Proste algorytmy klasyfikacji tekstu

Raport końcowy

Iryna Mikutskaya Bakhmut Maksym

Styczeń 2017

1. **Opis zadania**

W ramach zadania - ”Proste algorytmy klasyfikacji tekstu (naiwny klasyfikator Bayesowski, kNN). Porównania ze standardowymi algorytmami klasyfikacji dostepnymi w R.” zaimplementowano 2 wymienione klasyfikatory które nastepnie zostały wykorzystane w zagadnieniu rozpoznawania spam wiadomosci SMS.

Do nauki i oceny klasyfikatorów wykorzystano zbiór *SMS Spam Collection* z repozytorium UCI.

1. **Opis algorytmów**

W sekcji znajduje sie opis wykorzystywanych algorytmów. W stosunku do informacji zawartych w dokumentacji wstepnej, opis został rozszerzony o podstawowe informacje i decyzje zwiazane z implementacja. Zgodnie z wymaganiami, algorytmy zostały zaimplementowane w języku R.

* 1. **Naiwny klasyfikator Bayesa**

Naiwny klasyfikator Bayesa to skuteczne narzędzie **kategoryzacji**. Klasyfikator automatycznie przydziela przykłady do kategorii. Kategorie muszą być z góry ustalone.

Klasyfikator bayesowski to wzór na prawdopodobieństwo, że przykład x należy do kategorii k. Dla danego przykładu x obliczamy prawdopodobieństwo przynależności kolejno do każdej kategorii. Następnie wybieramy najbardziej prawdopodobną kategorię.

Wzór naiwnego klasyfikatora Bayesa:



Celem klasyfikacji jest zrozumienie do jakiej kategorji należy przykład, w związku z tym nam jest potrzebne nie samo prawdopodobieństwo, a najbardziej prawdopodobna kategoria. Klasyfikator bayesowski wykorzystuje ocenę *maximum a posteriori estimation* dla odnalezienia najbardziej wiarygodnej kategorii (klasy). Krótko mówiąc, to klasa z maksymalną wiarygodnością (prawdopodobieństwem).

Czyli musimy obliczyć prawdopodobieństwo dla wszystkich klas i wybrać tę klasę, która ma maksymalne prawdopodobieństwo.

Załużmy, że mamy zbiór obiektów *D =* {*d1, d2 ..., dm*}*,* każdy z których ma jakiś zestaw cech ze zbioru wszystkich cech *F=*{*f1, f2 ..., fq*}, a także należy do jednej z kilku klas *C =* {*c1, c2 ..., cr*}*.* Naszym zadaniem jest znalezienie najbardziej wiarygodnej klasy wchodzącego obiektu *d*, opierając się na zestaw jego cech *Fd* = {f*d*1, f*d*2 ..., f*d*n}. Innymi słowami, musimy znaleźć taką wartość zmiennej *C*, przy pomocy której będzie osiągnięty aposterioryczny maksimum (maximum a posteriori probability, MAP).

Naszym celem jest:



Zgodnie z twierdzeniem Bayesa otrzymamy:



Ze względu na to, że szukamy argument, który zmaksymalizowałby funkcję prawdopodobieństwa, i to, że mianownik (prawdopodobieństwo przykładu) nie zależy od tego argumentu i jest w tym wypadku stałą, i w żaden sposób nie okazuje wpływu na klasy, możemy śmiało skreślić wartość pełnego prawdopodabieństwa *P*(*d*)

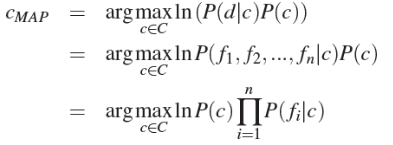


Model naiwnego klasyfikatora Bayesowskiego przyjmuje dwa dopuszczenia, dlatego i jest taki naiwny:

* porządek, w jakim ustawione są cechy obiektu, nie ma znaczenia;
* niezależność prawdopodobieństwa kolejnych cech – „model cech niezależnych”:



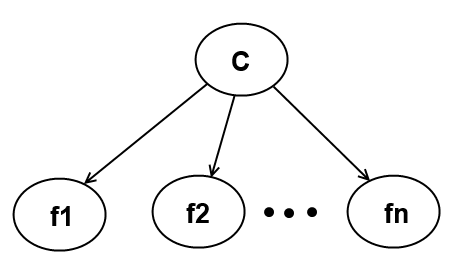
To jest akurat skutek stosowania dopuszczeń:



Tutaj stosuje się świetna właściwość logarytmu, co pozwala nam uniknąć straty precyzji przy operowaniu bardzo małymi wartościami:



Możemy przedstawiać graficzny model naiwnego klasyfikatora Bayesowskiego w ten sposób:



**2.2 Algorytm *k-NN***

Algorytm k-najbliższych sąsiadów (k-NN) jest metodą nie parametryczną, stosowaną do klasyfikacji i/lub regresji. W obu przypadkach, wejście składa się z k najbliższych przykładów szkoleniowych w przestrzeni. Wynik wyjściowy zależy od tego, czy K-NN stosuje się klasyfikacji lub regresji.

Będziemy rozpatrzywali sitosowanie algorytmu k-NN w klasyfikacji.

W klasyfikacji, wyjściem jest przynależność przykładu do klasy (predykcja). Obiekt zostaje sklasyfikowany większością głosów swoich sąsiadów, a obiekt jest przypisany do klasy najczęściej spotykanej wśród swoich k najbliższych sąsiadów (k jest dodatnią liczbą całkowitą, zazwyczaj małe). Jeśli k = 1, po czym przedmiot jest po prostu przypisana klasie tego jednego najbliższego sąsiada.

k-NN jest rodzajem lazy learning, w którym funkcja jest aproksymowana tylko lokalnie, a wszystkie obliczenia odłożone są do momentu klasyfikacji. Algorytm k-NN jest jednym z najprostszych  algorytmów uczenia maszynowego.

Sąsiedzi są pobierane z zestawu obiektów, dla których klasa (sklasyfikowane k-NN) lub wartość właściwości obiektu (dla regresji k-NN) jest znana. To może być traktowane jako zestaw szkoleniowy dla algorytmu, choć nie jest wymagany żadny wyraźny etap szkolenia.

Najbliższe sąsiedzi są wyszukiwane przez obliczenia metryk dla danego przykładu ze zbioru testującego z każdym przykładem z modeli, wykorzystanej dla klasyfikacji. Te metryki pokazują stopień podobieństwa między dwoma przykładami.

W danym projekcie były wykorzystane metryki Euklidowa oraz kosinusowa.

Metryka Euklidesowa:

,

gdzie v2, v1 – przykłady, dla których szukamy normę.

Metryka kosinusowa:

**3 Opis eksperymentów**

Celem zadania było odniesienie (określenie klasy) przekazanej wiadomości SMS do jednej z klas: „spam” albo „ham”.

**3.1 Charakterystyka zbioru danych**

Danymi do uczenia, testowania i oceny klasyfikatorów był wybrany zbiór SMS Spam Collection archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection.

Wybrany zbiór danych jest zbiorem etykietowanych wiadomości które zostały zebrane na potrzeby badania spamu wysyłanego SMS'ami. Zbiór składa się z 4827 zwykłych wiadomości oraz z 747 wiadomości zawierających spam. Cała kolekcja wiadomości jest umieszczona w jednym pliku, wśród nich występują zestawy wiadomości oznaczonych jako niepożądane ("spam") oraz zestawy wiadomości pożądanych ("ham").

Ponizej kilka przykładów:

* ham Why are u up so early?
* ham Ya that one is slow as poo
* spam Bloomberg -Message center +447797706009 Why wait? Apply for your future http://careers. bloomberg.com
* ham "Im on gloucesterroad what are uup to later?"
* ham Yes:)here tv is always available in work place..
* spam YES! The only place in town to meet exciting adult singles is now in the UK. Txt CHAT to 86688 now! 150p/Msg.
* ham Lol no ouch but wish i'd stayed out a bit longer

**3.2 Wstęp do implementacji algorytmu naiwnego klasyfikatora Bayesa**

Dla zrozumienia działania tego algorytmu obejrzyjmy na początku prosty przykład.

Zbiór *D* składa się z wiadomości SMS. Każda wiadomość ma etykietę z C = {ham, spam}. Dla sformułowania pojęcia cechy będziemy wykorzystywali model przedstawienia bag of words (bag-of-words model**)**, pokażemy to na przykładzie: mamy w zbiorze tylko 2 wiadomości *ham*:

* hi how are you
* how old are you

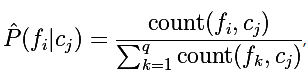
Możemy zbudować tabelę:

|  |  |
| --- | --- |
| *Tabela 1* | |
| Słowo | Częstość wystąpienia |
| hi | 1 |
| how | 2 |
| are | 2 |
| you | 2 |
| old | 1 |

Tylko 8 słów w korpusie nie-SPAM wiadomości. Po normalizacji otrzymamy aposterioryczne prawdopodobieństwo słowa, wykorzystując maximum likelihood estimation (metodę maksymalnego prawdopodobieństwa). Dla przykładu wiarygodność słowa "how", pod warunkiem, że wiadomość nie jest *spamem*, będzie taka:

*P(fi = «how» | C = ham) = 2/8 = ¼*

Albo możemy zapisać tę metodę w ogólnym wyglądzie:



gdzie q — ogólna ilość unikalnych słów w słowniku.

Musimy zwrócić uwagę na następny problem. Mamy zbiór z dwóch *ham* wiadomości. Do klasyfikacji przyszła nowa wiadomość: "hi bro". Dopuszczalnie, apriorystyczne prawdopodobieństwo *nie spam* P (*ham*) = 1/2. Wtedy prawdopodobieństwo słów:

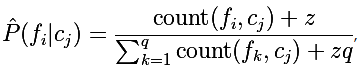
1. *P(«hi» | ham) = 1/8*
2. *P(«bro» | ham) = 0/8 = 0*





Otrzymamy albo błąd, albo nieskończoność z minusem, ponieważ logarytm od zera nie istnieje. Gdybyśmy nie wykorzystali logarytmowanie, to otrzymalibyśmy prosto 0, tzn. prawdopodobieństwo tej wiadomości równałoby się zeru, co w zasadzie dużej korzyści nam nie daje.

Uniknąć tego pozwala rozmywanie Laplace’a czy k-additive smoothing — ta metoda pozwala robić rozmywanie przy obliczeniu wiarygodności kategorii danych. W naszym wypadku to będzie wyglądało w ten sposób:



gdzie *z* >= 0 – współczynnik rozmywania, a *q* – w naszym wypadku to ilość słów w klasie; a *q* – ogólna ilość słów, które były wykorzystane przy nauczaniu modelu.

Przyjmijmy, że przy przeczytaniu *ham* i *spam* wiadomości znaleźliśmy 10 unikalnych słów, wtedy P ("hi" | ham) = (1 + 1) / (8 + 1\*10) = 2/18 = 1/9 przy współczynniku rozmywania *z* = 1. A zerowe prawdopodobieństwo przestaje być takim: P ("bro" | ham) = (0 + 1) / (8 + 1\*10) = 1/18.

Z punktu widzenia Bayesa, ta metoda odpowiada matematycznemu oczekiwaniu aposteriorycznego rozkładu, wykorzystując jako apriorystyczny rozkład — rozkład Dirichleta (Dirichlet distribution), z parametrem *z*.

**3.3 Implementacja algorytmu naiwnego klasyfikatora Bayesa**

W ramach wstępnego przetwarzania danych, zostały wykonane następujące kroki:

* zmniejszyliśmy tekst do małych liter;
* usunęliśmy wszystkie znaki interpunkcji;
* wszystkie ciągi liczbowe zamieniliśmy jedynką.

Open\_data <- function(fileName = "E:/10/Data/SMSSpamCollection")

Wynikiem funkcji stało:

data1 – nasz przekształcony zbiór danych, każda wiadomość to ciąg oddzielnych słów, która jest zaznaczona etykietą *spam* lub *ham*

data1$klasa – etykiety każdej wiadomości ze zbioru

data1$data – ciąg słów każdej wiadomości ze zbioru

Następna funkcja dzieli nasz zbiór danych na części w odpowiednich proporcjach, generując tym samym indeksy zbiorów danych takich jak treningowy i testowy.

New\_dataset <- function(dataSet, proportions = c(0.7, 0.3))

Wynikiem jest zbiór indeksów wiadomości data2, który składa się z 2 podzbiorów – train i test (data2$train, data2$test). Każdy podzbiór zawiera w sobie zarówno jak ham, tak i spam wiadomości.

Tutaj dzielę zbiór w proporcji 0.7, 0.3, ale w ostatniej części kodu można będzie wpisać inne wartości podziału. Wpisane wartości będą przekazane do powyższej funkcji.

W następnej funkcji z wejściowego zbioru danych tworzymy model:

Nasz\_model <- function(data, laplaceFactor = 0)

Ostatnia funkcja, odnosząca się do modelu, klasyfikuje wchodzące SMS'y, wykorzystując nasz model.

Klasyfikacja wiadomości:

Klasyfikacja <- function(s, model, preprocess = T)

Do testowania modelu na zbiorze będzie wykorzystana następna funkcja.

Testowanie modelu:

Model\_testowy<- function(data, model)

{

errors <- 0

for(i in 1:length(data$klasa))

{

prawdopodobna\_klasa<-Klasyfikacja(data$data[i], model, preprocess = F)

if(prawdopodobna\_klasa != data$klasa[i])

{

errors <- errors + 1

}

}

return(errors/length(data$klasa))

}

Do wyszukiwania optymalnego *k*-additive smoothing jest wykorzystana kroswalidacja na testowym zbiorze.

Kroswalidacja modelu:

CrossValidation<-function(trainData, testData, laplaceFactorValues, showLog = F)

{

cvErrors <- rep(NA, length(laplaceFactorValues))

for(i in 1:length(laplaceFactorValues))

{

model <- Nasz\_model(trainData, laplaceFactorValues[i])

cvErrors[i] <- Model\_testowy(testData, model)

if(showLog)

{

print(paste(laplaceFactorValues[i], ": error is ", cvErrors[i], sep=""))

}

}

return(cvErrors)

}

**3.4 Algorytm k-NN**

Algorytm k-NN implementowano w pliku knn.R

W ramach wstępnego przetwarzania danych, zostały wykonane następujące kroki:

* zmniejszyliśmy tekst do małych liter;
* wyodrębniono strony internetowe, wszystkie strony zastąpiono na „wbst” (od słowa „website”);
* ciągi numeryczne z 11 znaków wyodrębniono jako numer teleponu, taki ciągi są zmienione na „phnnmbr” (od słów „phone number”);
* usunęliśmy wszystkie znaki interpunkcji;
* wyodrębniono końcówki SMS’ów „xxx” jako „msgend” (od słów „message ending”), w tych miejscach na końcach używane były ciągi symbolów „x” zamias imieni nadawcy
* wyodrębniono „xxx” w środku powiadomień
* usunięte nadmierne spacje

Z wczytanego pliku tekstowego jest tworzony data frame, w którym są 2 kolumny. Pierwsza zawiera klasy powiadomień (ham/spam), druga – treść SMS. Pod czas odczytu pliku i tworzenia data frame’u również są usunięte tak nazywane „stop words”. Dla tego jest stworzona metoda read\_lines\_classified.

Dalej jest utworzona macierz zer i jedynek, w której kolumny to są słowa (swojego rodzaju słownik naszego modelu), a indeksy wierszy odpowiadają SMS’om w data frame’ie. Dla tego służy metoda create\_word\_matrix.

Potem obieramy podział naszego zbioru na trenujący oraz testowy. Niestety, przez bardzo wielką złożoność algorytmu nie udało się wybierać dość duże zbioru trenujące i testujące. Dlatego dla tworzenia zbioru trenującego wykorzystano jest 250 przykładów, a testowano nie więcej 25 przykładów.

Wlaśnie, sam plik był utworzony tak, żeby dla jego uruchomienia trzeba było robić jak najmniej dodatkowych działań.

**4 Wyniki**

**4.1 Naiwny klasyfikator Bayesa**

Poniższy kod odczytuje dane, tworzy modele dla wartości parametru k-additive smoothing od 0 do 10. Wybieramy najlepszy wynik, testujemy model na wcześnie niewykorzystywanym testowym zbiorze, potem budujemy wykres zmiany błędu na kroswalidacyjnym zbiorze od parametru k-additive smoothing i końcowy poziom błędu na testowym zbiorze.

rm(list = ls())

source("E:/10/spam0.R")

set.seed(14880)

fileName <- "E:/10/Data/SMSSpamCollection"

data1 <- Open\_data()

data2 <- New\_dataset(data1, proportions = c(0.7, 0.3))

laplaceFactorValues <- 1:10

cvErrors <- CrossValidation(data1[data2$train, ], data1[data2$test, ], 0:10, showLog = T)

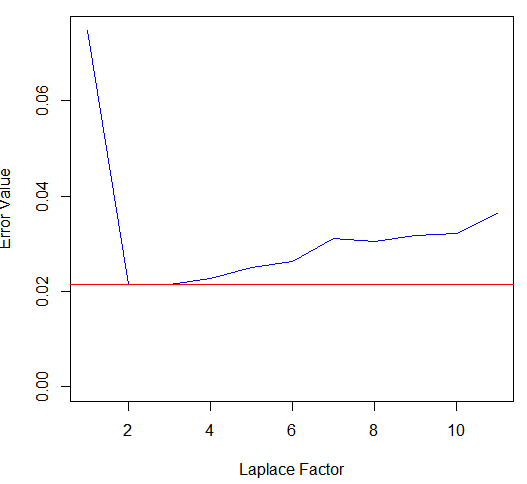
bestLaplaceFactor <- laplaceFactorValues[which(cvErrors == min(cvErrors))]

model <- Nasz\_model(data=data1[data2$train, ], laplaceFactor=bestLaplaceFactor)

testResult <- Model\_testowy(data1[data2$test, ], model)

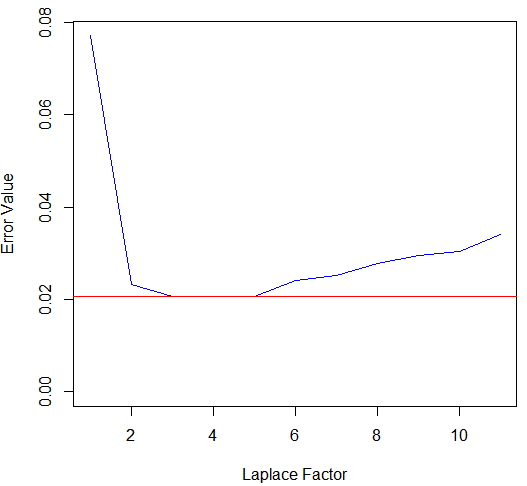
plot(cvErrors, type="l", col="blue", xlab="Laplace Factor", ylab="Error Value", ylim=c(0, max(cvErrors)))

abline(h=testResult, col="red")



Jak widzimy, ta metoda jest bardzo efektywna, wartość błędu na zbiorze testowym (stosunek niesłusznie zaklasyfikowanych wiadomości do ogólnej ilości wiadomości) jest 2.15 % przy wartości parametru laplaceFactor 2 albo 3.

Przy podziale zbioru w proporcji 0.8, 0.2, najmniejsza wartość błędu na zbiorze testowym równa się 2.06 %.



**4.2 Algorytm k-NN**

Dla oszacowania wydajności utworzonego algorytmu on był uruchomiony z różnymi zakresami danych.

*Tabela 2. Wyniki testowania algorytmu k-NN*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | metric | k | overall | ham | spam |
| 1 | e | 10 | 0.857 | 0.920 | 0.5 |
| 2 | c | 10 | 0.643 | 0.750 | 0.0 |
| 3 | e | 10 | 0.760 | 0.750 | 0.8 |
| 4 | c | 10 | 0.560 | 0.700 | 0.0 |
| 5 | e | 6 | 0.762 | 0.789 | 0.5 |
| 6 | c | 4 | 0.714 | 0.789 | 0.0 |

*Tabela 3. Zakresy, wykorzystane dla tworzenia modeli oraz testowania*

|  |  |  |
| --- | --- | --- |
|  | train | test |
| 1 | 1:250 | 5561:5574 |
| 2 | 420:740 | 51:75 |
| 3 | 4250:4600 | 1265:1285 |

W wyniku mamy taką sytuację, że dla danej implementacji i testowych zbiorów ilość sąsiadów nie wpłynęła na dokładność klasyfikacji, a korzystając z metryki kosinusowej, nie potrafiliśmy prawidłowo wyznaczyć przynajmniej jedno powiadomienie SPAM. Z danych testów nie można na pewnie powiedzieć, że klasyfikacja za pomocą metryki kosinusowej jest zła i niepoprawna, dlatego, że nam mogły prosto trafić takie dane. Na testach, prowadzonych dla sprawdzania działalności kolejnych metod w pliku knn.R było widać, że metryka kosinusowa działa i obliczenia odbywają się poprawnie. W teorii metryka kosinusowa musiałaby dawać lepszy wynik, niż Euklidysowa, ale z tabeli wyżej uzyskaliśmy odwrotne efekty.

Dla przyspieszania algorytmu były usunięte niektóre dodatkowe przetwarzania wierszy powiadomień oraz normy wektorów nie są obliczane w każdej iteracji predykcji, co dało możliwość korzystać z tego algorytmu.

Dla polepszenia wyników predykcji można jeszcze użyć steming tekstu, co jest dość przydatne i często korzystanie w obróbce tekstów dla tworzenia dobrego słownika modeli i bardziej skutecznego procesu klasyfikacji. Ta się odbywa dlatego, że po stemingu zostają tylko słowa semantyczne i oni są w formie bezokolicznika (dla czasowników) oraz tylko korzeń słów (dla rzeczowników).

Także można korzystać z innych wartości atrybutów niż binarne, dla obliczenia metryk. Wtedy wynik będzie bardziej odpowiedni.

kNN — prosty algorytm klasyfikacji, a więc dla zastosowania w prawdziwych problemach jest często nieskuteczny. Oprócz dokładności klasyfikacji problemem również jest jego wydajność obliczeniowa. Jeśli w zbiorze trenującym jest N obiektów, w testującym — M i wymiarowość przestrzeni — K, to ilość operacji klasyfikacji badanej próby można ocenić jako *O*(*K\*M\*N*). Niemniej jednak Algorytm k-NN jest dobrym przykładem do odkrywania Machine Learning.