Przegląd modeli sieci neuronowych

Modele sieci neuronowych

Na poprzednich stronach zapoznałeś się z modelem Perceptron, siecią Feedforward i siecią Hopfield. Dowiedziałeś się, że różnice między modelami leżą w ich architekturze, kodowaniu i przypominaniu. Naszym celem jest teraz przedstawienie pełnego obrazu tych i innych modeli sieci neuronowych. W dalszych Częściach pokażemy szczegóły i implementacje niektórych sieci. Modele, które pokrótce omówimy to: perceptron, Hopfield, Adalina, propagacja wsteczna sprzężenia do przodu, dwukierunkowa pamięć asocjacyjna, stan mózgu w pudełku, neokognitron, rozmyta pamięć asocjacyjna, ART1 i ART2. Implementacje niektórych z nich w C++ oraz rola logiki rozmytej w niektórych zostaną omówione w kolejnych rozdziałach. Na razie nasza dyskusja będzie dotyczyła cech wyróżniających sieć neuronową. Podążymy za tym opisem niektórych modeli.

Warstwy w sieci neuronowej

Sieć neuronowa ma swoje neurony podzielone na podgrupy lub pola, a elementy w każdej podgrupie są umieszczone w rzędzie lub kolumnie na diagramie przedstawiającym sieć. Każda podgrupa jest następnie określana jako warstwa neuronów w sieci. Bardzo wiele modeli sieci neuronowych ma dwie warstwy, sporo ma jedną warstwę, a niektóre mają trzy lub więcej warstw. W niektórych sieciach, takich jak sieć Feed-forward z propagacją wsteczną, możliwych jest wiele dodatkowych, tak zwanych warstw ukrytych. Gdy sieć ma pojedynczą warstwę, sygnały wejściowe są odbierane w tej warstwie, przetwarzanie jest wykonywane przez jej neurony, a wyjście jest generowane w tej warstwie. Gdy obecna jest więcej niż jedna warstwa, pierwsze pole jest przeznaczone dla neuronów, które dostarczają sygnały wejściowe do neuronów w następnej warstwie. Każda sieć ma warstwę neuronów wejściowych, ale w większości sieci jedynym celem tych neuronów jest dostarczanie danych wejściowych do następnej warstwy neuronów. Jednak w niektórych sieciach istnieją połączenia zwrotne lub połączenia powtarzające się, więc neurony w warstwie wejściowej mogą również wykonywać pewne przetwarzanie. W sieci Hopfield, którą widziałeś wcześniej, warstwy wejściowe i wyjściowe są takie same. Jeśli pomiędzy warstwą wejściową i wyjściową znajduje się jakakolwiek warstwa, można ją ogólnie nazwać warstwą ukrytą lub warstwą o specjalnej nazwie na cześć badacza, który zaproponował jej włączenie w celu uzyskania określonej wydajności sieci. Przykładami są warstwy Grossberga i Kohonena. Liczba ukrytych warstw nie jest ograniczona, z wyjątkiem zakresu problemu, który jest rozwiązywany przez sieć neuronową.

Warstwa jest również nazywana polem. Następnie różne warstwy mogą być oznaczone jako pole A, pole B itd. lub w skrócie FA, FB.

Sieć jednowarstwowa

Sieć neuronowa z pojedynczą warstwą może również przetwarzać niektóre ważne aplikacje, takie jak implementacje układów scalonych lub sterowanie linią montażową. Najpowszechniejszą funkcją różnych modeli sieci neuronowych jest rozpoznawanie wzorców. Ale jedna sieć, zwana Brain-State-in-a-Box, która jest jednowarstwową siecią neuronową, może wykonywać uzupełnianie wzorców. Adaline to sieć z polami A i B neuronów, ale agregacja lub przetwarzanie sygnałów wejściowych jest wykonywane tylko przez neurony pola B. Sieć Hopfield to jednowarstwowa sieć neuronowa. Sieć Hopfielda tworzy skojarzenia między różnymi wzorcami (heteroasocjacja) lub kojarzy wzorzec ze sobą (autoasocjacja). Możesz to scharakteryzować jako umiejętność rozpoznawania danego wzoru. Pomysł postrzegania tego jako przypadku rozpoznawania wzorca staje się bardziej istotny, jeśli wzorzec jest prezentowany z pewnym szumem, co oznacza, że we wzorze występuje niewielka deformacja i jeśli sieć jest w stanie powiązać go z prawidłowym wzorcem. Perceptron technicznie ma dwie warstwy, ale

ma tylko jedną grupę ciężarków. Dlatego nadal nazywamy ją siecią jednowarstwową. Druga warstwa składa się wyłącznie z neuronu wyjściowego, a pierwsza warstwa składa się z neuronów, które odbierają dane wejściowe. Ponadto neurony w tej samej warstwie, w tym przypadku warstwie wejściowej, nie są ze sobą połączone, to znaczy między dwoma neuronami w tej samej warstwie nie powstają żadne połączenia. Z drugiej strony w sieci Hopfield nie ma oddzielnej warstwy wyjściowej, a co za tym idzie jest to sieć stricte jednowarstwowa. Ponadto wszystkie neurony są ze sobą w pełni połączone. Poświęćmy więcej czasu na jednowarstwowy model Perceptronu i omówmy jego ograniczenia, a tym samym zmotywujmy do badania sieci wielowarstwowych.

Funkcja XOR i perceptron

Zdolność Perceptronu do oceny funkcji została zakwestionowana, gdy Minsky i Papert udowodnili, że prosta funkcja, taka jak XOR (funkcja logiczna wykluczająca lub), nie może być poprawnie oceniona przez Perceptron. Funkcja logiczna XOR, f(A,B), wygląda następująco:

A	B	f(A,B)= XOR(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

Podsumowując zachowanie XOR, jeśli oba wejścia mają tę samą wartość, wyjście to 0, w przeciwnym razie wyjście to 1. Minsky i Papert pokazali, że nie można wymyślić właściwego zestawu wag dla neuronów w pojedynczym warstwę prostego Perceptronu do oceny funkcji XOR. Powodem tego jest to, że taki Perceptron, jeden z pojedynczą warstwą neuronów, wymaga oceny funkcji, aby można ją było liniowo oddzielić za pomocą wartości funkcji. W dalszej części wyjaśniona jest koncepcja liniowej separowalności. Ale najpierw pokażmy, dlaczego prosty perceptron nie potrafi obliczyć tej funkcji. Ponieważ istnieją dwa argumenty dla funkcji XOR, w warstwie wejściowej byłyby dwa neurony, a ponieważ wartością funkcji jest jedna liczba, byłby jeden neuron wyjściowy. Dlatego potrzebne są dwie wagi w_1 i w_2 oraz wartość progowa . Przyjrzyjmy się teraz warunkom, jakie muszą spełniać w i aby wyjścia odpowiadające danym wejściom były takie jak dla funkcji XOR. Najpierw wyjście powinno wynosić 0, jeśli wejścia to 0 i 0. Aktywacja działa jako 0. Aby uzyskać wyjście 0, potrzebujesz 0 < . To twój pierwszy warunek. Tabela pokazuje ten i dwa inne warunki, których potrzebujesz i dlaczego.

Input	Activation	Output	Needed Condition
0, 0	0	0	$0 < \theta$
1, 0	w_1	1	$w_1 > \theta$
0, 1	w_2	1	$w_2 > \theta$
1, 1	$w_1 + w_2$	0	$w_1 + w_2 < \theta$

Z pierwszych trzech warunków można wywnioskować, że suma dwóch wag musi być większa niż , co samo w sobie musi być dodatnie. Wiersz 4 jest niezgodny z wierszami 1, 2 i 3, ponieważ wiersz 4 wymaga, aby suma dwóch wag była mniejsza niż . Potwierdza to twierdzenie, że nie jest możliwe obliczenie funkcji XOR za pomocą prostego perceptronu. Geometrycznie, przyczyną tego niepowodzenia jest to, że wejścia (0, 1) i (1, 0), z którymi chcesz uzyskać wyjście 1, są usytuowane po przekątnej naprzeciwko siebie, gdy są wykreślane jako punkty na płaszczyźnie, jak pokazano poniżej na schemacie wyjścia (1=T, 0=F):

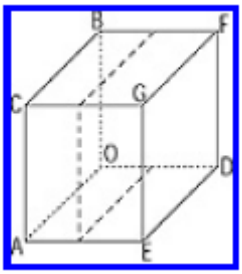
F T

T F

Nie możesz oddzielić T i F linią prostą. Oznacza to, że nie można narysować linii w płaszczyźnie w taki sposób, aby ani (1, 1) ->F ani (0, 0)->F nie znajdowały się po tej samej stronie linii co (0, 1) -> T i (1, 0) -> T.

Oddzielność liniowa

Oznacza to, że liniowo separowalny jest rodzaj bariery liniowej lub separatora - linia w płaszczyźnie lub płaszczyzna w przestrzeni trójwymiarowej lub hiperpłaszczyzna w wyższych wymiarach - tak, aby zbiór danych wejściowych, które powodują powstanie jednej wartości dla funkcji, wszystkie leżą po jednej stronie tej bariery, podczas gdy po drugiej stronie leżą dane wejściowe, które nie dają tej wartości dla funkcji. Hiperpłaszczyzna to powierzchnia w wyższym wymiarze, ale z równaniem liniowym określającym ją w podobny sposób, w jaki definiuje się linię w płaszczyźnie i płaszczyznę w przestrzeni trójwymiarowej. Aby koncepcja była nieco jaśniejsza, rozważ problem, który jest podobny, ale, podkreślmy, nie taki sam, jak problem XOR. Wyobraź sobie sześcian o długości 1 jednostki dla każdej z jego krawędzi i leżący w dodatnim oktancie w układzie współrzędnych xyz-prostokątnych z jednym rogiem na początku. Pozostałe narożniki lub wierzchołki znajdują się w punktach o współrzędnych (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0) i (1, 1, 1). Nazwij początek O i siedem punktów wymienionych odpowiednio jako A, B, C, D, E, F i G. Wtedy dowolne dwie przeciwległe do siebie ściany można rozdzielić liniowo, ponieważ można zdefiniować płaszczyznę oddzielającą jako płaszczyznę w połowie odległości między tymi dwiema ścianami, a także równoległą do tych dwóch ścian. Rozważmy na przykład ściany określone przez zbiór punktów O, A, B i C oraz przez zbiór punktów D, E, F i G. Są one równoległe i oddalone od siebie o 1 jednostkę, jak widać na rysunku 5.1 . Płaszczyzna oddzielająca te dwie ściany może być postrzegana jako jedna z wielu możliwych płaszczyzn — dowolna płaszczyzna pomiędzy nimi i równoległa do nich. Jednym z przykładów, dla uproszczenia, jest płaszczyzna, która przechodzi przez punkty (1/2, 0, 0), (1/2, 0, 1), (1/2, 1, 0) i (1/2, 1, 1). Oczywiście wystarczy określić tylko trzy z tych czterech punktów, ponieważ płaszczyzna jest jednoznacznie określona przez trzy punkty, z których nie wszystkie znajdują się na tej samej linii. Zatem jeśli pierwszy zestaw punktów odpowiada wartości, powiedzmy, +1 dla funkcji, a drugi zestaw wartości –1, to jednowarstwowy perceptron może określić, za pomocą jakiegoś algorytmu uczącego, prawidłowe wagi dla połączeń, nawet jeśli zaczniesz od wag początkowo równych 0.

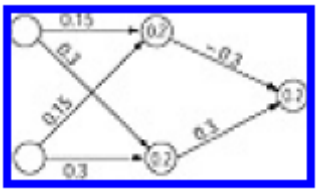


Rozważmy zbiór punktów O, A, F i G. Ten zbiór punktów nie może być liniowo oddzielony od innych wierzchołków sześcianu. W takim przypadku byłoby to niemożliwe w przypadku jednowarstwowej . Perceptron do określenia właściwych wag dla neuronów w ocenie rodzaju funkcji, o której mówiliśmy.

Drugie spojrzenie na funkcję XOR: Perceptron wielowarstwowy

Wprowadzając zestaw kaskadowych Perceptronów, otrzymujesz sieć Perceptron z warstwą wejściową, warstwą środkową lub ukrytą oraz warstwą wyjściową. Zobaczysz, że wielowarstwowy Perceptron

może ocenić funkcję XOR, a także inne funkcje logiczne (AND, OR, MAJORITY itp.). Brak rozdzielności, o której mówiliśmy wcześniej, można przezwyciężyć dzięki drugiemu etapowi, że tak powiem, wag połączeń. Potrzebujesz dwóch neuronów w warstwie wejściowej i jednego w warstwie wyjściowej. Połóżmy ukrytą warstwę z dwoma neuronami. Niech w11, w12, w21 i w22 będą wagami połączeń od neuronów wejściowych do neuronów warstwy ukrytej. Niech v1, v2 będą wagami połączeń między neuronami warstwy ukrytej a neuronem zewnętrznym. Wybierzemy w (wagi) i wartości progowe θ_1 θ_2 w neuronach warstwy ukrytej, tak aby wejście (0, 0) generowało wektor wyjściowy (0, 0), a wektor wejściowy (1, 1) generuje (1, 1), podczas gdy wejścia (1, 0) i (0, 1) generują (0, 1) jako wyjście warstwy ukrytej. Dane wejściowe do neuronów warstwy wyjściowej będą pochodzić ze zbioru {(0, 0), (1, 1), (0, 1)}. Te trzy wektory można rozdzielić (0, 0) i (1, 1) po jednej stronie linii oddzielającej, a (0, 1) po drugiej stronie. Dobierzemy √s (wagi) i τ , wartość progową na neuronie wyjściowym tak, aby wejścia (0,0) i (1,1) powodowały wyjście 0 dla sieci, a wyjście 1 było spowodowane przez wejście (0, 1). Układ sieci w obrębie etykiet wag i wartości progowych wewnątrz węzłów reprezentujących warstwę ukrytą i neurony wyjściowe pokazano na rysunku 5.1a. W tabeli 2 przedstawiono wyniki działania tej sieci.



Input	Hidden Layer Activations	Hidden Layer Outputs	Output Neuron activaton	Output of network
(0, 0)	(0, 0)	(0, 0)	0	0
(1, 1)	(0.3, 0.6)	(1, 1)	0	0
(0, 1)	(0.15, 0.3)	(0, 1)	0.3	1
(1, 0)	(0.15, 0.3)	(0, 1)	0.3	1

Z Tabeli jasno wynika, że powyższy perceptron z ukrytą warstwą oblicza

Funkcja XOR pomyślnie.

Uwaga: Aktywacja powinna przekroczyć wartość progową aktywacji neuronu. Gdy wyjście neuronu jest pokazane jako 0, dzieje się tak dlatego, że wewnętrzna aktywacja tego neuronu nie osiągnęła wartości progowej.

Przykład sześcianu ponownie odwiedzony

Wróćmy do przykładu sześcianu z wierzchołkami w początku O i punktami oznaczonymi A, B, C, D, E, F i G. Załóżmy, że zbiór wierzchołków O, A, F i G daje wartość 1 dla funkcji do oceny, a pozostałe wierzchołki dają –1. Te dwa zestawy nie są liniowo rozdzielone, jak wspomniano wcześniej. Prosty Perceptron nie może ocenić tej funkcji. Czy dodanie kolejnej warstwy neuronów może pomóc? Odpowiedź brzmi tak. Jaka byłaby rola tej dodatkowej warstwy? Odpowiedź jest taka, że wykona ostateczne przetwarzanie problemu po tym, jak poprzednia warstwa wykonała wstępne przetwarzanie. Może to zrobić dwie separacje w tym sensie, że zestaw ośmiu wierzchołków może być rozdzielony - lub podzielony - na trzy separowalne podzbiory. Jeśli to partycjonowanie może również pomóc w zbieraniu w każdym podzbiorze, takim jak wierzchołki, czyli te, które mapują tę samą wartość funkcji, sieć odniesie sukces w swoim zadaniu oceny funkcji, gdy agregacja i progowanie zostaną wykonane w neuronie wyjściowym.

Strategia

Tak więc strategia polega najpierw na rozważeniu zbioru wierzchołków, które dają wartość +1 dla funkcji i określeniu minimalnej liczby podzbiorów, które można zidentyfikować jako dające się oddzielić od pozostałych wierzchołków. Jest oczywiste, że skoro wierzchołki O i A leżą na jednej krawędzi sześcianu, mogą tworzyć jeden podzbiór, który można rozdzielić. Pozostałe dwa wierzchołki, mianowicie F i jeden dla G, które odpowiadają wartości +1 dla funkcji, mogą tworzyć drugi podzbiór, który również można oddzielić. Nie musimy zawracać sobie głowy ostatnimi czterema wierzchołkami z punktu widzenia dalszego podziału tego podzbioru. Oczywiste jest, że jedna nowa warstwa trzech neuronów, z których jeden uruchamia się dla wejść odpowiadających wierzchołkom O i A, jeden dla F i G, a trzeci dla pozostałych, ułatwi wtedy prawidłową ocenę funkcji w neuron wyjściowy.

Detale

W tabeli 3 wymieniono wierzchołki i ich współrzędne wraz z flagą wskazującą, do którego podzbioru w podziale należy wierzchołek. Zauważ, że możesz myśleć o działaniu perceptronu wielowarstwowego jako o ocenie przecięcia i sumy podzbiorów liniowo rozłącznych.

Vertex	Coordinates	Subset
O	(0,0,0)	1
A	(0,0,1)	1
B	(0,1,0)	2
C	(0,1,1)	2
D	(1,0,0)	2
E	(1,0,1)	2
F	(1,1,0)	3
G	(1,1,1)	3

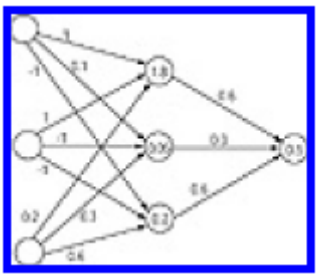
Sieć, która jest dwuwarstwowym perceptronem, czyli dwiema warstwami wag, ma trzy neurony w pierwszej warstwie i jeden wyjściowy w drugiej warstwie. Pamiętaj, że liczymy te warstwy, w których neurony agregują wchodzące do nich sygnały za pomocą wag połączeń. Pierwsza warstwa z trzema neuronami jest ogólnie określana jako warstwa ukryta, ponieważ druga warstwa nie jest ukryta i znajduje się po prawej stronie układu sieci neuronowej. Tabela 4 podaje przykład wag, których można użyć do połączeń między neuronami wejściowymi a neuronami warstwy ukrytej. Istnieją trzy neurony wejściowe, po jednym dla każdej współrzędnej wierzchołka sześcianu.

Input Neuron #	Hidden Layer Neuron #	Connection Weight
1	1	1
1	2	0.1
1	3	-1
2	1	1
2	2	-1
2	3	-1
3	1	0.2
3	2	0.3
3	3	0.6

Teraz podajemy w tabeli 5 wagi połączeń między trzema neuronami warstwy ukrytej a neuronem wyjściowym.

Hidden Layer Neuron #	Connection Weight
1	0.6
3	0.3
3	0.6

Nie jest jasne, czy te wagi wystarczą, czy nie. Aby określić aktywację neuronów warstwy ukrytej, potrzebujesz tych wag, a także potrzebujesz wartości progowej dla każdego neuronu, który przetwarza. Neuron warstwy ukrytej uruchomi się, to znaczy wygeneruje 1, jeśli ważona suma odbieranych sygnałów jest większa niż wartość progowa. Jeśli neuron wyjściowy odpala, wartość funkcji przyjmuje +1, a jeśli nie odpala, wartość funkcji wynosi –1. Tabela 6 podaje wartości progowe. Rysunek b przedstawia sieć neuronową z wagami połączeń i wartościami progowymi.



Layer	Neuron	Threshold Value
hidden	1	1.8
hidden	2	0.05
hidden	3	-0.2
output	1	0.5

Wydajność Perceptronu

Kiedy wprowadzisz współrzędne wierzchołka G, który ma 1 dla każdej współrzędnej, pierwszy neuron warstwy ukrytej agreguje te dane wejściowe i otrzymuje wartość 2,2. Ponieważ wartość 2,2 jest większa niż wartość progowa pierwszego neuronu w warstwie ukrytej, neuron ten odpala, a jego wyjście 1 staje się wejściem do neuronu wyjściowego przy połączeniu o wadze 0,6. Ale potrzebujesz również aktywacji innych neuronów warstwy ukrytej. Opiszmy wydajność ze współrzędnymi G jako wejściami do sieci. Opisuje to Tabela 7.

Vertex/ Coordinates	Hidden Layer	Weighted Sum	Comment	Activation	Contribution to Output	Sum
G:1,1,1	1	2.2	>1.8	1	0.6	
	2	-0.8	<0.05	0	0	
	3	-1.4	<-0.2	0	0	0.6

Suma ważona na neuronie wyjściowym wynosi 0,6 i jest większa niż wartość progowa 0,5. W związku z tym neuron wyjściowy odpala, a na wierzchołku G funkcja ma wartość +1. Tabela 8 przedstawia

wydajność sieci z pozostałymi wierzchołkami sześcianu. Zauważysz, że sieć oblicza wartość +1 w wierzchołkach O, A, F i G oraz a –1 w pozostałych.

	Hidden Layer Neuron#	Weighted Sum	Comment	Activation	Contribution to Output	Sum
O :0, 0, 0	1	0	<1.8	0	0	
	2	0	<0.05	0	0	
	3	0	>-0.2	1	0.6	0.6*
A :0, 0, 1	1	0.2	<1.8	0	0	
	2	0.3	>0.05	1	0.3	
	3	0.6	>-0.2	1	0.6	0.9*
B :0, 1, 0	1	1	<1.8	0	0	
	2	-1	<0.05	0	0	
	3	-1	<-0.2	0	0	0
C :0, 1, 1	1	1.2	<1.8	0	0	
	2	0.2	>0.05	1	0.3	
	3	-0.4	<-0.2	0	0	0.3
D :1, 0, 0	1	1	<1.8	0	0	
	2	.1	>0.05	1	0.3	
	3	-1	<-0.2	0	0	0.3
E :1, 0, 1	1	1.2	<1.8	0	0	
	2	0.4	>0.05	1	0.3	
	3	-0.4	<-0.2	0	0	0.3
F :1, 1, 0	1	2	>1.8	1	0.6	
	2	-0.9	<0.05	0	0	
	3	-2	<-0.2	0	0	0.6*

Inne sieci dwuwarstwowe

Wiele ważnych modeli sieci neuronowych ma dwie warstwy. Jednym z przykładów jest sieć feedforward z propagacją wsteczną w swojej najprostszej formie. Paradygmat ART1 Grossberga i Carpentera wykorzystuje sieć dwuwarstwową. Sieć kontrpropagacji ma warstwę Kohonena, po której następuje warstwa Grossberga. Dwukierunkowa pamięć asocjacyjna (BAM), maszyna Boltzmana, rozmyta pamięć asocjacyjna i czasowa pamięć asocjacyjna to inne sieci dwuwarstwowe. W przypadku autoasocjacji sieć jednowarstwowa może wykonać zadanie, ale w przypadku heteroskojarzenia lub innych podobnych mapowań potrzebna jest co najmniej sieć dwuwarstwowa. Wkrótce podamy więcej szczegółów na temat tych modeli.

Wiele warstw sieci

Neocognitron Kunihiko Fukushimy, znany z identyfikowania odręcznych znaków, jest przykładem sieci z kilkoma warstwami. Niektóre wcześniej wymienione sieci mogą być również wielowarstwowe dzięki dodaniu większej liczby ukrytych warstw. Możliwe jest również połączenie dwóch lub więcej sieci

neuronowych w jedną sieć, tworząc odpowiednie połączenia między warstwami jednej podsieci z warstwami innych. To z pewnością stworzyłoby sieć wielowarstwową.

Połączenia między warstwami

Widziałeś już pewną różnicę w sposobie tworzenia połączeń między neuronami w sieci neuronowej. W sieci Hopfielda każdy neuron był połączony ze sobą w tej samej warstwie, która była obecna w sieci. W Perceptronie neurony w tej samej warstwie nie były ze sobą połączone, ale połączenia były między neuronami w jednej warstwie a tymi w następnej warstwie. W pierwszym przypadku połączenia są opisane jako boczne. W tym drugim przypadku połączenia są przesyłane dalej, a sygnały przesyłane są dalej w sieci. Istnieją również dwie inne możliwości. Wszystkie neurony w dowolnej warstwie mogą mieć dodatkowe połączenia, przy czym każdy neuron jest połączony ze sobą. Druga możliwość polega na tym, że istnieją połączenia między neuronami w jednej warstwie a neuronami w warstwie poprzedniej, w którym to przypadku następuje zarówno przekazywanie sygnału do przodu, jak i do tyłu. Dzieje się tak, jeśli sprzężenie zwrotne jest funkcją sieci. Rodzaj układu neuronów sieciowych oraz rodzaj połączeń między neuronami stanowi architekturę danego modelu sieci neuronowej.

Instar i Outstar

Outstar i instar to terminy zdefiniowane przez Stephena Grossberga dla sposobów patrzenia na neurony w sieci. Neuron w sieci innych neuronów otrzymuje dużą liczbę sygnałów wejściowych spoza granic neuronu. To jest jak gwiazda promieniująca do wewnątrz, stąd termin instar. Ponadto neuron może wysyłać swoje dane wyjściowe do wielu innych miejsc docelowych w sieci. W ten sposób zachowuje się jak gwiazdor. Każdy neuron jest więc jednocześnie instar i outstar. W stadium początkowym otrzymuje bodźce z innych części sieci lub spoza sieci. Zauważ, że neurony w warstwie wejściowej sieci mają przede wszystkim połączenia z dala od neuronów w następnej warstwie, a zatem zachowują się głównie jak gwiazdy zewnętrzne. Neurony w warstwie wyjściowej mają wiele połączeń, które do niej docierają, a zatem zachowują się głównie jak stadia rozwojowe. Sieć neuronowa wykonuje swoją pracę poprzez nieustanną interakcję stadiów rozwojowych i zewnętrznych. Warstwa instar może stanowić warstwę konkurencyjną w sieci. Outstar można również opisać jako węzeł źródłowy z niektórymi powiązanymi węzłami ujścia, do których źródło zasila. Grossberg identyfikuje wejście źródłowe z bodźcem warunkowym, a wejścia pochłaniające z bodźcami nieuwarunkowanymi. Sieć kontrpropagacji Roberta Hechta-Nielsena to model zbudowany z instars i outstars.

Masy na połączeniach

Przypisania wag do połączeń między neuronami nie tylko wskazują siłę sygnału, który jest podawany do agregacji, ale także rodzaj interakcji między dwoma neuronami. Rodzaj interakcji to współpraca lub rywalizacja. Typ spółdzielczy sugeruje waga dodatnia, a konkurencja waga ujemna na połączeniu. Dodatni związek ciężaru jest przeznaczony do tego, co nazywamy wzbudzeniem, podczas gdy ujemny związek ciężaru jest określany jako hamowanie.

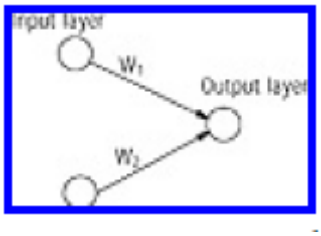
Inicjalizacja wag

Inicjowanie struktury wagi sieci jest częścią tak zwanej fazy kodowania operacji sieciowej. Istnieje kilka algorytmów kodowania, różniących się modelem i aplikacją. Być może odniosłeś wrażenie, że macierze wag użyte w przykładach szczegółowo omawianych do tej pory zostały ustalone arbitralnie; lub jeśli istnieje metoda ich ustawienia, nie zostaniesz poinformowany, co to jest. Możliwe jest rozpoczęcie od losowo wybranych wartości wag i odpowiednie dostosowanie wag w miarę przechodzenia sieci przez kolejne iteracje. To też by to ułatwiło. Na przykład, w przypadku nadzorowanego treningu, jeśli błąd między pożądanym a obliczonym wynikiem jest używany jako kryterium przy dostosowywaniu wag,

można równie dobrze ustawić początkowe wagi na zero, a proces uczenia zajmie się resztą. Poniższy mały przykład ilustruje ten punkt.

Mały przykład

Załóżmy, że masz sieć z dwoma neuronami wejściowymi i jednym wyjściowym, z połączeniami do przodu między neuronami wejściowymi a wyjściowym, jak pokazano na rysunku . Sieć jest wymagana do wyprowadzenia 1 dla wzorców wejściowych (1, 0) i (1, 1) oraz wartości 0 dla (0, 1) i (0, 0). Istnieją tylko dwie wagi połączeń w_1 i w_2.



Ustawmy początkowo obie wagi na 0, ale potrzebna jest również funkcja progu. Użyjmy następującej funkcji progowej, która różni się nieco od tej użytej w poprzednim przykładzie:

$$f(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases}$$

Powodem zmodyfikowania tej funkcji jest to, że jeśli f(x) ma wartość 1, gdy x = 0, to bez względu na wagi, wynik będzie działał do 1 z wejściem (0, 0). Uniemożliwia to uzyskanie poprawnego obliczenia dowolnej funkcji, która przyjmuje wartość 0 dla argumentu (0, 0). Teraz musimy wiedzieć, w jaki sposób dostosowujemy ciężary. Procedura, którą zastosowalibyśmy w tym przykładzie, jest następująca.

• Jeśli wydruk z wzorcem wejściowym (a, b) jest zgodny z oczekiwaniami, nie dostosowuj wag.

• Jeśli wynik z wzorcem wejściowym (a, b) jest mniejszy niż powinien, zwiększ każdy z w1 i w2 o 1.

• Jeśli wyjście z wzorcem wejściowym (a, b) jest większe niż powinno być, odejmij 1 od w1, jeśli iloczyn aw1 jest mniejszy niż 1, i podobnie dostosuj w_2.

Tabela 9 pokazuje, co się dzieje, gdy postępujemy zgodnie z tymi procedurami i przy jakich wartościach ustalają się wagi.

step	w_1	w_2	a	b	activation	output	comment
1	0	0	1	1	0	0	desired output is 1; increment both w's
2	1	1	1	1	2	1	output is what it should be
3	1	1	1	0	1	1	output is what it should be
4	1	1	0	1	1	1	output is 1; it should be 0.
5							subtract 1 from w_2
6	1	0	0	1	0	0	output is what it should be
7	1	0	0	0	0	0	output is what it should be
8	1	0	1	1	1	1	output is what it should be
9	1	0	1	0	1	1	output is what it should be

Tabela.9 pokazuje, że wektor wag sieci zmienił się z początkowego (0, 0) do końcowego wektora wag (1, 0) w ośmiu iteracjach. Ten przykład nie dotyczy sieci do dopasowywania wzorców. Jeśli się nad tym zastanowisz, zdasz sobie sprawę, że sieć jest zaprojektowana do odpalania, jeśli pierwsza cyfra we wzorcu to 1, a nie inaczej. Analogią do tego rodzaju problemu jest ustalenie, czy dany obraz zawiera określony obiekt w określonej części obrazu, np. kropka powinna wystąpić w literze i. Jeśli początkowe wagi są dobierane w sposób nieco ostrożny i aby nadać im szczególne znaczenie, wówczas szybkość działania można zwiększyć w sensie osiągnięcia zbieżności przy mniejszej liczbie iteracji niż w innym przypadku. Dlatego ważne są algorytmy kodowania. Przedstawiamy teraz niektóre algorytmy kodowania.

Inicjowanie wag dla sieci autoasocjacyjnych

Rozważmy sieć, która ma kojarzyć każdy wzorzec wejściowy ze sobą i która otrzymuje wzorce binarne jako dane wejściowe. Wykonaj dwubiegunowe mapowanie na wzorcu wejściowym. Oznacza to, że zamień każde 0 na –1. Nazwij odwzorowany wzorzec wektorem x, gdy zapiszemy go jako wektor kolumnowy. Ta transpozycja, ten sam wektor zapisany jako wektor wierszowy, to xT. Macierz zamówienia o wielkości x otrzymasz podczas tworzenia produktu xxT. Uzyskaj podobne macierze dla innych wzorców, które chcesz przechowywać w sieci. Dodaj te macierze, aby otrzymać macierz wag, która będzie używana na początku, tak jak to zrobiliśmy w Części 4. Proces ten można opisać następującym równaniem:

$$\mathbf{W} = \varsigma_i \mathbf{x}_i \mathbf{x}_i^{\mathrm{T}}$$

Inicjalizacja wagi dla sieci heteroasocjacyjnych

Rozważmy sieć, która ma skojarzyć jeden wzorzec wejściowy z innym wzorcem i pobiera wzorce binarne jako dane wejściowe. Wykonaj dwubiegunowe mapowanie na wzorcu wejściowym. Oznacza to, że zamień każde 0 na –1. Nazwij odwzorowany wzór wektorem x, gdy zapiszesz go jako wektor kolumnowy. Uzyskaj podobne dwubiegunowe mapowanie dla odpowiedniego skojarzonego wzorca. Nazwij to tak. Otrzymasz macierz o rozmiarze x na rozmiar y, gdy utworzysz iloczyn xyT. Uzyskaj podobne macierze dla innych wzorców, które chcesz przechowywać w sieci. Dodaj te macierze, aby uzyskać macierz wag, która będzie używana na początku. Poniższe równanie odwzorowuje ten proces:

$$\mathbf{W} = \varsigma_i \mathbf{x}_i \mathbf{y}_i^{\mathrm{T}}$$

W jednym z wielu interesujących paradygmatów, które napotykasz w modelach i teorii sieci neuronowych, zwycięzca bierze wszystko. Cóż, jeśli z tłumu neuronów w danej warstwie wyłoni się jeden zwycięzca, musi istnieć konkurencja. Ponieważ w takiej rywalizacji każdy jest dla siebie, w tym przypadku każdy neuron dla siebie, konieczne byłoby posiadanie połączeń bocznych, które wskazują na tę okoliczność. Połączenia boczne z dowolnego neuronu do innych powinny mieć ujemną wagę. Lub neuron o największej aktywacji jest uważany za zwycięzcę i tylko jego wagi są modyfikowane w procesie uczenia, pozostawiając wagi innych bez zmian. Zwycięzca bierze wszystko oznacza, że tylko jeden neuron w tej warstwie jest aktywowany, a pozostałe nie. Może się to zdarzyć w warstwie ukrytej lub w warstwie wyjściowej. W innej sytuacji, gdy konkretna kategoria danych wejściowych ma zostać zidentyfikowana spośród kilku grup danych wejściowych, musi istnieć podzbiór neuronów, które są przeznaczone do obserwowania, jak to się dzieje. W tym przypadku w przypadku neuronów odległych wzrasta inhibicja, a dla sąsiednich w takim podzbiorze wzrasta pobudzenie. Zwrot na środku, poza otoczeniem opisuje zjawisko odległego hamowania i bliskiego wzbudzenia. Wagi są również podstawowymi składnikami sieci neuronowej, ponieważ z jednej strony odzwierciedlają pamięć przechowywaną przez sieć, az drugiej podstawę uczenia się i treningu.

Wejścia

Widzieliście, że wzajemnie ortogonalne lub prawie ortogonalne wzorce są wymagane jako stabilne przechowywane wzorce dla sieci Hopfielda, które omawialiśmy wcześniej dla dopasowania wzorców. Podobne ograniczenia występują również w przypadku innych sieci neuronowych. Czasami nie jest to ograniczenie, ale cel modelu sprawia, że pewien rodzaj danych wejściowych staje się naturalny. Z pewnością w kontekście klasyfikacji wzorców binarne wzorce wejściowe ułatwiają konfigurowanie problemu. Sygnały binarne, bipolarne i analogowe to odmiany wejść. Sieci, które akceptują sygnały analogowe jako wejścia, są przeznaczone dla modeli ciągłych, a te, które wymagają wejść binarnych lub bipolarnych, są przeznaczone dla modeli dyskretnych. Wejścia binarne mogą być podawane do sieci dla modeli ciągłych, ale sygnały analogowe nie mogą być wprowadzane do sieci dla modeli dyskretnych (chyba że są one rozmyte). Gdy możliwości wejściowe są dyskretne lub analogowe, a możliwości modelu są dyskretne lub ciągłe, potencjalnie istnieją cztery sytuacje, ale jedna z nich, w której wejścia analogowe są brane pod uwagę dla modelu dyskretnego, jest nie do utrzymania. Przykładem modelu ciągłego jest sytuacja, w której sieć ma dostosować kąt, o jaki należy obrócić kierownicę ciężarówki, aby zaparkować ciężarówkę w miejscu parkingowym. Jeżeli sieć ma rozpoznawać znaki alfabetu, sposób dyskretyzacji znaku pozwala na zastosowanie modelu dyskretnego. Jakie są rodzaje danych wejściowych dla problemów, takich jak przetwarzanie obrazu lub analiza pisma ręcznego? Pamiętając, że sztuczne neurony, jako elementy przetwarzające, agregują swoje dane wejściowe za pomocą wag połączeń, a neuron wyjściowy używa funkcji progowej, wiesz, że dane wejściowe muszą być numeryczne. Znak odręczny można nałożyć na siatkę, a dane wejściowe

mogą składać się z komórek w każdym rzędzie siatki, w których występuje część znaku. Innymi słowy, dane wejściowe odpowiadające jednemu znakowi będą zestawem sekwencji binarnych lub w skali szarości, zawierającym jedną sekwencję dla każdego wiersza siatki. 1 w określonej pozycji w sekwencji dla wiersza oznacza, że odpowiedni piksel jest obecny (czarny) w tej części siatki, podczas gdy 0 oznacza, że tak nie jest. Rozmiar siatki musi być wystarczająco duży, aby pomieścić największą badaną postać, a także najbardziej złożone funkcje.

Wyjścia

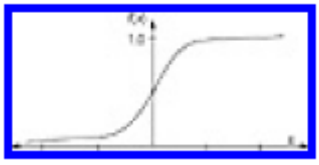
Dane wyjściowe z niektórych sieci neuronowych to wzór przestrzenny, który może zawierać wzór bitowy, w niektórych wartość funkcji binarnej, a w innych sygnał analogowy. Rodzaj mapowania przeznaczony dla wejść determinuje oczywiście rodzaj wyjść. Wynikiem może być klasyfikowanie danych wejściowych lub znajdowanie powiązań między wzorcami o tym samym wymiarze co dane wejściowe. Funkcje progowe dokonują ostatecznego odwzorowania aktywacji neuronów wyjściowych na wyjścia sieciowe. Ale wyniki z pojedynczego cyklu działania sieci neuronowej mogą nie być końcowymi wynikami, ponieważ będziesz iterować sieć w kolejnych cyklach działania, dopóki nie zobaczysz zbieżności. Jeśli konwergencja wydaje się możliwa, ale zajmuje dużo czasu i wysiłku, to znaczy, jeśli nauka jest zbyt wolna, możesz przypisać poziom tolerancji i zadowolić się osiągnięciem przez sieć bliskiej zbieżności.

Funkcja progowa

Wyjście dowolnego neuronu jest wynikiem ewentualnego progowania jego wewnętrznej aktywacji, która z kolei jest ważoną sumą wejść neuronu. Progowanie czasami jest wykonywane w celu zmniejszenia aktywacji i odwzorowania jej na sensowne wyjście dla problemu, a czasami w celu dodania błędu. Progowanie (skalowanie) jest ważne w przypadku sieci wielowarstwowych, aby zachować znaczący zakres operacji każdej warstwy. Najczęściej używaną funkcją progową jest funkcja sigmoidalna. Można użyć funkcji krokowej, funkcji rampy lub po prostu funkcji liniowej, tak jak wtedy, gdy po prostu dodajesz bias do aktywacji. Funkcja sigmoidalna dokonuje odwzorowania aktywacji na przedział [0, 1]. Równania podano poniżej dla różnych wspomnianych funkcji progowych.

Funkcja sigmoidalna

Więcej niż jedna funkcja nosi nazwę funkcji sigmoidalnej. Różnią się one formułą i zakresem. Wszystkie mają wykres podobny do rozciągniętej litery s. Poniżej podajemy dwie takie funkcje. Pierwsza to hiperboliczna funkcja tangensa z wartościami w (–1, 1). Druga to funkcja logistyczna i ma wartości od 0 do 1. Dlatego wybierasz tę, która pasuje do żądanego zakresu. Wykres sigmoidalnej funkcji logistycznej przedstawiono poniżej



1. $f(x) = \tanh(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$

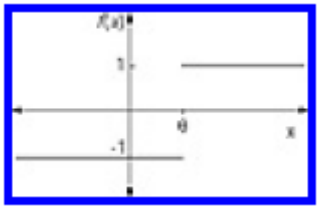
2. $f(x) = 1 / (1+ e^{-x})$

Zauważ, że pierwsza funkcja tutaj, hiperboliczna funkcja tangensa, może być również zapisana jako $1 - 2e^{-x} / (e^{x} + e^{-x})$ po dodaniu i odjęciu e-x od licznika, a następnie uproszczeniu. Jeśli teraz pomnożysz w drugim członie zarówno licznik, jak i mianownik przez ex, otrzymasz $1 - 2/ (e^{2x} + 1)$. Gdy x zbliża się do

- , ta funkcja osiąga -1, a gdy x zbliża się do + , idzie do +1. Z drugiej strony, druga funkcja tutaj, sigmoidalna funkcja logistyczna, idzie do 0, gdy x zbliża się do - i do +1, gdy x zbliża się do + . Możesz to zobaczyć, jeśli przepiszesz 1 / (1+ e^{-x}) jako 1 - 1 / (1+ e^{x}), po manipulacjach podobnych do tych powyżej. Możesz myśleć o równaniu 1 jako bipolarnym odpowiedniku równania binarnego 2. Obie funkcje mają ten sam kształt. Rysunek 5.3 to wykres sigmoidalnej funkcji logistycznej (numer 2 na poprzedniej liście).

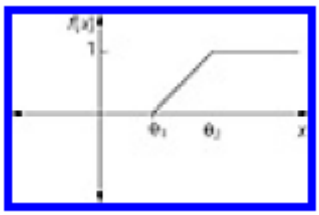
Funkcja kroku

Funkcja kroku jest również często używana jako funkcja progowa. Funkcja zaczyna się od 0 i pozostaje na lewo od pewnej wartości progowej θ. Przeskok do 1 następuje dla wartości funkcji na prawo od , a następnie funkcja pozostaje na poziomie 1. Ogólnie rzecz biorąc, funkcja skokowa może mieć skończoną liczbę punktów, w których występują skoki równej lub nierównej wielkości. Gdy skoki są równe iw wielu punktach, wykres będzie przypominał schody. Interesuje nas funkcja kroku, która przechodzi od 0 do 1 w jednym kroku, gdy tylko argument przekroczy wartość progową θ. Możesz również mieć dwie wartości inne niż 0 i 1 w definiowaniu zakresu wartości takiej funkcji kroku. Wykres funkcji kroku przedstawiono na rysunku.



Funkcja rampy

Aby w prosty sposób opisać funkcję rampy, najpierw rozważ funkcję step, która w pewnym momencie wykonuje skok z 0 do 1. Zamiast pozwolić mu na taki nagły skok w pewnym momencie, pozwól mu stopniowo zyskiwać na wartości, wzdłuż linii prostej (wygląda jak rampa) w skończonym przedziale sięgającym od początkowego 0 do końcowego 1. W ten sposób otrzymujesz funkcja rampy. Możesz myśleć o funkcji rampy jako odcinkowo liniowej aproksymacji sigmoidy. Wykres funkcji rampy jest przedstawiony na rysunku.



Funkcja liniowa

Funkcja liniowa to prosta funkcja podana równaniem o postaci:

$f(x) = \alpha x + \beta$

Gdy α = 1, zastosowanie tej funkcji progowej sprowadza się do prostego dodania obciążenia równego β sumie wejść.

Aplikacje

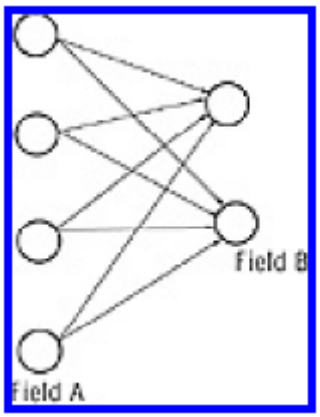
Jak pokrótce wskazano wcześniej, obszary zastosowań obejmują zazwyczaj auto- i heteroskojarzenia, rozpoznawanie wzorców, kompresję danych, uzupełnianie danych, filtrowanie sygnału, przetwarzanie

obrazu, prognozowanie, rozpoznawanie pisma ręcznego i optymalizację. Rodzaj połączeń w sieci oraz rodzaj zastosowanego algorytmu uczenia należy dobrać odpowiednio do aplikacji. Na przykład sieć z połączeniami bocznymi może wykonywać autoskojarzenia, podczas gdy typ feed-forward może wykonywać prognozowanie.

Niektóre modele sieci neuronowych

Adaline i Madaline

Adaline to akronim od adaptacyjnego elementu liniowego, dzięki Bernardowi Widrowowi i Marcianowi Hoffowi. Jest podobny do perceptronu. Wejściami są liczby rzeczywiste z przedziału [–1,+1], a uczenie opiera się na kryterium minimalizacji błędu średniokwadratowego. Adaline ma dużą pojemność do przechowywania wzorów. Madaline oznacza wiele Adalines i jest szeroko stosowaną siecią neuronową. Składa się z neuronów pola A i pola B, a każdy neuron pola A ma jedno połączenie z każdym neuronem pola B. Rysunek przedstawia schemat Madaline.



Propagacja wsteczna

Algorytm uczenia wstecznej propagacji błędów do uczenia sieci sprzężenia do przodu został opracowany przez Paula Werbosa, a później przez Parkera oraz Rummelharta i McClellanda. Ten rodzaj konfiguracji sieci jest najczęściej używany ze względu na łatwość jej uczenia. Szacuje się, że ponad 80% wszystkich opracowywanych projektów sieci neuronowych wykorzystuje wsteczną propagację błędów. W propagacji wstecznej istnieją dwie fazy cyklu uczenia się, jedna służy do propagowania wzorca wejściowego przez sieć, a druga do dostosowania wyjścia poprzez zmianę wag w sieci. To sygnały błędu są propagowane wstecznie w działaniu sieci do ukrytych warstw. Część sygnału błędu, którą odbiera neuron warstwy ukrytej w tym procesie, jest oszacowaniem wkładu konkretnego neuronu w błąd wyjściowy. Dostosowanie na tej podstawie wagi połączeń, błędu kwadratowego lub jakiejś innej metryki jest zmniejszane w każdym cyklu i ostatecznie minimalizowane, jeśli to możliwe.

Dwukierunkowa pamięć asocjacyjna

Dwukierunkowa pamięć asocjacyjna (BAM) i inne modele opisane w tej sekcji zostały opracowane przez Barta Kosko. BAM to sieć z połączeniami sprzężenia zwrotnego z warstwy wyjściowej do warstwy wejściowej. Wiąże element zestawu wzorców wejściowych z elementem zestawu wzorców wyjściowych, który jest najbliższy, a zatem wykonuje heteroskojarzenie. Wzory mogą mieć wartości binarne lub dwubiegunowe. Jeżeli znane są wszystkie możliwe wzorce wejściowe, macierz wag połączeń można wyznaczyć jako sumę macierzy otrzymanych przez wzięcie iloczynu macierzy wektora wejściowego (jako wektora kolumnowego) z jego transpozycją (zapisaną jako wektor wierszowy). Wzór uzyskany z warstwy wyjściowej w jednym cyklu działania jest przesyłany z powrotem do warstwy

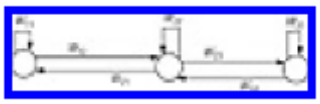
wejściowej na początku następnego cyklu. Proces trwa, dopóki sieć nie ustabilizuje się na wszystkich wzorcach wejściowych. Osiągnięty w ten sposób stan stabilny jest określany jako rezonans, pojęcie stosowane w adaptacyjnej teorii rezonansu. Na rysunku w Części 8 znajdziesz układ węzłów reprezentujących neurony w sieci BAM oraz połączenia między nimi. Są dwa pola neuronów. Sieć jest w pełni połączona z połączeniami zwrotnymi i połączeniami do przodu. Nie ma połączeń bocznych ani powtarzających się. Pamięci skojarzeniowe rozmyte są podobne do wspomnień skojarzeniowych dwukierunkowych, z tą różnicą, że ustala się związek między wzorcami rozmytymi.

Pamięć czasowa asocjacyjna

Innym typem pamięci skojarzeniowej jest skojarzeniowa pamięć czasowa. Amari, pionier w dziedzinie sieci neuronowych, skonstruował model Temporal Associative Memory, który posiada sprzężenia zwrotne między warstwą wejściową i wyjściową. Mocną stroną tego modelu jest to, że może przechowywać i pobierać wzory czasoprzestrzenne. Przykładem wzorca czasoprzestrzennego jest kształt fali segmentu mowy.

Stan mózgu w pudełku

Wprowadzona przez Jamesa Andersona i innych, ta sieć różni się od jednowarstwowej w pełni połączonej sieci Hopfield tym, że Brain-State-in-a-Box wykorzystuje również to, co nazywamy połączeniami powtarzalnymi. Każdy neuron ma połączenie ze sobą. W przypadku dostępnych wzorców docelowych używana jest zmodyfikowana reguła uczenia Hebba. Dopasowanie do wagi połączenia jest proporcjonalne do iloczynu żądanej mocy wyjściowej i błędu w obliczonej mocy wyjściowej. Więcej na temat uczenia się Hebbowskiego znajdziesz w rozdziale 6. Sieć ta jest biegła w tolerancji szumów i może wykonać uzupełnianie wzorców. Rysunek przedstawia sieć Brain-State-in-a-Box.



Co jest w imieniu?

Bardziej jak to, co jest w pudełku? Załóżmy, że znajdziesz następujące elementy: jest kwadratowe pole, a jego rogi to lokalizacje, w których ma się znajdować jednostka. Istota nie znajduje się w jednym z rogów, ale w pewnym momencie wewnątrz pudełka. Następna pozycja podmiotu jest określana poprzez obliczenie zmiany w każdej współrzędnej pozycji, zgodnie z macierzą wag i funkcją zgniatania. Ten proces jest powtarzany, dopóki istota nie usadowi się w jakiejś pozycji. Wybór matrycy wag jest taki, że gdy jednostka osiągnie róg kwadratowego pudełka, jej pozycja jest stabilna i nie ma już żadnego ruchu. Można by się domyślać, że istota w końcu osiada w rogu najbliższym jej początkowej pozycji w pudełku. Mówi się, że tego rodzaju przykład jest powodem, dla którego modelowi nadano nazwę Brain-State-in-a-Box. Jego mocną stroną jest to, że reprezentuje przekształcenia liniowe. Za pomocą tego modelu można osiągnąć pewien rodzaj skojarzeń wzorów. Jeśli niekompletny wzorzec jest powiązany z kompletnym wzorcem, byłby to przykład autoasocjacji.

Kontrpropagacja

Jest to model sieci neuronowej opracowany przez Roberta Hechta-Nielsena, który ma jedną lub dwie dodatkowe warstwy między warstwą wejściową i wyjściową. Jeśli jest jeden, warstwa środkowa to warstwa Grossberga z kilkoma gwiazdami zewnętrznymi. W drugim przypadku warstwa Kohonena lub warstwa samoorganizująca się następuje po warstwie wejściowej, a następnie następuje warstwa

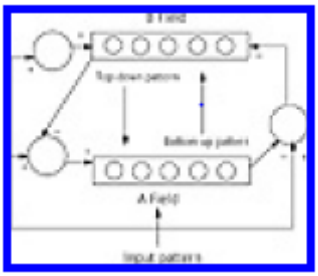
Grossberga z gwiazdami zewnętrznymi. Model wyróżnia się znacznym skróceniem czasu szkolenia. Dzięki temu modelowi zyskujesz narzędzie, które działa jak tabela przeglądowa.

Neokognitron

W porównaniu do wszystkich innych modeli sieci neuronowych, Neocognitron Fukushimy jest bardziej złożony i ambitny. Pokazuje zalety sieci wielowarstwowej. Neocognitron to jeden z najlepszych modeli rozpoznawania odręcznych symboli. Używanych jest wiele par warstw zwanych warstwą S dla warstwy prostej i warstwą C dla warstwy złożonej. W każdej warstwie S znajduje się kilka płaszczyzn zawierających proste komórki. Podobnie w każdej warstwie C znajduje się taka sama liczba płaszczyzn zawierających złożone komórki. Warstwa wejściowa nie ma takiego układu i są jak warstwa wejściowa w każdej innej sieci neuronowej. Liczba płaszczyzn prostych komórek i złożonych komórek w parze warstw S i C jest taka sama, płaszczyzny te są sparowane, a złożone komórki płaszczyzn przetwarzają dane wyjściowe prostych komórek płaszczyzny. Proste komórki są szkolone tak, aby odpowiedź prostej komórki odpowiadała określonej części obrazu wejściowego. Jeśli ta sama część obrazu występuje z pewnymi zniekształceniami pod względem skalowania lub obrotu, reaguje na to inny zestaw prostych komórek. Złożone dane wyjściowe komórek wskazują, że jakaś prosta komórka, której odpowiadają, została uruchomiona. Podczas gdy proste komórki odpowiadają na to, co znajduje się w ciągłym regionie obrazu, złożone komórki odpowiadają na podstawie większego regionu. Gdy proces jest kontynuowany do warstwy wyjściowej, składnik Clay warstwy wyjściowej odpowiada, odpowiadając całemu obrazowi przedstawionemu na początku w warstwie wejściowej.

Adaptacyjna teoria rezonansu

ART1 to pierwszy model adaptacyjnej teorii rezonansu dla sieci neuronowych opracowany przez Gail Carpenter i Stephena Grossberga. Teoria ta została opracowana w celu rozwiązania dylematu stabilność-plastyczność. Sieć ma być na tyle plastyczna, by nauczyć się ważnego wzorca. Ale jednocześnie powinna pozostać stabilna, gdy w pamięci krótkotrwałej napotka kilka zniekształconych wersji tego samego wzorca. Model ART1 ma neurony pola A i B, wzmocnienie i reset, jak pokazano na rysunku.



Pomiędzy neuronami pól A i B występują połączenia top-down i bottom-up. Neurony w polu B mają połączenia boczne oraz połączenia rekurencyjne. Oznacza to, że każdy neuron w tym polu jest połączony z każdym innym neuronem w tym polu, w tym z samym sobą, oprócz połączeń z neuronami w polu A. Wejście zewnętrzne (lub sygnał oddolny), sygnał odgórny i sygnał Wzmocnienie stanowią trzy elementy zbioru, z których co najmniej dwa powinny stanowić +1 do odpalenia neuronu w polu A. Nazywa się to zasadą dwóch trzecich. Dlatego początkowo wzmocnienie byłoby ustawione na +1. Pomysł jednego zwycięzcy znajduje zastosowanie również w polu B. Zysk nie przyczyniłby się do fazy odgórnej; w rzeczywistości będzie to hamować. Reguła dwóch trzecich pomaga w dążeniu do stabilności po uzyskaniu rezonansu lub równowagi. Parametr czujności służy do określenia resetu parametrów. Parametr czujności odpowiada, w jakim stopniu można przewidzieć kategorię

rezonansową. Część systemu, która zawiera wzmocnienie, nazywana jest podsystemem uwagi, podczas gdy reszta, część zawierająca reset, nazywana jest podsystemem orientującym. Działanie odgórne odpowiada podsystemowi orientacji, a działanie oddolne odnosi się do podsystemu uwagi. W ART1 podejmowana jest próba klasyfikacji wzorca wejściowego w odniesieniu do przechowywanych wzorców, a jeśli się nie powiedzie, generowana jest nowa przechowywana klasyfikacja. Szkolenie nie jest nadzorowane. Istnieją dwie wersje treningu: wolna i szybka. Różnią się one stopniem, w jakim wagi otrzymują czas na osiągnięcie ostatecznych wartości. Powolny trening jest regulowany równaniami różniczkowymi, a szybki trening równaniami algebraicznymi. ART2 jest analogowym odpowiednikiem ART1, który jest przeznaczony do dyskretnych przypadków. Są to samoorganizujące się sieci neuronowe, jak można wywnioskować z faktu, że trening jest obecny, ale nienadzorowany. Model ART3 służy do rozpoznawania zakodowanego wzorca poprzez równoległe wyszukiwanie i został opracowany przez Carpentera i Grossberga. Próbuje naśladować aktywność przekaźników chemicznych w mózgu podczas czegoś, co można uznać za równoległe poszukiwanie rozpoznawania wzorców.

Podsumowanie

Omówiono podstawowe pojęcia dotyczące warstw sieci neuronowej, połączeń, wag, wejść i wyjść. Szczegółowo podano przykład, w jaki sposób dodanie kolejnej warstwy neuronów w sieci może rozwiązać problem, którego bez tego nie dałoby się rozwiązać. W skrócie przedstawiono kilka modeli sieci neuronowych. Uczenie się i szkolenie, które stanowią podstawę zachowania sieci neuronowych, nie zostały tutaj uwzględnione, ale zostaną omówione w następnej części.