre 98% a cross validačné skóre 93%.

2. **Random Forest Classifier**
   V tejto metóde som nastavovala parametre *n_estimators* a *max_depth*. Prvý parameter určuje počet stromov v lese, druhý parameter určuje maximálnu hĺbku každého stromu. Druhý parameter je nastavený na 500. Ak by som nenastavila prvý parameter, tak je klasifikácia veľmi rýchla, ale menej presná, skóre bolo 93%, cross validačné skóre bolo iba 85%. Ak som ale nastavila *n_estimators=200*, tak skóre sa zlepšilo na 96% a cross validačné skóre na 91%.

3. **Decision Tree Classifier**
   V tejto metóde som nezadávala žiadne voliteľné parametre. Nie je úplne najpresnejšia v porovnaní s ostatnými metódami, ale je veľmi rýchla. Bez žiadnych parametrov má skóre 86% a cross validačné skóre má iba 77%. Táto metóda nie je evidentne úplne vhodná na túto multi klasifikáciu.

4. **Logistic Regression**
   Táto metóda nie je priveľmi vhodná na tento typ problému. Zvolila som si ju ale pre porovnanie s ostatnými. Ako voliteľný parameter som zadala *C=20.0*, čo znamená podobne ako pri *SVC* cenu za chybu. Skóre úspešnosti je len 72% a cross validačné skóre je tiež len 71%. Je ale zaujímave, že táto metóda je jediná, ktorá má podobné cross validačné skóre ako klasické skóre.

4 Implementácia

Najprv som si stiahla dáta a uložila do súboru *data.csv*. Ďalším krokom bolo navrhnúť si, ako bude vyzerať program. Zvolila som dve triedy, jedna slúži na úpravu datasetu a trénovanie, druhá slúži na vizualizáciu dát.

Prvá trieda *LetterRecognition* dostane názov súboru, kde sú uložené dáta a môže dostať nepovinný paramater, ktorý určuje veľkosť trénovacej a testovacej množiny. Ak parameter nie je zadaný, berie sa 0.75. Zadané číslo musí byť v intervale $\langle 0, 1 \rangle$. V konštruktore si načíta dáta a podľa druhého parametra rozdelí dáta na trénovaciu a testovaciu množinu. Potom už len rozdelí obidve množiny na X a y, pričom X obsahuje atribúty definujúce písmeno a y obsahuje názvy tried. Jediná nenumerická hodnota boli názvy tried. Mám 26 tried, každá trieda pre jedno písmeno anglickej abecedy. Písmena som nahradila číslami od 0 do 25 a až v takomto tvare som ich pridala do y pre trénovaciu aj testovaciu množinu.

Ďalším krokom bolo natrénovať jednotlivé metódy na použitom trénovacej množine a otestovať na testovacej množine. Každá metóda sa nachádza

vo svojej metóde v triede. Metódy sú navrhnuté tak, aby sa najprv natrénovali na trénovacej množine a potom predikovali y na testovacej množine. Každá metóda vypíše skóre úspešnosti a cross validačné skóre na testovacej množine. V každej metóde som si ukladala predikované hodnoty a originálne hodnoty do súborov pre porovnanie. Všetky metódy vracajú svoj natrénovaný klasifikátor. Obyčajné skóre a cross validačné skóre sú ďalšie dve metódy v triede, ktoré berú ako parameter predikovaný model. Vrátia príslušné skóre na testovacej množine.

Posledná časť slúži na vykresľovanie grafov. Tu používam druhú triedu *Visualisation*, kde sú pripravené metódy na zobrazenie grafov. Prvý graf používa metódu PCA na redukciu dimenzií a zobrazuje ešte nenatrénované dáta. Metóda používa dve dimenzie zredukované z PCA. Táto metóda sa zavolať dvoma spôsobmi, jeden zobrazí trénovacie dáta a druhý testovacie. Druhý graf zobrazuje *ROC (Receiver Operating Characteristic)*, ktorý vyhodnocuje úspešnosť klasifikátorov. Táto metóda sa používa pri *Support Vector Classifier* a *Logistic Regression*. Metóda dostane ako parameter klasifikátor, ktorý použije a vykreslí krivku. Druhý parameter je voliteľný, v prípade, že je zadaný, tak sa vykreslí *ROC* krivka iba pre jednu triedu, teda pre konkrétne písmeno, ktoré je zadané. V prípade, že sa parameter nezadá, vykreslia sa *ROC* krivky pre všetky triedy do jedného grafu.

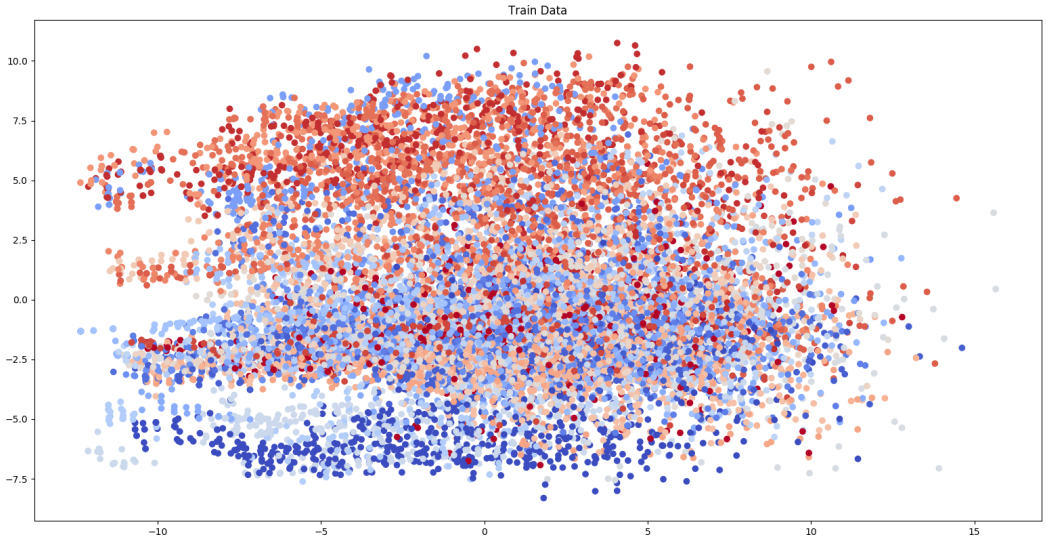
Používanie programu je celkom intuitívne, stačí si vytvoriť inštanciu triedy *LetterRecognition* a na nej volať jednotlivé metódy. Dajú sa volať metódy na zobrazenie grafov. Ďalšími metódami sú klasifikačné metódy, ktoré vypíšu skóre a cross validačné skóre a vrátia klasifikátor. Po uložení klasifikátora môžme zavolať metódu na vykreslenie *ROC* krivky, ktorá dostane ako parameter klasifikátor, prípadne ako druhý parameter môže dostať konkrétne písmeno, pre ktoré sa vykreslí *ROC* krivka.

Program je naimplementovaný v jazyku Python. Použila som knižnice na spracovanie dát, jednotlivé klasifikačné metódy a metódy na vykresľovanie grafov. Najviac využívané knižnice boli určite *numpy* a *pandas*, tie som využívala na prácu s dátami. S knižnicami na trénovanie modelu sa pracovalo celkom intuitívne, horšie bolo vykreslenie *ROC* krivky na multi klasifikáciu.

5 Vyhodnotenie

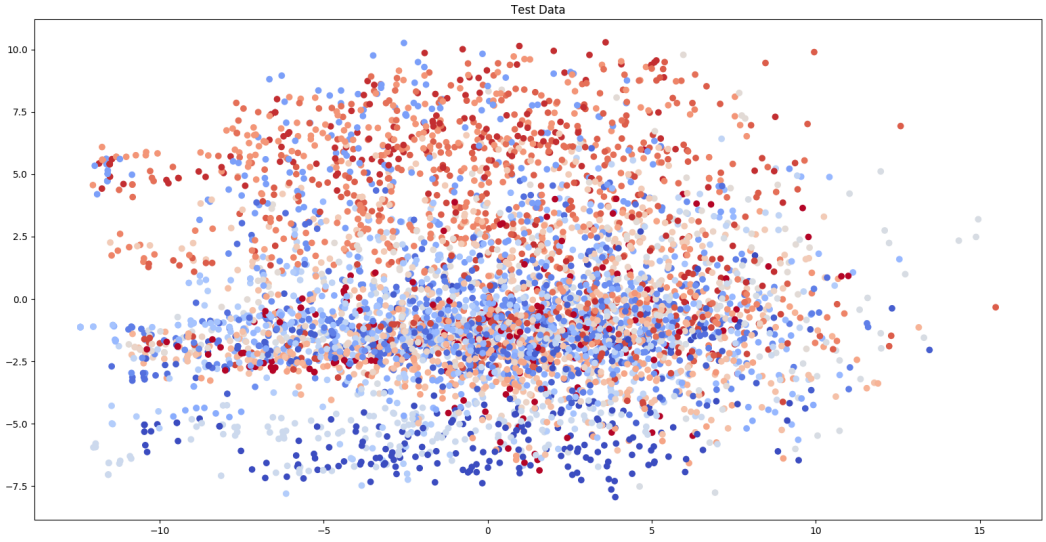
Pre vyhodnotenie sú dáta rozdelené v pomere 0.75 na trénovaciu a testovaciu množinu. Najprv som spravila grafy na zobrazenie trénovacích a testovacích dát. Keďže dáta nie sú len dvojrozmerné, pomocou PCA metódy som si vybrala dva najdôležitejšie komponenty, z ktorých robím graf.

Pre trénovaciu množinu vyzerá graf nasledovne:



Obr. 2: Trénovacie dáta

Pre testovaciu množinu vyzerá graf takto:

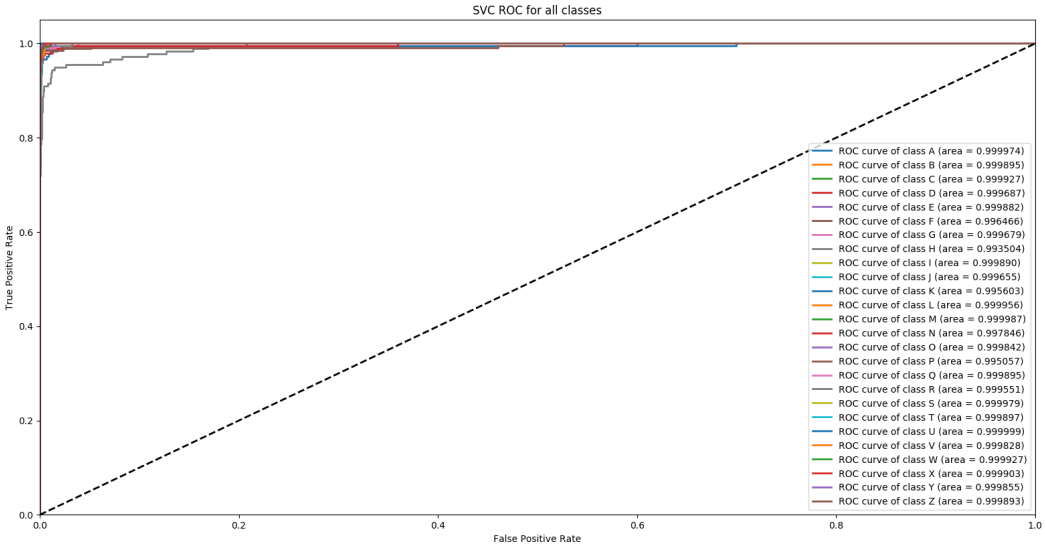


Obr. 3: Testovacie dáta

Po natrénovaní metód na trénovacej množine som ich testovala na testovacej množine. Z metód: *Support Vector Classifier* a *Logistic Regression* som

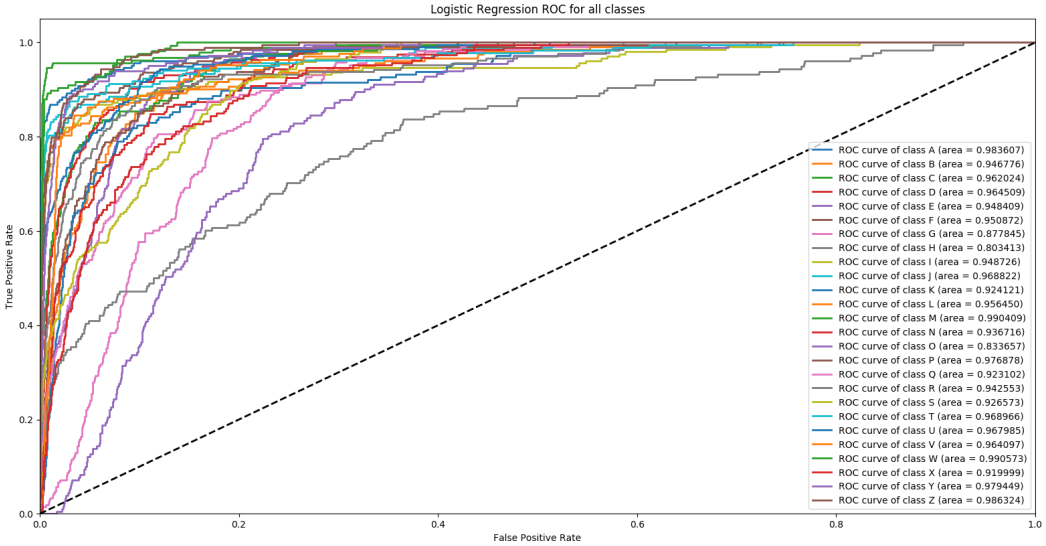
vytvorila *ROC* krivku na určenie kvality natrénovania klasifikátora.

Výsledkom metódy *Support Vector Classifier* sú takéto *ROC* krivky:



Obr. 4: ROC pre Support Vector Classifier

Výsledkom metódy *Logistic Regression* sú takéto *ROC* krivky:



Obr. 5: ROC pre Logistic Regression

Pri *ROC* krivke kvalita natrénovania klasifikátora sa nachádza v intervale $\langle 0, 1\rangle$. Čím bližšie k 1 sa area krivky nachádza, tým lepšie je klasifikátor natrénovaný. Pre obe metódy sú vytvorené grafy, ktoré zobrazujú krivky pre všetky triedy v jednom grafe.

Z prvého grafu je jasne vidieť, že táto metóda má vysokú úspešnosť a viditeľne iba v jednej triede je trochu horšia. Keď sa ale pozrieme na legendu, tak zistíme, že je to trieda H, ktorá má areu rovnú 0.993504, čo je stále vynikajúce skóre.

Výsledkom metódy *Logistic Regression* je výrazne horší graf. Stále sa všetky krivky nachádzajú v ľavom hornom rohu, čo je dobre, ale oproti predchádzajúcej metóde je jasne vidieť, že táto metóda klasifikuje príklady horšie.

Ešte som porovnávala metódy podľa času trénovania. Spravila som 3 merania pre každú metódu. V nasledujúcej tabuľke sú zobrazené časy trvania a ich priemer pre všetky metódy:

	SVC	RFC	DTC	LR
1.	22.5246	9.4536	0.1883	15.0372
2.	22.4552	9.4317	0.1840	12.9366
3.	22.6087	8.9994	0.1798	12.9654
Priemer	22.5295	9.2949	0.1840	13.6464

Vo všetkých metódach sú časy veľmi podobné, až na metódu *Logistic Regression*, kde sa jeden čas líši až o viac ako dve sekundy.

6 Záver

Najprv som si upravila dáta tak, aby boli všetky iba numerické. Potom som si dáta rozdelila na trénovaciu a testovaciu množinu v zadanom pomere. Ak pomer nebol zadaný, defaultom je 0.75. Ďalej som implementovala štyri metódy na multiklasifikáciu, *Support Vector Classifier*, *Random Forest Classifier*, *Decision Tree Classifier* a *Logistic Regression*. Potom som implementovala vizualizáciu údajov. Prvý graf zobrazuje dáta zredukované pomocou PCA na dva komponenty. Druhý graf zobrazuje *ROC* krivky pre všetky triedy dát. Tento graf sa dal využiť pri zobrazovaní kvality natrénovanie klasifikátorov pri metódach: *Support Vector Classifier* a *Logistic Regression*.

Vyhodnotila som dáta nasledovne. Najprv som zobrazila grafy na ukázanie trénovacích a testovacích dát. Po natrénovaní klasifikátorov som zobrazila grafy, ktoré vyjadrujú *ROC* krivky pre všetky triedy pri metódach *Support Vector Classifier* a *Logistic Regression*. Potom som ešte vyhodnocovala jednotlivé metódy podľa času trénovania.

Z grafov sme mohli jasne vidieť, že prvá metóda klasifikovala dáta najlepšie, ale zároveň bola najpomalšia. Druhá metóda bola podobná prvej, ale zároveň bola dvojnásobne rýchlejšia ako prvá metóda. Tretia metóda netrvala ani sekundu, ale bola už dosť nepresná a štvrtá metóda trvala dlhšie než druhá a bola horšia ako tretia. Najlepšou metódou na multiklasifikovanie tohto datasetu je teda prvá metóda *Support Vector Classifier*. Najrýchlejšou a zároveň úspešnou metódou je druhá metóda *Random Forest Classifier*.

Celá implementácia aj s dokumentáciou sa nachádza na githube: `https://github.com/katuskaa/letterRecognition`.