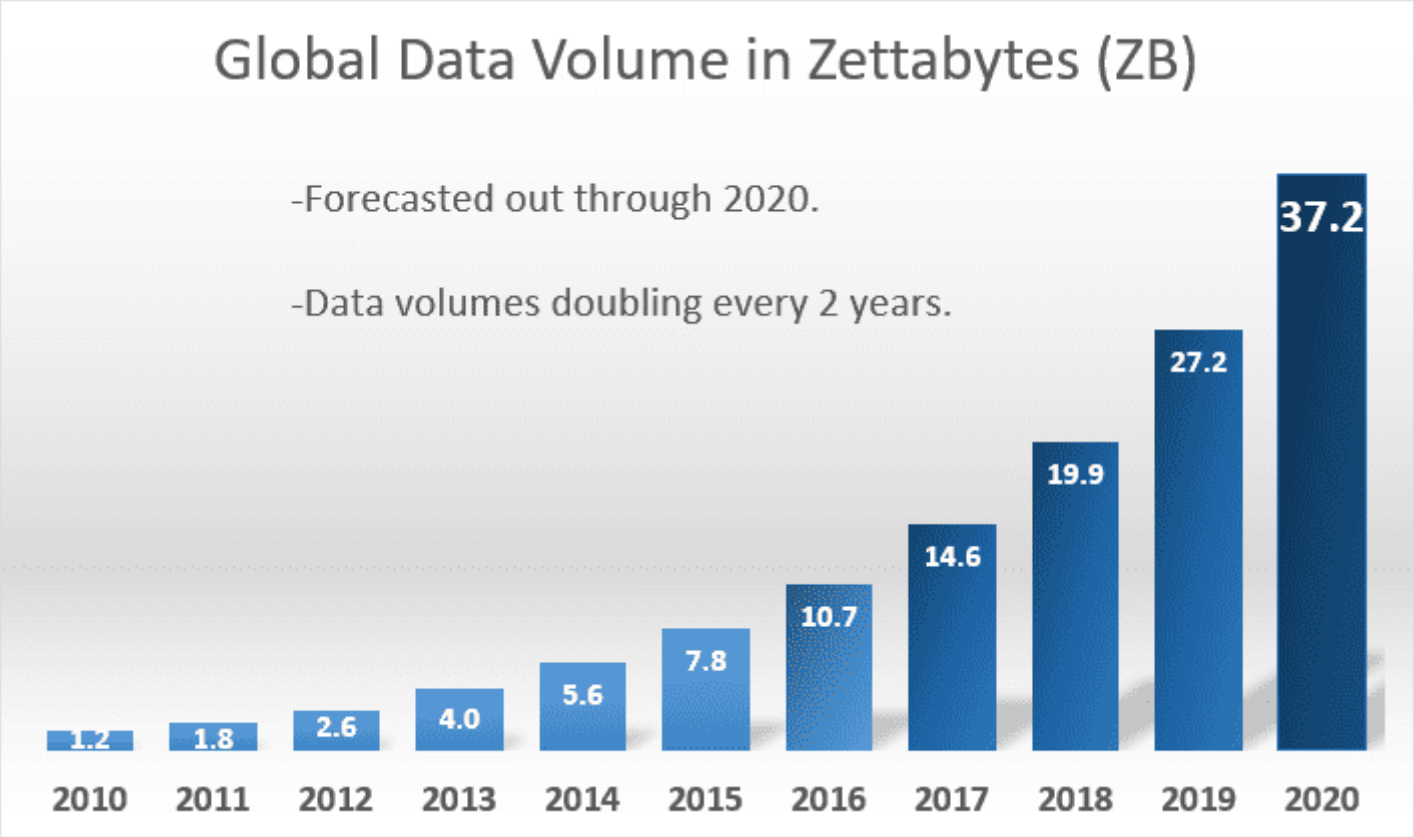
Metody uczenia maszynowego dla danych strumieniowych

Stanisław Kaźmierczak

Agenda/Źródła

1. Motywacja
2. Porównanie z tradycyjnym ML
3. Algorytmy próbkowania
4. Przegląd algorytmów strumieniowych
5. Przykłady i wyzwania

Era danych (1)

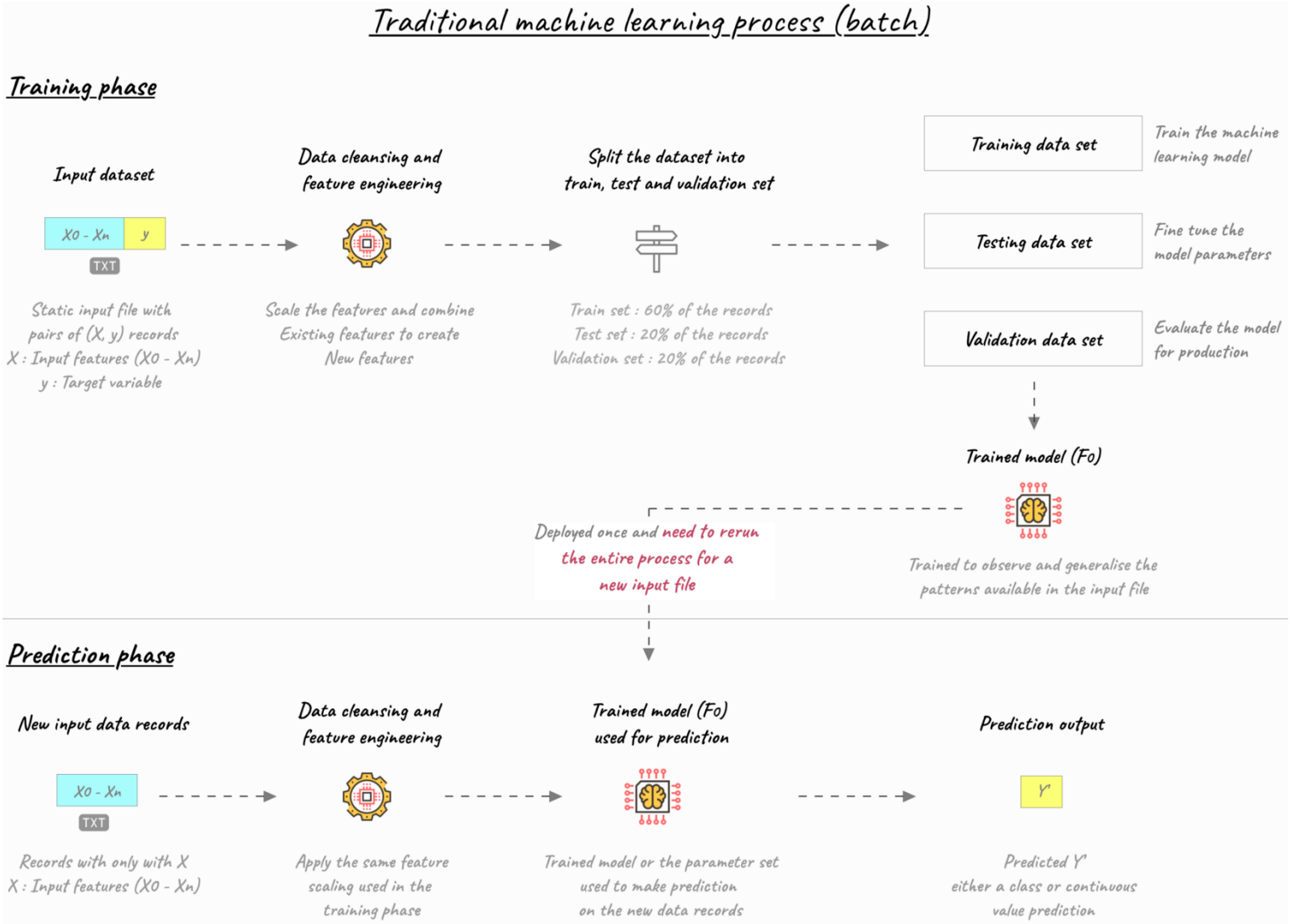


[1]

Era danych (2)

- W 2018 roku w ciągu minuty
  - Użytkownicy YouTube obejrzeli 4.3 miliona filmów
  - Amazon wysłał 1,111 paczek
  - Na Twitterze pojawiło się 0.5 miliona wpisów
  - Wrzucono 50,000 zdjęć na Instagrama
  - Wyszukiwarka Google została użyta 3.8 miliona razy
- W ciągu sekundy w przeliczeniu na osobę generowanych jest ok. 1.7 MB

Tradycyjne (nadzorowane) ML



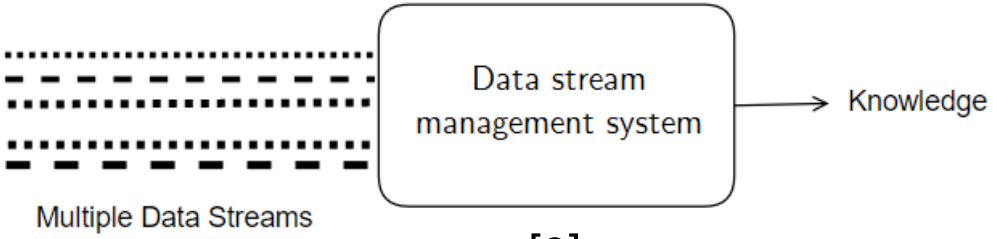
[2]

Tradycyjne ML – ograniczenia

- Model jest ograniczony do danych, które otrzymał jako statyczne wejście
- Nie potrafi dostosować się do zmian, które zachodzą w czasie
- Ogólniej: za każdym razem, gdy pojawiają się nowe dane, proces uczenia należy przeprowadzić od początku
  - Kosztowne
  - Czasochłonne
  - Czasami niemożliwe z powodu braku dostępu do wcześniejszych danych
- Większość rzeczywistych danych ma charakter przyrostowy
  - Większość badań skupia się na analizie klasycznych zbiorów benchmarkowych w wersji statycznej

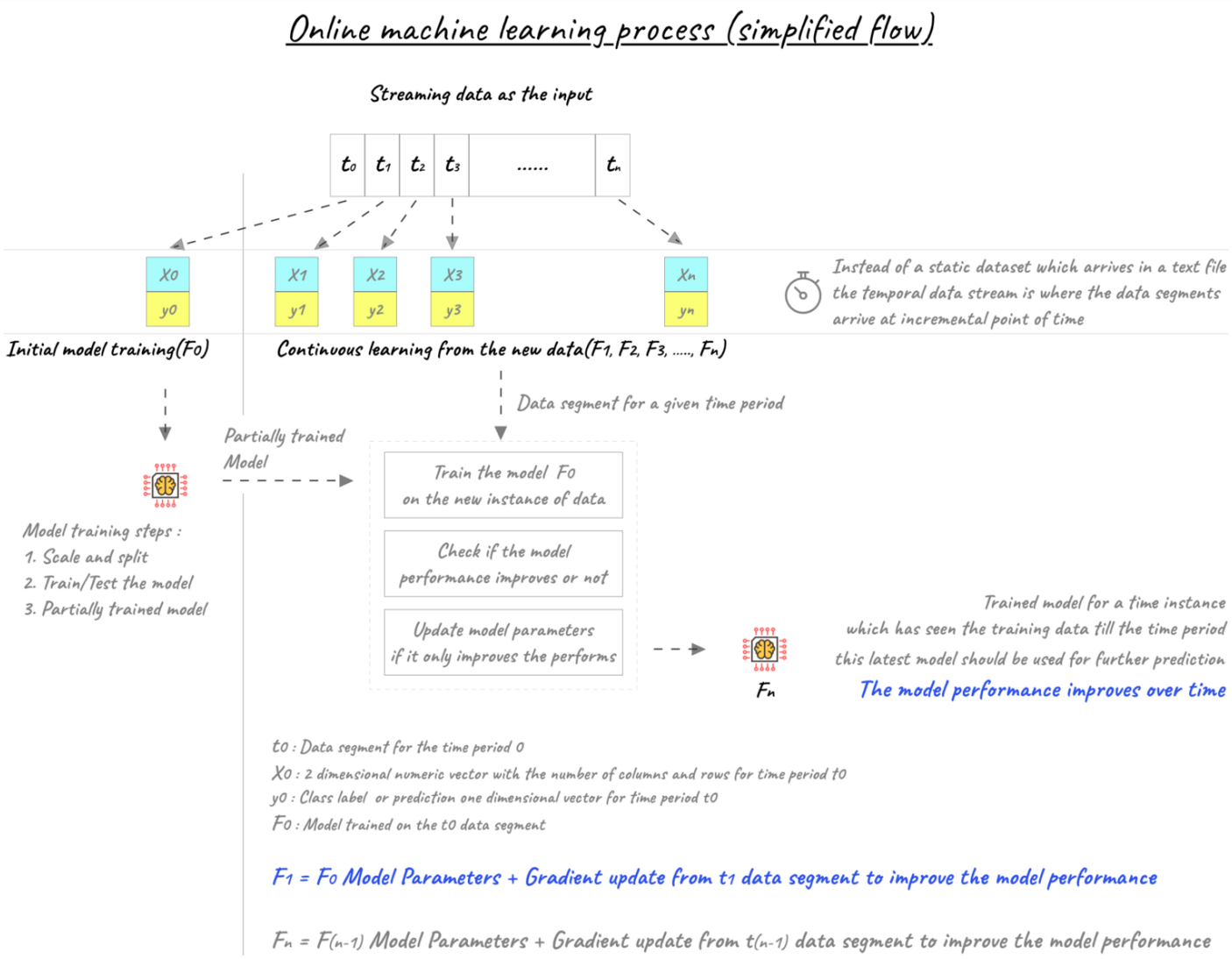
Strumień danych – założenia

- Napływające dane stanowią uporządkowaną sekwencję
- Strumień danych jest w ogólności nieskończony
  - Nie wiemy czy/kiedy się skończy
  - Nie wiemy też czy/jak szybkość napływu danych się zmieni
- Dane ewoluują w czasie (*concept drift*)
- Dane są ulotne
  - Przeprocesowane dane są usuwane lub agregowane w postaci statystyk
- Czasami predykcja musi odbywać się w czasie niemal rzeczywistym



[3]

Streaming/online/incremental ML



[2]

Zastosowania

- Wszędzie tam, gdzie charakter danych zmienia się w czasie
- Gdy ogólny model należy dotrenować na danych szczegółowych (i wrażliwych)
  - Urządzenia związane z monitorowaniem stanu pacjenta
- Gdy urządzenie działa offline i nie jesteśmy w stanie wysłać nowych danych
- W przypadku dużych zbiorów danych, które nie mieszczą się w całości do pamięci
- Gdy dotrenowywanie i dystrybucja nowego modelu jest niemożliwa/zbyt kosztowna

Stability-plasticity dilemma

- Dane zmieniają się w czasie
- Projektując system uczący chcielibyśmy, aby był on jednocześnie stabilny jak i plastyczny
  - Stabilność – system nie zmienia się pod wpływem niewłaściwych/niereprezentatywnych danych
  - Plastyczność – system adaptuje się do istotnych danych
- Dylemat
  - Jak osiągnąć stabilność bez sztywności oraz plastyczność bez chaosu?
  - Jak zachować nabytą wiedzę (do danych mamy dostęp tylko raz)

Próbkowanie

- Szybkość napływu danych może wymuszać w niektórych sytuacjach konieczność wyboru części danych
- Próbkowanie zbiornikowe (*Reservoir sampling*)
  - Cel: wybrać, bez zwracania, k-elementową próbę losową z n-elementowej populacji
  - Dozwolony jest tylko jednorazowe użycie każdego elementu
  - Rozmiar populacji n jest nieznany
  - Kolejne elementy populacji stają się stopniowo dostępne dla algorytmu
  - Algorytm nie może wracać do wcześniejszych elementów
  - W każdej chwili algorytm musi posiadać rozwiązanie (nie wiemy kiedy nastąpi zakończenie działania)

Algorytm *R*

```
(* S has items to sample, R will contain the result *)
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i := 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i := k+1 to n
    (* randomInteger(a, b) generates a uniform integer from the inclusive range {a, ..., b} *)
    j := randomInteger(1, i)
    if j <= k
        R[j] := S[i]
```

[4, 5]

- Dla każdego i, prawdopodobieństwo wybrania i-tego elementu wynosi k/i
  - Zatem na zakończenie działania prawdopodobieństwo wybrania dowolnego elementu wynosi k/n
- Konieczność losowania dla każdego elementu populacji implikuje liniową złożoność algorytmu

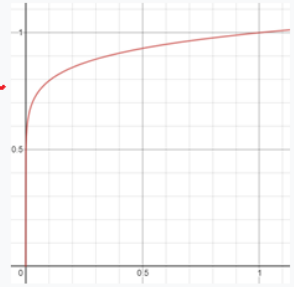
Algorytm L

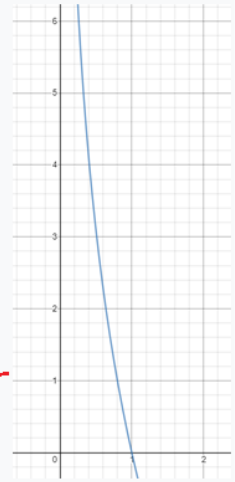
```
(* S has items to sample, R will contain the result *)
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  (* random() generates a uniform (0,1) random number *)
  W := exp(log(random())/k)

  while i <= n
      i := i + floor(log(random())/log(1-W)) + 1
      if i <= n
          (* replace a random item of the reservoir with item i *)
          R[randomInteger(1,k)] := S[i]  // random index between 1 and k, inclusive
          W := W * exp(log(random())/k)
```

z każdą iteracją wartość W maleje



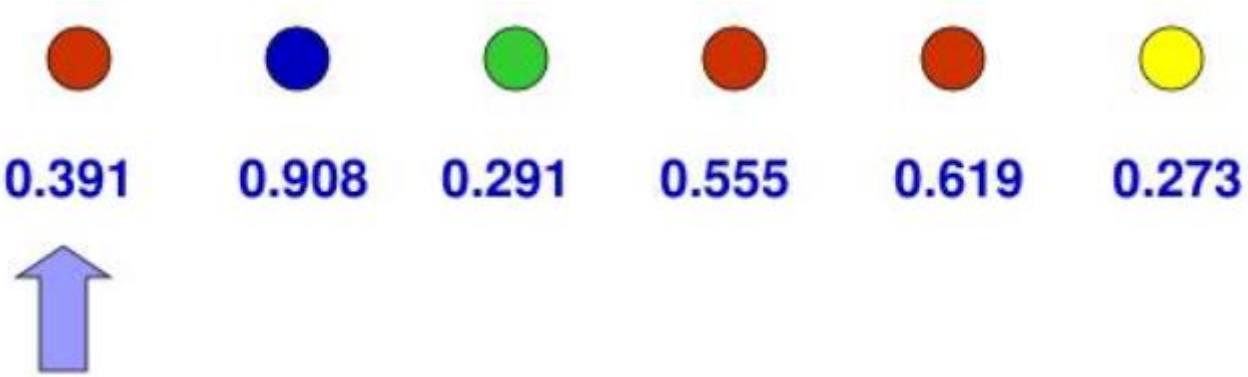


[4, 6]

- Liczba pomijanych elementów pochodzi z rozkładu geometrycznego i jest liczona w czasie stałym
- Czas działania: $O(k(1 + log(n/k)))$

Min-wise sampling

- Dla każdego elementu losujemy liczbę z przedziału [0, 1]
- Przechowujemy elementy z najmniejszą wylosowaną wartością
- Pozwala łączyć dane z wielu źródeł



[7]

Definicja problemu

- Strumień danych: s_1, s_2, ..., s_t
  - $s_i = (x_i, y_i) \in R^n \times \{1, ..., C\}$
- Modele: h_1, h_2, ..., h_t
  - $h_i = (x_i, y_i) \in R^n \times \{1, ..., C\}$
  - h_i zależy wyłącznie od h_{i-1} oraz ostatnich p instancji s_i,..., s_{i-p}
  - p jest ściśle określone

Naturalne rozszerzenia algorytmów (1)

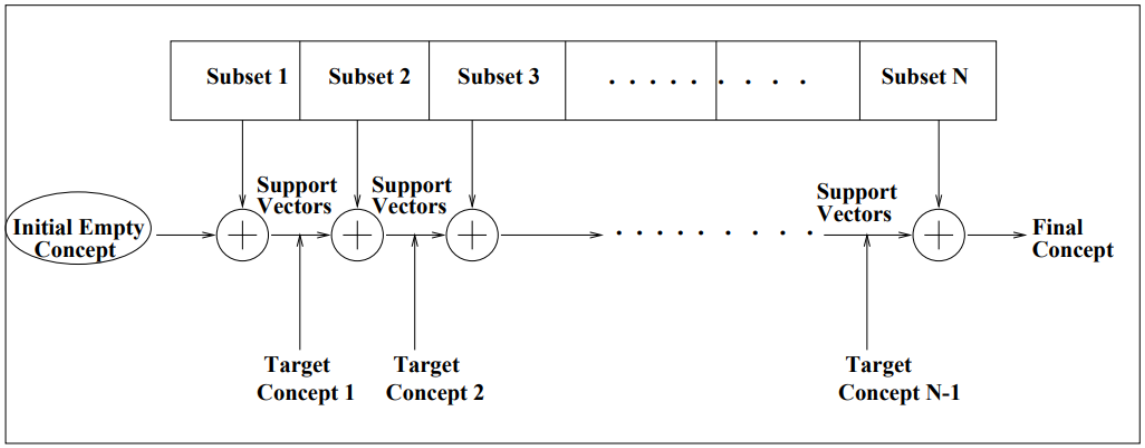
- Część algorytmów w sposób naturalny da się rozszerzyć o naukę przyrostową
  - Pierwsza wersja modelu jest efektem nauki na pierwszym batchu
  - Nauka na kolejnych batchach modyfikuje oryginalne parametry modelu
  - Waga wpływu nauki z kolejnego batcha zależna jest od:
    - Wielkości batcha
    - Numeru batcha
    - Oczekiwanej częstości napływu danych
    - Szybkości zmiany charakteru/właściwości danych
- Algorytm 1: sieć neuronowa

Naturalne rozszerzenia algorytmów (2)

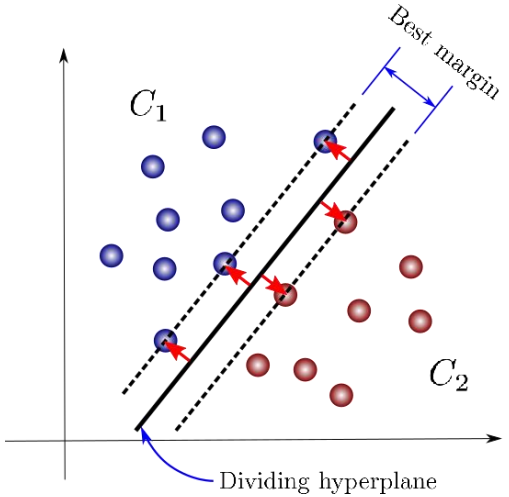
- Algorytm 2: *Stochastic gradient descent*
  - Sam w sobie ma charakter przyrostowy
- Algorytm 3: naiwny Bayes
  - Wystarczy przechowywać liczbę instancji dla każdej klasy oraz średnie wartości cech dla każdej klasy
  - Z punktu widzenia algorytmu, jest to wiedza tożsama z posiadaniem wszystkich instancji
  - Wadą jest założenie o niezależności cech

Incremental SVM

- Mała część egzemplarzy treningowych stanowi wektory nośne [8]
- W każdym kroku zapamiętywane są tylko te egzemplarze, które stanowią wektory nośne
- Nowa granica decyzyjna obliczana jest na podstawie bieżących wektorów nośnych oraz nowych egzemplarzy



[9]



[10]

Online Random Forest (1)

- Przyrostowa wersja lasów losowych
- Liczba drzew stała w całym procesie uczenia
- Gdy liczba instancji w liściu przekroczy ustalony próg, następuje podział liścia
- Tak jak w lasach losowych, przy podziale testowany jest wylosowany podzbiór cech
- Zamiast znajdywania lokalnie najlepszego podziału, testowane są losowe wartości progu (liczba testów jest z góry zadana)
  - Zasada wzięta z *Extremely randomized trees*
  - Testy zarówno po cechach jak i progach
  - Redukcja wariancji kosztem zwiększenia biasu

Online Random Forest (2)

- Oceną podziału jest współczynnik Giniego:

  $Gini(S) = 1 - \sum_{i=1}^{N} p_i^2$

  S – zbiór przykładów należących do N klas

  p_i – względna częstość występowania klasy i w S
- Popularny model ze względu na:
  - Wysoką jakość predykcji
  - Prostotę
  - Łatwość zrównoleglania
  - Niewrażliwość na różną skalę cech

Ewaluacja w trybie *offline*

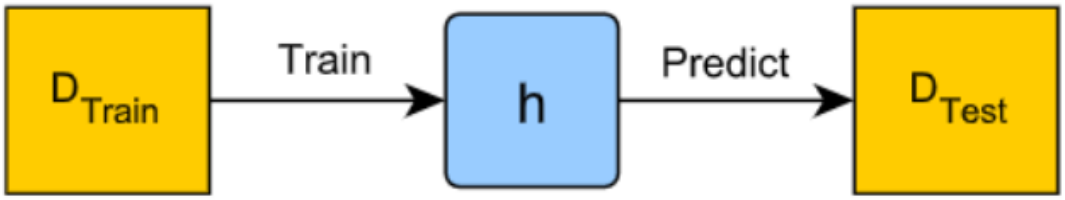


Figure 1: Classical scheme of evaluating a batch algorithm in off-line mode.

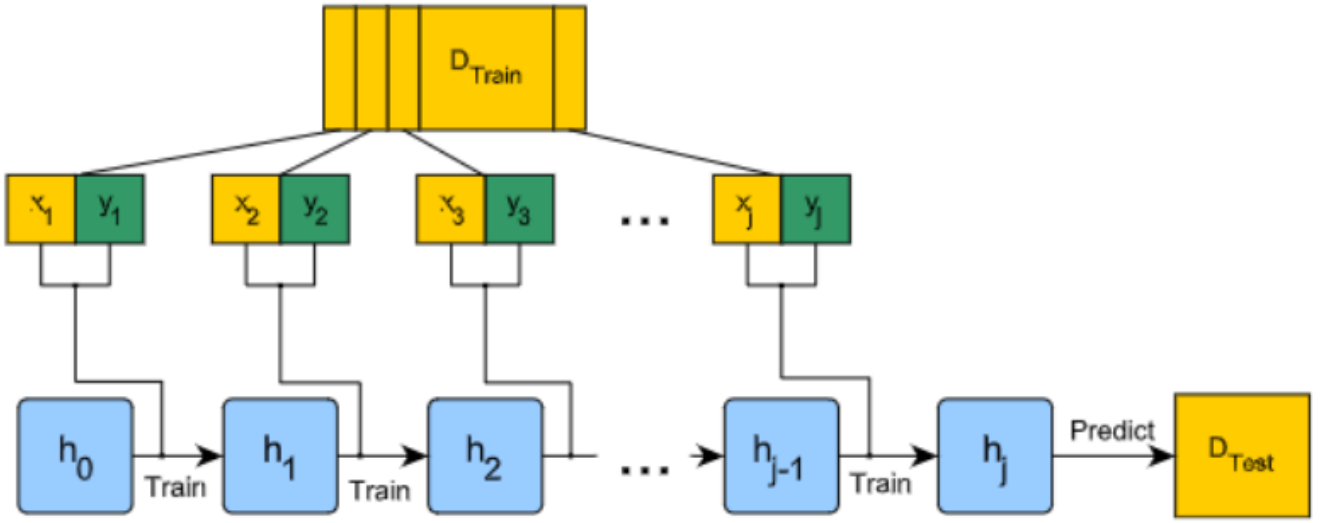


Figure 2: The process of testing an incremental algorithm in the off-line setting. Noticeably, only the last constructed model is used for prediction. All data used during training $(\mathbf{x}_i, y_i)$ is obtained from the training set D_{train}

[11]

Ewaluacja w trybie *online*

- Funkcja błędu do chwili t wyrażona jest wzorem:

$$E(S) = \frac{1}{t}\sum_{i=1}^{t} L(h_{i-1}(x_i, y_i))$$

- Wynik uzyskany w trybie *online* generalnie nie powinien być lepszy niż *online*
  - Wyjątek 1: duża zmienność danych w czasie
  - Wyjątek 2: wysoka autokorelacja i predykcja poprzedniej etykiety

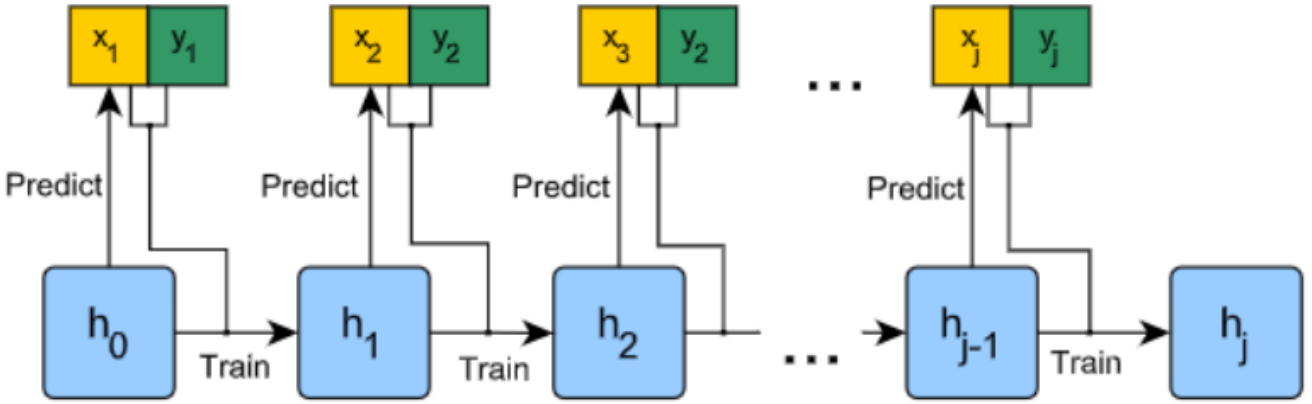
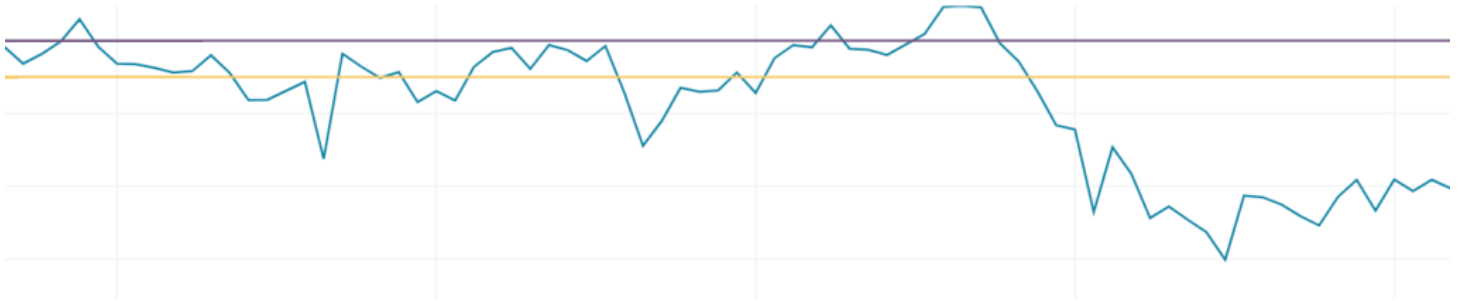


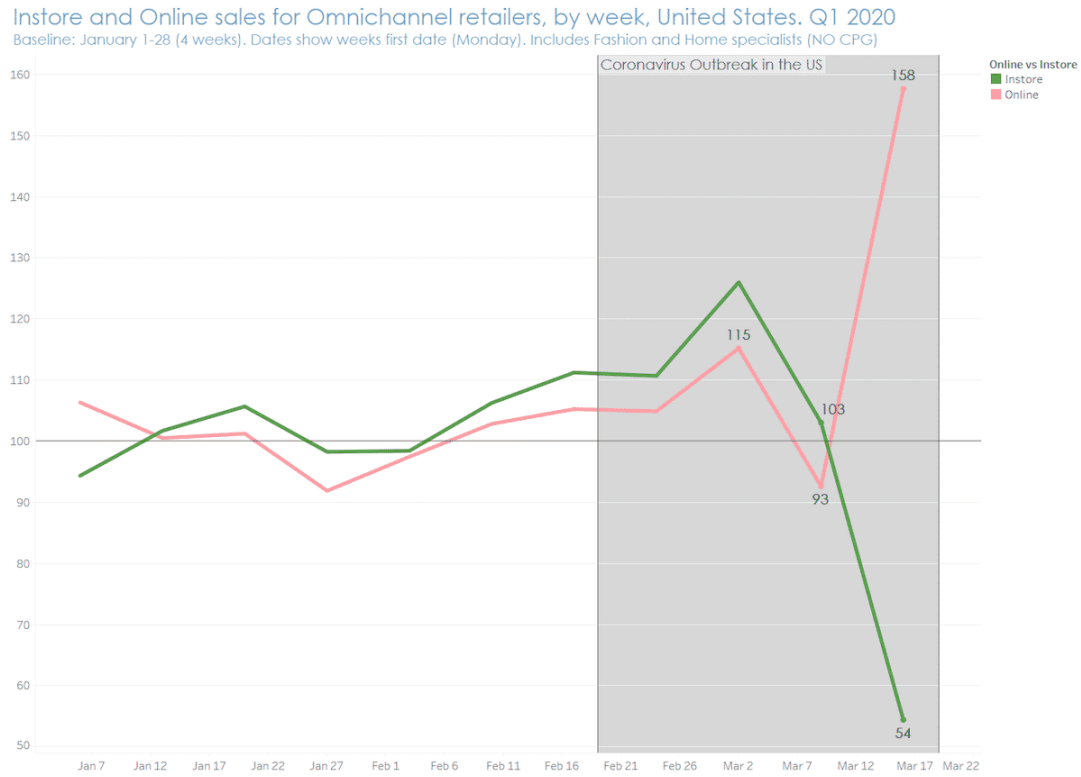
Figure 3: The online-learning scheme. Data is not split into training- and testing set. Instead, each model predicts subsequently one example, which is afterwards used for the construction of the next model.

[11]

Potrzeba douczania – przykład 1



[13]



[14]

Potrzeba douczania – przykład 2

[12]

Wyzwania

- Podatność na katastroficzne zapominanie (*catastrophic interference*)
- Niekompletność informacji spowodowana m.in. ochroną danych
- Preprocessing danych
- Wykrywanie *driftu*
- Niezbalansowanie klas (*drift* klasy mniejszościowej)
- Obsługa danych opóźnionych
- Próbkowanie wartościowych informacji (gdy przyrost danych jest zbyt duży, aby całość wykorzystać w uczeniu)
- Agregacja danych
- Ewaluacja algorytmów

Źródła (1)

1. http://chicagoanalyticsgroup.com/blog/december-20th-2016
2. https://medium.com/analytics-vidhya/data-streams-and-online-machine-learning-in-python-a382e9e8d06a
3. https://towardsdatascience.com/introduction-to-stream-mining-8b79dd64e460
4. https://en.wikipedia.org/wiki/Reservoir_sampling
5. Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, *11*(1), 37-57.
6. Li, K. H. (1994). Reservoir-sampling algorithms of time complexity O (n (1+ log (N/n))). *ACM Transactions on Mathematical Software (TOMS)*, *20*(4), 481-493.

Źródła (2)

7. https://www.slideserve.com/lyris/data-stream-algorithms-intro-sampling-entropy
8. Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.
9. Syed, N. A., Huan, S., Kah, L., & Sung, K. (1999). Incremental learning with support vector machines.
10. https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3
11. Losing, V., Hammer, B., & Wersing, H. (2018). Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, *275*, 1261-1274.
12. https://ipvm.com/reports/face-masks

Źródła (3)

13. https://doordash.engineering/2020/09/15/retraining-ml-models-covid-19/
14. https://www.criteo.com/blog/coronavirus-consumer-trends/