UNIWERSYTET MIKOŁAJA KOPERNIKA

MARCIN BUCZKOWSKI

ZASTOSOWANIE IDEI GEOMETRYCZNYCH W TEORI UKŁADÓW UCZĄCYCH SIĘ

Praca magisterska wykonana
w Katedrze Metod Komputerowych
Wydziału Fizyki i Astronomii
pod kierunkiem
prof. dr. hab. Włodzisława Ducha

TORUŃ 1995

Spis treści

1 Wstęp

W pracy tej zostały omówione algorymy służące do nauki sieci neuronowych oraz architektury takich sieci. Najważniejszą ich cechą wspólną jest dokonywanie w ten czy inny sposób podziału przestrzeni danych wejściowych w celu usprawnienia i uproszczenia ich działania. Powodem tworzenia takich sieci są wady najczęściej stosowanych w metodach sztucznej inteligencji wielowarstwowych sieci z jednostkami ukrytymi, opartych na funkcjach sigmoidalnych. Choć mogą one aproksymować dowolną zależność wejście—wyjście to metoda ich nauki, wsteczna propagacja błędu, jest powolna oraz bardzo kapryśna. Również inne własności tych sieci, przede wszystkim zdolność do generalizacji czyli znajdowania odpowiedzi na nieznane wcześniej dane są w niektórych przypadkach (np. klasyfikacji wzorców bitowych) niezadawalające. Drugą cechą wyróżniającą prezentowane tu metody jest wykorzystywanie własności geometrycznych przestrzeni danych, przede wszystkim sposobu mierzenia odległości.

Układ pracy jest następujący:

Rozdział pierwszy, drugi i trzeci zawiera omówienie funkcji radialnych, przedstawionych jako rozwiązanie zagadnienia aproksymacyjnego oraz ich zastosowanie w różnych rodzajach sieci.

Rozdział czwarty przedstawia sposób modyfikacji kształtu funkcji radialnych za pomocą zmian metryki przestrzeni, na której są one określone.

Rozdział piąty poświęcony jest algorytmom dokonującym podziału przestrzeni danych, służącym do klasyfikacji i aproksymacji danych.

Rozdział szósty opisuje zastosowanie metryki do klasyfikacji wzorców bitowych w przestrzeniach dyskretnych.

2 Pochodzenie funcji radialnych

Funkcje radialne są jednym z licznych rozwiązań zagadnienia aproksymacyjnego. Można za pomocą odpowiednio dużej ich liczby przybliżyć dowolną funkcję [Was], [Pg]. Zagadnienie to można sformuować w sposób następujący. Posiadamy zbiór $g = \{(\mathbf{x}_i, y_i) \in R^d \times R\}_{i=1}^N$ N punktów pochodzących z próbkowania nieznanej funkcji $f(\mathbf{x}_i)$ w wybieranych losowo punktach $\mathbf{x}_i$. Chcemy znaleźć (estymować) postać funkcji f i móc otrzymywać jej wartości poza punktami $\mathbf{x}_i$. Zakładamy, że dane te mogą być zaszumione. Tak postawiony problem ma nieskończenie wiele rozwiązań, czyli jest źle określony. Aby można go było rozwiązać jednoznacznie, na poszukiwaną funkcję f trzeba nałożyć dodatkowe warunki. Od tego, jakiego rodzaju to będą warunki, zależny będzie rodzaj metody aproksymacyjnej.

W przypadku funkcji radialnych owym dodatkowym warunkiem będzie żądanie jak największej ich gładkości. Oznacza to, że chcemy, aby poza zbiorem g (czyli tam, gdzie nie mamy o funkcji żadnych danych) funkcja f nie przejawiała zbędnych oscylacji, zmian przebiegu, itp.

Poszukiwanie takiej funkcji możemy przeprowadzić za pomocą rachunku wariacyjnego, budując odpowiedni funkcjonał, a następnie minimalizując go [PG]. Posiada on następującą postać

$$H[f] = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2 + \lambda \phi[f] \,. \tag{1}$$

Pierwszy człon opisuje, na ile uzyskane wyniki są zgodne ze znanymi danymi – posiadanymi wartościami funkcji. Podstawowym warunkiem znalezienia dobrego rozwiązania jest jego zgodność z tym, co już znamy. Drugi człon jest funkcjonałem mierzącym gładkość funkcji, zwanym stabilizatorem. Stała λ określa za-

leżność między obydwoma członami, czyli między gładkością funkcji a jej zgodnością ze znanymi danymi. Stała ta nazywana jest parametrem regularyzacji. Określenie funkcjonału $\phi[f]$ zależy do tego, co będziemy rozumieć przez *gładkość*. Podane niżej określenie gładkości pochodzi z pracy [PG]. Funkcja będzie tym gładsza, im mniej oscyluje. Funkcjonał $\phi[f]$ musi mieć mniejszą wartość dla tej z dwu funkcji, której rozwinięcie fourierowskie zawiera mniej składowych o wyższych częstościach. Ma on postać całki z transformaty Fouriera funkcji f – funkcji $\tilde{f}$ dzielonej przez transformatę $\tilde{G}$ pewnej funkcji G:

$$\phi[f] = \int\limits_{R^d} d\mathbf{s} \frac{\left|\tilde{f}(\mathbf{s})\right|^2}{\tilde{G}(\mathbf{s})} \,. \tag{2}$$

Funkcja $\tilde{G}$ jest dodatnio określona i ma następującą własność: $\tilde{G} \to 0$, gdy $\|\mathbf{s}\| \to +\infty$. Wszędzie, jeśli nie zaznaczono inaczej, norma $\|\cdot\|$ jest normą L_2. Oznacza to, że $\frac{1}{\tilde{G}}$ jest filtrem górnopasmowym i że funcjonał ϕ faktycznie mierzy moc fourierowskich składowych funkcji f. Zastosowanie filtra górnopasmowego powoduje, że funkcjonał $\phi[f]$ ma wartość większą dla tej funkcji, której moc przypadająca na wyższe częstości jest wyższa. W ten sposób mierzy on gładkość funkcji.

Funcjonał ϕ jest seminormą, tzn. posiada k wymiarową podprzestrzeń zerową. Znaczy to, że istnieją takie różne od zerowego elementy przestrzeni funkcyjnej, na której jest określony ten funkcjonał, dla których wartość jego jest zerowa. Przy założeniu, że G jest funkcją rzeczywistą (a więc $\tilde{G}$ jest funkcją symetryczną,$\tilde{G}(\mathbf{s}) = \tilde{G}(-\mathbf{s})$, por. [SK], str. 160), funkcja będąca rozwiązaniem zagadnienia wariacyjnego ma postać:

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x} - \mathbf{x}_i) + \sum_{\alpha=1}^{k} d_\alpha \psi_\alpha(\mathbf{x}) \,. \tag{3}$$

$\psi_{\alpha=1}^{k}$ jest bazą w k wymiarowej podprzestrzeni zerowej funkcjonału ϕ. Oznacza to, że do funkcji, która minimalizuje funkcjonał $\phi[f]$ można dodać dowolną kombinację funkcji $\psi_{\alpha=1}^{k}$, nie powodując zmiany wartości funkcjonału $\phi[f]$.

Funkcje $G(\mathbf{x}-\mathbf{x}_i)$, których kombinacją liniową przybliżamy funkcję f, nie są jeszcze funkcjami radialnymi. Aby nią była, funkcjonał ϕ musi spełniać dodatkowy warunek:

$$\phi[f(x)] = \phi[f(Rx)]\,,$$

gdzie R oznacza przekształcenie obrotu. Otrzymana wówczas funkcja G nie zależy od kierunku a tylko od odległości w przestrzeni R^n: $G(x) = G(\|x\|)$. Oznacza to, że żadna zmienna nie jest wyróżniona i ma taką samą wagę, jak pozostałe.

Od tego, jaką zastosowano funkcję G, zależy rodzaj funkcji radialnych. Najpopularniejsze funkcje gaussowskie maja zerowymiarową podprzestrzeń zerową (a więc funkcjonał ϕ jest normą, a nie seminormą).

Funkcje radialne określa się ogólnym mianem RBF (ang. *Radial Basis Functions*). Tym skrótem określa się również metody wykorzystujące te funkcje, np. sieć RBF.

Funkcje radialne można również otrzymać na gruncie teorii probabilistycznych - teorii estymatorów bayesowskich [PG] (patrz. Dodatek). Załóżmy, że ciąg y_i w zbiorze g powstał w wyniku nałożenia na dokładne wartości funkcji jakiejś funkcji szumu o niezależnym rozkładzie normalnym.

Możemy w tym wypadku zastować wzór Bayesa w następującej formie [PG], [Z]:

$$P[f|g] \propto P[g|f]P[f]\,. \tag{4}$$

Występujące tu rozkłady prawdopodobieństwa mają takie znaczenia:

1^0. $P[f|g]$ to warunkowy rozkład prawdopobieństwa tego, że funkcja f jest przybliżeniem wynikającym ze zbioru g. Im funkcja f jest lepszym przybliżeniem, tym prawdopodobieństwo wynikające z tego rozkładu jest wyższe.

2^0. $P[g|f]$ to warunkowy rozkład prawdopodobieństwa tego, że mając funkcję f, otrzymamy zbiór wyników g. Ponieważ, jak wyżej wzmiankowano, wyniki otrzymane z funkcji f są zaszumione, to rozkład ten jest modelem szumu. Ma on postać:

$$P[g|f] \propto exp\Big(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2\Big)\,. \tag{5}$$

3^0. $P[f]$ bezwarunkowy rozkład *a priori* funkcji f. Opisuje naszą początkową wiedzę o funkcji f. W przypadku dyskretnego próbkowania funkcji f rozkład ma postać

$$P[f] \propto exp(-\alpha\phi[f])\,. \tag{6}$$

ϕ jest naszym funkcjonałem gładkości, α liczbą dodatnią. Reprezentuje on wiedzę (czyli w tym wypadku naszą chęć uzyskania jak najgładszej funkcji) *a priori* o funkcji f. Prawdopodobieństwo rośnie wraz z maleniem wartości funkcjonału $\phi[f]$ [PG].

Możemy teraz zapisać wzór Bayesa (4) w postaci:

$$P[f|g] \propto exp\Big(-\frac{1}{2\sigma^2}\Big[\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2 + 2\alpha\sigma^2\phi[f]\Big]\Big)\,. \tag{7}$$

Maksymalizacja tego prawdopobieństwa prowadzi do znanego już równania wariacyjnego:

$$H[f] = \sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2 + \lambda\phi[f]\,. \tag{8}$$

Stała $\lambda = \alpha\sigma^2$ równa jest iloczynowi kwadratu wariancji rozkładu szumu σ i parametru α charakteryzującego naszą wiedzę aprioryczną o funkcji f. Opisuje ona zależność między tymi czynnikami.

3 Rodzaje funkcji radialnych

Funkcje radialne określone na R^n można z grubsza (i nieformalnie) podzielić na takie, które w sposób jednorodny zależą od zmiennej wielowymiarowej (czyli odległość jest obliczana w R^n) i takie, które są zbudowane z funkcji mniejwymiarowych (konkretnie określonych na R^1). W pierwszej grupie możemy wyróżnić:

Wielowymiarowe funkcje sklejane. Funkcjonał gładkości ma postać

$$\phi[f] = \int\limits_{R^d} d\mathbf{s} \|\mathbf{s}\|^s |\tilde{f}(\mathbf{s})|^2 , \tag{9}$$

czyli

$$\tilde{G}(\mathbf{s}) = \frac{1}{\|\mathbf{s}\|^{2m}} . \tag{10}$$

Stąd otrzymamy po dokonaniu transformaty Fouriera następujące funkcje:

$$G(\mathbf{x}) = \begin{cases} \|\mathbf{x}\|^{2m-d} \ln \|\mathbf{x}\| & \text{, gdy } 2m > d \text{ oraz } d \text{ jest parzyste;} \\ \|\mathbf{x}\|^{2m-d} & \text{w przeciwnym razie.} \end{cases} \tag{11}$$

Podprzestrzeń zerowa seminormy ϕ zbudowana jest z wielomianów stopnia $k = \binom{d+m-1}{d}$.

Funkcje gaussowskie. Otrzymujemy je z następującego funkcjonału

$$\phi[f] = \int\limits_{R^d} d\mathbf{s} \; exp\Big(\frac{\|\mathbf{s}\|^2}{\beta}\Big) |\tilde{f}(\mathbf{s})|^2 , \tag{12}$$

i mają one postać

$$G(\mathbf{x}) = exp\Big(- \frac{\|\mathbf{x}\|^2}{\beta} \Big) . \tag{13}$$

Parametr β jest liczbą dodatnią. Funkcje gaussowskie są dodatnio określone, a funkcjonał $\phi[f]$ jest normą , stąd podprzestrzeń zerowa zawiera tylko element zerowy. Nie ma więc dodatkowego członu w równaniu (3). Wadą tych funkcji w porównaniu z funkcjami sklejanymi jest obecność parametru β.

Zestawienie. Inne funkcje radialne zawarte są w zestawieniu [PG]:

$$G(r) = e^{-\beta r^2} \qquad k = 0$$

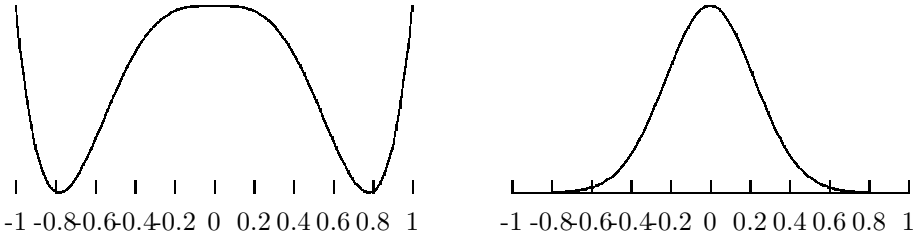
$$G(r) = \sqrt{r^2 + c^2} \qquad k = 1$$

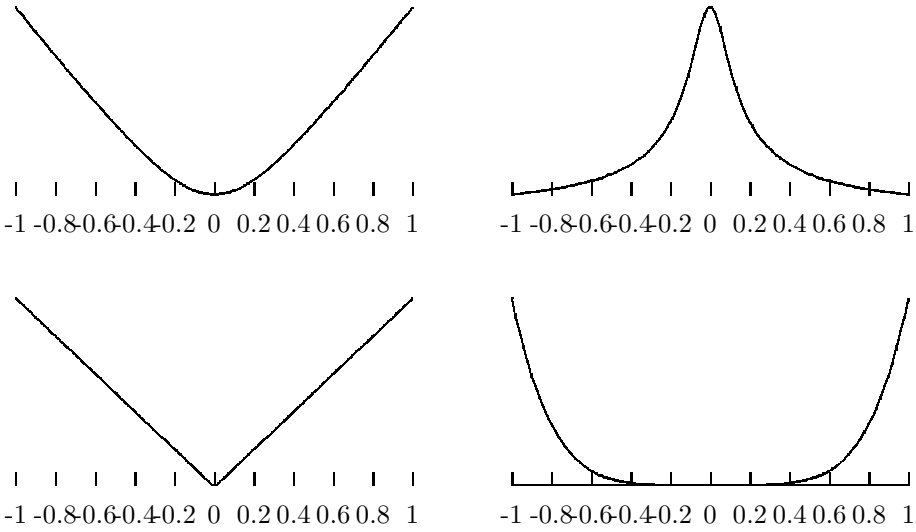
$$G(r) = \frac{1}{\sqrt{r^2 + c^2}} \qquad k = 0$$

$$G(r) = r^{2n+1} \qquad k = n$$

$$G(r) = r^{2n}\, ln\, r \qquad k = n$$

Liczba k oznacza wymiar podprzestrzeni zerowej seminormy (wartość zero oznacza pełną normę).





Rys.(1) Różne rodzaje jednowymiarowych funkcji radialnych. Od góry, wierszami: $f(r) = r^2 log\,(r)$, $f(r) = exp(-\frac{r^2}{0.1})$, $f(r) = \sqrt{r^2 + 0.1}$, $f(r) = \frac{1}{\sqrt{r^2 + 0.01}}$, $f(r) = r$, $f(r) = r^3$, gdzie $r = |x|$.

Do drugiej grupy można zaliczyć funkcje zwane iloczynem tensorowym (ang. *tensor product*) i addytywne funkcje sklejane (ang. *additive splines*). Pierwsze z nich mają postać

$$G(\mathbf{x}) = \prod_{j=1}^{d} g(x_j)\,, \tag{14}$$

gdzie x_j oznacza i-tą składową wektora x. Otrzymujemy je z funkcjonału zbudowanego w następujący sposób

$$\phi[f] = \int\limits_{R^d} d\mathbf{s} \frac{|\tilde{f}(\mathbf{s})|^2}{\prod\limits_{j=1}^{d} \tilde{g}(s_j)}\,. \tag{15}$$

Zatem funkcja $\tilde{G}(\mathbf{s})$ wygląda następująco:

$$\tilde{G}(\mathbf{s}) = \prod_{j=1}^{d} \tilde{g}(s_j)\,. \tag{16}$$

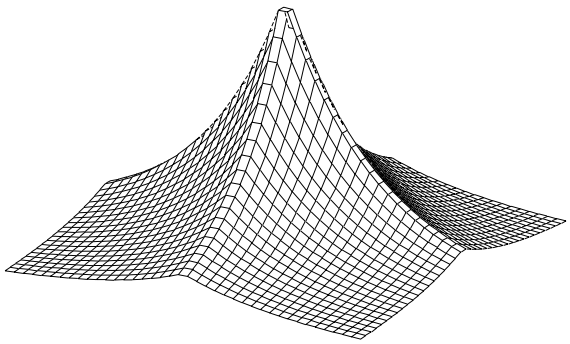
$\sim$ oznacza transformatę fourierowską danej funkcji. Przez dobór odpowiednich jednowymiarowych funkcji $\tilde{g}(s)$ można osiągnąć ciekawe funkcje wielowymiarowe. Jeśli wybierzemy $\tilde{g}(s) = e^{-s^2}$ (jednowymiarowa funkcja gaussowska), to funkcja G będzie wielowymiarową funkcją gaussowską:

$$G(\mathbf{x}) = e^{-\|\mathbf{x}\|^2} = e^{-\sum\limits_i x_i^2}\,. \tag{17}$$

Z kolei $\tilde{g}(s) = \dfrac{1}{1+s^2}$ da taką funkcję:

$$G(\mathbf{x}) = e^{-\|\mathbf{x}\|_{L_1}} = e^{-\sum\limits_i |x_i|}\,. \tag{18}$$

Norma L_1 wymaga o wiele mniejszego wysiłku obliczeniowego niż norma L_2.

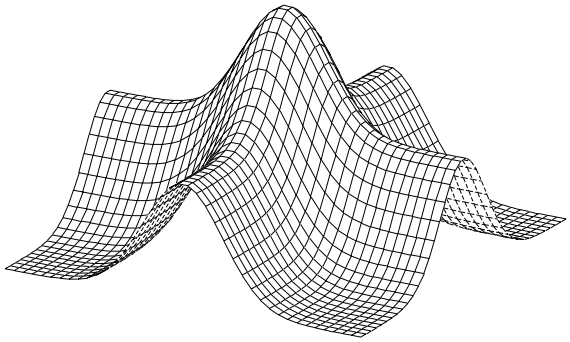


Rys.(2) Wykres funkcji $f(x,y) = exp(-\dfrac{|x|}{5}) * exp(-\dfrac{|y|}{5}) = exp(-\dfrac{|x|+|y|}{5})$.

Addytywne funkcje sklejane są określone poprzez sumę jednowymiarowych f_μ nazywanych składowymi funkcji f:

$$f(\mathbf{x}) = \sum_{\mu=1}^{d} f_\mu(x^\mu)\,. \tag{19}$$

Funkcje f_μ zależą od μ-składowej wektora $\mathbf{x}$. Tak określona funkcja jest bardzo wygodna ze względów obliczeniowych oraz możliwości śledzenia jej zależności od poszczególnych składowych wektora $\mathbf{x}$.



Rys.(3) Wykres funkcji $f(x) = exp(-\frac{x^2}{10}) + exp(-\frac{y^2}{10})$.

4 Zastosowanie funkcji radialnych

Funkcje radialne znajdują zastosowanie zarówno w algorytmach sieciowych, jak i w typowych algorytmach aproksymacyjnych, nie odwołujących się do architektury sieciowej. Sieci wykorzystywane są do przybliżania funkcji oraz do klasyfikacji.

Sieci typu RN. Najprostszą siecią wykorzystującą funkcje radialne są sieci RN (ang. *Regularization Network*). Są to sieci jednowarstwowe, o wektorowym wejściu ($\mathbf{x}$) i pojedyńczym wyjściu. Środkową, ukrytą warstwę stanowią funkcje radialne umieszczone w punktach $\mathbf{x}_i$, należących do zbioru zależności wejście – wyjście podanego sieci w celu jej nauki. Sieć ta realizuje bezpośrednio wzór (3). Liczba jej wejść równa jest wymiarowi przestrzeni danych – rys.(13).

W tym prostym wariancie nauce podlegają tylko wagi c_i. Najczęściej znajdują zastosowanie w takich sieciach wielowymiarowe funkcje gaussowskie [Was]. Wariancje tych funkcji są stałe, nie ulegają zmianie podczas nauki (w wielu wymiarach funkcję gaussowską charakteryzuje macierz kowariancji Σ_{ij}, będąca w tej metodzie wielokrotnością macierzy jednostkowej, a więc nie wyróżniająca żadnego kierunki w przestrzeni wejściowej). Sieć taka nie posiada wag połączeń wejść z neuronami warstwy ukrytej, wybór właściwego neuronu (czyli funkcji radialnej) wynika z obliczenia odległości wektora wejściowego od centrum funkcji (w terminologi rozkładów – średniej) – wyrażonej poprzez wartość funkcji, szybko malejącej wraz ze wzrostem odległości od centrum funkcji.

Wielkość wariancji określa rozmiar obszaru, z którego dane są przez daną funkcję „rozpoznawane". W ten sposób wprowadza się „ziarnistość" przestrzeni danych wejściowych. Funkcje służą tylko do przechowywania nauczonych danych i stwierdzenia, na ile nieznane dane leżą od nich daleko. Wadą takiej sieci jest umiejscawianie funkcji radialnych we wszystkich punktach przestrzeni wejściowej podanych jako zbiór do nauki. Spowalnia to trochę działanie sieci (w porównaniu z sieciami wielowarstwowymi, opartymi na sigmoidach), gdyż aby znaleźć właściwą funkcję, trzeba obliczyć wartości wszystkich funkcji w sieci. Ma to szczególne znaczenie w przypadku dużych zbiorów danych.

Sieci takie można przerobić tak, by posiadały wyjście wektorowe [PG] [Was] rys.(14). Dalszą modyfikacją tych sieci jest umożliwienie obejmowania przez pojedyńczą funkcję większej liczby przykładów poprzez dopuszczenie do zmian parametrów (σ, Σ). Jeśli dane rozłożone są równomiernie w całej dostępnej sieci przestrzeni, to do ich opisania wystarczą funkcje umieszczone w jednakowych odstępach o tych samych wariancjach. Natomiast, gdy dane wykazują tendencje do grupowania się w pewnych obszarach przestrzeni (ang. *clustering*), to dobrym pomysłem wydaje się być zlokalizowanie takich obszarów i opisanie ich mniejszą liczbą funkcji radialnych, umieszczonych w centrach takich obszarów, niekoniecznie dokładnie w punktach, z których pochodzą dane treningowe. Odchylenia standartowe możemy wyznaczyć poprzez obliczenie odległości od N najbliższych sąsiadów i nadanie tej wartości parametrowi σ. Dobre rezultaty oraz przyspieszenie nauki daje ustalenie $N = 1$, czyli ograniczenie się do najbliższego sąsiada [Was].

Sieci typu GRN. Rozwinięciem sieci RN są sieci typu GRN (*Generalized Regularization Network*) [PG]. Kosztem wzrostu liczby parametrów zmiejszono ilość potrzebnych funkcji radialnych. Uzyskano to w wyniku wprowadzenia zamiast $\mathbf{x}_i$ nowych zmiennych $\mathbf{z}_i = \mathbf{W}\mathbf{x}_i$, gdzie $\mathbf{W}$ jest macierzą przekształcenia liniowego (niekoniecznie kwadratową). Funkcja $G(\mathbf{z}_i)$ jest funkcją radialną w zmiennych $\mathbf{z}_i$. Warunek maksymalnej gładkości stosuje się teraz do funkcji $F(\mathbf{z}) = F(\mathbf{W}\mathbf{x})$ takiej, że $f(\mathbf{x}) = F(\mathbf{W}\mathbf{x})$. Wyrażenie na $f(\mathbf{x})$ ma teraz postać

$$f(\mathbf{x}) = F(\mathbf{W}\mathbf{x}) = \sum_{i=1}^{n} c_i G(\mathbf{W}\mathbf{x} - \mathbf{W}\mathbf{t}_i)\,, \tag{20}$$

w którym $n \leq N$, zaś t_i są centrami funkcji radialnych. Mamy więc mniej funkcji niż danych przykładów i więcej parametrów do nauki – macierz $\mathbf{W}$ i oczywiście

współczynniki rozwinięcia c_i. Do wyznaczenia macierzy $\mathbf{W}$ i współczynników c_i można zastosować np. metodę najmniejszych kwadratów. Położenia t_i są dobierane albo heurystycznie, albo również traktujemy je jako parametry do wyznaczenia.

Dokonanie przekształcenia zmiennych oznacza, że uważamy, iż pewne zmienne mają znaczenie, a inne nie lub że znaczące zmienne są kombinacjami liniowymi zmiennych, które posiadamy.

Sieci te można uczyć na podstawie ustalonego zbioru przykładów (ang. *batch learning*), jak również przedstawiając im dane w sposób ciągły (ang. *on–line*). Ten drugi sposób wymaga dokonywania grupowania danych (ang. *clustering*), aby liczba funkcji radialnych nie rosła nieograniczenie. Można to wykonać w sposób następujący [Was]. Ustalamy stały promień r. Lokujemy funkcję radialną na pierwszym otrzymanym wektorze. Jeśli następne wektory leżą dalej niż r od pierwszego, to ustawiamy tam nową funkcję. Jeśli nie, to nic nie dodajemy – posiadana funkcja dobrze opisuje znane dotąd dane. Powtarzamy tę procedurę dla wszystkich napływających danych i wszystkich istniejących funkcji.

Ciekawą modyfikację sieci wykorzystujących funkcje radialne można znaleźć w pracy [LN], nazwaną tam siecią VI (ang. *Validity Index*). Funkcji radialnych jest mniej niż wektorów wejściowych, tak jak w sieci GRN. Dodanie dodatkowych węzłów przetwarzających sygnały pochodzące z funkcji radialnych pozwoliło obliczać granice błędu dla wszystkich m wyjść tej sieci oraz sygnalizować przypadek ekstrapolacji danych. Jest to możliwe dzięki obliczaniu przez sieć lokalnych gęstości danych w otoczeniach poszczególnych funkcji radialnych. W celu uproszczenia obliczeń autorzy [LN] zastosowali zamiast funkcji radialnych prostokątną, wielowymiarową funkcję przynależności o promieniu a i środku umiesz-

czonym w centrum rozważanej grupy danych. Łatwo więc policzyć, ile dana funkcja obejmuje punktów. Porównując tę wielkość z liczbą wszytkich danych, uzyskujemy informacje o lokalnej gęstości danych. Takie podejście pozwala wykrywać obszary mało reprezentatywne dla przybliżanej funkcji. Jest to ważne z tego względu, że wiarygodność wyników dawanych przez sieć typu RBF uwarunkowana jest posiadaniem niepustego zbioru treningowego niezależnych zmiennych oraz dokładnością wykonywanego przybliżenia.

Do przechowywania danych, tak jak ma to miejsce w powyższych przykładach, mogą być użyte tylko te funkcje radialne, które tak jak funkcje gaussowskie szybko maleją wraz ze wzrostem odległości, są więc dobrze zlokalizowane.

Pojęcie lokalności można sformalizować w sposób następujący [BV]. Zdefiniujmy funkcję błędu (tj. funkcję danych wejściowych sieci a nie tylko jej parametrów ω) sieci w ten sposób, że zawierać ona będzie zależne od x wagi błędów popełnianych podczas obliczania wartości wyjściowej dla danego x, zamiast globalnej średniej tych błędów. Oznaczmy przez $\hat{y}$ pożądaną odpowiedź sieci na wartość wejściową x. Błąd popełniony przez sieć dającą wynik $f_\omega(x)$ niech wynosi $J[\hat{y}, f_\omega(x)]$. Funkcję błędu w punkcie x_0 możemy zapisać w formie ważonej średniej po zbiorze treningowym zawierającym l elementów:

$$Err(x_0;\omega) = \frac{1}{l}\sum_{i=1}^{l} K(x_i - x_0, b) J[\hat{y}_i, f_\omega(x_i)]\,, \tag{21}$$

gdzie funkcja $K(x_i - x_0, b)$ pełni rolę zależnej od x wagi. Parametr b opisuje rozmiar tej funkcji, czyli określa stopień lokalności, w zakresie od 0 (ograniczenia się tylko do jednego punktu) do $+\infty$ (obejmowania całej przestrzeni). Podczas nauki sieci zostaje wyróżniony pewien podzbiór W wszystkich wag sieci [BV]. Nauka polega na minimalizacji funkcji błędu (21) względem parametrów sieci, w

wyniku czego otrzymamy optymalny wektor tych parametrów:

$$\begin{aligned} \omega^*(x_0, b) &= arg \min_{\omega \in W} Err(x_0; \omega) \\ &= arg \min_{\omega \in W} \frac{1}{l} \sum_{i=1}^{l} K(x_i - x_0, b)\, J[\hat{y}_i, f_\omega(x_i)]\,. \end{aligned} \tag{22}$$

Znalezione w ten sposób parametry ω^* zależą zarówno od parametru b , jak i od położenia w przestrzeni wejściowej x. Są zatem określone lokalnie. Funkcje $K(x_0 - x_i, b)$ mogą być różne, w szczególności są to funkcje prostokątne:

$$K(x_0 - x, b) = \begin{cases} 1, & \text{gdy } \|x_0 - x\| \leq \dfrac{b}{2}; \\ 0, & \text{gdy } \|x_0 - x\| > \dfrac{b}{2} \end{cases} \tag{23}$$

lub funkcje gaussowskie o $\sigma = \dfrac{b}{2}$.

Zajmijmy się metodami klasyfikacji wektorów wejściowych. Niech wektor wyjściowy y ma n składowych.Wartość $y_i = 1$ składowej i tego wektora oznacza przynależność wektora x do klasy i. W zależności od rodzaju funkcji $K(x_0 - x_i, b)$, wielkości b oraz miary J , możemy otrzymać różne rodzaje metod. Weźmy jako miarę błędu funkcję kwadratową $J[y, \hat{y}] = (y - \hat{y})^2$ oraz załóżmy, że poszukujemy w otoczeniu punktu x_0 stałego przybliżenia $\hat{y}^*$:

$$\hat{y}^* = arg \min_{\hat{y}} \frac{1}{l} \sum_{i=1}^{l} K(x_i - x_0, b)\, (y_i - \hat{y})^2\,, \tag{24}$$

gdzie x_0 to wzorzec testowy, x_i i y_i to przykłady treningowe.

Jeśli zastosujemy prostokątną funkcję K, to uzyskamy metodę kNN. Poszukiwane przybliżenie $\hat{y}^*$ to średnia wyników, pochodzących z otoczenia punktu x_0 o takiej średnicy b, że zawiera dokładnie k znanych wzorców.

W metodzie funkcji radialnych mamy R ustalonych funkcji gaussowskich, lokalizujących dane treningowe, scharakteryzowanych położeniami x_r i wariancja-

mi σ_r, $r = 1, \ldots, R$. Możemy zminimalizować funkcjonał (24) biorąc za K poszczególne funkcje gaussowskie, co prowadzi do otrzymania ważonych średnich $\hat{y}_r^*$ dla każdej funkcji. Wynik dawany przez całą sieć jest średnią wartości wszystkich funkcji gaussowskich mnożonych przez tak otrzymane wartości $\hat{y}_r^*$:

$$\hat{y}(x) = \sum_{r=1}^{R} \hat{y}_r^* K(x - x_r, \sigma_r) \,. \tag{25}$$

Dochodzimy więc do znanego wzoru opisującego sieć RBF (3) , mając za wagi wartości $\hat{y}_r^*$. To, że wektory wyjściowe są używane jako wagi, nie powinno dziwić, jeśli weźmiemy pod uwagę odmianę sieci RBF zwaną GRNN (ang. *Generalized Regression Neural Network*) [Was], nauka której polega na przypisaniu wagom składowych wektorów wyjściowych ze zbioru treningowego.

Wprowadzenie tak zdefiniowanego pojęcia lokalności pozwala na uzyskiwanie lepszych wyników uczenia sieci, poprzez dopuszczenie do zmian parametrów opisujących lokalność. Umożliwia to kontrolę nad zależnościami między zdolnością sieci do generalizacji a jej pojemnością i dokładnością otrzymywanych wyników. Przykładem sieci, której parametr b ma nieskończoną wartość, jest wielowarstwowa sieć z jednostkami ukrytymi, opisywanymi sigmoidami. W takiej sieci jej parametry — wagi i progi są zmieniane zależnie od wszystkich treningowych danych wejściowych.

Aproksymacja. Zastosowanie funkcji radialnych bezpośrednio do aproksymacji funkcji omówione zostanie na podstawie pracy [All] opisującej metody przybliżania i wizualizacji funkcji określonych w 1, 2 lub 3 wymiarach. Funkcjami tymi były rozkłady różnych wielkości mierzonych podczas doświadczeń fizyki wysokich energii. Głównym problemem występującym w tych zagadnieniach jest mała gęstość danych w więcejwymiarowych przestrzeniach. Są one

w dużym stopniu puste. Wynikają stąd trudności w zastosowaniu tradycyjnych metod aproksymacji, szczególnie tych, które opierają się na regularnych siatkach punktów. Inną sprawą jest duża liczba parametrów potrzebnych do wyznaczenia wartości funkcji w tych metodach oraz niekontrolowane nieraz zachowanie przybliżającej funkcji pomiędzy węzłami (np. wielomiany wyższych stopni). Gdy brak danych, najlepszym rozwiązaniem może być po prostu zależność liniowa. Tych wad nie mają odpowiednio dobrane funkcje radialne. W [All] zastosowano funkcje postaci

$$\phi_j(r) = \sqrt{r^2 + \Delta_j{}^2}\,. \tag{26}$$

Funkcja przybliżająca ma postać kombinacji liniowej

$$s(x) = \sum_{j=1}^{n} \alpha_j \phi_j(\|x - x_j\|) \tag{27}$$

o współczynnikach rozwinięcia α_j, centrach x_j i parametrach skalujących Δ_j. Ich wartości trzeba określić.

Aby znaleźć położenia funkcji x_j, należy zauważyć, że funkcje te mają najmniejszy promień krzywizny równy Δ_j dla $r = 0$. Ponieważ promień krzywizny jest odwrotnie proporcjonalny do drugiej pochodnej funkcji, to wystarczy znaleźć wśród danych wejściowych takie, dla których dyskretna druga pochodna ma największą wartość i tam umieścić funkcje radialne. Ilość funkcji w rozwinięciu regulujemy dobraniem odpowiedniego progu nałożonego na wartość drugiej pochodnej, po przekroczeniu którego umieszczamy w danym miejscu funkcję bazową. Wartości pozostałych parametrów , po określeniu położeń funkcji, są dobierane za pomocą zaawansowanych mutacji metody najmniejszych kwadratów. Więcej szczegółów można znaleźć w [All]. Opisana metoda realizowana jest przez pakiet programów HBOOK [All]. Wykorzystuje się ją przede wszystkim do

aproksymacji rozkładów różnych wielkości np. różniczkowych przekrojów czynnych. Dziedziną może być dowolna przestrzeń (do celów wizualizacji danych, oczywiście co najwyżej trójwymiarowa). Zasadniczym celem jest zastąpienie tradycyjnych histogramów wykresami gładkich funkcji. Funkcje radialne pozwalają to wykonać stosunkowo niskim kosztem obliczeniowym.

Sigmoidy. Warto tu wspomnieć o funkcji sigmoidalnej, będącą podstawą do tworzenia wielowarstwowych sieci z warstwami ukrytymi, choć nie mieści się ona w formalizmie funkcji radialnych [PG]. Wartości podawane na wejściu takiej sieci są przetwarzane przez funkcje sigmoidalne pierwszej warstwy, z których uzyskane wartości są kierowane do następnej warstwy, itd. Parametry ukryte takiej sieci, to współczynniki rozwinięcia kombinacji liniowych tych funkcji (wagi) oraz parametry skalujące (progi).

To, co naprawdę realizuje taka sieć, to aproksymacja zależności pomiędzy wejściem a wyjściem $F : X \longrightarrow Y$ za pomocą funkcji [D1]:

$$F_w(X) = \sigma(\sum_{i_1} W_{i_1}^{(1)} \sigma(\sum_{i_2} W_{i_2}^{(2)} \sigma(\ldots(\sum_{i_k} W_{i_k}^{(k)})\ldots))) \,, \tag{28}$$

gdzie $\sigma(x) = (1 + e^{-\alpha\, x})^{-1}$ jest funkcją sigmoidalną.

Funkcja błędu takiej sieci jest następująca:

$$E(W) = \frac{1}{2} \sum_p \sum_i \left(Y_i^p - F_w\left(X_i^{(p)} \right) \right)^2 , \tag{29}$$

gdzie (Y_i^p, X_i^p) to próbki przybliżanej zależności F (przykłady), sumowanie zaś odbywa się po wszystkich przykładach p i ich współrzędnych i. W trakcie nauki sieci minimalizuje się $E(W)$ ze względu na współczynniki W. Pojawia się wówczas nietrywialny problem omijania lokalnych minimów.

Często stosowane funkcje radialne to funkcje gaussowskie. Wykorzystują je metody nie odwołujące się jawnie do formalizmu RBF. Stosowane są one w metodach klasyfikacyjnych oraz w systemach typu FSM, wszędzie tam, gdzie trzeba określić położenie czegoś w jakiejś przestrzeni oraz określić tego czegoś rozmiary (poprzez dobór odpowiedniej wartości dyspersji funkcji gaussowskiej).

Model FSM. System FSM (ang. *Feature Space Model*) [D1] ma za zadanie bezpośrednią budowę funkcji realizowanej przez sieć, bez pomocy kosztownych metod minimalizacji funkcji błędu. Gromadzi on tzw. fakty, tj. zbiory wartości wejściowych i wyjściowych jako punkty w N wymiarowej przestrzeni ($N = n + m$, gdy budujemy funkcję $f : R^n \longrightarrow R^m$). W każdym takim fakcie zlokalizowana jest funkcja G, np. taki iloczyn jednowymiarowych funkcji gaussowskich

$$\begin{aligned} G(\mathbf{X}, \mathbf{Y}, \sigma) &= exp\Big(- \sum_{i=1}^{N} \tfrac{(X_i - D_i)^2}{\sigma_i} \Big) = \prod_{i=1}^{N} exp\Big(- \tfrac{(X_i - D_i)^2}{\sigma_i} \Big) \\ &= \prod_{i=1}^{N} g(X_i, D_i, \sigma_i) \,, \end{aligned} \tag{30}$$

gdzie

$\mathbf{X} = (X_1, X_2, \ldots, X_n)$, to wektor wejściowy

$\mathbf{D} = (D_1, D_2, \ldots, D_n)$, to umiejscowienie funkcji G (faktu)

$\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, to wektor dyspersji.

Można również zastosować inne niż gaussowskie funkcje, np. niesymetryczne gaussiany lub iloczyny funkcji sigmoidalnych, zwiększając jednocześnie liczbę parametrów. Fakty zgromadzone w przestrzeni FSM opisuje funkcja FSM, będąca ważoną sumą funkcji G po wszystkich faktach:

$$FSM(\mathbf{X}, \mathbf{D}, \sigma) = \sum_p W_p G(\mathbf{X}, \mathbf{D}^p, \sigma^p) = \sum_p W_p \prod_i e^{\frac{-(X_i - D_i^p)^2}{\sigma_i^p}} \,. \tag{31}$$

Funkcja taka nie zanika tylko w pobliżu punktów $\mathbf{D}^p$. Parametrami adaptacyjnymi są wagi W_p i ewentualnie dyspersje σ_i^p. Badając wartość funkcji FSM, możemy

określać prawdziwość (tj. istnienie) danej relacji wejście—wyjście lub na podstawie niepełnej informacji (niekompletnego wektora $\mathbf{D}$) znajdować jej brakującą część (i to za pomocą jednowymiarowych przeszukiwań).

5 Modyfikacja kształtu funkcji radialnych za pomocą metryki

Wiele funkcji stosowanych w metodach sztucznej inteligencji za argument posiada odległość (bądź jej kwadrat) pomiedzy punktami w przestrzeni, na której są określone. Stąd ważny jest sposób liczenia tej odległości. Zmieniając go, zmieniamy kształty i własności funkcji. Najprostszym sposobem zmiany odległości jest wprowadzenie tensora metrycznego różniącego się od zwykle stosowanego euklidesowego, dającego odległość typu L_2. Pozwala to na modyfikacje kształtu poziomic funkcji radialnych od okręgów (metryka euklidesowa), poprzez elipsy aż do prostych równoległych, czyli usunięcia zależności od pewnych zmiennych lub ich kombinacji liniowych.

Tensorem metrycznym lub metryką będziemy nazywać symetryczny tensor drugiego rzędu o wymiarze równym wymiarowi danej przestrzeni. Będzie on (a właściwie jego macierzowe przedstawienie) oznaczany przez **g**.

Kwadrat odległości dwu punktów o współrzędnych $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ i $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ w n–wymiarowej przestrzeni dany jest wzorem:

$$d^2(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} g_{ij}(x_i - y_i)(x_j - y_j)\,. \tag{32}$$

Ponieważ zajmować się tu będziemy tylko metryką określoną globalnie, moż-

na zastosować w powyższym wzorze różnice skończone zamiast infinitezymalnych.

Przypadek dwuwymiarowy

Przypadek dwuwymiarowy jest pouczający z tego względu, że wyniki można łatwo zobrazować oraz rozszerzyć na przypadek wielowymiarowy. Kwadrat odległości na płaszczyźnie można zapisać w następujący sposób:

$$d^2(p_1, p_2) = g_{11}(x_1 - x_2)^2 + 2g_{12}(x_1 - x_2)(y_1 - y_2) + g_{22}(y_1 - y_2)^2 \,, \quad (33)$$

gdzie

$p_1 = (x_1, y_1)$

$p_2 = (x_2, y_2)$.

Zbadajmy, jaki kształt mają krzywe zbudowane z punktów (x, y), leżących w stałej odległości r od środka układu współrzędnych, czyli okręgi w danej metryce.

Opisane są one równaniem:

$$d(p_1, p_2) = g_{11}x^2 + 2g_{12}xy + g_{22}y^2 - r^2 = 0 \,. \quad (34)$$

Jest to równanie drugiego stopnia opisujące krzywe stożkowe (elipsę, hiperbolę, parabolę) lub ich zniekształcenia (punkt, proste równoległe lub przecinające się). O tym, jaka jest to krzywa, decydują dwa tzw. wyznaczniki:

$$\delta = g_{11}g_{22} - g_{12} \quad (35)$$

i

$$\Delta = -g_{11}g_{22}r^2 + g_{12}r^2 = -r^2\,\delta \,. \quad (36)$$

Przedstawiam tu tabelkę pochodzącą z [Mat], opisującą rodzaj krzywej zależności od tych wyznaczników.

Nie wszystkie występujące w tabelce krzywe można otrzymać z równania (34). Wynika to z niewystępowania w równaniu wyrazów liniowych w x i y. Nie dostaniemy krzywych leżących na przekątnej tabeli. Warto zwrócić uwagę na fakt, że przy założeniu $r^2 \neq 0$ (a tak będziemy cały czas zakładać) typ krzywej nie zależy od r^2, a co najwyżej od znaku r^2 - dla dalszego rozróżnienia typów prostych.

Δ	Rodzaj obrazu geometrycznego równania	$\delta < 0$	$\delta > 0$	$\delta = 0$		
				$-g_{11}\, r^2 < 0$	$-g_{11}\, r^2 = 0$	$-g_{11}\, r^2 > 0$
$\Delta = 0$	„Krzywe zniekształcone" (proste, punkt lub obraz urojony)	Dwie proste przecinające się rzeczywiste	Dwie proste urojone, przecinające się w rzeczywistym punkcie	Dwie proste równoległe	Dwie proste przystające (pokrywające się)	Obraz urojony (dwie proste urojone równoległe)
$\Delta \neq 0$	Krzywe właściwe (krzywe 2 stopnia — stożkowe)	Hiperbola	Elipsa rzeczywista (ew. okrąg.), gdy $\Delta\, g_{11} < 0$ $(g_{11} > 0)$. Elipsa urojona, gdy $\Delta\, g_{11} > 0$ $(g_{11} > 0)$	Parabola		

Tab. (1) Rodzaj krzywej w zależności od wyznaczników δ i Δ.

$\Delta = 0$**.** W tym wypadku prawdziwa jest także równość $\delta = 0$, w związku z czym otrzymamy jedną lub dwie proste równoległe w zależności od wartości i znaku wielkości $-g_{11}r^2$.

$-g_{11}r^2 = 0$**.** Ponieważ z założenia mamy $r^2 > 0$, to musi zachodzić $g_{11} = 0$. W tym przypadku równanie ma postać:

$$2g_{12}\,x\,y + g_{22}\,x^2 = r^2 \tag{37}$$

i opisuje dwie proste pokrywające się. Ponieważ $\delta = 0$, to $g_{12} = \pm\sqrt{g_{22}g_{11}}$ i równanie prostych ma postać

$$y = \pm\frac{r}{\sqrt{g_{22}}} \tag{38}$$

$-g_{11}r^2 < 0$**.** Obrazem równania są dwie proste równoległe. Z powyższej nierówności wynika, że g_{11} i r^2 mają te same znaki. Jeśli $r^2 > 0$, to z warunku $\delta = g_{11}g_{22} - {g_{12}}^2 = 0$ wynika, że $g_{12} = \pm\sqrt{g_{11}g_{22}}$. Wstawiając tę wielkość do równania (34), otrzymamy równania prostych:

$$g_{11}x^2 \pm 2\sqrt{g_{11}g_{22}}\,xy + g_{22}\,x^2 = r^2 \tag{39}$$

lub

$$(\sqrt{g_{11}}x \pm \sqrt{g_{22}}y)^2 = r^2\,. \tag{40}$$

Możemy stąd otrzymać równania dwu zestawów prostych równoległych

$$\begin{aligned}\sqrt{g_{11}}x + \sqrt{g_{22}}y &= \pm r^2\\ \sqrt{g_{11}}x - \sqrt{g_{22}}y &= \pm r^2\end{aligned} \tag{41}$$

Gdy $r^2 < 0$, to także $g_{11} < 0$ i musi zachodzić $g_{22} < 0$, aby kwadrat g_{12} był dodatni. Otrzymamy także dwa zestawy dwu prostych równoległych, należy tylko w powyższych wzorach zamienić $\pm$ na $\mp$.

$-g_{11}r^2 > 0$**.** Równanie(34) opisuje proste równoległe urojone tj. o urojonych współczynnikach.

$\Delta \neq 0$**.** Warunek $\Delta \neq 0$ implikuje warunek $\delta \neq 0$, w związku z czym na pewno nie otrzymamy z równania (34) paraboli a tylko hiperbolę i elipsę (lub okrąg). Oto niektóre ich własności.

$\delta > 0$**.** Ponieważ $\delta = g_{11}g_{22} - {g_{12}}^2 > 0$, to składowe metryki g_{11} i g_{22} nie mogą być równe zeru i muszą mieć te same znaki Środek elipsy ma współrzędne $(0, 0)$ ze względu na brak wyrazów liniowych w x, y w równaniu (34). Jeśli $g_{11} = g_{22}$, to elipsa staje się okręgiem, a zmienne są skalowane w obu wymiarach o czynnik $g_{11}(= g_{22})$.

$\delta < 0$**.** Ponieważ $\delta = g_{11}g_{22} - {g_{12}}^2 < 0$, to mogą być równe zeru lub posiadać takie wartości, aby nierówność była spełniona. W szczególności mamy na pewno hiperbole w przypadku, gdy mają przeciwne znaki.

Hiperbola i elipsa posiadają zarówno osie symetrii jak i środek. Kąt nachylenia α osi symetrii hiperboli i elipsy względem osi x (lub y dla drugiej osi) kartezjańskiego układu współrzędnych dany jest wzorem:

$$\operatorname{tg} 2\alpha = \frac{2g_{12}}{g_{11} - g_{22}} \,. \tag{42}$$

Znak d^2 w dwu wymiarach Za pomocą dowolnej metryki możemy otrzymać kwadrat odległości dany wzorem (32), zarówno dodatni jak i ujemny. W dwu wymiarach łatwo określić, kiedy otrzymamy jaki znak. Jeśli z równania (34) o współczynnikach określonych przez elementy metryki otrzymamy elipsę, to $d^2(\mathbf{x}, \mathbf{y})$

jest zawsze nieujemne (lub zawsze ujemny, gdy jest to elipsa o współczynnikach urojonych — tab.(1)). W przypadku hiperboli, płaszczyzna podzielona jest przez asymptoty hiperbol na obszary o dodatnim i ujemnym kwadracie odległości. Metryka dająca proste równoległe albo narzuca wszędzie $d^2(\mathbf{x}, \mathbf{y}) \geq 0$, albo $d^2(\mathbf{x}, \mathbf{y}) \leq 0$.

Przypadek wielowymiarowy

W przestrzeni o dowolnym wymiarze nie możemy przeprowadzić takiej klasyfikacji krzywych jak w dwu wymiarach. Można jedynie określić rodzaj krzywej w dwuwymiarowych przekrojach. Ponieważ we wzorze (32) na kwadrat odległości występują iloczyny zmiennych, można, pamiętając że typ krzywej nie zależy od wyrazu stałego w równaniu (34), określić typ krzywej w poszczególnych płaszczyznach (np. xy, xz, yz w trzech wymiarach). Wykorzystujemy w tym celu elementy tensora metrycznego, odpowiadające interesującym nas wymiarom.

Znak d^2 w wielu wymiarach. W wielu wymiarach znak kwadratu odległości można określić badając znaki wartości własnych macierzy g. Jak wiadomo [TW] istnieje taka macierz P, że po dokonaniu przekształcenia

$$P^T \, \mathbf{g} \, P \, , \tag{43}$$

otrzymamy macierz posiadającą na przekątnej wartości -1, 0, 1, które oznaczają odpowiednio istnienie ujemnych, zerowych i dodatnich wartości własnych. Jeśli więc w tej macierzy pojawią się minus jedynki, będzie to oznaczało istnienie takich obszarów w przestrzeni, w których znak $d^2(\mathbf{x}_1, \mathbf{x}_2)$ jest ujemny.

Zastosowanie metryki w funkcjach radialnych

Obliczoną wg wzoru (32) odległość możemy wprowadzić do funkcji radialnych. Nie ma z tym żadnych problemów, o ile jest ona dodatnia. Taką gwarancję mamy, gdy metryka prowadzi do elips (elipsoid) lub prostych równoległych jako krzywych stałej odległości. Odległość taka nazywa się odległością Mahalanobisa [Was], można ją zapisać wzorem:

$$\begin{aligned} d(\mathbf{x}, \mathbf{u}_i) &= [(\mathbf{x} - \mathbf{u}_i)^T \, \mathbf{g} \, (\mathbf{x} - \mathbf{u}_i)]^{\frac{1}{2}} \\ &= \left[\sum_{\mu\nu} g_{\mu\nu} (x_\mu - u_{i\mu}) \, (x_\nu - u_{i\nu}) \right]^{\frac{1}{2}} , \end{aligned} \tag{44}$$

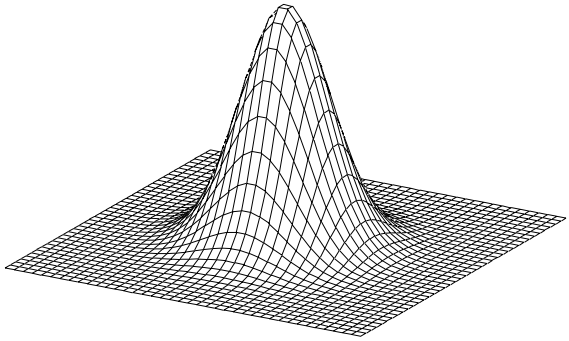
gdzie

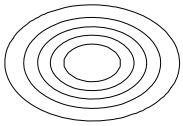
- $\mathbf{g}$,to metryka
- $\mathbf{x}$,to wektor wejściowy
- $\mathbf{u}_i$,to położenie i-funkcji radialnej .

Tak obliczana odległość pozwala na otrzymanie dowolnie obróconych, eliptycznych (elipsoidalnych) poziomic funkcji radialnej rys.(4, 5, 6, 7). Jeśli macierz g ma zerowe wartości własne, to odległość nie zależy od części ze zmiennych i w danym kierunku poziomice są prostymi. Gdy posługujemy się funkcjami gaussowskimi, metrykę g możemy zinterpretować jako odwrot ność macierzy kowariancji wektora wejściowego:

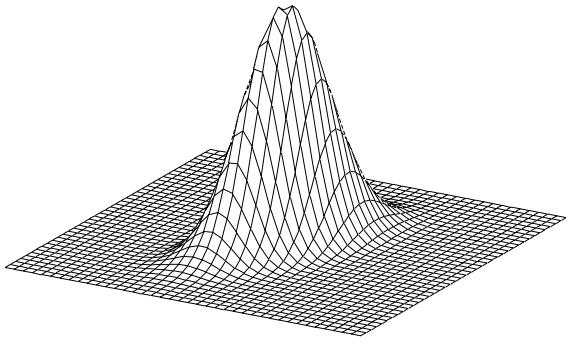
$$g = [E((\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T)]^{-1} , \tag{45}$$

gdzie $\mathbf{m}$ jest średnią tego rozkładu, zaś E oznacza wartość oczekiwaną. Zerowa wartość własna oznacza nieskończone rozmycie funkcji w jakimś kierunku. Na poniższych rysunkach przedstawiono kształty kilku takich funkcji oraz ich poziomic.

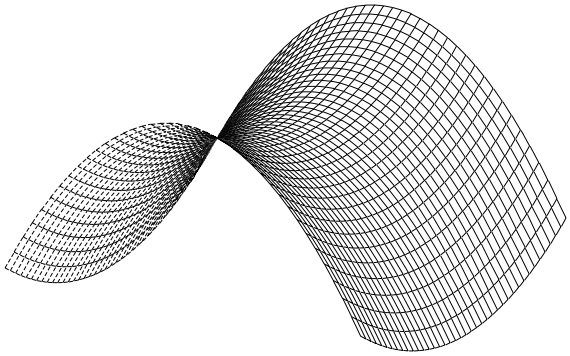


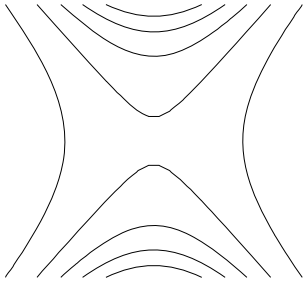


Rys.(4) Funkcja gaussowska, wielokrotność metryki euklidesowej $\mathbf{g} = \frac{1}{10}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

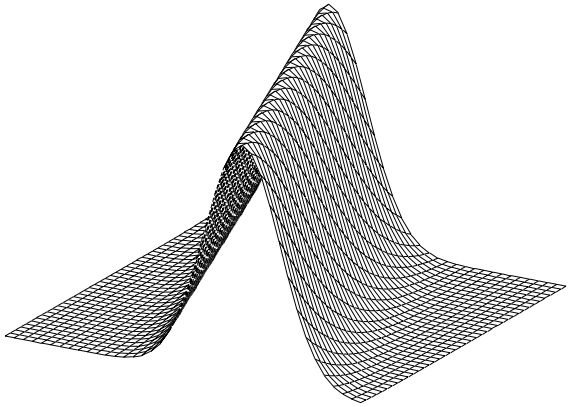


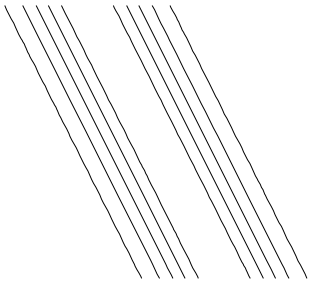
Rys.(5) Funkcja gaussowska, metryka o postaci $\mathbf{g} = \begin{pmatrix} \frac{1}{3} & -\frac{1}{10} \\ -\frac{1}{10} & \frac{1}{9} \end{pmatrix}$.





Rys.(6) Funkcja gaussowska, metryka o postaci $\mathbf{g} = \begin{pmatrix} -\frac{1}{15} & 0 \\ 0 & \frac{1}{45} \end{pmatrix}$.





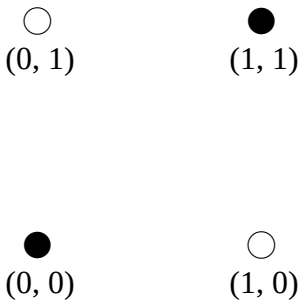
Rys.(7) Funkcja gaussowska, metryka o postaci $\mathbf{g} = \begin{pmatrix} \frac{1}{9} & \frac{1}{27} \\ \frac{1}{27} & \frac{1}{81} \end{pmatrix}$.

Ujemny kwadrat odległości jest dość kłopotliwy. Funkcję radialną typu gaus-

sowskiego można z taką wielkością obliczyć lecz rośnie ona nieograniczenie, gdy $d^2 \rightarrow -\infty$ rys.(6). Inne funkcje stają się funkcjami zmiennej zespolonej i nie wydaje się, aby był z nich jakiś pożytek.

6 Lokalne algorytmy uczenia

Lokalne algorytmy dokonują podziału przestrzeni danych wejściowych zgodnie z zasadą „dziel i rządź". Rozwiązanie złożonego problemu polega na jego podziale na mniej złożone podproblemy, rozwiązaniu ich, a następnie zbudowaniu na podstawie otrzymanych rozwiązań rozwiązania całości. Takie postępowanie prowadzi do uproszczenia algorytmu uczenia się sieci. Często stosowane wielowarstwowe sieci z jednostkami ukrytymi uczone metodą wstecznej propagacji błędu są przykładem przeciwnego, globalnego podejścia do problemu. Proces uczenia takich sieci jest długi oraz nie zawsze prowadzi do prawidłowego zakończenia, co wynika z skomplikowanego kształtu powierzchni funkcji błędu.



Rys.(8) Problem XOR.

Bodajże najprostszym rodzajem sieci jest jednowarstwowy perceptron. Ma on niewielkie możliwości - może dokonać podziału przestrzeni wejściowej jedynie

za pomocą linii prostej (hiperpłaszczyzny). Stąd problemy rozwiązywalne przez niego ograniczają się do tzw. liniowo–separowalnych. Przykładem problemu liniowo–nieseparowalnego jest problem XOR rys.(8). Za pomocą linii prostej nie możemy oddzielić białych punktów od czarnych. Można tego jednak dokonać w sposób następujący: przeprowadzić linię tak, aby w dowolny sposób oddzielała dwa punkty od pozostałych, a następnie w każdej otrzymanej parze oddzielić również linią punkt czarny od białego. Jest to przykład lokalnego algorytmu wykorzystującego większą liczbę prostych elementów - perceptronów.

Na podobnej zasadzie działa opisana w [Kv] sieć *Adaptive Mixture of Local Neural Networks*. Zbudowana jest z t identycznych sieci lokalnych N_i $i = 1 \ldots t$ i sieci bramkującej $S^{(g)}$ (indeks g od ang.*gating*). Sieć bramkująca ma tyle wejść co sieci N_i i t wyjść. Oznaczmy wartości otrzymywane z sieci lokalnych przez $\mathbf{x}_o^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots)$, zaś z sieci bramkującej przez $\mathbf{x}_o^{(g)} = (x_1^{(g)}, x_2^{(g)}, \ldots, x_t^{(g)})$. Załóżmy, że wszystkie wartości zawierają się w przedziale $(0, 1)$. Sieć bramkująca produkuje współczynniki proporcjonalności p_i

$$p_i = \frac{x_i^{(g)}}{\sum\limits_{(j=1)}^{(t)} x_j^{(g)}} \quad , i = 1 \ldots t \, . \tag{46}$$

Współczynniki p_i spełniają

$$\sum_{(j=1)}^{(t)} p_i = 1 \, .$$

Można je wykorzystać dwojako: traktując jako prawdobieństwa tego, że sieć N_i podaje prawidłową odpowiedź i wybrać odpowiedź z sieci dla, której p_i jest największe, bądź utworzyć wynik dawany przez całą sieć w postaci kombinacji liniowej

$$\tilde{\mathbf{x}}_o = p_1 \mathbf{x}_o^{(1)} + \ldots + p_i \mathbf{x}_o^{(t)} \, . \tag{47}$$

Wektor $\tilde{\mathbf{x}}_o$ jest ograniczony przez wektory $\mathbf{x}_o^{(i)}$ sieci lokalnych w tym sensie, że zachodzi nierówność

$$\bigvee_i \min_{1\le j\le t}\{x_i^{(j)}\} \le \widetilde{x}_i \le \max_{1\le j\le t}\{x_i^{(j)}\}\,. \tag{48}$$

Sieć taką uczy się na przykładach. Sieci lokalne uczone są zwykłą metodą gradientową (jeśli mają tylko dwie warstwy - wejściową i wyjściową), ewentualnie metodą wstecznej propagacji. Pochodna funkcji błędu sieci bramkującej ma trochę inną postać niż dla zwykłej sieci. Wprowadźmy funkcję błędu dla całej sieci

$$E = \frac{1}{2}\sum_{i=1}^{t} p_i(\mathbf{x}_o^{(i)} - \hat{\mathbf{x}}_o)^2\,, \tag{49}$$

gdzie

x_k jest aktywnością wyjściową k - neuronu w sieci lokalnej i,

$\hat{x}_o$ pożądaną wartością wyjściową całej sieci.

Oto pochodne funkcji błędu względem progów $\vartheta_i^{(g)}$ neuronów sieci bramkującej

$$\frac{\partial E}{\partial \vartheta_i^{(g)}} = f'(\xi_i^{(g)})\Big(g_i^{(g)} + \sum_l \frac{\partial E}{\partial \vartheta_l^{(g)}}\omega_{li}^{(g)}\Big)\,, \tag{50}$$

gdzie

$$g_i^{(g)} = \begin{cases} \dfrac{\frac{1}{2}(x_o^{(i)} - \hat{x}_o)^2 - E}{\sum_j x_j^{(g)}} & \text{, gdy neuron k należy do warstwy wyjściowej;} \\ 0 & \text{, gdy nie należy,} \end{cases}$$

$\omega_{(li)}^{(g)}$, to waga połączenia między neuronem l i i ,

$\xi_i^{(g)}$, to aktywacja i-neuronu $\xi_i^{(g)} = \sum_i \omega_{(ij)}^{(g)} x_j + \vartheta_i^{(g)}$,

x_j , to sygnał dochodzący od neuronu j,

f' , pochodna funkcji przejścia neuronu - zwykle sigmoidy.

Wartość $g_i^{(g)}$ jest proporcjonalna do różnicy między błędem lokalnej sieci i a błędem całej sieci. Jeśli sieć lokalna j daje mniejszy błąd, to $\vartheta_j^{(g)}$ neuronu j sieci bramkującej wzrasta, powodując wzrost jego aktywności wyjściowej, co prowadzi do wyboru sieci j jako dającej najlepszą odpowiedź. Okazuje się, że po odpowiednio długiej nauce wartości wyjściowe $\mathbf{x}_o^{(g)} = (x_1^{(g)}, x_2^{(g)}, \ldots, x_t^{(g)})$ dążą do posiadania jednej składowej bliskiej jedynce a pozostałych bliskich zeru. Oznacza, to że sieci lokalne specjalizują się w rozpoznawaniu pewnych danych wejściowych.

W pracy [Kv] znajdują się dwa przykłady zastosowania takich sieci. Pierwszym jest dodawanie dwu liczb dwubitowych:

$$\begin{array}{cccc} & & \alpha_1 & \alpha_2 \\ + & & \alpha_3 & \alpha_4 \\ \hline & \alpha_5 & \alpha_6 & \alpha_7 \end{array} \tag{51}$$

α_i mają wartości 0 lub 1. Takie dodawanie można przeprowadzić na szesnaście różnych sposobów, otrzymując liczby w zakresie od 0 do 6 (dziesiętnie). Dane wejściowe nie są liniowo separowalne. Do nauczenia się takiego dodawania potrzebna jest zwykła sieć o co najmniej dwu ukrytych neuronach. Mieszanina sieci lokalnych zbudowana jest z czterech sieci lokalnych – dwuwarstwowych o czterech neuronach wejściowych i trzech wyjściowych oraz także dwuwarstwowej sieci bramkującej o czterech wejściach i czterech wyjściach. Są to wszystko zwykłe perceptrony. Funkcja błędu jest uogólnieniem funkcji (49) poprzez dodatkowe sumowanie po wszystkich szesnastu relacjach wejście – wyjście. Jak się okazało, do prawidłowego rozpoznania wzorców wejściowych, używane są tylko dwie spośród czterech sieci lokalnych.

Drugim przykładem jest negacja ciągu czterech bitów w zależności od warto-

ści pierwszego:

$$F : (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \longrightarrow (\alpha_5, \alpha_5, \alpha_7)$$

tak, że

$$(\alpha_5, \alpha_5, \alpha_7) = \begin{cases} (\alpha_2, \alpha_3, \alpha_4) & \text{, gdy } \alpha_1 = 0; \\ (\overline{\alpha}_2, \overline{\alpha}_3, \overline{\alpha}_4) & \text{, gdy } \alpha_1 = 1. \end{cases}$$

Występuje tu także szesnaście kombinacji. W tym przypadku sieć składa się z dwu sieci lokalnych o czterech neuronach wejściowych i trzech wyjściowych oraz z dwuwarstwowej sieci bramkującej o czterech wejściach i dwu wyjściach. Sieć taka po nauczeniu zachowuje się w ten sposób, że jedna z sieci lokalnych specjalizuje się w przetwarzaniu danych o zerowym pierwszym bicie, zaś druga w danych o tym bicie ustawionym. Prawidłową odpowiedź (sieć lokalną dającą dobry wynik) wybiera w zależności od danych wejściowych sieć bramkująca, dzieląc w ten sposób dane wejściowe na dwa podzbiory.

Jak widać wprowadzenie sieci lokalnych pozwala na zmianę architektury sieci z „pionowej" – dla sieci z jednostkami ukrytymi na „poziomą". Ułatwia to znacznie naukę, gdyż eliminuję potrzebę ustalania ukrytych parametrów. Sieci lokalne są dwuwarstwowymi perceptronami wyspecjalizowanymi w rozwiązywaniu części problemu, zaś podział przestrzeni danych wejściowych dokonywany jest przez sieć bramkującą. Produkuje ona współczynniki p_i wskazujące prawdopodobieństwo uzyskania prawidłowego wyniku przez sieć lokalną i.

Omówimy teraz inny rodzaj sieci lokalnej służącej do aproksymacji funkcji typy $f : R^n \longrightarrow R^m$ oraz do klasyfikacji. Ma ona strukturę drzewiastą i dokonuje sukcesywnego podziału dziedziny funkcji na sekwencję zagnież dżonych obszarów. Podział jest dokonywany przez sieci bramkujące, zaś w każdym obszarze funkcja jest przybliżana poprzez jak najprostszą zależność (stałą bądź liniową) przez tzw. sieci ekspertowe.

Na rys.(16) przedstawiono strukturę dwupoziomowej sieci HME (ang. *Hierarchical Mixtures of Experts*) [JJ]. Na wejściach (liściach) znajdują się sieci ekspertowe. Realizują one funkcję wektorową, zwaną uogólnioną liniową (ang. *generalized linear*), dając w wyniku wektor μ_{ij}

$$\mu_{ij} = f(\mathbf{U}_{ij}\mathbf{x})\,, \tag{52}$$

gdzie i, j, to indeksy sieci ekspertowej, $\mathbf{U}_{ij}$, to macierz przekształcenia liniowego, zaś f, to zadana funkcja przejścia. Jeśli sieć ma być używana do aproksymacji, to funkcja f będzie funkcją tożsamościową. Do celów klasyfikacji f będzie funkcją logistyczną. W węzłach drzewa umieszczone są sieci bramkujące, także realizujące uogólnione funkcje liniowe. Na wyjściu i najwyższej sieci bramkującej otrzymujemy wartość

$$g_i = \frac{e^{\xi_i}}{\sum\limits_k e^{\xi_k}}\,, \tag{53}$$

gdzie

$$\xi_i = \mathbf{v}_i^T\mathbf{x}\,, \tag{54}$$

$\mathbf{v}_i$ jest wektorem wag. Wielkości g_i sumują się do jedynki oraz są dodatnie. Za ich pomocą przeprowadzony jest podział przestrzeni wejściowej. W analogiczny sposób działają sieci bramkujące na niższym poziomie. Produkują one wielkości

$$g_{j|i} = \frac{e^{\xi_{ij}}}{\sum\limits_k e^{\xi_{ik}}}\,, \tag{55}$$

gdzie

$$\xi_{ij} = \mathbf{v}_{ij}^T\mathbf{x}\,.$$

Indeksy $i\,j$ oznaczają wyjście j sieci bramkującej i. $g_{i|j}$ są także dodatnie, sumują się do jedynki i wprowadzają podział przestrzeni wewnątrz podziału ustalone przez sieć bramkującą wyższego rzędu. Na każdym poziomie drzewa, poza

liśćmi, tworzona jest kombinacja liniowa wyników otrzymanych z sieci leżących poniżej. Współczynnikami tej kombinacji są oczywiście wartości g. Na drugim poziomie mamy

$$\mu_i = \sum_j g_{i|j}\mu_{ij}\,, \tag{56}$$

na pierwszym zaś

$$\mu = \sum_i g_i\mu_i\,. \tag{57}$$

Wektor μ jest wynikiem całej sieci. Zarówno współczynniki g jak i wektory μ zależą od wektora wejściowego $\mathbf{x}$. Funkcja realizowana przez sieć jest zatem nieliniowa. Można pokazać na przykładzie jednopoziomowej sieci jak dokonuje ona podziału przestrzeni wejściowej. Mamy tylko jedną sieć bramkującą i dwie podłączone do niej sieci ekspertowe – liście. Wartość g_1 wyraża się wzorem

$$g_1 = \frac{e_1^\xi}{e_1^\xi + e_2^\xi} = \frac{1}{1 + e^{-(\mathbf{v}_1 - \mathbf{v}_2)^T\mathbf{x}}}. \tag{58}$$

Jest to funkcja logistyczna (sigmoida), której oś podejmowania decyzji (czyli podziału przestrzeni) wyznaczona jest przez wektor $\mathbf{v}_1 - \mathbf{v}_2$. Wynikiem działania sieci jest kombinacja $g_1\mu_1 + g_2\mu_2$. Ponieważ $g_2 = 1 - g_1$, to wzdłuż osi podziału mamy $g_1 = g_2 = \frac{1}{2}$ i obie sieci ekspertowe mają jednakowy udział w tworzeniu wyniku końcowego. Gdy odsuwamy się od osi podziału, jedna z sieci ma większy wkład od drugiej. W ten sposób wyniki z obu sieci ekspertowych są brane pod uwagę. Taki podział przestrzeni wejściowej nazywany jest „miękkim" (ang. *soft*). Wielkość wektora $\mathbf{v}_1 - \mathbf{v}_2$ określa własności funkcji logistycznej. Dla dużego wektora $\mathbf{v}_1 - \mathbf{v}_2$ podział jest ostry i realizowana przez sieć funkcja jest funkcją kawałkami liniową, w przeciwnym wypadku wynik działania sieci zbliża się do średniej $\frac{1}{2}\mu_1 + \frac{1}{2}\mu_2$.

Działanie sieci typu HME można opisać statystycznie jako podejmowanie na każdym poziomie (posuwając się z góry na dół) stochastycznych decyzji, zagnieżdżających się i tworzących drzewo. Możemy interpretować $g_i(\mathbf{x}, \mathbf{v}_i^0)$ oraz $g_{j|i}(\mathbf{x}, \mathbf{v}_{ij}^0)$ jako prawdobieństwa związane z decyzjami odpowiednio na pierwszym i na drugim poziomie drzewa. Indeks 0 oznacza „prawdziwe" wartości parametrów [JJ]. Pojawiające się dalej wielkości bez indeksów przypisane są konkretnym realizacjom tej sieci. Rozkład opisujący te prawdopodobieństwa jest rozkładem wielomianowym [JJ], [Z]. Jest on zdefiniowany następująco. Weźmy m niezależnych doświadczeń, mogących zakończyć się jednym z n wyników. p_i jest prawdopodobieństwem i–tego wyniku. Niech Y_i będzie liczbą doświadczeń zakończonych wynikiem $i, i = 0, 1, \ldots n$. Łączny rozkład zmiennych losowych Y_i nazywamy rozkładem wielomianowym [Z]. Rozkład ten dany jest wzorem:

$$P(Y_1 = y_1, Y_2 = y_2, \ldots Y_n = y_n) = \frac{m!}{(y_1!)(y_2!)\ldots(y_n!)} p_1^{y_1} p_2^{y_2} \ldots p_n^{y_n} . \quad (59)$$

Zachodzi:

$$\sum_{i=1}^{n} p_i = 1 . \quad (60)$$

Rozklad ten należy do rodziny wykładniczej rozkładów [Z], [JJ].

Podjęte decyzje tworzą ścieżkę wzdłuż gałęzi drzewa i następuje przypisanie argumentowi $\mathbf{x}$ wartości wyjściowej $\mathbf{y}$. W wyniku działania sieci eksperckiej otrzymujemy

$$\mu_{ij}^0 = f(\mathbf{U}_{ij}^0 \mathbf{x}) , \quad (61)$$

gdzie f jest funkcją przejścia. Otrzymana wartość μ_{ij}^0 jest średnią pewnego rozkładu warunkowego P modelującego dane:

$$P(\mathbf{y}|\mathbf{x}, \theta_{ij}^0) . \quad (62)$$

Parametry tego rozkładu składają się z wag $\mathbf{U}_{ij}^0$ oraz dyspersji ϕ_{ij}^0 [JJ]:

$$\theta_{ij}^0 = \begin{bmatrix} \mathbf{U}_{ij}^0 \\ \phi_{ij}^0 \end{bmatrix} . \tag{63}$$

Całkowite prawdopodobieństwo otrzymania $\mathbf{y}$ z $\mathbf{x}$ po przejściu całego drzewa jest iloczynem prawdopodobieństw z obu poziomów:

$$P(\mathbf{y}|\mathbf{x}, \theta^0) = \sum_i g_i^0(\mathbf{x}, \mathbf{v}_j^0) \sum_j g_{j|i}(\mathbf{x}, \mathbf{v}_{ij}^0) P(\mathbf{y}|\mathbf{x}, \theta_{ij}^0) . \tag{64}$$

Rozkład $P(\mathbf{y}|\mathbf{x}, \theta_{ij}^0)$ jest wybierany w zależności od zastosowania sieci z tzw. rodziny wykładniczej rozkładów [Ama], [Z]. W przypadku regresji jest to rozkład gaussowski $P(\mathbf{y}|\mathbf{x}, \theta_{ij}) = exp\Big(-\frac{1}{2\,\sigma^2}(\mathbf{y} - \mu_{ij})^T(\mathbf{y} - \mu_{ij})\Big)$ i parametry dyspersji to elementy macierzy kowariancji $\Sigma = \sigma^2\,\mathbf{I}$, zaś w przypadku klasyfikacji rozkład Bernoulliego $P(y|\mathbf{x}, \theta_{ij}) = \mu_{ij}^y\,(1 - \mu_{ij})^{1-y}$, w którym stała μ_{ij} jest prawdopodobieństwem zakwalifikowania $\mathbf{x}$ jako sukcesu. Warto zauważyć, że rozkład Bernoulliego nie posiada parametrów dyspersji.

Prawdopodobieństwa dzielimy na *a priori* i *a posteriori*. Tymi pierwszymi są $g_i(\mathbf{x}, \mathbf{v}_i^0)$ oraz $g_{j|i}(\mathbf{x}, \mathbf{v}_{ij}^0)$, gdyż obliczane są tylko na podstawie znajomości wartości wejściowej $\mathbf{x}$. Prawdopobieństwa *a posteriori* możemy obliczyć, znając jednocześnie wartości wyjściowe, za pomocą wzoru Bayesa:

$$h_i = \frac{g_i \sum\limits_j g_{j|i} P_{ij}(\mathbf{y})}{\sum\limits_i g_i \sum\limits_j g_{j|i} P_{ij}(\mathbf{y})} \tag{65}$$

oraz

$$h_{j|i} = \frac{g_{j|i} P_{ij}(\mathbf{y})}{\sum\limits_j g_{j|i} P_{ij}(\mathbf{y})} \tag{66}$$

dla pierwszego i drugiego poziomu, gdzie $P_{ij}(\mathbf{y}) = P(\mathbf{y}|\mathbf{x}, \theta_{ij}^0)$.

Możemy teraz znaleźć logarytm estymatora wiarygodności największej parametrów θ, opartego na zbiorze relacji wejście–wyjście $\mathcal{X} = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_{t=1}^{N}$:

$$l(\theta; \mathcal{X}) = \sum_t ln \sum_i g_i^{(t)} \sum_j g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)}) \tag{67}$$

Mając taki estymator możemy szukać parametrów sieci, która będzie najlepiej przybliżała dane treningowe, poprzez różniczkowanie tej funkcji względem parametrów θ i znalezienia jej maksimum. Na tej podstawie zostały skonstruowane algorytmy uczenia parametrów zarówno w wersji *on–line* jak i *batch*'owej [JJ].

Ciekawsze jest rozwiązanie problemu nauki sieci HME za pomocą algorytmu EM.

Algorytm EM jest statystyczną metodą uczenia sieci neuronowych. Jeśli dane przetwarzane przez sieć są zaszumione i w związku z tym opisywane jakimś rozkładem prawodopodobieństwa , możemy spróbować opisać działanie sieci w sposób statystyczny [Ama], [PG]. Dla danej sieci określmy warunkowy rozkład prawdopodobieństwa $P(\mathbf{x}|\mathbf{y};\theta))$. Służy on do obliczania prawdopodobieństwa otrzymania w wyniku działania sieci wektora $\mathbf{y}$, jeśli na wejściu podaliśmy wektor $\mathbf{x}$. Parametry $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$ określają ten rozkład i możemy je zmieniać. Determinują one działanie sieci. Możemy zdefiniować n-wymiarową rozmaitość S zawierającą wszystkie możliwe do uzyskania przez zmiany θ rozkłady prawdopodobieństwa. Parametry θ pełnią rolę współrzędnych w tej przestrzeni. Z kolei można w rozmaitości θ wyróżnić podrozmaitość M, na którą będą się składać wszystkie rozkłady opisujące daną sieć. Dane służące do nauki sieci określają pewien rozkład prawdopodobieństwa. W przypadku, gdy nie posiadamy pełnych danych, zamiast jednego rozkładu (punktu) w S otrzymamy podrozmaitość D, zawierającą możliwe rozkłady danych. Nauka sieci, czyli poszukiwanie takich parametrów θ, aby wyniki dawa-

ne przez sieć były jak najbardziej zbliżone do danych relacji wejście—wyjście, polega na znalezieniu takich punktów w M i D, które minimalizują odległość pomiedzy tymi podrozmaitościami. Punkt należący do M daje parametry tej sieci, a punkt należący do D pozwala znaleźć brakujące dane. Algorytm EM jest iteracyjną metodą poszukiwania estymatora największej wiarygodności, czyli punktu w M oraz jednocześnie znajdowania warunkowej wartości oczekiwanej nieznanych zmiennych–punktu w D. Składają się nań trzy kroki:

1. Ustalenie początkowych parametrów opisujących sieć. Zostaje w ten sposób wyznaczony punkt w M. Jest to początkowy stan sieci.

2. Krok E (ang. *Estimation*). Na podstawie znanego teraz rozkładu opisującego sieć należy policzyć warunkową wartość oczekiwaną. Warunkiem jest zachowanie znanej nam części danych. Otrzymujemy tymczasowy punkt należący do podrozmaitości D.

3. Krok M (ang. *Maxymalization*). Na podstawie znanych, teraz kompletnych danych, trzeba obliczyć estymator największej wiarygodności, czyli znaleźć punkt należący do M, przedstawiający rozkład prawdopodobieństwa realizowany przez sieć najlepiej przybliżającą znany na tym etapie rozkład danych.

Takie naprzemienne wykonywanie kroków E i M jest zbieżne i prowadzi do uzyskania punktów minimalizujących (lokalnie) odległość między M i D [Ama], [JJ]. Punkty te opisują najlepszą sieć i dostarczają brakujące dane.

Aby móc zastosować algorytm EM do nauki sieci HME, należy wprowadzić owe zmienne ukryte. Niech zbiór $\mathcal{Y}$ będzie zbiorem danych kompletnych, posiadającym obok znanych zmiennych $\mathcal{X}$ zmienne ukryte $\mathcal{Z}$. Zbiór $\mathcal{X}$ zawiera zatem zmienne niekompletne. Odpowiedni rozkład prawdopodobieństwa oznaczmy przez $P(\mathbf{x}, \mathbf{z}|\mathbf{x}, \theta)$. Jest to rozkład warunkowy zmiennych $\mathbf{y}$ i $\mathbf{z}$. Logarytm esty-

matora największej wiarygodności $l_c(\theta;\mathcal{Y})$ obliczamy zgodnie z tym rozkładem i odnosi się on do zbioru kompletnego $\mathcal{Y}$ (indeks c). Możemy teraz wykonać kroki algorytmu EM. Krok E polega na obliczeniu wartości oczekiwanej estymatora dla kompletnego zestawu zmiennych:

$$Q(\theta,\theta^{(p)}) = E[l_c(\theta;\mathcal{Y})|\mathcal{X}]\,, \tag{68}$$

gdzie p numeruje kolejne iteracje.

Jest to warunkowa wartość oczekiwana, gdyż żądamy, aby znane dane znajdujące się w zbiorze $\mathcal{X}$ pozostały bez zmian (jest to warunek pozostania w podrozmaitości D).

Krok M maksymalizuje tak otrzymaną funkcję ze względu na parametry $\theta^{(p)}$, dając nowy ich zestaw:

$$\theta^{(p+1)} = arg\max_{\theta} Q(\theta,\theta^{(p)})\,, \tag{69}$$

co prowadzi do ulepszenia estymatora uzyskanego na podstawie danych kompletnych. Interesujący nas tak naprawdę estymator oparty o niekompletne dane $l(\theta;\mathcal{X})$ także powiększa się z kroku na krok, dążąc do wartości stacjonarnej [JJ].

W celu zastosowania algorytmu EM do nauki sieci HME wprowadza się dodatkowe, ukryte, zmienne z_i i $z_{j|i}$. Na każdym poziomie drzewa tylko jedna z nich może mieć wartość jeden, pozostałe mają wartość zero. Są to więc pewnego rodzaju flagi. Przez z_{ij} oznaczmy iloczyn zmiennych z_i i $z_{j|i}$. Flagi owe określają, która z sieci eksperckich została wybrana podczas działania sieci. Rozkład prawdopodobieństwa zawierający te nieznane zmienne można zapisać, wykorzystując ich własności w postaci:

$$\begin{aligned} P(\mathbf{y}^{(t)}, z_{ij}^{(t)}|\mathbf{x}^{(t)},\theta) &= g_i^{(t)} g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)}) \\ &= \prod_i \prod_j \{g_i^{(t)} g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)})\}^{z_{ij}^{(t)}}\,. \end{aligned} \tag{70}$$

Stąd wzór na logarytm estymatora największej wiarygodności, obliczonego na podstawie kompletnych danych wygląda następująco:

$$\begin{aligned} l_c(\theta, \mathcal{Y}) &= \sum_t \sum_i \sum_j z_{ij}^{(t)} ln \left\{ g_i^{(t)} g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)}) \right\} \\ &= \sum_t \sum_i \sum_j z_{ij}^{(t)} \left\{ ln\, g_i^{(t)} + ln\, g_{j|i}^{(t)} + ln\, P_{ij}(\mathbf{y}^{(t)}) \right\} . \end{aligned} \tag{71}$$

Porównując ten wzór ze wzorem (67) na logarytm estymatora określonego na podstawie danych niekompletnych widzimy, że w wyniku wprowadzenia zmiennych z_i i $z_{j|i}$ jego struktura uległa uproszczeniu. Mamy teraz sumę logarytmów zamiast logarytmu sumy. Wartość oczekiwana parametrów θ^p, obliczona za pomocą tego estymatora jest następująca:

$$Q(\theta, \theta^{(p)}) = \sum_t \sum_i \sum_j h_{ij}^{(t)} \left\{ ln\, g_i^{(t)} + ln\, g_{j|i}^{(t)} + ln\, P_{ij}(\mathbf{y}^{(t)}) \right\} , \tag{72}$$

gdzie $h_{ij} = h_i h_{j|i}$. Wykorzystano tu fakt, że $E[z_{ij}^{(t)}|\mathcal{X}] = h_{ij}^{(t)}$ [JJ]. Mając tak określoną funkcję zmiennych θ, możemy przystąpić do wykonywania kroku M, czyli szukania maksimum tej funkcji ze względu na parametry θ. Jak pamiętamy, są to parametry $\mathbf{U}_{ij}$ sieci eksperckich oraz $\mathbf{v}_i$ i $\mathbf{v}_{ij}$ sieci bramkujących obu poziomów. Sposób zależności funkcji Q od parametrów poszczególnych sieci powoduje, że otrzymamy trzy niezależne problemy maksymalizacyjne:
dla parametrów sieci eksperckich

$$\theta_{ij}^{(p+1)} = arg \max_{\theta_{ij}} \sum_t h_{ij}^{(t)}\, ln\, P_{ij}(\mathbf{y}^{(t)}) ; \tag{73}$$

dla parametrów sieci bramkujących pierwszego poziomu

$$\mathbf{v}_i^{(p+1)} = arg \max_{\mathbf{v}_i} \sum_t \sum_k h_k^{(t)}\, ln\, g_k^{(t)} \tag{74}$$

oraz dla parametrów sieci bramkujących poziomu drugiego

$$\mathbf{v}_{ij}^{(p+1)} = arg \max_{\mathbf{v}_{ij}} \sum_t \sum_k h_k^{(t)} \sum_l h_{l|k}^{(t)} ln\, g_{l|k}^{(t)} . \tag{75}$$

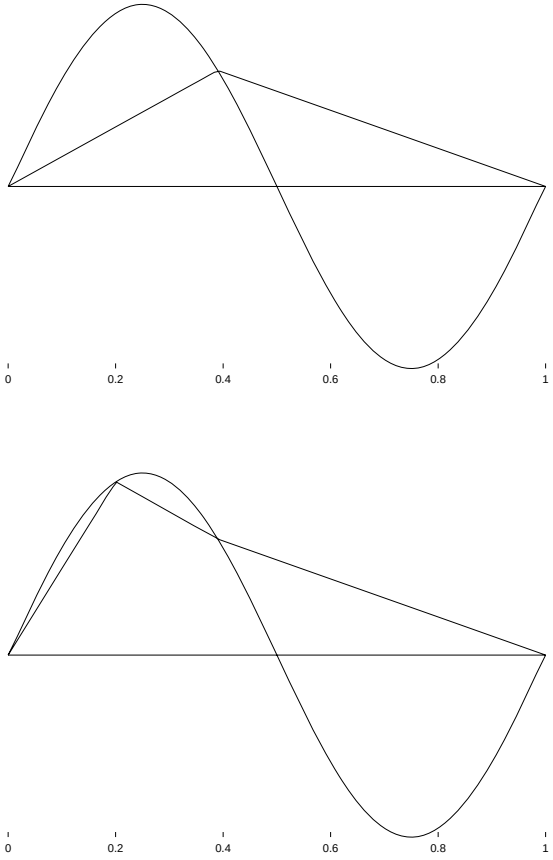
Zastosowanie algorytmu EM i wprowadzenie ukrytych zmiennych z doprowadziło do uzyskania szeregu niezależnych zagadnień maksymalizacyjnych, z których każde odnosi się również do pewnego estymatora największej wiarygodności [JJ]. Problem nauki całej sieci rozpadł się na niezależne zagadnienia maksymalizacyjne określone dla poszczególnych sieci eksperckich oraz, także niezależne, zagadnienia klasyfikacyjne dla sieci bramkujących.

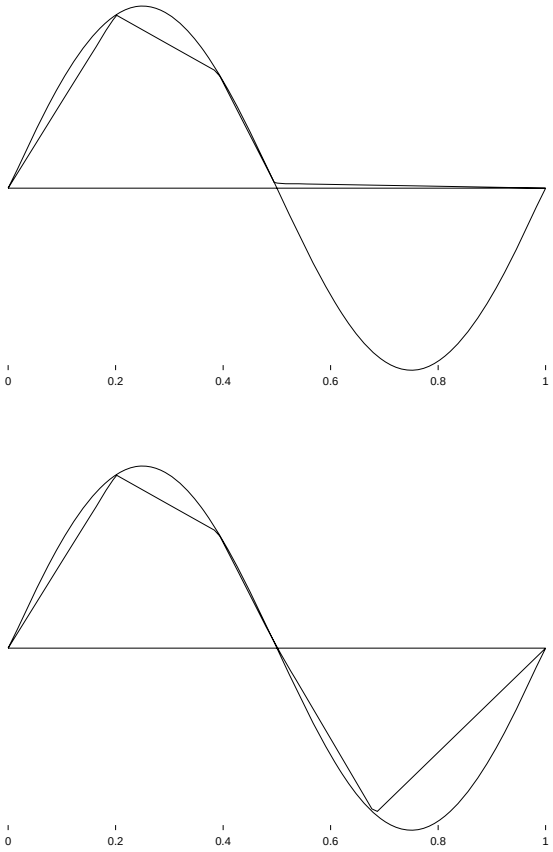
Przykład prostego algorytmu lokalnego

Prosty algorytm przybliżający funkcję za pomocą odcinków prostej. Dane podawane są w postaci par (x, y), uzyskanych w wyniku losowego próbkowania funkcji $f : R^1 \longrightarrow R^1$. Po otrzymaniu dwu takich par (x_1, y_1), (x_2, y_2), punkty przez nie reprezentowane są łączone prostą $l_1(x)$. Zadając maksymalną wartość błędu przybliżenia ϵ patrzymy, czy dla następnych punktów zachodzi $|y_i - l_1(x_i)| \geq \epsilon$. Jeśli tak jest, to prostą $l_1(x)$ dzielimy na dwie w punkcie x_i; jeśli $x_1 \leq x_i \leq x_2$ lub prowadzimy nową prostą do x_i, jeżeli ten znajduje się poza tym przedziałem. Dodajemy zatem nowe odcinki prostych, gdy posiadane dotychczas przybliżenie jest niezadawalające. Jest to przykład algorytmu działającego w oparciu o stale napływające dane (ang. *on–line*). Podział przestrzeni wejściowej (czyli prostej R^1) jest ostry, dokonany za pomocą prostokątnej funkcji przynależności. Jeśli nowy punkt znajdzie się pomiędzy dwoma już znanymi, to wartość funkcji w tym punkcie obliczana jest na podstawie funkcji liniowej przechodzącej przez te dwa punkty, bez żadnego udziału sąsiednich przybliżeń. Użycie takiej funkcji prowadzi do linearyzacji struktury sieci. W tym prymitywnym algorytmie wystarczy przechowywć tylko punkty definiujące proste.

Przykład jego działania pokazany jest na rys.(9), dla funkcji $sin(2\pi x), x \in$

$[0, 1], \epsilon = 0.4$. Duża wartość ϵ została wybrana w celu umożliwienia wizualizacji procesu przybliżania.





Rys.(9) Kolejne kroki przybliżenia dokonywanego poprzez dodawanie odcinków prostej.

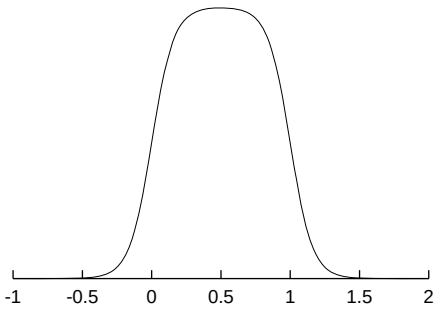
Aby wygładzić funkcję przybliżającą możemy wprowadzić nową, rozmytą funcję dzielącą przestrzeń wejścową. W przybliżeniu będą więc uczestniczyły wartości funkcji liniowych sąsiednich przedziałów.

Niech będzie to iloczyn dwu funkcji sigmoidalnych o następującej postaci:

$$\chi(x, x_p, x_k) = \frac{1}{1 + e^{-\frac{x - x_p}{\beta}}} * \frac{1}{1 + e^{-\frac{x_k - x}{\beta}}}, \tag{76}$$

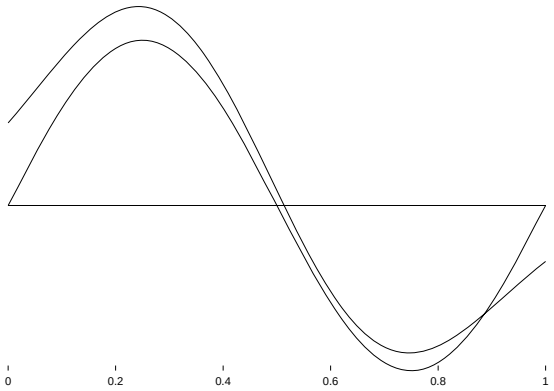
gdzie parametr β reguluje nachylenie zboczy tej funkcji, zaś x_p, x_k to początek i koniec przedziału.

Na rys.(10) widzimy funkcję $\chi(x, 0, 1)$ z parametrem $\beta = 0.08$.

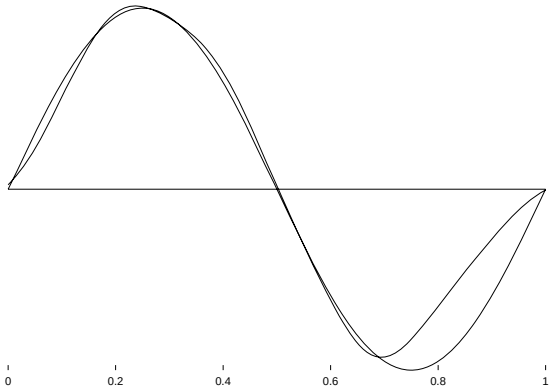


Rys.(10) Rozmyta funkcja podziału $\chi(x, 0, 1)$.

Przybliżenia uzyskane po wprowadzeniu takiej funkcji podziału z globalnym parametrem β widzimy na rys.(11) dla $\beta = 0.15$ i na rys.(12) dla $\beta = 0.05$.



Rys.(11) Wygładzone przybliżenie, $\beta = 0.15$.



Rys.(12) Wygładzone przybliżenie, $\beta = 0.05$.

Wydaje się konieczne znajdowanie wielkości β dla każdego przedziału oddzielnie, iteracyjnie w metodzie *on—line* lub poprzez minimalizacje kwadratowej funkcji błędu względem każdego β, na podstawie zbioru treningowego w wersji *batch*'owej.

Algorytm ten może pracować z danymi do nauki zebranymi w zadany zbiór $\{(x_i, y_i)\}_{i=1}^{n}$ (ang. *batch*). Trzeba określić położenie odcinków prostych, przy założeniu, że łączyć one będą punkty należące do tego zbioru. Weźmy trzy kolejne punkty $(x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2})$. Niech skrajne punkty łączy prosta $l(x)$ o równaniu:

$$y = \frac{y_i - y_{i+2}}{x_i - x_{i+2}}\, x + \frac{y_{i+2}x_i - y_i x_{i+2}}{x_i - x_{i+2}}\,. \tag{77}$$

Możemy policzyć wartość tego liniowego przybliżenia dla x_{i+1} i porównać z żądaną wartością y_{i+1}, obliczając błąd:

$$\begin{aligned} Err(x_{i+1}) &= \Big| l(x_{i+1}) - y_{i+1} \Big| \\ &= \left| \frac{y_i(x_{i+1} - x_{i+2}) - y_{i+1}(x_i - x_{i+2}) + y_{i+2}(x_i - x_{i+1})}{x_i - x_{i+2}} \right|. \end{aligned} \tag{78}$$

Zdefiniujmy teraz w punkcie x_{i+1} dyskretną drugą pochodną przybliżanej funkcji f:

$$d''_f(x_{i+1}) = \frac{2\left(\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} + \frac{y_i - y_{i+1}}{x_{i+1}-x_i}\right)}{x_{i+2}-x_i}\,. \tag{79}$$

Jej dziwna forma wynika z tego, że jest obliczana na nierównomiernej siatce punktów x_i, x_{i+1}, x_{i+2}. W przypadku, gdy punkty leżą oddalone o stałą odległość h, wzór ten przechodzi w znany symetryczny wzór na drugą pochodną: $d''_f(x_{i+1}) = \frac{y_i - 2\,y_{i+1} + y_{i+2}}{h^2}$. Zauważyć można następującą zależność:

$$\frac{1}{2}|d''_f(x_{i+1})|(x_{i+2}-x_{i+1})(x_{i+1}-x_i) = Err(x_{i+1})\,. \tag{80}$$

Warunek $Err(x_{i+1}) \leq \epsilon$ można więc przełożyć na warunek na tak policzoną drugą pochodną w punkcie x_{i+1}:

$$\frac{1}{2}|d''_f(x_{i+1})|(x_{i+2}-x_{i+1})(x_{i+1}-x_i) \leq \epsilon\,. \tag{81}$$

Skalownie pochodnej poprzez szerokości przedziałów pozwala wyeliminować skutki zamiany zmiennych typu $x \to \alpha x$, które zmieniają drugą pochodną, a nie mają wpływu na popełniany błąd.

W tej wersji algorytmu umiejscawianie odcinków prostej będzie więc przeprowadzane na podstawie wartości obliczonej we wszystkich punktach (poza krańcowymi) zbioru $\{(x_i, y_i)\}_{i=1}^n$ drugiej pochodnej i zadanej wartości błędu ϵ.

7 Zastosowanie funkcji odległości do klasyfikacji wzorców bitowych

Wzorcem bitowyn b o długości n nazywamy ciąg n wartości 0 lub 1, $b = \{b_1, b_2, \ldots, b_n\}$. Przestrzeń, do której należą takie ciągi, oznaczmy przez B^n (analogia do R^n),

gdzie $B = \{0, 1\}$. Można ją przedstawić jako zbiór wierzchołków n wymiarowego hipersześcianu o jednostkowym boku.

Klasyfikacja polega na przydzieleniu wzorców wejściowych do klas, których klasyfikator należy nauczyć na podstawie podanych przykładów. Musimy zatem posiadać zbiór przykładowych wzorców wraz z opisem klas, do których one należą. Ich liczba powinna wystarczyć do określenia wszystkich parametrów klasyfikatora. Zadaniem klasyfikatora jest prawidłowe przypisanie klasy znanym wzorcom oraz określenie klasy nieznanego wcześniej wzorca - generalizacja lub stwierdzenie, że nie potrafi dokonać klasyfikacji. Klasy opisane są poprzez należące do nich wzorce np. wzorzec B należy do klasy zawierającej wzorzec A.

Konstrukcję takiego klasyfikatora można oprzeć na funkcji odległości, zdefiniowanej na przestrzeni B^n i posiadającej podlegające adaptacji parametry. Poprzez dobór odpowiednich wartości tych parametrów można osiągnąć żądaną klasyfikację. Odległość to funkcja $d : B^n \times B^n \rightarrow R^+$, spełniająca następujące warunki :

– $d(x, y) = 0 \Leftrightarrow x = y$

– $d(x, y) = d(y, x)$ symetria

– $d(x, y) + d(y, z) \geq d(x, z)$ nierówność trójkąta.

Działanie klasyfikatorów opartych na funkcji odległości polega na zamianie relacji wejścia – wyjścia zadanych w postaci zbioru do nauki na warunki nałożone na funkcję odległości. Funkcja wyjściowa jest określona następująco. Dla wartości wejściowych znanych już klasyfikatorowi jest ona równa odpowiedniej wartości wyjściowej (także znanej). Dla nowych wartości wejściowych zależy od wartości znanych wzorców leżących w odległości mniejszej, niż pewne τ od wzorca wejściowego. Wartością wyjściową klasyfikatora jest np. stwierdzenie, że dwa

wzorce należą do tej samej klasy, bądź że mają tę samą parzystość.

W celu użycia funkcji odległości do klasyfikacji wzorców bitowych lepiej nałożyć na nią dalsze ograniczenia, aby uzyskać ostre rozgraniczenia dla wzorców należących do różnych klas. Niech więc, jeśli wzorce należą do tej samej klasy, ich odległość wynosi 0, a w przeciwnym wypadku 1. Prowadzi to do następujących warunków na funkcję d:

$$\begin{aligned} d(x,y) = 0 \wedge d(y,z) = 0 &\Rightarrow d(x,z) = 0 \\ d(x,y) = 1 \wedge d(y,z) = 0 &\Rightarrow d(x,z) = 1 \\ d(x,y) = 0 \wedge d(y,z) = 1 &\Rightarrow d(x,z) = 1 \\ d(x,y) = 1 \wedge d(y,z) = 1 &\Rightarrow d(x,z) \in \{0,1\}\,. \end{aligned} \tag{82}$$

Klasyfikator służący do klasyfikacji wzorców na podstawie tzw. własności (ang. *features*) został opisany w [DD]. Własności rozumiane są jako ustawienia bitów na konkretnych pozycjach wzorca np. wyróżnikiem klasy jest jedynka na drugiej pozycji wzorca. Odpowiednią funkcję odległości dla wzorców o długości n można zdefiniować następująco

$$d_\lambda(a,b) = \sum_{i=1}^{n} \lambda_i |a_i - b_i|\,, \tag{83}$$

gdzie

$$\lambda_i \in R^+,$$

$$a = (a_1, a_2, \ldots, a_n),$$

$$b = (b_1, b_2, \ldots, b_n).$$

Taka funkcja spełnia wszystkie warunki (82) [DD].

Pojemność klasyfikatora działającego w oparciu o funkcję d_λ można określić następująco. Niech p będzie liczbą λ_i równych zeru. Wartości odpowiednich

składowych wzorca nie są więc istotne dla zaklasyfikowania ich do danej klasy. Do klasy może zatem należeć 2^p różnych wzorców. Ilość klas równa jest ilości możliwych umieszczeń p nieistotnych wartości w ciągu o długości n i wynosi $\binom{n}{p}$. Możliwe sposoby klasyfikacji dla wzorców o długości $n = 2$ podane są w tabelce tab.(2).

Można teraz podać przykład konstrukcji takiej funkcji dla trzech wzorców $A, B, C, n = 7$[DD]:

$$A = (0, 1, 0, 1, 0, 1, 1),$$

$$B = (0, 1, 1, 1, 0, 1, 0),$$

$$C = (1, 1, 0, 1, 1, 1, 1).$$

Chcemy je tak sklasyfikować, aby $class(A) = class(B) \neq class(C)$, czyli wzorce a i B należą do tej samej klasy, zaś wzorzec C do innej. Takie założenia stanowią zbiór danych do nauki klasyfikatora.

Odlegoci	Metryka	Klasyfikacja
$d_\lambda = \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}$	$\lambda = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$C_1 = \{(0,0),(0,1)\}$ $C_2 = \{(1,0),(1,1)\}$
$d_\lambda = \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}$	$\lambda = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$C_1 = \{(0,0),(1,0)\}$ $C_2 = \{(0,1),(1,1)\}$
$d_\lambda = \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}$	$\lambda = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$C_1 = \{(0,0),(0,1),(1,0),(1,1)\}$
$d_\lambda = \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}$	$\lambda = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$C_1 = \{(0,0)\}$ $C_2 = \{(0,1)\}$ $C_3 = \{(1,0)\}$ $C_4 = \{(1,1)\}$

Tab. (2) Klasyfikacje wzorców o długości $n = 2$ [DD].

Funkcja odległości musi zatem spełniać następujące warunki:

$$d_\lambda(A, B) = 0,$$

$$d_\lambda(A, C) = 1,$$

$$d_\lambda(B, C) = 1.$$

Ostatni warunek jest nadmiarowy, wynika on z własności (82).

Z pierwszych dwu warunków możemy ułożyć równania na współczynniki λ_i:

$$d_\lambda(A, B) = 0 = \lambda_3 + \lambda_7 \Rightarrow \lambda_3 = \lambda_7 = 1$$

$$d_\lambda(A, C) = 1 = \lambda_1 \Rightarrow \lambda_1 = 1.$$

Pozostałe współczynnki mają dowolne wartości. Widzimy, że własnością, która wyróżnia elementy klasy, do której należą A i B od elementów klasy, do której należy C, jest wartość pierwszego elementu ciągu, zaś różnice na pozycjach 3 i 7 nie mają znaczenia. Weźmy teraz czwarty wzorzec $D = (0, 1, 1, 1, 0, 1, 1)$ i policzmy odległości od pozostałych elementów:

$$d_\lambda(A, D) = 0,$$

$$d_\lambda(B, D) = 0,$$

$$d_\lambda(C, D) = 1.$$

Wzorzec ten został więc zaliczony do tej samej klasy, do której należą wzorce A i B. Jest to przykład generalizacji. Inną własność takiego klasyfikatora można opisać w następujący sposób. Niech wzorzec E ma postać $E = (0, 0, 0, 1, 1, 0, 1)$. Nie należy on do tej samej klasy co C, gdyż różni się od elementu C wartością na pozycji jeden. Do klasy zawierającej A i B może należeć lub nie, gdyż jak wspomniano wyżej wartości λ_i na pozycjach $2, 5, 6$ są nieokreślone. Jeśli zechcemy go włączyć do klasy zawierającej A i B, musimy wyzerować współczynniki $\lambda_2, \lambda_5, \lambda_6$, gdyż wzorzec E różni się od A i B wartościami tych składowych. Wprowadzenie takiej poprawki do funkcji odległości spowoduje, że do $class(A)$

będą zaliczane wszystkie wzorce (6) różniące się od A pozycjami 2, 5, 6 (dla których $d_\lambda(A,X) = \sum_{i\in 2,5,6} \lambda_i$).

Inny przykład zastosowania klasyfikatora opartego na funkcji odległości, także pochodzący z [DD], to klasyfikator dokonujący podziału wzorców ze względu na parzystość tj. parzystą bądź nieparzystą ilość składowych o wartości 1. Klasyfikator określa dla dwu wzorców, czy mają one tę samą (odległość równa 0), czy różną (odległość równa 1) parzystość.

Jest on oparty na funkcji odległości d_c postaci:

$$d_c(a,b) = |\sum_{i=0}^{n} \gamma_i \delta(((\sum_{k=1}^{n} a_k) - i), 0) - \sum_{i=0}^{n} \gamma_i \delta(((\sum_{j=1}^{n} b_k) - i), 0)|\,, \qquad (84)$$

gdzie

$$\gamma_i \in R^+$$

oraz

$$\delta(x,o) = \begin{cases} 1, & \text{gdy } x = 0; \\ 0, & \text{gdy } x \neq 0. \end{cases}$$

Aby funkcja dawała poprawne wartości, należy znaleźć wartości parametrów $\lambda_i, i = 0 \ldots n$. Parametr λ_i mnożony przez niezerową wartość stwierdza istnienie i ustawionych bitów we wzorcu. Do klasyfikacji ze względu na parzystość wystarczy więc parametrom o indeksach parzystych nadać te same wartości (np. 1), a wszystkim parametrom nieparzystym inne (np. 0).

W ten bardzo prosty sposób można dokonać operacji, z którą ciężko dają sobie rade sieci ze wsteczną propagacją błędu.

Stosuje się również funkcje odległości o wartościach ciągłych [DD], [Alb].

Funkcję odległości dwu wzorców b^1 i b^2 możemy także zapisać w ogólnej

postaci [Alb]:

$$d_s(b^1, b^2) = \sum_{i=1}^{n} \lambda_i |b_i^1 - b_i^2| (seq(b^1) + seq(b^2)) \,, \tag{85}$$

gdzie

$$seq(b^1) = \sum_{j+1}^{n} \beta_j \cdot \sum_{k=1}^{n-j+1} \alpha_k \cdot \prod_{l=0}^{j-1} b_{k+l}^i + \sum_{j+1}^{n} \gamma_j \cdot \sum_{k=1}^{n-j+1} \alpha_k \cdot \prod_{l=0}^{j-1} \delta(b_{k+l}^i, 0) \tag{86}$$

oraz $i \in \{0, 1\}$, $\lambda_i, \alpha_k, \beta_j, \gamma_j \geq 0$. Parametry α_k wskazują, gdzie zaczynają się ciągi jedynek i zer, β_j określają ilość sąsiadujących ze sobą jedynek, zaś γ_j zer.

Okazuje się [Alb], że stosując tak zdefiniowaną funkcję odległości można osiągnąć dowolne uporządkowanie odległości przez dobór odpowiednich wartości parametrów(autor [Alb] zakłada, że rozkład prawdopobieństwa pojawiania się wzorców jest jednostajny). Ponieważ funkcja odległości d_s dla $\lambda_i = 1$ oraz $\lambda_i, \alpha_k, \beta_j, \gamma_j = 0$ przechodzi w miarę Hamminga d_H [Was], to podczas nauki tych parametrów najlepiej zacząć od d_H, następnie zmieniać λ_i, a dopiero na końcu wprowadzać niezerowe (ale małe) pozostałe parametry. Powoduje to powstawanie na początku dużych, gładko wyróżnionych klas.

Metody takie znalazły zastosowanie do porównywania sekwencji molekuł [Alb] oraz do klasyfikacji fonemów [DD], [Alb].

8 Dodatek. Estymatory Bayesowskie

Zacznijmy od przypomnienia twierdzenia Bayesa o prawdopobieństwie warunkowym. Jeśli zdarzenie B może zajść tylko w wyniku zajścia któregoś z rozłącznych zdarzeń A_i, to istnieje następująca zależność:

$$P(A_i|B) = \frac{P(B|A_i)\, P(A_i)}{\sum\limits_i P(B|A_i)\, P(A_i)} \,, \tag{87}$$

gdzie

$P(A_i)$, to prawdopodobieństwo wystąpienia zdarzenia A_i,

$P(B|A_i)$, to warunkowe prawdopodobieństwo zajścia B, jeśli wystąpiło A_i,

$P(A_i|B)$, to prawdopodobieństwo tego, że zdarzenie A_i było przyczyną zajścia zdarzenia B.

Wzór ten znalazł zastosowanie przy obliczaniu prawdopodobieństw *a posteriori* h_i i $h_{j|i}$ w węzłach drzewa decyzyjnego sieci HME.

Można zrobić z niego jeszcze inny użytek [EDJ]. Zamiast interpretować A i B jako zdarzenia, stwierdźmy, że są to hipotezy. Wówczas prawdopobieństwo $P(A_i)$ trzeba traktować jako poziom ufności dla hipotezy A_i; jest to wiedza *a priori*. $P(A_i|B)$ opisuje wtedy wiedzę *a posteriori*, uzyskaną w wyniku zajścia zdarzenia B (np. wykonania doświadczenia). $P(B|A_i)$ to prawdopodobieństwo wystąpienia B, jeśli spełniona jest hipoteza A_i.

Nie zawsze znamy całkowite prawdopodobieństwo (dla hipotez wykluczających się) $P(B) = \sum_i P(B|A_i)\, P(A_i)$ zajścia zdarzenia B przy dowolnej hipotezie. Wzór (87) przechodzi wówczas w następującą relacje:

$$P(A_i|B) \propto P(B|A_i)\, P(A_i)\,. \tag{88}$$

Pozwala ona porównywać prawdopobieństwa *a posteriori* różnych hipotez.

W przypadku ciągłych zmiennych losowych zamiast prawdopodobieństw mamy ich rozkłady, opisane gęstościami. Załóżmy, że N wymiarowa zmienna losowa $\mathbf{X}$ ma gęstość rozkładu $p(\mathbf{X}|\theta)$ zależną od parametru θ, mogącego przyjmować różne wartości. Estymacja tego parametru polega na znalezieniu jego wartości na podstawie znanych obserwacji $\mathbf{X}$. W klasycznej metodzie estymacji zakłada się, że parametr θ ma jakąś, nieznaną , ale ustaloną wartość, której poszukujemy. Od-

mienne jest podejście bayesowskie, które stwierdza, że θ jest także zmienną losową, której rozkład $p(\theta)$ znamy na podstawie dokonanych uprzednio doświadczeń lub który wynika z jakiś naszych przekonań, czy założeń. Jest to nasza, subiektywna, wiedza *a priori* lub, w dwu ostatnich przypadkach, stwierdzenie naszej niewiedzy. Wzór Bayesa ma teraz postać:

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)\, p(\theta)}{\int p(\mathbf{X}|\theta)\, p(\theta)\, d\theta}\,. \tag{89}$$

Jeśli nie znamy rozkładu brzegowego zmiennej $\mathbf{X}$, występującego w mianowniku, otrzymamy tylko następującą proporcjonalność:

$$p(\theta|\mathbf{X}) \propto p(\mathbf{X}|\theta)\, p(\theta)\,, \tag{90}$$

wykorzystaną np. podczas wyprowadzania funkcji radialnych w [PG]. Mając do dyspozycji znany zbiór dokonanych obserwacji $\mathbf{X}^0$, możeny porównać $p(\theta)$ i $p(\theta|\mathbf{X}^0)$, sprawdzając jak nasza wiedza zmienia się pod wpływem doświadczenia [EDJ] i modyfikować naszą wiedzę o parametrze θ. Rozkład $p(\theta|\mathbf{X})$ można wykorzystać, poprzez maksymalizacje, do znalezienia wartości θ w rozpatrywanym przypadku. W pracy [PG] maksymalizacja tego prawdopodobieństwa prowadzi do otrzymania formuły wariacyjnej na funkcje radialne. Hipotezą, czy wiedzą aprioryczną, jest tam założenie o gładkości szukanej funkcji f, wyrażone przez funkcjonał gładkości (a wynikające raczej z niewiedzy o funkcji f).

Podejście bayesowskie ma również zastosowanie w teorii podejmowania decyzji [Z], [EDJ].

9 Opis rysunków

Rys.(13) Rysunek przedstawia sieć typu RN o n wejściach i jednym wyjściu. Zawiera ona m funkcji radialnych (tutaj gaussowskich opisanych średnią μ i wariancją σ) do przechowywania wejściowych wektorów treningowych. Wielkości h_i to wartości poszczególnych funkcji na wektorze wejściowym $\overline{\mathbf{x}}$, zaś w_j to waga danej funkcji. Na wyjściu sieci dostajemy ważoną sumę wartości funkcji. Rysunek pochodzi z [Was].

Rys.(14) Widzimy tu sieć typu RBF o n wejściach i m wyjściach, czyli przybliżającą funkcję $f : R^n \longrightarrow R^m$. Rodzaj tej sieci zależy od tego, jak danym treningowym przyporządkowywane są funkcje radialne. Dodatkowe węzły dokonują normalizacji wartości wyjściowych, dzieląc wyniki otrzymane z jednej funkcji przez sumę wyników wszystkich funkcji w warstwie ukrytej. Rysunek pochodzi z [Was].

Rys.(15) Rysunek przedstawia sieć typu *Adaptive Mixture of Local Neural Networks*. Składa się ona z t eksperckich sieci lokalnych i z sieci bramkującej g. $W_i^{(o)}$ to wartości obliczane przez poszczególne sieci lokalne ($i = 1, \ldots, t$), $W_g^{(o)}$ to wynik sieci bramkującej. Selektor stochastyczny dokonuje wyboru wyniku tej sieci eksperckiej, dla której sieć bramkująca określi największe prawdopodobieństwo uzyskania prawidłowego wyniku. Może on być zastąpiony układem obliczającym kombinacje liniową poszczególnych wyników. Rysunek pochodzi z [Kv].

Rys.(16) Narysowana jest tu dwupoziomowa sieć typu HME. Zbudowana jest z sieci eksperckich (na liściach drzewa) i dwu poziomów sieci bramkujących.

Danymi wejściowymi są wektory $\mathbf{x}$, wynikiem wektor $\mathbf{y}$. Rysunek pochodzi z [JJ].

Bibliografia

[All] John Allison, *Multiquadratic radial basis functions for representing multidimensional high energy physics data*, Computer Physics Communications 77 (1993) 377–395

[PG] Federico Girosi, Michael Jones, Tomaso Poggio, *Priors, Stabilizers and Basic Functions: from regularizations to radial, tensor and additive splines*,(MIT A. I. Memo No. 1430 and C. B. C. L Paper No. 75, March, 1994)

[LN] J. A. Leonard, M. A. Kramer, L. H. Ungar, *Using Radial Basis Functions to Approximate a Function and Its Error Bounds*, IEEE Transactions on Neural Networks. 3 (1992) 624–627

[JJ] Michael I. Jordan, Robert A. Jacobs, *Hierarchical Mixtures of Experts and the EM Algorithm*,(MIT A. I. Memo No. 1440 and C. B. C. L Memo No. 83, August 6, 1993)

[Kv] Vladimír Kvasnička, *Adaptive Mixture of Local Neural Networks*, Neural Network Work 2 (1993), 161–174

[Ama] Shun-ichi Amari, *Information Geometry of the EM and em Algorithms for Neural Networks*, (Departament of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, Bunkyo − ku, Tokyo 133, Japan)

[D1] W. Duch, G. H. F. Diercksen, *Feature Space Mapping as a Universal Adaptive System* , Computer Physics Communications 87 (1995) 341–371

[D2] W. Duch, G. H. F. Diercksen, *Neural Networks as Tool to Solve Problems in Physics and Chemistry*, Computer Physics Communications 82 (1994) 91–103

[Alb] Gabriele Scheler, *The Use of an Adaptive Measure in Generalizing Pattern Learning*, Artificial Neural Networks 2, (I. Aleksander and J. Taylor, (Editors)), Elsevier Science Publishers 1992

[Z] E. L. Lehmann, *Teoria estymacji punktowej*, Wydawnictwo Naukowe PWN, Warszawa 1991

[Was] Philip D. Wasserman, *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, New York (1993)

[Mat] J. Królikowski, C. Steckiewicz, Matematyka Wzory Definicje Tablice, Wyd. WKŁ

[TW] Bernard F. Schutz, *Wstęp do ogólnej teorii względności*, Wydawnictwo Naukowe PWN, Warszawa 1995

[EDJ] W. T. Eadie, D. Drijard, F. E. James, M. Roos, B. Sadoulet, *Metody statystyczne w fizyce doświadzalnej*, Państwowe Wydawnictwo Naukowe Warszawa 1989

[BV] Léon Bottou, Vladimir Vapnik *Local Learning Algorithms* Neural Comput. 4 (1992) 888–901

[SK] Steven G. Krantz, *Teoria funkcji wielu zmiennych zespolonych*, PWN, Warszawa 1991

[DD] Gabriele Scheler, *Pattern Classification with Adaptive Distance Measures*, FKI–189–94, Technische Universitat Munchen