Ćwiczenie 5

Modele bayesowskie

Adam Nowakowski (300428)

1. Zadanie

Zadaniem była implementacja *naiwnego klasyfikatora Bayesa*, którego należało użyć do zbadania, która para atrybutów ze zbioru danych *wine* pozwala osiągnąć najlepszą dokładność klasyfikacji. Do oceny zbioru należało wykorzystać algorytm n-krotnej walidacji krzyżowej.

2. Wprowadzenie teoretyczne

Zaczynając rozważania na temat naiwnego klasyfikatora Bayesa, należy najpierw przybliżyć na czym polega zadanie klasyfikacji w uczeniu maszynowym.

Klasyfikacja, podobnie jak regresja, należą do grupy zadań **uczenia nadzorowanego**. Na początku mamy dostępny zbiór danych uczących etykietowanych, czyli takich par $\langle x_i, y_i \rangle$, gdzie x_i to i-te wejście naszego nieznanego procesu bądź zależności, a y_i to jego wyjście, które są w ogólności wielowymiarowe. Czyli nasze wejście x_i może składać się z wielu atrybutów. Naszym zadaniem jest na podstawie próbek uczących możliwie jak najdokładniej przybliżyć nieznany nam proces i zbudować jego model, opierając się na próbkach uczących. Innymi słowy poszukujemy funkcji decyzyjnej $f(x): X \mapsto Y$, która przekształca przestrzeń wejść X w przestrzeń wyjść Y minimalizując przy tym funkcję straty $q\big(d(x)\big)$. Zadanie klasyfikacji różni się od regresji tym, że jego przestrzeń wyjść Y jest dyskretna. Na podstawie atrybutów wejściowych danej próbki przypisujemy jej klasę.

Jako przykład objaśniający powyższe zagadnienia może posłużyć nam zbiór, na którym pracowaliśmy podczas tego zadania. Dotyczył on informacji o winach. Naszymi danymi wejściowymi były zestawy trzynastu atrybutów, opisujących (charakteryzujących) daną butelkę wina. Dotyczyły np. zawartości alkoholu, intensywności koloru, czy zawartości magnezu w trunku. Wyjściem natomiast była jednowymiarowa, dyskretna zmienna, która określała jakiego rodzaju (klasy) jest to wino. W naszym zbiorze danych były trzy rodzaje win: (1, 2, 3). Naszym zadaniem było znalezienie zależności pomiędzy danymi wejściowymi a wyjściowymi, by było możliwe wywnioskowanie (przewidzenie) jakiej klasy jest nowe wino, które jeszcze nie pojawiło się w naszej bazie.

Naiwny klasyfikator Bayesa jest modelem, który pozwala nam na rozwiązanie zadania klasyfikacji. Wywodzi się bezpośrednio z twierdzenia Bayesa, które mówi:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Dla ogólnego przypadku klasyfikacji powyższy wzór będzie wyglądał następująco:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n|y)P(y)}{P(x_1, \dots, x_n)}$$

gdzie:

$x_1, \dots, x_n$ – atrybuty wejściowe

y – przewidywana klasa

$P(y|x_1, \dots, x_n)$ - prawdopodobieństwo wystąpienia klasy y pod warunkiem, że dla danego wektora wejściowego zaobserwowaliśmy wartości atrybutów $x_1, \dots, x_n$

$P(x_1, \dots, x_n|y)$ – prawdopodobieństwo wystąpienia danych atrybutów, pod warunkiem zaobserwowania klasy y, co możemy estymować z danych uczących

$P(y)$ – prawdopodobieństwo wystąpienia klasy y – można łatwo wyliczyć z danych uczących, dzięki temu, że wiemy jak często występuje dana klasa.

$P(x_1, \dots, x_n)$ – prawdopodobieństwo zaobserwowanych atrybutów

Model naiwnego klasyfikatora Bayesa wygląda następująco:

$$\hat{y} = \arg\max_{y \epsilon Y} P(y)P(x_1, \dots, x_n|y)$$

W tym modelu zakładamy, że wszystkie atrybuty wejściowe są niezależnymi zmiennymi losowymi. Stąd wynika „naiwność” naszego klasyfikatora. Ułatwia to nam estymację składnika $P(x_1, \dots, x_n|y)$, który przy powyższych założeniach przyjmuje formę:

$$P(x_1, \dots, x_n|y) = \prod_{i=1}^{n} P(x_i|y)$$

Teraz nasz klasyfikator wygląda następująco:

$$\hat{y} = \arg\max_{y \epsilon Y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

Składniki $P(y)$ oraz $P(x_i|y)$ można wyliczyć bezpośrednio z danych uczących. Przewidywana klasa jest wybierana spośród innych jako ta z największym prawdopodobieństwem wystąpienia.

3. Implementacja

W katalogu `/implementacja` znajdują się cztery skrypty w kodzie Pythona:

- *`naive_bayes_classifier.py`* – zawiera klasę, w której zaimplementowano naiwny klasyfikator Bayesa.
- *`task.py`* – skrypt, który rozwiązuje problem podany w poleceniu przy użyciu implementacji naiwnego klasyfikatora Bayesa i walidacji krzyżowej.
- *`prepare_data.py` – program, który przygotowuje surowe dane z pliku `wine.data` tworząc z nich tablicę struktur.*
- *`cross_validation`*`.py` – zawiera implementację k-krotnej walidacji krzyżowej

Klasa `Nbc` zawiera pola i metody, które opisują naiwny klasyfikator Bayesa.

- Model uczony jest za pomocą metody `train()`. Z danych wyliczana jest średnia oraz wariancja, które zapisywane są do `self.data`. Te informacje potrzebne są do obliczenia rozkładów normalnych atrybutów pod warunkiem danej klasy.
- Klasyfikacja odbywa się w metodzie `answer()`. Tutaj z danych zapisanych w `self.data` dla każdej klasy obliczane są prawdopodobieństwa wystąpienia danego zestawu atrybutów ze zmiennej `entity` pod warunkiem, że jest on z danej klasy, według wzoru:

$$P(y_j|x_1,\dots,x_n) = P(y_j)\prod_{i=1}^{n}\frac{1}{\sigma_{ji}\sqrt{2\pi}}\exp\left(\frac{-(x_i-\mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

gdzie:

y_j – j-ta klasa
$x_1,\dots,x_n$ – atrybuty wejściowe, dla których liczymy wyjście z nauczonego modelu
σ_{ji} – odchylenie standardowe i-tego parametru pod warunkiem j-tej klasy liczone na podstawie próbek uczących
μ_{ji} – średnia arytmetyczna i-tego parametru pod warunkiem j-tej klasy liczona na podstawie próbek uczących

Wzór ma taką postać ze względu na to, że założyliśmy, że poszczególne parametry uwarunkowane na poszczególnych klasach to zmienna losowa o rozkładzie normalnym.

- Za pomocą metody `test()` możliwe jest obliczenie błędu nauczonego modelu na innym zbiorze danych.

4. Rozwiązanie zadania

Za pomocą 4-krotnej walidacji krzyżowej sprawdziłem każdą z par atrybutów wejściowych. Z powodu pewnej losowości występującej w mojej implementacji walidacji test powtórzyłem 10 razy i wziąłem z tych wyników średnią.

Etykiety atrybutów			
1	Alcohol	8	Nonflavanoid phenols
2	Malic acid	9	Proanthocyanins
3	Ash	10	Color intensity
4	Alcalinity of ash	11	Hue
5	Magnesium	12	OD280/OD315 of diluted wines
6	Total phenols	13	Proline
7	Flavanoids		

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
2	0.211	X	X	X	X	X	X	X	X	X	X	X
3	0.271	0.381	X	X	X	X	X	X	X	X	X	X
4	0.217	0.362	0.369	X	X	X	X	X	X	X	X	X
5	0.304	0.322	0.461	0.351	X	X	X	X	X	X	X	X
6	0.157	0.302	0.298	0.299	0.282	X	X	X	X	X	X	X
7	0.086	0.190	0.162	0.156	0.162	0.219	X	X	X	X	X	X
8	0.218	0.423	0.379	0.399	0.331	0.325	0.198	X	X	X	X	X
9	0.206	0.368	0.380	0.388	0.348	0.363	0.226	0.404	X	X	X	X
10	0.169	0.227	0.248	0.220	0.223	0.135	0.096	0.207	0.176	X	X	X
11	0.123	0.366	0.292	0.299	0.226	0.214	0.106	0.301	0.337	0.160	X	X
12	0.124	0.327	0.249	0.227	0.207	0.245	0.166	0.299	0.357	0.124	0.281	X
13	0.156	0.191	0.237	0.239	0.248	0.192	0.095	0.209	0.189	0.121	0.113	0.086

Najlepszą dokładność daje klasyfikacja po **1. i 7**. atrybucie oraz **13. i 12**, uzyskując **0.086**. Jednak dobre wyniki można tez znaleźć przy połączeniu 7. z 10. oraz 7. z 13. Klasyfikacja po wszystkich atrybutach daje błąd na poziomie **0.023**, więc powyższe wyniki nie odstają aż tak bardzo. Przy klasyfikacji po najlepszych atrybutach (1, 7, 13, 12) zbliżamy się do coraz mniejszych wartości i otrzymujemy błąd na poziomie **0.050.** Klasyfikując po samym 7. atrybucie otrzymujemy błąd **0.197**, co jak na jeden atrybut nie jest wcale złym wynikiem.