Metody Inżynierii Wiedzy
Sieci rekurencyjne - wykład 9

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materiały: $ftp(public) : //aszmigie/MIW$

Szeregi czasowe

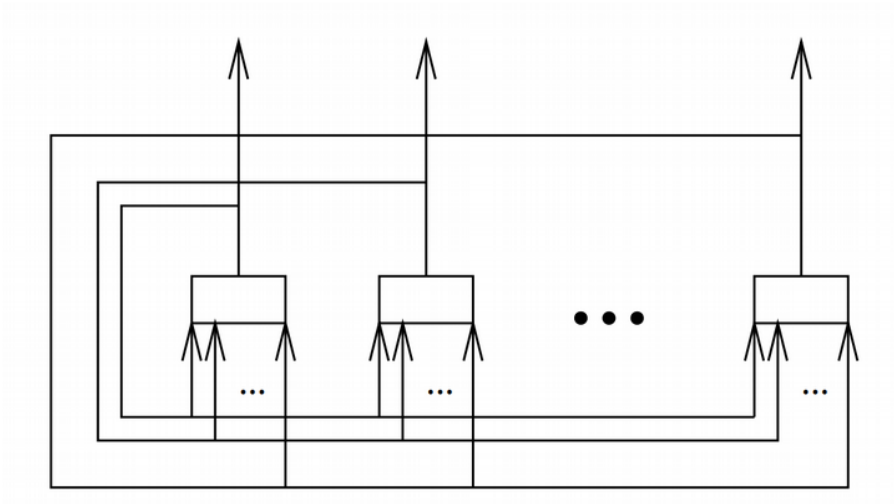
- Ciąg obserwacji pokazujący kształtowanie się badanego zjawiska w kolejnych okresach czasu (sekundach, dniach, latach, itp.),
- Uporządkowany chronologicznie zbiór wartości badanej cechy lub określonego zjawiska zaobserwowanych w różnych momentach (przedziałach) czasu,
- Ciąg obserwacji x_t zapisywanych w ściśle określonym czasie,
- Realizacja procesu stochastycznego, którego dziedziną jest czas. Proces stochastyczny definiowany jest wtedy jako ciąg zmiennych losowych indeksowanych przez czas t, a szereg czasowy jest wtedy jego pojedynczą realizacją.

Sieć Hopfilda

- Podstawowy model bazuje na jednostkach nieliniowych S_i o wartościach ze zbioru $\{-1, 1\}$, oraz skokowej funkcji aktywacji (zakładamy,że $sign(0) = 1$).
- Sieć składa się z jednej warstwy neuronów połączonych każdy z każdym za poprzez wagi w_{ij}, oznaczających połączenie jednostki j-tej do i-tej.
- ogólna postać pojedynczego neuronu i

$$S_i = sign(\sum_j w_{ij} \cdot S_j - b_i)$$

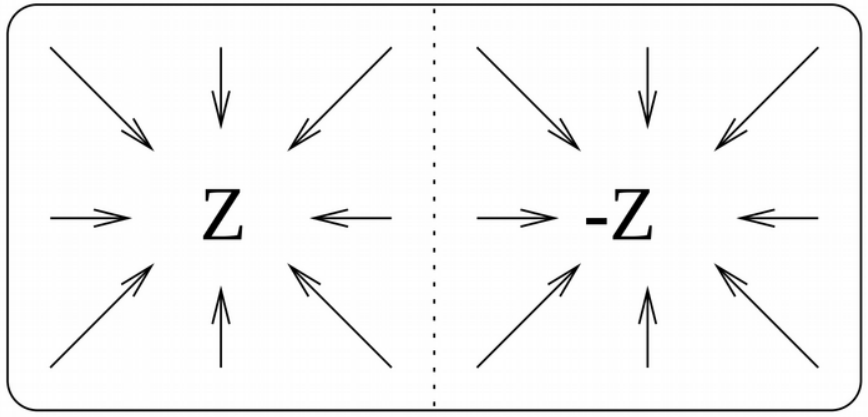
Struktura sieci Hopfielda



- Można wyróżnić sztuczne wejście zwane biasem b_i wówczas model neuronu przyjmie postać:

$$S_i = sign(\sum_j w_{ij} \cdot S_j)$$

Zapamiętane wzorce - przypadek jednego wzorca



- Możemy sobie wyobrazić, że wzorzec Z stanowi punkt przyciągania (atraktor) w przestrzeni stanów sieci.
- Oprócz tego jest jeszcze jeden atraktor, a mianowicie taki w którym wszystkie stany są odwrócone $-Z$

Przypadek wielu wzorców

- Przypuśćmy, że mamy p wzorców losowych i oznaczmy je $Z_1, Z_2, \ldots Z_p$ Najprostszym sposobem, jaki można zastosować do ustalenia odpowiednich współczynników, jest superpozycja składników dla pojedynczych wzorców.
- Dla N neuronów i p wzorców wagi w_{ij} określa się jako:

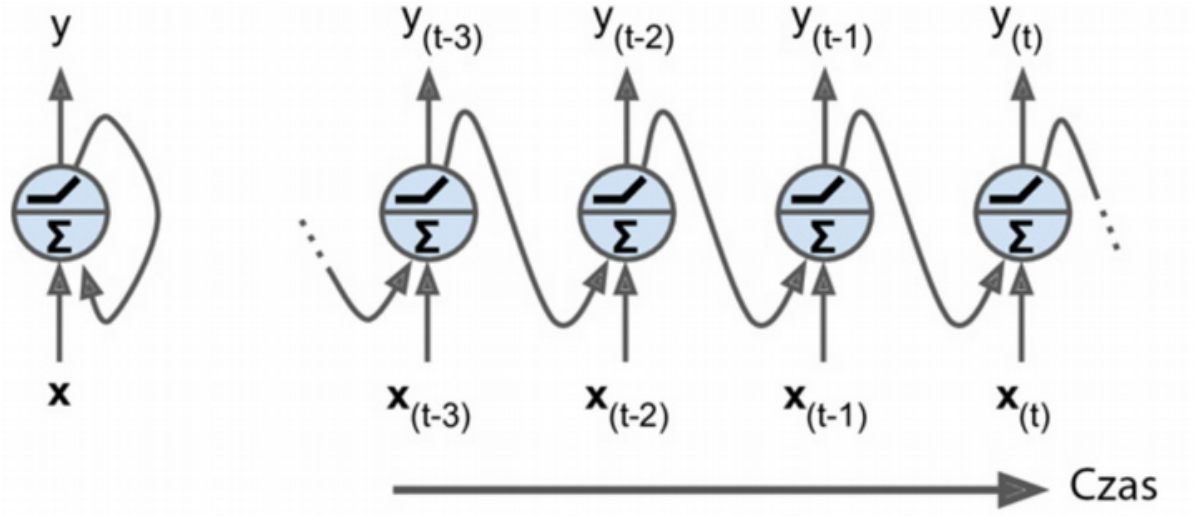
$$w_{ij} = \frac{1}{N} \sum_{m=1}^{p} Z_i^m \cdot Z_J^m$$

- Wzór ten nazywany jest “regułą Heba” lub “uogólnioną regułą Heba” ponieważ zmiany wag są proporcjonalne do korelacji aktywności presynaptycznej i postsynaptycznej. neuron

Pamięć asocjacyjna

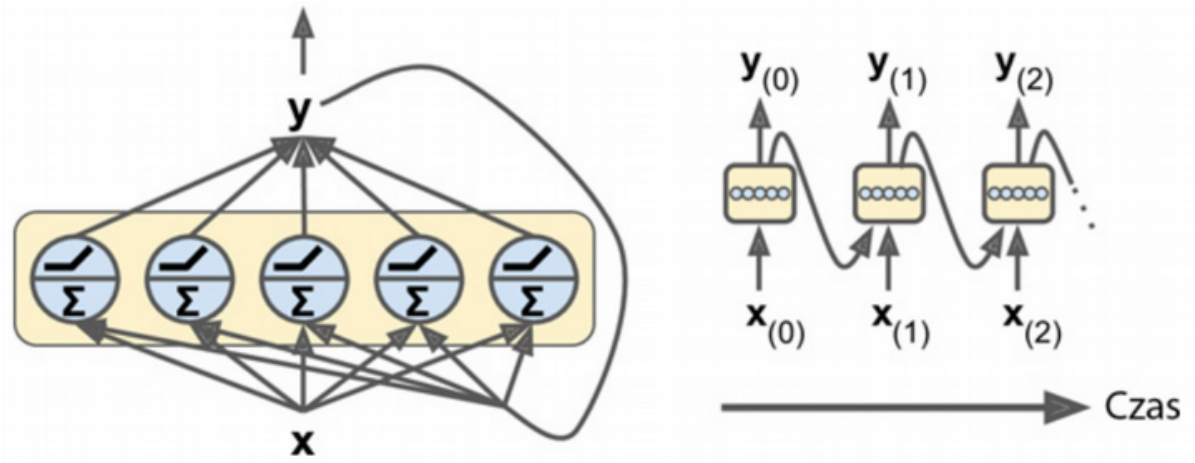
- zdolność sieci do prawidłowego zinterpretowania danych zniekształconych lub niekompletnych (zakłócenia i zniekształcenia)
- sieć może być skuteczna również wtedy, gdy stopień "zaszumienia" sygnału wejściowego wyklucza praktyczne użycie jakichkolwiek innych metod filtracji,
- Optymalizacja – rozwiązywanie trudnych problemów optymalizacyjnych,

Neuron rekurencyjny



- W każdym takcie, zwanym również ramką (ang. frame), neuron rekurencyjny (ang. recurrent neuron) oprócz danych wejściowych $x(t)$ otrzymuje również własne wyniki z poprzedniego taktu, $y(t-1)$.
- Możemy przedstawić działanie tej mikroskopijnej sieci na osi czasu. Proces ten nazywamy rozwijaniem sieci w czasie (ang. unrolling the network through time).

Warstwa neuronów rekurencyjnych



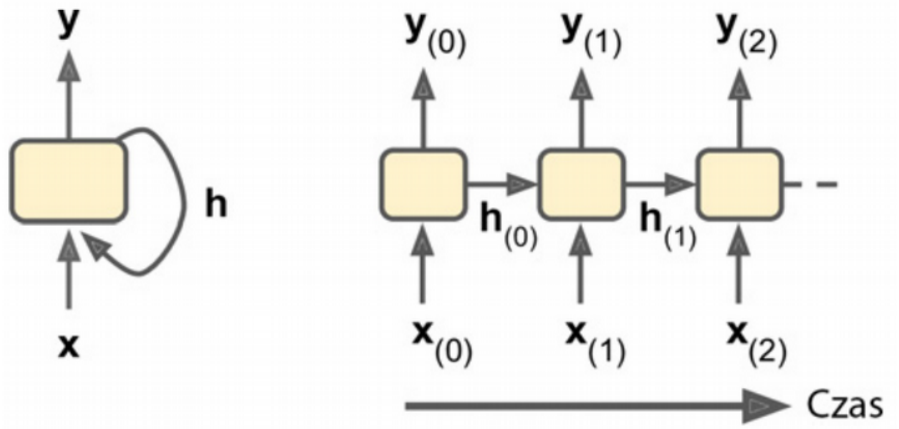
- W każdym takcie t neurony otrzymują na wejściu wektor wejściowy $x(t)$ i wektor wyjściowy z poprzedniego kroku $y(t-1)$,
- Na wejściu, jak i na wyjściu stosowane są wektory,
- Wagi w_x są przeznaczone dla danych wejściowych $x(t)$, a wagi w_y dla danych wyjściowych z poprzedniego taktu $y(t-1)$,
- Dla warstwy będą to macierze W_x i W_y.

Opis warstwy neuronów rekurencyjnych

$$Y_t = \Phi(X_t \cdot W_x + Y_{t-1} \cdot W_y + b) = \Phi([X_t Y_{t-1}] \cdot \begin{bmatrix} W_x \\ W_y \end{bmatrix} + b)$$

- $Y(t)$ — macierz $m \times n$ - m liczbę próbek w warstwie, n liczba neuronów,
- $X(t)$ — macierz $m \times n$ dane wejściowe dla wszystkich przykładów,
- W_x — macierz $n_{wejścia} \times n_{neurony}$ - wagi połączeń dla wejść,
- W_y — macierz $n_{neurony} \times n_{neurony}$ - wagi połączeń dla danych wyjściowych z poprzedniego taktu,
- b — wektor biasów,
- Macierze wag W_x i W_y tworzą macierz o wymiarach $(n_{wejścia} + n_{neurony}) \times n_{neurony}$
- $[X_t Y_{t-1}]$ poziome połączenie macierzy X_t i Y_{t-1}.

Komórki pamięci

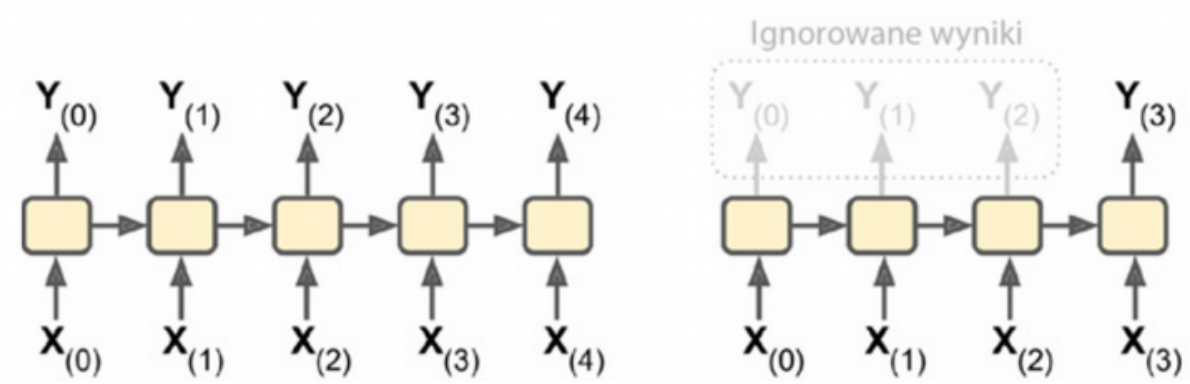


- Wyjście neuronu rekurencyjnego w takcie t stanowi funkcję wszystkich danych wejściowych z poprzednich ramek czasowych (efekt pamięci).
- Fragment sieci neuronowej zachowujący informacje o stanie nazywamy *komórką pamięci* (ang. memory cell)
- Stan komórki w takcie t, oznaczany symbolem $h(t)$ (ang. hidden):

$$h(t) = f(h(t-1), x(t))$$

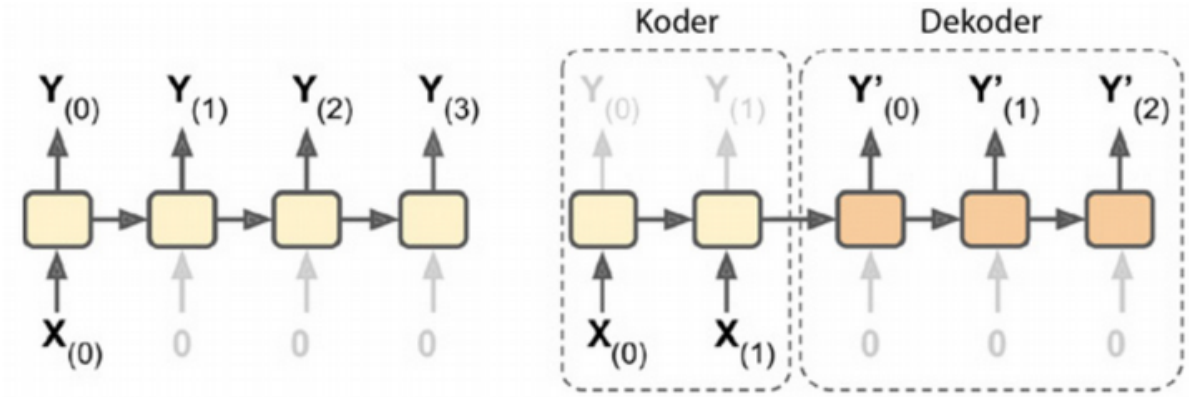
- Wyjście $y(t)$ w kroku t zależy od stany i bieżących danych wejściowych.

Sekwencje wejść i wyjść



- Sieć rekursyjna może jednocześnie pobierać sekwencję danych wejściowych i generować za ich pomocą wyniki na wyjściach,
- Taki rodzaj sieci przydaje się do prognozowania danych szeregów czasowych,
- Możemy przekazać sieci sekwencję danych wejściowych i ignorować wszystkie wyniki oprócz najnowszego (sieć sekwencyjno-wektorowa).
- Możemy dostarczyć pojedynczą daną w pierwszym takcie i pozwolić sieci generować sekwencję (sieć wektorowo-sekwencyjna).

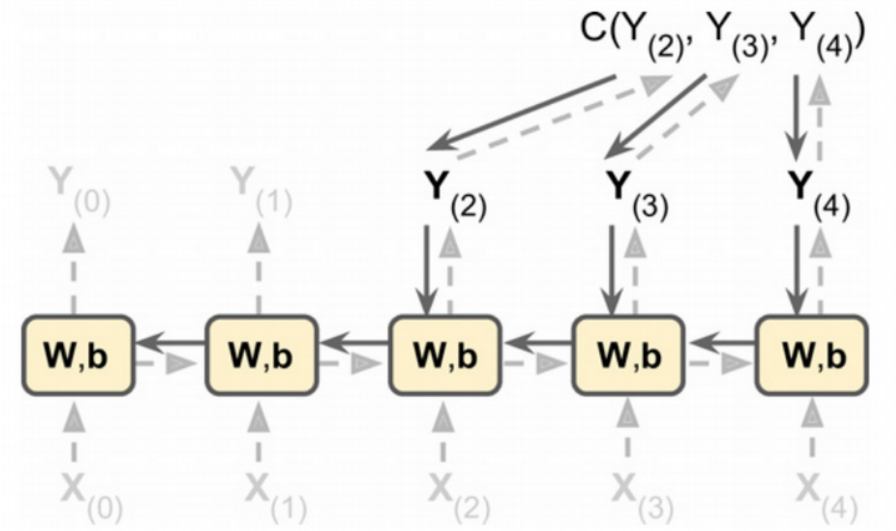
Kodery i dekodery



Możemy łączyć obie architektury sieci:

- Sieć sekwencyjno-wektorową (koder ang. *encoder*), można przyłączyć z siecią wektorowo-sekwencyjna (dekoder ang. *decoder*),
- Takie rozwiązanie może wykorzystać do tłumaczenia zdań,
- Koder przekształca zdanie do postaci wektorowej, natomiast dekoder przekształca ten wektor w przetłumaczone zdanie,
- Taki dwuetapowy model, zwany koderem-dekoderem, sprawuje się znacznie lepiej niż zwykła sekwencyjna sieć RSN.

Uczenie rekurencyjnych sieci neuronowych



- Sieć należy ją rozwinąć w czasie, a następnie zastosować wobec niej algorytm propagacji wstecznej (propagacją wsteczną w czasie ang. *backpropagation through time* — BPTT).
- najpierw wykonywany jest przebieg do przodu poprzez rozwiniętą sieć a następnie sekwencja wyjściowa jest oceniana za pomocą funkcji kosztu

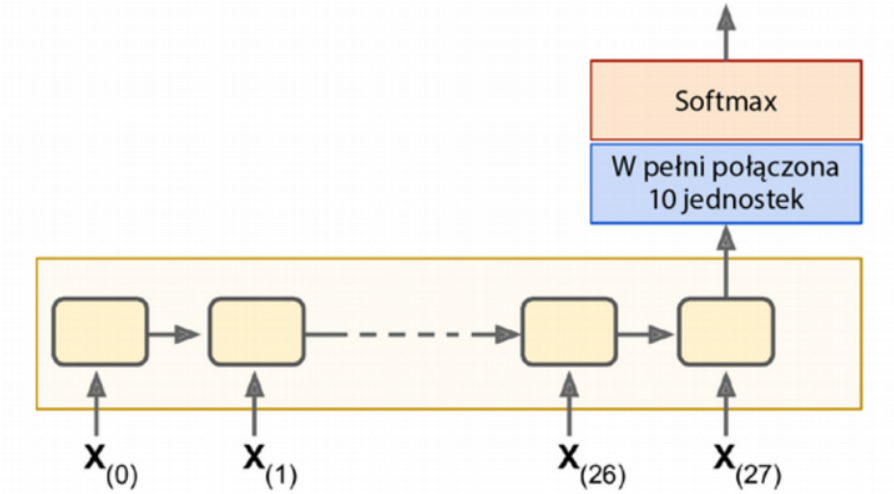
$$C(Y_{t_{min}}, Y_{t_{min}+1}, \ldots, Y_{t_{max}}$$

gdzie t_min i t_max są, odpowiednio, pierwszym i ostatnim taktem, a

gradienty funkcji kosztu zostają wstecznie rozprowadzone po całej rozwiniętej sieci,

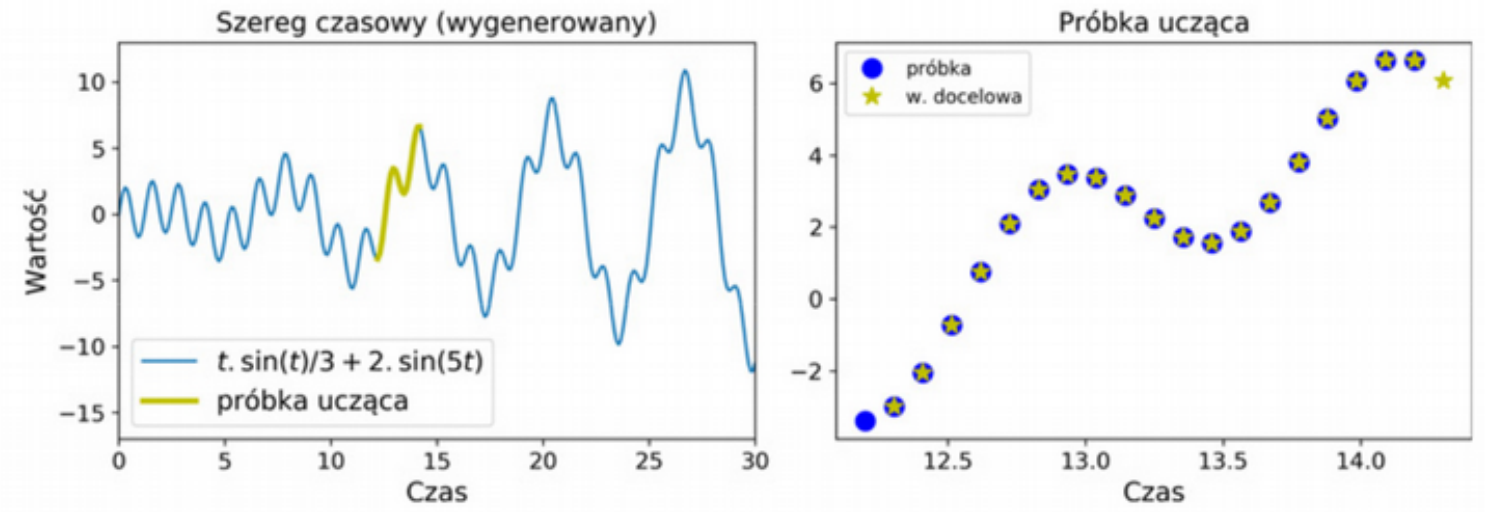
- Na koniec parametry modelu są aktualizowane za pomocą gradientów obliczonych na etapie BPTT.
- Gradienty są rozprowadzane przez wszystkie wyjścia wykorzystane przez funkcję kosztu, a nie wyłącznie przez ostateczne wyjście
- Funkcja kosztu jest obliczana przy użyciu ostatnich trzech wyjść sieci, $Y_(2)$, $Y_(3)$ i $Y_(4)$, zatem gradienty będą rozprzestrzeniały się przez te trzy wyjścia, ale już nie przez wyjścia $Y_(0)$ i $Y_(1)$.
- Te same parametry W i b są wykorzystywane w każdym takcie, propagacja wsteczna zsumuje wszystkie ramki czasowe.

Uczenie klasyfikatora sekwencji



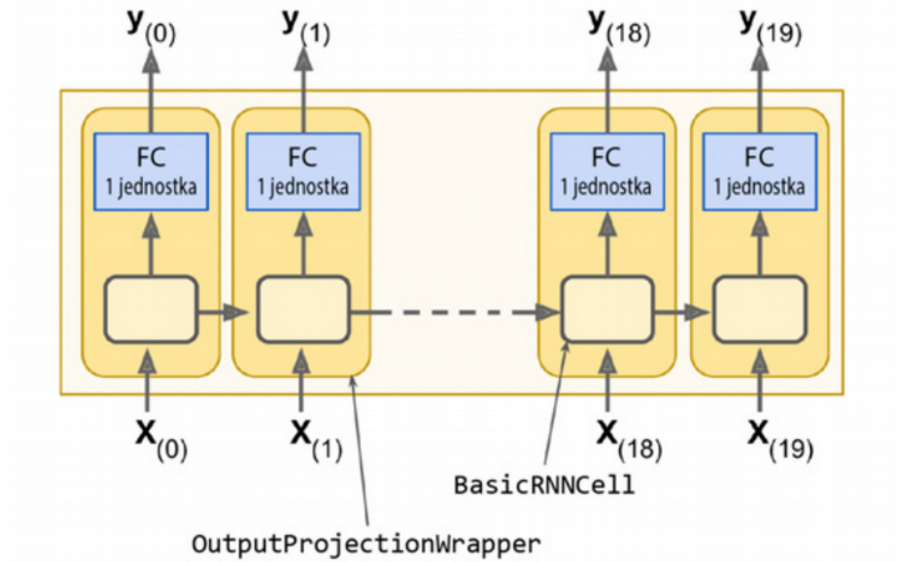
- Każdy przykład stanowi sekwencję 28 pikseli (gdyż każdy obraz ma rozmiar 28×28 pikseli),
- Komórka zawiera 150 neuronów rekurencyjnych oraz warstwę 10 neuronów podłączonych do wyjścia w ostatnim takcie, a do tego wstawimy także funkcję aktywacji *softmax*,
- Warstwy ukryte zostają zastąpione rozwiniętą siecią rekurencyjną,
- 28 wyjście przechowuje wyłącznie ostateczny stan sieci.

Uczenie w celu przewidywania szeregów czasowych



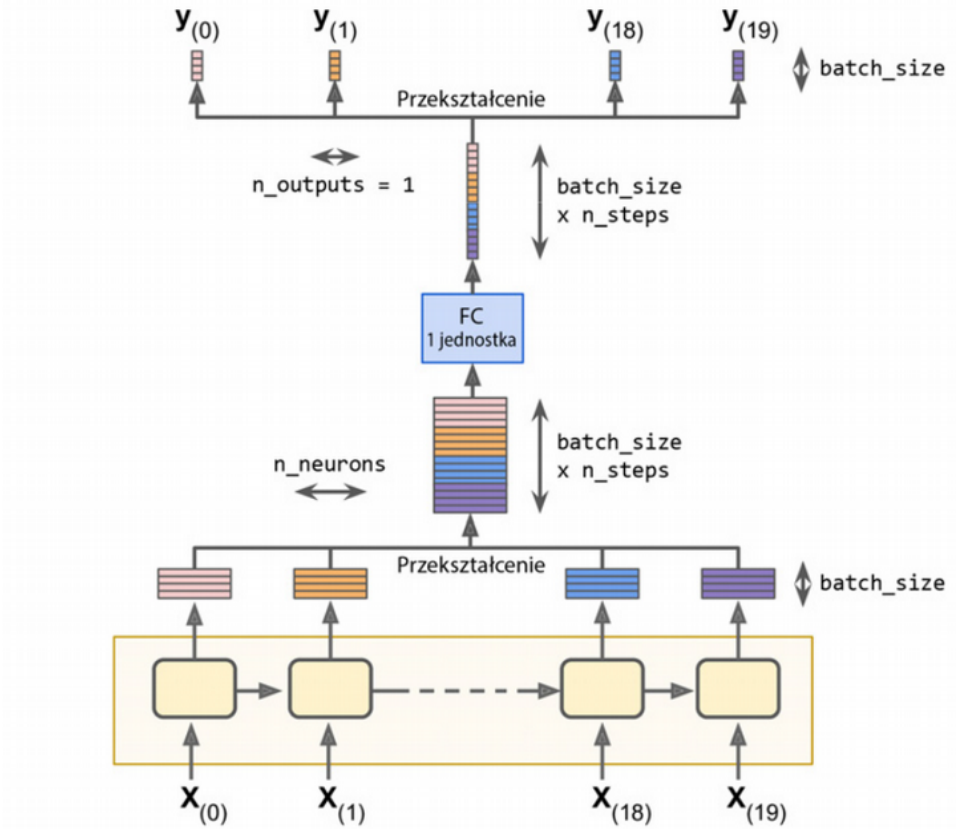
- Każda próbka ucząca stanowi losowo dobraną sekwencję 20 kolejnych wartości szeregu czasowego, natomiast sekwencja docelowa jest przesunięta o jeden takt w przyszłość,
- Sieć należy rozwijać przez 20 taktów (każda próbka ucząca będzie składać się z dwudziestu danych wejściowych).
- Przykłady docelowe również stanowią sekwencję 20 danych wejściowych, z których każda zawiera po 1 wartości.

Komórki RSN wykorzystujące rzutowanie wyników



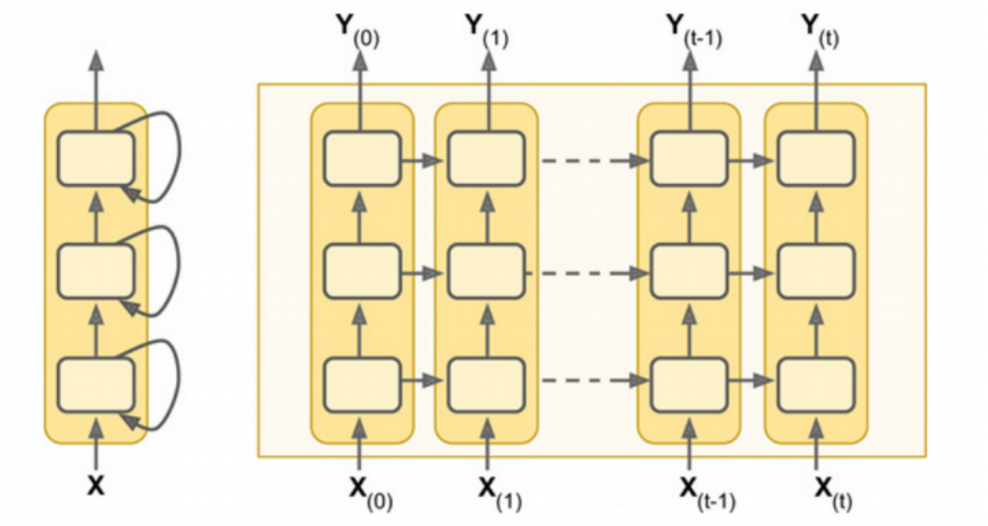
- Komórka *BasicRNNCell* umieszczona jest w węźle opakowującym *OutputProjectionWrapper* - zachowuje się ona jak zwykła komórka przesyłająca,
- Węzeł *OutputProjectionWrapper* dodaje warstwę w pełni połączonych neuronów liniowych na szczycie każdego wyjścia,
- Wszystkie w pełni połączone warstwy wykorzystują te same (modyfikowalne) wagi i człony obciążenia.

Redukcja wymiarowości sekwencji wyjściowych sieci



- Gromadzimy wszystkie dane wyjściowe, stosujemy rzutowanie, po czym rozdzielamy utworzony stos
- Wprowadzamy jedną w pełni połączoną warstwę na całość, a nie oddzielnie dla każdego taktu (przyśpieszenie obliczeń).

Głębokie sieci rekurencyjne



- Popularnym rozwiązaniem jest tworzenie stosów składających się z wielu warstw komórek - w ten sposób otrzymujemy rekurencyjną sieć głęboką (ang. *deep RNN*).
- Aby zaimplementować głęboką sieć rekurencyjną, możemy stworzyć kilka komórek podstawowych i umieścić je w węźle.

Wprowadzanie metody porzucania

- Bardzo głębokie sieci RSN mogą ulegać przetrenowaniu. Aby temu zapobiec, często jest wprowadzana metoda porzucania
- Możemy umieścić warstwę porzucania na początku lub na końcu sieci rekurencyjnej,
- Technika porzucania powinna być używana wyłącznie na etapie trenowania).

Problem uczenia w sieciach wielotaktowych

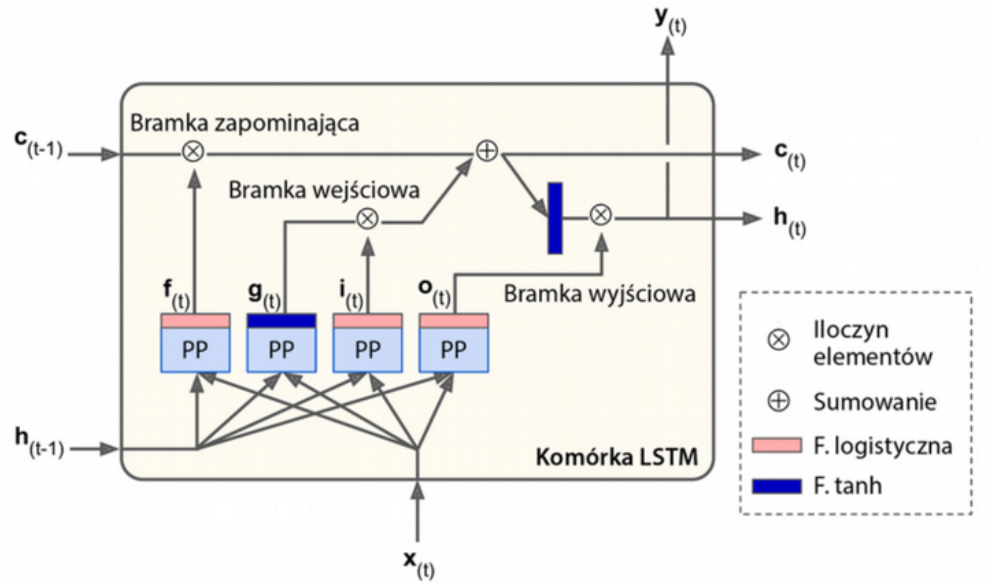
- W celu wyuczenia długich sekwencji - rozwinięta sieć w czasie okazuje się bardzo głęboka.
- Sieć głęboka narażona na problem zanikających gradientów, a proces jej uczenia może być nieskuteczny,
- Możemy skorzystać z technik redukujących wspomniany problem: dobra inicjacja parametrów, nienasycające funkcje aktywacji (np. ReLU), normalizacja wsadowa, obcinanie gradientów czy szybsze optymalizatory.
- Najprostszym i najpopularniejszym rozwiązaniem jest rozwijanie sieci jedynie na ograniczoną liczbę taktów podczas uczenia (ograniczoną propagacją wsteczną w czasie (ang. *truncated backpropagation through time*).
- Skróceniu taktów można zaradzić tak, żeby skrócone sekwencje zawierały zarówno stare, jak i świeże dane,

- W wyniku przekształceń danych zachodzących podczas kolejnych taktów po każdym takcie następuje utrata części informacji.
- Po pewnym czasie stan sieci RSN nie zawiera w sobie praktycznie żadnych śladów danych wejściowych.
- Receptą na ten problem są zaprojektowane różne typy komórek pamięci długoterminowej.
- Komórki pamięci długoterminowej okazały się tak skuteczne, że od czasu ich wprowadzenia podstawowe komórki rekurencyjne praktycznie przestały być używane.

Komórka LSTM

- Jeżeli wyobrazimy sobie komórkę LSTM jako czarną skrzynkę to że przypomina ona bardzo komórkę podstawową,
- Komórka LSTM jest znacznie wydajniejszą - szybciej uzyskuje zbieżność oraz jest w stanie wykrywać w danych długoterminowe zależności.
- Komórki LSTM zarządzają dwoma wektorami stanów, które są domyślnie oddzielne w celu zwiększenia wydajności. Wektor $h(t)$ możemy interpretować jako stan krótkotrwały, a $c(t)$ — długoterminowy.
- Komórka LSTM jest w stanie nauczyć się rozpoznawać ważne dane na wejściu (zadanie bramki wejściowej), przechowywać je w stanie długoterminowym tak długo, dopóki są potrzebne (rola bramki zapominającej),

Budowa komórka LSTM

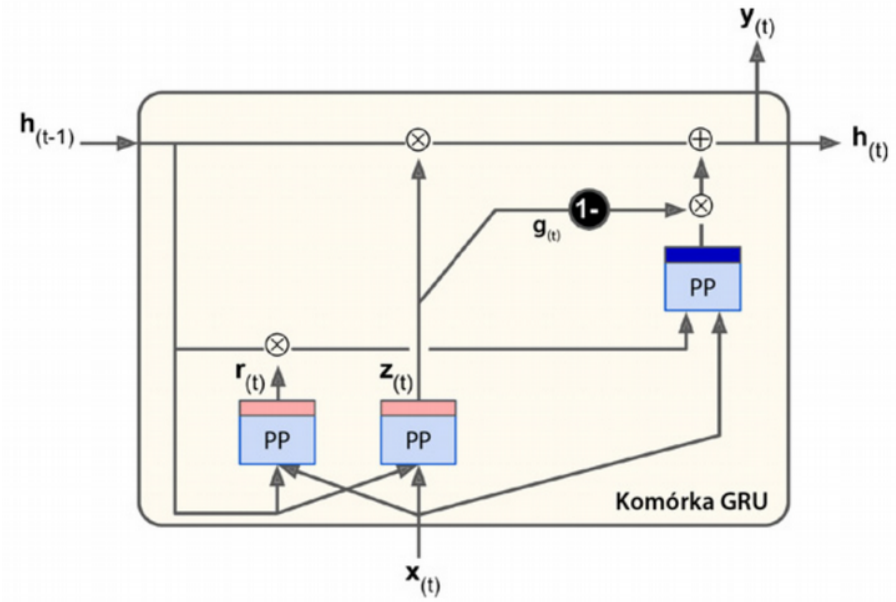


Do czterech różnych, w pełni połączonych warstw (PP — pełne połączenie) zostają przekazane bieżący wektor wejściowy $x(t)$ i poprzedni krótkotrwały wektor stanu $h(t-1)$. Każda z tych warstw pełni inną funkcję:

- **Stan długoterminowy** $c_{(t-1)}$ porusza się w sieci w prawo, widzimy zatem, że najpierw przechodzi przez bramkę zapominającą (ang. *forget gate*) (dodawane są wspomnienia wyselekcjonowane poprzez bramkę wejściową — ang. *input gate*).

- **Stan wynikowy** $c_{(t)}$ jest przesyłany dalej bez wprowadzania jakichkolwiek przekształceń. Po operacji sumowania stan długoterminowy zostaje skopiowany i przepuszczony przez funkcję tangensa hiperbolicznego. Rezultat jest filtrowany przez bramkę wyjściową - w ten sposób uzyskujemy stan krótkotrwały $h_{(t)}$
- Trzy pozostałe warstwy są kontrolerami bramek (ang. *gate controllers*). Korzystają one z logistycznej funkcji aktywacji (ich wyniki mieszczą się w zakresie od 0 do 1).
  - **Bramka zapominająca** (sterowana przez wektor $f_{(t)}$) określa, które składniki pamięci długoterminowej powinny zostać usunięte.
  - **Bramka wejściowa** (sterowana przez wektor $i_{(t)}$) określa, które elementy wektora $g_{(t)}$ powinny być dodawane do stanu długoterminowego,
  - **Bramka wyjściowa** (sterowana przez wektor $o_{(t)}$) określa, które składowe stanu długoterminowego powinny być odczytywane i wysyłane na wyjście w danym takcie,

Komórka GRU (ang. *Gated Recurrent Unit*)

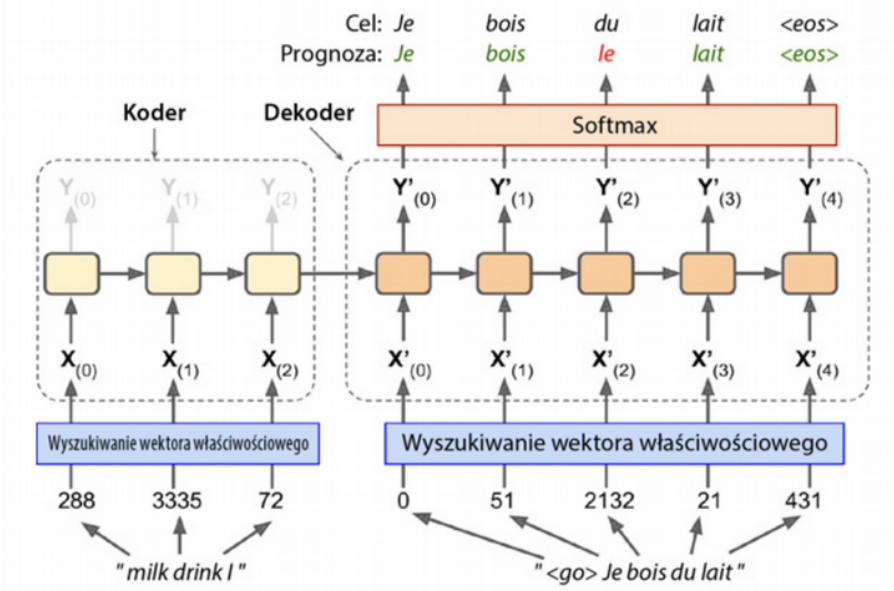


Jednostka GRU stanowi uproszczoną wersję komórki LSTM i osiąga zbliżoną wydajność:

- Obydwa wektory stanów zostają scalone w jeden wektor $h_{(t)}$,
- Bramki zapominająca i wejściowa są sterowane przez wspólny kontroler - wartość 1, to bramka wejściowa zostaje otwarta, a zapominająca — zamknięta,

- Nie ma bramki wyjściowej - w każdym takcie na wyjściu pojawia się pełny wektor stanu.
- Zostaje umieszczony dodatkowy kontroler, określający, która część poprzedniego stanu zostanie ukazana w warstwie głównej.

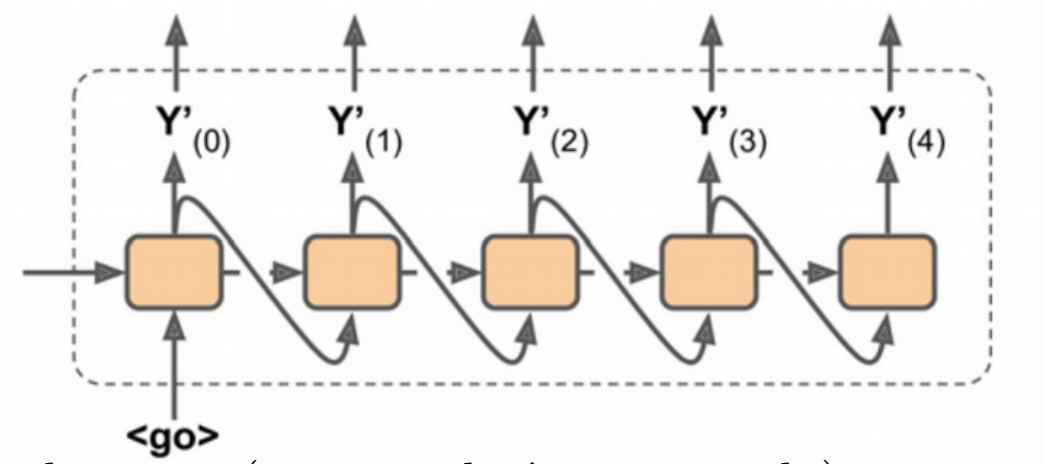
Sieć typu koder-dekoder służąca do tłumaczenia maszynowego



- Wprowadzamy do kodera angielskie zdania, a dekoder generuje przekład w języku francuskim.
- Francuskie zdania są również dostarczane na wejście dekodera, ale o jeden takt wcześniej przed angielskim zdaniem.
- Na wejściu dekodera pojawia się słowo, które powinno być wynikiem w poprzednim takcie

- Szyk angielskich zdań zostaje odwrócony przed ich wczytaniem do kodera. Na przykład zdanie “I drink milk” przyjmuje postać “milk drink I” - pierwszy wyraz zostanie wczytany jako ostatni i będzie on przetłumaczony jako pierwszy przez dekoder.
- W każdym takcie dekoder wyświetla wynik dla każdego słowa w słowniku wyjściowym (tj. francuskim), a następnie funkcja softmax przekształca te wyniki w prawdopodobieństwa.
- Wyraz o największym prawdopodobieństwie zostaje podany na wyjście. Mechanizm ten bardzo przypomina regularne zadanie klasyfikacji, dlatego możemy wyuczyć ten model za pomocą funkcji softmax

Dostarczanie słowa będącego wynikiem poprzedniego taktu jako wejścia na etapie wnioskowania



Na etapie wnioskowania (już po zakończeniu nauki) nie mamy do dyspozycji zdania docelowego, które byśmy przekazali dekoderowi. Zamiast tego wprowadzamy po prostu wyraz, który stanowił wynik w poprzednim takcie

- Wszystkie sekwencje wejściowe (do kodera i dekodera) są stałej długości. Oczywiście jednak długości te mogą się różnić.
- W przypadku dużego rozmiaru słownika wyjściowego wyliczanie

prawdopodobieństwa dla każdego dostępnego słowa jest bardzo czasochłonne.

- Możemy tego uniknąć, pozwalając dekoderowi umieszczać na wyjściu znacznie mniejsze wektory, a następnie oszacować stratę bez konieczności wyliczania wartości dla każdego wyrazu w słowniku docelowym.

Zadania na laboratoria

1. Dla notowań giełdowych spółki .....[a] zaproponuj i zrealizuj podział tych danych na *dane treningowe* i *dane testowe*,
2. Zaproponuj liniowy model autoregresyjny (model AR) biorący pod uwagę kilka wcześniejszych notowań spółki (nie więcej niż 50). Określ parametry modelu stosując metodę najmniejszych kwadratów. Zweryfikuj poprawność modelu,
3. Zaproponuj sieć neuronową rekurencyjną do predykcji notowań. Wytrenuj sieć (zastosuj znane ci techniki trenowania sieci jak porzucanie, skalowanie etc.). Zweryfikuj poprawność modelu,
4. Do predykcji notowań użyj dodatkowych danych (np. wolumen lub WIG),
5. Porównaj modele AR i rekurencyjne sieci neuronowe.

[a] spółka i okres notowań do ustalenia z prowadzącym. Historyczne dane z notowań dostępne są pod adresem *http://bossa.pl/notowania/metastock/*