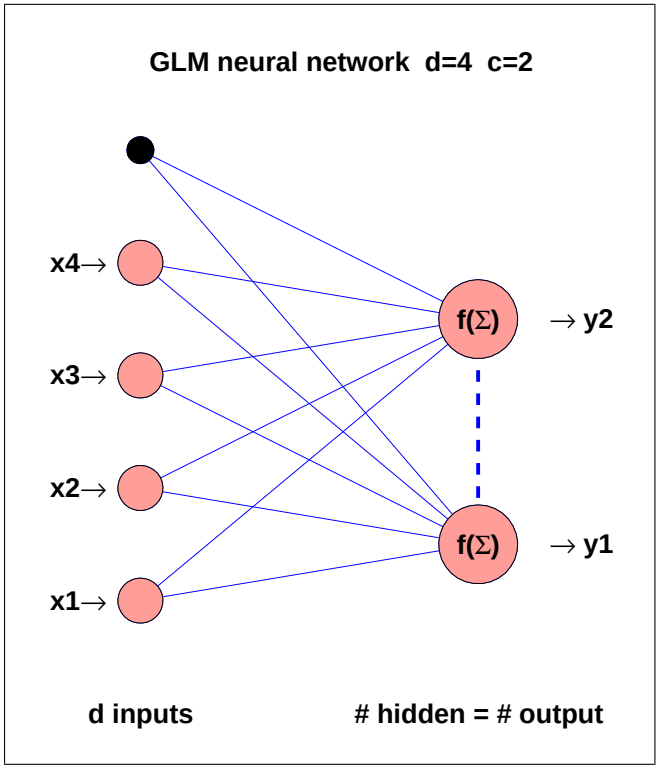
plik wglm.tex, 26 listopada 2005, 28.02.2006, 5.12.09

10 Klasyfikacja do K grup, Sieć probabilistyczna GLM

Jest to jeden z najprostszych modeli sztucznych sieci neuronowych. Może służyć do obliczeń zagadnień formułowanych jako modele liniowe (Linear Models) oraz tzw. uogólnione modele liniowe (Generalized Linear Models [1]). Pomimo swej prostoty, sieć GLM potrafi obliczać zagadnienia regresji wielokrotnej, liniowe zagadnienia najmniejszych kwadratów, oraz klasyfikacje dla kilku (c) grup. Obliczenia są bardzo szybkie, istnieje tylko jedno minimum optymalizujące zadane kryterium błędu.

10.1 Architektura sieci GLM

Architektura sieci GLM zaimplementowanej w pakiecie Netlab jest pokazana na Rysunku 10.1.



Oznaczenia
d – l. wejść, na rysunku $d = 4$
$\mathbf{x} = [x_1, \ldots, x_d]^T$ – wektor wejściowy
c – liczba neuronów warstwy ukrytej, i zarazem liczba wyjść, na rysunku $c = 2$
$\mathbf{w_j} = [w_{j1}, \ldots, w_{jd}]^T, \quad j = 1, \ldots, c$ wagi, z jakimi sumuje j-ty neuron
b_j – bias j-tego neuronu
a_j – aktywacja j-tego neuronu, $a_j = \mathbf{x}^T \mathbf{w_j} + b_j$
$f(.)$ – funkcja aktywacji, określona na aktywacjach a_j, może to być 'linear', 'logistic', 'softmax'
$y_j = f(a_j)$ – wynik na wyjściu j-tego neuronu, $j = 1, \ldots, c$
Połączenie między neuronami warstwy ukrytej (wyjściowej) jest uruchamiane tylko w przypadku funkcji aktywacji *softmax*

Rysunek 10.1: *Architektura sieci GLM o $d = 4$ wejściach i $c = 2$ wyjściach. Sieć GLM ma tylko jedną warstwę ukrytą o c neuronach, która to warstwa jest jednocześnie warstwą wyjściową* {*myGLM.eps*}

Oznaczenia:
$\mathbf{x} = (x_1, \ldots, x_d)^T$ — wektor danych (input vector), zmienne objaśniające
$\mathbf{y} = (y_1, \ldots, y_c)^T$ — wektor wyników (output vector), zmienne prognozowane
$a_j = \sum_{i=1}^{d} w_{ji} x_i + b_j, \quad j = 1, \ldots, c$ — aktywacje wejściowe neuronów warstwy ukrytej.

Wyjście y_j jest obliczane za pomocą funkcji aktywacji $f(.)$ wybieranej przez użytkownika. Argumentem funkcji f są aktywacje a_j (wynik sumowania, na rysznku oznaczony jako Σ), a rezultatem jest wynik $y_j = f(a_j)$ stanowiący odpowiedź sieci na sygnał wejściowy $\mathbf{x}$.

[1] są to takie modele, które przez prostą transformację dają się sprowadzić do modeli liniowych, por. [3]

Sieć GLM w Netlabie ma wbudowane następujące trzy funkcje aktywacji:

- funkcja *liniowa* $y_j = a_j$
- funkcja *logistyczna* $y_j = \frac{1}{1 + exp(-a)}$
- funkcja *softmax* $\frac{exp(a_j)}{\sum_{j'} exp(a_{j'})}$

Każda z wymienionych funkcji aktywacji jest właściwa dla innego zagadnienia:

funkcja liniowa – dla zagadnień regresyjnych typu $y = b_0 + b_1 x_1 + \ldots + b_d x_d + e$

funkcja logistyczna – dla klasyfikacji do 2 grup; obliczona wartość $y \in [0, 1]$ wyraża prawdopodobieństwo, że wektor $\mathbf{x}$ należy do grupy określanej jako 'klasa 1'

funkcja softmax – klasyfikacja danych do c grup (klas). Obliczone wartości wyjściowe $(y_1, \ldots, y_c)$ spełniają warunki: $0 \leq y_j \leq 1$, $\sum_{j=1}^{c} y_j = 1$.

10.2 Implementacja sieci GLM w pakiecie Netlab

Istotną rolę odgrywają tu trzy funkcje: glm, glmfwd i glmtrain. Są one wywoływane następująco:

```
net = glm(n_in, n_out, actfn);
Y = glmfwd(net, X);   lub   [Y, A] = glmfwd(net, X);
[net, options]= glmtrain(net, options, data, targets);
```

Funkcja GLM tworząca strukturę net

Pierwsza z funkcji tworzy obiekt 'net'.
Oczywiście zamiast nazwy 'net' można użyć dowolnej innej nazwy, np. 'my_net' lub 'jan' :

```
net = glm(n_in, n_out, actfn)
```

Znaczenie parametrów wejściowych:

n_in :	d, liczba neuronów warstwy wejściowej
n_out:	c, liczba neuronów warstwy wejściowej
actfn :	rodzaj funcji aktywacji: 'linear', 'logistic', 'softmax'.

Mogą jeszcze wystąpić dalsze parametry, np. prior, parametr opcyjny, wykorzystywany przy modelach Bayesowskich, określający rozrzut inicjowanych wag.

Utworzona struktura net ma następujące pola:

type	'glm'
n_in	d, liczba wejść (liczba cech danych)
n_out	c, liczba wyjść sieci
nwts	liczba wszystkich współczynników wagowych i biasów
actfn	funkcja aktywacji: string 'linear', 'logistic', 'softmax'
w1	tablica wag o wymiarze $d \times c$
b1	tablica biasów o wymiarze $1 \times c$

W przypadku używania złożonego modelu Bayesowskiego, struktura glmnet przewiduje dalsze pola zawierające parametry Bayesowskie: prior i beta. Jednak my nie będziemy zajmować się tym przypadkiem.

Struktura 'net' zainicjowana jak wyżej zawiera przyjęte losowo wartości wag oraz biasów. Np. Wartości wag zostały zainicjowane jako $randn(d, c)/sqrt(d-1)$

Funkcja GLMFWD służąca pracy w trybie odtworzeniowym

Funkcja glmfwd pozwala utworzonej sieci pracować w trybie odtworzeniowym, tzn. dla dostarczonych danych wejściowym dostarcza odpowiednie wyniki sieci.

Wywoływanie funkcji:

```
[Y, A] = glmfwd(net, X);   lub   Y = glmfwd(net, X);
```

Parametr X oznacza tu tablicę danych $\mathbf{X}$ o wymiarze $N \times d$ dla których chcemy otrzymać wyniki. Tablica ta zawiera N wektorów wejściowych:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1}^T \\ \vdots \\ \mathbf{x_N}^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \dots & \dots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nd} \end{bmatrix} \tag{10.1}$$

Jako rezultat działania funkcji otrzymujemy tablicę Y o wymiarze $N \times c$ oraz tablicę aktywacji A o wymiarze $N \times c$. Są one obliczane według następującego wzoru:

$$\underbrace{\mathbf{A}}_{N\times c} = \underbrace{\mathbf{X}}_{N\times d}\underbrace{\mathbf{W}}_{d\times c} + \underbrace{\mathbf{1}_N * \underbrace{\mathbf{b}}_{1\times c}}_{N\times c}, \qquad \underbrace{\mathbf{Y}}_{N\times c} = \underbrace{f(A)}_{N\times c},$$

gdzie $\mathbf{W} = [\mathbf{w_1}, \dots, \mathbf{w_c}]$ oraz $\mathbf{b} = [b_1, \dots, b_c]$ oznaczają wagi i biasy zapamiętane w polach net.w1 i net.b1 struktury 'net'; natomiast $f(.)$ oznacza zadeklarowaną funkcję aktywacji.

Oczywiście, jeśli wagi są przypadkowe, to otrzymane wyniki będą też przypadkowe. Dlatego też utworzona sieć powinna być najpierw 'wyuczona', inaczej mówiąc, wytrenowana, za pomocą specjalnej procedury. Stanowi ją funkcja glmtrain.

Funkcja glmtrain uczy sieć rozpoznawania zadanych wzorców.

Funkcja GLMTRAIN trenująca sieć wg. próbki uczącej

```
[net, options]= glmtrain(net, options, data, targets);
```

Parametry wejściowe:
options. Praca modelu, a przede wszystkim otrzymywane jako rezultat tej pracy wyniki, są uwarunkowane tzw. *opcjami*. Opcje te są zapamiętywane w tablicy options(1:18)[2]. Po uruchomieniu systemu Netlab jest dostępna tablica foptions(1:18), która ma zainicjowane wartości domyślne. Użytkownik może skopiować tę tablicę do swojej tablicy options, a następnie – już na gruncie swoich 'options' – nadać poszczególnym elementom odpowiednie wartości.

[2]Najważniejsze znaczenia warunków zakodowanych w tablicy options są opisane w komentarzu do funkcji glmtrain

Najważniejsze znaczenia warunków zakodowanych w tablicy options są opisane w komentarzu do funkcji glmtrain. Można również otrzymać opis tych warunków przez rozkaz help foptions. Najczęściej wykorzystywane są następujące opcje dotyczące procesu uczenia (trenowania):

options(1) – wartość 1 oznacza drukowanie błędu E po wykonaniu każdej iteracji; 0 oznacza pomijanie tych wydruków,

options(2) – wymagana dokładność wag;

options(3) – wymagana dokładność błędu E;

options(8) – zwracana wartość błędu (po wykonaniu obliczeń),

options(14) – maksymalna liczba iteracji.

data . Tablica 'data' powinna zawierać tzw. próbkę uczącą o postaci tablicy $\mathbf{X}_{N\times d}$ (zdefiniowanej wzorem [10.1]). Liczba N odpowiada w tym przypadku liczbie wektorów uczących $\mathbf{x_i}$ wchodzących w skład próbki uczącej.

targets. Jest to tablica pokazująca wzorce, które sieć ma się nauczyć rozpoznawać. Tablica ta jest postaci podobnej jak tablica 'data', z tym, że zawiera tylko c kolumn. Oznaczmy tę tablicę symbolem $\mathbf{T} = (t_{ij})$.

$$\mathbf{T} = \begin{bmatrix} \mathbf{t_1}^T \\ \vdots \\ \mathbf{t_N}^T \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1c} \\ \vdots & \dots & \dots & \vdots \\ t_{N1} & t_{N2} & \dots & t_{Nc} \end{bmatrix} \tag{10.2}$$

W zagadnieniach regresyjnych tablica $\mathbf{T}$ zawiera wartości numeryczne, natomiast w zagadnieniach klasyfikacyjnych – zmienne zero-jedynkowe, określające przynależność do poszczególnych klas.

Próbka ucząca składa się z dwu tablic: tablicy 'data', zawierającej zmienne objaśniające, oraz tablicy 'targets' zawierającej wartości docelowe. Jest odpowiedniość między wierszami obu tablic. Każdy wiersz $\mathbf{t_j}^T$ tablicy 'targets' stanowi zbiór wyników, jakich sieć ma dostarczyć jako odpowiedź na wektor zmiennych objaśniających $\mathbf{x_j}^T$.

$$data = \begin{bmatrix} \mathbf{x_1}^T \\ \vdots \\ \mathbf{x_N}^T \end{bmatrix} \quad \text{rozmiar } N \times d \qquad targets = \begin{bmatrix} \mathbf{t_1}^T \\ \vdots \\ \mathbf{t_N}^T \end{bmatrix} \quad \text{rozmiar } N \times c$$

Proces trenowania. Podczas uczenia sieć poprawia iteracyjnie swoje wagi w ten sposób, aby wyniki dostarczone przez sieć (tj. tablica $\mathbf{Y}$) możliwie mało różniły się od wektora wartości pożądanych (docelowych) danych w tablicy $\mathbf{T}$ ('targets'). W tym celu należy określić błąd E oraz algorytm aktualizujący wagi. Zagadnienia określania błędu sieci i jego minimizacji są omawiane w rozdziale 3.

Skuteczność wytrenowania sieci sprawdzamy na oddzielnej próbce nazywanej *próbką testową*, dla której oblicza się wielkość błędu. Dla zagadnień klasyfikacyjnych wielkość błędu sprawdza się na tzw. *macierzy pomieszania* (ang. *confusion matrix*, na podstawie której liczy się liczbę elementów sklasyfikowanych poprawnie.

10.3 Obliczanie błędu sieci

[e, y, a]= glmerr(net, x, t);

Wyniki 'y' i 'a' są opcyjne. Można wywoływać tylko: e= glmerr(net, x, t);

Parametry wejściowe:

net – struktura, utworzona funkcją 'glm' i ewtl. zmodyfikowana przez 'glmtrain',
x – tablica danych, wymiaru $N \times d$,
t – tablica wartości docelowych, wymiaru $N \times c$.

Wyniki:

e – tablica wymiaru $N \times 1$, błąd sieci, zależy od przyjętej funkcji aktywacji
opcyjnie:
y – wyniki sieci podawane na wyjściu, czyli tablica $Y_{N\times c}$
a – aktywacje neuronów warstwy ukrytej, tablica

$$\mathbf{A}_{N\times c} = [\tilde{\mathbf{X}}\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{X}}\tilde{\mathbf{w}}_c] = \mathbf{X}\tilde{\mathbf{W}},$$

znak tildy oznacza tu odpowiednio poszerzone tablice $\mathbf{X}$ – o kolumnę jedynek, oraz $\mathbf{W}$ – o biasy neuronów:

$$\tilde{\mathbf{X}} = [\mathbf{X}, \mathbf{1}_N], \quad \tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_c] = \begin{bmatrix} \mathbf{w}_1 & \ldots & \mathbf{w}_c \\ b_1 & \ldots & b_c \end{bmatrix}.$$

Błąd e może być zapamiętany w tablicy options jako element options(8).
A oto treść funkcji obliczającej błąd sieci:

```
[y, a] = glmfwd(net, x);

switch net.outfn

  case 'linear'     % Linear outputs
    edata = 0.5*sum(sum((y - t).^2));

  case 'logistic'   % Logistic outputs
    edata = - sum(sum(t.*log(y) + (1 - t).*log(1 - y)));

  case 'softmax'    % Softmax outputs
    edata = - sum(sum(t.*log(y)));

  otherwise
    error(['Unknown activation function ', net.outfn]);
end
```

10.4 Obliczanie wag neuronów

Algorytm obliczania wag zależy od przyjętej funkcji aktywacji. W Netlabie dopuszcza się następujące funkcje aktywacji: liniowa, logistyczna, softmax.

Wprowadzamy tu **specyficzne** oznaczenie **wag** neuronów: będziemy je oznaczać symbolem $\boldsymbol{\beta}$ (a nie symbolem $\mathbf{w}$, jak to robimy w innych rozdziałach).

Dla c neuronów warstwy ukrytej szukamy c wektorów wagowych $\boldsymbol{\beta}_j$, gdzie

$$\boldsymbol{\beta}_j = [\beta_1, \ldots, \beta_d, \beta_0]^T, \ j = 1, \ldots, c.$$

Ostatni, $(d+1)$-szy element oznacza bias j-tego neuronu.

Liniowa funkcja aktywacji

W przypadku liniowej funkcji aktywacji funkcję błędu E wyznacza się na zasadzie najmniejszych kwadratów różnic (metoda Least Squares). Aby wyznaczyć najlepsze wagi i biasy, jest jednorazowo rozwiązywany układ równań liniowych

$$\mathbf{X}\mathsf{B} = \mathbf{T} \tag{10.3}$$

gdzie: $\mathbf{X}_{N\times(d+1)}$ – tablica danych z dodatkową kolumną 'jedynek' na ostatnim miejscu,

$\mathsf{B}_{(\mathbf{d+1})\times\mathbf{c}}$ – wagi neuronów, kolumnami, ostatnia składowa kolumny zawiera bias danego neuronu:

$$\mathsf{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_c] = \begin{bmatrix} \mathbf{w}_1 & \ldots & \mathbf{w}_c \\ b_1 & \ldots & b_c \end{bmatrix}$$

$\mathbf{T}_{N\times c}$ – tablica wartości docelowych.

W przypadku sieci GLM mamy tyle neuronów co wyjść, stąd $H \equiv c$.

Wypisany wyżej układ równań (10.3) jest nadokreślonym układem równań liniowych na wagi i biasy sieci. Układ ten rozwiązuje się w Netlabie przez operację `matrix left divide`.

Logistyczna funkcja aktywacji

Tutaj jako funkcję błędu przyjmujemy $E = -log\mathcal{L}$; patrz rozdział 3 notatek i skrypt w paragrafie 4.3 powyżej. Zależności między danymi wejściowymi ($\mathbf{X}$), wagami sieci ($\mathbf{B}$) i wartościami docelowymi ($\mathbf{T}$) są nieliniowe, ale – przy logistycznej funkcji aktywacji – podpadają pod tzw. uogólniony model liniowy (GLM) opisany dalej w sekcjach 4.7 i 4.8. Algorytm znajdywania wag w takim modelu sprowadza się do sekwencyjnego rozwiązywania układu równań liniowych "ważonych" (wagi indywidualne dla każdego osobnika) (10.4), w kórych jako niewiadoma występuje wektor wag $\boldsymbol{\beta}_j$ charakteryzujący j-ty neuron [3], $j = 1, \ldots, c$.

Rozważymy najpierw przypadek, gdy liczba klas = 2, a $c = 1$. Jest to stosunkowo często stosowany przypadek, gdyż przy liczbie klas > 2 stosujemy raczej funkcję softmax.

Przy dwóch klasach mamy: $\mathbf{B}_{(d+1)\times 1} = \boldsymbol{\beta}_{(d+1)\times 1}$, czyli sieć składa się z jednego neuronu i mamy tylko jeden wektor wag $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_0)^T$; również $\mathbf{T} = \mathbf{t}_{N\times 1}$.

[3] mamy tu dwa rodzaje wag: osobnicze, określające 'wagę' każdego wiersza tablicy $\mathbf{X}$ przy wyznaczaniu wartości docelowych , $\mathbf{T}$, oraz wagi neuronów używanych do wyznaczania aktywacji neuronu w odpowiedzi na sygnał wejściowy $\mathbf{x}^n$.

Niech $\mathbf{\Pi}_{N\times 1} = (\pi_1, \ldots, \pi_N)^T$ oznacza wektor prawdopodobieństw, że i-ty osobnik należy do klasy '1' ('sukces' w rozkładzie binomialnym, przy osobnikach zróżnicowanych ze względu na zmienne objaśniające). Prawdopodobieństwa te będziemy szacować iteracyjnie. Najpierw przyjmiemy jakąś wartość początkową $\mathbf{p}$, a następnie, w kolejnych iteracjach, wartość tę będziemy 'poprawiać' (mówiąc dokładniej, wartość 'poprawioną $\mathbf{p}$ będziemy otrzymywać – z procedury `glmerr`).

Wartość $\mathbf{\Pi}$ posłuży do skonstruowania układu równań (rLS)

$$\mathsf{W}^{1/2}\mathbf{X}\boldsymbol{\beta}^{(k+1)} = \mathsf{W}^{1/2}\mathsf{z}, \tag{10.4}$$

gdzie: W= diag$(w_{11}, \ldots, w_{NN})$ oznacza wagi i-tego osobnika (równania), przy czym
$w_{ii} = [p_i(1-p_i)]^{1/2}$, $i = 1, \ldots, N$,
$\mathsf{z}_{N\times 1} = \log(\mathbf{p}./(1-\mathbf{p})) + (\mathbf{t}-\mathbf{p})./[\mathbf{p}(1-\mathbf{p})]$,
wektor pomocniczy obliczony z szacunku $\mathbf{p}$ w (k)-tej iteracji.

Trenowanie sieci odbywa się w postaci wsadowej (batch) na podstawie całej próbki uczącej, tj. wszystkich elementów $\{\mathbf{x}_i, t_i\}$, $(i = 1, \ldots, N)$ – przy użyciu wzoru (10.4).

Inicjacja (krok $k = 0$) polega na wstępnym przyjęciu $\mathbf{p} = (\mathbf{t} + 0.5)/2$. Z wartości tych wyznaczamy wstępne wartości W i z; następnie rozwiązujemy układ równań ze względu na wektor $\boldsymbol{\beta}$. Znajomość $\boldsymbol{\beta}$ (wag sieci) pozwala nam zaktualizować wartość $\mathbf{p}$.

Czynności te powtarzamy dla $k = 1, \ldots,$ aż osiągniemy żądaną dokładność, lub wyczerpiemy zadeklarowaną liczbę iteracji.

W każdej iteracji: (a) Przy znanym $\mathbf{p}$ wyznaczamy W oraz z.

(b) Mając te wartości, rozwiązujemy układ równań 10.4 – robimy to przez operację `matrix left divide`; następnie wyznaczamy błąd sieci i nowe wartości $\mathbf{p}$.

Przypadek, gdy liczba klas c jest większa od 2. Rozbijamy zagadnienie na c niezależnych problemów z dwoma klasami i rozwiązujemy każdy problem oddzielnie, jak w przypadku 2 klas.

Wartości stopu

Funkcja aktywacji softmax

W przypadku aktywacji **softmax** i options(5)=1, to powtarzamy c-krotnie (tyle razy, ile jest wyjść) postępowanie stosowane przy aktywacji logistycznej. Ostateczne wyniki (p-stwa á posteriori) są skalowane tak, aby sumowały się do jedności. W przypadku options(5)=0 stosujemy dokładne obliczenia Hesjanu.

10.5 Moduły demonstracyjne do modelu GLM

Pakiet Netlab oferuje moduły demglm1 i demglm2 ilustrujące klasyfikację do dwóch i trzech grup przy użyciu sieci GLM korzystającej z funkcji aktywacji 'logistic' (2 grupy danych) i 'softmax' (3 grupy danych).

10.5.1 Moduł demglm1

W module demglm1 generuje się najpierw dwie grupy danych dwuwymiarowych. Wygenerowne dany zostają wykreślone na płaszczyźnie, każda grupa innym kolorem. Dane

zostają podzielone na próbkę uczącą i próbkę testową czyli sprawdzającą. Sieć `glm` ma tutaj architekturę

input: d=2 → hidden: H=1 → output: c=1

Macierz wartości docelowych składa się tylko z jednej kolumny $\mathbf{t}_{N\times 1}$, zawierającej '1', gdy wygenerowany wektor $\mathbf{x}^T$ należy do pierwszej grupy, oraz wartość '0', gdy należy do grupy drugiej.

Następnie tworzymy strukturę 'net' za pomocą konstruktora `glm` (deklarując funkcję 'logistic' jako funkcję aktywacji) i trenujemy utworzoną sieć za pomocą funkcji `glmtrain`.

Jako rezultat trenowania otrzymujemy wagi sieci zaadaptowane do rozpoznawania klas danych. Samo rozpoznawanie jest wykonywane za pomocą funkcji `glmfwd`. Dla danego wektora $\mathbf{x}$, który ma być rozpoznany, sieć oblicza najpierw aktywację (znak tildy oznacza odpowiednio poszerzone wektory $\mathbf{x}$- poszerzony o wartość 1, i $\mathbf{w}$ - poszerzony o bias)

$$a = a(\mathbf{x}) = \tilde{\mathbf{x}}^T \tilde{\mathbf{w}},$$

a następnie swój wynik

$$y = y(\mathbf{x}) = \frac{1}{1 + \exp\{-a(\mathbf{x})\}}\,.$$

Na tym samym wykresie, na którym wyrysowane punkty indywidualne wygenerowanej próbki, można również wykreślić powierzchnię funkcji $y(\mathbf{t}|\mathbf{x}, \mathbf{w})$ i jej warstwice. W szczególności interesujące są warstwice dla $y = 0.1, 0.5, 0.9$. Odpowiedni fragment skryptu wykreślający te warstwice przedstawiono ponyżej.

```
% fragment skryptu glmdem1.m,
% wykresla kontury i powierzchnie 'mesh' i 'surf'
x = -4.0:0.2:5.0;   y = -4.0:0.2:5.0; % jak g"esta ma byc siatka
[X, Y] = meshgrid(x,y);               % utworzone X, Y sa dwuwymiarowe
X = X(:); Y = Y(:);                   % teraz X, Y s"a kolumnami
Z = glmfwd(net,[X Y]);                % [X Y] to tablica danych o 2 kolumnach
Z = reshape(Z, length(x), length(y));
v = [0.1 0.5 0.9];                    % wysoko"sci na ktorych przekroje
[c, h] = contour(x, y, Z, v); % wykreslanie konturow na istniejacym rysunku
title('Model GLM', 'Fontsize',12)
set(h, 'linewidth', 3)
clabel(c,h,'fontsize',15,'fontweight','bold','color','r',...
         'labelspacing',72,'rotation',0)  % opisywanie konturow
fh4=figure, surf(x,y,Z) % Wykreslanie powierzchni pokrytej plytkami
fh5=figure, mesh(x,y,Z) % Wykreslanie powierzchni zaznaczonej liniami
```

Jakość klasyfikacji można sprawdzić za pomocą tzw. macierzy pomieszania (*confusion matrix*) – otrzymamy wtedy tablicę 2×2 pokazującą dla i=0,1 ile elementów klasy i zostało rozpoznanych jako należących do klasy 2 (oznaczanej jako '0'), a ile – do klasy 1.

Taką tablicę możemy otrzymać za pomocą funkcji `conffig` (wywołanie: `conffig(y_test,t_test);`).

10.5.2 Moduł demglm2

W modelu tym porównuje się wyniki otrzymane z sieci neuronowej z wynikami otrzymanymi Bayesowską metodą prawdopodobieństw a posteriori. Jest to przedstawione za pomocą interesującej grafiki.

10.6 Uczenie sieci: liniowa funkcji aktywacji

Liniowa funkcja aktywacji prowadzi do funkcji błędu, która jest kwadratową funkcją wag. Szukamy wtedy wag , które dają minimum następującej formy kwadratowej {normalEq}:

$$\mathbf{W}_0 : \quad \min_{\mathbf{W}} (\mathbf{Y} - \mathbf{X}\mathbf{W})^T (\mathbf{Y} - \mathbf{X}\mathbf{W}), \tag{10.5}$$

gdzie $\mathbf{W}_{(d+1)\times c}$ – wagi i biasy podstawione do jednej tablicy,
$\mathbf{Y}_{N\times c}$ – tablica zawierająca wyniki sieci
$\mathbf{X}_{N\times(d+1)}$ – tablica danych 'data' poszerzona o kolumnę jedynek $\mathbf{I}_N$.

Warunek (10.5) prowadzi do tzw. układu równań normalnych, będących układem liniowych równań na elementy macierzy $\mathbf{W}$, dających rozwiązanie w jednym kroku.

Alternatywnie można otrzymać rozwiązanie przez pseudoodwrotność. Rozpatrujemy wtedy układ równań

$$\mathbf{X}\mathbf{W} = \mathbf{Y},$$

które rozwiązujemy przez pseudoodwrotność (funkcja `pinverse` w Matlabie).

Niech $\mathbf{X}^\dagger$ oznacza pseudoodwrotność macierzy $\mathbf{X}$. Wtedy $\mathbf{W} = \mathbf{X}^\dagger\mathbf{Y}$ daje nam szukane rozwiązanie.

10.7 Uczenie sieci: logistyczna funkcja aktywacji

Funkcje błędu dla zagadnień klasyfikacji do 2 lub $c > 2$ klas zostały przedstawione w rozdziale 3. Są to wzory wynikające z odpowiednich modeli probabilistycznych (binomialnych) i związaną z tymi modelami funkcją `wiarogodności` obserwacji. Funkcja błędu E oblicza się wtedy jako logarytm z wiarogodności $\mathcal{L}$ wzięty ze znakiem przeciwnym. Wynikające stąd wzory na błędy (entropia krzyżowa) zależą od wag $\mathbf{w}$, które należy wyznaczyć iteracyjnie w procesie uczenia (treningu) sztucznej sieci neuronowej.

Jeżeli funkcja błędu E jest różniczkowalna, to w celu minimizacji błędu możemy stosować metody gradientowe. Zasada jest wtedy taka, że dysponując wektorem $\mathbf{w}^{(k)}$ otrzymanym w k-tym kroku, "poprawiamy" ten wektor poruszając się o małą odległość w kierunku, w którym funkcja E maleje najbardziej, czyli w kierunku ujemnego gradientu.

W ten sposób w kolejnych krokach $\{k\}$, $k = 0, 1, \ldots$ mamy nadzieję zbliżać się do minimum funkcji E.

Na początku algorytmu musimy podać jakieś wstępne przybliżenie $\mathbf{w}^{(0)}$. Często jest to przybliżenie wygenerowane losowo.

W dalszych krokach[4] ciąg kolejnych przybliżeń jest konstruowany wg zasady: mając wektor $\mathbf{w}^{(k)}$, konstruujemy $\mathbf{w}^{(k+1)}$ wedlug wzoru

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \frac{\partial E}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}^{(k)}}.$$

Współczynnik η nosi nazwę współczynnika uczenia (*learning rate*).

Jednak nie za bardzo wiadomo, jaka ma być wartość η. Często wartość tę wyznacza się eksperymentalnie.

Jedną z uznawanych metod numerycznych, gwarantujących zbieżność (chociaż może to być tylko zbieżność lokalna) jest metoda Newtona - Raphsona. Metoda ta poleca konstruować $(k+1)$sze przybliżenie wag wg wzoru:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mathbf{H}^{-1} \frac{\partial E}{\partial \mathbf{w}}.$$

[4] przedstawimy dalej algorytm opisany w książce Nabney'a [4]

Metoda ta wymaga odwracania hesjanu, co przy większych rozmiarach wag $\mathbf{w}$ może być kłopotliwe, również dlatego, że wymagana jest dodatnia określoność macierzy $\mathbf{H}$.

Fisher zaproponował metodę, która zamiast hesjanu $\mathbf{H}$ podstawia wartość oczekiwaną tej macierzy (metoda ta jest nazywana *Fisher's scoring method* [5], u nas $E = -log\mathcal{L}$, gdzie $\mathcal{L}$ jest wiarygodnością próby, a wektor $\mathbf{w}$ pełni rolę wektora parametrów) {wzór FisherScor}

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \{E[\mathbf{H}]\}^{-1}\frac{\partial E}{\partial \mathbf{w}}. \tag{10.6}$$

Jeśli Funkcja aktywacji f jest funkcją liniową, to Hesjan dla tak określonego modelu liniowego wynosi $\mathbf{X}^T\mathbf{X}$ [4], str 129–132. Hesjan dla przypadku f='logistic' jest pokazany niżej.

Rozpatrzmy **zagadnienie klasyfikacyjne dla dwóch klas** określonych binarnie, gdy dla danego wektora danych $\mathbf{x}$ sieć ma podać wynik $y^n(\mathbf{x})$ określający przynależność do grupy (klasy) 1 ($0 \leq y^n \leq 1$. Prawdziwa przynależność do klasy jest określona podaną wartością $t^n \in \{0, 1\}$:

$$\mathbf{x}^n \rightarrow y^n \quad \text{ma przybliżać} \quad t^n$$

Zakładamy, że funkcją aktywacji f jest funkcja **logistic**

Niech $\mathbf{\Pi}$ oznacza macierzą diagonalną: $\mathbf{\Pi} = diag\{y^n(1-y)^n\}$.
Mamy (wzory cytowane za [4], str 129–132)

- Gradient $\partial E/\partial \mathbf{w} = \mathbf{X}^T\mathbf{\Pi}\mathbf{e}$, gdzie $e^n = (y^n - t^n)/f'(a^n)$, f - funkcja aktywacji, a^n – aktywacja neuronu w odpowiedzi na sygnał wejściowy $\mathbf{x}^n$,
- Hesjan dla tego samego modelu równa się $\mathbf{X}^T\mathbf{\Pi}\mathbf{X}$,

Wypisane wyżej wzory pozwalają na prosty **algorytm** obliczeń, zastosowany w pakiecie Netlab. Dla posiadanej wartości $\mathbf{w}^{(k)}$ tworzymy zmienną

$$\mathbf{z_k} = \mathbf{X}\mathbf{w}^{(k)} + \mathbf{e}.$$

Następnie rozwiązujemy iteracyjnie ważone liniowe zagadnienia najmniejszych kwadratów (iterrative reweighted least squares equations) czyli następujący układu równań normalnych (wektor $\mathbf{w}$ jest tu wektorem kolumnowym rozmiaru $(d+1)\times 1$, a $\mathbf{X}$ macierzą danych poszerzoną o kolumnę jedynek): {wzór iksPi}

$$(\mathbf{X}^T\mathbf{\Pi}_k\mathbf{X})\mathbf{w}^{(k+1)} = \mathbf{X}^T\mathbf{\Pi}_k\mathbf{z}_k. \tag{10.7}$$

Układ wypisany powyżej jest równoważny zwykłemu układowi równań normalnych, jeśli podstawimy za macierz danych tablicę $\mathbf{X}^T\mathbf{\Pi}_k^{1/2}$, a za zmienną zależną tablicę $\mathbf{\Pi}_k^{1/2}\mathbf{z}_k$.

Wartości $\mathbf{\Pi}_k$ i $\mathbf{z}_k$ zmieniają się w każdej iteracji, co zaunaczono wskaźnikiem 'k'.

W przypadku **klasyfikacji do więcej grup i** używania **funkcji aktywacji 'softmax'** metoda minimalizacji błędu zależy od options(5); jeżeli options(5)=1, to stosuje się metodę IRLS jak w przypadku f='logistic' (Nabney [4] podaje również wzory na single output softmax model. Nie są one dużo bardziej skomplikowane aniżeli te, podane wyżej dla modelu logistycznego), w przeciwnym przypadku jest stosowana dokładna metoda numeryczna Newtona-Raphsona z obliczaniem odwrotności hesjanu.

W wyniku trenowania otrzymuje się aktualizację wag i biasów (ogólnie: wektora wag $\mathbf{w}$) oraz nową wartość błędu E. Ten ostatni jest zapamiętany jako options(8).

[5] Fisher w swoim podejściu rozważal logarytm wiarygodności $l = log\mathcal{L}$ przedstawiany jako funcja parametru θ i szukał *maksimum* funkcji l ze względu na parametr θ. Funcja $l(\theta)$ wyznaczona dla N-elementowej próby o niezależnych elementach dawała się przedstawić jako suma $\sum_{i=1}^{N} l_i(\theta))$

10.8 Metoda GLM – uogólniony model liniowy

W tej sekcji zostanie przedstawiona teoria uogólnionych modeli liniowych GLM rozwinięta na gruncie statystyki matematycznej. Model ten (tj. GLM) obejmuje jako szczególny przypadek zagadnienie klasyfikacji do 2 grup danych, gdy p-stwo klasyfikacji π zależy od obserwowanych dla każdego osobnika zmiennych objaśniających $X_1, \ldots, X_d$.

10.8.1 Rodzina wykładnicza rozkładów probabilistycznych

Definicja rodziny wykładniczej

Rozkład $f(y;\boldsymbol{\theta})$ należy do rodziny wykładniczej rozkładów, jeśli daje się zapisać w postaci: $\{wykladniczy\}$

$$f(y;\boldsymbol{\theta}) = exp\,[\,a(y)b(\theta) + c(\theta) + d(y)], \tag{10.8}$$

gdzie funkcje b,c,d są różniczkowalne conajmniej dwa ray.

Jeżeli $a(y) \equiv y$, to mówimy, że $b(\boldsymbol{\theta})$ jest parametrem **naturalnym** rozkładu, a postać (10.8) jest postacią kanoniczną rozkładu.

Zaznaczmy tutaj, że wiele znanych i używanych w praktyce rozkładów probabilistycznych należy do tej rodziny; między nimi są również rozkład binarny (zero-jedynkowy, Bernoulliego), rozkład dwumianowy, Poissona, multinomialny, gamma, i rozkład normalny. Przy rozpatrywaniu zagadnień klasyfikacyjnych interesuje nas najbardziej rozkład zero-jedynkowy (Bernoulliego), będący szczególnym przypadkiem rozkładu dwumianowego (dla $n = 1$).

Przykład: Rozkład dwumianowy $Y \sim binomial(n, \pi)$. Tutaj π, prawdopodobieństwo 'sukcesu', jest interesującym nas paramerem, a n, długość serii, jest dane. Dla $n = 1$ otrzymujemy rozkład binarny.

Funkcja rozkładu p-stwa daje się zapisać w postaci kanonicznej jako $(y = 0, 1, \ldots, n)$:

$$f(y;\pi) = exp\Big[ylog\pi - ylog(1-\pi) + n(1-\pi) + log\binom{n}{y}\Big]$$

Tak więc mamy tu: $b(\pi) = log\pi - log(1-\pi) = log[\pi/(1-\pi)]$.

Jest to zarazem parametr naturalny dla rozkładów dwumianowego i binarnego (zero–jedynkowego).

Własności rozkładów rodziny wykładniczej

$$E[a(Y)] = -c'(\theta)/b'(\theta). \tag{10.9}$$

$$Var[a(Y)] = \frac{b''(\theta)c'(\theta) - c''(\theta)b'(\theta)}{[b'(\theta)]^3}. \tag{10.10}$$

Wartośc oczekiwana i wariancja są tu liczone ze względu na rozkład zmiennej losowej Y.

Przykład: Rozkład dwumianowy c.d. Dla rozkładu dwumianowego mamy:

$$\theta = \pi, \quad a(Y) \equiv Y, \quad b(\pi) = \log[\pi/(1-\pi)], \quad c(\pi) = n\log(1-\pi)$$

$$b'(\theta) = \frac{1}{\pi(1-\pi)}, \quad b''(\theta) = \frac{\pi - (1-\pi)}{\pi^2(1-\pi)^2}, \quad c'(\theta) = \frac{-n}{1-\pi}, \quad c''(\theta) = \frac{-n}{(1-\pi)^2}.$$

Skąd – ponieważ $a(Y) \equiv Y$ – znane wzory: $E(Y) = n\pi, \quad Var(Y) = n\pi(1-\pi)$.

Wiarygodność $l(\theta; y) = \log \mathcal{L}$ i jej pochodne dla rodziny wykładniczej

Niech $l(\theta; y) = \log \mathcal{L}$ oznacza logarytm funkcji wiarogodności. Dla rodziny wykładniczej mamy

$$l(\theta; y) = a(y)b(\theta) + c(\theta) + d(y)$$

Aby wyznaczyć parametr dający maximum wiarygodności, należy obliczyć pochodną względem θ i przyrównać ją do zera (akładamy, że funkcja ta jest dostatecznie regularna, a maximum nie leży na brzegu dziedziny funkcji pochodnej). Dla rodziny wykładniczej definiujemy:

$$U(\theta; y) = \frac{\partial l(\theta; y)}{\partial \theta} = a(y)b'(\theta) + c'(\theta).$$

Definicja Funkcja U nosi nazwę **score statistics**.
Wartość oczekiwana funkcji U: E(U)= 0. Dowód:

$$E(U) = \{E[a(Y)]\}b'(\theta) + c'(\theta) = [-c'(\theta)/b'(\theta)]\ b'(\theta) + c'(\theta) = 0.$$

Wariancja funkcji U: $Var(U) = E(U^2) = -E(U')$, gdzie $U' = \partial U/\partial\theta$.
Wariancję U możemy również obliczyć z ogólnego wzoru jako wariancję transformowanej liniowo zmiennej a(y):

$$\mathcal{I} = Var(U) = [b'(\theta)^2]\, Var[a(Y)] = \frac{b''(\theta)c'(\theta)}{b'(\theta)} - c''(\theta)$$

Zauważmy, że $\mathcal{I}$ zależy jedynie od parametrów przyjętego modelu, nie zależy natomiast od wartości oczekiwanych Y.

Definicja. Wariancja U nosi nazwę **informacji**.
Jako pochodna $U' = \partial U/\partial\theta$ (dokładnie: $Var(U) = -E(U')$) wyznaczona w punkcie $\hat{\theta}$ dostarcza ona informacji, jak szybko zmienia się funkcja wiarogodności (a właściwie jej logarytm) w okolicy maksimum. Jeśli pochodna ta zmienia się bardzo powoli (funkcja jest płaska), to precyzja wyznaczonego estymatora jest mała, a wariancja wyznaczonego estymatora duża.

Przykład: Rozkład dwumianowy c.d. Dla rozkładu dwumianowego mamy:

$$U = a(Y)b'(\theta) + c'(\theta) = \frac{y}{\pi(1-\pi)} - \frac{-n}{1-\pi}\,.$$

Biorąc pod uwagę, że dla r. dwumianowego $Var(Y) = n\pi(1-\pi)$, otrzymujemy

$$Var(U) = \frac{n\pi(1-\pi)}{\pi^2(1-\pi)^2} = \frac{n}{\pi(1-\pi)} = \mathcal{I}\,.$$

10.8.2 Ogólna koncepcja GLM

Metoda GLM jest oparta na statystycznej teorii uogólnionych modeli liniowych wprowadzonych przez Neldera i Wedderburna w 1972 roku (por [7]). Teoria ta oraz referencje do innych monografii i prac na temat GLM są podane m.in. w książce Annette Dobson [3].

Metoda GLM uogólnia statystyczną teorię modelu regresyjnego (czyli wyznaczania zależności (regresji) zmiennej Y od zmiennych objaśniających $X_1, \ldots, X_d$).

U podstaw uogólnionego modelu liniowego leży założenie, że obserwowane zmienne losowe $Y_1, \ldots, Y_N$ zależą od indywidualnych parametrów $\theta_1, \ldots, \theta_N$, które to parametry są w praktyce nieznane. Co gorsza – parametry te są niemożliwe do wyestymowania, jeżeli

dysponujemy tylko jedną obserwacją dla każdego Y_i. Ponadto, przy większych próbkach, parametrów do wyestymowania byłoby zbyt wiele.

W tej sytuacji powstała teoria uogólnionych modeli liniowych GLM, nazywana również GLIM, od Generalzed Linear Models. Teoria ta stara się

a) zmniejszyć liczbę parametrów opisujących daną zbiorowość
b) przy możliwie małej liczbie parametrów wymodelować zależności nieliniowe obserwowanej zmiennej Y od zmiennych objaśniających $X_1, \ldots, X_d$.

Metoda GLM stara się scharakteryzować rozkłady obserwowanych zmiennych losowych $Y_1, \ldots, Y_N$ mniejszą liczbą parametrów $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)^T$, $(d < N)$ takich, że kombinacja liniowa przyjmowanych parametrów $\beta_1, \ldots, \beta_d$ jest pewną funkcją wartości oczekiwanej i-tej obserwowanej wartości zmiennej Y_i.

Innymi słowy: Ideą uogólnionego modelu liniowego jest "that the image of the mean response by a given *link function* can be modelled *via* a linear relationship." [6].

Niech $Y_1, \ldots, Y_N$ będą obserwowanymi wartościami próbkowymi oznaczającymi wartości zmiennej Y zaobserwowane dla N różnych osobników. Zakładamy, że:

- Realizacje $Y_1, \ldots, Y_N$ są niezależne.
- Rozkład Y_i, $(i = 1, \ldots, N)$ opisuje się rozkładem $f(y; \theta_i)$, tj. tym samym rozkładem dla każdego i, ale z parametrem θ_i który może być indywidualny dla osobnika nr. i,
- Rozkład $f(y_i; \theta_i)$ należy do rodziny wykładniczej rozkładów i jest postaci kanonicznej, co oznacza, że daje się zapisać w postaci: $\{wykladniczyCan\}$

$$f(y_i; \theta_i) = exp\,[\, y_i b(\theta_i) + c(\theta_i) + d(y)]. \tag{10.11}$$

- Rozkład Y_i zależy od obserwowanych zmiennych objaśniających $X_1, \ldots, X_d$ których wartości są znane: tzn. dla każdego Y_i znamy odpowiadające wartości $\mathbf{x}_i^T = (x_{i1}, \ldots, x_{id})$. Zależność ta jest postaci

$$g(\mu_i) = \mathbf{x}_i^T \boldsymbol{\beta}. \tag{10.12}$$

gdzie:

μ_i oznacza wartość oczekiwaną zmiennej losowej Y_i: tj. mamy $E(Y_i) = \mu_i$.
$g(.)$ jest tzw. funkcją–łącznikiem (**link function**); jest to funkcja monotoniczna i różniczkowalna.

Tak więc zależność Y_i ze zmiennymi objaśniającymi nie jest bezpośrednia, ale pośrednia, poprzez zastosowanie funkcji-linku $g(.)$.

Powtarzając: Teoria uogólnionych modeli liniowych (GLM) wiąże wartości oczekiwane μ_i z obserwowanymi wektorami $\mathbf{x}_i$ poprzez funkcję g. Struktura zależnościowa 10.12 może być zapisana w innej postaci, ułatwiającej obliczanie funkcji odwrotnej i pochodnej:

$$g(\mu_i) = \eta_i, \quad \text{gdzie} \quad \eta_i = \mathbf{x}_i^T \boldsymbol{\beta} \tag{10.13}$$

Tak więc mamy

$Y_1, \ldots, Y_N$	-	obserwowane zmienne losowe, wzajemnie niezależne
$\mu_1, \ldots, \mu_N$	-	wartości oczekiwane wyznaczone jako $\mu_i = E(Y_i)$
$\mathbf{x}_1, \ldots, \mathbf{x}_N$	-	odpowiadające im wektory zmiennych objaśniających
$\eta_1, \ldots, \eta_N$	-	odpowiadające im kombinacje liniowe $\eta_i = \mathbf{x}_i^T \boldsymbol{\beta}$.

W ten sposób, za pomocą równań (10.12) i (10.13), – dla dowolnego osobnika i, dla którego uzyskaliśmy obserwację Y_i – zostało określone, że :

wartość oczekiwana zmiennej Y_i tego osobnika, czyli wartość μ_i jest funkcją (kombinacją) liniową wartości $x_{i1}, \ldots, x_{id}$ stanowiących zmienne objaśniające charakteryzujące tego osobnika.

Składowe wektora $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$ określające kombinację liniową $\mathbf{x}_i^T\boldsymbol{\beta}$ są wspólne dla $i = 1, \ldots, N$ i stanowią parametery uogólnionego modelu liniowego.

Tym samym rozkład obserwowanej wartości Y_i, $i = 1, \ldots, N$, jest opisany funkcją gęstości postaci:

$$f_i(y) = f(y_i; \mathbf{x}_i, \boldsymbol{\beta}).$$

Jeżeli funkcja $f(.)$ należy do rodziny wykładniczej, to **estymacja parametru $\boldsymbol{\beta}$** jest stosunkowo prosta. Szczegółowe wzory – wraz z ich wyprowadzeniem – dla estymatorów największej wiarygodności można znaleźć w [3], str 39–41, lub [7], str 21–40.

Ostateczny wynik jest taki, że estymatory te można wyznaczyć w sposób iteracyjny za pomocą iteracyjnej metody najmniejszych kwadratów (Iterative Reweighted Least Squares). W kroku $(k+1)$ tego algorytmu rozwiązuje się układ równań liniowych, a rozwiązanie tego układu (oznaczane dalej jako $\boldsymbol{\beta}^{(k+1)}$) dostarcza kolejnego przybliżenia dla wektora $\boldsymbol{\beta}$ dającego maksimum funkcji wiarogodności [3], str 39–41.

W $k+1$-tej iteracji rozwiązujemy następujący układ równań (symbole y_i oznaczają zaobserwowane realizacje zmiennych losowych Y_i): $\{irls\}$

$$\mathbf{X}^T\mathbf{W}\mathbf{X}\boldsymbol{\beta}^{(k+1)} = \mathbf{X}^T\mathbf{W}\mathbf{z}, \tag{10.14}$$

gdzie

$\mathbf{X}_{N\times(d+1)}$ jest tablicą danych poszerzoną o kolumnę jedynek,

$\boldsymbol{\beta}^{(k+1)}$ o wymiarach $(d+1)\times 1$ jest szukanym wektorem parametrów, otrzymamy go rozwiązując układ równań (10.14),

$\mathbf{z} = \mathbf{z}(k) = (z_1, \ldots, z_N)^T$ jest wektorem pomocniczym, powstałym z rozwinięcia 1-go rzędu (1st order) funkcji $g(y) \approx g(\mu) + (y-\mu)g'(\mu) = \eta + (y-\mu)\partial\eta/\partial\mu$:

$$z_i = \hat{\eta}_i + (y_i - \hat{\mu}_i)\frac{\partial\eta_i}{\partial\mu_i} = \sum_{j=1}^{d+1} x_{ij}\beta_j^{(k)} + (y_i - \mu_i)\frac{\partial\eta_i}{\partial\mu_i},$$

z wartościami μ_i i $\partial\eta_i/\partial\mu_i$ wyznaczonymi w punkcie $\boldsymbol{\beta}^{(k)}$,

$\mathbf{W} = \mathbf{W}(k)$ jest macierzą diagonalną $diag\{w_{ii}\}$ [6] rozmiaru $N \times N$, której elementy stanowią wariancję rozwinięcia $g(y)$ wyznaczoną przy założeniu. że η_i i μ_i są ustalone i znane (wyznaczone w poprzedniej, k-tej, iteracji):

$$w_{ii}^{-1} = var(y_i)(\frac{\partial\eta_i}{\partial\mu_i})^2,$$

Układ równań (10.14) przypomina układ równań normalnych dla modelu liniowego (np. modelu regresji wielokrotnej), jednak – wobec faktu zależności $\mathbf{W}$ i $\mathbf{z}$ od aktualnej wartości $\boldsymbol{\beta}^{(k)}$ – musi być rozwiązywany iteracyjnie, startując z jakiejś wartości początkowej $\boldsymbol{\beta}^{(0)}$

Teoria GLM pozwala nam znaleźć nie tylko estymatory $\hat{\boldsymbol{\beta}}$ paramaterów $\boldsymbol{\beta}$, ale również ich wariancje i kowariancje. Mamy, por. Dobson [3], str 63:

$$cov(\hat{\boldsymbol{\beta}}) = \mathcal{I}^{-1},$$

gdzie $\mathcal{I}$ jest wartością oczekiwaną hesjanu (**macierzą informacji**) o elementach

$$\mathcal{I}_{hk} = E[\{\partial^2 \log \mathcal{L}/\partial\beta_h\partial\beta_k\}],\ h,k = 1, \ldots, d+1$$

wyznaczonego z logarytmu wiarygodności $\mathcal{L}$ w punkcie $\hat{\boldsymbol{\beta}}$. Pokazuje się, że $\mathcal{I} = \mathbf{X}^T\mathbf{W}\mathbf{X}$.

[6] elementy w_{ii} oznaczają tutaj wagi kolejnych wierszy tablicy danych X i nie mają nic wspólnego z wagami sieci neuronowej

10.8.3 Zastosowanie GLM w klasyfikacji do 2 klas

Obserwowana zmienna losowa Y_i ma rozkład binarny (Bernoulliego), z prawdopodobieństwem *sukcesu* π_i, gdy:

$$Pr\{Y_i\} = \begin{cases} 1, & \text{z p-stwem } \pi_i, \\ 0, & \text{z p-stwem } 1-\pi_i. \end{cases}$$

W zagadnieniach klasyfikacyjnych w przypadku klasyfikacji do dwu grup parametr π_i jest interpretowany jako oczekiwane prawdopodobieństwo sklasyfikowania osobnika nr i do klasy o etykiecie '1'; natomiast wyrażenie $1-\pi_i$ oznacza zaklasyfikowanie osobnika nr i do klasy komplementarnej (oznaczonej umownie etykietą '0' lub '2').

Wskaźnik 'i' przy p-stwie π_i oznacza możliwość, że każdy osobnik (nr 'i') mógł zostać wylosowany z rozkładu binarnego (Bernoulliego) o innym prawdopodobieństwie sukcesu.

Wiadomo, że w rozkładzie binarnym wartość oczekiwana zmiennej losowej Y_i jest równa π_i. Oznaczmy ogólnie

$$\mu_i = E(Y_i) = \pi_i.$$

Tak więc każdy element ciągu $\{Y_i\}$ może mieć inną wartość oczekiwaną. Wartość ta może zależeć od pewnych dodatkowych zmiennych, nazywanych zmiennymi objaśniającymi. Zmienne te są dane, dla każdego elementu ciągu $\{Y_i\}$, w postaci wektora $\mathbf{x}_i = (x_{i1}, \ldots, x_{id}, 1)^T$.

Niech η_i oznacza kombinację liniową rozważanych zmiennych objaśniających:

$$\eta_i = \mathbf{x}_i^T \boldsymbol{\beta}, \qquad i = 1, \ldots, N,$$

gdzie $\boldsymbol{\beta}$ oznacza wektor nieznanych współczynników (w dalszym ciągu postaramy się je wyestymować na podstawie przyjętego modelu i zaobserwowanych wartości $y_1, \ldots, y_N$).

Przyjmijmy, że wartość oczekiwana Y_i, czyli parametr π_i zależy tylko od kombinacji liniowej $\eta_i = \mathbf{x}_i^T \boldsymbol{\beta}$, a zależność ta jest postaci {wzór *flogi*}

$$E(Y_i) = \mu_i = \pi_i = \frac{1}{1 + exp\{-\eta_i\}} = \frac{1}{1 + exp\{-\mathbf{x}_i^T \boldsymbol{\beta}\}}\,. \tag{10.15}$$

Chcąc stosować metody GLM, przyjmujemy jako **funkcję-łącznik** (link-function) funkcję **logit** {wzór *logit*}:

$$g(\mu_i) = \log(\frac{\pi_i}{1-\pi_i}) = \eta_i = \mathbf{x}_i^T \boldsymbol{\beta}. \tag{10.16}$$

Pokażemy teraz, jak zdefiniowana funkcja *logit* zależy od wektora zmiennych objaśniających $\mathbf{x}_i$ i od wektora parametrów $\boldsymbol{\beta}$. Mamy ($M = [1 + exp(-\eta_i)]$, jest to mianownik we wzorze 10.15)

$$g(\mu_i) = logit(\mu_i) = \log(\frac{\pi_i}{1-\pi_i}) = \log(\frac{1/M}{exp(-\eta_i)/M}) = \eta_i = \mathbf{x}_i^T \boldsymbol{\beta}.$$

Tak więc wprowadzona *logi* spełnia warunki funkcji-łącznika w uogólnionym modelu liniowym.

Pokażemy teraz, że **funkcją odwrotną do funkcji** *logit* **jest funkcja logistyczna**.

Mamy bowiem

$$\log(\frac{\pi_i}{1-\pi_i}) = \eta_i \quad \Leftrightarrow \quad \log(\frac{1-\pi_i}{\pi_i}) = -\eta_i \quad \Leftrightarrow \quad 1-\pi_i = \pi_i \exp\{-\eta_i\}.$$

Stąd – wyznaczając z ostatniej równości powyżej – wartość π_i otrzymujemy $\pi_i = 1/[1 + exp\{-\eta_i\}]$, czyli wzór (10.15).

Tym samym możemy do estymacji parametrów $\boldsymbol{\beta}$ stosować ogólne wzory teorii GLM, w szczególności wzór (10.14).

Dla rozkładu binarnego mamy:

$$E(Y_i) \equiv \mu_i = \pi_i, \quad Var(Y_i) = \pi_i(1-\pi_i). \tag{10.17}$$

Wyznaczanie wag w układzie IRLS (wzór 10.14)

Pochodna $\partial\mu_i/\partial\eta_i$ potrzebna do wyznaczenia wag układu równań IRLS jest wyznaczana w prosty sposób jako (wykorzystujemy, że $\mu_i \equiv \pi_i$

$$\frac{\partial\mu_i}{\partial\eta_i} = \frac{\partial}{\partial\eta_i}\frac{1}{1+\exp\{-\eta_i\}} = -\frac{-exp\{-\eta_i\}}{(1+\exp\{-\eta_i\})^2} = \frac{exp\{-\eta_i\}}{1+\exp\{-\eta_i\}}\,\frac{1}{1+\exp\{-\eta_i\}} = (1-\pi_i)\pi_i.$$

Wiadomo również, że $Var\{y_i\} = \pi_i(1-\pi_i)$ (własność rozkładu binarnego). Otrzymujemy wtedy jako wagi [7] układu równań (10.14)

$$w_{ii} = \frac{1}{Var(Y_i)}\Big(\frac{\partial\mu_i}{\partial\eta_i}\Big)^2 = \frac{1}{\pi_i(1-\pi_i)}\cdot\pi_i^2(1-\pi_i)^2 = \pi_i(1-\pi_i).$$

Wyznaczanie zmiennej pomocniczej z w układzie IRLS (wzór 10.14)

Obliczamy najpierw, korzystając z wzoru 10.16

$$\frac{\partial\eta_i}{\partial\mu_i} = \frac{1}{\mu_i} + \frac{1}{1-\mu_i} = \frac{1}{\mu_i(1-\mu_i)} = \frac{1}{\pi_i(1-\pi_i)}.$$

Stąd otrzymujemy wartości zmiennej pomocniczej **z** jako:

$$z_i = a_i + (y_i-\mu_i)\cdot\frac{\partial\eta_i}{\partial\mu_i} = a_i + (y_i-\mu_i)\frac{1}{\pi_i(1-\pi_i)}, \quad \text{gdzie } a_i = \sum_{j=1}^{d+1} x_{ij}\beta_j^{(k-1)}.$$

W sieciach neuronowych (sieć GLM) podstawiamy: $y_i \leftarrow t^i$, $\pi_i \leftarrow y^i = p^i$, wynik sieci dla i-tego osobnika (obliczony przez `glmfwd` wywoływaną we wnętrzu `glmerr`).

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] Dobson A. J., *An Introduction to Generalized Linear Models.* Chapman and Hall, London, New York. 1990, 2nd edition 2002.

[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/

[6] S. Dossou-Gbété, W. Tinsson, Factorial experimental designs and generalized linear models. SORT 29(2), July-December 2005, 249–368.

[7] P. McCullagh, J.A. Nelder, Generalized Linear Models. Chapman & Hall, 1995, 2nd Ed., Reprint 1995.

[7] nie mylić z wagami sieci neuronowej!