plik wk3.tex, 27 października 2008, popr. 6.XI.

3 Błąd sieci i algorytmy jego minimizacji

3.1 Sposoby określania błędu – Ogólne omówienie

Funkcja błędu $E = E(\mathbf{w})$ jest w ogólnym przypadku nieliniową funkcją wektora wag $\mathbf{w} = [w_1, w_2, \ldots, w_S]$, gdzie S jest sumaryczną liczbą wag łącznie z wartościami progowymi (biasami). Funkcja E przedstawia NAJCZĘŚCIEJ

- w zagadnieniach regresyjnych: sumę kwadratów różnic między wartościami docelowymi (t^n) a wartościami $y^n(\mathbf{x}^n|\mathbf{w})$ wyprodukowanymi przez sieć;
- w zagadnieniach klasyfikacyjnych: ujemny logarytm z prawdopodobieństw á posteriori lub z wiarygodności $P(\mathbf{t}^n|\mathbf{x}^n)$ określającej otrzymanie wartości docelowych t^n pod warunkiem zaobserwowania wektora zmiennych objaśniających $\mathbf{x}^n$.

Zagadnienia regresyjne

W przypadku zagadnień regresyjnych (tj. aproksymacji lub predykcji funkcji wielu zmiennych) błąd jest wyznaczany najczęściej według zasady najmniejszych kwadratów, tj. jako {sse}

$$E = (1/2)\sum_{n=1}^{N}\sum_{k=1}^{c}(t_k^n - y_k^n)^2, \tag{3.1}$$

gdzie t_k^n jest podanym wzorcem (wartością docelową), a $y^n = y^n(\mathbf{w}, \mathbf{x}^n)$ jest wynikiem obliczonym przez sieć (założyliśmy c wyjść). Chcemy, aby kryterium E było możliwie małe.

Wartości docelowe $\mathbf{T}_{N\times c} = (t_k^n),\ k = 1, \ldots, c,\ n = 1, \ldots, N$ mogą na ogół przyjmować wartości rzeczywiste z przedziału $(-\infty, +\infty)$ lub $[0, +\infty)$. W takich przypadkach używa się liniowej funkcji aktywacji. Otrzymujemy wtedy (przy liniowej—tożsamościowej funkcji aktywacji) {modelR}:

$$\mathbf{Y} = \ddot{\mathbf{X}}\mathbf{W}, \tag{3.2}$$

gdzie $\ddot{\mathbf{X}}$ oznacza tablicę danych powiększoną o dodatkową kolumnę jedynek:

$$\ddot{\mathbf{X}} = [\mathbf{X},\ \mathbf{1}_N].$$

Wzór (3.2) jest równaniem nadokreślonym na wagi ($\mathbf{W}$) sieci. Mamy tu układ równań liniowych, z którego – stosując metodę najmniejszych kwadratów – możemy wyznaczyć wagi bezpośrednio w jednym działaniu, stosując operację matlabowską `left divide` (funkcja: mldivide) oznaczana symbolem '/'. Wagi z wzoru 3.2 otrzymamy wtedy prostą instrukcją $\mathbf{W} = \ddot{\mathbf{X}}/\mathbf{Y}$.

Alternatywą jest rozwiązanie układu równań normalnych.

Błąd sieci definiuje się wtedy jako 1/2 kwadratu normy Frobeniusa :

$$E = \frac{1}{2}sum(diag((\mathbf{T} - \mathbf{Y})^T(\mathbf{T} - \mathbf{Y}))).$$

Zagadnienia klasyfikacyjne

W przypadku zagadnień klasyfikacyjnych funkcja błędu jest minimalizowana na podstawie kryterium nazywanego *entropią krzyżową* (*cross-entropy*), ktore zostanie omówione niżej. Kryterium to jest równoważne ujemnemu logarytmowi wiarygodności zaobserwowania wartości danych w tablicy **T**.

Dla $c > 1$ klas błąd E wynosi:

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n \log y_k^n.$$

Dla $c = 1$ (dwie klasy) wzór powyższy redukuje się do następującego:

$$E = -\sum_{n=1}^{N}\{t^n \ln y^n + (1-t^n)\ln(1-y^n)\}.$$

Chcemy, aby wartość tego kryterium była możliwie mała.

Praktycy oceniają jakość (przydatność lub nieprzydatność) reguły klasyfikacyjnej na podstawie frakcji poprawnie lub też niepoprawnie zaklasyfikowanych osobników.

3.2 Definicje błędu wynikające z wiarygodności zdarzeń

3.2.1 Określenie wiarygodności próbki (sample likelihood)

Niech $\mathbf{x}$, $\mathbf{y}$, $\mathbf{t}$ oznaczają wektory zmiennych objaśniających, wartości prognozowanych przez sieć oraz wartości docelowych.

Niech $p(\mathbf{t}|\mathbf{x})$ oznacza prawdopodobieństwo (funkcję gęstości prawdopodobieństwa) zaobserwowania wektora $\mathbf{t}$ ($\mathbf{t} \in R^c$) pod warunkiem wystąpienia wektora $\mathbf{x}$. **Symbol $p(.)$ oznacza tu prawdopodobieństwo dla zmiennych dyskretnych lub funkcję gęstości p-stwa dla zmiennych ciągłych**.

Jeżeli założymy, że wyniki $t_1, t_2, \ldots, t_c$ są niezależne [1] to prawdopodobieństwo $p(\mathbf{t}|\mathbf{x})$ możemy przedstawić jako iloczyn prawdopodobieństw:

$$p(\mathbf{t}|\mathbf{x}) = \prod_{k=1}^{c} p(t_k|\mathbf{x}).$$

Funkcja $p(\mathbf{t}|\mathbf{x})$ została tu zdefiniowana dla jednego wektora danych, bez precyzowania, który to jest wektor. W rzeczywistości mamy do czynienia z próbką uczącą, która składa się z N takich wektorów.

W sieciach neuronowych indywidualne wektory próbki ('wzorce') są na ogół numerowane górnym wskaźnikiem ^{n}, $n = 1, \ldots, N$, przy czym N oznacza całkowitą liczebnośc próbki (liczbe wektorów danych w tej próbce, inaczej: liczbę wierszy tablicy danych $\mathbf{X}$, jeśli wszystkie dane zostały zebrane w jedną tablicę $\mathbf{X}$). Jeszcze inaczej, naszą próbkę możemy przedstawić jako zbiór par $(\mathbf{x}^n, \mathbf{t}^n)$, $(n = 1, \ldots, N)$. Jeżeli założymy, że elementy próbki są *i.i.d*, tzn. *independent*, *identically distributed*, to możemy określić łączne prawdopodobieństwo zaobserwowania ciągu par $(\mathbf{x^n}, \mathbf{t^n})$, $n = 1, \ldots, N$, jako iloczyn { product}

$$p\left(\mathbf{t}^1, \ldots, \mathbf{t}^N | \mathbf{x}^1 \ldots \mathbf{x}^N\right) = \prod_{n=1}^{N} p\left(\mathbf{t}^n | \mathbf{x}^n\right). \tag{3.3}$$

[1] czy i kiedy możemy tak zakładać, oto jest pytanie

Prawdopodobieństwo warunkowe określone wyżej zależy w sposób ukryty od pewnych parametrów, które oznaczymy ogólnie symbolem $\mathbf{w}$. W przypadku obliczeń za pomocą sieci typu GLM parametry te nazwiemy *wagami* sieci [2]. Napiszmy to wyraźnie: {L }

$$\mathcal{L} = p\left(\mathbf{t}^1, \ldots, \mathbf{t}^N | \, \mathbf{x}^1 \ldots \mathbf{x}^N; \mathbf{w}\right) = \prod_{n=1}^{N} p\left(\mathbf{t^n} | \, \mathbf{x^n}; \mathbf{w}\right). \tag{3.4}$$

Funkcja $\mathcal{L}$ określona powyższym wzorem jest nazywana **funkcją wiarygodności** lub po prostu **wiarygodnością** próbki. Przy znanych wartościach $\mathbf{t}^n$ oraz $\mathbf{x}^n$ możemy patrzeć na $\mathcal{L}$ jako na funkcję wag: $\mathcal{L} = \mathcal{L}(\mathbf{w})$. Będziemy chcieli wybrać wagi $\mathbf{w}$ ten sposób, aby uczynić wiarygodność $\mathcal{L}$ możliwie dużą.

Maksymizacja funkcji $\mathcal{L}(\mathbf{w})$ jest równoważna maksymizacji *logarytmu* $\mathcal{L}(\mathbf{w})$ lub **minimizacji** *logarytmu* $\mathcal{L}(\mathbf{w})$ wziętego ze znakiem przeciwnym. Oznaczmy: {Errfun}

$$E = -\log \mathcal{L} = -\log p\left(\mathbf{t}^1, \ldots, \mathbf{t}^N | \, \mathbf{x}^1 \ldots \mathbf{x}^N; \mathbf{w}\right) = \sum_{n=1}^{N} p\left(\mathbf{t}^n | \, \mathbf{x}^n; \mathbf{w}\right). \tag{3.5}$$

W zagadnieniach klasyfikacyjnych funkcja E określona powyżej jest używana jako **funkcją błędu**.

Dla sieci o architekturze GLM błąd E jest wyznaczany za pomocą funkcji glmerr. Jak wynika ze wzoru [3.5], błąd E zależy od przyjętego modelu $p\left(\mathbf{t}^n | \, \mathbf{x}^n\right)$.

W dalszym ciągu przedstawimy modele $p\left(\mathbf{t^n} | \, \mathbf{x^n}\right)$ dla prostych zagadnień regresyjnych i klasyfikacyjnych.

3.2.2 Obliczanie błędu E w zagadnieniach regresyjnych

Obserwujemy zmienną losową $\underline{\mathrm{T}}=(\underline{t}_1, \ldots, \underline{t}_c)$. O zmiennej $\underline{\mathrm{T}}$ wiadomo, że zależy od pewnych zmiennych objaśniajacych $\mathbf{x} = (x_1, \ldots, x_d)$. Niech $\mathbf{t} = (t_1, \ldots, t_c)^T$ oznacza obserwowaną wartość zmiennej losowej $\underline{\mathrm{T}}$.

Zakładamy następujący model {modellin}

$$\mathbf{t} = \mathbf{y}(\mathbf{x}; \mathbf{w}) + \mathbf{e}. \tag{3.6}$$

W modelu (3.6) wartości $\mathbf{t}$ uważamy za realizację zmiennej losowej $\underline{\mathrm{T}}$. Zmienna ta składa się z dwóch składników: składnika stałego $\mathbf{y}(\mathbf{x}; \mathbf{w})$ i składnika losowego $\mathbf{e}$.

Składnik stały $\mathbf{y}$ jest znaną funkcją zmiennych objaśniających i pewnych parametrów, oznaczanych tu umownie symbolem $\mathbf{w}$ (np. mogło by być: $y = x_1 w_1 + x_2 w_2 + w_0$). Funkcja $\mathbf{y}(\mathbf{x}; \mathbf{w})$ może być zaimplementowana za pomocą sieci neuronowej o zadanej architekturze i zadanej funkcji aktywacji (wektor $\mathbf{w}$ zawiera wszystkie wagi i biasy spakowane w jeden wektor, GLM używa w tym celu funkcji glmpack). Jest to część deterministyczna modelu, mówiąca jaka jest wartość należna zmiennej T przy ustalonych wartościach $\mathbf{x}$ i ustalonych wagach.

Można spodziewać się (i tak jest faktycznie), że wartość $\mathbf{y}(\mathbf{x}; \mathbf{w})$ dostarczona przez sieć nie daje dokładnej wartości $\mathbf{t}$, tylko pewne jej przybliżenie, różniące sie od dokładnej wartości o wielkość $\mathbf{e}$. Będziemy chcieli wybrać wagi w ten sposób, aby zminimizować różnicę (błąd) $\mathbf{e}$.

Przyjmijmy, że wielkości $\mathbf{e}$ są losowe.

[2] wektor $\mathbf{w}$ zawiera wszystkie wagi i biasy sieci w postaci spakowanej np. funkcją glmpack z Netlaba

Mamy: $\mathbf{e} = [e_1, \ldots, e_c]^T$. Załóżmy, że składowe e_k są niezależnymi zmiennymi losowymi i podlegają rozkładowi normalnemu z wartością oczekiwaną równą 0 i wariancją σ^2 (rozkład ten jest również nazywany rozkładem Gaussa i oznaczany) $\mathcal{N}(\imath, \sigma^{\in})$: {ErrGauss}

$$p\,(e_k) = \frac{1}{(2\pi\sigma^2)^{1/2}}\; exp\left\{ -\frac{e_k^2}{2\sigma^2} \right\}. \tag{3.7}$$

Wtedy p-stwo $p(t_k|\mathbf{x})$ (jest to faktycznie funkcja gęstości) możemy zapisać jako {GaussTk}

$$p\,(t_k|\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{1/2}}\; exp\left\{ -\frac{[t_k - y_k(\mathbf{x}\,;\mathbf{w})]^2}{2\sigma^2} \right\}. \tag{3.8}$$

Symbol $\mathbf{w}$ we wzorze powyżej oznacza wektor wag i biasów w postaci spakowanej.

Dla c cech wyjściowych mamy $\mathbf{t}{=}(t_1, \ldots, t_c)^T$, a p-stwo $p(\mathbf{t}\,|\mathbf{x})$ jest wtedy iloczynem p-stw $p(t_k|\mathbf{x})$ dla poszczególnych składowych. Po podstawieniu wskaźników $n = 1, \ldots, N$ oznaczających kolejne wektory próbki uczącej, zlogarytmowaniu i pominięciu składników, które nie zależą od $\mathbf{w}$, otrzymujemy kolejno wiarygodność $\mathcal{L}$, a z niej funkcję błędu E

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}([t_k^n - y_k(\mathbf{x^n};\mathbf{w})]^2. \tag{3.9}$$

Otrzymaliśmy w ten sposób kryterium najmniejszej sumy kwadratów odchyleń. To samo kryterium jest używane przy liniowej aproksymacji funkcji wielu zmiennych.

Przy kryterium (3.9), optymalny wektor $\mathbf{w}$ (tj. taki, który minimizuje błąd E) otrzymuje się jako rozwiązanie układu równań liniowych, nazywanych *układem równań normalnych*.

Czasami układ równań normalnych jest źle uwarunkowany, co powoduje rozbieżność wag (niektóre składowe powiększają się w nieskończoność). Aby temu zapobiec, wprowadza się tzw. składnik *regularyzacji* (*weight decay penalty term*, w statystyce 'ridge regression'). Minimuzuje się wtedy zmodyfikowany błąd E_r

$$E_r = E + \alpha\sum_{i,j} w_{ij}^2 \quad (\alpha > 0).$$

3.2.3 Obliczanie błędu E w zagadnieniach klasyfikacyjnych

• Rozważamy zagadnienie klasyfikacji do jednej z c klas $\mathcal{C}_1, \ldots, \mathcal{C}_c$. Zadeklarowana zmienna $\mathbf{t}$ ('target') jest w tym przypadku dyskretna, $t_k \geq 0$, $\sum_k t_k = 1$. 0bliczana przez sieć wartość $y_k(\mathbf{x^n};\mathbf{w})$ oznacza prawdopodobieństwo, że zaobserwowany wektor $\mathbf{x^n}$ należy do klasy nr. k (por. np. [1, 2]):

$$P(\mathcal{C}_k|\,\mathbf{x^n}) = y_k(\mathbf{x^n};\mathbf{w}).$$

Mamy: $y_k \in [0,1]$, $\sum_{k=1}^{c} y_k = 1$.

Jeżeli każdą ze składowych y_k obliczamy niezależnie od pozostałych, to p-stwo $p\,(\mathbf{t^n}|\,\mathbf{x^n})$ możemy zapisać (ze wzoru multinomialnego) jako

$$p\,(\mathbf{t^n}|\,\mathbf{x^n}) = \prod_{k=1}^{c}(y_k^n)^{t_k^n}, \quad \text{gdzie } y_k^n = y_k(\mathbf{x^n};\mathbf{w}).$$

Stąd wynika ogólna formuła na funkcję błędu {ErrMulti}

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n\, \log\, y_k^n. \tag{3.10}$$

• W przypadku, gdy mamy do czynienia **tylko z dwoma klasami**, możemy uprościć zagadnienie do $c = 1$ przez odpowiednie zakodowanie przynależności do klas: Zmienna $\mathbf{t}$ jest teraz jednowymiarowa i może przyjmować tylko wartości 1 lub 0. Wobec tego mamy $\mathbf{t} = t$, a p-stwo $p(\mathbf{t} \mid \mathbf{x}) = p(t \mid \mathbf{x})$ możemy zapisać w postaci

$$p(t \mid \mathbf{x}) = y^t(1-y)^{1-t},$$

gdzie y oznacza ogólnie obliczone przez sieć p-stwo wystąpienia atrybutu oznaczanego symbolem '1'.

Powyższy zapis obejmuje zarówno przypadek wystąpienia '1' jak i '0'.

Jeżeli mamy do czynienia z próbką niezależnych par $(\mathbf{x}^1, t^1), \ldots, (\mathbf{x}^N, t^N)$, to wiarygodność dla n-tej próbki $(n = 1, \ldots, N)$ zapisujemy jako

$$p(t^n \mid \mathbf{x}^n) = y^{t_n}(1-y)^{1-t^n},\ (n = 1, \ldots, N),$$

gdzie $y = y^n = y(\mathbf{x}^n; \mathbf{w})$.

Wiarygodność otrzymania całego ciągu $\{\mathbf{x}^n, t^n\},\ n = 1, .., N$ jest iloczynem wiarygodności wypisanych powyżej. Logarytm wiarygodności, wzięty ze znakiem przeciwnym, wynosi wtedy

$$E = -\sum_{n=1}^{N}\{t^n \ln y^n + (1-t^n)\ln(1-y^n)\} = \sum_{n=1}^{N} E^n. \tag{3.11}$$

Kryterium to nosi nazwę *entropii krzyżowej* (cross-entropy) i jest sumą entropii cząstkowych E^n wnoszonych przez każdy wzorzec $(\mathbf{x}^n, t^n)$.

Wzór [3.11] jest szczególnym przypadkiem wzoru [3.10].

3.2.4 Klasy algorytmów minimalizacyjnych

Generalnie możemy tu wyróżnić trzy klasy:

1. Algorytmy klasyczne. Są to algorytmy związane z zagadnieniami minimalizacyjnymi w ogóle. Nie wykorzystują specyfiki sieci neuronowych i są stosowane w wielu innych zagadnieniach optymalizacyjnych. Najczęściej są to algorytmy gradientowe, chociaż są znane również algorytmy bezgradientowe.

   W szczególności należy tu również metoda *Iterative Reweighted Least Squares* wywodząca się ze statystycznej metody GLM (Generalized Linear Model).

2. Algorytmy związane specyficznie z sieciami neuronowymi. Zalicza się tu przede wszystkim algorytm *propagacji wstecznej* czyli *backpropagation*. Podstawą tego algorytmu jest metoda gradientowa; jednak obliczenie poszczególnych gradientów odbywa się w specjalny sposób z uwzględnieniem rozłożenia neuronów w poszczególnych warstwach. Algorytm backpropagation jest stosowany przede wszystkim wtedy gdy liczba warstw ukrytych jest duża (większa niż 2), chociaż może być stosowany również przy 2 warstwach. Metoda ta nie będzie omawiana w tym opracowaniu.

   Należy tu również metoda trenowania perceptronu w przypadku, gdy funkcja aktywacji jest funkcją *hardlimit*. Jest to metoda o znaczeniu historycznym, używana pierwotnie przez Cullocha i Pittsa, również później przez Rosenblatta.

3. Algorytmy z elementami stochastycznymi. Należą tu w pierwszym rzędzie: algorytm symulowanego wyżarzania, algorytmy genetyczne i algorytmy ewolucyjne.

   Metody te nie będą omawiane w tym opracowaniu.

W dalszym ciągu omówimy jedynie kilka algorytmów klasycznych wykorzystujących metody gradientowe, oraz metodę trenowania perceptronu gdy funkcją aktywacji jest funkcja hardlimit.

3.3 Algorytmy klasyczne do minimalizacji funkcji $E(w)$

Generalnie rzecz biorąc, są to algorytmy iteracyjne.

3.3.1 Zasada algorytmów iteracyjnych

Z jakiegoś początkowego przybliżenia wektora $\mathbf{w}^{(0)}$ konstruujemy w kolejnych krokach (iteracjach) ciąg kolejnych przybliżeń wektora $\mathbf{w}$ który ma dać minimum funkcji błędu $E(\mathbf{w})$. Przypuśćmy, że po k krokach otrzymaliśmy wartość $\mathbf{w}^{(k)}$. W następnym kroku, $(k+1)$-szym, konstruujemy następne przybliżenie, $(k+1)$-sze, według zasady:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta\mathbf{w}^{(k)}. \tag{3.12}$$

Cała sztuka algorytmu leży w tym, aby w skuteczny sposób znaleźć poprawkę $\Delta\mathbf{w}^{(k)}$. Poprawka ta polega na ogół na znalezieniu pewnego wektora $\mathbf{p}^{(k)}$ i dodaniu części tego wektora do posiadanego aktualnie wektora $\mathbf{w}^{(k)}$ (nie możemy dodać za dużo, bo moglibyśmy "przeskoczyć" właściwe minimum). Tak więc podstawiamy

- $$\Delta\mathbf{w}^{(k)} = \eta\mathbf{p}^{(k)}. \tag{3.13}$$

  Dalej poznamy różne metody znajdywania kierunku $\mathbf{p}$. W najprostszym przypadku może to być ujemny gradient funkcji E w punkcie $\mathbf{w}^{(k)}$, co oznacza że podstawiamy $\mathbf{p}^{(k)} = -\nabla E|_{w^{(k)}}$. Wtedy poprawka $\Delta\mathbf{w}^{(k)}$ przyjmie postać:

  $$\Delta\mathbf{w}^{(k)} = -\eta\nabla E\Big|_{\mathbf{w}^{(k)}}. \tag{3.14}$$

  Współczynnik η nosi nazwę współczynnika uczenia. Powinien on przyjmować wartości z przedziału $[0, 1]$. Naogół wprowadza się go jako funkcję malejącą numeru iteracji.

  Jeżeli wartości η są zbyt małe, to postęp w zbliżaniu się do minimum jest zbyt wolny; jeżeli η jest zbyt duże, to wartości $\mathbf{w}$ mogą zmieniać się zbyt chaotycznie.

- Aby przeciwdziałać zbyt chaotycznym zmianom, wprowadza się do wzoru na $\Delta\mathbf{w}^{(k)}$ dodatkowy człon, tzw. *momentum* lub *pęd*:

  $$\Delta\mathbf{w}^{(k)} = -\eta\underbrace{\nabla E\Big|_{\mathbf{w}^{(k)}}}_{gradient} + \rho\underbrace{(\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)})}_{momentum}. \tag{3.15}$$

  We wzorze powyższym $\rho > 0$ (por [1]). Często przyjmuje się $1 > \rho > 0$. Wtedy równanie 3.15 przyjmuje postać:

  $$\Delta\mathbf{w}^{(k)} = (1-\rho)(-\eta\underbrace{\nabla E\Big|_{\mathbf{w}^{(k)}}}_{gradient}) + \rho\underbrace{(\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)})}_{momentum}.$$

  Dodatkowy człon $momentum = (\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}) = \Delta\mathbf{w}^{(k-1)}$ zapobiega raptownym zmianom kolejno wyznaczonych wartości $\mathbf{w}^{(k)}$; można powiedzieć, że do pewnego stopnia zachowuje kierunek (tendencję) zmian.

  Określenie odpowiednich wartości η i ρ jest sprawą wyczucia i doświadczenia. Technika *bold driver technique* ([1], str 269), pozwala wyznaczyć jednocześnie współdziałające wartości η i ρ.

3.3.2 Aproksymacja funkcji błędu za pomocą rozwinięcia w szereg Taylora – i co z tego wynika

Rozwinięcie funkcji $E(\mathbf{w}+\mathbf{p})$ dookoła punktu $\mathbf{w}$:

$$E(\mathbf{w}+\mathbf{p}) = E(\mathbf{w}) + \mathbf{g}(\mathbf{w})^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T\mathbf{H}(\mathbf{w})\mathbf{p} + 0(h^3), \tag{3.16}$$

gdzie:

$$\mathbf{g}(\mathbf{w}) = \nabla E\Big|_{\mathbf{w}} = \left[\frac{\partial E}{\partial w_1}\Big|_{\mathbf{w}}, \ldots, \frac{\partial E}{\partial w_S}\Big|_{\mathbf{w}}\right]^T, \tag{3.17}$$

$$\mathbf{H} = \Big(\frac{\partial^2 E}{\partial w_i \partial w_j}\Big)\Big|_{\mathbf{w}}, \quad i,j = 1,\ldots,S. \tag{3.18}$$

Wyrażenie $\mathbf{g}(\mathbf{w})$ nazywa się gradientem funkcji (powierzchni) E w punkcie $\mathbf{w}$ i określa płaszczyznę styczną do tej powierzchni.

Macierz $\mathbf{H}$ nazywa się hesjanem i zawiera w sobie informacje o krzywiźnie powierzchni.

Różne algorytmy służące minimizacji w dużym stopniu korzystają z przedstawionego wyżej rozwinięcia (3.16); przy czym niektóre odmiany tych algorytmów biorą tylko człony liniowe, a inne zarówno człony liniowe jak i kwadratowe.

3.3.3 Algorytm największego spadku

W rozwinięciu (3.16) ograniczamy się tylko do członu liniowego rozwinięcia:

$$E(\mathbf{w}^{(k)} + \mathbf{p}^{(k)}) = E(\mathbf{w}^{(k)}) + \mathbf{g}(\mathbf{w}^{(k)})^T\mathbf{p}^{(k)}. \tag{3.19}$$

Chcemy, żeby

$$E(\mathbf{w}^{(k+1)}) < E(\mathbf{w}^{(k)}).$$

Będzie tak, gdy

$$\mathbf{g}(\mathbf{w}^{(k)})^T\mathbf{p}^{(k)} < 0\,.$$

Wynika stąd, że będzie tak (tj. nastąpi obniżenie błędu), gdy przyjmiemy:

$$\mathbf{p}^{(k)} = -\mathbf{g}(\mathbf{w}^{(k)})\,.$$

W ten sposób otrzymaliśmy *kierunek* poprawki. Samej poprawki $\Delta\mathbf{w}^{(k)}$ dokonujemy korzystając z wzoru 3.14 lub 3.15.

Wadą metody jest niewykorzystywanie informacji o krzywiźnie funkcji (powierzchni błędu). Ponadto, w okolicy minimum gradient bywa mały, a więc zbliżanie się do minimum staje się bardzo powolne.

Zaletą metody jest jej prostota, małe wymagania co do pamięci, stosunkowo mała złożoność obliczeniowa.

Ponadto – można uzyskać poprawę efektywności metody przez zastosowanie momentum.

Dalsze modyfikacje tej metody to: reguła *delta-bar-delta* (wyznacza m.in. w specjalny sposób współczynnik uczenia η) i metoda *quickprop* Fahlmana.

3.3.4 Metoda Newtona

Korzystamy tutaj z przybliżenia kwadratowego funkcji $E(\mathbf{w})$:

$$E(\mathbf{w}^{(k)} + \mathbf{p}^{(k)}) \approx E(\mathbf{w}^{(k)}) + \mathbf{g}(\mathbf{w}^{(k)})^T \mathbf{p}^{(k)} + \frac{1}{2}(\mathbf{p}^{(k)})^T \mathbf{H}(\mathbf{w}^{(k)})\mathbf{p}^{(k)}. \tag{3.20}$$

Szukamy takiego wektora $\mathbf{p}^{(k)}$, który by – dodany do wektora $\mathbf{w}^{(k)}$ – pozwolił nam osiągnąć minimum powierzchni błędu.

Jeżeli osiągnęliśmy minimum, to pochodna w tym punkcie powinna być równa zero:

$$\frac{\partial E(\mathbf{w}^{(k)} + \mathbf{p}^{(k)})}{\partial \mathbf{p}} = 0.$$

Różniczkując obustronnie równość (3.20) ze względu na wektor $\mathbf{p}^{(k)}$, i korzystając z tego, że pochodna lewej strony wyznaczana w minimum jest równa zeru, otrzymujemy:

$$0 = \mathbf{g}(\mathbf{w}^{(k)}) + \mathbf{H}(\mathbf{w}^{(k)})\mathbf{p}^{(k)}.$$

Wynika stąd, że jako poszukiwany wektor kierunkowy $\mathbf{p}^{(k)}$ doprowadzający nas do minimum należy przyjąć

$$\boxed{\mathbf{p}^{(k)} = -[\mathbf{H}(\mathbf{w}^{(k)})]^{-1}\mathbf{g}(\mathbf{w}^{(k)})\,.} \tag{3.21}$$

Aby zapewnić ten warunek, należy w każdym cyklu określić wartość gradientu $\mathbf{g}$ oraz Hesjanu $\mathbf{H}$ w punkcie znanego (ostatniego) rozwiązania $\mathbf{w}^{(k)}$.

Wtedy:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - [\mathbf{H}(\mathbf{w}^{(k)})]^{-1}\mathbf{g}(\mathbf{w}^{(k)})\,. \tag{3.22}$$

Wzór ten jest trudny do zrealizowania w praktyce, gdyż

- wymaga w każdym kroku iteracyjnym (k) obliczenia wartości hesjanu $\mathbf{H}$ w punkcie $\mathbf{w}^{(k)}$,
- wymaga odwrócenia tego hesjanu, do czego jest potrzebna dodatnia określoność tego hesjanu – i to w każdym kroku iteracji – czego sam algorytm nie jest nam w stanie zapewnić.

W tym stanie rzeczy w praktycznych implementacjach rezygnuje się z wyznaczania dokładnej wartości hesjanu, a zamiast tego korzysta się z jego wartości przybliżonej, która w dalszym ciągu będzie oznaczana jako $\hat{\mathbf{H}}(\mathbf{w}^{(k)})$.

Jedną z najpopularnieszych metod służących temu celowi jest metoda zmiennej metryki.

3.3.5 Metoda zmiennej metryki – quasi-Newton

W metodzie tej w każdym kroku iteracji modyfikuje się hesjan $\hat{\mathbf{H}}$ z kroku poprzedniego o pewną poprawkę.

Poprawka może być tak dobrana, aby aktualna wartość hesjanu $\hat{\mathbf{H}}(\mathbf{w}^{(k)})$ przybliżała krzywiznę funkcji celu E zgodnie z zależnością:

$$\hat{\mathbf{H}}(\mathbf{w}^{(k)})(\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}) = \mathbf{g}(\mathbf{w}^{(k)}) - \mathbf{g}(\mathbf{w}^{(k-1)}),$$

gdzie $\mathbf{g}(.)$ oznacza odpowiedni gradient.

Na podstawie powyższego założenia można otrzymać wzory określające hesjan w kroku k-tym w zależności od wartości hesjanu w kroku $(k-1)$-szym oraz od przyrostów gradientu i optymizowanej zmiennej $\mathbf{w}$. Przybliżenie hesjanu $\hat{\mathbf{H}}(\mathbf{w}^{(k)})$ powinno być tak skonstruowane, aby zapewnić istnienie odwrotności $\mathbf{V} = [\hat{\mathbf{H}}(\mathbf{w}^{(k)})]^{-1}$.

Ostrowski (str. 57) podaje dwa algorytmy posiadające tę własność. Są to:
- algorytm Broydena-Goldfarba-Fletchera-Shanno (BFGS) oraz
- algorytm Dawidona-Fletchera-Powella (DFP).

Metoda zmiennej metryki charakteryzuje się zbieżnością superliniową; jest więc znacznie lepsza od liniowo zbieżnej metody największego spadku.

Według Osowskiego (str 57) metoda ta jest obecnie uważana za jedną z najlepszych metod optymalizacji funkcji wielu zmiennych.

Jej wadą jest stosunkowo duża złożoność obliczeniowa wynikająca z konieczności wyznaczania S^2 elementów hesjanu, a także duże wymagania co do pamięci przy przechowywaniu macierzy hesjanu. Jej skuteczność sprawdzono na komputerach osobistych dla sieci nie przekraczających tysiąca parametrów.

3.3.6 Algorytm Levenberga-Marquardta

Jest inną, bardzo popularną odmianą newtonowskiej metody optymalizacji. Wartość dokładna hesjanu ze wzoru (3.21) zostaje zastąpiona wartością aproksymowaną $\hat{\mathbf{H}}(\mathbf{w}^{(k)})$ określaną na podstawie informacji zawartej w gradiencie, a także przy uwzględnieniu czynnika regularyzującego (Osowski, str. 58–59).

3.3.7 Metoda sprzężonych gradientów

Można by usiłować budować ciąg kolejnych kierunków $\mathbf{p}^{(k)}$ określających kolejne poprawki $\Delta\mathbf{w}^{(k)}$ (por. wzór (3.13)) jako ciąg wektorów $\mathbf{p}^{(k)}$ wzajemnie ortogonalnych. Zapewnia to przeszukiwanie przestrzeni optymalizowanych parametrów w niezależnych kierunkach. Jednak w takim przypadku nie wykorzystujemy informacji o krzywiźnie powierzchni błędu. Aby to zrobić, konstruujemy ciąg kolejnych przybliżeń, czyli ciąg wektorów $\mathbf{p}^{(k)}$ w ten sposób, żeby one były wzajemnie sprzężone względem hesjanu $\mathbf{H}$ lub jego przybliżenia.

Należy ty zaznaczyć, że takich kierunków można skonstruować conajwyżej tyle, ile jest parametrów (wag) do optymalizacji. W praktyce wykonuje się tylko część dopuszczalnej liczby iteracji (tzn. wyznaczania sprzężonych kierunków poszukiwań), poczym inicjuje się od nowa losowy kierunek przeszukiwań, i do niego konstruuje się dalsze, odpowiednio sprzężone kierunki.

Definicja. Wektory $\mathbf{p}_j$ i $\mathbf{p}_i$ są wzajemnie sprzężone względem macierzy $\mathbf{G}$, jeśli jest spełniony warunek:

$$\mathbf{p}_j^T \mathbf{G} \mathbf{p}_i = 0.$$

Osowski (str 59) pokazuje, że kolejny kierunek $\mathbf{p}^{(k)}$ można otrzymać jako kombinację liniową aktualnego gradientu $\mathbf{g}^{(k)}$ i kierunków $\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)}$ wyznaczonych w poprzednich iteracjach (oczywiście pod wrunkiem, że liczba iteracji nie przekroczyła wymiaru przestrzeni będącej dziedziną minimalizowanej funkcji błędu):

$$\mathbf{p}^{(k)} = -\mathbf{g}_k + \sum_{j=0}^{k-1} \beta_{kj} \mathbf{p}^{(j)},$$

gdzie $\mathbf{g}_k$ jest gradientem funkcji błędu wyznaczonym w punkcie $\mathbf{w}_k$.

Po uwzględnieniu warunków ortogonalności i warunków sprzężenia względem hesjanu **H** otrzymujemy

$$\mathbf{p}^{(k)} = -\mathbf{g}_k + \beta_{k-1}\mathbf{p}^{(k-1)},$$

gdzie współczynnik β_{k-1} może być wyznaczony np. według jednej z następujących reguł (Osowski, str. 60, Bishop, str 280-281):

$$\begin{aligned}
\beta_{k-1} &= \frac{\mathbf{g}_k^T(\mathbf{g}_k - \mathbf{g}_{k-1})}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} && \text{(Polak-Ribiere)} \\
\beta_{k-1} &= \frac{\mathbf{g}_k^T(\mathbf{g}_k - \mathbf{g}_{k-1})}{-(\mathbf{p}^{k-1})^T\mathbf{g}_{k-1}} \\
\beta_{k-1} &= \frac{\mathbf{g}_k^T(\mathbf{g}_k - \mathbf{g}_{k-1})}{(\mathbf{p}^{k-1})^T(\mathbf{g}_k - \mathbf{g}_{k-1})} && \text{(Hestenes-Stiefel)} \\
\beta_{k-1} &= \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} && \text{(Fletcher-Reeves)}
\end{aligned}$$

Tak więc – korzystając z zasady sprzężenia – możemy konstruować kierunki poszukiwań.

Następnym etapem jest znalezienie minimum funkcji na danym kierunku $\mathbf{p}^k$ – przez tzw. minimalizację kierunkową. Wykorzystujemy tu aproksymację kwadratową wynikającą z rozwinięcia Taylora (3.20) postaci:

$$E(\mathbf{w}^{(k)} + \alpha\mathbf{p}^{(k)}) \approx E(\mathbf{w}^{(k)}) + \mathbf{g_k}^T(\mathbf{w}^{(k)} + \alpha\mathbf{p}^{(k)}) + \frac{1}{2}(\mathbf{w}^{(k)} + \alpha\mathbf{p}^{(k)})^T\mathbf{H}(\mathbf{w}^{(k)} + \alpha\mathbf{p}^{(k)}).$$

Teraz szukamy takiej wartości α która dałaby minimum funkcji błędu na kierunku $(\mathbf{w}^{(k)} + \alpha\mathbf{p}^{(k)})$.

Obliczając pochodną cząstkową względem α i przyrównując ją do zera otrzymamy

$$\alpha_{min}^{(k)} = \frac{\mathbf{p}^{(k)}\mathbf{g}_k}{(\mathbf{p}^{(k)})^T\mathbf{H}\mathbf{p}^{(k)}}. \tag{3.23}$$

Jak można zauważyć, do wyznaczenia wielkości $\alpha_{min}^{(k)}$ wystarczy znajomość $\mathbf{H}\mathbf{p}^{(k)}$ – który można otrzymać pośrednio, bez oddzielnego wyznaczania całego hesjanu **H**.

Typowo, obydwa te kroki (wyznaczanie nowego, sprzężonego kierunku $\mathbf{p}^{(k)}$ i szukanie stałej α_k dającej minimum na tym kierunku) są stosowane naprzemiennie.

Własności metody. Według Osowskiego (str 60) metoda sprzężonych gradientów wykazuje zbieżność zbliżoną do liniowej i z tego powodu jest mniej skuteczna niż metoda zmiennej metryki, ale zdecydowanie szybsza niż metoda największego spadku. Stosuje się ją powszechnie jako jedyny skuteczny sposób algorytmu optymalizacji przy bardzo dużej liczbie zmiennych sięgających nawet kiludziesięciu tysięcy.

3.3.8 Metoda gradientów sprzężonych z regularyzacją (scaled conjugate gradients)

Algorytm *scg* jest dość skomplikowany. Jest on opisany u Osowskiego na str. 68–71 i Bishopa str 282–285, jednak opisy w obu tych źródłach różnią się.

Generalnie, algorytm ten wiąże sie z nazwiskiem Mollera i pozwala unikać bezpośredniego szukania minimum na danym kierunku przez tzw. line search. Algorytm stosuje regularyzację dla uniknięcia trudności wynikających z faktu, że hesjan rozważany w tej metodzie (wynikający z aproksymacji kwadratowej funkcji błędu w punkcie $\mathbf{w}^{(k)}$) na ogół bywa niedodatnio określony lub źle uwarunkowany.

Regularyzacja jakiejś wielkości wiąże się generalnie z dodaniem pewnego warunku na tę wielkość. Z dodaniem nowego warunku wiąże się dodanie nowego parametru do rozpatrywanego modelu.

W omawianym zagadnieniu regularyzacja może zawierać dwa momenty:

i) uczynić hesjan dodatnio określony przez dodanie do macierzy $\mathbf{H}$ pewnej krotności λ macierzy jednostkowej; tym samym w dalszym ciągu będzie rozpatrywana macierz $\mathbf{H}+\lambda\mathbf{I}$;

ii) wprowadzić regularyzację do formuły na długość kroku $\alpha^{(k)}$ wyznaczanego według wzoru (3.23), co powoduje, że krok ten będzie wyznaczany z następującego warunku

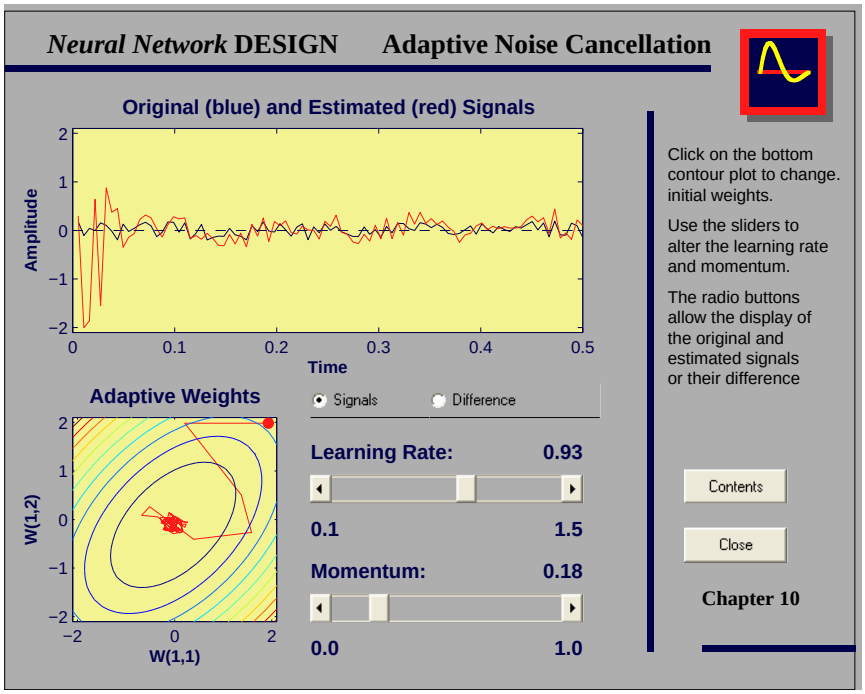
$$\alpha_{min}^{(k)} = \frac{\mathbf{p}^{(k)}\mathbf{g_k}}{(\mathbf{p}^{(k)})^T\mathbf{H}\mathbf{p}^{(k)} + \lambda_k||\mathbf{p}^{(k)}||^2}. \tag{3.24}$$

Parametr λ_k występujący w warunku regularyzacji jest dobierany indywidualnie w każdym kroku iteracji.

Jak pisze Bishop (str. 285), rezultaty symulacji komputerowych wykazują, że w niektórych przypadkach algorytm *scg* może dać znaczącą poprawę w szybkości obliczeń w stosunku do zwykłego algorytmu sprzężonych gradientów.

3.4 Demonstracje uczenia się perceptronu, nnd10nc i nnd4db

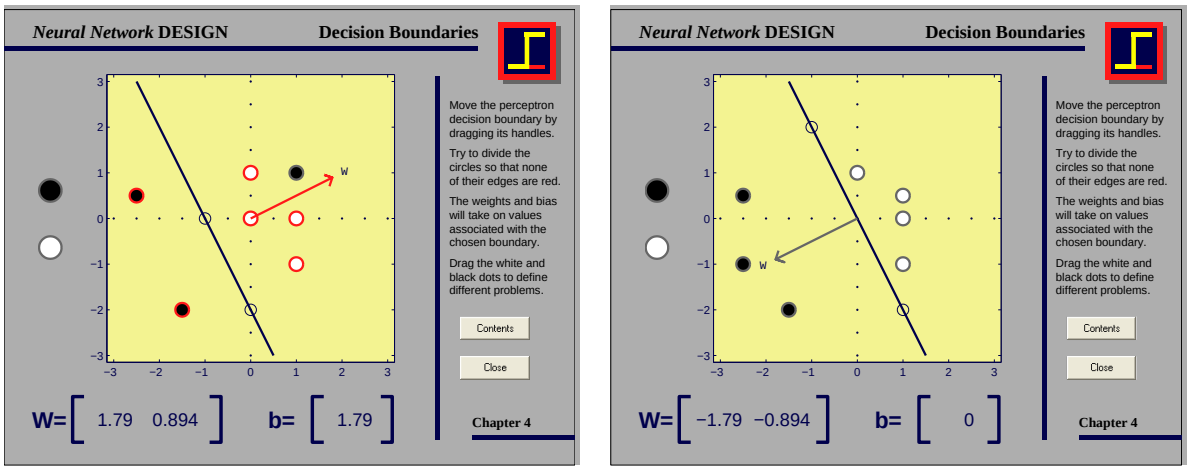
Rysunek 1 (nnd10nc.eps) przedstawia uczenie się perceptronu w zależności od *współczynnika uczenia* i wielkości *momentum*. Nieodpowiednie ustawienie tych parametrów może spowodować duże skoki między wynikami uzyskiwanymi w kolejnych iteracjach.



Oryginalny, zmienny w czasie sygnał w module nnd10nc jest zakłócony szumem modulowanym przez sinusoidę. Sieć musi odfiltrowywać szum (por. applin2 lub demolin8).

Rysunki 2a i 2b przedstawiają wykreślanie tzw. granicy decyzyjnej (*decision boundary*, moduł nnd4db) w przypadku 2 grup reprezentowanych przez dwa rodzaje punktów: czarnych (grupa docelowa) i białych, rozumianych jako grupa alternatywna. Kolor czerwony w obrysie punktu oznacza błędną klasyfikację. Wektor wag powinien wskazywać kierunek na grupę docelową, kolor czerwony tego wektora oznacza zły zwrot wektora wag.

Generalnie, równanie $w1*x+w2*y+b=0$ oznacza prostą rozgraniczającą płaszczyznę (x,y) na 2 części; wartość dodatnia funkcji $z = w1 * x + w2 * y + b$ określa domenę grupy czarnych, a wartość ujemna – domenę grupy białych.



Zamieszczone poniżej demonstracje są bardziej szczegółowo omówione w książce [3]. Teksty procedur są włączone do pakietu firmowego Matlaba jako nnet Toolbox.

Interesujące są również demonstracje demolin8 lub applin2.

Literatura

[1] Ch. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1996.

[2] Ch. M. Bishop, *Neural networks: a pattern recognition perspective*. Technical Report NCRG/96/001, http://www.ncrg.aston.ac.uk/

[3] Hagan M.T, Demuth H.B. and Beale M., Neural Network Design. PWS Publishing Company, Thompson, Boston, 1995.

[4] Ian Nabney, *Netlab: Algorithms for Pattern Recognition*. Springer 2001. Seria: Advances in Pattern Recognition. ISBN 1–85233–440–1.

[5] Netlab neural network software, Neural Computing Research Group, Division of Electric Engineering and Computer Science, Aston University, Birmingham UK, http://www.ncrg.aston.ac.uk/

Spis treści

3 Błąd sieci i algorytmy jego minimizacji **15**
3.1 Sposoby określania błędu – Ogólne omówienie 15
3.2 Definicje błędu wynikające z wiarygodności zdarzeń 16
3.2.1 Określenie wiarygodności próbki (sample likelihood) 16
3.2.2 Obliczanie błędu E w zagadnieniach regresyjnych 17
3.2.3 Obliczanie błędu E w zagadnieniach klasyfikacyjnych 18
3.2.4 Klasy algorytmów minimalizacyjnych 19
3.3 Algorytmy klasyczne do minimalizacji funkcji $E(w)$ 20
3.3.1 Zasada algorytmów iteracyjnych 20
3.3.2 Aproksymacja funkcji błędu za pomocą rozwinięcia w szereg Taylora – i co z tego wynika 21
3.3.3 Algorytm największego spadku 21
3.3.4 Metoda Newtona 22
3.3.5 Metoda zmiennej metryki – quasi-Newton 22
3.3.6 Algorytm Levenberga-Marquardta 23
3.3.7 Metoda sprzężonych gradientów 23
3.3.8 Metoda gradientów sprzężonych z regularyzacją (scaled conjugate gradients) 24
3.4 Demonstracje uczenia się perceptronu, nnd10nc i nnd4db 25

References **26**