Metody systemowe i decyzyjne w informatyce

Laboratorium – Python – Zadanie nr 3

Regresja logistyczna

autorzy: A. Gonczarek, J.M. Tomczak, S. Zaręba, M. Zięba, J. Kaczmar

Cel zadania

Celem zadania jest zaimplementowanie modelu regresji logistycznej wraz z algorytmami uczącymi do zadania detekcji twarzy na zdjęciu.

Detekcja twarzy jako problem klasyfikacji

Problem detekcji twarzy na obrazie $\mathcal{I}$ polega na zaklasyfikowaniu wybranych fragmentów obrazu jako twarzy człowieka. Korzystając z techniki okna przesuwnego (ang. *shifting window*) obraz dzielony jest na prostokąty, na których dokonywana jest klasyfikacja. Każdy wyszczególniony fragment obrazu opisany jest za pomocą wektora cech $\mathbf{x} = (1, \phi^1(\mathcal{I}) \dots \phi^{D-1}(\mathcal{I}))^{\mathrm{T}}$ otrzymanych za pomocą deskrytporów HOG (ang. *Histogram of Oriented Gradients*).[1] Dla każdego fragmentu obrazu należy rozwiązać problem klasyfikacji z dwoma klasami $y \in \{0, 1\}$, polegający na przypisaniu fragmentowi obrazu etykiety twarzy, $y = 1$, lub braku twarzy, $y = 0$.

Zdefiniujmy prawdopodobieństwo wystąpienia twarzy na zadanym fragmencie obrazu reprezentowanym przez cechy $\mathbf{x}$ za pomocą funkcji sigmoidalnej:

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}), \tag{1}$$

gdzie $\mathbf{w}$ oznacza D-wymiarowy wektor parametrów, funkcja sigmoidalna:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{2}$$

Klasyfikacja zadanego fragmentu odbywa się poprzez sprawdzenie, czy prawdopodobieństwo wystąpienia twarzy jest większe niż zadana wartość progowa $\theta \in [0, 1]$:

$$y^* = \begin{cases} 1, & \text{jeśli } p(y = 1|\mathbf{x}, \mathbf{w}) \geqslant \theta, \\ 0, & \text{w przeciwnym przypadku.} \end{cases} \tag{3}$$

Klasyfikator w tak podanej postaci nazywa się **regresją logistyczną**.

[1] Sposób otrzymania deksryptorów nie jest istotny dla zadania i jest pominięty w opracowaniu. Zainteresowanych odsyłamy do Wikipedii, gdzie można znaleźć odnośniki do prac źródłowych `http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients`.

Uczenie modelu jako problem optymalizacji

Dla tak określonego problemu detekcji twarzy kluczowym aspektem jest wyznaczenie parametrów $\mathbf{w}$, czyli uczenie modelu. Zakładamy, że dysponujemy zbiorem treningowym $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, gdzie $\mathbf{x}_n$ jest zestawem fragmentów obrazów reprezentowanym przez cechy HOG oraz wartością stałą równą 1, y_n jest etykietą obrazu określającą, czy na fragmencie widnieje twarz, czy nie.

W celu uczenia modelu wykorzystamy metodę **maksymalnej wiarygodności** (ang. *maximum likelihood*).

Dla zadanego $\mathbf{x}$, zmienna losowa y jest zmienną losową binarną o następującym rozkładzie:

$$p(y|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})^y(1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}))^{1-y}. \tag{4}$$

Wprowadźmy następujące oznaczenie:

$$\sigma_n \equiv \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n). \tag{5}$$

Dla rozkładu (4) i oznaczenia (5), funkcja wiarygodności przyjmuje następującą postać:

$$p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} \sigma_n^{y_n}(1 - \sigma_n)^{1-y_n}. \tag{6}$$

Biorąc negatywny logarytm otrzymujemy:

$$-\ln p(\mathcal{D}|\mathbf{w}) = -\sum_{n=1}^{N} \Big(y_n \ln \sigma_n + (1 - y_n) \ln(1 - \sigma_n)\Big). \tag{7}$$

Funkcję celu uczenia definiujemy w oparciu o negatywny logarytm funkcji wiarygodności, którą dalej oznaczamy w następujący sposób:[2]

$$\boxed{L(\mathbf{w}) = -\frac{1}{N} \ln p(\mathcal{D}|\mathbf{w}).} \tag{8}$$

Zadanie wyznaczenia wartości parametrów minimalizujących funkcję celu jest zadaniem optymalizacji bez ograniczeń:

$$\boxed{\text{minimalizuj (po } \mathbf{w}) \quad L(\mathbf{w})}$$

Zauważmy, że postać σ_n zależy od parametrów $\mathbf{w}$, co uniemożliwia wyznaczenia analitycznego rozwiązania (7). Dlatego należy posłużyć się rozwiązaniem numerycznym. W tym celu zastosujemy **algorytm gradientu prostego** i **algorytm stochastycznego gradientu prostego**.

[2] Przemnożenie funkcji celu przez stałą nie zmienia rozwiązania, natomiast przeskalowanie negatywnego logarytmu funkcji wiarygodności przez odwrotność liczby danych pozwoli na uniezależnienie się od liczności zbioru danych i w konsekwencji na prostsze porównanie rozwiązań.

Algorytm gradientu prostego

Algorytm gradientu prostego przedstawiono poniżej. Aby móc zastosować algorytm gradientu prostego należy wyznaczyć gradient $\nabla_{\mathbf{w}} L(\mathbf{w})$.

Algorithm 1: Metoda gradientu prostego

Wejście: Funkcja L, punkt startowy $\mathbf{w}_0 \in \mathbf{dom}L$, liczba epok K, krok uczenia η

Wyjście: Punkt optymalny $\mathbf{w}$ dla funkcji celu L

$\mathbf{w} \longleftarrow \mathbf{w}_0$;
for $k = 1, \ldots, K$ **do**
  $\Delta\mathbf{w} \longleftarrow -\nabla L(\mathbf{w})$;
  $\mathbf{w} \longleftarrow \mathbf{w} + \eta\Delta\mathbf{w}$;
end

UWAGA! W Pythonie funkcję celu przekazujemy do algorytmu optymalizacji jako argument, a następnie wewnątrz algorytmu bezpośrednio ją wywołujemy, aby wyliczyć jej wartość i gradient.

Algorytm stochastycznego gradientu prostego

Algorytm gradientu prostego wymaga policzenia gradientu dla całego zbioru danych, co w przypadku dużej liczności danych wiąże się z długim czasem działania. Dodatkowo, liczenie gradientu dla całego zbioru danych zazwyczaj prowadzi do dużej liczby kroków potrzebnych do zbiegnięcia metody. W celu zaradzenia tym problemom stosuje się **stochastyczny gradient prosty**, który operuje na podzbiorach danych.

Zauważmy, że funkcję celu możemy zapisać za pomocą sumy ze względu na obserwacje:

$$-\ln p(\mathcal{D}|\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} \ln p(y_n|\mathbf{x}_n, \mathbf{w}).$$

W ogólniejszej postaci możemy podzielić zbiór danych na M „paczek" danych o rozmiarze N_b (ang. *mini-batch*) $\mathcal{D} = \{\mathcal{X}_m, \mathcal{Y}_m\}_{m=1}^{M}$ i wówczas:

$$\begin{aligned} L(\mathbf{w}) &= -\frac{1}{N}\ln p(\mathcal{D}|\mathbf{w}) \\ &= -\frac{1}{N}\sum_{n=1}^{N} \ln p(y_n|\mathbf{x}_n, \mathbf{w}) \\ &= -\frac{1}{MN_b}\sum_{m=1}^{M}\sum_{n_b=1}^{N_b} \ln p(y_{n_b}|\mathbf{x}_{n_b}, \mathbf{w}) \\ &= -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N_b} \ln p(\mathcal{Y}_m|\mathcal{X}_m, \mathbf{w}) \\ &= \frac{1}{M}\sum_{m=1}^{M} L(\mathbf{w}; \mathcal{X}_m, \mathcal{Y}_m), \end{aligned}$$

gdzie w ostatnim kroku zdefiniowaliśmy wartość funkcji celu policzoną na pojedynczym mini-batchu:

$$L(\mathbf{w}; \mathcal{X}_m, \mathcal{Y}_m) = -\frac{1}{N_b} \sum_{n_b=1}^{N_b} \ln p(y_{n_b}|\mathbf{x}_{n_b}, \mathbf{w}). \tag{9}$$

Zauważmy, że dla $N_b = N$ otrzymujemy funkcję celu (8). W szczególnym przypadku można stosować mini-batche o rozmiarze $N_b = 1$, ale w praktyce wartość tę przyjmuje się większą, np. $N_b = 50$. Okazuje się, że dla $N_b > 1$ algorytm stochastycznego gradientu prostego zdecydowanie szybciej zbiega niż algorytm gradientu prostego.

Algorytm stochastycznego gradientu prostego przedstawiono poniżej. Aby móc zastosować algorytm stochastycznego gradientu prostego należy wyznaczyć gradient $\nabla_{\mathbf{w}} L(\mathbf{w}; \mathcal{X}_m, \mathcal{Y}_m)$.

Algorithm 2: Metoda stochastycznego gradientu prostego

Wejście: Funkcja L, punkt startowy $\mathbf{w}_0 \in \mathbf{dom}L$, rozmiar mini-batcha N_b, liczba epok K, krok uczenia η

Wyjście: Punkt optymalny $\mathbf{w}$ dla funkcji L

```
w ⟵ w_0;
Podziel zbiór danych 𝒟 na mini-batche {𝒳_m, 𝒴_m}_{m=1}^M;
for k = 1, ..., K do
    for m = 1, ..., M do
        Δw ⟵ −∇_w L(w; 𝒳_m, 𝒴_m);
        w ⟵ w + ηΔw;
    end
end
```

W implementacji mini-batche należy stworzyć z zachowaniem kolejności przykładów, tj. przykładowo dla zbioru złożonego z 400 przykładów i mini-batcha o wielkości 100, pierwszy mini-batch powinien zawierać pierwsze 100 przykładów, drugi kolejne 100 itd.

Regularyzacja

Zazwyczaj w procesie uczenia stosuje się zmodyfikowaną funkcję celu poprzez wprowadzenie regularyzacji. W rozważanym problemie wprowadzamy regularyzację ℓ_2 na parametry (oprócz wyrazu wolnego, czyli bez wagi dla cechy x_0, która jest stale równa 1):

$$L_\lambda(\mathbf{w}) = L(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}_{-0}\|_2^2, \tag{10}$$

gdzie $\mathbf{w}_{-0} = (w_1, \ldots, w_{D-1})^{\mathrm{T}}$ jest wektorem wag bez pierwszego parametru, tzw. wyrazu wolnego (ang. *bias*) w_0, $\lambda > 0$ oznacza współczynnik regularyzacji.

Selekcja modelu

W rozważanym problemie mamy do czynienia z dwoma wielkościami, których nie wyuczamy w oparciu o dane, tj. wartość progowa klasyfikacji θ oraz wartość współczynnika regularyzacji λ. W przypadku, gdy dysponujemy zbiorem walidacyjnym $\mathcal{D}_{val}$, możemy przeprowadzić selekcję tych wartości. W celu oceny modelu w oparciu o wybrane wielkości obu parametrów, stosować będziemy miarę **F-measure**:

$$F(\mathcal{D}_{val}; \theta, \lambda, \mathbf{w}) = \frac{2TP}{2TP + FP + FN}, \tag{11}$$

gdzie TP (ang. *true positives*) oznacza liczbę przykładów z $\mathcal{D}_{val}$ o etykiecie 1, które model o parametrach $\mathbf{w}$ zaklasyfikował jako 1, FP (ang. *false positives*) to liczba przykładów o etykiecie 0, które model zaklasyfikował jako 1, FN (ang. *false negatives*) to liczba przykładów o etykiecie 1, które model zaklasyfikował jako 0.

Wybór miary F-measure w rozważanym zadaniu nie jest przypadkowy, ponieważ pozwala ona na ocenę zbiorów, które nie są zbalansowane, tj. liczności klas w zbiorze uczącym są znacząco różne. Zauważmy, że w problemie detekcji twarzy do czynienia mamy z sytuacją, gdy zdecydowana większość obrazu nie zawiera twarzy. W konsekwencji otrzymujemy problem danych niezbalansowanych.

Procedura selekcji modelu jest następująca:[3]

Algorithm 3: Procedura selekcji modelu

Wejście: Zbiór walidacyjny $\mathcal{D}_{val}$, zbiór wartości progowych Θ, zbiór wartości współczynnika regularyzacji Λ

Wyjście: Wartości optymalnych θ i λ

```
for λ ∈ Λ do
    Znajdź rozwiązanie w dla funkcji celu L_λ(w) ;
    for θ ∈ Θ do
        Policz wartość F(D_val; θ, λ, w) ;
    end
end
Zwróć wartości λ i θ, dla których wartość F-measure była największa.
```

W implementacji do szukania wartości parametrów $\mathbf{w}$ zastosować algorytm stochastycznego gradientu prostego. Funkcję celu z regularyzacją ℓ_2 przekazywać algorytmowi poprzez odpowiednio definiowany wskaźnik na funkcję.

[3] Zwróćmy uwagę, że wybór wartości progowej klasyfikacji nie wymaga ponownego wyuczenia modelu.

Testowanie poprawności działania

Do sprawdzania poprawności działania zaproponowanych rozwiązań służy funkcja `main` w pliku `main.py`.

W pliku `main.py` nie wolno czegokolwiek zmieniać ani dopisywać.

Dodatkowo, aby program zadziałał, należy zainstalować pakiet `pillow`. W Windowsie można zrobić to w następujący sposób:

1. Uruchomić linię poleceń Start -> cmd i wpisać:
   `pip install pillow`

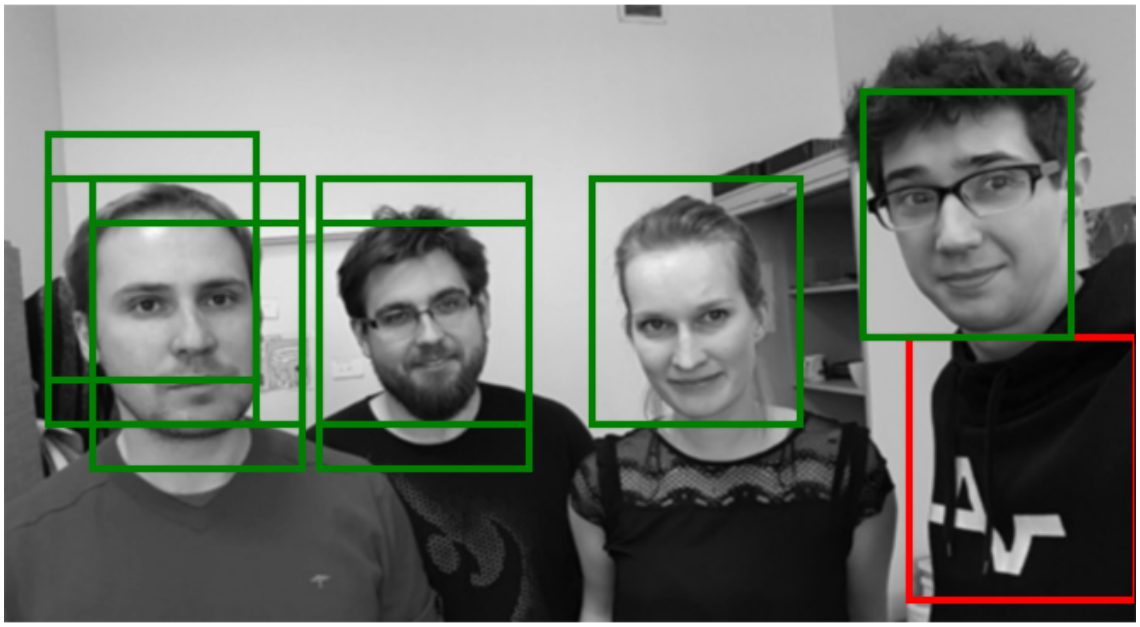
Instrukcja wykonania zadania

Należy zaimplementować wszystkie funkcje w pliku `content.py`

1. Zaimplementować funkcję `sigmoid` realizującą funkcję sigmoidalną (2).
2. Wyznaczyć gradient funkcji celu (8), następnie zaimplementować jej wartość oraz wartość gradientu w funkcji `logistic_cost_function`.
3. Zaimplementować funkcję `gradient_descent` wyznaczającą wartości parametrów **w** za pomocą algorytmu gradientu prostego. Funkcja dodatkowo ma zwracać wartości funkcji celu dla każdej iteracji algorytmu.
4. Zaimplementować funkcję `stochastic_gradient_descent` wyznaczającą wartości parametrów **w** za pomocą stochastycznego algorytmu gradientu prostego. Funkcja dodatkowo ma zwracać wartości funkcji celu wywołanej dla całego zbioru treningowego dla każdej iteracji algorytmu.
5. Wyznaczyć gradient funkcji celu (10), następnie zaimplementować jej wartość oraz wartość gradientu w funkcji `regularized_logistic_cost_function`.
6. Zaimplementować funkcję `prediction` dokonującej predykcji (3) na podstawie nauczonego modelu w pliku.
7. Zaimplementować wyliczenie miary F-measure (11) w funkcji `f_measure`.
8. Zaimplementować procedurę selekcji modelu w funkcji `model_selection`. Dodatkowo funkcja ma zwrócić parametry dla najlepszego modelu oraz macierz z wartościami miary F-measure dla każdej pary hiperparametrów (λ, θ).

Efekt końcowy działania programu powinien być taki, jak na Rysunku 1.

UWAGA! Wszelkie nazwy funkcji i zmiennych w pliku `content.py` muszą pozostać zachowane.



Rysunek 1: Wynik poprawnie działającego zadania.

Pytania kontrolne

1. Wyznaczyć pochodną sigmoidalnej funkcji logistycznej. Zapisać ją jedynie przy pomocy wartości funkcji sigmoidalnej $\sigma(a)$.

2. Wyznaczyć gradient funkcji celu (8) lub gradient funkcji celu z regularyzacją (10).

3. Co to jest model regresji logistycznej? W jaki sposób modeluje warunkowe prawdopodobieństwo?

4. Za co odpowiada wartość progowa θ? W jaki sposób systematycznie podejść do ustalenia jej wartości?

5. Co to jest miara F-measure? Do czego jest wykorzystywana w powyższym zadaniu? Dlaczego zamiast niej nie stosuje się tutaj zwykłej poprawności klasyfikacji na ciągu walidacyjnym?

6. Za co odpowiada η w algorytmie gradientu prostego i stochastycznego gradientu prostego? Jak algorytmy będą zachowywać się dla różnych wielkości tego parametru?

7. Na czym polega detekcja obiektu na zdjęciu? Dlaczego jest to problem klasyfikacji?

8. Dlaczego algorytm stochastycznego gradientu prostego zbiega znacznie szybciej? Jakie jest znaczenie wielkości mini-batcha dla zbieżności algorytmu? Jak będzie zachowywał się dla małych mini-batchy, a jak dla dużych?

9. W jaki sposób można dodać regularyzację ℓ_2 na parametry modelu regresji logistycznej? Jaki efekt wówczas osiągniemy? Kiedy konieczne jest stosowanie regularyzacji, a kiedy nie?

10. Na czym polega procedura selekcji modelu w tym zadaniu? Jakie hiperparametry wyznaczamy? Które z nich wymagają każdorazowego nauczenia modelu, a które nie i dlaczego?