5. Clasificatorul Naive Bayes

Algoritmi de învăţare generativi
Antrenarea clasificatorului Naive Bayes
Medierea Laplace
Predicţia datelor

În lucrările de laboratoar precedente au fost prezentaţi algoritmi ce modelează distribuţia condiţionată $p(y|x, \theta)$ a lui y, dându-se caracteristicile x şi parametrii θ [1]. Spre exemplu, regresia logistică modelează $p(y|x, \theta)$ sub forma $h_\theta(x) = g(\theta^T x)$, unde g este funcţia sigmoid.

O posibilă problemă de clasificare este de a face diferenţa între elefanţi ($y = 1$) şi câini ($y = 0$) pe baza unui set de caracteristici (eng. features) ce descriu un animal. Dându-se o bază de date de antrenare, un algoritm precum regresia logistică încearcă să determine o linie dreaptă, denumită margine de decizie, ce separă elefanţii de câini. Pentru a clasifica un nou animal ca fiind elefant sau câine, metoda verifică de ce parte a marginei de decizie se află caracteristicile animalului.

Abordarea prezentată în acestă lucrare de laborator construieşte mai întâi un model ce descrie modul în care arată elefanţii, urmat de construcţia unui model separat ce descrie modul în care arată câinii. În final, pentru a clasifica un animal, putem calcula cât de mult se aseamănă el cu modelul ce descrie elefanţii, pe de o parte, şi cât de mult se aseamănă cu modelul ce descrie câinii, pe de altă parte.

Algoritmii precum regresia logistică, ce modelează $p(y|x)$ direct, sau algoritmii ce învaţă să mapeze direct spaţiul caracteristicilor de intrare $\mathbb{X}$ la etichetele (eng. labels) $\{0, 1\}$, sunt denumiţi *algoritmi de învăţare discriminanţi*. În cele ce urmează, vom discuta despre o altă clasă de algoritmi, intitulaţi *algoritmi de învăţare generativi*, ce modelează distribuţiile $p(x|y)$ şi $p(y)$. Spre exemplu, dacă y indică faptul că un exemplu este un câine (0) sau un elefant (1), atunci $p(x|y = 0)$ modelează distribuţia caracteristicilor (eng. features) câinilor, iar $p(x|y = 1)$ modelează distribuţia caracteristicilor elefanţilor.

După modelarea lui $p(y)$ (denumită *probabilitatea apriori*) şi a lui $p(x|y)$, algoritmul calculează probabilitatea ulterioară $p(y|x)$ a lui y dându-se x, prin regula Bayes:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}, \tag{5.1}$$

unde numitorul are expresia:

$$p(x) = p(x|y = 1)p(y = 1) + p(x|y = 0)p(y = 0). \tag{5.2}$$

Nu este necesar să calculăm numitorul pentru a efectua o predicţie utilizând $p(y|x)$:

$$\begin{aligned} \arg\max_y p(y|x) &= \arg\max_y \frac{p(x|y)p(y)}{p(x)} \\ &= \arg\max_y p(x|y)p(y). \end{aligned}$$

5.1 Clasificatorul Naive Bayes

În studiul clasificatorului Naive Bayes, vom lua în considerare caracteristici x_i ce iau valori discrete.

În acestă lucrare de laborator vom construi un filtru de email-uri spam, ce va detecta mesaje nesolicitate şi comerciale în limba engleză. Acest algoritm poate filtra mesajele spam şi le poate salva, spre exemplu, într-un folder separat. Clasificarea de email-uri face parte dintr-un spectru mai larg de probleme intitulate *clasificare de text*.

Luăm în considerare că avem la dispoziţie o bază de date de antrenare ce conţine email-uri etichetate ca spam sau non-spam. Primul pas în construcţia filtrului de spam este specificarea caracteristicilor x_i ce descriu un email.

Vom reprezenta un email printr-un vector de caracteristici a cărui lungime este egală cu numărul de cuvinte dintr-un dicţionar. În particular, dacă un email conţine cuvântul de la poziţia i din dicţionar, atunci vom seta $x_i = 1$; altfel $x_i = 0$. Spre exemplu, vectorul:

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{matrix}, \tag{5.3}$$

este utilizat pentru a descrie un email ce conţine cuvintele "a" şi "buy", dar nu şi "aardvark", "aardwolf", sau "zygmurgy". Setul de cuvinte codat în vectorul de caracteristici este denumit *vocabular*. Astfel, dimensiunea lui x este egală cu dimensiunea vocabularului.

Luând în considerare vectorul de caracteristici definit mai sus, vom construi în continuare modelul generativ $p(x|y)$. Dacă am folosi, spre exemplu, un vocabular de 50000 de cuvinte, atunci $x \in \{0, 1\}^{50000}$ (x este un vector de 50000 de dimensiuni, format din 0 şi 1). Dacă

ar fi să modelăm x explicit cu o distribuţie multinominală pe cele 2^{50000} posibile rezultate, atunci am obţine un vector de parametrii de dimensiune $2^{50000} - 1$ (în mod evident cu un număr mult prea mare de parametrii).

Pentru a modela $p(x|y)$ se vor considera caracteristicile x_i ca fiind independente între ele, dându-se y. Această presupunere este denumită *presupunerea Naive Bayes*, iar algoritmul rezultant poartă numele de *clasificatorul Naive Bayes*. Spre exemplu, considerăm că $y = 1$ reprezintă un email spam, iar cuvintele "buy" şi "price" au indecşii 2087, respectiv 39831. În cazul unui email spam ($y = 1$), informaţia x_{2087} (informaţia că mesajul conţine cuvântul "buy") nu va avea niciun efect asupra valorii lui x_{39831} (informaţia că mesajul conţine cuvântul "price"). Formal, acest lucru se poate scrie ca:

$$p(x_{2087}|y) = p(x_{2087}|y, x_{39831}). \tag{5.4}$$

A se lua în considerare că egalitatea 5.4 *nu* indică faptul că variabilele x_{2087} şi x_{39831} sunt independente. Independenţa variabilelor una de cealaltă ar fi scrisă $p(x_{2087}) = p(x_{2087}|x_{39831})$. În 5.4 presupunem că x_{2087} şi x_{39831} sunt condiţional independente *dându-se* y.

Astfel, obţinem:

$$\begin{aligned} p(x_1, ..., x_{50000}|y) &= p(x_1|y) \cdot p(x_2|y, x_1) \cdot p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, ..., x_{49999}) \\ &= p(x_1|y) \cdot p(x_2|y) \cdot p(x_3|y) \cdots p(x_{50000}|y) \\ &= \prod_{i=1}^{n} p(x_i|y). \end{aligned}$$

Prima egalitate este dedusă direct din proprietăţile standard ale probabilităţilor, iar a doua foloseşte presupunerea Naive Bayes. Algoritmul rezultat funcţionează bine în multe aplicaţii, cu toate că presupunerea Naive Bayes este puternică.

5.2 Antrenarea clasificatorului Naive Bayes

Pentru a parametriza modelul, vom lua în considerare distribuţia Bernoulli ϕ:

$$p(y = 1, \phi) = \phi. \tag{5.5}$$

Distribuţia Bernoulli modelează variabilele aleatoare ce iau doar valorile 0 şi 1. Ea poate fi utilizată pentru a modela procesul de "dat cu banul", unde 1 reprezintă "cap", iar 0 reprezintă "pajură" (sau vice versa). În acest caz $\phi_{y=0} = \phi_{y=1} = 0.5$ (probabilitatea de a obţine "cap" este egală cu probabilitatea de a obţine "pajură"). În cazul unui ban "manipulat" (ce tinde, spre exemplu, să cadă în special pe "cap") $\phi_{y=0} \neq \phi_{y=1}$.

Modelul filtrului de spam este parametrizat astfel:

$$\phi_{i|y=1} = p(x_i = 1|y = 1), \tag{5.6}$$

$$\phi_{i|y=0} = p(x_i = 1|y = 0), \tag{5.7}$$

$$\phi_y = p(y = 1). \tag{5.8}$$

Dându-se o bază de date de antrenare, $(x^{(i)}, y^{(i)}); i = 1, ..., m$, se poate scrie probabilitatea comună a datelor:

$$\mathcal{L}(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}). \tag{5.9}$$

Maximizând faţă de ϕ_y, $\phi_{i|y=0}$ şi $\phi_{i|y=1}$, obţinem estimatul probabilităţii maxime (eng. Maximum Likelihood Estimation):

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \ \wedge \ y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}, \tag{5.10}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \ \wedge \ y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}, \tag{5.11}$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}, \tag{5.12}$$

unde, simbolul "$\wedge$" reprezintă operaţia binară "ŞI logic". De asemenea, $1\{z\}$ reprezintă aşa numita *funcţie indicator* ce ia valoarea 1 atunci când expresia y este adevărată:

$$1\{\text{Adevărat}\} = 1, \tag{5.13}$$

$$1\{\text{Fals}\} = 0. \tag{5.14}$$

Spre exemplu, $1\{2 = 3\} = 0$, deoarece $2 \neq 3$, iar $1\{1 + 1 = 2\} = 1$.

Parametrii condideraţi au o interpretare naturală. Spre exemplu, $\phi_{j|y=1}$ este fracţiunea din email-urile spam $(y = 1)$ în care apare cuvântul j.

5.3 Medierea Laplace

Statistic, nu este o idee bună estimarea probabilităţii unui anumit eveniment ca fiind zero, doar pentru că nu a fost observat în baza de date de antrenare. *Medierea Laplace* este utilizată pentru a evita situaţia în care anumiţi parametrii ϕ_j ar putea deveni zero, făcând

astfel împărţirea imposibilă. Noul estimat, după medierea Laplace, devine:

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + V}. \tag{5.15}$$

În formula de mai sus a fost adăugat 1 la numărător şi V la numitor. În exemplul de faţă vom considera valoarea lui V egală cu numărul de cuvinte din dicţionar.

5.4 Predicţia datelor

Parametrii $\phi_{j|y=1}$, $\phi_{j|y=0}$ şi ϕ_y vor fi calculaţi din setul de date de antrenare. Pentru a efectua o predicţie, vom utiliza aceşti parametrii în comparaţia cantităţilor $p(x|y = 1)p(y = 1)$ şi $p(x|y = 0)p(y = 0)$. Odată ce am obţinut estimatele parametrilor, putem efectua predicţii pe noi exemple x astfel:

$$\begin{aligned} p(y = 1|x) &= \frac{p(x|y = 1) \cdot p(y = 1)}{p(x)} \\ &= \frac{(\prod_{i=1}^{n} p(x_i|y = 1)) \cdot p(y = 1)}{(\prod_{i=1}^{n} p(x_i|y = 1)) \cdot p(y = 1) + (\prod_{i=1}^{n} p(x_i|y = 0)) \cdot p(y = 0)}. \end{aligned}$$

Aşa cum am menţionat în introducere, nu este necesar calculul numitorului din formula precedentă pentru a efectua o predicţie.

Computaţional este mai eficient să calculăm probabilităţile logaritmice $\log p(x|y = 1)$ şi $\log p(y = 1)$. Astfel, înmulţirile necesare calculului probabilităţilor vor fi înlocuite de adunări.

5.5 Cerinţe

Să se scrie un script Python ce va clasifica mesaje email scrise în limba engleză ca fiind spam sau non-spam.

5.5.1 Baza de date de antrenare

Pentru a implementa algoritmul, descărcaţi datele din arhiva "lab5Data.zip".

Baza de date este împarţită în două subseturi, şi anume 700 de email-uri de antrenare şi 260 de email-uri de testare. Fiecare subset conţine 50% mesaje spam şi 50% mesaje non-spam. Adiţional, email-urile au fost procesate în felul următor:

(a) Anumite cuvinte, precum "and", "the" şi "of" sunt foarte comune în frazele din limba engleză şi nu sunt relevante în decizia de a clasifica un mesaj ca spam sau nu. Aceste cuvinte au fost înlăturate din email-uri.

(b) Cuvintele care au acelaşi înţeles, însă o terminaţie diferită, au fost ajustate în aşa fel încât să aibă aceeaşi formă. Spre exemplu, "include", "includes" şi "included" vor fi

reprezentate de ”include”. Toate literele din corpul email-ului au fost convertite la litere mici.

(c) Numerele şi semnele de punctuaţie au fost înlăturate. Tab-urile, noile linii şi spaţiile au fost convertite la un singur spaţiu.

Mai jos se găsesc două exemple de mesaje înainte şi după procesare.

Mesajul non-spam ”5-1361msg1” înainte de preprocesare

```
Subject: Re: 5.1344 Native speaker intuitions

The discussion on native speaker intuitions has been extremely
interesting, but I worry that my brief intervention may have
muddied the waters. I take it that there are a number of
separable issues. The first is the extent to which a native
speaker is likely to judge a lexical string as grammatical
or ungrammatical per se. The second is concerned with the
relationships between syntax and interpretation (although even
here the distinction may not be entirely clear cut).
```

Mesajul non-spam ”5-1361msg1” după preprocesare

```
re native speaker intuition discussion native speaker intuition
extremely interest worry brief intervention muddy waters number
separable issue first extent native speaker likely judge lexical
string grammatical ungrammatical per se second concern
relationship between syntax interpretation although even here
distinction entirely clear cut
```

Mesajul spam ”spmsgc19” după preprocesare

```
financial freedom follow financial freedom work ethic
extraordinary desire earn least per month work home special
skills experience required train personal support need ensure
success legitimate homebased income opportunity put back control
finance life ve try opportunity past fail live promise
```

Arhiva "lab5Data.zip" conţine fişierul "train-features-full.txt", ce reprezintă matricea de antrenare x, formată din 700 de linii şi 2500 de coloane. Fiecare linie reprezintă un email, iar fiecare coloană reprezintă un cuvânt din dicţionar. Un element dintr-o linie a matricei x reprezintă numărul de apariţii în email al cuvântului reprezentat de acel element.

Fişierul "train-labels.txt" conţine etichetele atribuite fiecărui email ($y = 1$ pentru email-urile spam şi $y = 0$ pentru email-urile non-spam) şi are dimensiunea 700×1.

5.5.2 Antrenarea

Paşii de antrenare ai algoritmului Naive Bayes, odată ce baza de date de antrenare a fost încărcată în Python, sunt următorii:

1. calculul ϕ_y;
2. calculul fiecărui $\phi_{j|y=1}$ pentru fiecare cuvânt din dicţionar şi stocarea rezultatelor într-un vector;
3. calculul fiecărui $\phi_{j|y=0}$ pentru fiecare cuvânt din dicţionar şi stocarea rezultatelor într-un vector.

5.5.3 Testarea

Odată ce parametrii modelului au fost calculaţi, ei pot fi utilizaţi în efectuarea predicţiilor. Pentru aceasta se vor folosi fişierele "test-features-full.txt" şi "test-labels.txt", ambele având structura fişierelor "train-features-full.txt", respectiv "train-labels.txt".

Paşii utilizaţi pentru clasificarea documentelor sunt următorii:

1. pentru fiecare document din baza de date de testare se calculează $\log p(x|y = 1) + \log p(y = 1)$;
2. similar, se calculează $\log p(x|y = 0) + \log p(y = 0)$;
3. se compară cele două cantităţi de la punctele (1) şi (2) şi se ia decizia de clasificare a email-ului în spam sau non-spam, luând în considerare cea mai mare valoare.

Bibliografie

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.