Metody Inżynierii Wiedzy

Uczenie nienadzorowane i częściowo nadzorowane

Dr inż. Michał Majewski

mmajew@pjwstk.edu.pl

materiały: *ftp*(*public*) : *//mmajew/MIW*

Uczenie nadzorowane

Uczenie nadzorowane to metoda uczenia maszynowego, w której model jest trenowany na **danych wejściowych, które mają przypisane etykiety (tj. dane wyjściowe)**.

Celem jest nauczenie modelu **przewidywania etykiet dla nowych**, niewidzianych wcześniej **danych**.

Proces treningu polega na minimalizowaniu różnicy między przewidywaniami modelu a rzeczywistymi etykietami poprzez dostosowywanie parametrów modelu.

Typowe zadania w uczeniu nadzorowanym to klasyfikacja (np. rozpoznawanie cyfr) i regresja (np. przewidywanie cen domów).

Przykłady algorytmów:

1. Drzewa decyzyjne
2. SVM
3. Sieci neuronowe

Dane wejściowe

pies kot pies

etykiety

Uczenie nienadzorowane

Uczenie nienadzorowane to metoda uczenia maszynowego, w której model jest **trenowany na danych wejściowych bez przypisanych etykiet**.

Celem jest **odkrywanie ukrytych struktur**, **wzorców lub zależności w danych**. W przeciwieństwie do uczenia nadzorowanego, model nie otrzymuje informacji o prawidłowych wynikach.

Typowe zadania w uczeniu nienadzorowanym to klastrowanie (np. grupowanie podobnych obiektów) i redukcja wymiarów (np. PCA w celu uproszczenia danych).

Przykłady algorytmów:

1. K-means
2. DBSCAN
3. PCA (Principal Component Analysis)

Dane wejściowe

? ? ?

Brak etykiet

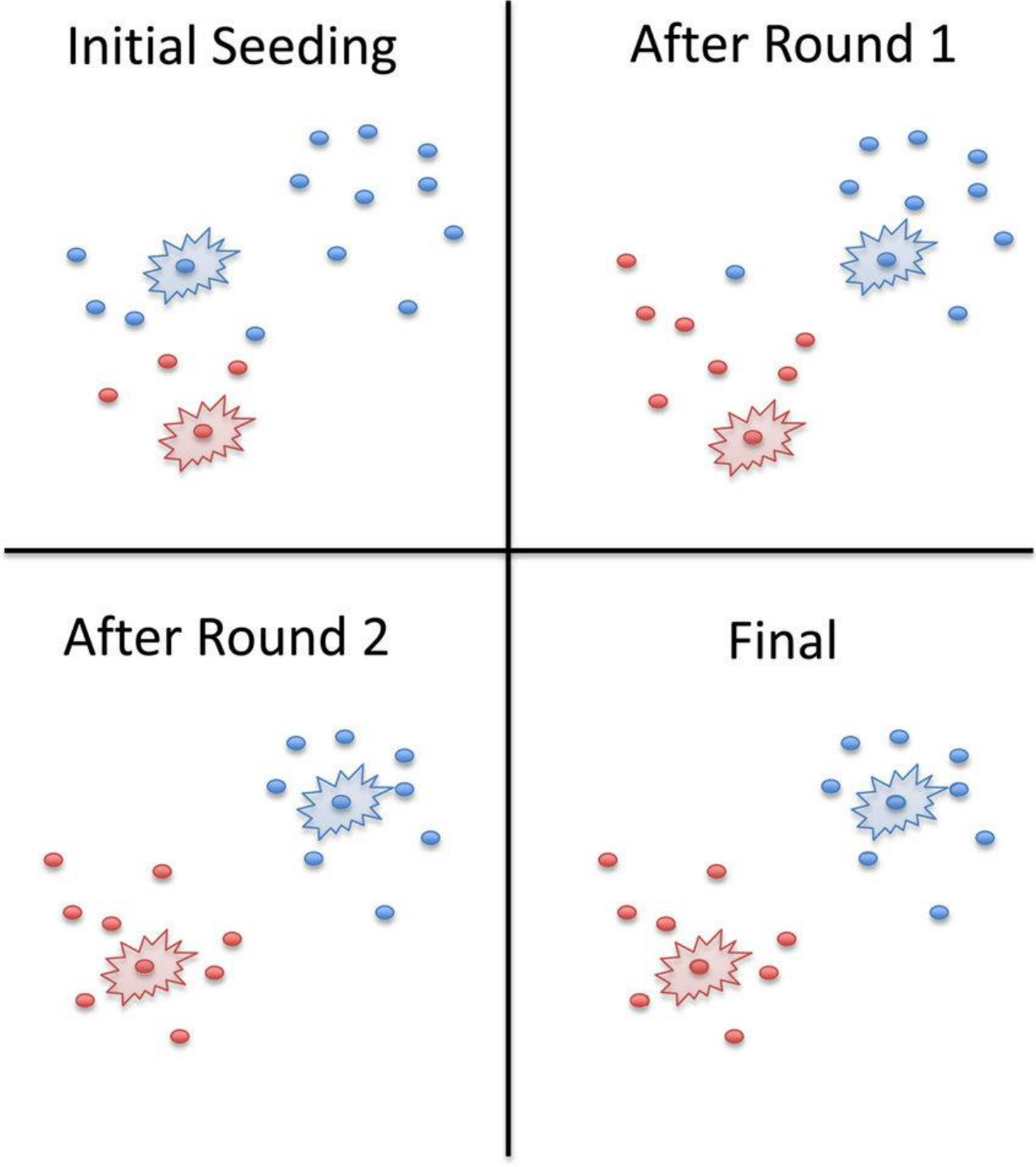
Algorytmy uczenia nienadzorowanego

K-means to algorytm klastrowania, który **dzieli dane na *K* klastrów**.

Celem jest **zminimalizowanie sumy kwadratów odległości między punktami danych a centroidami klastrów**.

Proces iteracyjny, który obejmuje przypisanie punktów do najbliższego centroidu oraz aktualizację pozycji centroidów, powtarza się, aż konwergencja zostanie osiągnięta.

K-means jest szybki i łatwy do implementacji, ale **wymaga podania liczby klastrów z góry**.

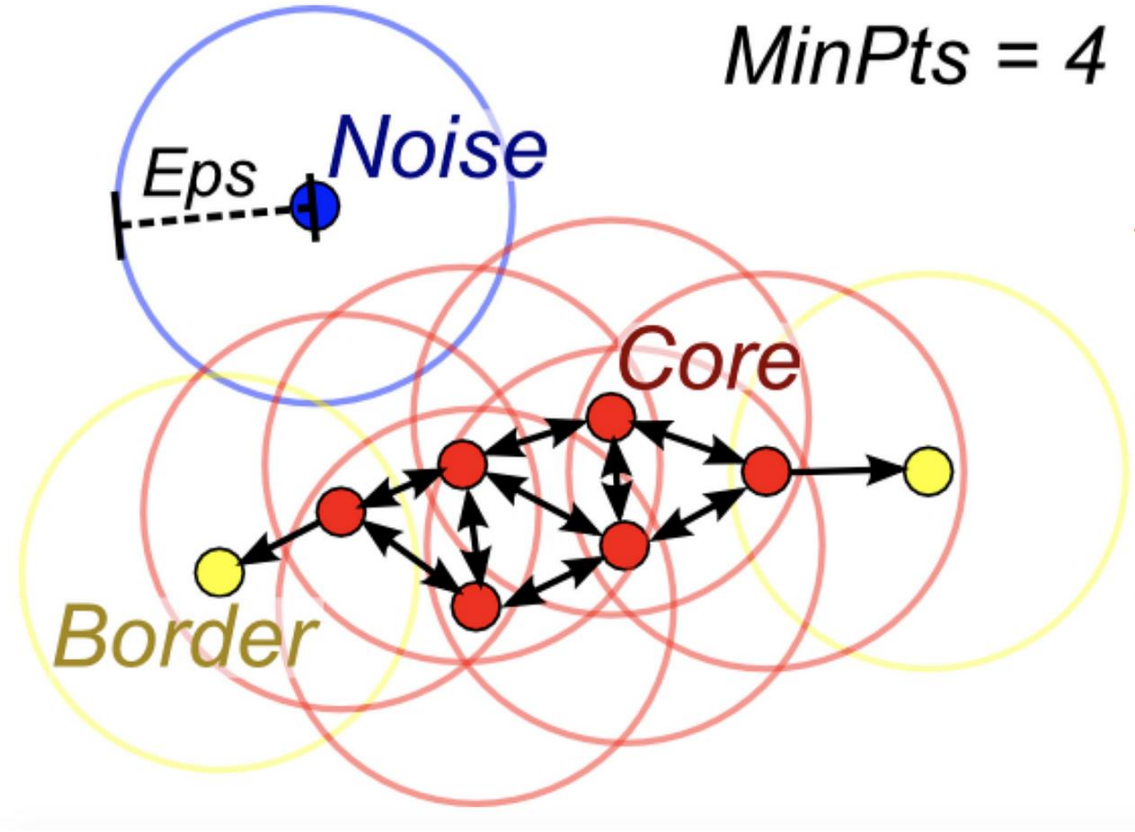


Algorytmy uczenia nienadzorowanego

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to algorytm klastrowania **oparty na gęstości**, który **identyfikuje klastry jako obszary o wysokiej gęstości punktów oddzielone obszarami o niskiej gęstości.**

W przeciwieństwie do K-means, DBSCAN **nie wymaga z góry określenia liczby klastrów** i potrafi wykrywać klastry o dowolnym kształcie oraz **radzić sobie z** hałasem (**punktami odstającymi**).

DBSCAN wymaga ustawienia **dwóch parametrów**: epsilon (ε), określającego **maksymalną odległość, w której dwa punkty są uważane za sąsiadujące**, oraz MinPts, określającego **minimalną liczbę punktów wymaganych do utworzenia klastra**.



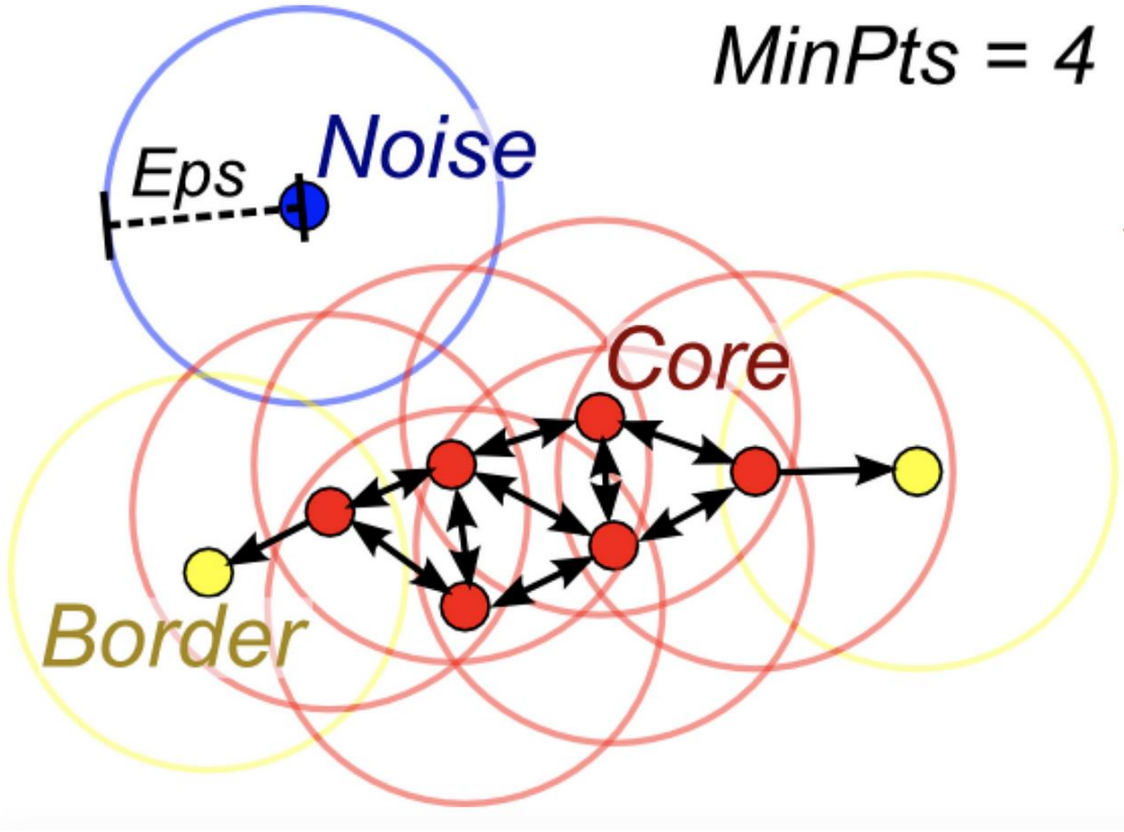
Algorytmy uczenia nienadzorowanego

DBSCAN

Czerwony: **punkty rdzeniowe**. Są to punkty, które mają co najmniej **3 sąsiadów** w promieniu **epsilon**. Punkty rdzeniowe stanowią **podstawę klastra**.

Żółty: **punkty brzegowe**. Są to punkty, które znajdują się **w promieniu epsilon od punktu rdzeniowego, ale same nie spełniają kryterium MinPts** (czyli nie mają co najmniej 3 sąsiadów w promieniu epsilon). Nadal są częścią klastra, ponieważ sąsiadują z punktem rdzeniowym.

Niebieski: **punkty szumu**. Są to punkty, które **nie są przypisane do żadnego klastra**. Nie znajdują się w promieniu epsilon żadnego punktu rdzeniowego i nie mają wystarczającej liczby sąsiadów, aby stać się punktem rdzeniowym.

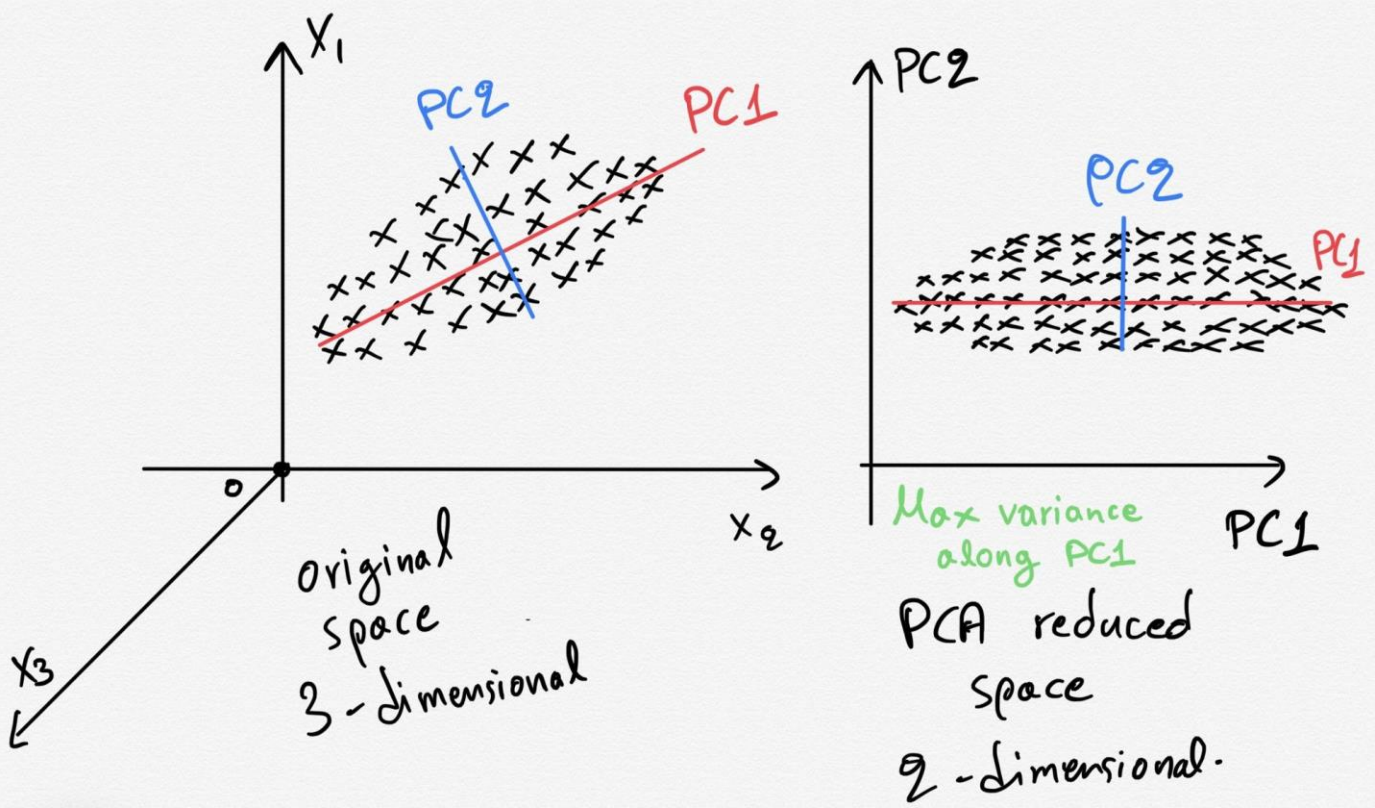


Algorytmy uczenia nienadzorowanego

PCA (Principal Component Analysis) to technika **redukcji wymiarów**, która przekształca dane do nowej przestrzeni o mniejszej liczbie wymiarów.

PCA **znajduje kierunki** (główne komponenty), wzdłuż których **dane mają największą wariancję,** i rzutuje dane na te kierunki.

Pozwala na uproszczenie danych i usunięcie korelacji między cechami, często używana do wizualizacji danych w 2D lub 3D.

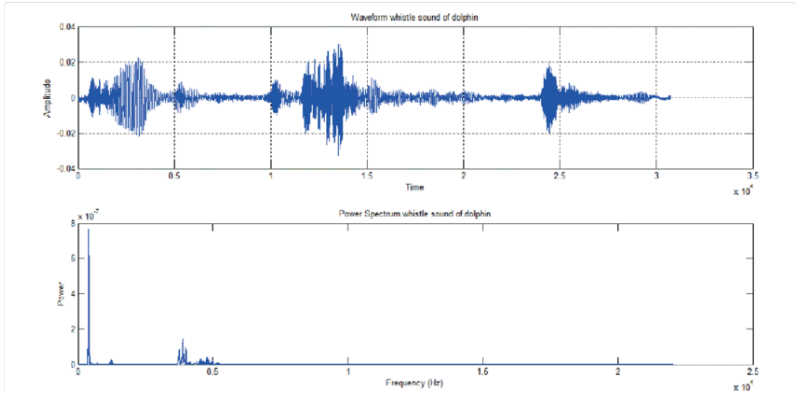


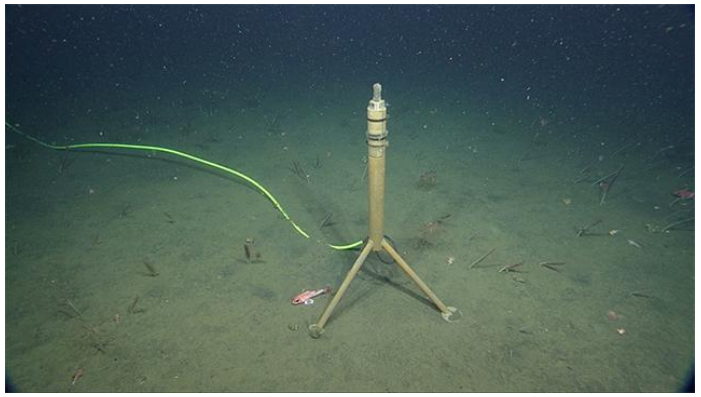
Uczenie częściowo nadzorowane

Uczenie częściowo nadzorowane (semi-supervised learning) to metoda uczenia maszynowego, która **wykorzystuje zarówno oznaczone, jak i nieoznaczone dane do treningu modelu.** Celem jest poprawa wydajności modelu w sytuacjach, gdzie **oznaczone dane są trudne do uzyskania lub kosztowne**, **ale nieoznaczone dane są łatwo dostępne**.

Kluczowe koncepcje i korzyści:

- **Wykorzystanie nieoznaczonych danych**: Pomaga modelowi lepiej zrozumieć strukturę danych, co może prowadzić do lepszych przewidywań.
- **Redukcja kosztów**: Mniej oznaczonych danych oznacza mniejsze koszty związane z etykietowaniem.
- **Lepsza wydajność**: Często osiąga lepsze wyniki niż modele wyłącznie nadzorowane, zwłaszcza gdy dostępne są duże ilości nieoznaczonych danych.





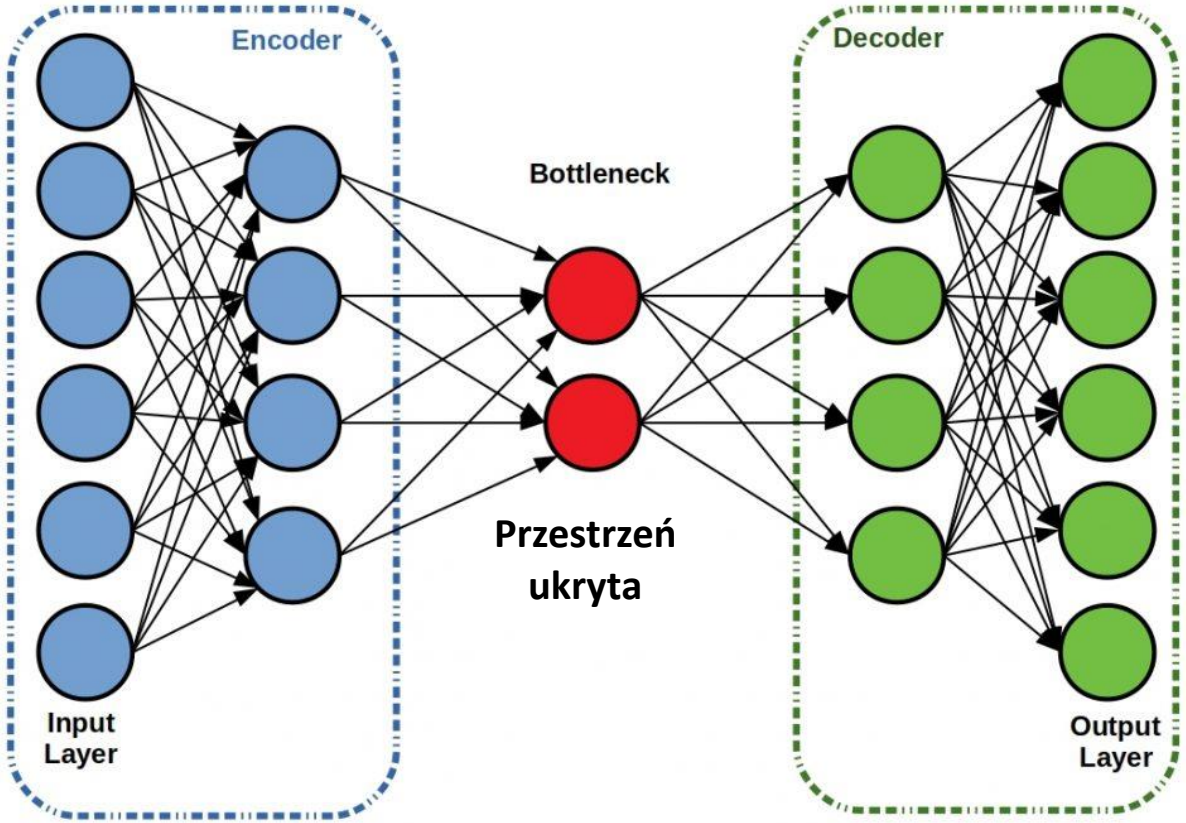
Autokoder

Autodekoder (ang. autoencoder) to **rodzaj sieci neuronowej**, który jest wykorzystywany **w uczeniu nienadzorowanym do kompresji i dekompresji danych**.

Jest to technika **redukcji wymiarów**, która ma na celu uczenie reprezentacji danych w sposób, który **minimalizuje błąd rekonstrukcji.**

Budowa autodekodera:

- ✓ **Enkoder:** Pierwsza część autodekodera, która **przekształca dane wejściowe do reprezentacji o mniejszej wymiarowości**. Najczęściej składa się z warstw splotowych lub gęstych.
- ✓ **Przestrzeń ukryta** *(bottleneck)***:** Jest to warstwa, która zawiera **reprezentację skompresowanych danych**. Jest to punkt w przestrzeni, który najlepiej opisuje dane wejściowe.
- ✓ **Dekoder:** Druga część autodekodera, która **rekonstruuje dane wejściowe z reprezentacji w przestrzeni ukrytej.** Może mieć taką samą strukturę jak enkoder, ale w odwrotnej kolejności.

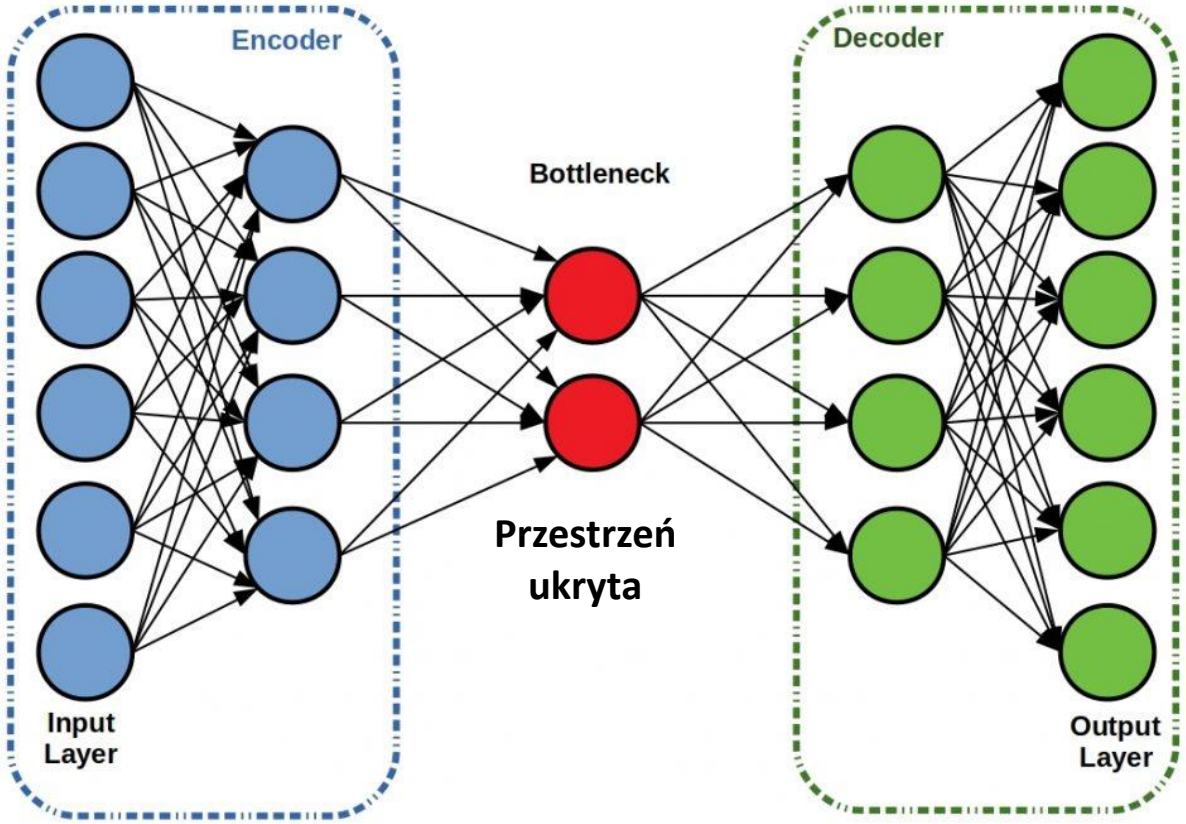


Autokoder

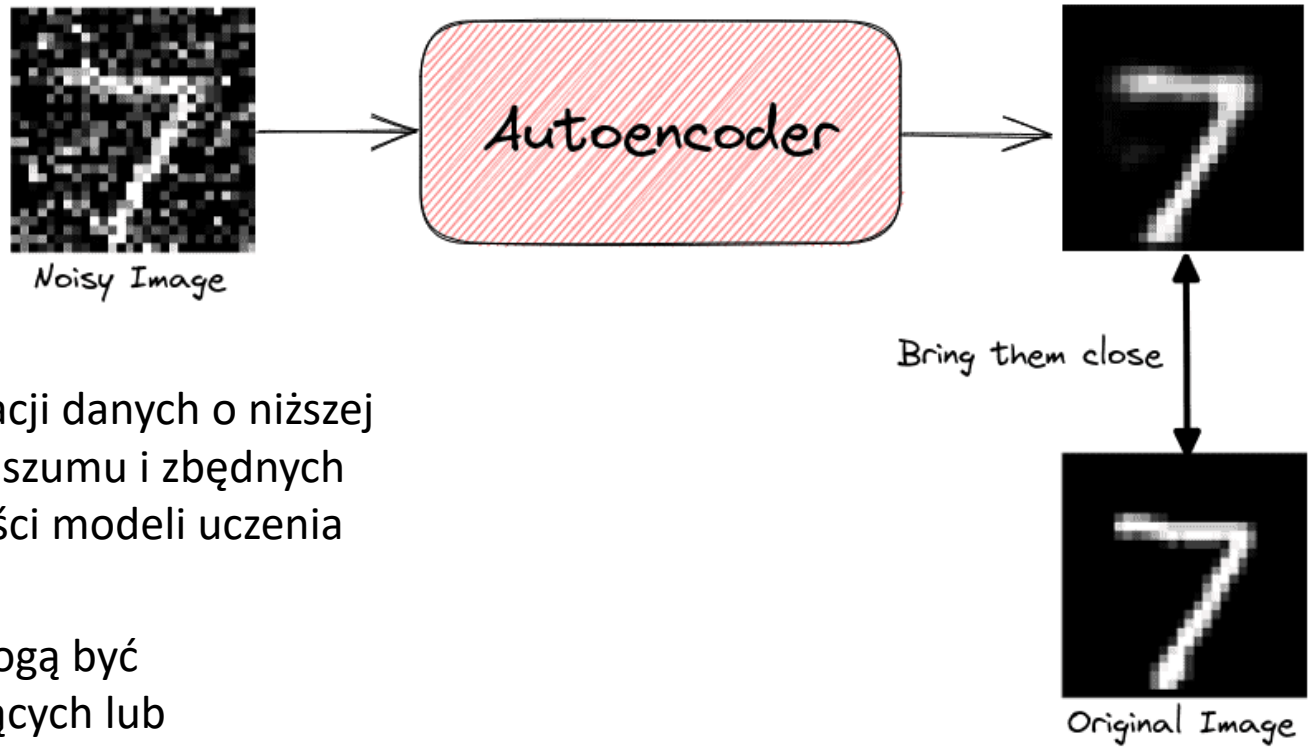
Autodekoder (ang. autoencoder) to **rodzaj sieci neuronowej**, który jest wykorzystywany **w uczeniu nienadzorowanym do kompresji i dekompresji danych**.

Działanie autodekodera:

- **Kompresja danych**: Dane wejściowe są przetwarzane przez enkoder, który tworzy skompresowaną reprezentację danych w przestrzeni ukrytej.
- **Dekompresja danych**: Skompresowana reprezentacja jest przekazywana do dekodera, który próbuje odtworzyć dane wejściowe z przestrzeni ukrytej.
- **Minimalizacja błędu rekonstrukcji**: Podczas treningu autodekodera dąży się do minimalizacji różnicy między danymi wejściowymi a ich rekonstrukcją. Jest to najczęściej realizowane przez minimalizację funkcji kosztu, takiej jak błąd średniokwadratowy (MSE).



Autokoder



Zastosowania autodekodera:

- **Redukcja wymiarów**: Uczenie reprezentacji danych o niższej wymiarowości może pomóc w eliminacji szumu i zbędnych informacji oraz w zwiększeniu skuteczności modeli uczenia maszynowego.
- **Rekonstrukcja danych**: Autodekodery mogą być wykorzystywane do rekonstrukcji brakujących lub uszkodzonych danych.
- **Generowanie danych**: Po nauczeniu się reprezentacji danych, autodekodery mogą generować nowe, podobne do tych, które były używane do treningu.

Kodujemy

Zaproponuj autokoder i wytrenuj go danymi treningowymi z wbudowanej bazy danych Keras minst (pomijając etykiety),

Usuń ostatnie warstwy autokodera, pozostawiając tylko enkoder,

Oblicz odpowiedź enkodera i wygeneruj kod dla danych wejściowych,

Użyj analizy KNN, aby określić klastry danych dla swojego kodu,

Dla 10 klastrów (10 cyfr) oznacz dane wejściowe,

Porównaj swoje wyniki z oryginalnymi etykietami.

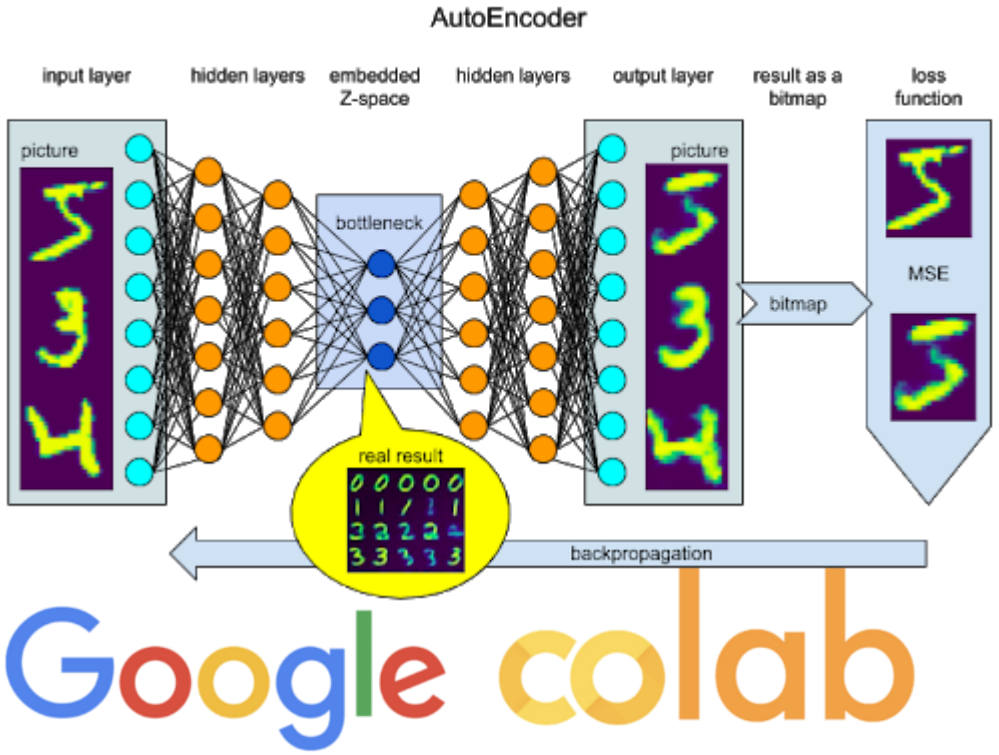
ftp mmajew/MIW/11/

00_simple_autoencoder_regularizers

01_Convolutional_autoencoder

02_autoencoder_noise

https://blog.keras.io/building-autoencoders-in-keras.html
https://keras.io/examples/vision/autoencoder/



Keras

Google colab

Kodujemy

encoded_imgs.mean()

oznacza ile średnio neuronów bierze udział w reprezentacji każdego obrazu

Rzadsze reprezentacje mają kilka zalet:

- **Redukcja redundancji** (nadmiarowość informacji): Mniej aktywnych neuronów może oznaczać bardziej efektywne i mniej redundantne kodowanie informacji.
- **Zwiększenie interpretowalności:** Rzadkie reprezentacje mogą ułatwić interpretację, ponieważ aktywność mniejszej liczby neuronów może bardziej jednoznacznie wskazywać na określone cechy wejściowych danych.
- **Regularizacja**: Rzadsze reprezentacje mogą działać jako forma regularizacji, pomagając modelowi w unikaniu przeuczenia, ponieważ model nie polega na dużej liczbie aktywnych neuronów do zakodowania informacji.

Keras

Google colab

Kodujemy

activity_regularizer=regularizers.l1(10e-5)

Jest to parametr warstwy Keras, który pozwala na zastosowanie funkcji regularizującej do aktywności tej warstwy. **Regularizacja** jest techniką, która ma na celu **zapobieganie przeuczeniu** (overfitting) modelu poprzez **dodanie kary do funkcji straty.**

regularizers.l1 odnosi się do typu regularizacji L1, znanej również jako regularizacja Lasso. Regularizacja L1 **dodaje do funkcji straty sumę wartości bezwzględnych wag** (lub w tym przypadku aktywności neuronów), co prowadzi **do wyzerowania niektórych z tych wag** (lub aktywności). W praktyce powoduje to, że **wiele wag staje się dokładnie zerowych, co skutkuje rzadkością (sparsity) rozwiązania.**

Celem jest wymuszenie rzadkości aktywności neuronów, co może pomóc w poprawie generalizacji modelu i zapobieganiu przeuczeniu.

Keras

Google colab