Metody odkrywania wiedzy
Dokumentacja końcowa

Piotr Jastrzębski
Marcin Nazimek

1 Temat projektu

16. Proste algorytmy klasyfikacji tekstu (TF-IDF, naiwny klasyfikator bayesowski, k-NN). Porównania ze standardowymi algorytmami klasyfikacji dostępnymi w R.

2 Szczegółowa interpretacja tematu projektu

Projekt zakładał wykorzystanie przez nas zbioru danych do analizy algorytmów klasyfikacji tekstów. Zbiór danych, który wykorzystaliśmy: *Spambase Data Set*, dostępny jest pod adresem [1]. Zawiera on 4601 instancji danych wiadomości e-mail zebranych w roku 1999. Wartości w nim zgromadzone, na których odbywała się klasyfikacja, należą zarówno do liczb rzeczywistych jak i całkowitych. Każda próbka danych zawiera dodatkowo decyzję klasyfikacyjną 1 – kiedy e-mail jest spamem albo 0, gdy nie jest. Charakter danych nie jest bardzo istotny dla samych algorytmów klasyfikacji, ale pozwala, w przypadku ich opracowywanie na łatwiejsze tropienie błędów, gdy wyniki są różne od spodziewanych.

Dane zgromadzone z każdej wiadomości to oprócz przynależności do zbioru spamu także:

- 48 liczb rzeczywistych $[0, 100]$ typu *word_freq_WORD* – odsetek słów w e-mailu, które pasują do klasy *WORD*
- 6 liczb rzeczywistych $[0, 100]$ typu *char_freq_CHAR* – odsetek znaków w e-mailu, które pasują do klasy *CHAR*
- 1 liczba rzeczywista $[1, ...]$ typu *capital_run_length_average* – średnia długość nieprzerwanej sekwencji wielkich liter
- 1 liczba całkowita $[1, ...]$ typu *capital_run_length_longest* – długość najdłuższej ciągłej sekwencji wielkich liter

- 1 liczba całkowita $[1, ...]$ typu *capital_run_length_total* – sumaryczna długość ciągłych sekwencji wielkich (całkowita liczba wielkich liter w e-mailu)

Klasa *WORD* zdefiniowana jest jako słowa: *make, address, all, 3d, our, over, remove, internet, order, mail, receive, will, people, report, addresses, free, business, email, you, credit, your, font, 000, money, hp, hpl, george, 650, lab, labs, telnet, 857, data, 415, 85, technology, 1999, parts, pm, direct, cs, meeting, original, project, re, edu, table, conference.* Stosunek ich liczności do liczby słów w całym tekście będzie użyty m.in przez naiwny klasyfikator bayesowski jako prawdopodobieństwo wystąpienia danego słowa.

3 Szczegółowy opis wykorzystanych algorytmów

Do klasyfikacji wykorzystane zostały przedstawione poniżej algorytmy. Są to algorytmy zaimplementowane przez nas samodzielnie jak i te, które dostarczone są jako pakiety do środowiska języka R.

3.1 Algorytmy własne

3.1.1 Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski oparty jest na założeniu o wzajemnej niezależności predyktorów. Często nie mają one żadnego związku z rzeczywistością i właśnie z tego powodu nazywa się je naiwnymi. Bardziej opisowe jest określenie — „model cech niezależnych". Ponadto model prawdopodobieństwa można wyprowadzić korzystając z twierdzenia Bayesa. W zależności od rodzaju dokładności modelu prawdopodobieństwa, naiwne klasyfikatory bayesowskie można skutecznie „uczyć" w trybie uczenia z nadzorem. W wielu praktycznych aplikacjach, estymacja parametru dla naiwnych modeli Bayesa używa metody maksymalnego prawdopodobieństwa a posteriori, czyli można pracować z naiwnym modelem Bayesa bez wierzenia w twierdzenie Bayesa albo używania jakichś metod Bayesa. Pomimo ich naiwnego projektowania i bardzo uproszczonych założeń, w wielu rzeczywistych sytuacjach naiwne klasyfikatory Bayesa często pracują dużo lepiej, niż można było tego oczekiwać.

Na bazie tych założeń można wyprowadzić ogólny wzór na prawdopodobieństwo, że dokument e-mail należy do kategorii spam (S). Został on przedstawiono wzorem 1.

$$p(S|F_1, ..., F_{57}) = p(S)\Pi_{i=1}^{57} p(F_i|S) \tag{1}$$

, gdzie S to decyzja, że wiadomość to spam (gdy $S = 1$, oraz, że jest to normalna wiadomość, gdy $S = 0$), a $F_1, ..., F_{57}$ to 57 cech każdej wiadomości wymienionych w punkcie 2. Często przyjmuje się dla analizy spamu, że

$p(S) = p(\neg S) = \frac{1}{2}$, gdyż zakładamy, że prawdopodobieństwo, że wiadomość jest spamem jest równe temu, że nim nie jest.

Dalej prawdopodobieństwo wg. Bayesa, że próbka jest spamem określone zostaje wzorem 2, a że nim nie jest wzorem 3. Są to wcześniej wspomniane prawdopodobieństwa a posteriori.

$$posteriori(S) = \frac{p(S)p(F_1|S)...p(F_{57}|S)}{evidence} \tag{2}$$

$$posteriori(\neg S) = \frac{p(\neg S)p(F_1|\neg S)...p(F_{57}|\neg S)}{evidence} \tag{3}$$

Wartością z mianownika *evidence* jest suma prawdopodobieństw określonych wzorem 4.

$$evidence = p(S)p(F_1|S)...p(F_{57}|S) + p(\neg S)p(F_1|\neg S)...p(F_{57}|\neg S) \tag{4}$$

Dla wartości ciągłych, gdy nie jest rozważane jedynie spełnienie lub niespełnienie cechy, a jej wartość, prawdopodobieństwo warunkowe przy założeniu, że próbka jest danej klasy określone jest wzorem 5. Jest ono wykorzystywane bezpośrednio w *evidence*.

$$p(F_i|S) = \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\left(\frac{-(x_i - \mu_i)^2}{2\sigma_i^2}\right) \tag{5}$$

Jako, że zbiór danych spełnia właśnie takie założenia, bazując na doświadczeniu zgromadzonym podczas przedmiotu ROB postaraliśmy się zaimplementować naiwny klasyfikator bayesowski samodzielnie. Punktem wyjścia był przykład [4], który pozwolił w sposób szybki przygotować rozwiązanie.

Warto zauważyć, że decyzja klasyfikacyjna bazuje na porównaniu wartości prawdopodobieństw a posterior 2 i 3. Jako, że część $p(S)$ oraz *evidence* jest wspólna dla obu, czynniki te zostały pominięte w implementacji.

3.1.2 Algorytm najbliższych sąsiadów

Algorytm k-NN (K Nearest Neighbours[1]) pozwala na wyznaczanie decyzji klasyfikacyjnej bazując na porównaniu wartości cech z całej dziedziny zbioru uczącego i ustalenie decyzji klasyfikacyjnej na podstawie klasy najbliższej (dla 1-NN) lub najbliższych (k-NN) znanych uprzednio sklasyfikowanych próbek. Definicja „najbliższych obserwacji" sprowadza się do minimalizacji pewnej metryki, mierzącej odległość pomiędzy wektorami zmiennych dwóch obserwacji. Zbiór uczący zawiera obserwacje, z których każda ma przypisany wektor zmiennych (w naszym przypadku o długości 57) oraz wartości oznaczającej klasę (dla analizy e-maili jest to 0 albo 1). Zarówno 1-NN jak i k-NN mają swoje mocne i słabe strony ujawniające się w procesie klasyfikacji.

[1] ang. K najbliższych sąsiadów

1-NN Zaletami klasyfikacji 1-NN są:

- Bezpośrednie użycie zbioru uczącego
- Małe prawdopodobieństwo błędu
- Prosta implementacja

Zaś do wad można zaliczyć:

- Duże koszty obliczeniowe dla dużych N
- Czułość na błędnie sklasyfikowane egzemplarze
- "losowe" zachowanie przy granicach klas

k-NN Metoda k-NN klasyfikuje próbki poprzez większościowe głosowanie K najbliższych sąsiadów. Wielkość K wpływa na parametry klasyfikacji, należy mieć zatem na uwadze, że:

- K powinno być duże, żeby zminimalizować prawdopodobieństwo błędnej klasyfikacji
- K powinno być małe - w stosunku do liczby próbek - by uzyskać możliwie dokładne granice między klasami

Nasza implementacja k-NN bazuje na znajdowaniu klasy dominującej pośród k najbliższych sąsiadów. Zdecydowaliśmy się na niestosowanie np. metody analizy głównych składowych, gdyż cechy zgromadzone w bazie są z definicji niezależne, a wprowadzenie PCA znacznie skomplikowałoby rozwiązanie. Przyjęliśmy, że k winno być nieparzyste, co w połączeniu z istnieniem tylko dwu klas powoduje uproszczenie klasyfikacji na podstawie decyzji większościowej w sąsiedztwie.

3.1.3 Klasyfikator TF-IDF[2]

Ważenie częstością termów - odwrotna częstość w dokumentach – jedna z metod obliczania wagi słów w oparciu o liczbę ich wystąpień, należąca do grupy algorytmów obliczających statystyczne wagi termów. Każdy dokument reprezentowany jest przez wektor, składający się z wag słów występujących w tym dokumencie. TF-IDF informuje o częstości wystąpienia termów uwzględniając jednocześnie odpowiednie wyważenie znaczenia lokalnego termu i jego znaczenia w kontekście pełnej kolekcji dokumentów. Niestety, mimo dużego wysiłku włożonego w próby wykorzystania tego algorytmu do procesu klasyfikacji nie udało nam się wypracować metody, która pozwoliłaby na uzyskanie decyzji klasyfikacyjnych.

[2]ang. TF – term frequency, IDF – inverse document frequency

3.2 Algorytmy dostępne w R

Dla języka R dostępne są m.in. następujące algorytmy klasyfikacji: *Maszyna wektorów nośnych* [3]*, penalizedSVM, k-NN, Outliers, drzewa decyzyjne, naiwny klasyfikator Bayesa, adaboost* oraz *JRip*. Istotnie, nie napotkaliśmy problemów przy porównaniu wyników algorytmów wbudowanych do tych, przygotowanych i zaimplementowanych przez nasz zespół. Spośród dostępnych algorytmów wybraliśmy poniższe rozwiązania języka R:

3.2.1 Metoda `naiveBayes`

Metoda, na której opiera się ten klasyfikator, została dokładnie wyjaśniona w punkcie 3.1.1.

3.2.2 Metoda `kNN`

Metoda, na której opiera się ten klasyfikator, została dokładnie wyjaśniona w punkcie 3.1.2.

3.2.3 Metoda `randomForest`

Lasy losowe są uznawane za jedną z najlepszych metod klasyfikacji. Pojedyncze klasyfikatory lasu losowego to drzewa decyzyjne. Decyzja jest podejmowana na podstawie głosowania pomiędzy tymi klasyfikatorami. Algorytmy RandomForest bardzo dobrze nadaje się do badania próby, gdzie wektor obserwacji jest dużego wymiaru.

3.2.4 Metoda `classifactionTree`

Drzewa klasyfikacyjne dostępne w pakiecie rpart jest typową implementacją drzewa służącego do określenia klasy danego obiektu na podstawie atrybutów. Drzewo składa się z korzenia (przedstawianego, podobnie jak w matematyce i informatyce, zwykle u góry rysunku) oraz gałęzi prowadzących z korzenia do kolejnych węzłów. W każdym węźle sprawdzany jest pewien warunek dotyczący danej obserwacji, i na jego podstawie wybierana jest jedna z gałęzi prowadząca do kolejnego węzła piętro niżej. Na dole znajdują się liście, w których odczytujemy do której z klas należy przypisać daną obserwację. Klasyfikacja danej obserwacji polega na przejściu od korzenia do liścia i przypisaniu do tej obserwacji klasy zapisanej w danym liściu.

[3] ang. Support Vector Machine (SVM)

4 Opis stosowanej procedury eksperymentalnej

4.1 Pytania, na które była poszukiwana odpowiedź

Jak przy każdym zadaniu klasyfikacyjnym o jej jakości decydują czynniki zależne od charakteru danych. Częstokroć jest tak, że nie najlepszy jest algorytm najszybszy, ale decyduje o tym, często różnie definiowana, jakość klasyfikacji. O ile prędkość jest wartością łatwomierzalną i jednoznaczną to jakoś klasyfikacji jest trudniejsza do zdefiniowania. W pracy nad poprawą jakości przydatna była też analiza macierzy pomyłek, gdyż podstawowym pytaniem, na które szukaliśmy odpowiedzi było, które klasyfikatory dają najlepsze decyzje klasyfikacyjne oraz jak parametry algorytmów wpływają na tę jakość. Charakterystyka zbioru wykorzystanego w tym celu opisana została w punkcie 4.2 a dobór parametrów w punkcie 4.3.

4.2 Charakterystyka zbiorów danych, które zostały wykorzystane

Ze zbioru wejściowego wydzieliliśmy stały zbiór testowy o rozmiarze 50 elementów (25 przykładów pozytywnych i tyle samo negatywnych) pozostawiając zbiór uczący z 4551 próbkami.

4.3 Parametry algorytmów, których wpływ na wyniki był badany

Podczas badania jakości klasyfikacji nie mieliśmy w praktyce dużej możliwości wpływu na parametry klasyfikatorów. Największą zmianę i dowolność konfiguracji zapewnia klasyfikator k-NN, dla którego zmiana parametru k może wpływać na końcowe decyzje. Postanowiliśmy przetestować jakość klasyfikacji dla k równego $1, 3, 5$ oraz 7.

4.4 Sposób oceny jakości modeli

Tabela 5 przedstawia wyniki zgromadzone w procesie testowania jakości algorytmów. Ze zbioru wejściowego wydzieliliśmy stały zbiór testowy o rozmiarze 50 elementów (25 przykładów pozytywnych i tyle samo negatywnych) pozostawiając zbiór uczący z 4551 próbkami. Tabela przedstawia przykłady zakwalifikowane jako TP (*True Positive*), FP (*False Positive*), TN (*True Negative*) oraz FN (*False Negative*). Policzono także miary jakości klasyfikacji i zaufania wyników – *precision*, zgodnie ze wzorem 6, *recall* (wzór 7), *trueNegativeRate* (wzór 8) oraz *accuracy* (wzór 9, będących wymierną miarą jakości klasyfikacji i poziomu zaufania).

$$precision = \frac{TP}{TP + FP} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

$$trueNegativeRate = \frac{TN}{TN + FP} \tag{8}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

5 Wyniki

	TP	FP	FN	TN	precision	recall	TNR[4]	accuracy
1-NN	21	4	8	17	0.84	0.72	0.81	0.76
[R] 1-NN	22	3	8	17	0.88	0.73	0.85	0.78
3-NN	21	4	6	19	0.84	0.78	0.83	0.80
[R] 3-NN	21	4	6	19	0.84	0.78	0.83	0.80
5-NN	21	4	7	18	0.84	0.75	0.82	0.78
[R] 5-NN	21	4	7	18	0.84	0.75	0.82	0.78
7-NN	20	5	4	21	0.80	0.83	0.81	0.82
[R] 7-NN	20	5	5	20	0.80	0.80	0.81	0.82
naiwny Bayes	24	7	1	18	0.77	0.96	0.72	0.84
[R] naiveBayes	23	12	2	13	0.66	0.92	0.52	0.72
[R] randomForest	25	0	0	25	1.00	1.00	1.00	1.00
[R] classifactionTree	21	2	4	23	0.91	0.84	0.92	0.88

Tabela 1: Wyniki klasyfikacji zbioru przy pomocy różnych algorytmów

6 Dyskusja wyników i wnioski

Analiza macierzy pomyłek sama w sobie nie wskazuje jakości klasyfikacji poszczególnych algorytmów. Z tego też względu zostały zastosowane uprzednio opisane współczynniki wyników klasyfikacji.

Najbardziej intuicyjny wydaje się współczynnik *accuracy* będący w praktyce odsetkiem poprawnie sklasyfikowanych wiadomości będących spamem i tych nim nie będących. Można przyjąć, że uznamy go za najważniejszy parametr jakości.

Algorytmy wbudowane R cechują się wysoką jakością klasyfikacji. Spośród nich prym wiedzie `randomForest`, który dla naszego zbioru testowego poprawnie sklasyfikował wszystkie próbki. Zaskakujące jest, że `naiveBayes`, który w praktyce jest naiwnym klasyfikatorem bayesowskim dał wyniki nieco gorsze niż klasyfikator Bayesa, który zaimplementowaliśmy sami. Podobna

sama sytuacja - tj. porównywalne wyniki naszego rozwiązania i wyników klasyfikatorów zaimplementowanych w języku R miała miejsce z algorytmami k-NN.

Dość miernie zaprezentował się klasyfikator k-NN, który przy tej liczbie wymiarów działał widocznie wolniej, a dodatkowo dawał wyniki klasyfikacji dużo słabsze niż konkurenci (najlepszy spośród k-NN: 7-NN miał parametr *accuracy* na poziomie 0.82).

Zważywszy, że klasyfikacja k-NN wiąże się to z każdorazową analizą 7 najbliższych próbek sensowne wydaje się rozważenie użycia 3-NN, który zapewnia niewiele gorsze wyniki biorąc pod uwagę tylko 3 sąsiadów (0.80).

W naszym zbiorze danych mamy do czynienia z wektorem kilkudziesięciu cech. W tym przypadku początkowo nie byliśmy w stanie ocenić, które cechy są istotne -– tzn. rozróżniają dobrze poszczególne klasy, a które nie. Podejrzewaliśmy także, że cechy nie są niezależne. Przy dużej liczbie wymiarów przestrzeni cech konieczny jest odpowiednio duży zbiór uczący. Złożoność obliczeniowa także rośnie zastraszająco z liczbą wymiarów. Wniosek: dobrze byłoby zmniejszyć liczbę wymiarów. Rozważaliśmy wykorzystanie metody analizy głównych składowych[5], w celu zmniejszenia wymiarowości przestrzeni cech ale końcowa decyzja wskazuje, że nie było to konieczne.

Literatura

[1] *Spambase Data Set*
`http://archive.ics.uci.edu/ml/datasets/Spambase`

[2] *R Documentation*
`http://www.r-project.org/other-docs.html`

[3] *tm – Text Mining Package*
`http://tm.r-forge.r-project.org`

[4] *Wzór na naiwny klasyfikator bayesowski*
`http://en.wikipedia.org/wiki/Naive_Bayes_classifier#Sex_classification`

[5] *Wikipedia*
`http://en.wikipedia.org/wiki/{Tfidf|k-NN|Naive_Bayes_classifier|Text_mining|Document_classification|PCA}`

[5] ang. Principal Component Analysis, PCA