Techniki eksploracji danych

Krzysztof Gajowniczek

Rok akademicki: 2020/2021

Section 1

Wykorzystywana notacja matematyczna

Zbiory danych

- $\boldsymbol{D} = \{\boldsymbol{x}_i^T, y_i\}_{i-1}^n$ - zbiór danych.
- $\boldsymbol{U}$ - zbiór uczący.
- $\boldsymbol{V}$ - zbiór walidacyjny.
- $\boldsymbol{T}$ - zbiór testowy

Zagadnienie regresji

$$\hat{y}_i = E(y_i|\boldsymbol{x}_i^T) = f(\boldsymbol{x}_i^T, \boldsymbol{\theta})$$

Zagadnienie klasyfikacji

$$\forall l, \hat{y}_i^{(l)} = P(y_i^{(l)}|\boldsymbol{x}_i^T) = f(\boldsymbol{x}_i^T, \boldsymbol{\theta})$$

Section 2

Normalizacja danych

- Normalizacja danych to skalowanie pierwotnych danych (np.: danych wejściowych) do małego, specyficznego przedziału.
- Na przykład do przedziału $[-1, 1]$ lub $[0, 1]$, czyli przedziałów najbardziej przydatnych podczas rozważania zagadnień związanych np. z sieciami neuronowymi.
- Normalizacja danych poprawia uwarunkowania numeryczne, gdyż nie pracuje się wtedy na "dużych" wartościach i wektorach o dużej długości.
- Normalizację danych można przeprowadzić przy pomocy kilku prostych metod.

Normalizacja min-max

$$x_i' = \frac{x_i - \min_i x_i}{\max_i x_i - \min_i x_i} * (max_{new} - min_{new}) + min_{new}$$

$$x_i' \in [min_{new}, max_{new}]$$

gdzie: max_{new} - nowa wartość maksymalna, min_{new} - nowa wartość minimalna.

Normalizacja Z-score (lub Zero-Mean)

$$x_i' = \frac{x_i - \overline{x}}{sd(x)}, x_i' \in [-\infty, \infty]$$

gdzie: sd - odchylenie standardowe, $\overline{x}$ - średnia arytmetyczna.

Section 3

Miary niepodobieństwa

Aksjomaty miar odległości

- $d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) \geq 0$
- $d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) = 0$ wtedy i tylko wtedy, gdy $\boldsymbol{x}_i^T = \boldsymbol{x}_n^T$
- $d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) = d(\boldsymbol{x}_n^T, \boldsymbol{x}_i^T)$
- $d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) \leq d(\boldsymbol{x}_i^T, \boldsymbol{x}_t^T) + d(\boldsymbol{x}_t^T, \boldsymbol{x}_n^T)$

Skala ilorazowa (zmienne ciągłe) - odległość euklidesa

$$d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) = \sqrt{\sum_{j=1}^{p} (x_i^{(j)} - x_n^{(j)})^2}$$

Skala porządkowa (interwałowa)

$$d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) = \sum_{j=1}^{p} \frac{|x_i^{(j)} - x_n^{(j)}|}{\#x^{(j)} - 1}$$

gdzie # jest mocą zbioru tj. liczbą unikalnych kategorii danej zmiennej.

Skala nominalna - odległość Hamminga

$$d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) = \frac{\sum_{j=1}^{p} I(x_i^{(j)} = x_n^{(j)})}{p}$$

gdzie I jest funkcją indykatorową.

Skala mieszana - odległość Gowera

$$d(\boldsymbol{x}_i^T, \boldsymbol{x}_n^T) = \frac{\sum_{j=1}^{p} G(x_i^{(j)}, x_n^{(j)})}{p}$$

gdzie

$$G(x_i, x_n) = \begin{cases} \frac{|x_i - x_n|}{\max_i x_i - \min_i x_i} & \text{skala ilorazowa} \\ \frac{|z_i - z_n|}{\max_i z_i - \min_i z_i} & \text{skala porządkowa} \\ I(x_i = x_n) & \text{skala nominalna} \end{cases}$$

gdzie

$$z_i = (x_i - 1)/(\max_k x - 1)$$

Section 4

Algorytm k-najbliższych sąsiadów (K-NN)

- Jest to jeden z najprostszych algorytmów uczenia maszynowego, często nauczany jako jeden z pierwszych algorytmów na kursach wprowadzających.
- Jest stosunkowo prosty, ale dość potężny, chociaż rzadko poświęca się czas na zrozumienie jego złożoności obliczeniowej i problemów praktycznych.
- Może być używany zarówno do klasyfikacji, jak i regresji z tą samą złożonością obliczeniową.

- W metodzie tej zaliczamy rozpoznawany obiekt do tej klasy, do której należy większość z jego K najbliższych sąsiadów (w przypadku zagadnienia regresji wyznaczamy wartość średnią w oparciu o K:

Zagadnienie regresji

$$\hat{y}_n = \frac{1}{K} \sum_{i=1}^{n} I(d(\boldsymbol{x}_n^T, \boldsymbol{x}_i^T) \leq d(\boldsymbol{x}_n^T, \boldsymbol{x}_{(K)}^T)) y_i$$

gdzie $\boldsymbol{x}_{(K)}^T$ jest K-tym co do odległości od $\boldsymbol{x}_n^T$ punktem z próby uczącej.

Zagadnienie klasyfikacji

$$\forall l, \hat{y}_n = \frac{1}{K} \sum_{i=1}^{n} I(d(\boldsymbol{x}_n^T, \boldsymbol{x}_i^T) \leq d(\boldsymbol{x}_n^T, \boldsymbol{x}_{(K)}^T)) I(y_i = l)$$

- K-NN jest algorytmem asocjacyjnym - podczas predykcji wyszukuje najbliższych sąsiadów i wylicza odpowiednie statystyki.
- Faza treningowa może, ale nie musi w ogóle istnieć, ponieważ generalnie mamy 2 możliwości:

Metoda Brute force

- Oblicza odległość od nowego punktu do każdego punktu w macierzy danych treningowych $\boldsymbol{U}$.
- Sortuje odległości i przyjmuje K najbliższych, a następnie przeprowadza głosowanie większościowe.
- Nie ma potrzeby oddzielnego szkolenia, więc bierzemy pod uwagę tylko złożoność predykcji.

Korzystanie ze struktur danych

- Organizuje punkty $\boldsymbol{U}$ w pomocniczą strukturę danych w celu szybszego wyszukiwania najbliższych sąsiadów.
- Takie podejście wykorzystuje dodatkową przestrzeń i czas (do tworzenia struktury danych podczas fazy uczenia) w celu szybszego prognozowania.

Subsection 1

Metoda brute force

Złożoność obliczeniowa

- Złożoność czasu szkolenia: $O(1)$
- Złożoność przestrzeni szkoleniowej: $O(1)$
- Złożoność czasowa predykcji: $O(K * n * p)$
- Złożoność przestrzeni predykcyjnej: $O(1)$

- Faza uczenia technicznie nie istnieje, ponieważ wszystkie obliczenia są wykonywane podczas przewidywania, więc mamy $O(1)$ zarówno dla czasu, jak i przestrzeni.
- Faza przewidywania to, jak sugeruje nazwa metody, proste, wyczerpujące wyszukiwanie.
- Jest to struktura zagnieżdżonej pętli, w której zewnętrzna pętla ma K kroków, a wewnętrzna n kroków.
- Trzeci punkt to $O(1)$, a czwarty to $O(K)$.
- Dodatkowo musimy wziąć pod uwagę liczbę wymiarów p, więcej kierunków oznacza dłuższe wektory do obliczania odległości.
- Jeśli chodzi o złożoność przestrzeni, potrzebujemy małego wektora, aby policzyć głosy dla każdej klasy.
- Jest on prawie zawsze bardzo mały i jest stały, więc możemy go traktować jako złożoność przestrzeni $O(1)$.

Subsection 2

Korzystanie ze struktur danych (k-d tree)

k-d tree (u nas p-d)

- Złożoność czasu szkolenia: $O(p * n * \log n)$
- Złożoność przestrzeni szkoleniowej: $O(p * n)$
- Złożoność czasowa predykcji: $O(K * \log n)$
- Złożoność przestrzeni predykcyjnej: $O(1)$

- W fazie uczenia musimy skonstruować drzewo k-d.
- Ta struktura danych dzieli k-wymiarową przestrzeń - tutaj k (u nas p) oznacza k wymiarów przestrzeni, nie mylić tego z K jako liczbą najbliższych sąsiadów!
- Umożliwia szybsze wyszukiwanie najbliższych punktów, ponieważ „wiemy, gdzie szukać” w tej przestrzeni.
- Można myśleć o tej strukturze jak o uogólnieniu BST (inarne drzewo poszukiwań) na wiele wymiarów.

- Konstruowanie k-d drzewa nie jest samo w sobie zadaniem uczenia maszynowego, ponieważ wywodzi się z dziedziny geometrii obliczeniowej.
- Złożoność czasowa jest zwykle $O(p * n * \log n)$, ponieważ wstawienie to $O(\log n)$ (podobnie do zwykłego BST) i mamy n punktów ze zbioru danych uczących, każdy z wymiarami d (u nas p).
- Złożoność przestrzeni wynosi $O(p * n)$ - zależy to od wymiarowości p, ponieważ więcej wymiarów odpowiada większej liczbie podziałów przestrzeni i większym drzewom (oprócz większej złożoności czasowej z tego samego powodu).

- Jeśli chodzi o fazę predykcji, struktura drzewa k-d naturalnie obsługuje operację „K najbliższego punktu sąsiadów", co jest dokładnie tym, czego potrzebujemy dla K-NN.
- Prostym podejściem jest po prostu zapytać drzewo K razy, usuwając znaleziony punkt za każdym razem - ponieważ zapytanie przyjmuje $O(\log n)$, to w sumie jest $O(K * \log n)$.
- Ale ponieważ drzewo k-d już wycina przestrzeń podczas budowy, po jednym zapytaniu w przybliżeniu wiemy, gdzie szukać - możemy po prostu przeszukać „otoczenie" wokół tego punktu.

- Powyższe złożoności są uśrednione, przy założeniu zrównoważonego drzewa k-d.
- Zakładane powyżej czasy $O(\log n)$ mogą spaść do $O(n)$ dla drzew niezrównoważonych, ale jeśli mediana jest używana podczas konstrukcji drzewa, zawsze powinniśmy otrzymać drzewo z wstawieniem około $O(\log n)$ / usuwanie / złożoność wyszukiwania.

Subsection 3

Korzystanie ze struktur danych (ball tree)

ball tree

- Złożoność czasu szkolenia: $O(p * n * \log n)$
- Złożoność przestrzeni szkoleniowej: $O(p * n)$
- Złożoność czasowa predykcji: $O(K * \log n)$
- Złożoność przestrzeni predykcyjnej: $O(1)$

- Algorytm ball tree to inne podejście do podziału przestrzeni, w której znajdują się punkty treningowe.
- W przeciwieństwie do drzew k-d, które dzielą przestrzeń za pomocą „cięć” o wartości środkowej, ball tree grupuje punkty w „kule” zorganizowane w strukturę drzewa.
- Przechodzą od największego (korzeń, ze wszystkimi punktami) do najmniejszego (liście, tylko z kilkoma lub nawet 1 punktem).
- Umożliwia szybkie wyszukiwanie najbliższego sąsiada, ponieważ sąsiedzi są w tych samych “kulach” lub przynajmniej blisko siebie.

Subsection 4

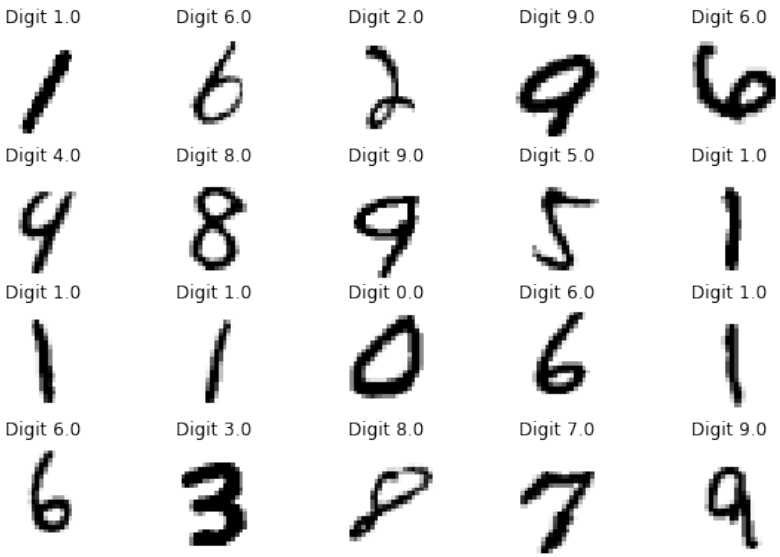
Wybór metody w praktyce

- Metoda brute force ma tylko jedną złożoność do przewidywania, $O(K * n)$.
- Inne metody muszą najpierw utworzyć strukturę danych, więc do uczenia i przewidywania uzyskują $O(p * n * \log n + K * \log n)$, nie biorąc pod uwagę złożoności przestrzeni, która może być również ważna.
- Dlatego tam, gdzie budowa drzew jest częsta, faza uczenia może przeważyć nad zaletami szybszego wyszukiwania najbliższego sąsiada.

- k-d tree czy ball tree?
- Zależy to od struktury danych - stosunkowo jednolite lub „dobrze zachowujące się” dane pozwolą lepiej wykorzystać drzewo k-d, ponieważ wycięcia przestrzeni będą działać dobrze (najbliższe punkty będą blisko liści po wszystkich cięciach).
- W przypadku bardziej skupionych danych „kule” z k-d tree będą lepiej odzwierciedlać strukturę, a tym samym umożliwią szybsze wyszukiwanie najbliższego sąsiada.

Zbiór danych MNIST

- MNIST ma 60,000 obrazów szkoleniowych i 10,000 obrazów testowych.



- Brute force $O(K * n)$: $3 * 10,000 = 30,000$
- k-d tree $O(K * \log n)$: $3 * \log 10,000 \sim 3 * 13 = 39$
- Porównanie: 39 / 30,000 = 0.0013

Section 5

Literatura

- *Friedman, J., Hastie, T., Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.*
- *Walesiak, M. (2002). Pomiar podobieństwa obiektów w świetle skal pomiaru i wag zmiennych.*
- *Gower, J. C. (1971). A general coefficient of similarity and some of its properties. Biometrics, 857-871.*
- *Implementacja K-d tree w różnych językach programowania.*