Sztuczna Inteligencja - Laboratorium

Robert Benke, Karol Draszawka, Szymon Olewniczak, Julian Szymański

March 2023

Rozdział 1

Regresja

Regresja jest jednym z podstawowych problemów uczenia maszynowego, wpisującym się w szerszą kategorię uczenia nadzorowanego. W problemach regresji naszym zadaniem jest znalezienie funkcji $f(\vec{x})$, która na podstawie wejściowego wektora obserwacji $\vec{x}$, możliwie najdokładniej oszacuje wartość interesującej nas cechy wynikowej. Przykładem problemu regresji może być szacowanie ceny mieszkania na podstawie cech takich jak powierzchnia, dzielnica, odległość od centrum. W tym wypadku nasz wejściowy wektor obserwacji składa się z trzech cech, a cena mieszkania stanowi naszą cechę wynikową.

W regresji, podobnie jak w innych problemach z dziedziny uczenia nadzorowanego, nie jesteśmy w stanie określić dokładnego wzoru funkcji $f(\vec{x})$. Jedyne co posiadamy to pewien zbiór wektorów obserwacji X powiązanych z rzeczywistymi wartościami cech wynikowych Y. Na podstawie tej ograniczonej informacji, chcemy stworzyć model, który sprawdzałby się dobrze w ogólnym przypadku.

1.1 Regresja liniowa

Regresja liniowa jest jedną z najprostszych technik wyznaczania przybliżonego wzoru funkcji $f(\vec{x})$ na podstawie posiadanego zbioru obserwacji. W modelu tym zakładamy, że naszą cechę wynikową (zwaną zmienną zależną) można wyliczyć na podstawie liniowej kombinacji cech wejściowych (zwanych zmiennymi niezależnymi):

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1.1}$$

gdzie $\hat{y}$ jest naszą prognozowaną wartością cechy wyjściowej, n określa liczbę cech wejściowych modelu, x_i to wartość i-tej cechy, a θ_j to j-ty parametr modelu, gdzie θ_0 stanowi wyraz wolny, zwany punktem obciążenia (*bias term*). Równanie 1.1 możemy zapisać także w formie wektorowej:

$$\hat{y} = \vec{\theta} \circ \vec{x} \tag{1.2}$$

gdzie $\vec{x}$ stanowi rozszerzony wektor cech wejściowych $\vec{x} = [1, x_1, \cdots, x_n]$.

Zadaniem naszego algorytmu uczenia maszyn jest znalezienie wektora parametrów $\vec{\theta}$ dla którego błąd pomiędzy wartością oczekiwaną y, a oszacowaną $\hat{y}$ będzie możliwie najmniejszy. Proces ten nazywamy treningiem modelu.

1.2 Podział danych

Mając do dyspozycji zbiór obserwacji powiązanych z rzeczywistymi cechami wynikowymi, z reguły zanim przystąpimy do treningu modelu, dokonujemy jego podziału na dwa podzbiory. Pierwszy z nich nazywamy zbiorem treningowym, a drugi testowym. Podziału takiego najczęściej dokonuje się w sposób losowy, przydzielając większość danych do zbioru treningowego (popularną proporcją jest np. 80% dane treningowe, 20% dane testowe). Dane treningowe służą nam do treningu modelu, czyli na ich podstawie staramy się określić optymalne wartości wektora $\vec{\theta}$. Dane testowe wykorzystujemy natomiast w celu oszacowania rzeczywistej jakości modelu na danych, które nie były dla niego dostępne podczas treningu. Z reguły, jak na to wskazuje intuicja, wyniki modelu są lepsze dla danych treningowych, niż dla danych testowych.

1.3 Funkcja kosztu

Funkcja kosztu pozwala określić nam różnicę pomiędzy rzeczywistymi wartościami cech wyjściowych, a wartościami wyliczonymi przez model, czyli innymi słowy błąd naszego modelu. Dla problemu regresji często wykorzystywaną funkcją kosztu jest błąd średnio-kwadratowy (*mean square error, MSE*):

$$MSE(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\vec{\theta} \circ \vec{x}^{(i)} - y^{(i)})^2 \tag{1.3}$$

gdzie m - to liczba elementów w zbiorze obserwacji, $\vec{\theta} \cdot \vec{x}^{(i)}$ to predykcja naszego modelu dla i-tego wektora cech, a $y^{(i)}$ to rzeczywista wartość cechy wyjściowej dla i-tego wektora cech.

Funkcję kosztu wykorzystujemy podczas oceny jakości naszego modelu przy wykorzystaniu zbioru testowego, jak i również w niektórych algorytmach treningu.

1.4 Gradient funkcji

Gradient funkcji jest jednym z centralnych pojęć w dziedzinie uczenia maszynowego. Gradientem funkcji f nazywamy wektor pochodnych cząstkowych funkcji po wszystkich jej argumentach:

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n} \right] \tag{1.4}$$

Zwrot wektora gradientu funkcji wskazuje nam kierunek najszybszego wzrostu wartości funkcji w danym punkcie.

W uczeniu maszynowym gradient funkcji wykorzystujemy najczęściej w celu wyznaczeniu wpływu poszczególnych wartości wektora parametrów $\vec{\theta}$ na wartość funkcji kosztu:

$$\nabla_\theta MSE(\theta) = \left[\frac{\partial}{\partial\theta_0}MSE(\theta), \cdots, \frac{\partial}{\partial\theta_n}MSE(\theta)\right] \tag{1.5}$$

W przypadku regresji liniowej $\frac{\partial}{\partial\theta_j}MSE(\theta)$ przyjmuje postać:

$$\frac{\partial}{\partial\theta_j}MSE(\theta) = \frac{2}{m}\sum_{i=1}^{m}(\vec{\theta}\circ\vec{x}^{(i)} - y^{(i)})x_j^{(i)} \tag{1.6}$$

gdzie $x_j^{(i)}$ to wartość j-tej cechy i-tego wektora wejściowego.

Równianie 1.5 możemy również zapisać w postaci macierzowej jako:

$$\nabla_\theta MSE(\theta) = \frac{2}{m}\dot{X}^T(\dot{X}\theta - y) \tag{1.7}$$

gdzie $\dot{X}$ to macierz obserwacji o wymiarach $(m \times n+1)$:

$$\dot{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix} \tag{1.8}$$

θ to wektor kolumnowy $(n+1 \times 1)$ reprezentujący wagi naszego modelu:

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} \tag{1.9}$$

a y to wektor kolumnowy $(m \times 1)$ reprezentujący wartości rzeczywiste odpowiadające poszczególnym wektorom obserwacji:

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{1.10}$$

Gradient funkcji kosztu pozwala nam odpowiedzieć na bardzo ważne pytanie: jak powinniśmy zmienić wartości wektora parametrów θ, aby dla danego zbioru obserwacji $\dot{X}$ błąd pomiędzy wartościami rzeczywistymi y, a predykcją modelu $\hat{y}$ zmniejszył się.

1.5 Rozwiązanie jawne

Regresja liniowa jest jednym z nielicznych modeli uczenia maszynowego, który posiada tzw. rozwiązanie jawne (*closed-form solution*). Pozwala ono w sposób analityczny wyznaczyć optymalne wartości wektora parametrów θ, które minimalizują błąd pomiędzy wartościami wyliczonymi $\hat{y}$, a rzeczywistymi y.

Najczęściej stosowaną metodą wyznaczania rozwiązania jawnego dla regresji liniowej jest metoda najmniejszych kwadratów (*least squares*), której celem jest zminimalizowanie dystansu między poszczególnymi punktami danych, a linią regresji:

$$argmin_\theta(\dot{X}\theta - y)^T(\dot{X}\theta - y) \tag{1.11}$$

W celu znalezienia wektora θ rozwiązującego zadany problem optymalizacyjny wyliczamy gradient z minimalizowanej funkcji, a następnie przyrównujemy go do 0 w celu znalezienia minimum:

$$\dot{X}^T(\dot{X}\theta - y) = 0 \tag{1.12}$$

Ostatecznie otrzymujemy równanie, pozwalające wyznaczyć nam optymalny wektor parametrów θ:

$$\theta = (\dot{X}^T\dot{X})^{-1}\dot{X}^T y \tag{1.13}$$

1.6 Metoda gradientu prostego

Niestety dla większości modeli uczenia maszynowego nie jesteśmy w stanie podać wzoru na rozwiązanie jawne. Jedyną metodą jaka nam wówczas pozostaje to iteracyjne poszukiwanie parametrów optymalizujących naszą funkcję straty. Jedną z najczęściej stosowanych metod jest metoda gradientu prostego (*gradient descent*).

W metodzie gradientu prostego rozpoczynamy poszukiwanie optymalnego wektora parametrów θ od wypełnienia go losowymi wartościami (z reguły z przedziału $(0, 1)$). Następnie, w kolejnych iteracjach algorytmu, wyliczamy wartość gradientu dla wybranej funkcji kosztu i aktualizujemy wartość θ przeciwnie do kierunku gradientu:

$$\theta' = \theta - \eta\nabla_\theta MSE(\theta) \tag{1.14}$$

gdzie η to tzw. współczynnik uczenia (*learning rate*), określający tempo dokonywanych zmian. Zbyt niski współczynnik uczenia powoduje, że uczenie przebiega bardzo powoli, z kolei zbyt wysoki sprawi że algorytm nie będzie w stanie zbiec do minimum. Optymalna wartość współczynnika uczenia jest zależna od danych i w celu jego określenia musimy przeprowadzać treningi wiele razy, dla różnych jego wartości. Poszukiwania z reguły rozpoczynamy od wartości 0.1, a następnie próbujemy kolejne niższe potęgi 10: 0.01, 0.001 itd.

Liczba iteracji przez którą należy wykonywać algorytm, również zależy od problemu i jest ściśle powiązana z wartością współczynnika uczenia. Trening modelu kończymy kiedy kolejne iteracje nie zmniejszają już naszej funkcji kosztu.

1.7 Standaryzacja

Czasami kiedy różnice pomiędzy skalami wartości poszczególnych zmiennych są duże, trening z wykorzystaniem metody gradientu prostego staje się niemożliwy. Rozwiązaniem tego problemu jest normalizacja danych. Jedną z najczęściej spotykanych technik normalizacyjnych stanowi standaryzacja Z (*Z-score normalization*). W standaryzacji Z naszym celem jest przekształcenie zmiennych w taki sposób, aby wartość średnia z ich populacji wyniosła 0, a odchylenie standardowe 1. Dokonujemy tego w następujący sposób:

$$z = \frac{x - \mu}{\sigma} \tag{1.15}$$

gdzie x to pierwotna wartość zmiennej, μ to średnia z populacji, a σ to odchylenie standardowe populacji.

W przypadku uczenia maszynowego populację rozumiemy jako zbiór wszystkich wartości wybranej cechy w naszym zbiorze treningowym. Normalizacji danych z reguły dokonujemy zarówno dla zmiennych zależnych, jak i niezależnych. Należy pamiętać że w celu obliczenia średniej i odchylenia standardowego nie należy wykorzystywać zbioru treningowego ze względu na ryzyko wycieku danych testowych (*testing dataset leakage*). W celu pomiaru jakości modelu na zbiorze testowym, należy go znormalizować przy wykorzystaniu średniej i odchylenia standardowego wyliczonego na zbiorze treningowym.

Rozdział 2

Algorytm genetyczny

Algorytm genetyczny (*genetic algorithm, GA*) to zbiorczy termin, którym określamy algorytmy optymalizacyjne inspirowane procesem doboru naturalnego. Nie stanowi on jednej konkretnej procedury, a zbiór różnych technik i praktyk, które mogą zostać wykorzystanie przy rozwiązywaniu konkretnego problemu. Algorytm genetyczny stanowi przykład metody heurystycznej, czyli nie dającej nam gwarancji znalezienia optymalnego rozwiązania, jednak próbującej się do niego jak najbardziej zbliżyć.

Aby zastosować algorytm genetyczny dla zadanego problemu optymalizacyjnego, muszą zostać spełnione dwa warunki:

1. dopuszczalne rozwiązania problemu muszą być możliwe do zakodowania w formie liczbowej,
2. musimy być w stanie zdefiniować tzw. funkcję dopasowania (*fitness*), która ocenia w sposób ilościowy jakość poszczególnych rozwiązań.

W przeciwieństwie do technik optymalizacyjnych wykorzystujących metodę gradientową, w algorytmie genetycznym nie wymagamy aby funkcja dopasowania była różniczkowalna. Umożliwia to między innymi na zastosowanie tego algorytmu do rozwiązywania problemów NP-trudnych, dla których z przyczyn wydajnościowych, często nie możemy wykorzystać algorytmów dokładnych.

2.1 Schemat działania

Algorytm genetyczny składa się z następujących kroków:

1. Tworzenie początkowej populacji rozwiązań
2. Wybór rodziców
3. Tworzenie kolejnego pokolenia
4. Mutacja

5. Aktualizacja populacji rozwiązań

Korki 2-5 powtarzamy aż do osiągnięcia zdefiniowanego kryterium stopu. W najprostszym przypadku może to być po prostu określona z góry liczba iteracji algorytmu.

2.1.1 Tworzenie początkowej populacji rozwiązań

W algorytmie genetycznym populacją nazywamy zbiór możliwych rozwiązań problemu optymalizacyjnego. Z kolei pojedyncze rozwiązanie nazywamy osobnikiem. W pierwszym kroku algorytmu generujemy początkowy zbiór możliwych rozwiązań problemu. Elementy do tego zbioru wybieramy w sposób losowy z przestrzeni wszystkich możliwych rozwiązań. Liczba osobników w początkowej populacji stanowi jeden z parametrów algorytmu.

2.1.2 Wybór rodziców

W algorytmie genetycznym nowe rozwiązania problemu generujemy na postawie łączenia ze sobą najlepszych dotychczasowych rozwiązań. Proces ten rozpoczynamy od wyboru z populacji najlepiej dopasowanych osobników (w sensie funkcji dopasowania), którzy następnie zostaną wykorzystani jako rodzice kolejnego pokolenia. Ten etap algorytmu nazywany jest selekcją. Liczba wybieranych rodziców stanowi parametr algorytmu.

Istnieje kilka popularnych algorytmów selekcji. Podstawowym problemem przed jakim one stają jest pogodzenie ze sobą dwóch sprzecznych wyzwań. Po pierwsze chcemy, aby jako rodzice kolejnego pokolenia wybierane były najlepiej dopasowane osobniki. Z drugiej strony chcemy uniknąć sytuacji, kiedy zbyt szybko zawęzimy się jedynie do wąskiego grona obecnie najlepszych rozwiązań, co może skutkować ugrzęźnięciem algorytmu w ekstremum lokalnym.

Selekcja ruletkowa

Selekcja ruletkowa (*roulette wheel selection*) stanowi jeden z najpopularniejszych algorytmów selekcji. W algorytmie tym każdemu osobnikowi w populacji przypisujemy prawdopodobieństwo wyboru, proporcjonalne do jego dopasowania:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.1}$$

gdzie f_i to wartość funkcji dopasowania i-tego osobnika, a N to wielkość naszej populacji. Wyliczone wartości tworzą nam rozkład prawdopodobieństwa. W celu przeprowadzenia selekcji próbkujemy wybraną liczbę razy z powstałego rozkładu. W wyniku działania selekcji ruletkowej jeden osobnik może być wybrany wiele razy.

2.1.3 Tworzenie kolejnego pokolenia

Kolejnym etapem algorytmu genetycznego jest utworzenie kolejnego pokolenia osobników. W tym celu łączymy rodziców w pary. Na podstawie każdej z pary rodziców w tzw. procesie krzyżowania (*crossover*) tworzymy zbiór dzieci. Liczba tworzonych dzieci dla jednej pary rodziców jest zależna od konkretnej metody krzyżowania. Wynikiem tego etapu jest zbiór osobników kolejnego pokolenia.

2.1.4 Mutacja

Następny krok algorytmu genetycznego stanowi tzw. mutacja (*mutation*). W etapie tym wprowadzamy pewne drobne losowe zmiany w osobnikach z nowego pokolenia. Celem mutacji jest zwiększenie różnorodności naszej populacji w celu uniknięcia ugrzęźnięcia procedury w ekstremum lokalnym.

2.1.5 Aktualizacja populacji rozwiązań

Ostatnim krokiem algorytmu genetycznego jest aktualizacja populacji rozwiązań. Możemy tutaj wykorzystać jeden z dwóch popularnych modeli: stanu ustalonego albo pokoleniowy. W modelu stanu ustalonego zachowujemy większość dotychczasowej populacji, zastępując nowymi osobnikami tylko część najsłabiej dopasowanych osobników. W celu odrzucenia najsłabiej dopasowanych możemy zastosować algorytm selekcji. Z kolei w modelu pokoleniowym zastępujemy całą dotychczasową populację nowymi osobnikami. Wybór konkretnego podejścia jest zależny od problemu.

Popularnym rozwinięciem etapu aktualizacji populacji jest wprowadzenie tzw. elityzmu. W aktualizacji populacji z elityzmem do następnej populacji zawsze przenosimy pewną stałą liczbę najlepiej dopasowanych osobników z poprzedniej populacji.