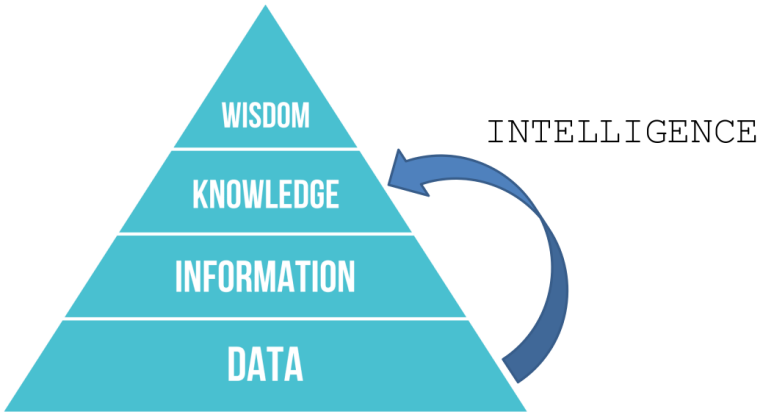
Uczenie Maszynowe

Definicja wiedzy



Data -> Information -> Knowledge -> Wisdom

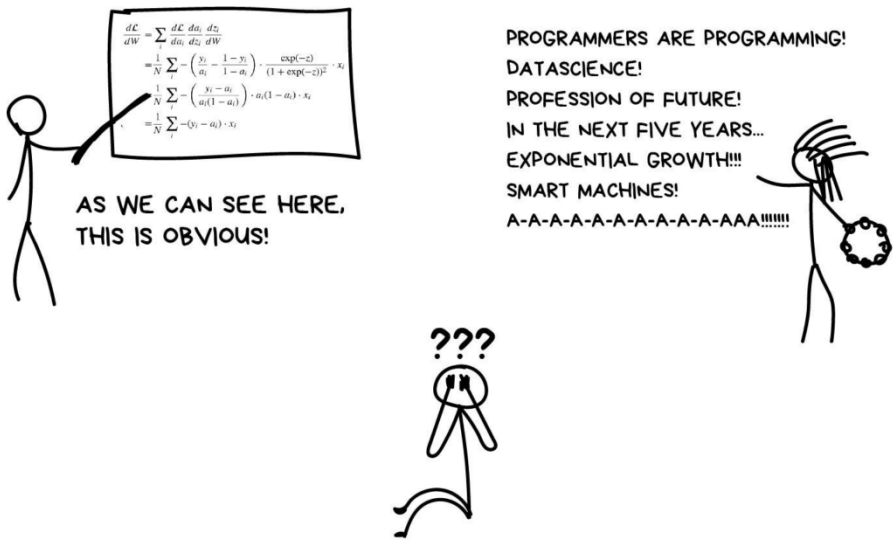
Data — dane — suche dane bez kontekstu
Information — informacja — dane z kontekstem (zestaw RGB -> obraz, cyfra -> temperatura)
Knowledge — wiedza — zestaw danych z kontekstem
Wisdom — mądrość
Inteligencja to proces przetworzenia danych surowych w wiedzę (Data/Information -> Knowledge)
Pozyskanie informacji z danych to jest **parsowanie** (prosty proces)

ML

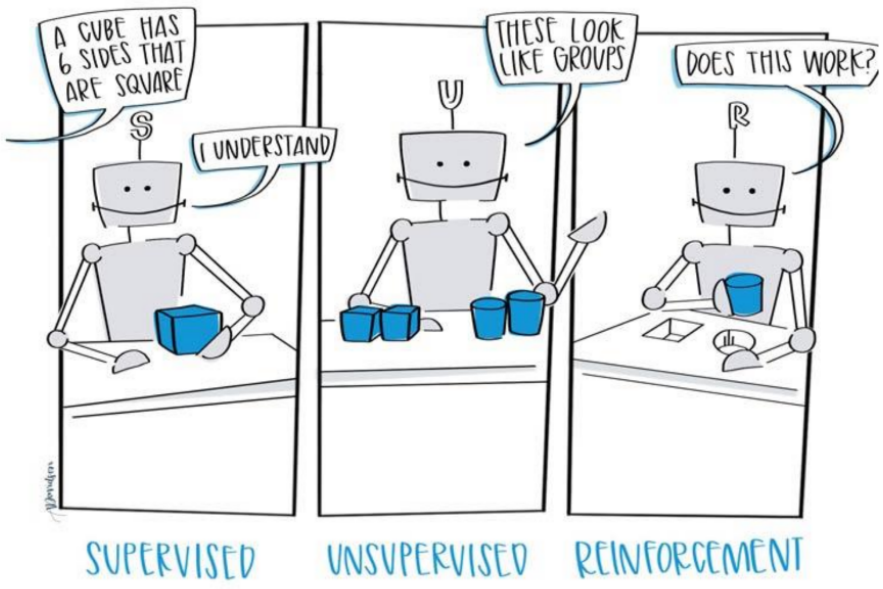
O **systemach uczących** mówimy w momencie gdy system jest w stanie automatycznie poprawiać swoje działanie na podstawie doświadczenia lub danych, bez jawnego programowania każdej możliwej sytuacji.
W uczeniu maszynowym predykcja nie jest jedynym zastosowaniem ML.

Skrajne nurty ML w mediach / pracach naukowych:

- **matematyczno-teoretyczny** — optymalizacje uczenia, brak przełożenia dla użytkownika / użytkowego zastosowania
- **ludzi niezwiązanych z IT**

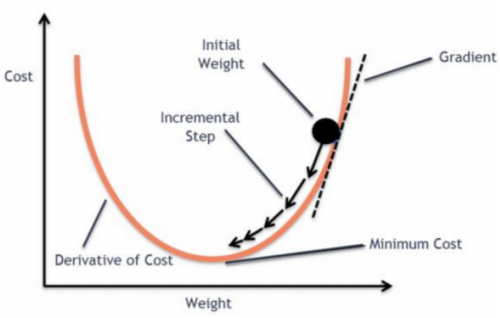


TYPY UCZENIA MASZYNOWEGO



SUPERVISED — NADZOROWANE

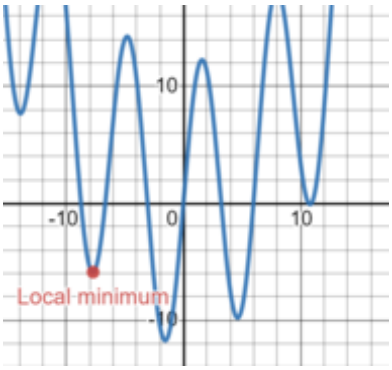
- **regresja** -> liczby rzeczywiste, przestrzeń ciągła, prognoza dokładnego wyniku
  - przykład z funkcją liniową
  - modelując szukamy funkcji która jak najlepiej odwzorowuje rzeczywistość
  - oceniając **jakość modelu** musimy określić **funkcję kosztu**
    - istnieją rózne funkcje kosztu i w zależności od problemu się dobiera odpowiednią
    - chcemy aby **funkcja kosztu** była **jak najmniejsza**
  - **pochodna i gradient**
    - **pochodna** pokazuję dynamikę zmiany w funkcji oraz do poszukiwania ekstremum
    - **gradient** to wektor pochodnych cząstkowych
    - **ekstremum lokalne funkcji jednej zmiennej** -> pochodna równa zero
      - np powierzchnia mieszkania wpływa na cenę
      - graf 2D wizualizuje
    - **ekstremum lokalne funkcji wielu zmiennych** -> gradient równy zero - np powierzchnia, status, rok budowy etc. wpływa na cenę - minimum powierzchnia (3D) wizualizuje
  - **algorytm spadku wzdłuż gradientu (gradient descent)**



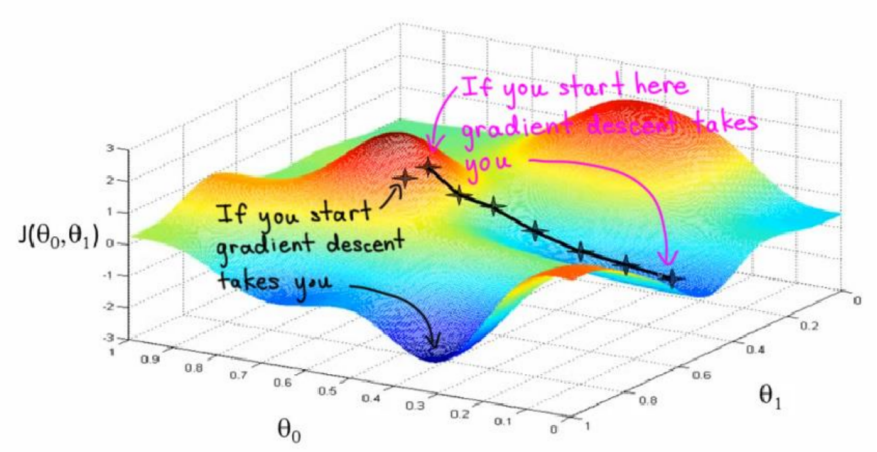
$$a = a - \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} \left(ax^{(i)} + b - y^{(i)}\right) \cdot x^{(i)}$$

$$b = b - \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} \left(ax^{(i)} + b - y^{(i)}\right)$$

* https://www.mladdict.com/linear-regression-simulator
* **stała uczenia — learning rate** — odpowiednio wysoka wartość pozwala na wydostanie się z **minimum lokalnego**

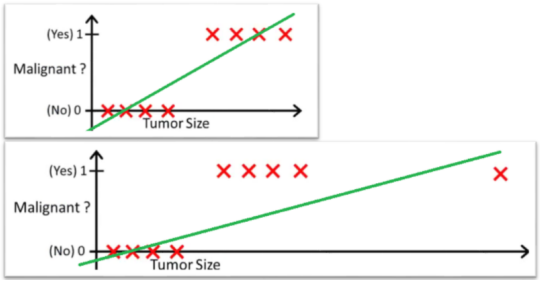


* ma tendencję do utykania w minimach lokalnych przez co ma znaczenie jakie są argumenty początkowe



* bardzo prosty algorytm ale ma swoje wady — w zależności gdzie zaczniemy możemy skończyć w różnym miejscu

- **klasyfikacja** -> dane dyskretne, np oceny (ograniczona skala 2-5.5), rzut kostką
  - regresja liniowa do klasyfikacji? — regresja liniowa jest w większości przypadków niemożliwa do klasyfikacji, ponieważ nowe próbki spoza zakresu zmieniałyby jej kształt i punkt aktywacji by ulegał mocnej zmianie




```
h(x)>0.5→malignant
```

```
h(x)>0.2→malignant
```

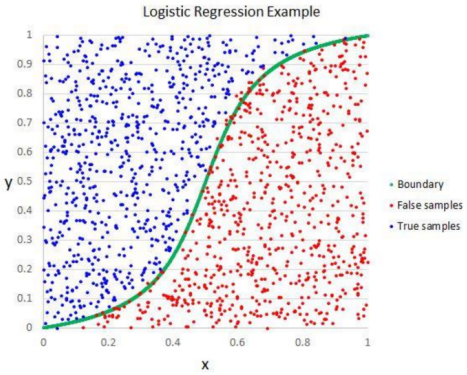
- regresja logistyczna

$$S: (-\infty,\infty) \longrightarrow \langle 0,1 \rangle$$

$$S(t) = \frac{1}{1+e^{-t}}$$

$$t = ax + b$$

$$S(t) = \frac{1}{1+e^{-(ax+b)}}$$

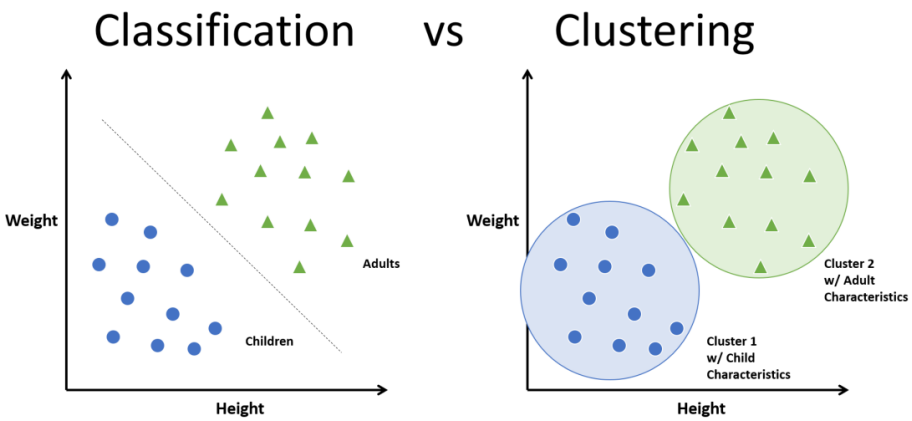


  * jedyne elementy które można zmienić to są współczynniki (a i b w przykładzie)
  * w modelu realistycznie jest podział tak/nie (kot/nie kot, pies/nie pies)
  * nadal uczy się ją algorytmem spadku wzdłuż gradientu

UNSUPERVISED — NIENADZOROWANE

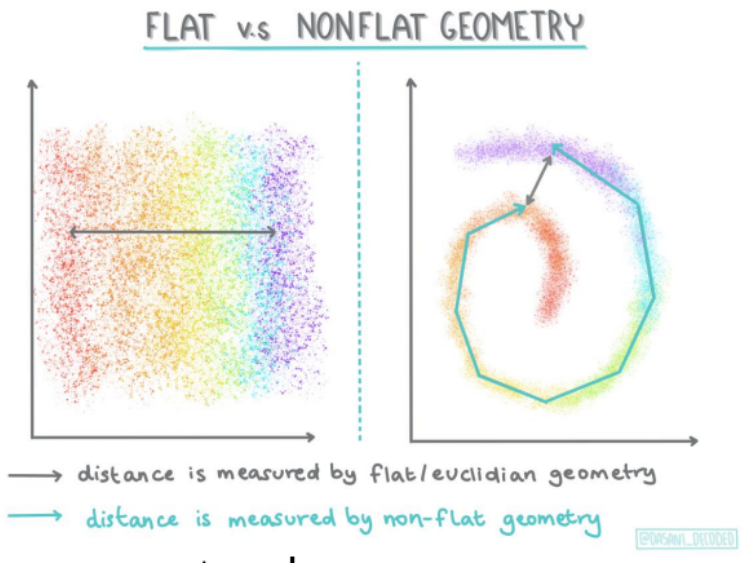
Uczenie nienadzorowane (clustering — analiza skupień) — wrzucenie danych do znalezienia jakichś grup, np. podział grup klientów

- **classification vs clustering** — algorytm klasyfikacyjny jest o wiele lepszy pod względem biznesowym, clustering jest bardziej do odkrycia grup wśród danych

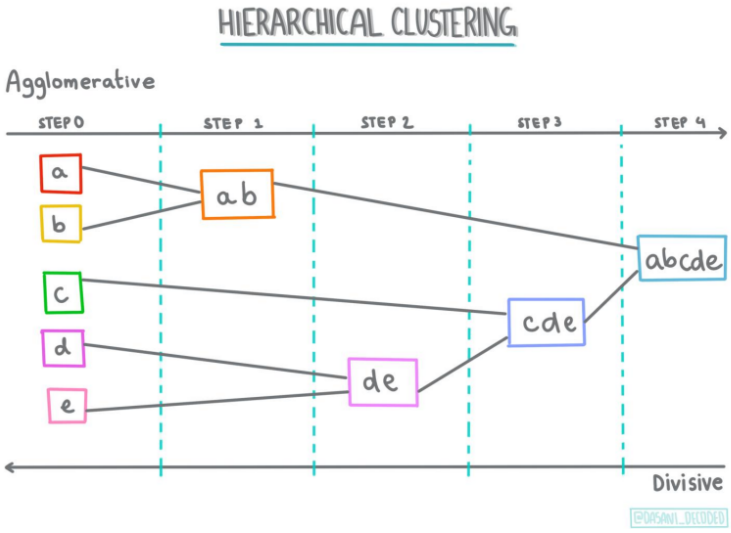


- **generowanie klastrów**
  - **indukcyjne vs transdukcyjne**
    * **indukcyjne** — klastry są pewną własnością całego rozkładu danych a nie tylko próby treningowej (może przypisywać nowe punkty bez konieczności ponownej klasteryzacji) np. k-means, GMM (Gaussian Mixture Models), uczenie reprezentacji
    * **transdukcyjne** — szukanie najbliższych próbek; grupuje tylko konkretne dane, które dostał na wejściu, bez tworzenia ogólnej funkcji decyzyjnej np. spektralna klasteryzacja, agglomerative / hierarchical clustering, DBSCAN
  - distance clustering vs density clustering vs constrained clustering
    * **distance** — odległość w przestrzeni (np 100 wymiarowej)
    * **density** — nagromadzenia (niekoniecznie odległości — przykład dwóch grup gdzie odległość pomiędzy 2 próbkami będzie mniejsza niż z innej próbki z nagromadzenia)
    * **constrained clustering** — metoda grupowania, w której proces klasteryzacji przebiega z uwzględnieniem wcześniej określonych ograniczeń (np. przykłady, które muszą lub nie mogą znaleźć się w tym samym klastrze)

- ”płaska” vs ”niepłaska” geometria

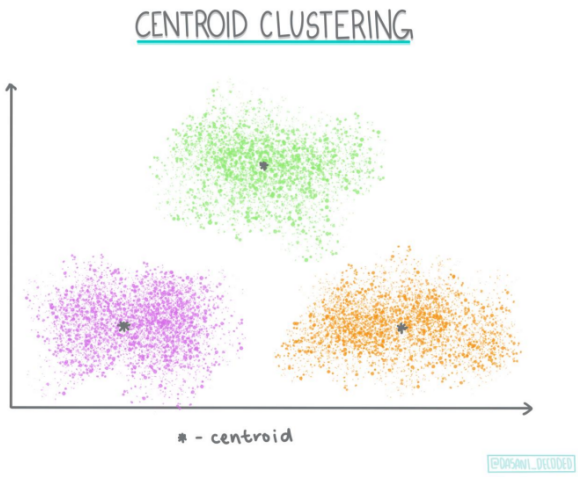


  * w jaki sposób dystans jest obliczany?
  * **płaska powierzchnia** (przestrzeń euklidesowa) — 2 wymiary pitagoras, więcej wymiarów dodawane pod pierwiastek
  * **niepłaska powierzchnia** (przestrzeń nieeuklidesowa) — przestrzeń w której nie obowiązują klasyczne zasady geometrii euklidesowej (np powierzchnia kuli — geometria sferyczna; przestrzeń zakrzywiona przez grawitację — ogólna teoria względności)
- typy algorytmów clusteringowych
  * **hierarchiczne**



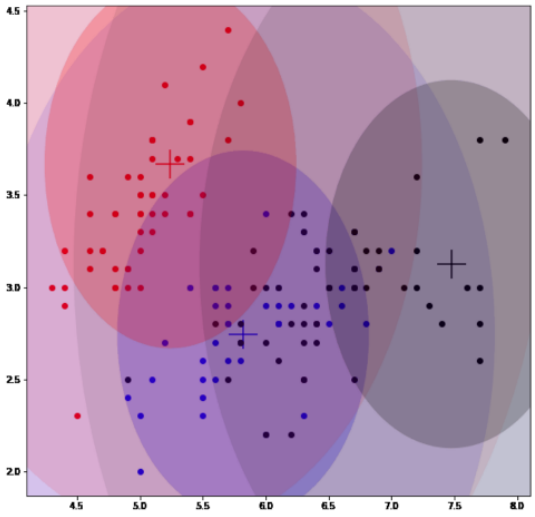
    · generują drzewa klastrów; klastry oraz podklastry tworzone są na podstawie odległości ich członków od i do innych obiektów
    · np. agglomerative algorithm

⋆ **centroidowe (centroid-based)**



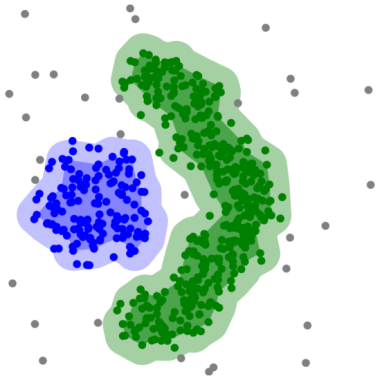
- wymaga podania parametru k, po którym algorytm określa punkt centralny każdego z k klastrów i grupuje dane wokół tych punktów centralnych
- przykład algorytm k średnich, punkt centralny jest określany przez najbliższą średnią
- problem — trzeba podać ilość klastrów
  elbow trick jako rozwiązanie problemu (ma nadal swoje problemy) + graf silhouette / n

⋆ **rozkładowe (distribution-based)**



- bazuje na założeniu że dane są rozmieszczone z jakimś rozkładem (np. Gaussa); wraz ze wzrostem odległości od środka rozkładu maleje prawdopodobieństwo, że punkt należy do rozkładu
- przykład gaussian mixture model
- **krzywa gaussa** — bliżej środka jest więcej próbek niż skrajnych

* **gęstościowe (density-based)**



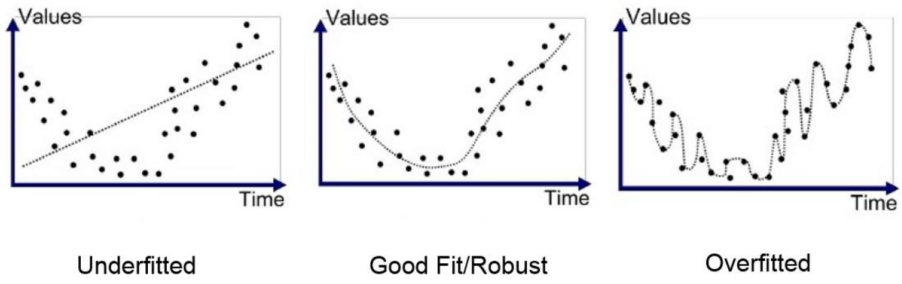
- zakłada że klaster to przestrzeń o dużym nagromadzeniu punktów na małym obszarze, oddzielony od innych klastrów pustymi obszarami, pojedyncze punkty w pustych obszarach to szum / outliery
- przykład dbscan

Reinforcement — Ze wzmocnieniem

Uczenie ze wzmocnieniem — w określonym środowisku z nagrodami i karami

Przeuczenie

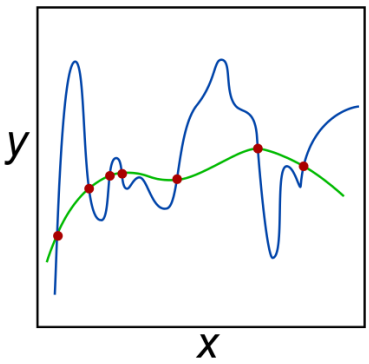
Wizualizacja



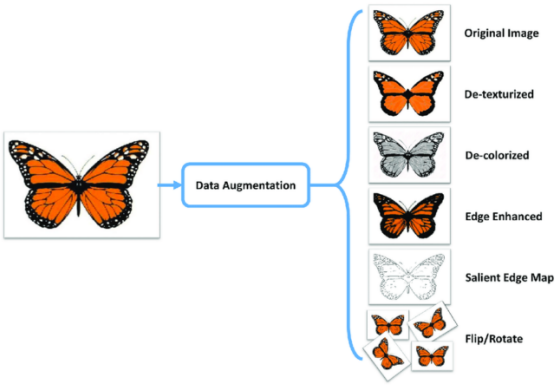
- underfitted
- good fit/robust
- overfitted

Co robić, jak żyć?

Regularyzacja

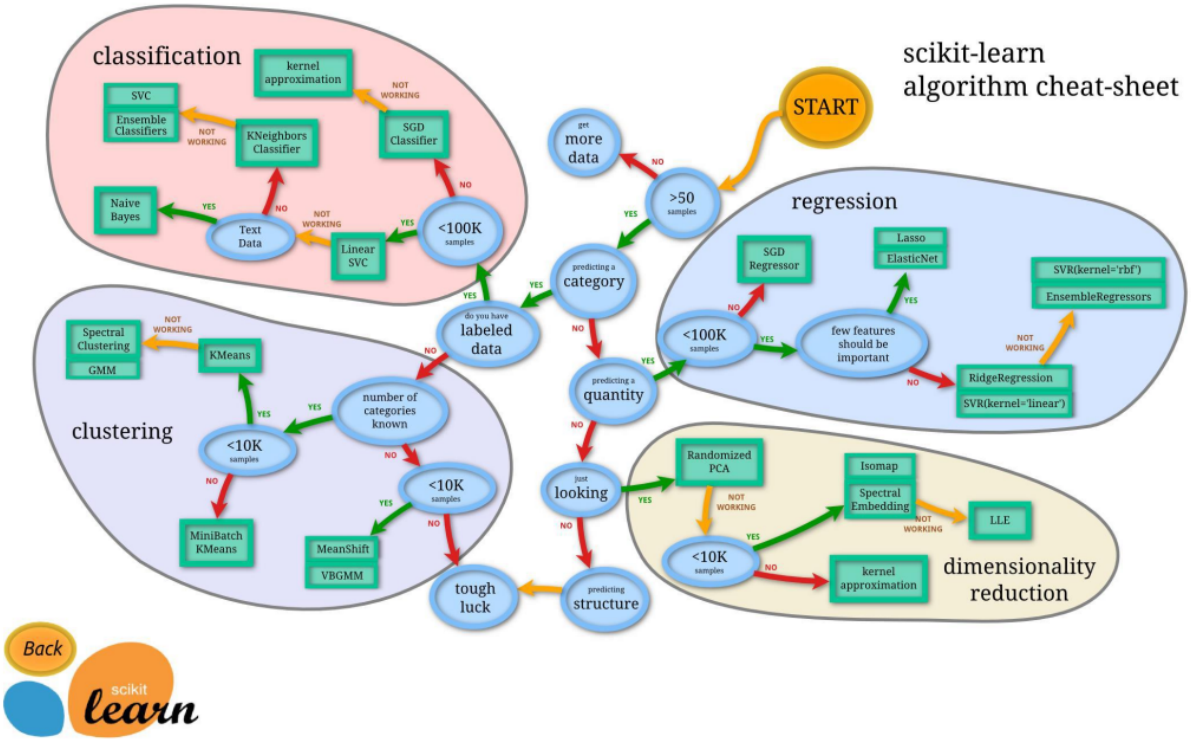


Data augmentation



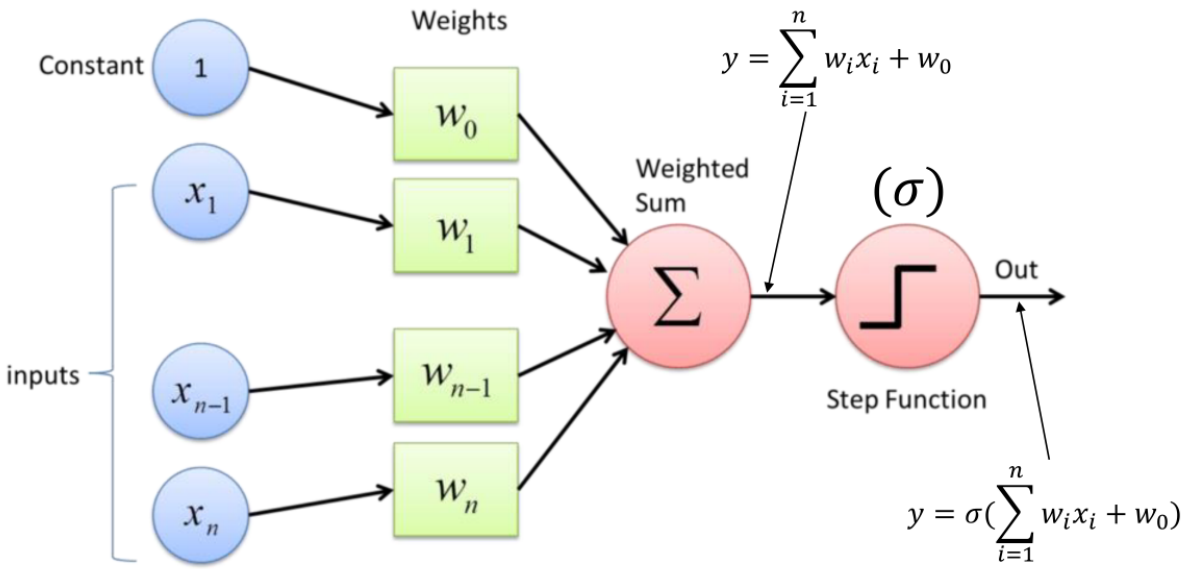
- **regularyzacja** — stosowana by zminimalizować ryzyko przeuczenia, wprowadza dodatkową karę za duże wartości wyuczonych wag
  - kara — najczęściej suma lub iloczyn wag, dodawana dodatkowo do funkcji kosztu
  - błąd średniokwadratowy z regularyzacją
- **data augmentation** — sztuczne zwiększenie zbioru treningowego przez wprowadzanie kontrolowanych modyfikacji istniejących danych. Celem jest poprawa ogólności modelu, redukcja overfittingu i zwiększenie różnorodności danych bez konieczności ich ponownego pozyskania.

Mapa uczenia maszynowego

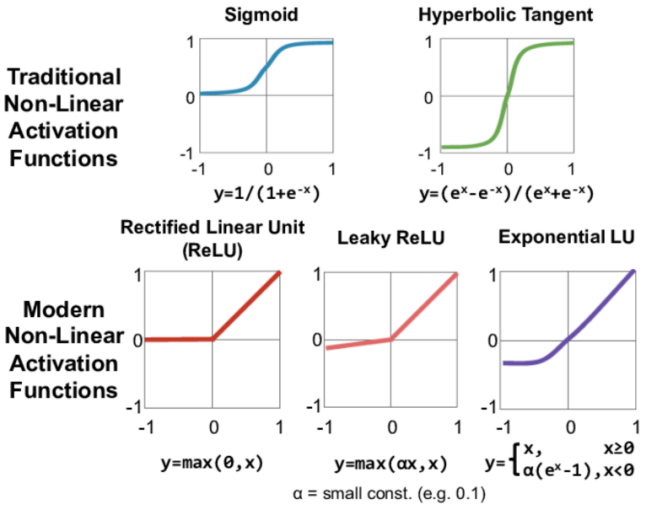


ANN

NEURON – KONCEPCJA



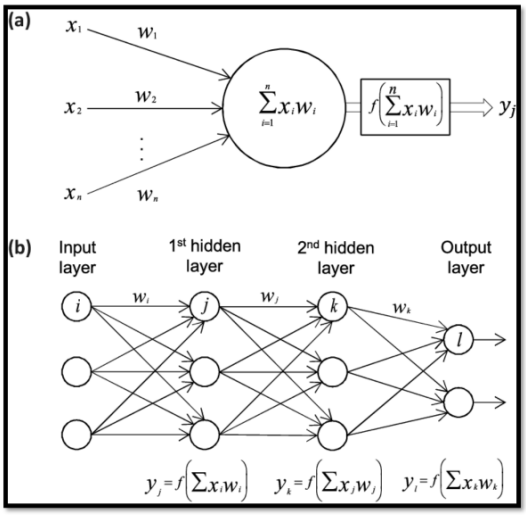
- odchylenie neuronu / bias
- blok aktywacyjny / funkcje aktywacji

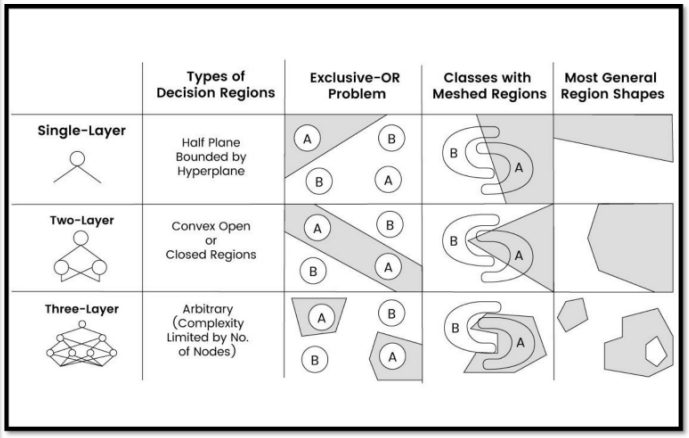


- do problemów nie separowalnych liniowo (tak/nie; A/B)

Warstwy i sieci, a po co?

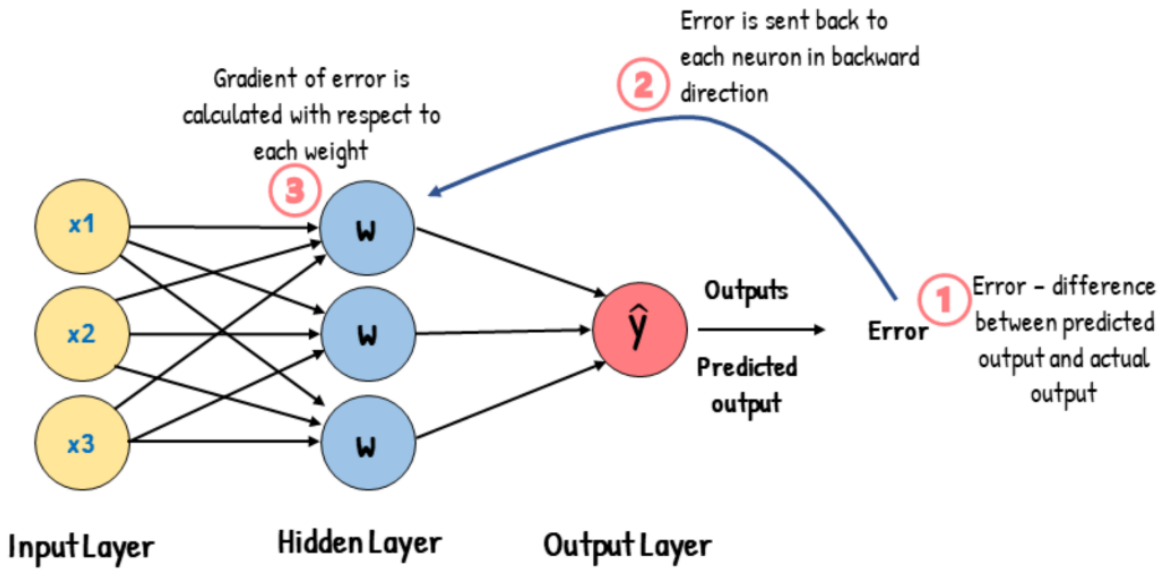
- warstwa wejściowa, ukryta, wyjściowa





- perceptron — sieć neuronowa z neuronami bez funkcji aktywacji
- softmax do klasyfikacji

Uczenie sieci — propagacja wsteczna



- błąd dotyczy błędu całej sieci nie tylko jednego neuronu, należy traktować neurony jako całość
- przykład z matematykami z NASA

YOLO — You Only Look Once

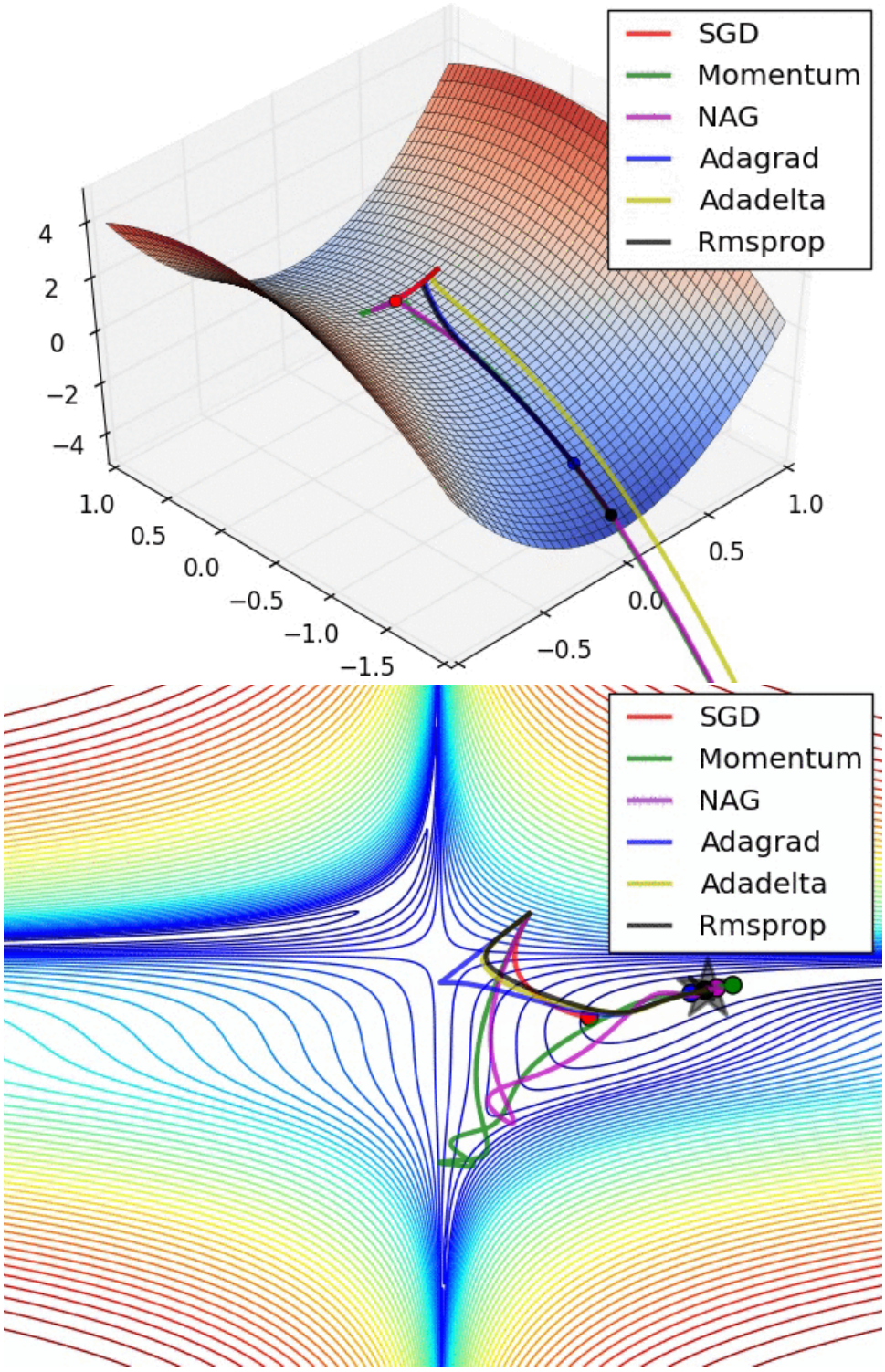
https://www.youtube.com/watch?v=Cgxsv1riJhI

Deep Learning

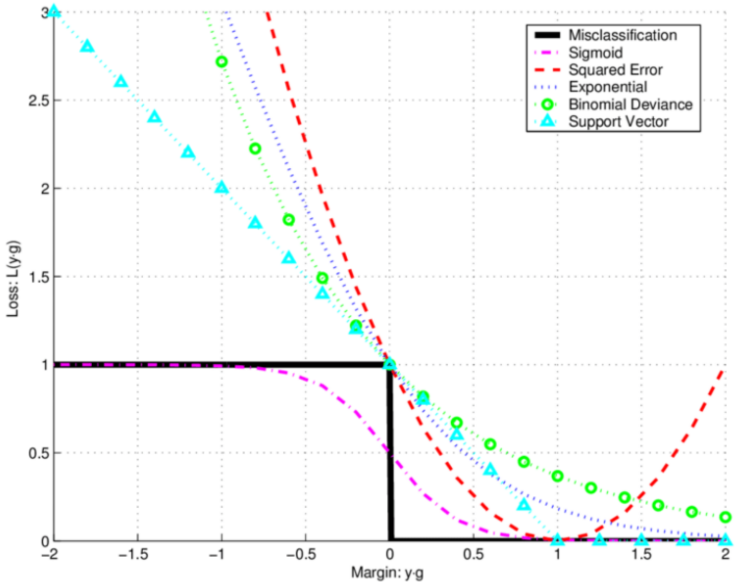
- Od 3 warstw wzwyż można powiedzieć o sieciach głębokich
- Warstwa gęsta — dense / fully-connected

Optymalizator

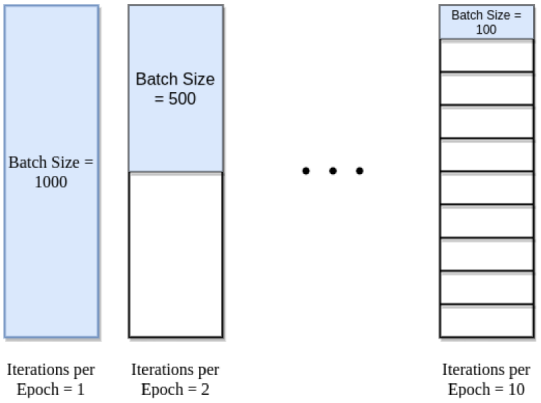
`https://imgur.com/a/visualizing-optimization-algos-Hqolp`



Funkcja straty/kosztu (loss function)



Batch / iteracja / epoka



Epoka — przepuszczenie wszystkich danych w **batch**'ach. Po każdej iteracji następuje propagacja wsteczna. Batch'e określa się z reguły wielkością danych a nie ile batchy jest w epokach (z reguły potęgi dwójki)

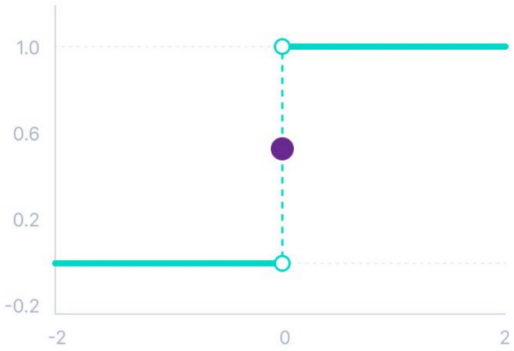
Zbiór walidacyjny

Uczenie sieci neuronowych z reguły trwa dłużej. Do sprawdzania performance'u sieci pomiędzy epokami wykorzystywany jest **zbiór walidacyjny**. Jest to najmniejszy zbiór danych i służy jako kontrolny zbiór dla epok (testowy jest z reguły za duży do takiego zastosowania)

Po co funkcja aktywacji

- Wprowadza nieliniowość
- Z reguły w ramach jednej warstwy neurony mają tą samą funkcję aktywacji

- **Signum**

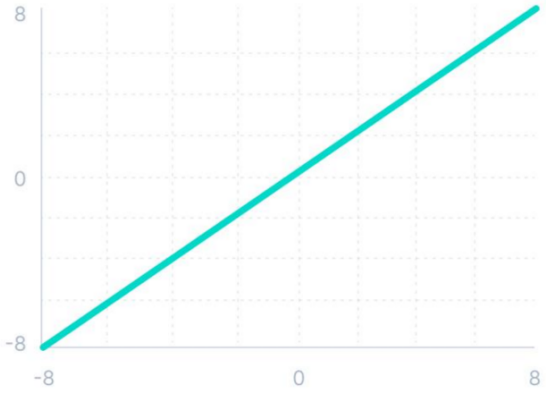


$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Cechy:

- nie może być funkcją aktywacji
- nie umożliwia wyjść o wielu wartościach (odpada przy klasyfikacji wieloklasowej)
- gradient nie istnieje, uniemożliwia zastosowanie propagacji wstecznej

- **Funkcja liniowa**

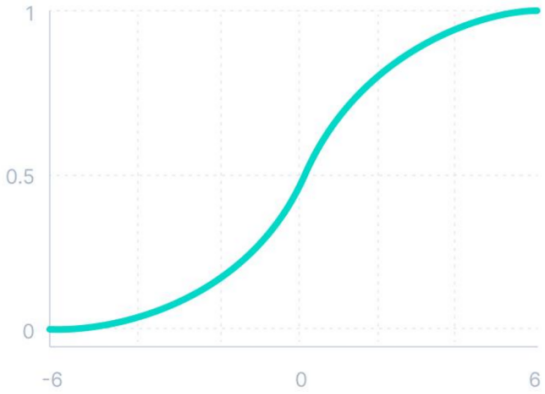


$$f(x) = x$$

Wady:

- sama nie wzbogaca przetwarzania
- propagacja wsteczna działa, ale nie jest zbyt znacząca, pochodna funkcji liniowej to liczba stała, niezależna od x

- **Sigmoid**

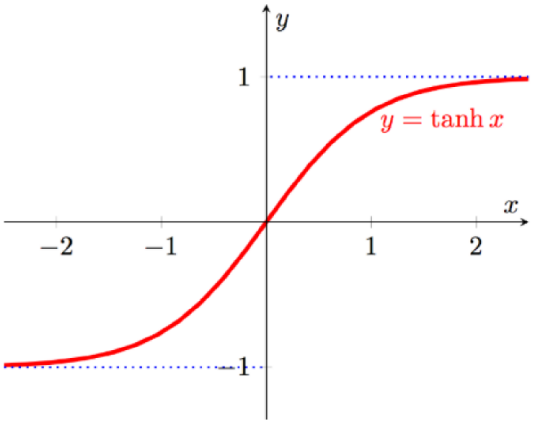


$$f(x) = \frac{1}{1 + e^{-x}}$$

Cechy:

- używana gdy na wyjściu ma pojawić się prawdopodobieństwo
- łagodny gradient, brak skoków na wyjściu
- niesymetryczna względem 0, nie daje możliwości ujemnego wyjścia
- nie jest dobrym w klasyfikacji wieloklasowej ale dobry w klasyfikacji wieloetykietowej
- często w warstwach wyjściowych

- **Tanh (tangens hiperboliczny)**



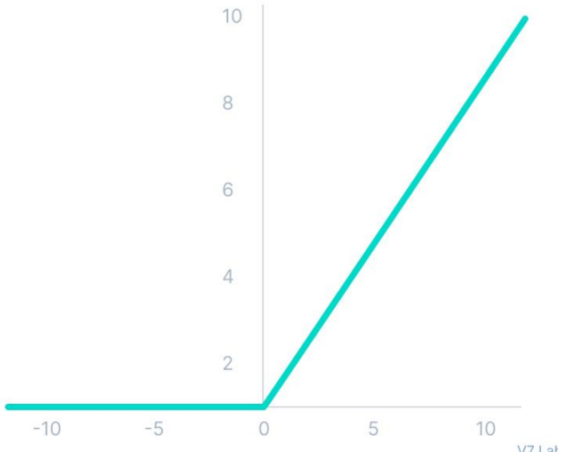
$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

Cechy:

- często w warstwach ukrytych

- używana w warstwach ukrytych, ,gdyż jej wartości lokują się między -1 a 1 (średnia oscyluje wokół 0), co pomaga wycentrować dane i ułatwić uczenie kolejnych warstw
- symetryczna względem 0, daje możliwości ujemnego wyjścia

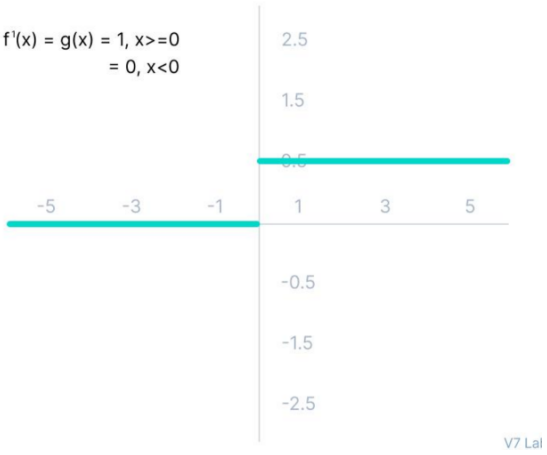
- **ReLU (Rectified Liniear Unit)**



$$f(x) = max\,(0, x)$$

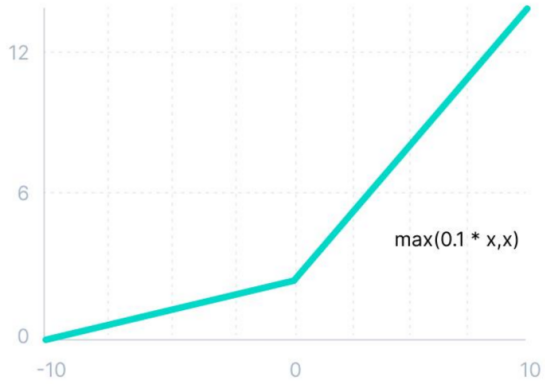
Cechy:

- nie wszystkie neurony są aktywowane (duży zakres zerowego wyjścia) — wydajna obliczeniowo, ale występuje problem **"dying ReLU"** — wagi w neuronach nieaktywnych nie są aktualizowane, więc niektóre neurony pozostają martwe (wejście neuronu mieści się zawsze w obszarze ujemnym i zwraca zawsze 0)



- zastosowania: problemy liniowe, zachowanie wartości bez zmian, odcięcie ujemnych wartości

- **Leaky ReLU**

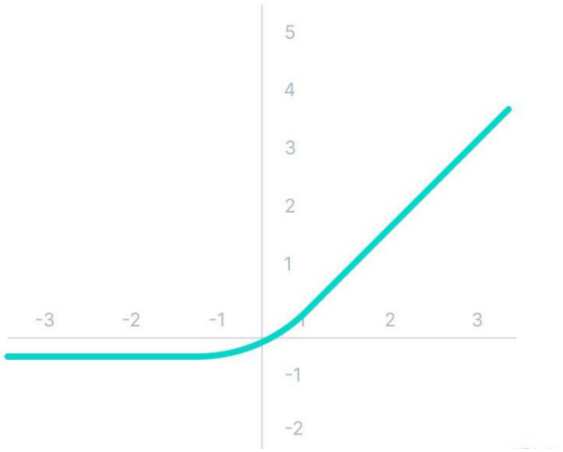


$$f(x) = max\,(0.1x, x)$$

Cechy:

- podobne zalety co ReLU, przy czym gradient dla ujemnych wartości nie jest zerowy
- **brak** problemu ”dying ReLU”
- gradient dla ujemnych wartości jest mały, co czyni proces uczenia bardziej czasochłonnym

- **ELU (Exponential Linear Unit)**

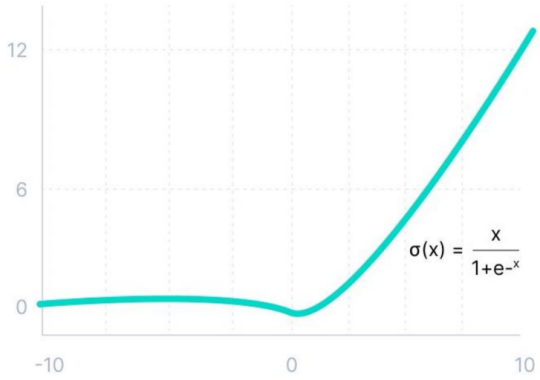


$$\begin{cases} x & for\ x \geqslant 0 \\ \alpha\left(e^x - 1\right) & for\ x < 0 \end{cases}$$

Cechy:

- podobne zalety co ReLU, przy czym gradient dla ujemnych wartości nie jest zerowy
- przebieg jest łagodniejszy niż Leaky ReLU, ale element wykładniczy zwiększa czas obliczeń
- **brak** problemu ”dying ReLU”

- **Swish**

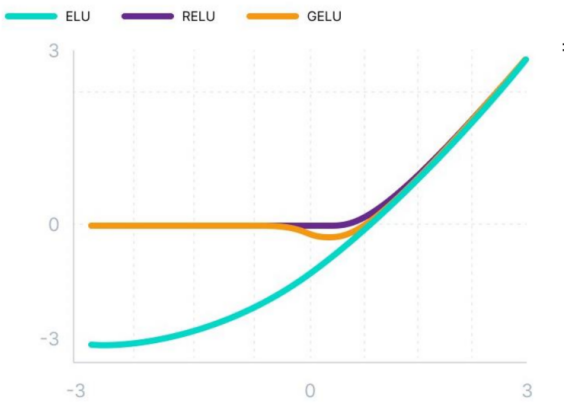


$$f(x) = x*sigmoid(x)$$

Cechy:

- użyta po raz pierwszy przez Google Brain Team
- ograniczona z dołu, nieograniczona z góry, można dodać modyfikującą stałą
- empirycznie stwierdzona większa skuteczność niż ReLU, zwłaszcza w bardzo głębokich sieciach (>40 warstw)
- **brak** problemu "dying ReLU"

- **GELU (Gaussian Error Linear Unit)**



$$f(x) = xP(X \leq x) = x\Phi(x)$$
$$= 0.5x\left(1 + tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)$$

Cechy:

- zamiast eliminować ujemne wejścia, modyfikuje je, uwzględniając ich wartość w oparciu o rozkład Gaussa — ujemne wartości są przepuszczane z mniejszym prawdopodobieństwem, a ich wpły na wynik jest mniejszy
- używana w modelach językowych

- **Softmax**

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\Sigma_j \exp(z_j)}$$

Cechy:

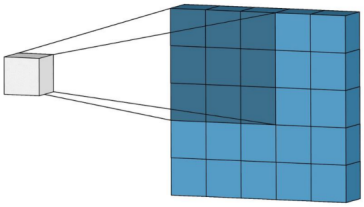
  - **nie da się wykreślić**
  - stosowana na warstwie wyjściowej
  - dobry do klasyfikacji wieloklasowej — kto jest na portrecie? zwróci szansę na każdą klasę

- Aktywacje na ostatnich warstwach

Rodzaj problemu	Funkcja aktywacji	Format danych wyjścia	Funkcja straty
Klasyfikacja binarna	sigmoid		binary_crossentropy
Klasyfikacja wieloklasowa	softmax	[1,0,0], [0,1,0],[0,0,1]	categorical_crossentropy
Klasyfikacja wieloklasowa	softmax	[1], [2], [3]	sparse_categorical_crossentropy
Klasyfikacja wieloetykietowa	sigmoid		binary_crossentropy
Regresja	brak (linear)		MSE
Regresja prawdopodobieństwa	sigmoid		MSE / binary_crossentropy

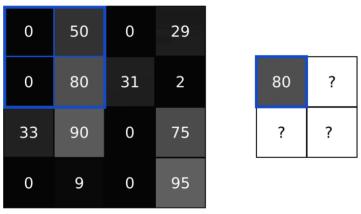
Sieci splotowe

Warstwa Splotowa — Convolutional Layer



Warstwy splotowe są podstwą sieci konwolucyjnych, ponieważ zawierają wyuczone filtry (kernele), które wyodrębniają cechy odróżniające od siebie różne obrazy. Wartości w filtrach są dobierane i zoptymalizowane podczas trenowania sieci. Wagi dobierane są dla całego filtra, który następnie przesuwany jest po całym zdjęciu.

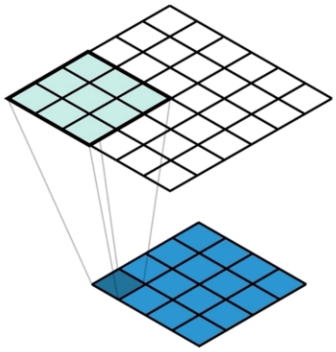
Warstwa Łącząca — Pooling Layer



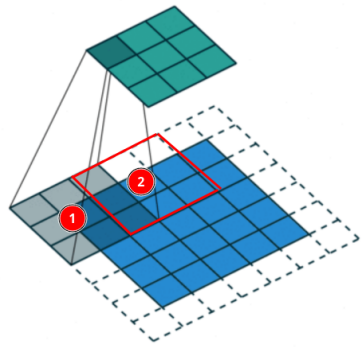
Sąsiednie piksele na obrazach mają zwykle podobne wartościm więc warstwy splotowe również generują wtedy podobne wartości dla sąsiednich pikseli — wtedy wiele informacji zawartych w danych wyjściowych jest zbędnych. Celem warstw pooling jest stopniowe zmniejszanie rozmiaru obrazu, co zmniejsza liczbę parametrów do wytrenowania, przy zachowaniu informacji o obrazie.

Hiperparametry Sieci Splotowych

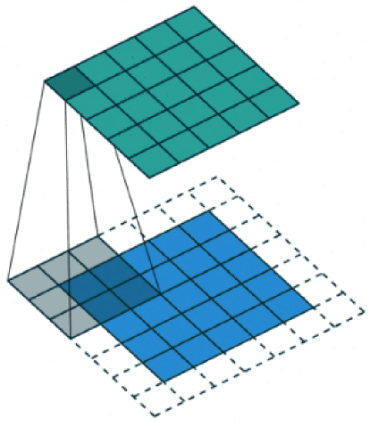
- **Kernel size** — rozmiar filtra odnosi się do wymiarów przesuwanego okna. Małe kernele są w stanie wydobyć z danych wejściowych znacznie większą ilość informacji zawierających lokalne cechy. Duży rozmiar filtra wyobrębnia mniej informacji, ale czasami możemy zyskać lepszą generalizację problemu. (Kernel 3x3)



- **Stride** — wskazuje o ile pikseli kernel powinien zostać przesunięty na raz. Czyli inaczej mówiąc, oznacza krok przesunięcia okna filtra. Najczęściej używa się kroku 1. (Przykład stride 2x2 z kernelem 3x3)



- **Padding** — wypełnienie definiuje sposób obsługi obramowania próbki. Umożliwia to otrzymanie rozmiaru wyjśccia takiego samego jak rozmiar wejścia (przy założeniu, że stride jest przesunięciem o jeden). Osiąga się to kosztem dodania dodatkowych (sztucznych) wag na krawędziach (najczęściej z wartością zero)



Jak projektować model?

Liczba warstw

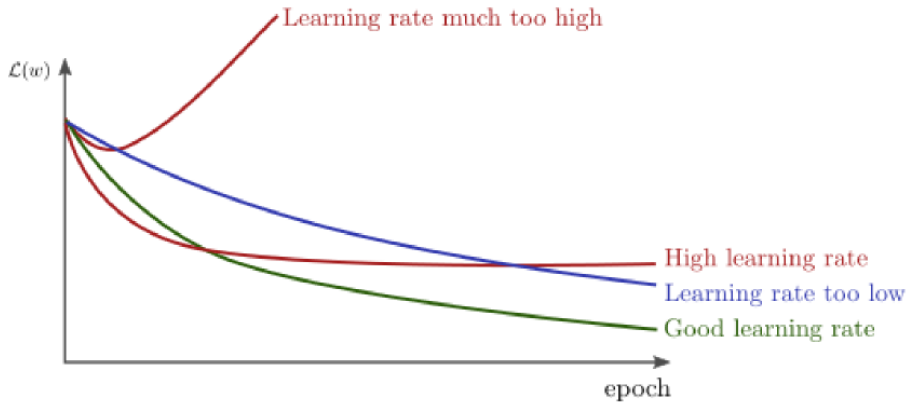
- dla każdego problemu istnieje optymalna liczba warstw ale jej nie znamy
- każda dodatkowa warstwa reprezentuje cechy abstrakcyjne -> im bardziej abstrakcyjny jest problem, tym więcej warstw będzie potrzebnych
- najlepiej zacząć od 3-4 warstw ukrytych, zwiększać dopóki koszt na wyjściu maleje
- **"brzytwa Ockhama"** — spośród konkurencyjnych wyjaśnień należy wybierać to najprostsze, które wystarczająco tłumaczy obserwacje. W kontekście uczenia maszynowego — prostszy model (z mniejszą liczbą parametrów, mniej złożony) jest zwykle preferowany, jeśli osiąga podobną skuteczność jak model bardziej skomplikowany.

Liczba neuronów

- liczba neuronów na pierwszych warstwach zależy od liczby cech wejściowych które wymagają reprezentacji
- jeśli zakładamy, że może wystąpić sporo cech wysokopziomowych, zwiększamy liczbę neuronów w dalszych warstwach
- konkretne wielkości najczęściej ustala się empirycznie mnożąc lucz dzieląc liczbę neuronów w danej warstwie przez 2 do n
- ponownie "brzytwa Ockhama"

Learning rate

- duże wartości powodują że możemy ominąć minimum funkcji kosztu
- małe wartości zwiększają długość procesu uczenia
- w praktyce najlepiej zacząć od 0.01 lub 0.001, a następnie obserwować koszt i czas uczenia i odpowiednio korygować



Niestabilne gradienty

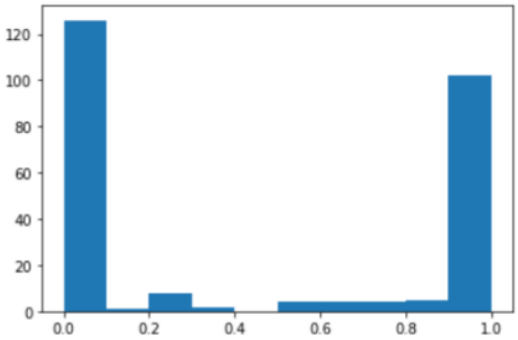
- propagacja wstecz polega na przekazywaniu wartości funkcji kosztu od wyjścia do wejścia
- parametry modyfikowane są proporcjonalnie do ich grafientu w odniesieniiu do kosztu (jeśli gradient parametru a w stosunku do kosztu jest wysoki i dodatni tzn. że ma duży wpływ na koszt)
- im warstwa bliżej wejściowej tym większe prawdopodobieństwo zanikania (vanishing) lub eskplozji (explosion) gradientu, a co za tym idzie — saturacji neuronu
- rozwiązania — batch normalization, inicjalizacja wag

Inicjalizacja parametrów

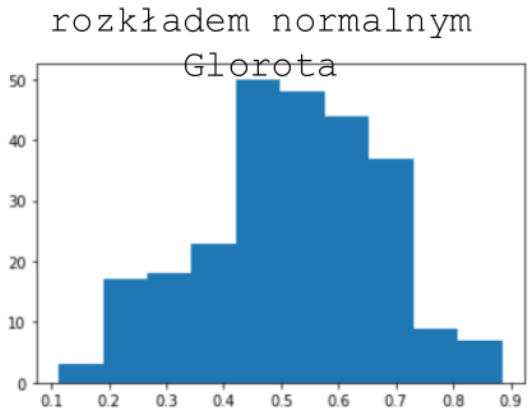
Normalizacja wsadowa – batch normalization

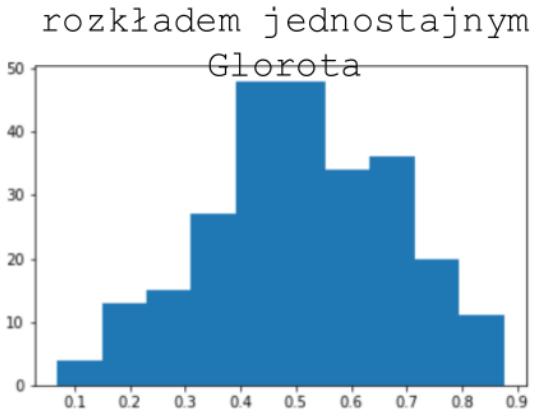
Aby sieć uczyła się efektywnie, chcemy, aby sygnał nie zanikał ani nie eksplodował. Oznacza to utrzymanie stałej wariancji wyników i gradientów w każdej warstwie.

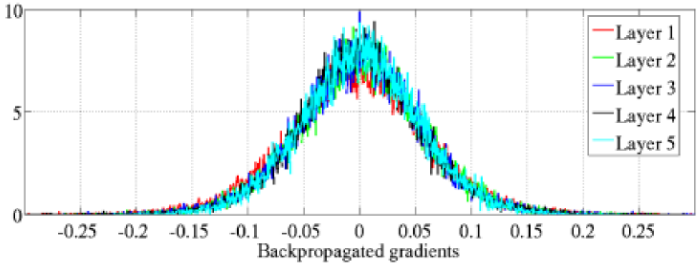
Rozkład wartości aktywacji przykładowej sieci przy inicjalizacji wag — rozkładem normalnym dąży ostatecznie podczas uczenia do rozkładu 0 i 1



Glorot i Bengio zaproponowali, że aby sygnał przepływał prawidłowo, wariancja wyjść każdej warstwy powinna być równa wariancji jej wejść, a gradienty powinny utrzymywać równą wariancję przed i po warstwie podczas propagacji wstecznej. Inicjalizacja Glorota ustawia wagę **każdej warstwy** zgodnie ze wzorem dla rozkładu jednostajnego oraz dla rozkładu normalnego.

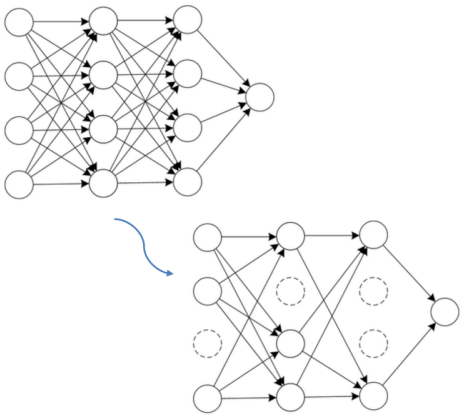






Przeuczenie – rozwiązania

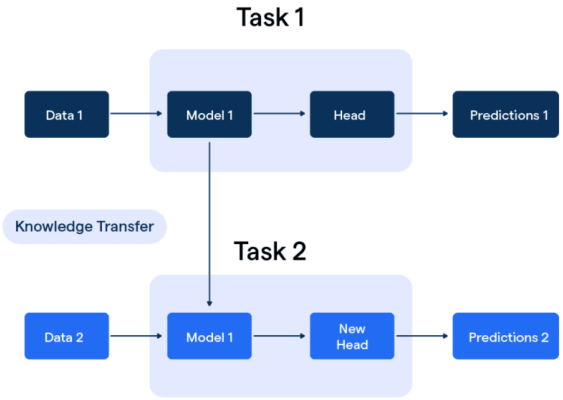
Dropout



- ma na celu zapobieganie nadmiernemu dopasowaniu modelu (podobnie jak regularyzacja)
- polega na "udawaniu" że w każdej iteracji treningowej losowo wybrane neurony w każdej warstwie "przestają istnieć"
- ogranicza nadmierny wpływ pojedynczego neuronu na działanie sieci
- stosowany głównie w dalszych warstwach

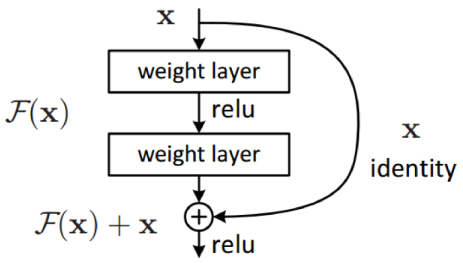
Transfer Learning

W użyciu biznesowym w większości przypadków lepiej użyć już gotowego modelu i dostosować do swojego zastosowania.

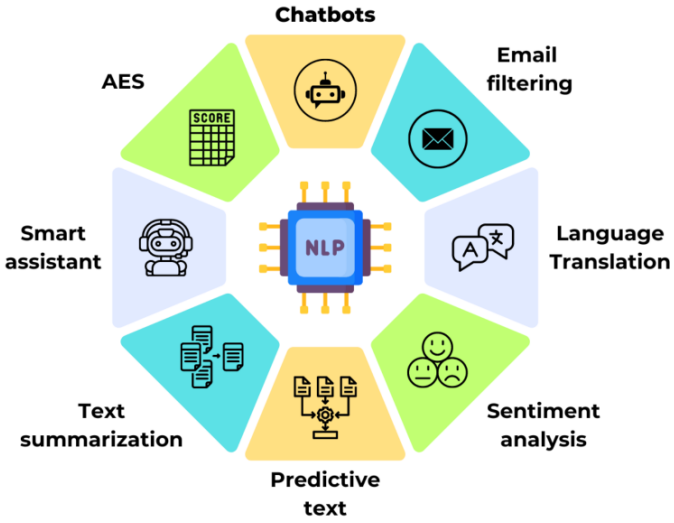


Sieci Rezydualne

Do rozwiązywania prostych problemów sieci są z reguły przygotowane do pomijania warstw

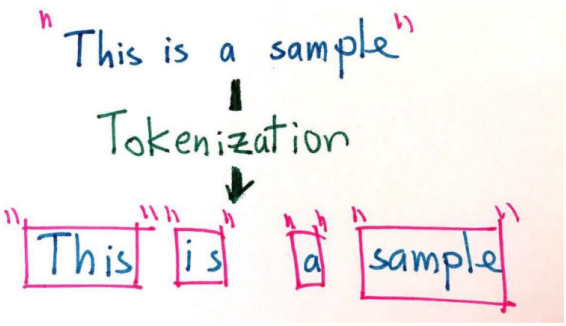


NLP — Natural Language Processing

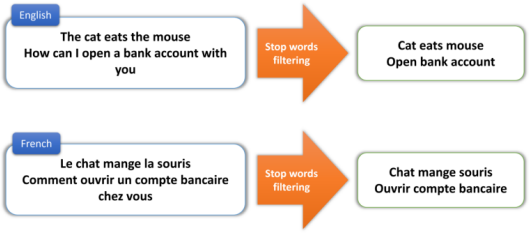


Wstępne przetwarzanie tekstu

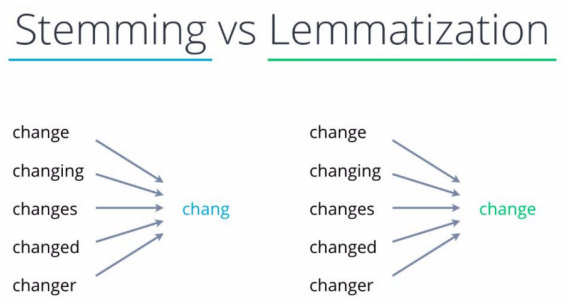
- **Tokenizacja** — przekształcenie dokumentu w listę zindeksowanych elementów języka



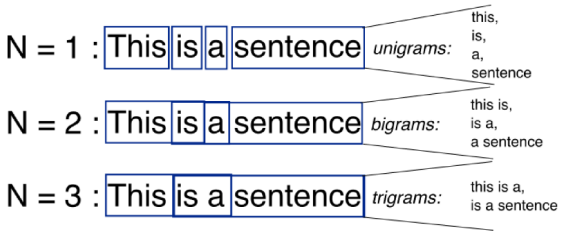
- **Stop words** — często używane słowa, które zwykle nie mają większego znaczenia w zdaniu i nie powodują że zdanie się czymkolwiek charakteryzuje. Warto usuwać wybrane stop words świadomie, np. NOT — zaprzeczenia mogą być istotne dla znaczenia zdań.



- **Stemming** — ucięcie końcówek wyrazu według prostych reguł. Nie uwzględnia gramatyki ani znaczenia słowa. Może prowadzić do form niepoprawnych językowo. Użyteczny przy niewielkim korpusie słów. Np.: biegać, biegł biega -> bieg; komputerowy, komputery -> komput
- **Lematyzacja** — sprowadza słowo do jego poprawnej poprawnej formy podstawowej (lematu). Wymaga analizy językowej — uwzględnia kontekst, część mowy i znaczenie. Dokładniejsza technika ale bardziej złożona obliczeniowo. Np.: szedł, idę, poszli -> iść; ładniejszy, ładna, ładni -> ładny

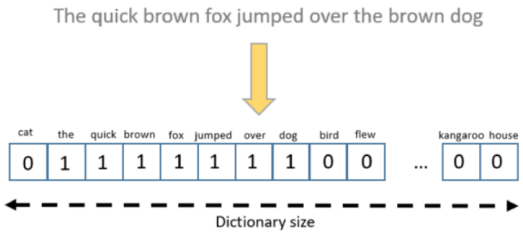


- **Usuwanie interpunkcji** — nie jest zawsze uzasadnione np. przy uczeniu pod kątem odpowiedzi na pytania
- **N-gramy** — analiza zbitek słów — niektóre słowa (tokeny) występują ze sobą tak często, że można je analizować jako jeden token, albo sprawdzić CZY warto je analizować jako jeden token.



Metody reprezentacji języka

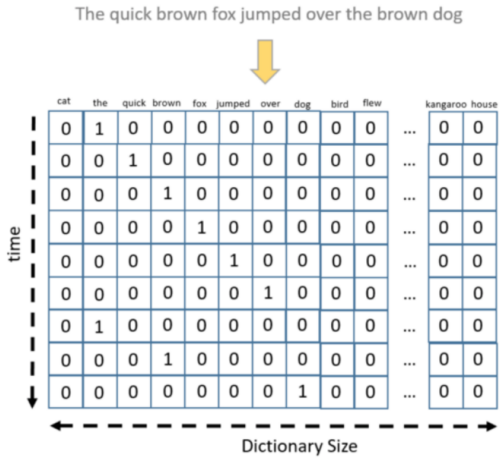
- **Wektoryzacja**
  - bardzo mało pokazuje
  - nie pokazuje kolejności i ilości wystąpień



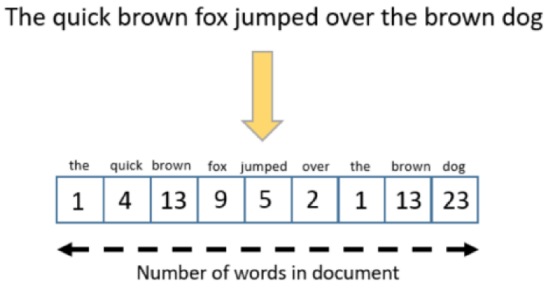
Dokument — całość tekstu który przetwarzamy

Korpus — słowa które wystąpiły w danym dokumencie

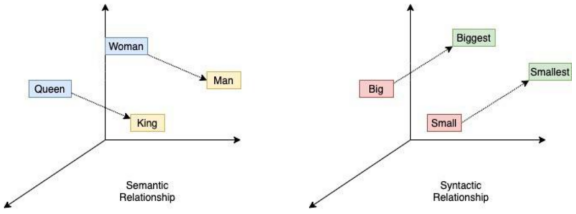
- **One-hot encoding**
  - niewydajne pamięciowo
  - pokazuje kolejność występowania



- **Index-based encoding**
  - aktualnie stosowana
  - z wektora słów tworzenie wektora liczb
  - brak reprezentacji podobieństwo znaczenia słów



- **Word embedding / Word2Vec** — wektorowa reprezentacja, zdolna do odwzorowania podobieństwa wyrazów (semantycznego i syntaktycznego) — podobne w konkretnym kontekście słowa zajmują zbliżoną pozycję w przestrzeni



$$x_{\text{królowa}} = x_{\text{król}} - x_{\text{mężczyzna}} + x_{\text{kobieta}}$$

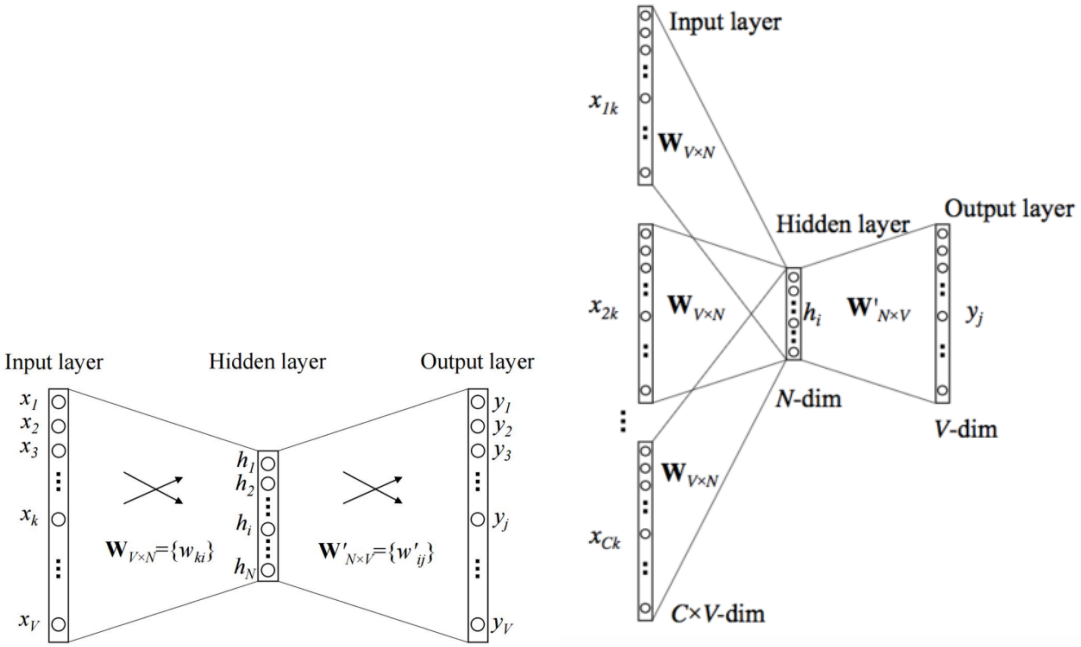
$$y_{\text{królowa}} = y_{\text{król}} - y_{\text{mężczyzna}} + y_{\text{kobieta}}$$

$$z_{\text{królowa}} = z_{\text{król}} - z_{\text{mężczyzna}} + z_{\text{kobieta}}$$

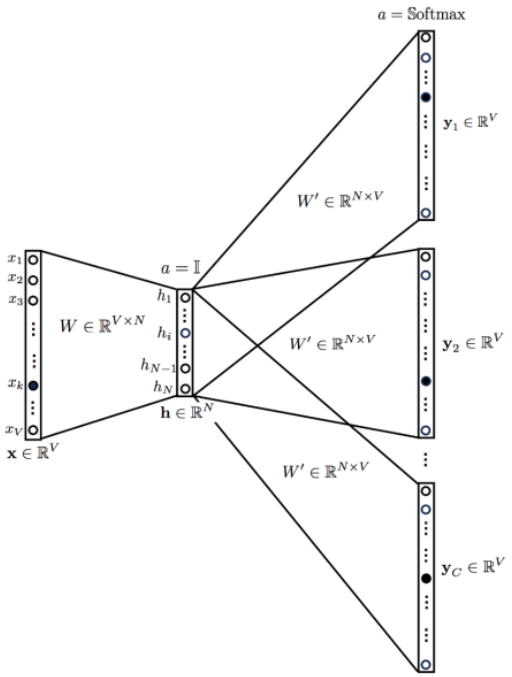
$$v_{\text{królowa}} = v_{\text{król}} - v_{\text{mężczyzna}} + v_{\text{kobieta}}$$

$$v_{\text{bezos}} - v_{\text{amazon}} + v_{\text{tesla}} = v_{\text{musk}}$$

- **Uczenie**
  - **Common Bag of Words** — metoda biorąca kontekst każdego słowa jako wejście i próbująca przewidzieć słowo pasujące do kontekstu



  - **SkipGram** — metoda biorąca słowo docelowe jako wejście i zwracająca na wyjście dla każdego słowa w tym samym zdaniu prawdopodobieństwo że jest słowem poprzedzającym.



Reinforcement Learning

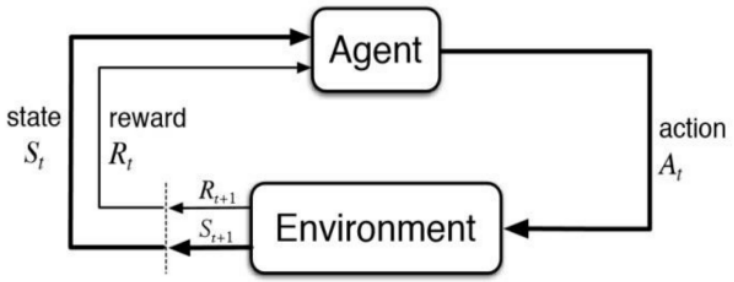
(Filmik - Uczenie alberta aby chodził) `https://www.youtube.com/watch?v=L_4BPjLBF4E`

- Agent wykonuje czynności w środowisku (np grze) w chwili t
- **Nagroda** — skalarna wartość będąca odpowiedzią na akcję wykonaną przez agenta w chwili t, np. 100 punktów za zebranie owocków przez PacMana; celem jest maksymalizacja zebranych nagród

- **Stan** — Informacja o zmianach w środowisku, które zaszły w wyniku akcji wykonanych przez agenta, w kroku t+1 jest to opis warunków, w którym agent podejmie kolejną decyzję
- **Pętla** — W pętli generowane są nagroda i stan, aż do osiągnięcia stanu końcowego, czyli np:
  - uzyskania największej możliwej nagrody
  - uzyskania określonego rezultatu
  - upływu założonego czasu
  - wykonania maksymalnej dozwolonej liczby ruchów
  - porażki agenta
- **Cel** — znalezienie funkcji, która umożliwi agentowi wykonanie odpowiedniej czynności a w każdym stanie s, funkcji wiążącej zasoby S i A. Funkcję tę nazywa się **funkcją polityki** i oznacza π. Nawet przy prostych problemach wyliczenie takiej funkcji jest niezwykle trudne, a nawet niewykonalne. Wykorzystuje się więc metodę szacującą optymalną czynnosc w danym stanie stosując **funckję wartości**
- **Q-learning**
  - Funkcja wartości określa jak wartościowy jest stan, jeśli agent postępuje zgodnie z właściwą dla tego stanu polityką
  - Q-wartość jest rozwinięciem – definiuje użyteczność akcji W POWIĄZANIU z danym stanem, jest to **skumulowana zdyskontowana przyszła nagroda**
  - Agent musi potrafić wyliczyć optymalną Q-wartość, powinien więc przeanalizować wszystkie czynności i wybrać tę dla której Q-wartość będzie największa, jest nadal problem trudny ale SZACOWALNY i NAUCZALNY

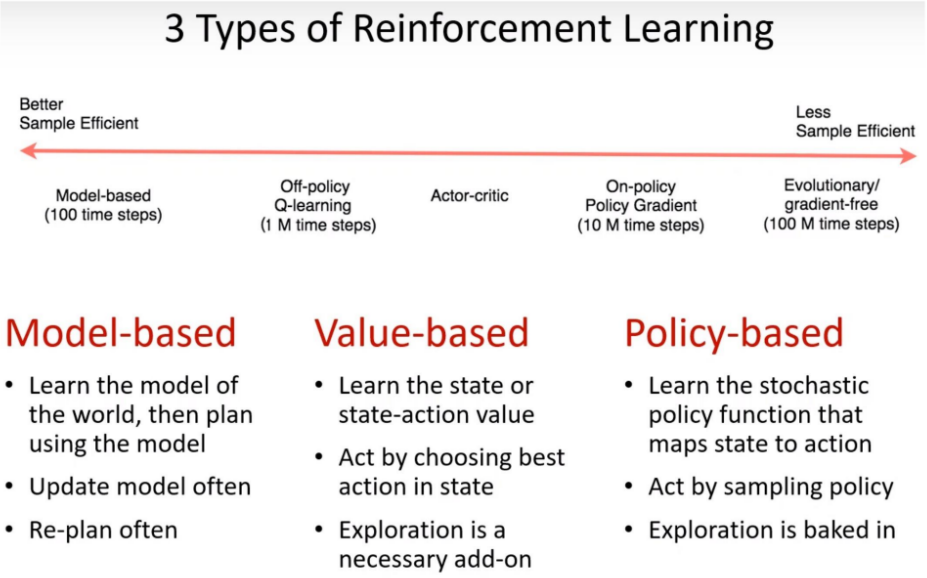
Reinforcement Learning Matematycznie

- proces decyzyjny Markowa — ciąg zdarzeń, w którym prawdopodobieństwo następnego zdarzenia zależy jedynie od wyniku poprzedniego



  - **S** — wszystkie możliwe stany
  - **A** — wszystkie możliwe akcje
  - **R** — rozkład nagrody dla pary (s, a)
  - **P** — prawdopodobieństwo przejścia do stanu s(t+1) dla pary (s,a)
  - **γ** — czynnik dyskontowy (decay), czyli natychmiastowa nagroda jest cenniejsza
- aktualna decyzja zależy tylko i wyłącznie od stanu w chwili T

Inne Rodzaje RL



- **Model-based**
- **Value-based**
- **Policy-based**