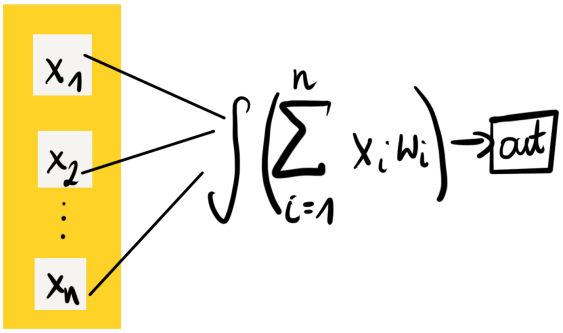
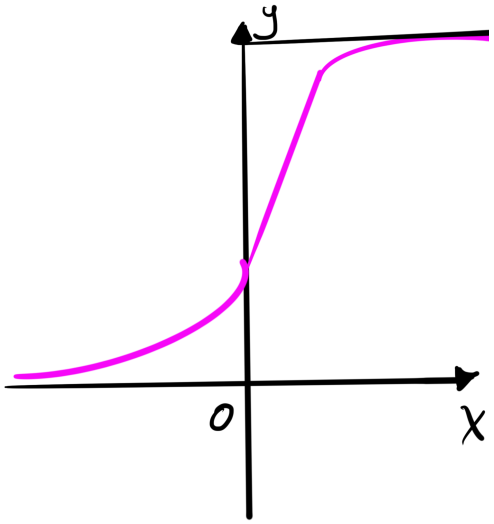
1. Zagadnienie perceptronu prostego.

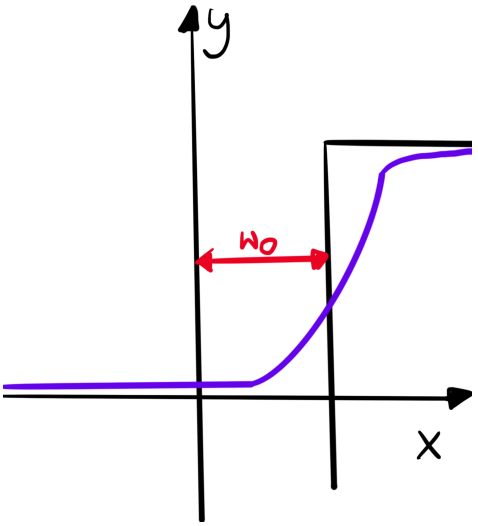
Opis: Perceptron jest prostym matematycznym modelem neuronu. Jest on prostszą siecią jednokierunkową. Składa się on jedynie z warstwy wejściowej i wyjściowej. Jako, że nie istnieją połączenia pomiędzy elementami warstwy wyjściowej, każdy z tych elementów może być traktowany niezależnie jako osobna sieć o m + 1 wejściach i jednym wyjściu.



składa się on z:
n wejść $x_1, x_2, \dots, x_n$
n wag stowarzyszonych z wejściami $w_1,\dots,w_n \in R$ funkcji aktywacji $f: R \to R$
Bias - dodatkowe wejście neuronu z własną wartością wagi
Działanie biasu można wytłumaczyć na podstawie interpretacji geometrycznej dla neuronu o pojedynczym wejściu, dla funkcji sigmoidalnej. Wartości funkcji zostaną przesunięte. Neuron z biasem powinien nauczyć się nawet takich wektorów, których zwykły neuron nie byłby w stanie się nauczyć. Wnioskiem więc jest, że dodanie dodatkowej wagi kosztem zwiększenia ilości obliczeń powodu poprawę własności neuronu.



Funkcja aktywacji neuronu bez biasu

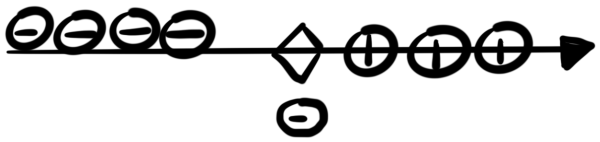


Funkcja aktywacji neuronu z biasem

Interpretacja geometryczna
1d. Jedno wejście x_1, jedna waga w_1 i próg Θ

$$O(x_1) = \begin{cases} -1 & \omega_1 x_1 < \Theta \quad x_1 < \frac{\Theta}{\omega_1} \\ +1 & \omega_1 x_1 > \Theta \quad x_1 \geq \frac{\Theta}{\omega_1} \end{cases}$$

Brzeg rozdzielający jest punktem, który dzieli prostą rzeczywistą na dwie półproste.



2d. Dwa wejścia x_1, x_2, dwie wagi w_1, w_2 i próg Θ

$$O(x_1) = \begin{cases} -1 & \omega_1 x_1 + \omega_2 x_2 < \Theta \Leftrightarrow x_2 < \frac{-\omega_1}{\omega_2} x_1 + \frac{\Theta}{\omega_2} \\ +1 & \omega_1 x_1 + \omega_2 x_2 \geq \Theta \Leftrightarrow x_2 \geq \frac{-\omega_1}{\omega_2} x_1 + \frac{\Theta}{\omega_2} \end{cases}$$

w przypadku 2d brzegiem rozdzielającym jest prosta dzieląca płaszczyznę.

Algorytm uczenia
Celem algorytmu uczenia perceptronu jest znalezienie zestawu wag $w_1,...,w_n$ i progu Θ, takich aby perceptron klasyfikował poprawnie wszystkie lub możliwie najwięcej przykładów uczących.

Podstawowy algorytm nauczania:

1. Ustawiamy wagi w := [0,0, …, 0]
2. Wybieramy przykład uczący(E^k, T^k)
3. Obliczamy $O(E^k) = sgn(w * E^k)$ Jeśli otrzymana liczba jest różna od T^k to uaktualniamy wagi: $w := w + T^k E^k$
4. Jeśli perceptron nie klasyfikuje dobrze wszystkich przypadków, to wracamy do punktu 2.

Algorytm ten jest analogiczny do SPLA, jednak jest pozbawiony losowości. $\eta = 0.5$

PLA - algorytm uczenia z kieszonką.
Idea algorytmu opiera się na zwiększeniu czasu życia wag z każdym z klasyfikowanym poprawnie przykładem. Najlepszy zestaw wag jest przechowywany w kieszonce, aby nie został nadpisany przez przypadkowe zmiany. Po zakończeniu algorytmu zwracany jest rekordowy zestaw. Przy odpowiednio długim działaniu, prawdopodobieństwo, że nieoptymalnym zestaw przeżyje najdłużej, znika do zera.

1. Losujemy wagi i próg w okolicy O, przypisujemy układowi wag zerowy czas życia i zapisujemy go w kieszonce jako rekordzistę.
2. Przechodzimy po losowych przykładach z listy.

3. Dla wybranego przykładu sprawdzamy czy E^j jest dobrze klasyfikowany (ERR = T^j - O = Q)

- jeśli tak to zwiększamy mu czas życia o 1, jeżeli jest to wynik lepszy od rekordzisty zmieniamy to na nowy układ wag. Wracamy do punktu 2.
- jeśli nie to korygujemy wagi i próg
  wi = w_1 + η * ERR * E^i_j
  Θ := Θ - η * ERR
  Nowemu układowi wag przypisujemy 0 jako czas życia i wracamy do punktu 2.

4. Algorytm kończymy po przebiegnięciu odpowiedniej liczby iteracji. Zwracamy najbardziej żywotny zestaw wag.

Zastosowanie:
Może zostać wykorzystane do klasyfikowania zbiorów liniowo separowalnych, aby mógł testować przynależność do więcej niż 2 klas. Należy użyć perceptronu z większą ilością neuronów, w którym klasy zakodowane są jako wyjścia perceptronów.

2. Przeuczenie sieci neuronowej

Przeuczenie sieci neuronowej jest sytuacją, gdy sieć uczy się przykładów “na pamięć”. Do takiej sytuacji dochodzi, gdy ma ona zbyt wiele punktów swobody. Czyli zbyt dużo neuronów do uczenia w porównaniu do skomplikowania problemu i ilości danych. Przeuczona sieć traci zdolność generalizacji.

Generalizacja - zdolność sieci do poprawnej klasyfikacji danych, na których sieć nie była uczona.

Klasycznym objawem przeuczenia sieci jest nieodpowiedni sposób kształtowania się wartości błędów. Jeśli błąd wyznaczony na podstawie zbioru uczącego bardzo szybko się zmniejsza, a błąd walidacyjny początkowo również maleje, a następnie zaczyna wzrastać, możemy zauważyć, że mamy do czynienia z przeuczeniem sieci.

Ochrona przed przeuczeniem:

1. Upraszczanie wag - małe wagi mają mały wpływ na ostateczny model. Gdy małe wagi znikają do zera, model upraszcza się.
2. Kontrola jakości - Polega na sprawdzeniu co jakiś czas jak radzi sobie sieć. Należy obliczyć błąd sieci na zbiorze testowym (innym niż przykłady uczące). Jeśli sieć jest wystarczająco dobra, kończymy uczenie sieci unikając przyuczenia w późniejszych etapach.
3. Uśrednienie modeli - Uczymy wiele sieci o różnych architekturach albo nawet rodzajach. Za ostateczny wynik bierzemy średnią wyników z wielu różnych sieci. Nawet jeśli niektóre się przyuczyły, to ze średniej dostaniemy dobrą generalizacje.
4. Losowanie wag - Gdyby neurony miały identyczne wagi na wejściach i wyjściach, to każdy uczyłby się tego samego, dlatego wagi muszą być zróżnicowane. Jeśli n to liczba wejść do neuronu, to losujemy wagi z rozkładu o odchyleniu standardowym równym $1/\sqrt{n}$ i średniej równej O.

3. Algorytm spadku gradientowego.

Jest to algorytm numeryczny, mający na celu znalezienie minimum zadanej funkcji celu, czyli wyznaczeniu z jego pomocą minimum (lokalnego) funkcji tj. $x \in R^d$, takie że, $f(x) <= f(y)$, gdzie y należy do pewnego otoczenia x.
funkcja $f: R^d \rightarrow R$ jest ciągła i różniczkowalna
(istnieją pochodne cząstkowe $\partial f/\partial x_1 \ldots \partial f/\partial x_d$)
$a^o \in R^d$ to punkt startowy.

Przy kilku założeniach $(\partial f/\partial x_1)(a^o)$ intuicyjnie może być interpretowana jako kierunek w którą stronę funkcja rośnie zmieniając swoją pierwszą współrzędną, gdy pozostałe współrzędne są już ustalone. Mając cały zestaw pochodnych cząstkowych mamy dokładny kierunek, w którym funkcja najszybciej rośnie. Zatem szukając maksimum należy iść w tym kierunku, natomiast szukając minimum w przeciwną stronę.

Algorytm:

1. Rozpocznij w losowym/wybranym $a^{(o)}$
2. Dla każdej współrzędnej g = 1 … d

$a_g^{(m+1)} = a_g^{(m)} - \eta * \partial f/\partial x_g * (a^{(m)})$
gdzie η jest bliskie zeru, dodatnim współczynnikiem uczenia.

3. Powtarzaj krok 2.

Wady:

- Dla dużych zestawów danych wykonuje nadmiarowe obliczenia dla tego samego przykładu uczącego.
- Ponieważ duże zbiory danych mogą nie zmieścić się w pamięci, przez co algorytm może być powolny i trudny do przetworzenia.
- Jako że, do obliczeń bierzemy cały zestaw danych, wagi modelu możemy aktualizować dla nowych danych.
- Może wpaść w minimum lokalne i nie wyjść z niego.

Zalety:

- Teoretyczna analiza wagi wskaźników konwergencji jest łatwa do zrozumienia.

Zastosowanie:
Algorytm spadku gradientowego można zastosować podczas uczenia nadzorowanego w wielowarstwowych sieciach neuronowych. Podczas sprawdzania sumy kwadratów błędów w kroku wstecz możemy użyć tego algorytmu w celu pominięcia losowania w którą stronę ma odbyć się minimalizacja.

4. Sieć Hopfielda

Sieć Hopfielda składa się z n neuronów, przy czym w danej chwili czasu t aktywny jest tylko 1. Każdy neuron wybierany jest z jednakowym prawdopodobieństwem a zmiany stanu neuronu następują w dyskretnych chwilach czasu. Architektura sieci charakteryzuje się

sprężeniem zwrotnym między warstwami sieci, oraz zależnościami dynamicznymi występującymi na każdym etapie działania. Zmiana stanu neuronu przenosi się na całą sieć.

Stan i-tego neuronu może być opisany następująco:

$$y_i(t+1) = f\left(\sum_{i=0}^{n}\left(\omega_{ij} y_j(t) + b_i\right)\right)$$

przy warunku początkowego $y_i(0) = x_j$, gdzie x_j oznacza wektor wejściowy $y_i(t+1)$ stan i-tego neuronu w chwili t+1, w_{ij}- macierz wag, b_j - bias, f -funkcja aktywacji.
f - jako funkcję aktywacji przyjmuje się signum:

$$f(y_j(t)) = \begin{cases} 1, \text{ gdy } \sum_{i=0}^{n} \omega_{ij} \cdot y_j(t) + b_i > 0 \\ 0, \text{ gdy } \sum_{i=0}^{n} \omega_{ij} \cdot y_i(t) + b_i \leq 0 \end{cases}$$

W działaniu sieci Hopfielda wyróżnia się dwa tryby:

- Uczenia - na podstawie dostarczonych do sieci wzorców uczących x_j dobierane są wagi w_{ij}.

Proces uczenia sieci kształtuje obszary przyciągania poszczególnych punktów równowagi, które odpowiadają danym uczącym.

- Odtworzenia - po ustaleniu wartości wag następuje proces przebiegający według zależności, kończących się w określonym minimum lokalnym, w którym y(t + 1) = y(t)

Dynamika Glaubera:
Jeżeli $\mathcal{T}_p << \mathcal{T}_n$ to sieć jest niewielka i można stosować następującą dynamikę:

1. Wylosuj neuron σ_i
2. Przypisz:

$$\sigma_i = \text{sign}\left(\sum_j \omega_{ij}\sigma_j + h_i\right)$$

3. Powtarzaj 1 i 2 do ustabilizowania się sytuacji.

Jako M_i oznaczamy lokalne pole wypadkowe dla jednostki i.

$$M_i = \sum_j \omega_{ij}\sigma_j + h_i$$

Dynamika Little'a
Jeśli $\mathcal{T}_p \simeq \mathcal{T}_n$ to działanie przybliżamy dynamiką synchroniczną.

1. Rozpocznij z losowego $\vec{F}^0$
2. Powtarzaj wielokrotnie
   2.1 Przypisz $\vec{F}^{t+1} := \text{sign}(W \cdot \vec{F}^t + H)$

Gdzie:

W - $[w_{ij}]_{i,j} = 1 \ldots N$ - macierz wag
H - wektor pól zewnętrznych
F^t - wektor spinów w i-tym kroku

Ogólna koncepcja dynamiki w sieciach rekurencyjnych.

1. Wektor zmienia swój spin i wysyła informację do sąsiadów.
2. Po zmianie jest nieaktywny przez pewien okres czasu $\mathcal{T}_r$ - czas refrakcji.
3. Po upływie $\mathcal{T}_N$ neuron może przyjmować i wysyłać impulsy.
4. Przesył impulsu zajmuje pewien okres czasu
   $\mathcal{T}_P$ - czas przesyłu, może znaleźć od rodzaju lub długości krawędzi.

5. PCA - Principal Component Analysis (analiza składowych głównych).

Jest to jedna ze statystycznych metod analizy czynnikowej. Zbiór danych składających się z N obserwacji, z których każda obejmuje K zmiennych, można interpretować jako chmurę N punktów w przestrzeni k-wymiarowej.

Celem PCA jest taki obrót układu współrzędnych, aby zmaksymalizować w pierwszej kolejności wariancję pierwszej współrzędnej, następnie wariację drugiej itd... . Tal przekształcone wartości współrzędnych nazywane są ładunkami wygenerowanych czynników (składowych głównych).

W ten sposób konstruowana jest nowa przestrzeń obserwacji w której najwięcej zmienności wyjaśniają początkowe czynniki.

PCA jest często używana do zmniejszania rozmiaru zbioru danych statystycznych, poprzez odrzucenie ostatnich czynników oraz do tworzenia modeli predykcyjnych.

Jako dane wejściowe podawana jest macierz zawierająca kolejne obserwacje, na podstawie których oznaczane będą główne składowe, czyli wektory bazowe nowej przestrzeni.

6. Splotowe sieci neuronowe - convolutional neural networks.

Jest to klasa głębokich sieci neuronowych, najczęściej stosowanych do analizowania obrazu i wideo, systemów rekomendujących i przetwarzania języka naturalnego.

CNN to uregulowane wersje wieloetapowych perceptronów. Zwykle oznaczają one w pełni połączone sieci, tzn. każdy neuron w jednej warstwie jest połączony ze wszystkimi neuronami w kolejnej warstwie. “Pełna łączność: tych sieci sprawia że, są one podatne na nadmierne dopasowanie danych.

CNN mają inne podejście do regulacji. Wykorzystują hierarchiczny wzorzec w danych i tworzą bardziej złożone wzorce przy użyciu mniejszych i prostszych wzorców.

Sieci splotowe zostały zainspirowane przez biologię. Wzór połączeń między neuronami przypomina budowę kociej kory wzrokowej. Poszczególne neurony korowe reagują na bodźce tylko w ograniczonym obszarze pola widzenia, zwanym jako pole receptywne. Pola recepcyjne różnych neuronów częściowo nakładają się na siebie tak, żę obejmują całe pole widzenia.

CNN wykorzystują stosunkowo mało przetwarzania wstępnego w porównaniu z innymi algorytmami klasyfikacji obrazów. Oznacza to, że sieć uczy się filtrów, które w tradycyjnych algorytmach były tworzone ręcznie. Niezależność od wcześniejszej wiedzy i wysiłku ludzkiego w projektowaniu funkcji stanowi główną zaletę CNN.

Warstwa splotowa jest głównym budulcem CNN. Parametry warstwy składają się z zestawu możliwych do nauczenia się filtrów, które mają małe pole receptywne, ale rozciągają się na pełną głębokość objętości wejściowej. Podczas przejścia w przodu każdy filtr jest splatany na całej szerokości i wysokości objętości wejściowej, obliczając iloczyn skalarny między wejściami filtrami tworząc dwuwymiarową mapę aktywacji tego filtra. W rezultacie sieć uczy się filtrów, które aktywują się, gdy wykryją określony typ funkcji w pewnym położeniu przestrzennym na wejściu.
W przeszłości do rozpoznawania obrazu stosowano tradycyjne modele wielowarstwowych sieci perceptronów (MLP), jednak ze względu na pełną łączność między węzłami, obrazy o wysokiej rozdzielczości przestawały się dobrze skalować. Obraz 1000 x 1000 RGB ma 3 mln pikseli to jest zbyt dużym wynikiem, aby w sposób efektywny przetwarzać go na dużą skalę przy pełnej łączności.

7. Metody walidacji jakości uczenia.

W przypadku walidacji prostej dane uczące są losowo podzielone na dwa rozłączne zbiory.

- próbka ucząca U
- próbka testowa T

W następnym kroku sieć zostaje uczona za pomocą próbki uczącej. Jakość sieci jest badana tylko za pomocą próbki testowej.

jakość := ilość przykładów T sklasyfikowanych poprawnie / wszystkie przykłady T

Uwagi i niebezpieczeństwa:

MInimum dla |U| jest około ¼ całego zbioru, natomiast z drugiej strony |U| nie powinno być większe niż 9/10 całego zbioru. Większy wpływ na wynik może mieć |U| / | U u T| niż zaimplementowany algorytm. Ostatecznie mamy informacje o możliwości generalizacji, ale algorytm uczenia sieci będzie korzystał tylko z ułamka dostępnej wiedzy.

Należy pamiętać, że podając wynik zawsze podajemy proporcję w jakich podzielono zbiór.

W przypadku k-krotnej walidacji krzyżowej dane uczące są losowo podzielone na k rozłącznych i równolicznych zbiorów: T_1, T_2, … , T_k
Kolejno dla i = 1, …, k powtarzamy:

- Uczenie sieci na zbiorze uczącym T_1 u … T_{i-1} u T_{i+1} u T_k
- Testujemy tak nauczoną sieć na danych T_i (na tych danych sieć nie była uczona).
- Zapamiętujemy rezultat jako r_i

Podaje podajemy wszystkie rezultaty r_i lub przynajmniej ich średnią, medianę, maksimum, minimum i odchylenie standardowe.

k-razy dwukrotna walidacja krzyżowa jest odmianą walidacji krzyżowej.
dla i = 1 … k powtarzamy:

- Wykonujemy 2-krotną walidację, za każdym razem losując zbiory treningowe i testowe od nowa.
- Zapamiętujemy wyniki r_{i1}, r_{i2} (po dwa na każdą iterację)
- Zwracamy statystyki dla uzyskanych wyników.
  Leave One Out jest odmianą walidacji krzyżowej, w której k = ilość elementów w T.

  Dla i = 1 … n powtarzamy:
  - Uczymy sieć na zbiorze uczącym T/T_i
  - Testujemy sieć na pozostałych przykładach T_i
  - Zapisujemy wynik r_i (będzie on +1 lub 0)
  - Obliczamy średnią i odchylenie standardowe wyników.

Zaletą tej metody walidacji jest możliwość stosowania jej w przypadku małej ilości danych i zbiorze T.

8.Radialna funkcja bazowa.

Sprawia one, że pojedynczy perceptron zachowuje pewną “pasmowość”, a zmiana kształtu aktywacji perceptron może poprawić działanie. W tym przypadku dane czasami układają się koncentrycznie. Wykorzystywana jest jako funkcja aktywacji.

Norma euklidesa:

$$f(\bar{x}) = \|\bar{x} - \bar{x}_0\|_2 = \sqrt{(x-x_0)^2 + (y+y_0)^2}$$

Funkcja Gaussa”

$$f(x) = \exp \frac{-\|x - x_0\|_2^2}{2\sigma^2}$$

Wykorzystanie:

- Ustalamy liczbę stosowanych funkcji.
- Dla każdej z nich losowo (jeśli mamy informacje o koncentracji danych, to deterministycznie) ustalamy punkty centralny x_0, wariację itd.
- Wyjście podpinamy jako dodatkowe wejście do perceptronu.
- Uczy tak zbudowany rozszerzony perceptron.

9.Normalizacja danych i przygotowanie

Korzystając z sieci neuronowej typu MLP wyposażonej w sigmoidalne funkcje aktywacji konieczne staje się przeprowadzenie skalowania lub minimalizacji danych podawanych na wejściach i wyjściach sieci. Brak odpowiedniej transformacji powoduje poważne zakłócenia w procesie uczenia oraz gorsze właściwości nauczonej sieci.

Oczekiwane wartości wejść i wyjść nie powinny przyjmować zarówno kresu górnego, ja i dolnego funkcji aktywacji (odpowiednio 0 i 1). DLa sigmoidy najlepiej gdy jest to przedział $[0.1, 0.9] \subset [0, 1]$, natomiast dla symetrycznej [-0.9, 0.9]. Dla wejść nie ma tak ścisłych ograniczeń. Mogą być one skalowane do takich samych wartości co wejścia, ważne by były bliskie zeru i miały małą amplitudę.

Normalizacja:
Występowanie dużych różnic w zakresie wartości przyjmowanych przez zmienne może źle wpłynąć na działanie neuronu, zaburzając wpływ poszczególnych wejść. Sieci uczone w oparciu o zmienne o dużych zakresach są również bardziej podatne na utknięcie w minimach lokalnych. Duże wartości zmiennych wejściowych prowadzą także do nasycenia sigmoidalnej funkcji aktywacji, której pochodna w takim przypadku zbiega do wartości zera, blokując tym samym proces uczenia.

10.Sieć Kohonena

Jest to sieć neuronowa uczona w trybie bez nauczyciela w celu wytworzenia niskowymiarowej (przeważnie dwu) zdyskretyzowanej reprezentacji przestrzeni wejściowej. Sieć Kohonena wyróżnia się tym od innej sieci, żę zachowuje odwzorowanie sąsiedztwa przestrzeni wejściowej. Wynikiem działania sieci jest klasyfikacja przestrzeni w sposób

grupujący. Zarówno na przypadki ze zbioru uczącego, jak i wszystkie inne wprowadzenia po procesie uczenia.

Dane:

- Dane uczące, numeryczne w przestrzeni R^m
- Graf G = (V, E)

Algorytm uczenia:

1. Przypisujemy neuronom małe losowe wagi (pozycje w R^d) $p_1,..,p_d$
2. Dla t = 1 … T wykonujemy
   2.1 Losujemy przykład P.
   2.2 Znajdujemy jednostkę v ∈ V, w której zestaw wag v(v) leży najbliżej P.
   2.3 Dla neuronu zwycięzcy i wszystkich jego sąsiadów s(v) wykonujemy:
   $\boldsymbol{\pi}$(w) = $\boldsymbol{\pi}$(w) + L(t)(P-$\boldsymbol{\pi}$(w)
   gdzie L(t) maleje od 1 do 0 wraz z postępem algorytmu np: L = 1 - (t -1)/ T

Celem tej sieci jest zmapowanie grafu na dane uczące, tak aby był rozmieszczony równomiernie, a sąsiednie wierzchołki leżały niedaleko.