Metody Inżynierii Wiedzy
Uczenie nienadzorowane i częściowo nadzorowane - wykład 10

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

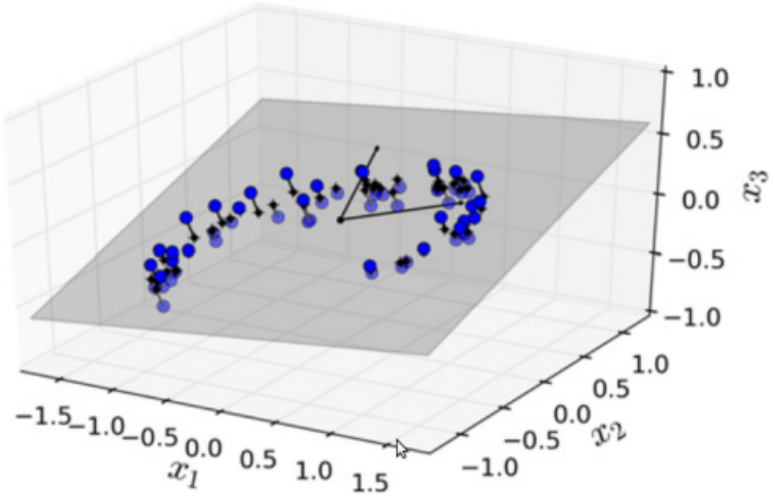
Klątwą wymiarowości (ang. curse of dimensionality).

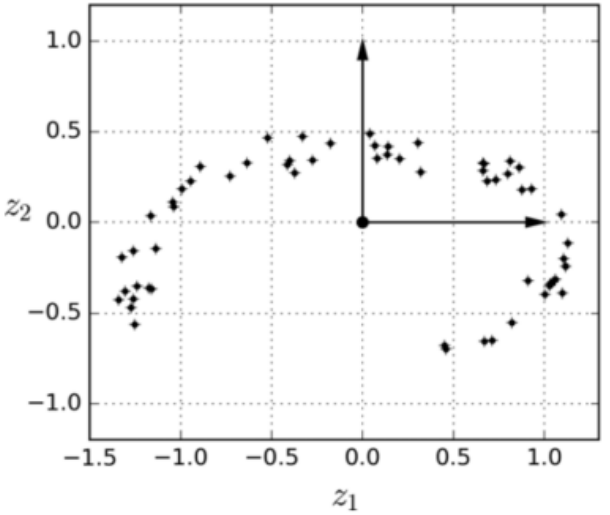
- Wiele problemów obejmuje tysiące cech opisujących każdy przykład uczący,
- Proces uczenia nie tylko przebiega bardzo powoli, ale utrudnia również znalezienie dobrego rozwiązania,
- W przypadku rzeczywistych zadań nieraz możliwe jest znaczne ograniczenie liczby cech,
- Redukcja wymiarowości doskonale nadaje się również do wizualizowania danych

Główne strategie redukcji wymiarowości

- **Rzutowanie** - W większości praktycznych problemów próbki uczące nie są równomiernie rozmieszczone we wszystkich wymiarach - wiele cech jest niemal stałych, natomiast inne są ze sobą silnie skorelowane,
- Wszystkie przykłady uczące znajdują się wewnątrz podprzestrzeni składającej się z mniejszej liczby wymiarów od pierwotnej przestrzeni cech.

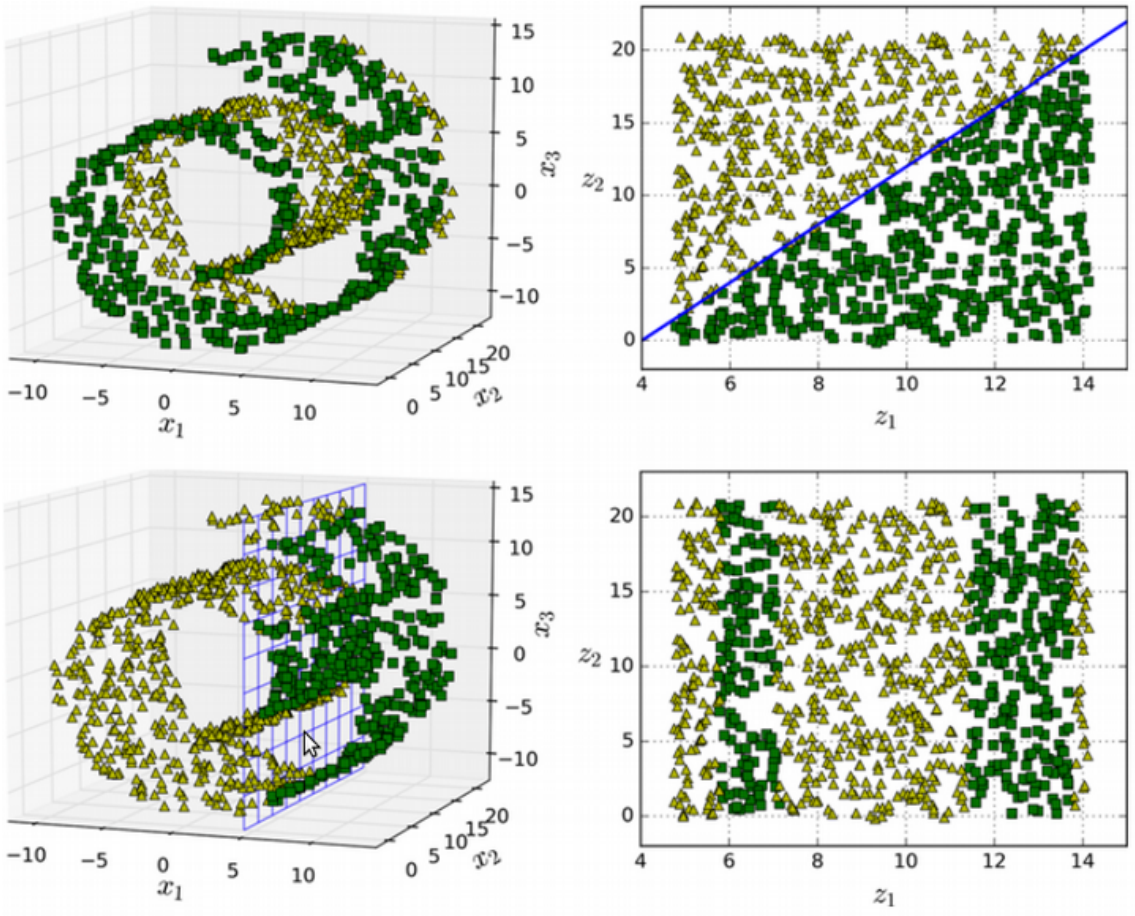
Rzutowanie





- Trójwymiarowy zbiór danych znajdujący się blisko dwuwymiarowej podprzestrzeni,
- Wszystkie przykłady uczące znajdują się blisko jednej płaszczyzny - dwuwymiarowa podprzestrzeń przestrzeni trójwymiarowej.
- Po zrzutowaniu przykładów uczących prostopadle na tę podprzestrzeń otrzymamy nowy, dwuwymiarowy zestaw danych.

Uczenie rozmaitościowe



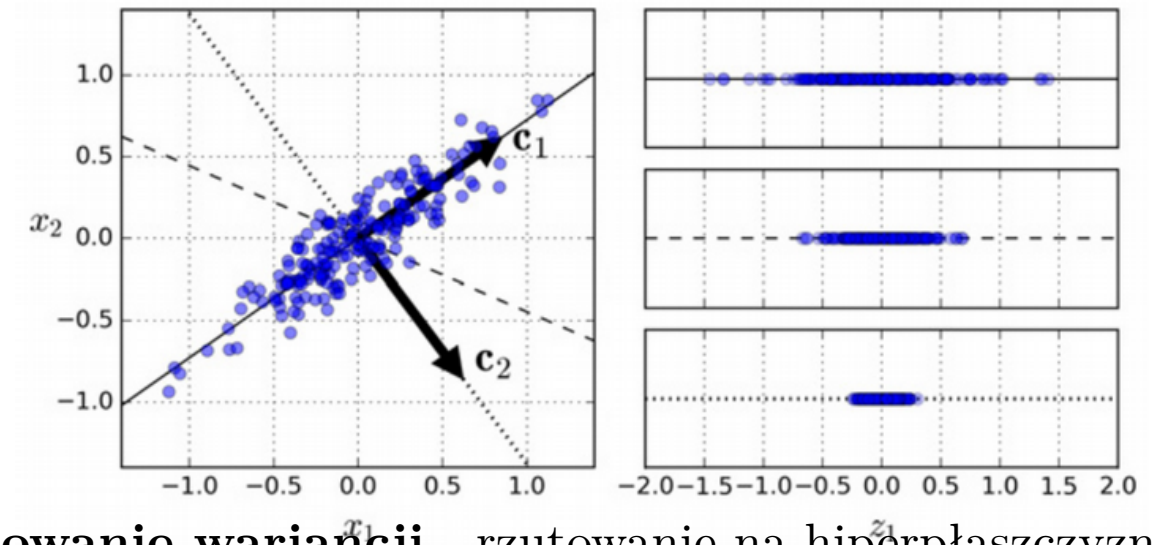
- Rzutowanie nie zawsze jest skuteczne,
- Zbiór danych Swiss roll stanowi przykład dwuwymiarowej **rozmaitości** (ang. manifold).

Uczenie rozmaitościowe

- Wiele algorytmów redukcji wymiarowości działa poprzez modelowanie rozmaitości, na której znajdują się próbki uczące; jest to proces uczenia rozmaitościowego (ang. manifold learning).
- Wykorzystywane jest w nim założenie rozmaitości (ang. manifold assumption), zwane także hipotezą rozmaitości (ang. manifold hypothesis):
- **Większość rzeczywistych, wielowymiarowych zbiorów danych znajduje się blisko rozmaitości składającej się ze znacznie mniejszej liczby wymiarów.**
- Założenie to jest bardzo często obserwowane empirycznie.

Analiza głównych składowych (ang. Principal Component Analysis — PCA)

Najpierw określa on hiperpłaszczyznę znajdującą się najbliżej danych, po czym są one na nią rzutowane.



- **Zachowanie wariancji** - rzutowanie na hiperpłaszczyznę reprezentowaną przez linię ciągłą zachowuje maksymalną wariancję,
- Hiperpłaszczyzna symbolizowana linią kropkowaną zachowuje bardzo małą wariancję,

Główne składowe

- Analiza PCA służy do wyznaczania osi zachowującej największą wartość wariancji zbioru uczącego (oznaczona linią ciągłą),
- Zostaje również znaleziona druga oś, prostopadła do pierwszej, która zachowuje największą wartość pozostałej wariancji (oznaczona linią kropkowaną).
- Wektor jednostkowy definiujący i-tą oś nosi nazwę i-tej głównej składowej (ang. principal component - PC)
- Kierunek głównych składowych nie jest stabilny - niewielka modyfikacja danych może powodować zmianę zwrotu - zazwyczaj znajdują się jednak na tych samych osiach.

Rozkładem według wartości osobliwych (ang. singular value decomposition — SVD)

która rozkłada macierz zestawu danych uczących X na iloczyn skalarny trzech macierzy:

$$X = U \cdot \Sigma \cdot V$$

gdzie V zawiera wszystkie główne składowe :

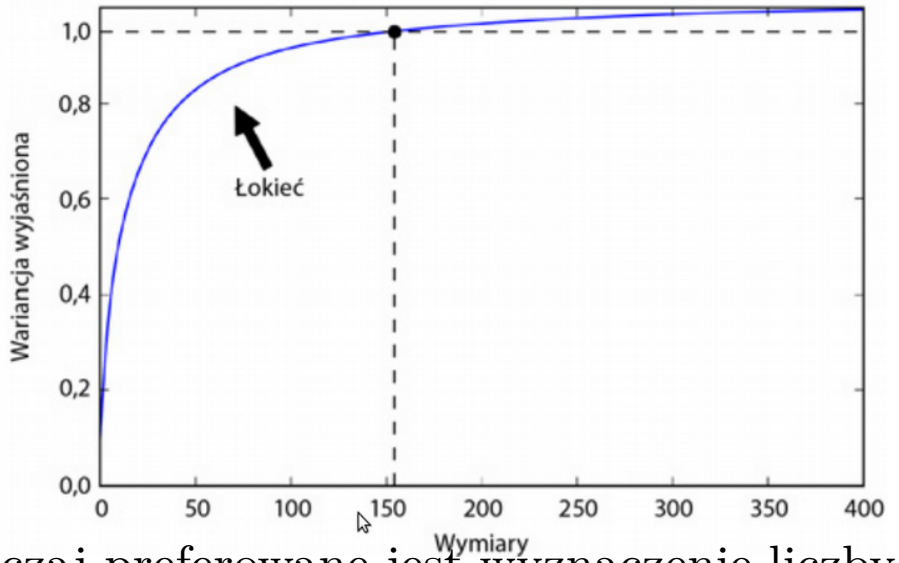
$$= \begin{bmatrix} | & | & \dots & | \\ C_1 & C_2 & \dots & C_n \\ | & | & \dots & | \end{bmatrix}$$

- Modułu NumPy posiada wbudowaną funkcję $svd()$ do wyliczania głównych składowych: $U, S, V = np.linalg.svd(X)$

Rzutowanie na d wymiarów

- Po zidentyfikowaniu wszystkich głównych składowych możemy zredukować wymiarowość zbioru danych do d wymiarów poprzez rzutowanie przykładów na hiperpłaszczyznę zdefiniowaną przez d pierwszych głównych składowych.
- Dzięki wybraniu tej hiperpłaszczyzny rzutowany zbiór danych zachowa największą możliwą wariancję oryginalnego zestawu danych

Wybór właściwej liczby wymiarów



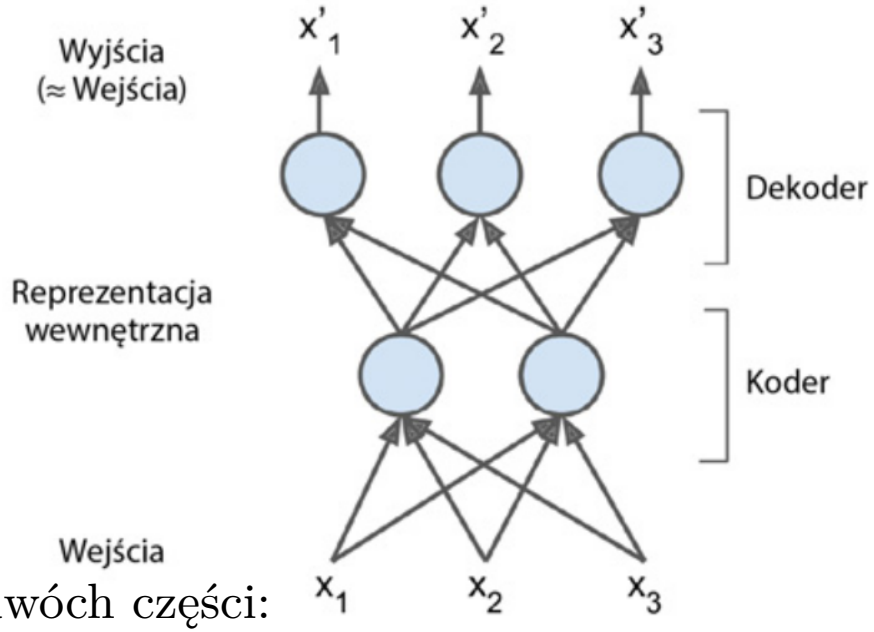
- Zazwyczaj preferowane jest wyznaczenie liczby wymiarów pozwalającej zachować wystarczająco wysoką wartość wariancji (np. 95%).
- Wygenerowanie wykresu wariancji w funkcji liczby wymiarów pozwala uchwycić zagięcie wykresu (łokieć),
- W przypadku redukcji wymiarów w celu wizualizacji najczęściej liczbę wymiarów jest ograniczana do dwóch lub trzech.

Autokodery

- sztuczne sieci neuronowe zdolne do uczenia się efektywnych reprezentacji danych wejściowych, tzw. kodowań (ang. codings), bez użycia jakiejkolwiek formy nadzorowania (zbiór uczący nie jest oznakowany etykietami).

- Kodowania te mają zazwyczaj znacznie mniejszą wymiarowość od danych wejściowych, dzięki czemu autokodery nadają się bardzo dobrze do redukowania wymiarowości
- Bardzo dobrze wykrywają cechy i mogą być używane do wstępnego, nienadzorowanego uczenia głębokich sieci neuronowych,
- Potrafią losowo generować nowe dane bardzo przypominające zbiory danych uczących - **model generatywny** (ang. generative model).

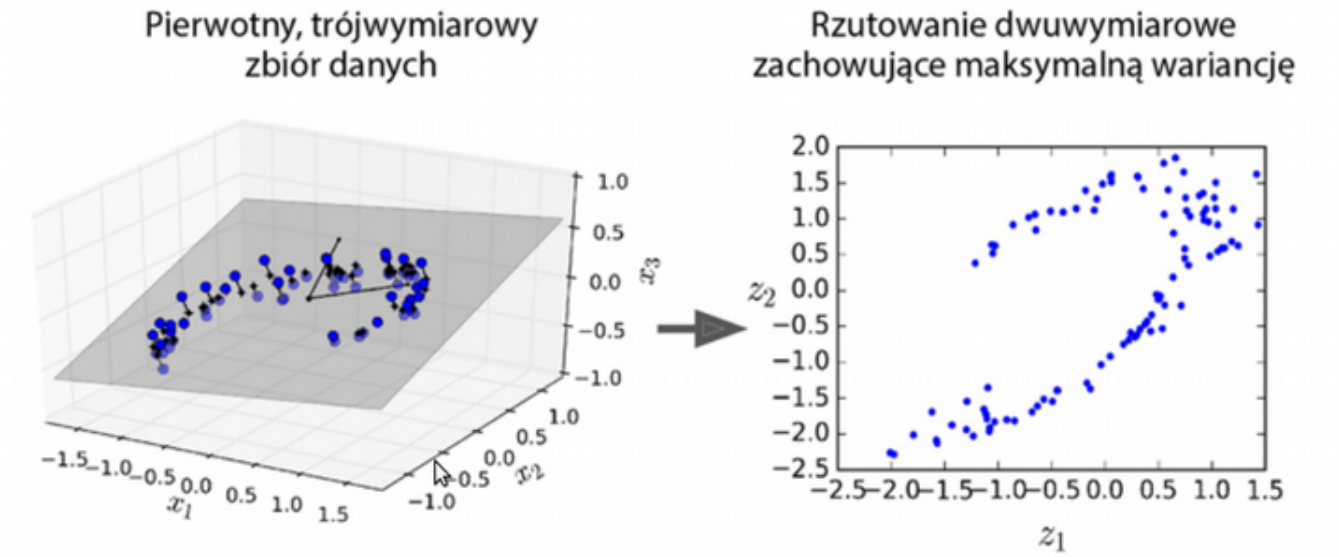
Budowa autokoder



Autokoder składa się z dwóch części:

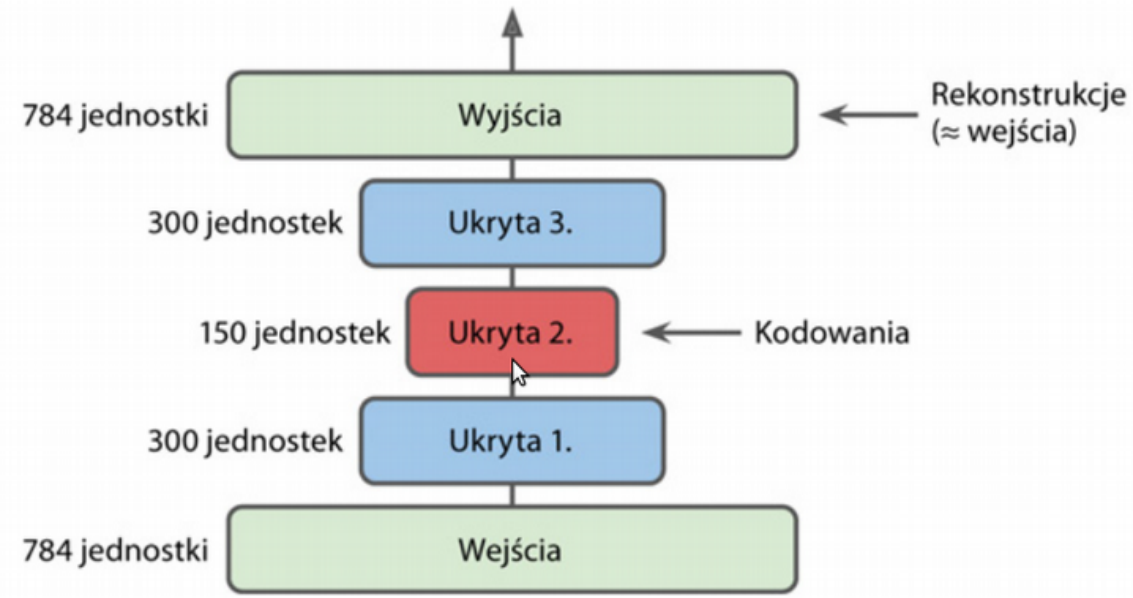
- **kodera** (ang. encoder; inna nazwa to sieć rozpoznawania — ang. recognition error), przekształcającego dane wejściowe do postaci reprezentacji wewnętrznej,
- **dekodera** (ang. decoder; ewentualnie sieć generatywna — ang. generative network), którego zadaniem jest konwersja tej reprezentacji wewnętrznej na dane wyjściowe

Analiza PCA za pomocą niedopełnionego autokodera liniowego



- Jeżeli autokoder korzysta wyłącznie z liniowych funkcji aktywacji, a funkcją kosztu jest MSE, to będzie on przeprowadzał analizę PCA

Autokodery stosowe

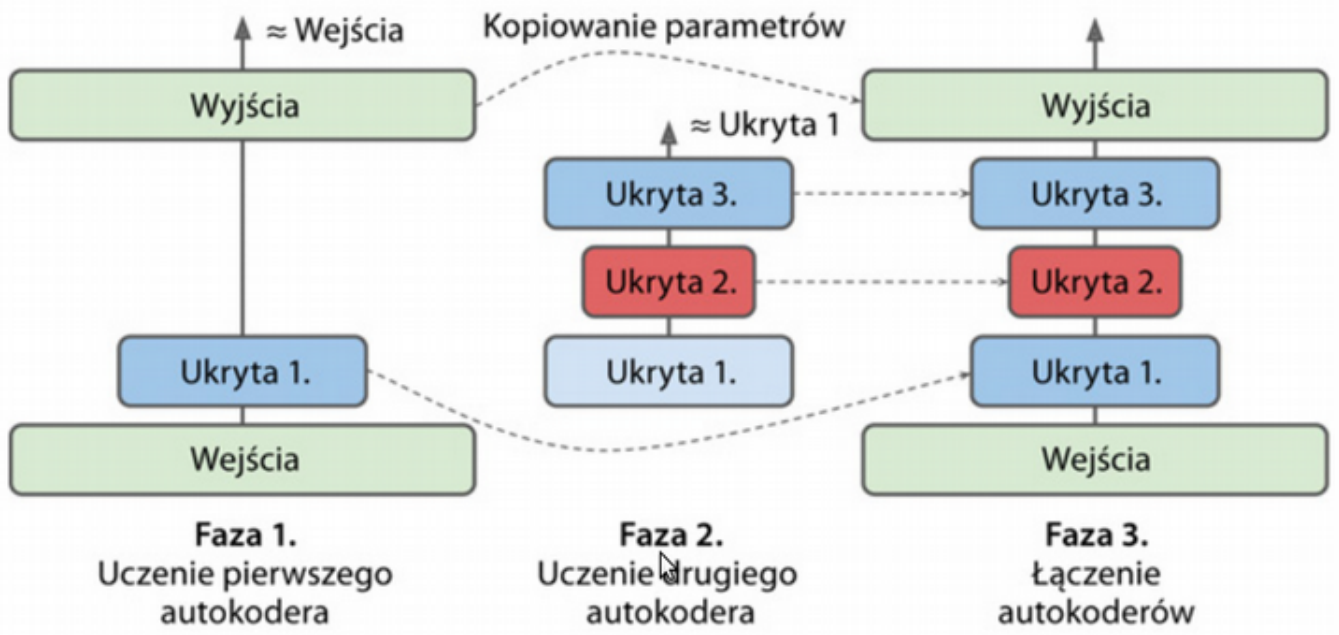


- Autokodery, które posiadają wiele warstw ukrytych są nazywane autokoderami stosowymi (ang. stacked autoencoders) lub głębokimi (ang. deep autoencoders).
- Kolejne warstwy ukryte pozwalają autokoderowi uczyć się bardziej skomplikowanych kodowań.
- Architektura autokodera stosowego najczęściej jest symetryczna względem centralnej warstwy ukrytej (kodowania).

Wiązanie wag

- W przypadku symetrycznych autokoderów popularnym rozwiązaniem jest wiązanie wag (ang. tie the weights) warstw dekodera z wagami warstw kodera,
- Redukujemy w ten sposób o połowę liczbę wag w modelu, co powoduje przyśpieszenie procesu uczenia i ograniczenie ryzyka przetrenowani,

Uczenie autokoderów pojedynczo



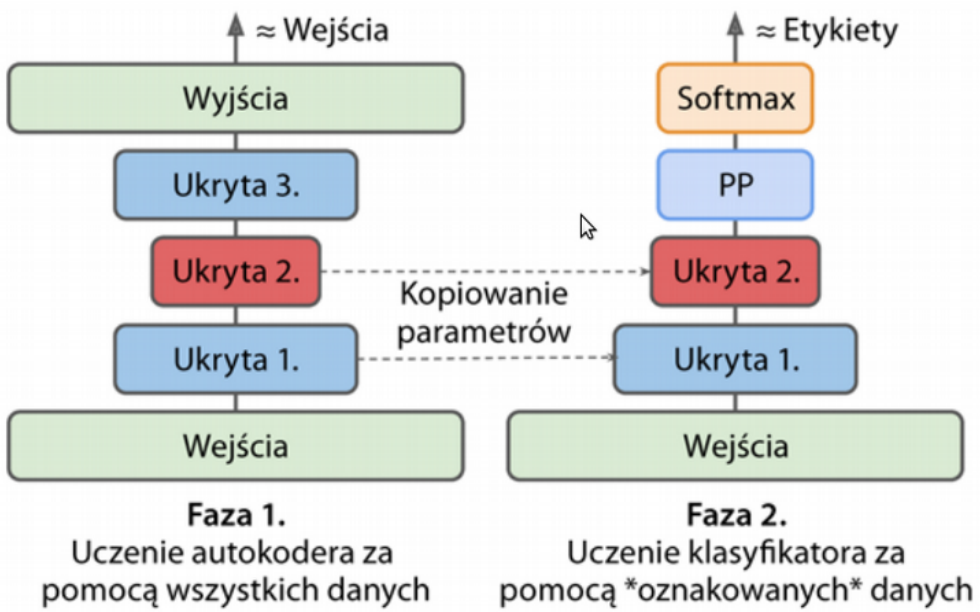
Zamiast uczyć naraz cały autokoder można trenować pojedynczo poszczególne autokodery, a następnie połącząc je w jeden autokoder stosowy

- **faza pierwsza** - pierwszy autokoder uczy się rekonstruować dane wejściowe,
- W **drugiej fazie** drugi autokoder uczy się rekonstruować wyniki

warstwy ukrytej generowane przez pierwszy autokoder,

- **Łączenie autokoderów** - najpierw łączymy warstwy ukryte każdego kodera, a następnie warstwy wyjściowe w odwrotnej kolejności.
- Można w ten sposób trenować wiele pojedynczych autokoderów i konstruować bardzo głębokie modele.

Nienadzorowane uczenie wstępne



W przypadku dużych, nieoznakowanych danych:

- Najpierw uczymy autokoder stosowy za pomocą wszystkich dostępnych danych,
- Następnie wykorzystać niższe warstwy do stworzenia sieci neuronowej przeznaczonej do określonego zadania i wytrenować ją przy użyciu oznakowanych danych.

Autokodery rzadkie

- Możemy sprawić, na przykład, żeby w warstwie kodującej występowało tylko 5% znacząco aktywnych neuronów. W ten sposób wymuszamy na autokoderze reprezentowanie każdego wejścia jako kombinacji niewielkiej liczby aktywacji.
- Dzięki temu każdy neuron warstwy kodowania zazwyczaj uczy się wykrywać jakąś przydatną cechę
- Aby były faworyzowane modele rzadkie, musimy najpierw zmierzyć rzeczywistą rzadkość warstwy kodowania w każdym przebiegu uczenia,

Dywergencja Kullbacka-Leiblera jako miara rzadkości

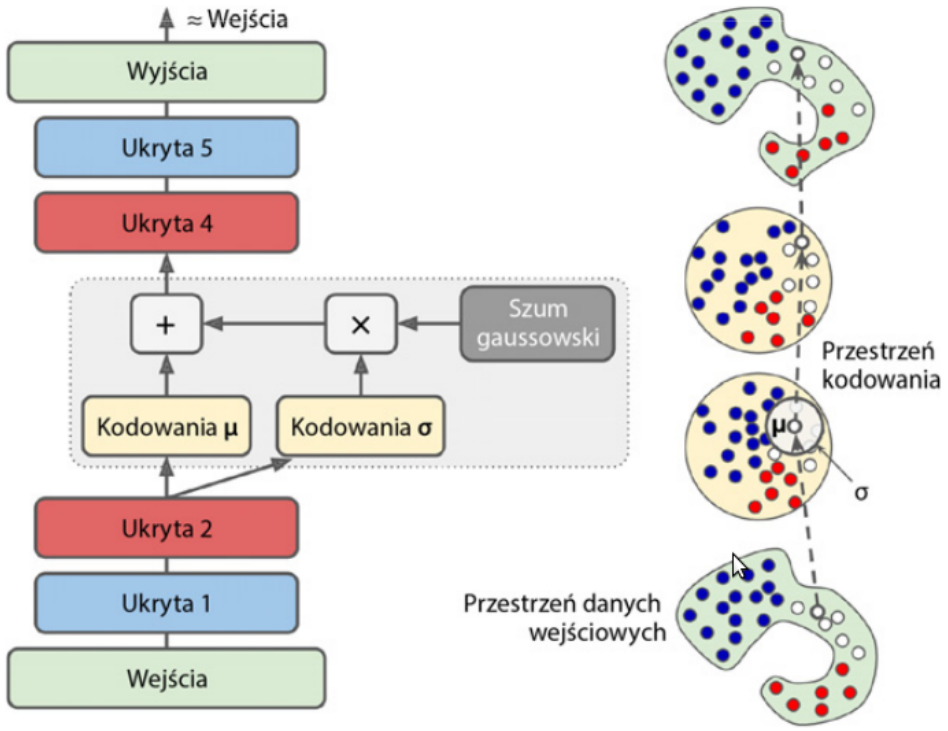
- Mając dwa dyskretne rozkłady prawdopodobieństwa P i Q, możemy obliczyć rozbieżność (odległość) pomiędzy nimi, np. za pomocą dywergencji Kullbacka-Leiblera

$$D_{KL}(P,Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

- Gdy chcemy zmierzyć rozbieżność pomiędzy docelowym prawdopodobieństwem p aktywacji neuronu w warstwie kodowania a rzeczywistym prawdopodobieństwem q (średnią aktywacją) wówczas

$$D_{KL}(p,g) = p \log \frac{p}{q} + (1-p)p \log \frac{1-p}{1-q}$$

Autokodery wariacyjne

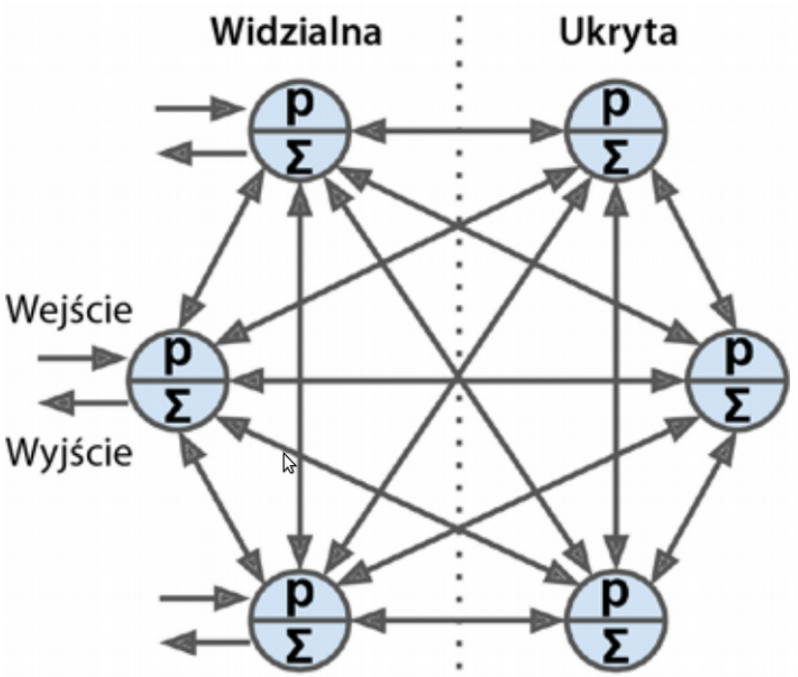


- Są autokoderami probabilistycznymi (ang. probabilistic autoencoders) - że generują częściowo losowe wyniki, nawet po wyuczeniu modelu,
- Stanowią klasę autokoderów generatywnych, czyli są w stanie tworzyć nowe próbki przypominające dane zawarte w zbiorze uczącym.

Autokodery wariacyjne

- Pomimo skomplikowanego rozkładu danych wejściowych autokoder wariacyjny jest w stanie generować kodowania przypominające próbkowania za pomocą prostego szumu gaussowskiego
- W czasie uczenia funkcja kosztu zmusza do stopniowego poruszania się po przestrzeni kodowania (zwanej również przestrzenią ukrytą - ang. latent space)
- Konsekwencją takiego zachowania jest fakt, że po wyuczeniu autokodera wariacyjnego możemy bardzo łatwo wygenerować nową próbkę: wystarczy próbkować losowe kodowanie z rozkładu gaussowskiego, rozkodować je.

Maszyny Boltzmanna



- Maszyna Boltzmana stanowi w pełni połączona topologia sieci neuronowej,
- Neurony w maszynach Boltzmanna dzielą się na dwie grupy: **jednostki widoczne** i **jednostki ukryte**

Neuron w maszynie Boltzman

- Stosowana funkcją aktywacji bazuje na rozkładzie Biolzmana - neurony są stochastyczne.
- Wartość wyjściowa jest prawdopodobieństwem, że i-ty neuron umieści na wyjściu wartość 1:

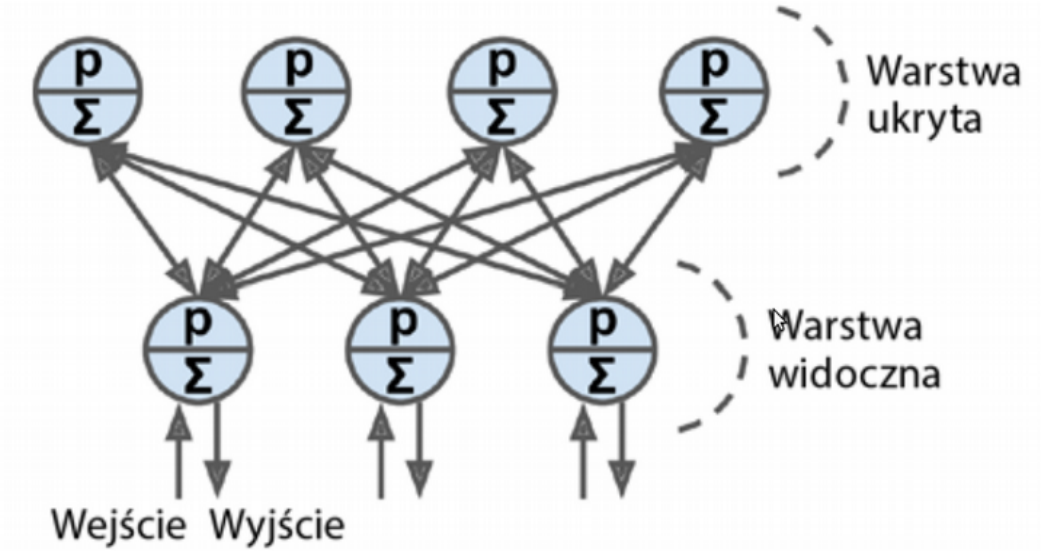
$$p(s_i^{next} = 1) = \frac{1}{1 + \exp^{-\frac{\Sigma_{j=1}^{N} w_{ij} s_j + b_i}{T}}}$$

  gdzie s_i jest stanem i-tego neuronu (0 lub 1), $w_{i,j}$ wagą połączenia pomiędzy i-tym i j-tym neuronem, b_i biasem, N liczbą neuronów a T jest parametrem zwanym **temperaturą**
- Wraz ze wzrostem temperatury zwiększa się losowość wyników,

Działanie maszyny Boltzmana

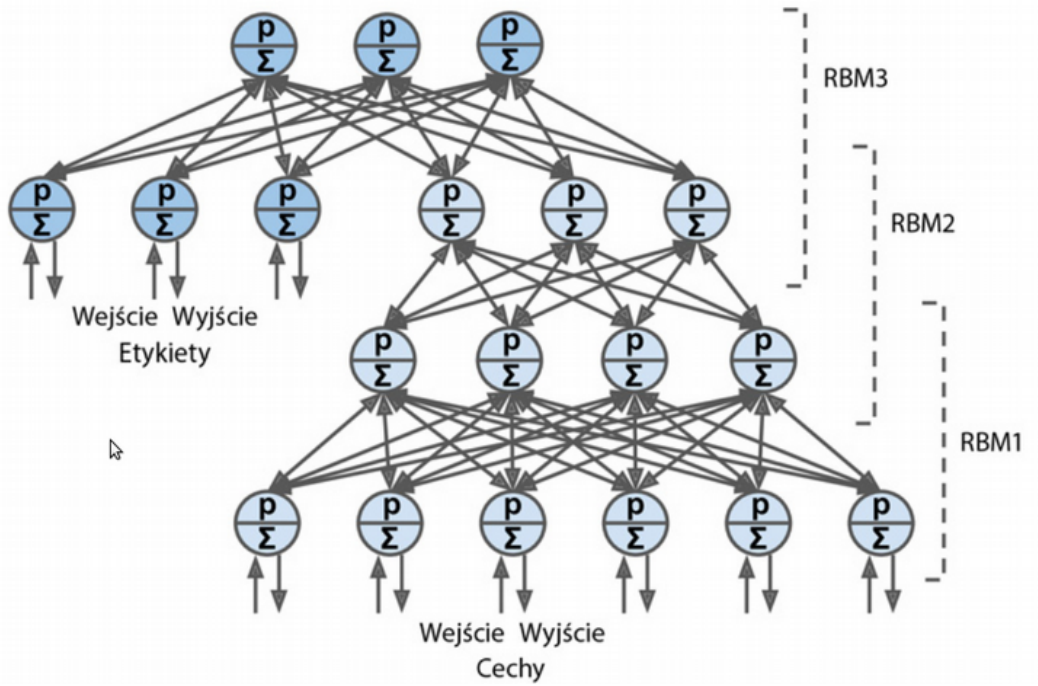
- Z powodu stochastycznej natury maszyna Boltzmanna nigdy nie osiągnie ustalonej, stabilnej konfiguracji,
- Będzie przeskakiwać pomiędzy wieloma konfiguracjami - prawdopodobieństwo zaobserwowania określonej konfiguracji będzie stanowiło jedynie funkcję wag połączeń i członów obciążeń,
- Sieć osiągnie stan, w którym pierwotna konfiguracja zostaje "zapomniana", znajduje się w równowadze cieplnej,
- Maszyna Boltzmana jest w stanie symulować szeroki zakres rozkładów prawdopodobieństwa. Jest to tak zwany **model generatywny**.
- **Uczenie maszyny Boltzmanna** polega na doborze parametrów pozwalających uzyskać *rozkład prawdopodobieństwa* zbioru danych uczących.

Ograniczone maszyny Boltzmanna



- W ograniczonej maszynie Boltzmanna nie ma połączeń pomiędzy poszczególnymi neuronami widocznymi lub neuronami ukrytymi,
- Połączone są ze sobą jedynie jednostki widoczne i ukryte.

Głębokie sieci przekonań (deep belief network - DBN)



- Jednostki ukryte pierwszej ograniczonej maszyny Boltzmanna stają się jednostkami widocznymi drugiej maszyny itd.

Uczenie Głębokich Sieci Przekonań

- Maszyna RMB1 jest trenowana w sposób nienadzorowany - uczy się rozpoznawać ogólne cechy w danych uczących.
- Dalej następuje proces uczenia maszyny RBM2, gdzie rolę jednostek wejściowych pełnią neurony ukryte maszyny RBM1 - uczenie jest nienadzorowane.
- Na koniec maszyna RBM3 jest uczona za pomocą jednostek ukrytych maszyny RBM2 pełniących funkcję wejść, a także dodatkowych jednostek widocznych reprezentujących etykiety docelowe - ten etap uczenia jest nadzorowany.
- W taki właśnie sposób głęboka sieć przekonań może być użyta w zadaniach klasyfikacji (nie potrzeba zbyt wielu oznakowanych danych uczących).

- Sieci DBN mogą działać również w drugą stronę. Jeśli pobudzimy jedną z jednostek etykiet, sygnał najpierw dotrze do jednostek ukrytych maszyny RBM 3, następnie zejdzie do maszyny RBM 2, a stamtąd do maszyny RBM 1, gdzie nowa próbka zostanie umieszczona na wyjściach neuronów widocznych.
- W ten sposób jesteśmy w stanie generować automatyczne podpisy obrazów i odwrotnie.

Próbkowanie Monte Carlo łańcuchami Markowa (ang. Markov Chain Monte Carlo (MCMC))

- MCMC jest metodą estymacji dowolnego rozkładu prawdopodobieństwa.
- Może być użyta w sytuacjach, w których nie znamy próbkowanej funkcji gęstości, a jedyne co możemy, to obliczać prawdopodobieństwo danej zmiennej losowej.
- Idea MCMC opiera się na zbudowaniu łańcucha Markowa z rozkładem stacjonarnym podobnym do próbkowanego rozkładu prawdopodobieństwa.
- Błądząc po łańcuchu niektóre stany będą odwiedzane częściej niż inne - stan ma przypisane pewne prawdopodobieństwo (określone częstotliwościowo),
- Prawdopodobieństwa wszystkich stanów definiują rozkład stacjonarny danego łańcucha Markowa.

Algorytm Metropolisa-Hastingsa MCMC

Macierz przejścia stanów mozna określić w następujący sposób:

1. Zacznij w losowym stanie,
2. Wybierz losowy, inny stan,
3. Przejdź do kolejnego stanu z prawdopodobieństwem równym $P = \min\{1, P_{t+1}|P_t\}$. Jeżeli stan zostanie odrzucony, to wybierz inny
4. Powtórz od kroku 2.

Historia przebytych stanów stanowi próbkę nieznanego rozkładu.

Zadania na ćwiczenia

- 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.