Metoda k-najbliższych sąsiadów

Rafał Bojarczuk

17 05 2020

Wstęp

Celem tej pracy jest zademonstrowanie skuteczności klasyfikatora k-najbliższych sąsiadów - jednego z popularniejszych algorytmów służących do zadań klasyfikacji. Jego popularność w dużej mierze wynika z jego prostoty. Mając zbiór obserwacji oraz informację o klasach do których one należą, możemy próbować odgadnąć etykietę nowej obserwacji patrząc na to, której klasy reprezentanci występują najczęściej wśród zbioru k najbliższych znanych obserwacji względem przyjętej metryki. Na przykład jeżeli 6 najbliższych sąsiadów (w kolejności od najmniejszej odległości do największej) nowej danej należy do klas numer 1,2,2,3,3,3 to dla k=1 zaklasyfikujemy nową obserwację jako reprezentanta klasy 1, dla k=3 jako 2 a dla k=6 jako 3. W przypadku "remisu" możemy wylosować klasę spośród najczęściej się powtarzających dla osiągnięcia nieobciążoności estymatora lub ważyć je ze względu na odległość - te znajdujące się bliżej mają "decydujący głos". Ja posłużę się pierwszą opcją.

Regresja porządkowa

Decyzja o wyborze klasy na podstawie mody z etykiet najbliższych sąsiadów jest standardowym i w miarę oczywistym podejściem w przypadku gdy przewidywana wartość jest zmienną na skali nominalnej. Lecz metoda k-najbliższych sąsiadów może mieć zastosowanie również dla danych porządkowych. Załóżmy, że na podstawie aktywności mózgu chcemy przewidzieć subiektywną odpowiedź pacjenta, jak duży odczuwa ból w skali od 1 do 10. Jeżeli aktywność mózgu posiada pewne cechy podobne (bliskie) do znanego nam bólu ocenianego wcześniej na 1 oraz pewne cechy podobne do bólu o natężeniu 3 to może sensownym będzie zaklasyfikować go jako coś pomiędzy, a nie zgadywać odpowiedź tylko ze zbioru {1,3}. W tym przypadku możemy użyć innych funkcji wyłaniających najbardziej prawdopodobną klasę spośród sąsiadów niż tylko modę. Porównamy działanie kilku takich funkcji:

- **srednia_a()** - wyznacza najbliższą wartość naturalną do średniej arytmetycznej z etykiet, w przypadku gdy taka wartość nie jest określona jednoznacznie - losuje losową z dwóch wartości
- **mediana()** - zwraca najbliższą wartość naturalną do mediany
- **minkara1.5()** - zwraca wartość u spośród etykiet sąsiadów, która minimalizuje funkcję straty $\sum_{j=1}^{k} |N[j] - u|^{1.5}$ gdzie $N[j]$ to etykieta j-tego sąsiada
- **minkara3.0()** - jak wyżej, tylko funkcją straty jest $\sum_{j=1}^{k} |N[j] - u|^{3.0}$

Zbiory danych

Do prezentacji wyników posłużę się kilkoma zbiorami danych pobranymi ze strony https://www.gagolewski.com/resources/data/ordinal-regression/. Zbiór *winequality_red* zawiera informacje na temat słodkości, stężenia alkoholu i siarczynów oraz innych cech czerwonych win wraz z nadaną przez ekspertów oceną w

Table 1: Red wine quality (selected columns)

response	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide
3	7.4	0.70	0.00	1.9	0.076	11
3	7.8	0.88	0.00	2.6	0.098	25
3	7.8	0.76	0.04	2.3	0.092	15
4	11.2	0.28	0.56	1.9	0.075	17
3	7.4	0.66	0.00	1.8	0.075	13

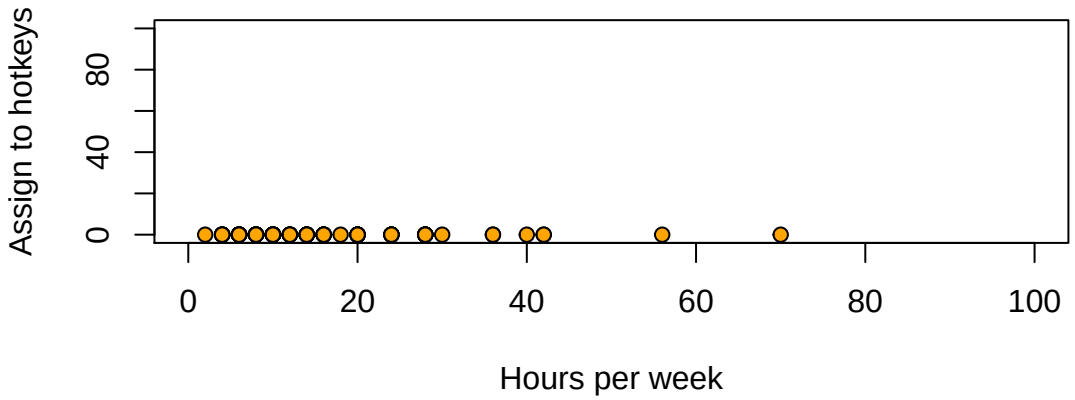
Table 2: StarCraft 2 skill (selected columns)

response	Age	HoursPerWeek	TotalHours	APM	SelectByHotkeys	AssignToHotkeys
5	27	10	3000	143.7180	0.0035152	0.0002197
5	23	10	5000	129.2322	0.0033038	0.0002595
4	30	10	200	69.9612	0.0011011	0.0003356
3	19	20	400	107.6016	0.0010335	0.0002131
3	32	10	500	122.8908	0.0011360	0.0003273

skali od 1 do 7. Zbiór *skill* zawiera dane na temat graczy StarCrafta 2 - wiek, łączny czas gry, liczba akcji wykonywanych na minutę oraz numer ligi w jakiej się znajdują - Bronze, Silver, Gold, ... itd. zakodowany liczbą od 1 do 8. Do testów użyłem też zbiorów *affairs*, *cement_strength* oraz *wisconsin_breast_ord*.

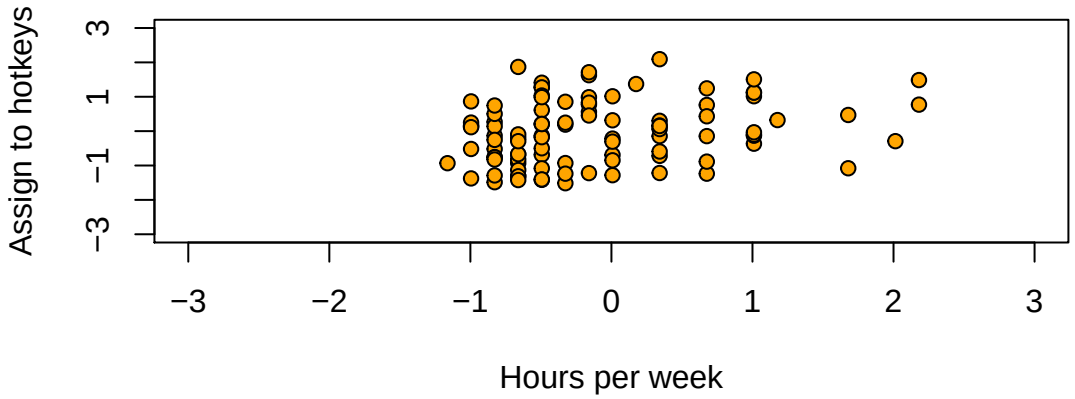
Przygotowanie danych do oceny jakości klasyfikatora

Jak widać wyżej, w zbiorze *skills* wartości w kolumnie HoursPerWeek wyraża się w dziesiątkach, natomiast inne kolumny takie jak, AssignToHotkeys lub SelectByHotkeys zawierają bardzo małe liczby, o niewielkiej wariancji. Powoduje to sytuację, że różnica między wartościami HoursPerWeek dwóch obserwacji będzie znacznie przeważała podczas obliczania odległości między danymi, a różnica między wartościami z kolumny AssignToHotkeys nie będzie grała żadnej roli. Dla uproszczenia załóżmy, że to jedyne dwie cechy definiujące umiejętności gracza StarCrafta, aby łatwo je zwizualizować.



Nie można rozróżnić obserwacji ze względu na cechę AssignToHotkeys!

Po zestandaryzowaniu kolumn:



Teraz lepiej. Nie w każdej analizowanej ramce występuje problem z drastycznie różniącymi się zakresami wartości, ale przed użyciem algorytmu zestandaryzowałem wszystkie - nie zaszkodzi, a może pomóc.

Ocena jakości

Skorzystamy z metody 5-krotnej walidacji do wyznaczenia podziału na zbiór treningowy (znanych obserwacji) oraz testowy (nieznanych danych). Oznacza to, że każdy zbiór podzielimy na 5 równych (bądź prawie równych) części $X^{(1)}, ..., X^{(5)}$, a następnie dla każdego $i \in \{1, 2, 3, 4, 5\}$ użyjemy $X^{(i)}$ jako zbiór testowy i $X \setminus X^{(i)}$ jako zbiór uczący. Wynikiem dla danego zbioru będzie średni błąd ze wszystkich podziałów. Będziemy mierzyć proporcję błędnie sklasyfikowanych danych do wszystkich, błąd średniokwadratowy oraz średnią odległość od właściwej etykiety (Mean Absolute Distance). Testowane będą wszystkie kombinacje parametrów: $k \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$ $p \in \{1, 2\} \vee p = \infty$ dla każdej z funkcji agregujących wymienionych na początku

Wyniki

Uznałem, że jako dwie najlepsze kombinacje (k, p, fun) wybiorę te które minimalizują błąd średniokwadratowy oraz błąd MAD między prawdziwymi a przewidywanymi klasami. Proporcja błędnych odpowiedzi do wszystkich tak samo traktuje zaklasyfikowanie 2jki jako 10tkę jak również 2jki jako 3jkę, a pierwszy przypadek jest dużo większą pomyłką. Ale wyświetlimy również najwyższą osiągniętą celność.

Zbiór *wine_quality*

MSE

	AggregatingFunction	k	p	ERR	MAD	MSE
40	srednia_a	19	1	0.4206642	0.4605166	0.5446494
80	median	19	2	0.4184502	0.4590406	0.5446494
100	minkara1.5	19	1	0.4206642	0.4605166	0.5446494

MAD

	AggregatingFunction	k	p	ERR	MAD	MSE
80	median	19	2	0.4184502	0.4590406	0.5446494

```
## Najwyższa celność:  58.15498 %
```

Zbiór *skill*

MSE

	AggregatingFunction	k	p	ERR	MAD	MSE
98	minkara1.5	15	1	0.6224888	0.7556222	1.044678

MAD

	AggregatingFunction	k	p	ERR	MAD	MSE
98	minkara1.5	15	1	0.6224888	0.7556222	1.044678

```
## Najwyższa celność:  38.62069 %
```

Zbiór *cement_strength*

MSE

	AggregatingFunction	k	p	ERR	MAD	MSE
21	mode	1	Inf	0.3939698	0.4532663	0.5859296
51	srednia_a	1	Inf	0.3939698	0.4532663	0.5859296
81	median	1	Inf	0.3939698	0.4532663	0.5859296
111	minkara1.5	1	Inf	0.3939698	0.4532663	0.5859296
141	minkara3.0	1	Inf	0.3939698	0.4532663	0.5859296

MAD

	AggregatingFunction	k	p	ERR	MAD	MSE
21	mode	1	Inf	0.3939698	0.4532663	0.5859296
51	srednia_a	1	Inf	0.3939698	0.4532663	0.5859296
81	median	1	Inf	0.3939698	0.4532663	0.5859296
111	minkara1.5	1	Inf	0.3939698	0.4532663	0.5859296
141	minkara3.0	1	Inf	0.3939698	0.4532663	0.5859296

```
## Najwyższa celność:  60.60302 %
```

Zbiór *winconsin_breast_ord*

MSE

	AggregatingFunction	k	p	ERR	MAD	MSE
138	minkara3.0	15	2	0.7684211	1.005263	1.521053

MAD

	AggregatingFunction	k	p	ERR	MAD	MSE
120	minkara1.5	19	Inf	0.7368421	0.9842105	1.531579

```
## Najwyższa celność:  36.31579 %
```

Zbiór *affairs*

MSE

	AggregatingFunction	k	p	ERR	MAD	MSE
48	srednia_a	15	2	0.8754717	1.543396	3.377359

MAD

	AggregatingFunction	k	p	ERR	MAD	MSE
69	median	17	1	0.6830189	1.418868	3.856604

```
## Najwyższa celność:  50.56604 %
```

Podsumowanie

Najlepsza celność jaką udało się nam osiągnąć to około 60% dla zbioru *cement_stregth*. Całkiem nieźle jak na tak prosty klasyfikator. Niestety dużo gorzej sobie poradził na przykład ze zbiorem *skills*. Zazwyczaj najmniejsze błędy obserwowaliśmy dla wysokich k (15-19), najprawdopodobniej dlatego, że są to dość złożone dane (liczba cech) jak na tak prostą metodę i skanując większy obszar dookoła algorytm podejmował lepszą decyzję, wyjątkiem był tu zbiór *cement_strength* gdzie najlepiej sprawdziło się k=1. Metoda k-nn działa dobrze, kiedy grupy utworzone przez klasy są zwarte, najlepiej oddalone od reprezentantów innych klas. Dużo gorzej radzą sobie gdy klasy na siebie nachodzą lub klasa nie tworzy jednego klastra, a na przykład kilka mniejszych. Weźmy za przykład zbiór czerwonych win - nie trudno wyobrazić sobie sytuację, w której byłoby dużo wysokich ocen zarówno wśród win o dużym stężeniu cukru jak i stosunkowo niewielkim, tak samo dobre mogę okazać się wina i mocniejsze i słabsze - w takim przypadku nie będzie jednego, dobrze określonego skupiska dobrych win w przestrzeni cech, a algorytm będzie miał problem z wyłonieniem najbardziej prawdopodobnej etykiety.