mgr inż. Marcin Mrukowicz

Algorytm fuzzy knn

Pierwszym krokiem w algorytmie fuzzy knn jest przypisanie **obiektom treningowym** stopnia przynależenia do każdej z klas decyzyjnych. Stopień przynależenia mieści się w zakresie od 0 do 1. Można wykorzystać „etykietowanie ostre" (ang. crisp labeling), gdzie jeżeli obiekt należy do danej klasy to przypisujemy mu wartość 1, a 0 gdy nie należy do tej klasy. Formalnie:

$$u_j(x) = \begin{cases} 1, & x\ należy\ do\ klasy\ decyzyjnej\ j \\ 0, & x\ nie\ należy\ do\ klasy\ decyzyjnej\ j \end{cases}$$

Autorzy podali jednak również bardziej złożony wzór, który pozwala wyliczyć „rozmytą etykietę":

$$u_j(x) = \begin{cases} 0.51 + (\frac{n_j}{k_{init}}) \cdot 0.49, & x\ należy\ do\ klasy\ decyzyjnej\ j \\ (\frac{n_j}{k_{init}}) \cdot 0.49, & x\ nie\ należy\ do\ klasy\ decyzyjnej\ j \end{cases}$$

gdzie k_{init} to liczba jego najbliższych sąsiadów (spośród obiektów treningowych) jakie bierzemy pod uwagę, a n_j to liczba tych sąsiadów, którzy należą do klasy decyzyjnej *j*. Parametr k_{init} służy do inicjalizacji algorytmu i może przyjąć inną wartość niż parametr k, służący do ustalania decyzji obiektów testowych. Jeżeli przykładowo ustalono k_{init} na 3 i **wszyscy Ci sąsiedzi** mieli klasę decyzyjną 0, oraz dany obiekt treningowy **miał klasę 0**, to jego u_0 = 1.0 (ponieważ $0.51 + (\frac{3}{3}) \cdot 0.49 = 1.0$) i oczywiście u_1 = 0.0 =$\frac{0}{3} \cdot 0.49 = 0.0$. Jeżeli jednak przykładowo tylko 2 spośród jego sąsiadów miało klasę decyzyjną 0, a jeden miał klasę decyzyjną 1 to $u_0 \approx$ 0,8366 (ponieważ $0.51 + (\frac{2}{3}) \cdot 0.49 \approx 0.836$) a $u_1 \approx$ 0,1633 $\approx \frac{1}{3} \cdot 0.49$. Wzór ten może być stosowany również gdy liczba klas decyzyjnych jest większa niż 2. Można zauważyć, że stosując ten wzór, jeżeli klasy są od siebie odseparowane, to jedynie obiekty leżące na pograniczu klas decyzyjnych będą otrzymywać etykiety pomiędzy 0 a 1, a obiekty oddalone od granicy decyzji będą otrzymywały etykiety będące albo 0 albo 1.

Kolejnym krokiem algorytmu jest przypisanie klas decyzyjnych (etykiet) obiektom testowym. Dla każdego obiektu testowego najpierw znajdujemy jego k-najbliższych sąsiadów. Następnie wyliczamy stopień przynależenia **obiektu testowego** do każdej z klas decyzyjnych j:

$$u_j(x) = \frac{\sum_{i=1}^{k} u_{ij} \cdot \left(\frac{1}{d(x, x_i)^{\frac{2}{(m-1)}}} \right)}{\sum_{i=1}^{k} \left(\frac{1}{d(x, x_i)^{\frac{2}{(m-1)}}} \right)}$$

gdzie u_{ij} to etykieta obiektu treningowego *i* (najbliższego sąsiada o indeksie *i*, czyli x_i), $d(x, x_i)$ to odległość obiektu *x* od x_i a $m > 1$ to parametr określający, jaki wpływ na decyzję będą mieć dalsi sąsiedzi. Stopień przynależenia obiektu to średnia ważona; Jeśli *m* wynosi dwa, wówczas wkład każdego sąsiedniego punktu jest ważony odwrotnością jego odległości od punktu klasyfikowanego. Wraz ze wzrostem *m* sąsiedzi są ważeni bardziej równomiernie , a ich względne odległości od punktu klasyfikowanego mają coraz mniejszy wpływ na końcowy wynik. Gdy m zbliża się do 1, bliżsi sąsiedzi

są ważeni znacznie silniej niż ci bardziej oddaleni, co powoduje, że tylko najbliżsi sąsiedzi mają istotny wpływ na wartość przynależenia obiektu do danej klasy decyzyjnej.

Algorytm fuzzy C-means

Algorytm fuzzy C-means został zaproponowany jako uogólnienie algorytmu k-means. Algorytm fuzzy C-means **minimalizuje następującą funkcję celu**:

$$J = \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^{m} d_{ij}^{2}, m \geq 1$$

gdzie *C* to zadana liczba skupień (klastrów), *N* to liczba obiektów uczących, d_{ij} to odległość obiektu *i* od środka skupienia (klastru) *j*, natomiast u_{ij} to miara przynależenia obiektu *i* do klastra *j*. Parametr *m* nazywany tutaj parametrem rozmycia (ang. fuzzifier parameter) określa stopień w jakim klastry będą współdzielić obiekty położone przy ich granicach. W klasteryzacji ostrej mogą być obiekty bliskie siebie, ale należące do różnych klastrów. W przypadku klasteryzacji rozmytej takie obiekty będą otrzymywać częściowe przynależenie do więcej niż jednego klastra. Większe wartości *m* będą w praktyce powodować większą skłonność algorytmu do obniżania stopnia przynależenia obiektów do jednego dominującego klastra na rzecz przypisywania go do więcej niż jednego klastra w podobnym stopniu. **Jeżeli ustali się parametr *m* na 1, to wtedy fuzzy c-means sprowadzi się do k-means (zakładając dodatkowo że u_{ij} może przyjąć jedynie wartości {0,1}).**

Pierwszym krokiem algorytmu jest inicjalizacja początkowych wartości u_{ij}. Może być ona **losowa**, albo jeżeli posiadamy pewną wiedzę o problemie predefiniowana. Tak samo jak w k-means inicjalizacja ma bardzo duże znaczenie, gdyż algorytm nie gwarantuje uzyskania optymalnego rozwiązania w każdym uruchomieniu, stąd powinna być powtórzona wielokrotnie w celu ustalenia optymalnego rezultatu.

Drugim krokiem jest policzenie centroidu (środka klastra) na bazie początkowych wartości u_{ij} dla każdego klastra *j*:

$$v_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

I następnie odległości obiektów uczących od każdego centroidu:

$$\boldsymbol{d_{ij} = \left\| x_i - v_j \right\|^2}$$

Trzecim krokiem jest aktualizacja u_{ij}:

$$u_{ij} = \left(\sum_{k=1}^{C} \left(\frac{d_{ij}}{d_{ik}} \right)^{\frac{2}{m-1}} \right)^{-1}, \text{m > 1}$$

$$u_{ij} = \begin{cases} 1 & jeżeli\ d_{ij} = \min\limits_{k} d_{ik} \\ 0\ w\ przeciwnym\ przypadku \end{cases}, \text{m = 1}$$

Ponadto jeżeli d_{ij}=0 czyli x_i=v_j, u_{ij}=1. Chodzi tutaj o przypadek że dany punkt jest identyczny z danym środkiem klastra, zatem przynależy do tego klastra w stopniu 1. Jeżeli istnieją dwa lub więcej identyczne środki klastrów to ustawiamy każdemu z nich przynależność $\frac{1}{r}$, gdzie r to liczba identycznych środków klastrów.

W czwartym kroku obliczamy wartość funkcji celu J. Jeżeli nowo policzona wartość nie różni się znacząco od poprzedniej, algorytm uzyskuje konwergencję i kończy działanie. Formalnie jeżeli $|J_{new} - J_{old}| < \varepsilon$ algorytm kończy działanie, w przeciwnym wypadku powracamy do kroku 2.

Algorytm fuzzy C-means z uwagi na fakt, że pozwala przypisać obiektom więcej niż jeden klaster jednocześnie, w dodatku w różnym stopniu może sprawdzać się lepiej niż klasyczne algorytmy analizy skupień w zadaniach takich jak wykrywanie anomalii, diagnoza medyczna (np. przypadek wątpliwy oznaczamy jako osobę w jakimś stopniu zdrową, a w jakimś cierpiącą na daną chorobę). W przypadku, gdy w danych nie znajdują się ściśle odseparowane wzorce klasteryzacja rozmyta również może okazać się bardziej przydatna.

ZADANIA

Przed przystąpieniem do wykonywania zadań i/lub trakcie ich wykonywania warto zaglądnąć do przypomnienia wiadomości ze sztucznej inteligencji (osobny plik).

1. Uruchom skrypt *BMI.py*. Odpowiedz na następujące pytania:
    - Jakie atrybuty warunkowe występują w przykładowym zbiorze danych?
    - Jaki atrybut decyzyjny występuje w przykładowym zbiorze danych?
    - W jakiej proporcji podzielono dane na treningowe i testowe?
    - Omów uzyskaną przez algorytm macierz pomyłek; czy algorytm poprawnie przypisuje klasy decyzyjne?
    - W jakim celu obliczono miary jakości za pomocą kroswalidacji?
    - Co zaprezentowano na wykresie w przestrzeni dwuwymiarowej?

    Wykonaj następujące zadania:

    - Zmień rozmiar zbioru testowego na 30% i ponownie uruchom skrypt, jak, wpłynęło to na uzyskane accuracy i macierz pomyłek?
    - Zmień liczbę k-najbliższych sąsiadów na odpowiednio 1 i 5. Jak wpłynęło to na uzyskiwane wyniki?
    - Zmień liczbę foldów w kroswalidacji na 3 i oceń uzyskane wyniki.

Na marginesie warto zauważyć, że podany przykład ma charakter dydaktyczny i jest znacznie uproszczony. Przede wszystkim liczba obiektów uczących jest dość mała i algorytm podczas podziału na części treningowe i testowe jest narażony na błędną estymację granicy decyzji, bo zwyczajnie zna za mało obiektów, aby potrafić ją prawidłowo rozpoznawać.

2. Uruchom skrypt *BMI_more_data.py*. Odpowiedz na następujące zadania:
    - Opisz co to jest DataFrame w pandas. Opisz jak wykorzystano bibliotekę pandas do załadowania danych do uczenia.
    - Opisz czym różni się zbiór danych w tym zadaniu od zbioru danych w zadaniu poprzednim.
    - Jaką istotną różnicę w działaniu widać pomiędzy klasyfikacją za pomocą algorytmu k-nn a klasteryzacją wykonywaną za pomocą algorytmów K-Means i DBSCAN i fuzzy c-means?
    - Czym różni się wynik działania algorytmu fuzzy c-means od k-means?

- Oblicz programowo w języku Python z macierzy pomyłek precyzję (ang. precision) ze wzoru $precision = \frac{TP}{TP+FP}$. Na podstawie dowolnego źródła, opisz tę miarę jakości klasyfikacji.

3. Uruchom skrypt *BMI_fuzzy_knn.py*. Wykonaj następujące zadania:
    - Porównaj macierze pomyłek uzyskane przez algorytmy k-nn i fuzzy k-nn.
    - Opisz co jest wynikiem wywołania metody *predict_proba* w algorytmie fuzzy k-nn.
    - Porównaj wynik wywołania metody *predict* w algorytmie fuzzy k-nn i wywołania funkcji *np.argmax* na metodzie *predict_proba* w tym samym algorytmie. Co zauważasz?
    - Który z algorytmów uzyskał wyższe wyniki w kroswalidacji?
    - Opisz różnice w wizualizacji działania algorytmu k-nn w wersji ostrej (crisp) a wersji rozmytej. Czy wizualizacja reprezentuje dokładnie taką samą ilość wiedzy?
    - Zauważ, że ustawiono wartości k_init=1 i m=1.1. Porównaj wyniki klasyfikatora dla wartości m={1.1, 2, 4, 10, 100}, szczególnie jeżeli chodzi o wizualizację jego przynależenia do atrybutu decyzyjnego.
    - Porównaj wyniki klasyfikatora dla wartości k_init ={1, 3, 5, 10}. Czy k_init dla tego zbioru uczącego ma wpływ na wynik?
4. Uruchom skrypt *seismic.py*. Wykonaj następujące zadania:
    - Usuń skalowanie atrybutów warunkowych. Czy zmieniło to jakość klasyfikacji?
    - Zauważ, że usunięto atrybuty typowo kategoryczne. Zamiast ich usunięcia zastosuj LabelEncoder.
    - Porównaj działanie fuzzy knn dla różnych wartości m i k_init, k.

5. Wykonaj następujące zadania:
    - Usuń skalowanie atrybutów warunkowych. Czy zmieniło to jakość klasyfikacji?
    - Zauważ, że usunięto atrybuty typowo kategoryczne. Zamiast ich usunięcia zastosuj LabelEncoder w osobnym pliku *seismic_label_encoder.py*
    - Porównaj działanie fuzzy knn dla różnych wartości m i k_init, k.

6. * Zadanie dodatkowe. Uruchom skrypt Iris_multiclass_example.py. Jest to zbiór danych wielodecyzyjnych (ang. multi-class). Wykonaj następujące zadania:
    - Opisz czym różnią się macierze pomyłek od klasycznych wersji binarnych.
    - Zauważ, że w kroswalidacji nie można było zastosować miary ROC AUC, ponieważ miara ta jest zdefiniowana dla klasyfikacji binarnej. Zastosowano jednak dekompozycję OVO (ang. one versus one) tego problemu i policzono odpowiednie uogólnienie tej miary. Na podstawie dowolnego źródła wiedzy, opisz co to jest dekompozycja OVO i jak można zastosować ją do obliczenia krzywej ROC w problemie klasyfikacji wielodecyzyjnej.
    - Opisz czym wizualizacja działania fuzzy k-nn różni się względem wersji binarnej.