

Podstawy Uczenia Maszynowego - laboratorium

Piotr Łuczak M.Sc.Eng.
piotr.luczak.1@p.lodz.pl

Lato 2022

1 Terminy zajęć laboratoryjnych

Zajęcia laboratoryjne będą odbywać się w tygodniach 1 - 15 semestru letniego w czwartki w godzinach 8:15 - 9:45 i 10:15 - 11:45 w sali E109 Instytutu Informatyki Stosowanej, budynek A12.

2 Terminy dostarczania rozwiązań zadań i wymogi ogólne

1. Tydzień 3 (2022-03-17) → Zadanie 1 - Regresja liniowa
2. Tydzień 4 (2022-03-24) → Zadanie 2 - Klasyfikacja liniowa
3. Tydzień 6 (2022-04-07) → Zadanie 3 - Regresja logistyczna
4. Tydzień 8 (2022-04-21) → Zadanie 4 - Klasyfikator SVM
5. Tydzień 10 (2022-05-05) → Zadanie 5 - Drzewo decyzyjne
6. Tydzień 11 (2022-05-12) → Zadanie 6 - Las losowy
7. Tydzień 13 (2022-05-26) → Zadanie 7 - Neuron
8. Tydzień 15 (2022-06-09) → Zadanie 8 - Sieć neuronowa

Aby uzyskać zaliczenie konieczne jest uzyskanie pozytywnej oceny z **każdego** z ćwiczeń; ocena końcowa będzie średnią z wszystkich ocen częściowych. Kryteria oceniania są kumulatywne: wymogiem koniecznym do uzyskania oceny 4 jest spełnienie podanych dla niej kryteriów oraz spełnienie kryteriów oceny 3.

Dostarczenie rozwiązania po zadanym terminie skutkuje obniżeniem oceny z ćwiczenia o 0.5. **Nieprzekraczalnym** terminem dostarczania rozwiązań jest początek sesji, t.j. **2022-06-22**, po tym terminie rozwiązania nie będą przyjmowane.

Każdy z zaimplementowanych algorytmów powinien być opatrzony ewaluacją czasu działania (w oparciu o przynajmniej 100 wykonań) oraz skuteczności. Skuteczność każdego klasyfikatora powinna być oceniona w oparciu o macierz pomyłek, czułość (sensitivity), swoistość (specificity) oraz krzywą ROC. W każdym z zadań wykorzystywane zbiory danych powinny być poprawnie podzielone na podzbiór uczący i testowy.

Wszystkie generatory z `sklearn.datasets` powinny być uruchamiane z parametrem `random_state` równym numerowi indeksu i `n_samples` równym concatenacji pierwszych dwóch i ostatnich dwóch numerów indeksu. Wszystkie porównania algorytmów działających na syntetycznych dwuwymiarowych zbiorach powinny wykorzystywać **binarne etykiety** klas i zawierać wizualizację powierzchni decyzyjnej wykonaną przy użyciu `matplotlib.pyplot.contourf`. Zbiory jednomodowe powinny być wygenerowane wykorzystując funkcje `scikit-learn` `sklearn.datasets.make_classification` oraz `sklearn.datasets.make_moons`. W przypadku zbioru wielomodowego należy wykorzystać tylko `sklearn.datasets.make_classification`.

3 Zadania

3.1 Regresja liniowa

Zaimplementuj **analityczną**(1) i **numeryczną**(2) (wykorzystując `minimize(method='Powell')`) wersje regresji liniowej. Porównaj działanie obu wersji na syntetycznym jednowymiarowym zbiorze wygenerowanym przy użyciu `sklearn.datasets.make_regression` z `noise` równym 16.

$$\vec{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \vec{y} \quad (1)$$

$$L(\vec{w}) = \frac{1}{N} (\vec{y} - \mathbf{X}^T \vec{w})^T (\vec{y} - \mathbf{X}^T \vec{w}) \quad (2)$$

Wygeneruj trzy zbiory danych przy użyciu gry FlapPy bird:

1. Minimalny zbiór punktów zawierający tylko jeden punkt dla każdej ominiętej przeszkody.
2. Zbiór składający się z kluczowych momentów przelotu, niezbędnych do bezpiecznego pokonania trasy.
3. Kompletny zbiór składający się ze wszystkich punktów na trasie przelotu.

Parametr `'seed'` powinien być równy numerowi indeksu. Aby zapisać aktualną pozycję postaci do pliku należy nacisnąć klawisz `'spacja'`. Wykorzystując `sklearn.linear_model.LinearRegression` wykonaj regresję dla każdego ze zbiorów z wielomianem 9 i 21 rzędu (z wykorzystaniem funkcji `sklearn.preprocessing.PolynomialFeatures`) i porównaj z wynikiem własnej implementacji **analitycznej**. Dla każdego modelu oblicz błąd średniokwadratowy i narysuj dane wyjściowe oraz wynik regresji. Który model najlepiej poradził sobie z dopasowaniem do trasy?

Zaimplementuj **analityczną** wersję regresji z regularyzacją Tichonowa/L2(3) (ridge regression). Porównaj działanie własnej implementacji z `sklearn.linear_model.Ridge` na zebranych zbiorach danych z FlapPy Bird dla wielomianu 16 rzędu.

$$\vec{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \vec{y} \quad (3)$$

3.1.1 Kryteria oceniania

Porównanie analitycznej i numerycznej implementacji regresji na syntetycznym zbiorze $\rightarrow 3$

Porównanie `sklearn` i analitycznej implementacji regresji na zbiorze FlapPy Bird $\rightarrow 4$

Porównanie `sklearn` i własnej implementacji ridge regression na zbiorze FlapPy Bird $\rightarrow 5$

3.2 Klasyfikacja liniowa

Zaimplementuj **analityczną** wersję klasyfikacji liniowej z regularyzacją Tichonowa(3) i wytrenuj ją na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę. Porównaj na obu zbiorach działanie własnej implementacji z `sklearn.linear_model.RidgeClassifier`.

Wykorzystując zbiór danych o chorobach serca z Cleveland dokonaj eksploracyjnej analizy danych w oparciu o 14 głównych cech. Policz ilość brakujących wartości a następnie usuń wiersze je zawierające. Zastąp zmienną przewidywaną wartością binarną opisującą występowanie choroby serca. Dla wszystkich cech policz podstawowe miary statystyczne (średnia/dominanta, odchylenie standardowe, minimum, maksimum) uwzględniając typ cechy (kategoryczna/dyskretna/ciągła). Przedstaw wartości cech w formie histogramów pokolorowanych zależnie od wartości przewidywanej. Narysuj macierz korelacji pomiędzy wartościami. Które 4 cechy pozwolą na najskuteczniejszą klasyfikację?

Porównaj skuteczność własnej implementacji i `RidgeClassifier` na zbiorze danych o chorobach serca wykorzystując **wszystkie** cechy. Następnie w oparciu o `RidgeClassifier` z `sklearn` i `sklearn.feature_selection.SequentialFeatureSelector` wybierz 4 najlepsze cechy. Porównaj automatyczny wybór cech do ręcznego, czy wybrane zostały inne cechy? Porównaj skuteczność własnej implementacji i `RidgeClassifier` na obu wybranych zestawach cech.

3.2.1 Kryteria oceniania

Porównanie `sklearn` i własnej implementacji klasyfikatora na syntetycznych zbiorach → 3

Eksploracja danych i uzasadniona odpowiedź na pytanie → 4

Porównanie `sklearn` i własnej implementacji na zbiorze o chorobach serca i różnych zestawach cech → 5

3.3 Regresja logistyczna

Zaimplementuj uczenie regresji logistycznej metodą **gradient descent**(4). Uczenie powinno obsługiwać zarówno zatrzymanie po osiągnięciu zadanego błędu jak i wykonaniu zadanej ilości iteracji. Wytrenuj klasyfikator na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę i porównaj jego działanie z `sklearn.linear_model.LogisticRegression`.

$$\Delta \vec{w} = -\eta(t_j - y_j)f(\vec{x}_j^T \vec{w})[1 - f(\vec{x}_j^T \vec{w})]\vec{x}_j \quad \text{gdzie } f(s) = \frac{1}{1 + e^{-\beta s}} \quad (4)$$

Wykorzystując zbiór Rain in Australia (© Commonwealth of Australia 2010, Bureau of Meteorology) dokonaj preprocessingu danych. Usuń kolumny mające więcej niż 30% brakujących wartości (oraz kolumnę 'Risk-MM' jeżeli istnieje) i odseparuj kolumnę z wartością przewidywaną ('RainTomorrow'). Dokonaj imputacji brakujących wartości zakładając że są one MCAR (Missing Completely At Random) - zmienne kategoryczne należy zastąpić dominantą a dane numeryczne medianą. Dokonaj winsoryzacji danych odstających ponad 1.5 IQR. Podziel dane dla każdego z regionów na zbiory testowe i treningowe ze stratyfikacją. Znormalizuj dane numeryczne i zakoduj (one-hot) dane kategoryczne. Należy zwrócić szczególną uwagę na cykliczny charakter komponentów w dacie - data jest dyskretną numeryczną wartością złożoną. Kodowanie kolumny 'Location' nie jest konieczne. Należy także zwrócić uwagę na potencjalny wyciek danych. Naucz osobny model `LogisticRegression` dla każdego z regionów. Który model ma najwyższą skutecz-

ność? Porównaj skuteczność najlepszego modelu ze skutecznością własnej implementacji uczonej na tym samym zbiorze.

Sprawdź skuteczność modeli regionalnych na krajowym zbiorze testowym zbudowanym ze zbiorów testowych wszystkich regionów. Który model osiągnął najwyższą skuteczność? Czy był to model o najwyższej skuteczności lokalnej? Porównaj skuteczność najlepszego modelu krajowego z `sklearn.dummy.DummyClassifier`. Co o prawdziwej skuteczności modelu mówi to porównanie? Czy budowanie modelu w oparciu o fragmentaryczne, lokalne dane to dobry sposób zmniejszenia wymaganej ilości danych uczących?

3.3.1 Kryteria oceniania

Porównanie `sklearn` i własnej implementacji klasyfikatora na syntetycznych zbiorach → 3

Poprawny preprocessing danych, nauczanie modeli dla regionów i porównanie ich skuteczności → 4

Weryfikacja skuteczności modeli regionalnych na zbiorze krajowym i porównanie skuteczności najlepszego modelu → 5

3.4 Klasyfikator SVM

Zaimplementuj uczenie liniowej maszyny wektorów nośnych w oparciu o metody optymalizacji numerycznej dostępne w `scipy` oraz funkcję kosztu:

$$L(\vec{w}, b) = \frac{1}{2} \vec{w}^T \vec{w} - \sum_i \lambda_i [y_i (\vec{w}^T \vec{x}_i + b) - 1] = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \vec{x}_i^T \vec{x}_j \quad (5)$$

Wytrenuj klasyfikator na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę i porównaj jego działanie z `sklearn.svm.SVC` z `kernel='linear'`.

Zaimplementuj uczenie maszyny wektorów nośnych wykorzystującej radialną i wielomianową funkcję jądra(6) i porównaj skuteczność i kształt granicy decyzyjnej z implementacją z `sklearn` z `kernel='rbf'` i `kernel='poly'` na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę.

$$L = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j K(\vec{x}_i, \vec{x}_j) \quad (6)$$

Wykorzystując zbiór Stellar Classification Dataset - SDSS17 (© SDSS) i algorytm selekcji modelu `sklearn.model_selection.GridSearchCV` dobierz optymalny zestaw hiperparametrów (`C` i `kernel`) i naucz model SVM (z `sklearn`) do klasyfikacji obiektów jako gwiazdy lub galaktyki. Przed doбором hiperparametrów uzupełnij brakujące wartości (jeżeli istnieją) i wyeliminuj nieistotne kolumny ('cam_col', 'MJD' oraz te kończące się na '_ID'), narysuj macierz korelacji pomiędzy wartościami i dokonaj normalizacji wartości. Do jakich wartości macierzy pomyłek należy dążyć jeżeli chcemy wykorzystać klasyfikator do automatycznej selekcji obiektów do obserwacji teleskopem podczas poszukiwania nowych galaktyk?

3.4.1 Kryteria oceniania

Implementacja liniowego SVMa → 3

Implementacja SVMa wykorzystującego funkcję jądra → 4

Analiza zbioru SDSS17 w oparciu o SVM → 5

3.5 Drzewo decyzyjne

Zaimplementuj uczenie drzewa decyzyjnego na bazie redukcji entropii(7), zakładając że prawdopodobieństwo wystąpienia klasy jest wprost proporcjonalne do proporcji występowania jej próbek w zbiorze danych.

$$H = - \sum_i p_i \log(p_i) \quad (7)$$

Wytrenuj klasyfikator na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę oraz zbiorze wielomodowym i porównaj jego działanie z `sklearn.tree.DecisionTreeClassifier` dla różnych maksymalnych głębokości drzewa (nieograniczonej, równej ilości cech i równej ilości klastrów).

Wykorzystując zbiór HTRU2 i algorytm `sklearn.model_selection.GridSearchCV` dobierz optymalny zestaw hiperparametrów struktury drzewa (`criterion`, `max_depth`, `min_samples_split` oraz `min_samples_leaf`) i naucz drzewo decyzyjne (z `sklearn`) detekcji pulsarów. Optymalizacja hiperparametrów musi być poprzedzona poprawnym preprocessingiem zbioru danych.

Wykorzystując zbiór HTRU2 zbadaj wpływ głębokości drzewa na szybkość inferencji oraz skuteczność modelu z `sklearn` i własnej implementacji. Pomiar skuteczności i szybkości powinien być powtórzony przynajmniej 10 razy dla każdej głębokości. Czy wybierając cel dla pierwszej sondy mającej za zadanie zbadać pulsar z bliska uzasadnione jest poszukiwanie optymalnej głębokości drzewa?

3.5.1 Kryteria oceniania

Implementacja drzewa decyzyjnego → 3

Analiza zbioru HTRU2 w oparciu o drzewo decyzyjne → 4

Analiza wpływu głębokości drzewa na jego skuteczność → 5

3.6 Las losowy

Zaimplementuj uczenie lasu losowego na bazie przyrostu GINI(8) zakładając że prawdopodobieństwo wystąpienia klasy jest wprost proporcjonalne do proporcji występowania jej próbek w zbiorze danych.

$$G = 1 - \sum_i p_i^2 \quad (8)$$

Wytrenuj klasyfikator na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę oraz zbiorze wielomodowym i porównaj jego działanie z gotowym algorytmem `sklearn.ensemble.RandomForestClassifier`.

Wykorzystując zbiór NASA JPL Asteroid i algorytm `sklearn.model_selection.GridSearchCV` dobierz optymalny zestaw hiperparametrów struktury lasu (`n_estimators`, `criterion`, `max_depth`, `min_samples_split` oraz `min_samples_leaf`) i naucz las losowy (z `sklearn`) detekcji obiektów bliskich ziemi (NEO). Powtórz proces dla detekcji potencjalnie niebezpiecznych obiektów (PHA), czy wybrane zostały takie same hiperparametry? Optymalizacja hiperparametrów musi być poprzedzona poprawnym preprocessingiem zbioru danych.

Wykorzystując zbiór NASA JPL Asteroid zbadaj wpływ głębokości lasu na szybkość inferencji oraz skuteczność modelu z `sklearn` i własnej implementacji w problemie detekcji obiektów bliskich

ziemi (NEO). Pomiar skuteczności i szybkości powinien być powtórzony przynajmniej 5 razy dla każdej głębokości.

3.6.1 Kryteria oceniania

Implementacja lasu losowego $\rightarrow 3$

Analiza zbioru NASA JPL Asteroid w oparciu o las losowy $\rightarrow 4$

Analiza wpływu głębokości lasu na jego skuteczność $\rightarrow 5$

3.7 Neuron

Zaimplementuj sztuczny model neuronu i wytrenuj go na syntetycznych zbiorach dwuwymiarowych danych z jednym klastrem na klasę. Neuron powinien być trenowany z zastosowaniem reguły delta:

$$\Delta \vec{w}_j = \eta \epsilon f'(s) \vec{x}_j = \eta (t_j - y_j) f'(\vec{w}^T \vec{x}_j) \vec{x}_j. \quad (9)$$

Zaimplementuj zmienną szybkość uczenia w oparciu o symulowane wyżarzanie cosinusowe(10) oraz uczenie neuronu w oparciu o batche. Korekta wagi powinna być oparta o średnią korekt dla wszystkich próbek w batchu. Porównaj szybkość uczenia neuronu przy użyciu batchy i pojedynczych próbek.

$$\eta(n) = \eta_{min} + (\eta_{max} - \eta_{min})(1 + \cos(\frac{n}{n_{max}}\pi)) \quad (10)$$

3.7.1 Kryteria oceniania

Uczenie i ewaluacja pojedynczego neuronu z logistyczną funkcją aktywacji $\rightarrow 3$

Uczenie i ewaluacja z aktywacjami: Heaviside, sin, tanh, sign, ReLu, leaky ReLu $\rightarrow 4$

Uczenie z wykorzystaniem zmiennej szybkości uczenia i batchy $\rightarrow 5$

3.8 Sieć neuronowa

Zaimplementuj płytką (do 5 warstw) wielowarstwową sieć neuronową i wytrenuj ją na syntetycznym zbiorze dwuwymiarowych danych z co najmniej trzema klastrami na klasę.

Dostosuj strukturę sieci i wytrenuj ją na zbiorze MNIST. Obrazy powinny być podawane jako 'spłaszczone' wektory pikseli.

3.8.1 Kryteria oceniania

Uczenie i ewaluacja trójwarstwowej sieci neuronowej z logistyczną funkcją aktywacji $\rightarrow 3$

Modularna struktura sieci umożliwiająca ustalenie dowolnej ilości i szerokości warstw $\rightarrow 4$

Wytrenowanie sieci na zbiorze MNIST $\rightarrow 5$

4 Wymagania implementacyjne

- Rozwiązania zadań muszą być zaimplementowane w języku Python (wersja 3.6 lub nowsza) z wykorzystaniem notebooków platformy Jupyter.
- Każde rozwiązanie musi być dostarczone zarówno w oryginalnym formacie **.ipynb** jak i w wersji wyeksportowanej do **.html**.
- Obliczenia powinny być zaimplementowane z wykorzystaniem biblioteki NumPy i poprawnie wykorzystywać mechanizmy wektoryzacji i broadcastingu.