# Podstawy Uczenia Maszynowego - laboratorium

Piotr Łuczak PhD Eng.
piotr.luczak.1@p.lodz.pl

Lato 2024

## 1 Terminy zajęć laboratoryjnych

Zajęcia laboratoryjne będą odbywać się w tygodniach 3 - 15 semestru letniego w czwartki w godzinach 8:15 - 10:00 (SIIUM3), 10:15 - 12:00 (SIIUM1) oraz 12:15 - 14:00 (SIIUM2) w sali E107 Instytutu Informatyki Stosowanej, budynek A12.

## 2 Terminy dostarczania rozwiązań zadań i wymogi ogólne

1. Tydzień 6 (2024-04-04) → Zadanie 1 - Regresja liniowa
2. Tydzień 7 (2024-04-11) → Zadanie 2 - Klasyfikacja liniowa
3. Tydzień ~~8~~ 9 ~~(2024-04-18)~~ (2024-04-25) → Zadanie 3 - Regresja logistyczna
4. Tydzień 10 (2024-05-09) → Zadanie 4 - Klasyfikator SVM
5. Tydzień 11 (2024-05-16) → Zadanie 5 - Drzewo decyzyjne
6. Tydzień 12 (2024-05-23) → Zadanie 6 - Las losowy
7. Tydzień 13 (2024-06-06) → Zadanie 7 - Neuron
8. Tydzień 15 (2024-06-20) → Zadanie 8 - Sieć neuronowa

Aby uzyskać zaliczenie konieczne jest uzyskanie pozytywnej oceny z **każdego** z ćwiczeń; ocena końcowa będzie średnią z wszystkich ocen cząstkowych. **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. **2024-06-24**, 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 konkatenacji 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. **Syntetyczne zbiory jednomodowe** (t.j. z jednym klastrem na klasę) powinny być wygenerowane wykorzystując funkcje scikit-learn sklearn.datasets.make_classification **oraz** sklearn.datasets.make_moons. W przypadku **syntetycznych zbiorów wielomodowych** (t.j. z wieloma klastrami na klasę) należy wykorzystać tylko sklearn.datasets.make_classification, lub, w niektórych wersjach biblioteki sklearn, sklearn.datasets.make_blobs.

# 3 Wymagania implementacyjne

- Każde rozwiązanie musi być dostarczone zarówno w oryginalnym formacie **.ipynb** jak i w wersji wyeksportowanej do **.html** wraz z wykresami.
- Rozwiązania zadań muszą być zaimplementowane w języku **Python** (wersja **3.10 lub nowsza**) z wykorzystaniem notebooków platformy **Jupyter**. Warto rozważyć wykorzystanie **darmowej** platformy Google Collab zamiast lokalnej instalacji.
- Obliczenia powinny być zaimplementowane z wykorzystaniem biblioteki **NumPy** i poprawnie wykorzystywać mechanizmy **wektoryzacji** i **broadcastingu**.
- Wyniki porównywania modeli powinny być opatrzone stosownym opisem w notebookach w komórkach typu markdown.
- Odpowiedzi na pytania zadane w poleceniach powinny być zawarte w notebookach w komórkach typu markdown.
- Wizualizacje działania klasyfikatorów powinny być zaimplementowane z użyciem funkcji matplotlib.pyplot.contourf oraz numpy.meshgrid
- Nazwy plików powinny być zgodne z formatem **{nr. zadania}_{nr. indeksu}_{imie}_{nazwisko}.{html|ipynb|zip}**

# 4 Zadania

## 4.1 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 uzyciu sklearn.datasets.make_regression z `noise` równym 16.

$$\vec{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\vec{y} \tag{1}$$

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

- $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ próbki w zbiorze uczącym
- $\vec{y} \in \mathbb{R}^N$ etykiety zbioru uczącego
- $\vec{w} \in \mathbb{R}^D$ wagi modelu
- $N$ ilość próbek w zbiorze / batchu

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.

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

- $\boldsymbol{I} = diag(1, 1, \ldots, 1)$ macierz jednostkowa
- $\lambda \in \mathbb{R}_{>0}$ współczynnik regularyzacji

### 4.1.1 Kryteria oceniania

Porównanie analitycznej i numerycznej implementacji regresji na syntetycznym zbiorze → 3
Porównanie sklearn i analitycznej implementacji regresji na zbiorze FlapPy Bird → 4
Porównanie sklearn i własnej implementacji ridge regression na zbiorze FlapPy Bird → 5

## 4.2 Klasyfikacja liniowa

Zaimplementuj **analityczną** wersję klasyfikacji liniowej z regularyzacją Tichonowa(3) i wytrenuj ją na **syntetycznych jednomodowych zbiorach** danych. 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 klasyfikacje?

Porównaj skuteczność własnej implementacji i RidgeClassifier na zbiorze danych o chorobach serca wykorzystując **14 głównych** cech. 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.

### 4.2.1 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

### 4.3 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 jednomodowych zbiorach** danych i porównaj jego działanie z sklearn.linear_model.LogisticRegression.

$$\Delta\vec{w_j} = -\eta(t_j - y_j)f(\vec{x_j}^T w_{\vec{j-1}})[1 - f(\vec{x_j}^T w_{\vec{j-1}})]\vec{x_j} \qquad \text{gdzie } f(s) = \frac{1}{1+e^{-\beta s}} \qquad (4)$$

- $\Delta\vec{w_j} \in \mathbb{R}^D$ $j$-ta korekta wektora wag
- $\eta \in \mathbb{R}_{>0}$ krok algorytmu uczenia (learning rate)
- $t_j \in \mathbb{R}$ etykieta $j$-tej próbki ze zbioru uczącego
- $y_j \in \mathbb{R}$ predykcja modelu dla $j$-tej próbki ze zbioru uczącego
- $\vec{x_j} \in \mathbb{R}^D$ $j$-ta próbka ze zbioru uczącego
- $w_{\vec{j-1}} \in \mathbb{R}^D$ wagi modelu po $(j-1)$-tej (poprzedniej) korekcie

Wykorzystując zbiór Rain in Australia (© Commonwealth of Australia 2010, Bureau of Meteorology) dokonaj preprocessingu danych. Usuń kolumny mające więcej in 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 daty - 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 skuteczność? 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?

#### 4.3.1 Kryteria oceniania

Porównanie sklearn i własnej implementacji klasyfikatora na syntetycznych zbiorach → 3
Poprawny preprocessing danych, nauczenie modeli dla regionów i porównanie ich skuteczności → 4
Weryfikacja skuteczności modeli regionalnych na zbiorze krajowym i porównanie skuteczności najlepszego modelu → 5

### 4.4 Klasyfikator SVM

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

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

- $\lambda_n \in \mathbb{R}_{\geq 0}$ waga $n$-tego wektora nośnego
- $\vec{x_n} \in \mathbb{R}^D$ $n$-ta próbka ze zbioru uczącego
- $y_n \in \mathbb{R}$ $n$-ta etykieta ze zbioru uczącego

Wytrenuj klasyfikator na **syntetycznych jednomodowych zbiorach** danych 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.

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

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 doborem 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?

#### 4.4.1 Kryteria oceniania

Implementacja liniowego SVMa → 3
Implementacja SVMa wykorzystującego funkcję jądra → 4
Analiza zbioru SDSS17 w oparciu o SVM → 5

## 4.5 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.

$$H = -\sum_i p_i log(p_i) \tag{7}$$

- $H \in \mathbb{R}_{>0}$ entropia
- $p_i \in [0, 1]$ prawdopodobieństwo $i$-tej klasy

Wytrenuj klasyfikator na **syntetycznych zbiory jednomodowych** oraz **syntetycznym zbiorze wielomodowym** i porównaj jego działanie z sklearn.tree.DecisionTreeClassifier dla różnych maksymalnych głębokości drzewa (nieograniczonej, równej ilości cech i równej całkowitej 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 uzasadniona jest optymalizaccja czasu uczenia i ewaluacji poprzez poszukiwanie optymalnej głębokości drzewa?

### 4.5.1 Kryteria oceniania

Implementacja drzewa decyzyjnego → 3
Analiza zbioru HTRU2 w oparciu o drzewo decyzyjne → 4
Analiza wpływu głębokości drzewa na jego skuteczność → 5

### 4.6 Las losowy

Zaimplementuj uczenie lasu losowego na bazie redukcji Gini(8) zakładając że prawdopodobieństwo wystąpienia klasy jest wprost proporcjonalne do proporcji występowania jej próbek w zbiorze danych. Przy implementacji należy pamiętać o procedurze **"bagging'u"** i **losowym wyborze cech na węzłach** - las losowy **nie jest** komitetem drzew.

$$G = 1 - \sum_i p_i^2 \tag{8}$$

- $G \in [0, 1]$ współczynnik różnorodności Gini'ego
- $p_i \in [0, 1]$ prawdopodobieństwo $i$-tej klasy

Wytrenuj klasyfikator na **syntetycznych zbiory jednomodowych** oraz **syntetycznym zbiorze wielomodowym** i porównaj jego działanie z gotowym algorytmem sklearn.ensemble.Random ForestClassifier.

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). Pamiętaj o usunięciu pół `Orbit_id` i `full_name`. 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 wielkości 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 wielkości.

#### 4.6.1 Kryteria oceniania

Implementacja lasu losowego → 3
Analiza zbioru NASA JPL Asteroid w oparciu o las losowy → 4
Analiza wpływu wielkości lasu na jego skuteczność → 5

## 4.7 Neuron

Zaimplementuj sztuczny model neuronu i wytrenuj go na **syntetycznych zbiorach jednomodowych**. Neuron powinien być trenowany z zastosowaniem reguły delta:

$$\Delta\vec{w_j} = \eta\epsilon_j f'(s_j)\vec{x_j} = \eta(t_j - y_j)f'(\vec{w_{j-1}}^T\vec{x_j})\vec{x_j}. \tag{9}$$

- $\Delta\vec{w_j} \in \mathbb{R}^D$ $j$-ta korekta wektora wag
- $\eta \in \mathbb{R}_{>0}$ krok algorytmu uczenia (learning rate)
- $\epsilon_j \in \mathbb{R}$ błąd klasyfikacji $j$-tej próbki ze zbioru uczącego
- $s_j \in \mathbb{R}$ stan neuronu dla $j$-tej próbki ze zbioru uczącego
- $f'() \in \mathbb{R}$ pochodna funkcji aktywacji
- $t_j \in \mathbb{R}$ etykieta $j$-tej próbki ze zbioru uczącego
- $y_j \in \mathbb{R}$ predykcja modelu dla $j$-tej próbki ze zbioru uczącego
- $\vec{x_j} \in \mathbb{R}^D$ $j$-ta próbka ze zbioru uczącego
- $\vec{w_{j-1}} \in \mathbb{R}^D$ wagi modelu po $(j-1)$-tej (poprzedniej) korekcie

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.

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

- $\eta(n)$ krok algorytmu (learning rate) w $n$-tej epoce
- $\eta(min)$ minimalny dopuszczalny krok algorytmu (learning rate)
- $\eta(max)$ maksymalny dopuszczalny krok algorytmu (learning rate)
- $n_{max}$ liczba epok uczenia

### 4.7.1 Kryteria oceniania

Uczenie i ewaluacja pojedynczego neuronu z logistyczną funkcją aktywacji → 3
Uczenie i ewaluacja z aktywacjami: Heaviside, sin, tanh, sign, ReLu, leaky ReLu → 4
Uczenie z wykorzystaniem zmiennej szybkości uczenia i batchy → 5

## 4.8 Sieć neuronowa

Zaimplementuj płytką (do 5 warstw) wielowarstwową siec neuronową z **dwoma** neuronami w warstwie wyjściowej i wytrenuj ją na **syntetycznym zbiorze wielomodowym** z co najmniej trzema klastrami na klasę.

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

### 4.8.1 Kryteria oceniania

Uczenie i ewaluacja trójwarstwowej sieci neuronowej z ReLu i logistyczną funkcją aktywacji → 3
Modularna struktura sieci umożliwiająca ustalenie dowolnej ilości i szerokości warstw → 4
Wytrenowanie sieci na zbiorze MNIST oraz podawanie danych w batch'ach'→ 5