Tematy na dziś:

- Omówimy kluczowe wyzwania i zagrożenia w obszarze cyberbezpieczeństwa wynikające z gwałtownego rozwoju technologii.
- Analiza zależności między sztuczną inteligencją a cyberbezpieczeństwem w kontekście identyfikacji i ograniczania zagrożeń cybernetycznych.
- Wykorzystamy notatniki Jupyter do eksperymentowania z modelami AI oraz ich implementacji w rzeczywistych problemach związanych z cyberbezpieczeństwem.

AI w Cyberbezpieczeństwie

- Sama liczba nowych zagrożeń pojawiających się codziennie stanowi główny powód potrzeby zastosowania SI w cyberbezpieczeństwie

  Brakuje wystarczającej liczby cyberwojowników, a obecni analitycy są przytłoczeni ilością zagrożeń.

- AI może być wykorzystywana do wspierania analityków poziomu Tier 1 w triage'u incydentów cybernetycznych

  Analitycy będą musieli zrozumieć, jak korzystać z SI i interpretować jej wyniki, gdy wykonują:

  Klasyfikację – odróżnianie zachowań bezpiecznych od złośliwych na podstawie danych oznaczonych (etykietowanych).
  Klasteryzację – grupowanie zachowań, gdy brak jest szczegółowych informacji.
  Predykcję – dynamiczne identyfikowanie zagrożeń w momencie ich występowania.

  Analitycy będą musieli używać SI do realizacji zadań z zakresu cyberbezpieczeństwa, takich jak:

  Ochrona sieci – opracowywanie narzędzi wykrywających złośliwe zachowania w sieciach.
  Ochrona punktów końcowych (endpoints) – opracowywanie narzędzi do wykrywania złośliwego zachowania w systemach.
  Bezpieczeństwo aplikacji – opracowywanie narzędzi służących do monitorowania aplikacji internetowych i innych aplikacji pod kątem złośliwych zachowań.
  Wykrywanie zagrożeń wewnętrznych – tworzenie narzędzi umożliwiających szczegółowe monitorowanie użytkowników pod kątem zachowań oszukańczych.

Uczenie nadzorowane (Supervised Machine Learning)

Algorytmy uczenia maszynowego (ML) uczą się na podstawie oznaczonych danych:

- Regresja (liniowa i logistyczna)
- Metoda k-najbliższych sąsiadów (k-NN)
- Maszyny wektorów nośnych (SVM)
- Drzewa decyzyjne i lasy losowe
- Sieci neuronowe (NN)

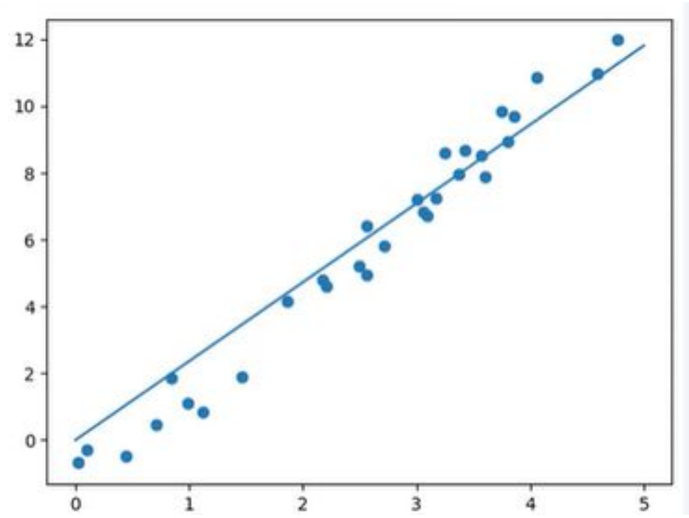
Jednym z zastosowań uczenia nadzorowanego w cyberbezpieczeństwie jest klasyfikacja spamu, wykorzystywana do tworzenia filtrów antyspamowych:

- Trenowane na przykładach spamu (złośliwych) oraz zwykłych (prawidłowych) wiadomości e-mail
- Testowane na wcześniej niewidzianych wiadomościach

Uczenie nadzorowane: Regresja liniowa

- Wykorzystano bibliotekę Python scikit-learn
  - Klasa LinearRegression
  - Pakiet linear_model
- Trenowanie modelu
  - y = 3*x – 2 + pool.randn(30) – zbiór danych treningowych określony liniową funkcją zakłóconą losowymi wartościami
  - lregr – obiekt klasy LinearRegression
  - lregr.fit(X, y) – trenowanie modelu
- Predykcja/Wynik
  - y_regr = lregr.predict(X_regr)

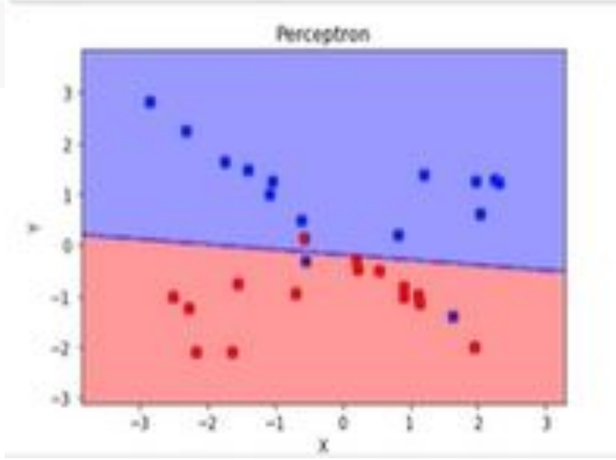
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

pool = np.random.RandomState(10)
x = 5 * pool.rand(30)
y = 3 * x - 2 + pool.randn(30)

# y = 3x - 2;
lregr = LinearRegression(fit_intercept=False)

X = x[:, np.newaxis]
lregr.fit(X, y)

lspace = np.linspace(0, 5)
X_regr = lspace[:, np.newaxis]
y_regr = lregr.predict(X_regr)

plt.scatter(x, y); # training dataset
plt.plot(X_regr, y_regr); # prediction
```



Uczenie nadzorowane: Sieć neuronowa

- Wykorzystano bibliotekę Python scikit-learn
  - Klasa perceptron
  - Pakiet linear_model
- Zbiór danych treningowych
  - Pakiet datasets
  - Klasa make_classification
    - X, y = make_classification(30, 2, 2, 0, weights=[.3, .3], random_state=300)
  - pct.fit(X, y) – trenowanie modelu
- Predykcja/Wynik
  - Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)

```
from matplotlib.colors import ListedColormap

# Thanks to Sebastian Raschka for 'plot_decision_regions' function
def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=cmap(idx),
                    marker=markers[idx], label=cl)

from sklearn.linear_model import perceptron

from sklearn.datasets import make_classification
X, y = make_classification(30, 2, 2, 0, weights=[.3, .3], random_state=300)

plt.scatter(X[:,0], X[:,1], s=50)

pct = perceptron.Perceptron(max_iter=100, verbose=0, random_state=300, fit_intercept=True, eta0=0.002)
pct.fit(X, y)

plot_decision_regions(X, y, classifier=pct)
plt.title('Perceptron')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



Uczenie nienadzorowane (Unsupervised Machine Learning)

- Algorytmy uczenia maszynowego (ML) uczą się bez wykorzystania danych oznaczonych:
  - Redukcja wymiarowości – kompresja danych o wysokiej wymiarowości do przestrzeni o niższym wymiarze
    - Analiza głównych składowych (PCA)
    - PCA Kernel
  - Grupowanie (klasteryzacja) – organizacja danych wejściowych w grupy
    - Metoda k-średnich (k-Means)
    - Hierarchiczna analiza skupień (HCA)
- Jednym z zastosowań uczenia nienadzorowanego w cyberbezpieczeństwie jest wykrywanie kampanii złośliwego oprogramowania oraz oszustw

Uczenie nienadzorowane: PCA oraz klasteryzacja

- data_df - Dane wejściowe jako obiekt biblioteki pandas
- PCA z biblioteki scikit-learn
  - Pakiet decomposition
  - Klasa PCA
    - pca.fit(X_data) – wprowadza dane 4-wymiarowe do algorytmu PCA
    - X_2D = pca.transform(X_data)
    - data_df["PCA1"] = X_2D[:, 0]
    - data_df["PCA2"] = X_2D[:, 1] – redukuje wymiar z 4 do 2

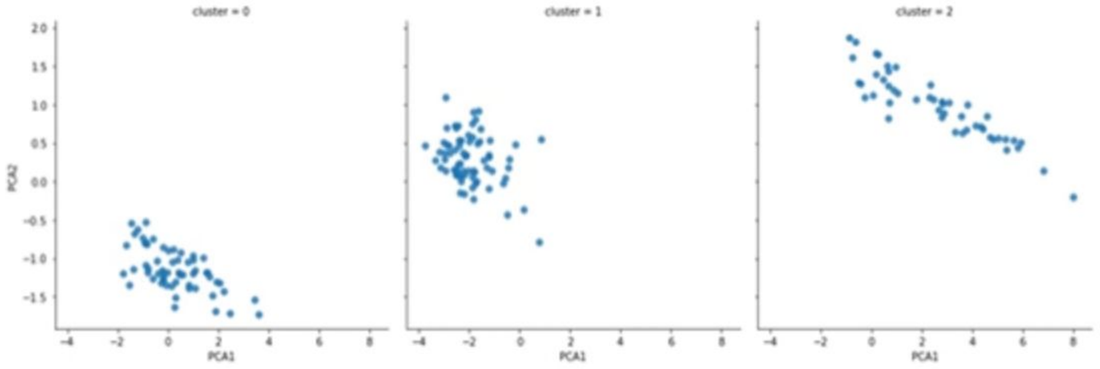
```
import pandas as pd
import seaborn as sns

data_df = pd.read_csv("../datasets/clustering.csv")
data_df.describe()

X_data = data_df.drop('class_1', axis=1)
y_data = data_df['class_1']

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_data)
X_2D = pca.transform(X_data)

data_df['PCA1'] = X_2D[:, 0]
data_df['PCA2'] = X_2D[:, 1]
```

Uczenie nienadzorowane: PCA oraz klasteryzacja

- Klasteryzacja z biblioteki scikit-learn
  - Pakiet mixture
  - Klasa GaussianMixture
    - gm.fit(X_data) – wprowadza nowe, zredukowane dane 2-wymiarowe do algorytmu klasteryzacji
    - y_gm = gm.predict(X_data) – przypisuje klastry nowym, zredukowanym 2-wymiarowym danym

```
from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=3, covariance_type='full')

gm.fit(X_data)
y_gm = gm.predict(X_data)


data_df['cluster'] = y_gm
sns.lmplot("PCA1", "PCA2", data=data_df, col='cluster', fit_reg=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x2a6095f5f8>
```



Uczenie wzmacniane (Reinforcement Learning)

- Algorytmy uczenia maszynowego przez wzmacnianie (ML) uczą się poprzez interakcję ze środowiskiem metodą prób i błędów, kierując się macierzą nagród:
  - Proces Markowa
  - Q-uczenie (Q-learning)
  - Metoda różnicy czasowej (TD)
  - Metody Monte Carlo
- Poprawna decyzja przynosi nagrodę dodatnią, a błędna decyzja – nagrodę ujemną
- Jednym z zastosowań uczenia przez wzmacnianie w cyberbezpieczeństwie jest wykrywanie polimorficznego złośliwego oprogramowania przy użyciu ukrytych modeli Markowa (HMM)

Rozwój analiz opartych na uczeniu maszynowym

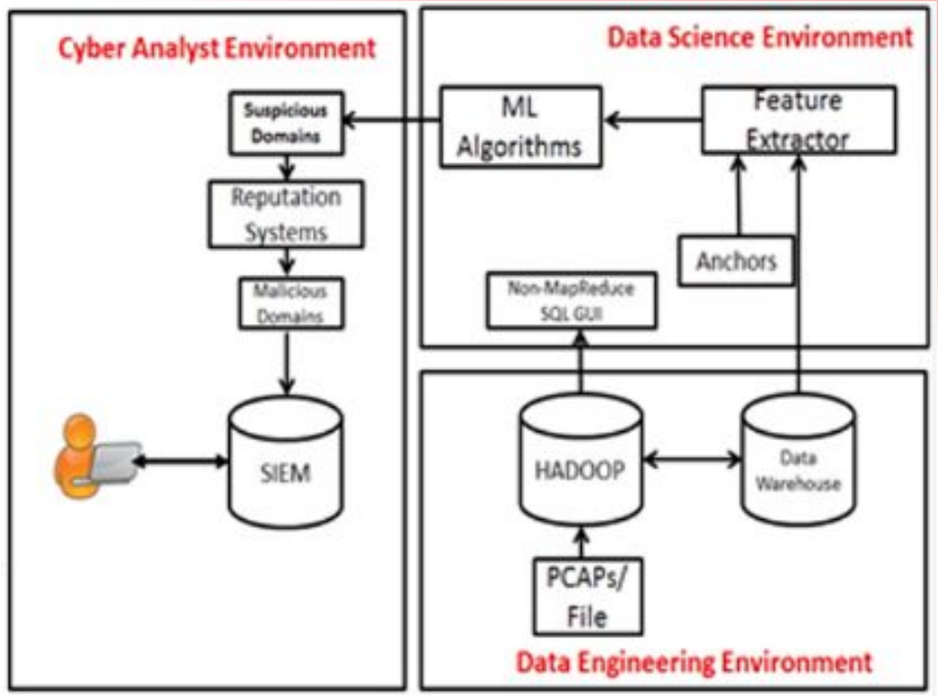
- Inżynier danych (Data Engineer) – Wyszukuje dane i opracowuje architekturę, aby dostarczyć dane do użytku Data Scientistowi w odpowiednim formacie.

- Data Scientist – Tworzy analizę ML poprzez:

  - Inżynierię cech (Feature engineering) – identyfikacja przekształceń matematycznych w celu wydobycia wzorców z danych

  - Inżynierię modeli (Model engineering) – trenowanie odpowiednich algorytmów ML do rozwiązywania problemów

  - Testy operacyjne – testowanie analizy ML w rzeczywistym środowisku



Rozwój analiz opartych na uczeniu maszynowym

- Najlepsze praktyki
  - Zbiory danych do wykrywania anomalii powinny zawierać wyłącznie dane dotyczące normalnego zachowania
  - Często konieczne jest oczyszczenie danych, co eliminuje dane nieistotne lub zduplikowane
  - Zbiory danych mogą mieć wiele cech (wymiarów), które nie dostarczają istotnych informacji – należy wtedy zredukować wymiarowość
  - Zbiory danych do klasyfikacji muszą mieć oznaczone dane zarówno złośliwe, jak i nieszkodliwe
    - Bardzo ważne jest, aby dane były reprezentatywne dla zjawisk, które mają być klasyfikowane

Wykrywanie zagrożeń e-mailowych przy użyciu AI: Podstawy

- E-mail jest jednym z najczęściej wykorzystywanych wektorów ataku
- Ze względu na ogromny wolumen ruchu e-mailowego, skuteczna automatyzacja może być tutaj pomocna
- Jednym z pierwszych i skutecznych zastosowań sztucznej inteligencji w cyberbezpieczeństwie był SpamAssassin (https://en.wikipedia.org/wiki/Apache_SpamAssassin)

Podstawowy filtr spamu (Spam Filters 101)

1. Śledź obecność i brak podejrzanych słów kluczowych (i przypisz im wagi wg znaczenia) w e-mailach oraz ich częstotliwość.
2. Przypisz wynik (score) na podstawie częstotliwości słów kluczowych dla każdego e-maila i zapisz je w tabeli.
3. Przeanalizuj tabelę i ustal próg (threshold), aby oddzielić spam od wiadomości prawidłowych (ham).
4. Okresowo przeliczaj próg na podstawie nowych danych odniesienia (ground truth).

Ciągłe innowacje ze strony spamerów wymuszają dynamiczne podejście, co czyni ten problem odpowiednim do rozwiązania za pomocą AI.

Przykład: Załóżmy, że słowa kluczowe to „prize” i „buy”; prosty klasyfikator: spam = obecność „buy” (B) i „prize” (S)

Email	Buy	Prize	Spam or Ham?
1	1	0	H
2	0	1	H
3	0	0	H
4	1	1	S

Podstawowy filtr spamu (Spam Filters 101)

Przykład: Dla funkcji punktowej: y = B + S

Można zastosować wagi: y = 2B + 3S

Próg uznania za spam: y > 4

Email	B	S	2B + 3S	Spam or Ham?
1	1	0	2	H
2	0	1	3	H
3	0	0	0	H
4	1	1	5	S

Perceptron

- Najbardziej podstawową wersją sieci neuronowej (NN) jest perceptron
- Wszystkie sieci neuronowe w pewnym stopniu naśladują zachowanie neuronów ludzkiego mózgu
- Perceptron to jedna z pierwszych udanych implementacji sztucznej inteligencji (AI) inspirowanych neuronem
  - W zasadzie jest to struktura warstwowa, która łączy dane wejściowe z wyjściem za pomocą:

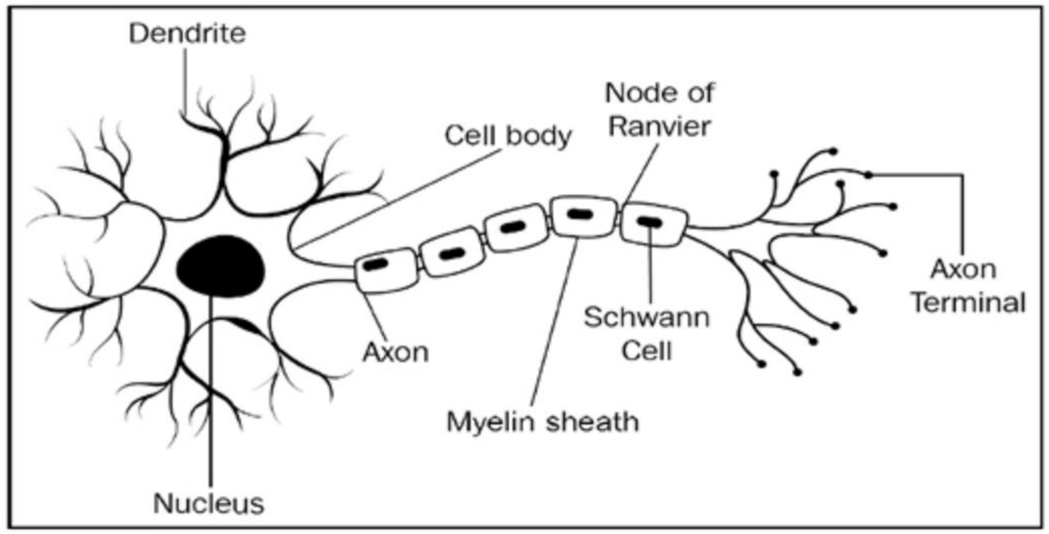
    Wag wejściowych

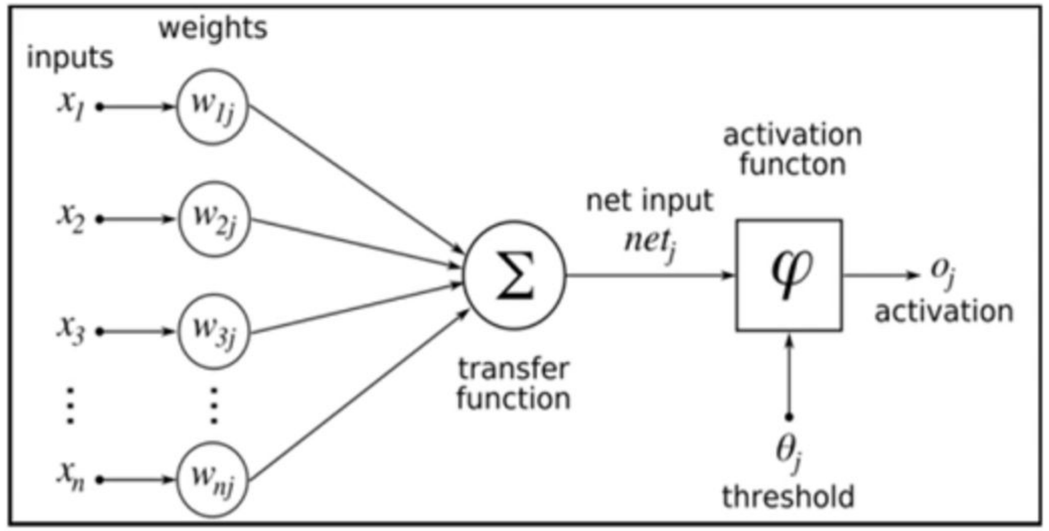
    Progu (threshold)

    Funkcji aktywacji

  - Główna różnica między modelami statystycznymi a modelami AI polega na iteracyjnej strategii optymalizacji AI

    Celem jest znalezienie optymalnych wag, które pozwolą na trafr przewidywania na podstawie nowych, nieznanych danych





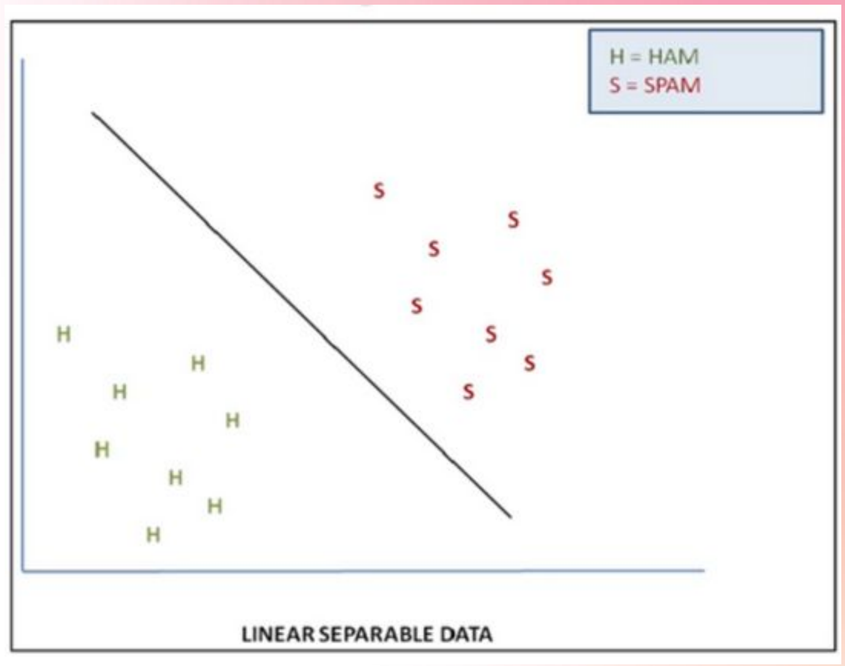
Perceptron Rosenblatta

Jeśli $w \cdot x \geq \Theta$, to f(y) = +1 – perceptron się aktywuje (1)

Jeśli $w \cdot x < \Theta$, to f(y) = -1 – perceptron się nie aktywuje (2)

$w_i = w_i + \Delta w_i$ – aktualizacja wagi (3)

$\Delta w_i = \lambda(y - y_i) \cdot x_i$ – zmiana wagi zależna od różnicy między wartością oczekiwaną a przewidywaną (4)



1. Opisuje aktywację perceptronu - Iloczyn danych wejściowych i wag musi przekroczyć próg aktywacji
2. Opisuje brak aktywacji perceptronu - Gdy suma ważona nie przekroczy progu – perceptron nie podejmuje działania
3. Uczenie perceptronu - Perceptron uczy się, iteracyjnie aktualizując swoje wagi w oparciu o różnicę między oczekiwaną wartością y (z oznaczonych danych treningowych), a przewidywaną wartością y_i. Ta różnica jest przeliczana na zmianę wag Δw_i
4. λ (lambda) – oznacza współczynnik uczenia się - Przyjmuje wartość pomiędzy 0 a 1 i określa, jak duży krok wykonuje perceptron przy aktualizacji wag

Filtr spamu z użyciem perceptronu – Krok 1 i 2

1. Mając publiczny zbiór danych ze spamem, filtrujemy wiersze zawierające słowa kluczowe z naszego przykładu – sex i buy.

2. Następnie tworzymy nowy zbiór danych i śledzimy częstotliwość występowania tych słów kluczowych w każdej wiadomości e-mail w nowym zestawie danych.

```
# Execute plot() inline without calling show()
%matplotlib inline
import warnings
warnings.simplefilter('ignore')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv('../datasets/sms_spam_perceptron.csv')

y = df.iloc[:, 0].values
y = np.where(y == 'spam', -1, 1)

X = df.iloc[:, [1, 2]].values
```

https://archive.ics.uci.edu/dataset/228/sms+spam+collection

1	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4	ham	U dun say so early hor... U c already then say...
5	ham	Nah I don't think he goes to usf, he lives around here though
6	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, Ã£1.50 to rcv
7	ham	Even my brother is not like to speak with me. They treat me like aids patent.
8	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
9	spam	WINNER!! As a valued network customer you have been selected to receivea Ã£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
		Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on

SMSSpamCollection

1	type	sex	buy
2	ham	0	1
3	ham	0	1
4	ham	1	1
5	spam	1	0
6	ham	0	1
7	spam	1	0
8	ham	0	1
9	ham	0	1
10	spam	2	1
11	ham	0	1
12	ham	0	1
13	ham	0	1
14	ham	0	1
15	ham	0	1

Filtr spamu z użyciem perceptronu – Krok 3 - 5

3. Nowy zbiór danych jest dzielony w proporcji 70% do 30% na zbiór treningowy i testowy

4. Następnie tworzymy obiekt klasy Perceptron z pakietu `linear_model` biblioteki `sklearn`, konfigurujemy liczbę iteracji oraz współczynnik uczenia (learning rate), a potem trenujemy model.

5. Na końcu testujemy perceptron na zbiorze testowym.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=0)
```

•••

```
from sklearn.linear_model import Perceptron

p = Perceptron(max_iter=40, eta0=0.1, random_state=0)
p.fit(X_train, y_train)
```

```
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=0.1,
      fit_intercept=True, max_iter=40, n_iter=None, n_iter_no_change=5,
      n_jobs=None, penalty=None, random_state=0, shuffle=True, tol=None,
      validation_fraction=0.1, verbose=0, warm_start=False)
```

```
y_pred = p.predict(X_test)
```

Filtr spamu z użyciem perceptronu – Krok 6 - 7

6. Następnie wizualizujemy część danych wejściowych na tle obszaru decyzyjnego.

7. Potem sprawdzamy dokładność perceptronu – i okazuje się, że jest całkiem dobra.

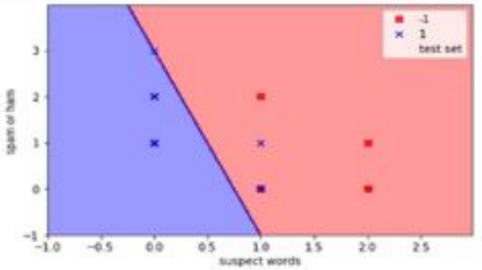
Zalety:

- Prosty, ale skuteczny

Wady:

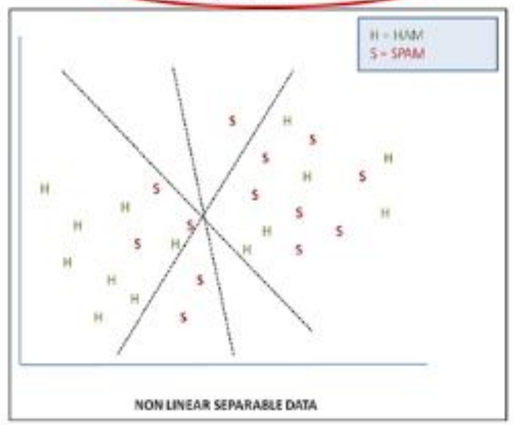
- Ograniczony typ problemów, które może rozwiązać:
  - Tylko liniowo separowalne dane
  - W przypadku danych nieliniowo separowalnych, algorytm będzie „oscylował” w przestrzeni decyzyjnej, nie osiągnie stabilnej granicy i da złe wyniki nawet po maksymalnej liczbie iteracji

```
from defs import plot_decision_regions

X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))

plot_decision_regions(X=X_combined, y=y_combined,
                      classifier=p, test_idx=range(-5, 5))
plt.xlabel('suspect words')
plt.ylabel('spam or ham')
plt.legend(loc='upper right')

plt.tight_layout()
plt.show()
```




```
[7]: from sklearn.metrics import accuracy_score

print('Misclassified samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
```

```
Misclassified samples: 3
Accuracy: 0.90
```



Maszyna wektorów nośnych (SVM – Support Vector Machine)

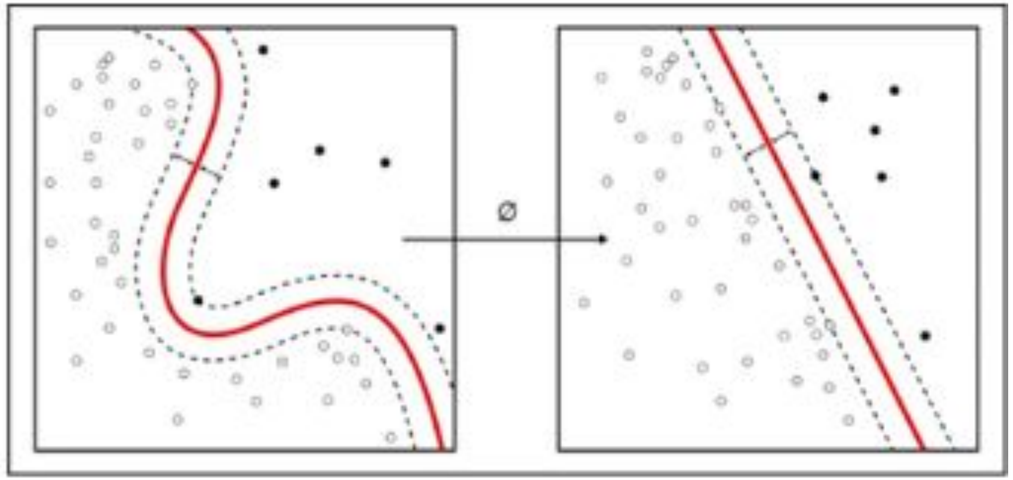
- Jest to algorytm uczenia nadzorowanego, podobnie jak perceptron
- SVM stara się znaleźć hiperpłaszczyznę (hyperplane), która najlepiej oddziela klasy danych, maksymalizując margines między tą hiperpłaszczyzną a danymi
- Może również klasyfikować dane nieliniowo separowalne
- Szersze marginesy zmniejszają liczbę błędów, ale zwiększają ryzyko nadmiernego dopasowania (overfittingu)
- SVM vs. Perceptron
  - Celem perceptronu jest klasyfikowanie danych poprzez minimalizację błędów klasyfikacyjnych (1) :
  - Celem SVM jest maksymalizacja marginesu pomiędzy hiperpłaszczyzną a najbliższymi dan[illegible] (2) :

$$y = \Sigma w_i * x_i \quad (1)$$

$$y = \Sigma w_i * x_i + \beta >= \mu \quad (2)$$

    - β to bias (przesunięcie), a μ to margines
    - β pozwala y przyjmować wartości większe lub równe zeru
    - Wartość y porównywana jest z μ, by upewnić się, że odległość każdego punktu od hiperpłaszczyzny jest większa lub równa marginesowi

Maszyna wektorów nośnych (SVM – Support Vector Machine)



Filtr spamu z użyciem SVM – Kroki 1–2

- Korzystając z publicznego zbioru danych o spamie, filtrujemy wiersze zawierające określone słowa kluczowe.

- Następnie budujemy nowy zbiór danych i śledzimy częstotliwość występowania tych słów kluczowych w każdej wiadomości e-mail w tym nowym zbiorze.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('../datasets/sms_spam_svm.csv')

y = df.iloc[:, 0].values
y = np.where(y == 'spam', -1, 1)

X = df.iloc[:, [1, 2]].values
```

https://archive.ics.uci.edu/dataset/228/sms+spam+collection

1	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4	ham	U dun say so early hor... U c already then say...
5	ham	Nah I don't think he goes to usf, he lives around here though
6	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, Â£1.50 to rcv
7	ham	Even my brother is not like to speak with me. They treat me like aids patent.
8	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
9	spam	WINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
		Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on

SMSSpamCollection

	type	suspect	neutral
2	ham	1	3
3	ham	49	30
4	spam	47	32
5	ham	46	31
6	ham	0	36
7	spam	4	39
8	ham	46	34
9	ham	0	34
10	spam	44	29
11	spam	49	31
12	ham	4	37
13	spam	48	34
14	spam	48	30
15	ham	43	30
16	ham	8	40
17	spam	7	44
18	ham	4	39
19	ham	1	3
20	ham	7	38
21	spam	1	38
22	ham	4	34
23	ham	1	37
24	ham	46	36
25	ham	1	33
26	ham	48	34
27	ham	0	30

sms_spam_svm

Filtr spamu z użyciem SVM – Kroki 3–5

3. Nowy zbiór danych dzielimy w proporcji 70% na dane treningowe i 30% na dane testowe.

4. Następnie tworzymy obiekt klasy SVC z pakietu `sklearn.svm`, konfigurujemy go jako klasyfikator liniowy i trenujemy.

5. Na koniec testujemy model SVM na zbiorze testowym.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
         X, y, test_size=0.3, random_state=0)

from sklearn.svm import SVC

svm = SVC(kernel='linear', C=1.0, random_state=1)
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)
```

Filtr spamu z użyciem SVM – Kroki 6–7

6. Następnie wizualizujemy część punktów danych wejściowych na tle regionu de

7. Sprawdzamy dokładność modelu SVM – całkiem dobra, zwłaszcza dla danych, perceptron nie potrafi sobie poradzić.

SVM można również stosować do wykrywania spamu wysyłanego w formie obrazów

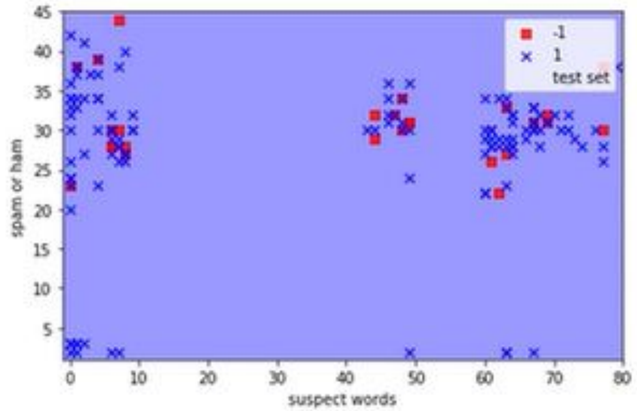
Typowe strategie wykrywania obrazów:

- Filtrowanie oparte na zawartości – próba identyfikacji podejrzanych słów kluczowych na podstawie rozpoznawania tekstu (OCR – optyczne rozpoznawanie znaków)

- Filtrowanie niezależne od zawartości – identyfikacja charakterystycznych cech w obrazach spamowych, które odróżniają je od normalnych obrazów

https://scholarworks.sjsu.edu/etd_projects/486/

[5]:
```
# Thanks to Sebastian Raschka for 'plot_decision_regions'
# https://github.com/rasbt/python-machine-learning-book
from defs import plot_decision_regions

X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))

plot_decision_regions(X_combined, y_combined,
                      classifier=svm, test_idx=range(-15, 15))
plt.xlabel('suspect words')
plt.ylabel('spam or ham')
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```



[6]:
```
from sklearn.metrics import accuracy_score

print('Misclassified samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
```

```
Misclassified samples: 7
Accuracy: 0.84
```