{plik wk10.tex 14.12.2005, popr. 18.12. i 06.01.2006}

10 Wizualizacja wielozmiennych danych

1. Wizualizacja Umat i Umati w sieciach Kohonena, por. par.7.4.1 (somtoolbox2).
2. Scatterplot matrix.
3. Metoda Sammona (somtoolbox2).
4. GTM, Generative Topographic Mapping (Netlab).
5. Parallel Coordinate plot, Wykres we współrzędnych równoległych (CSToolbox, funkcja csparrallel, modyfikacja: paraIris.m na dysku Z, wymaga modyfikacji do indywidualnych danych).
6. Kanoniczne funkcje dyskryminacyjne: funkcje cand2 i candK na dysku Z, wymagają adaptacji do indywidualnych danych.
7. Neuroscale (Netlab) – nie omawiamy.

Wiele przykładów wizualizacji wielozmiennych danych można zobaczyć w modułach demonstracyjnych SOM_DEMO1, SOM_DEMO2, SOM_DEMO3 i SOM_DEMO4 pakietu Somtoolboxb2. I tak:

SOM_DEMO1 – Basic properties. Pokazuje siatki mapy, trenowanie, znajdywanie BMU (best matching unit jest to najbliższy prototyp do zadanego punktu), określanie som_quality, czyli jakość reprezentacji przez wskaźniki q_e– quantization error, i q_t – topological error.

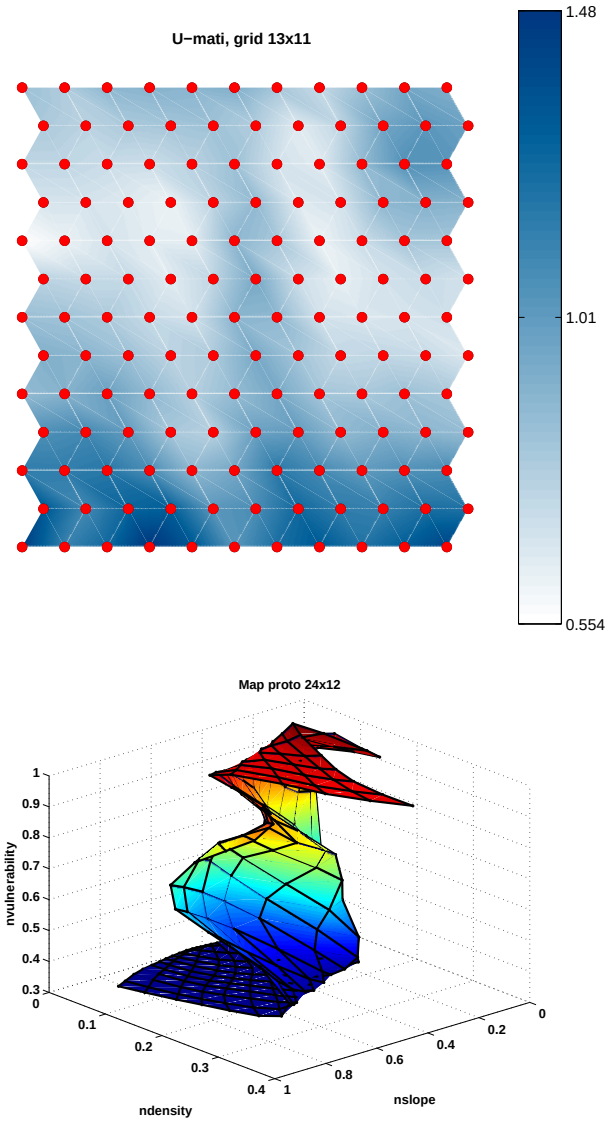
SOM_DEMO2 – Basic usage of SOM Toolbox. Funkcje: som_make, som_show, som_show_add, som_grid, som_autolabel, som_hits.

SOM_DEMO3 – Visualization, part I . Funkcje: som_show, som_grid, som_show_add, som_cplane, som_pieplane, som_barplane, som_plotplane,
pcaproj, cca (Curvilinear Component Analysis, nieliniowa PCA), sammon,
som_umat, som_hits.

SOM_DEMO4 – Visualisation, part II . Wyodrębnianie skupień: k-means, scatterplot-matrix na danych i prototypach, kmeans_clusters (ile jest skupień), modelowanie rozkładu danych za pomocą mieszanin rozkładów gaussowskich, algorytmy LVQ1 i LVQ3 znajdujące prototypy zadeklarowanych klas.

10.1 Metoda Umat i Umati

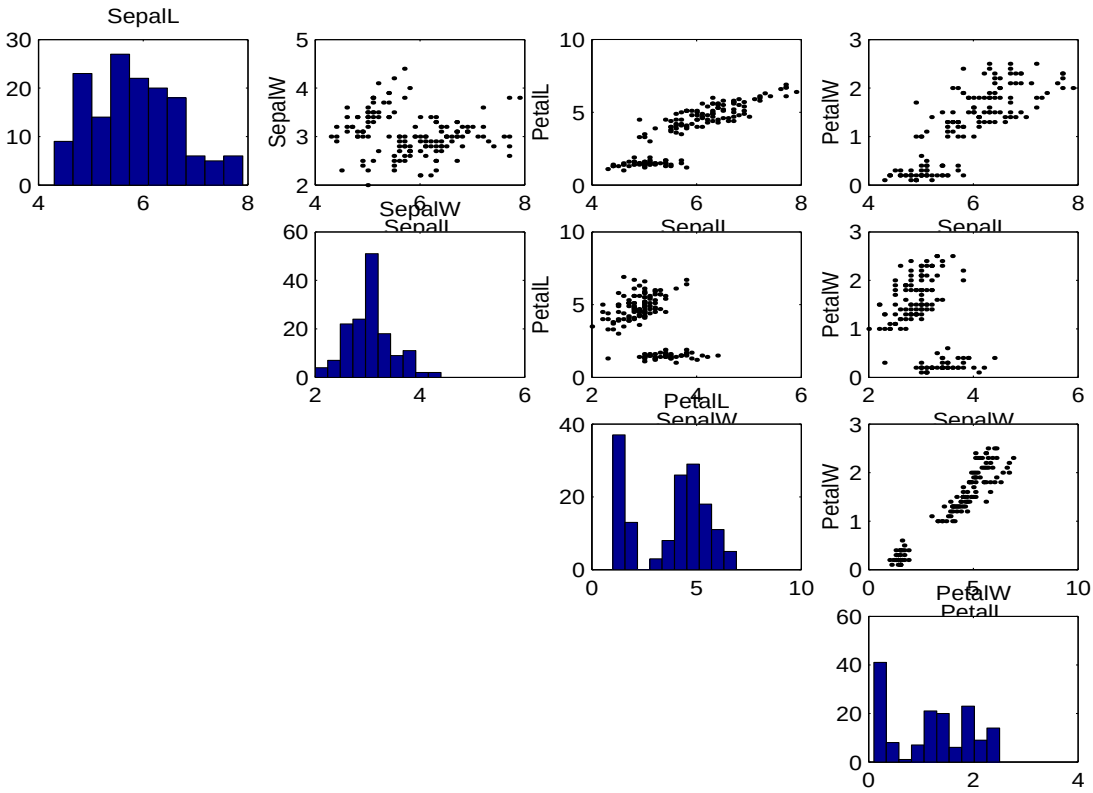
Została opisane w sekcji 9 poświęconej mapom Kohonena).



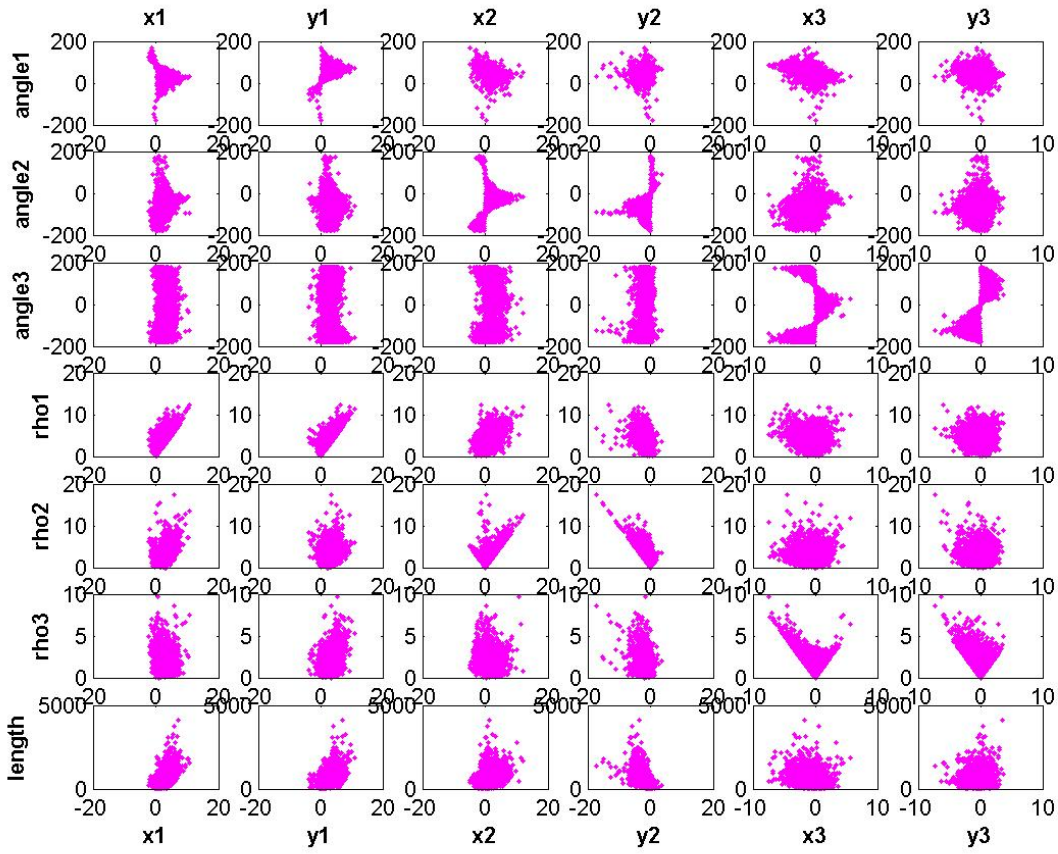
Rysunek 10.1: ***Góra***: *Mapa Kohonena otrzymana przez funkcję* `som_show` *z argumentem 'umati'. Pokazuje za pomocą palety kolorów odległości między prototypami danych. Zaznaczone punkty odpowiadają wektorom referencyjnym na mapie.* ***Dół***: *Dane 'Erosion'. Lokalizacja prototypów w 3D z zaznaczonymi połączeniami wynikającymi z sąsiedztwa na mapie Kohonena. Uczenie rozpoczęto z regularnej siatki prototypów ułożonych na płaszczyźnie* $< PC1, PC2 >$. *Pliki gtmPara5/map2.eps i Plik gtmPara5/koh1b.eps*

10.2 Wykresy typu 'scatterplot-matrix'

```
% z som_grid
 %  wykres typu scatterplot-matrix
  %     Here are the histograms and scatter plots of the four variables.
fh1=figure; k=1;
for i=1:4,
  for j=1:4,
    if i==j,
      subplot(4,4,k); hist(sD.data(:,i)); title(sD.comp_names{i})
    elseif i<j,
      subplot(4,4,k);
      plot(sD.data(:,i),sD.data(:,j),'k.')
      xlabel(sD.comp_names{i})
      ylabel(sD.comp_names{j})
    end
    k=k+1;
  end
end;
drawnow;
```



Rysunek 10.2: *Wykres typu scatterplot-matrix dla 4 cech danych* iris, *otrzymany za pomocą skryptu job_grid.m, plik gtmpara5/fig9_3.eps*



Rysunek 10.3: *Wykres typu scatterplot-matrix dla danych 'geny', pokazujący zależności między wybranymi parami cech. Tablica danych jest rozmiaru* 3300×13*; wiersze odpowiadają genom drożdży. Każdy gen jest scharakteryzowany* $d = 13$*-oma cechami. plik scatmgmj.jpg*

10.3 Odwzorowanie Sammona

Omówienie algorytmu odwzorowania Sammona

Mamy n wektorów d-wymiarowych $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^T$, $(i = 1, 2, ..., n)$, interpretowanych jako zbiór punktów leżących w przestrzeni Euklidesowej R^d. Punkty te chcemy odwzorować do przestrzeni M-wymiarowej $(M = 2, 3)$; odpowiednie odwzorowania (rzuty) będziemy oznaczać jako $\mathbf{y}_i = (y_{i1}, \ldots, y_{iM})^T$. Tak więc dla każdego punktu $\mathbf{x}_i$ należy znaleźć jego rzut $\mathbf{y}_i$:

$$\mathbf{x}_i \to \mathbf{y}_i\,, \quad \text{gdzie} \;\; \mathbf{x}_i \in R^d,\; \mathbf{y}_i \in R^M.$$

W obu przestrzeniach (tj. R^d i R^M) określamy pojęcie odległości $d(i, j) = d_{ij}$ między punktami o numerze i oraz j leżącymi w tej przestrzeni. Do określenia odległości można zastosować dowolną metrykę, w szczególności euklidesową. W przypadku przyjęcia tej ostatniej, odległość między dwoma punktami definiuje się jako pierwiastek kwadratowy z sumy kwadratów różnic liczonych po wszystkich składowych tych punktów.

Wprowadźmy następujące oznaczenia:

$D^*_{ij} = D(\mathbf{x}_i, \mathbf{x}_j)$ – odległości między punktami w przestrzeni R^d,
$D_{ij} = D(\mathbf{y}_i, \mathbf{y}_j)$ – odległości między odpowiednimi rzutami w przestrzeni R^M,

gdzie

$$D_{ij} = \sqrt{\sum_{s=1}^{M} (y_{is} - y_{js})^2}, \quad D^*_{ij} = \sqrt{\sum_{s=1}^{d} (x_{is} - x_{js})^2}\,.$$

Należy tak wyznaczyć punkty $\mathbf{y}_i$, aby zminimalizować funkcję błędu E (zdefiniowaną w postaci $M * n$ niewiadomych y_{is}

$$E = \frac{1}{c} \sum_{i<j}^{n} \left| D^*_{ij} - D_{ij} \right|^2 / D^*_{ij}, \qquad \text{gdzie} \;\; c = \sum_{i<j}^{n} D^*_{ij}. \tag{10.1}$$

W minimalizacji funkcji błędu E określonej wzorem (10.1) Sammon zastosował iteracyjną metodę optymalizacyjną Newtona, uproszczoną do postaci

$$y_{pq}(k+1) = y_{pq}(k) - \eta \Delta_{pq}(k), \tag{10.2}$$

w której η jest współczynnikiem uczenia (najczęściej z przedziału $[0.3; 0.4]$), k oznacza numer iteracji, natomiast $\Delta_{pq}(k)$ jest ilorazem odpowiedniej składowej gradientu i diagonalnego składnika hesjanu – wyznaczonych w k-tej iteracji:

$$\Delta_{pq}(k) = \frac{\delta E}{\delta y_{pq}} \;/\; |\frac{\delta^2 E}{\delta^2 y^2_{pq}}|$$

Przy definicji funkcji błędu w postaci (10.1) odpowiednie składowe gradientu i hesjanu sa dane wzorami: (wzory przepisane z książki: Osowski [3], str 267–268, na odpowiedzialność autora książki)

$$\frac{\delta E}{\delta y_{pq}} = -\frac{2}{c} \sum_{j=1, j\neq p}^{n} [\frac{D^*_{pj} - D_{pj}}{D_{pj} D^*_{pj}}][y_{pq} - y_{jq}], \tag{10.3}$$

$$\frac{\delta^2 E}{\delta^2 y^2_{pq}} = -\frac{2}{c} \sum_{j=1, j\neq p}^{n} \frac{1}{D_{pj} D^*_{pj}} [(D^*_{pj} - D_{pj}) - \frac{(y_{pq} - y_{jq})^2}{D_{pj}} (1 + \frac{D^*_{pj} - D_{pj}}{D_{pj}})]. \tag{10.4}$$

Odwzorowanie Sammona jest odwzorowaniem nieliniowym punktów z przestrzeni R^d na odpowiednie ich 'rzuty', tj. punkty leżące w przestrzeni R^M.

Przykład obliczeń na danych 'iris'

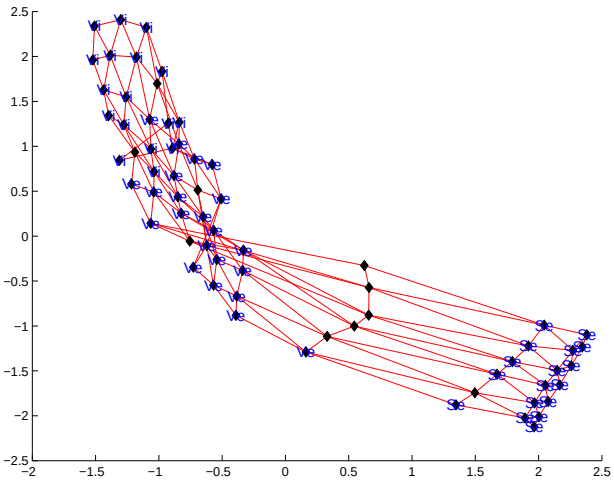
Obliczenia związane w odwzorowaniem Sammona można wykonać za pomocą funkcji sammon znajdującej się w pakiecie somtb2. Odwzorowaniu mogą podlegać zarówno wektory danych zapamiętane w tablicy D lub w strukturze **sD** typu data_struct , jak również wektory wagowe (prototypy) – zapamiętane w strukturze som_struct i odgrywające rolę reprezentantów całego zbioru danych.

Przykładowe wywołania procedury:
Y = sammon(D,2); % projection to 2-dim space,
Y = sammon(sM,3); % projects the codebook vectors,
Y = sammon(sM,3,[],[],[],Md); % uses distance matrix Md

Po obliczeniu tablicy projekcji Y może nastąpić wizualizacja graficzna przetransformowanych danych znajdujących się w tablicy Y np. za pomocą zwykłej funkcji plot lub też za pomocą funkcji som_grid.

Przykładowe wywołanie funkcji som_grid:
som_grid(sM,'Coord',Y); lub też, z bardziej wyspecyfikowanymi parametrami:
som_grid(sM,'Coord',sammon(sM,2),'LineColor','r','Label',sM.labels, ...
'labelColor','b', 'Marker','d', 'MarkerColor', 'k')

Otrzymamy wtedy rysunek 10.4. Rysunek ten pokazuje nie tylko odwzorowane punkty (które w tym przypadku są wektorami kodowymi danych) ale również sieć powiązań między tymi punktami. Sieć ta odpowiada ułożeniu punktów na mapie Kohonena.



Rysunek 10.4: *Wektory kodowe (prototypy) danych 'Iris' odwzorowane metodą Sammona na dwu-wymiarową płaszczyznę* $< y1, y2 >$. *Wektory kodowe zostały obliczone ze znormalizowanych danych. Wykres wykonano za pomocą funkcji* som_grid*; powiązania między punktami odpowiadają sąsiedztwu między wektorami referencyjnymi na mapie Kohonena, plik samm2.eps*

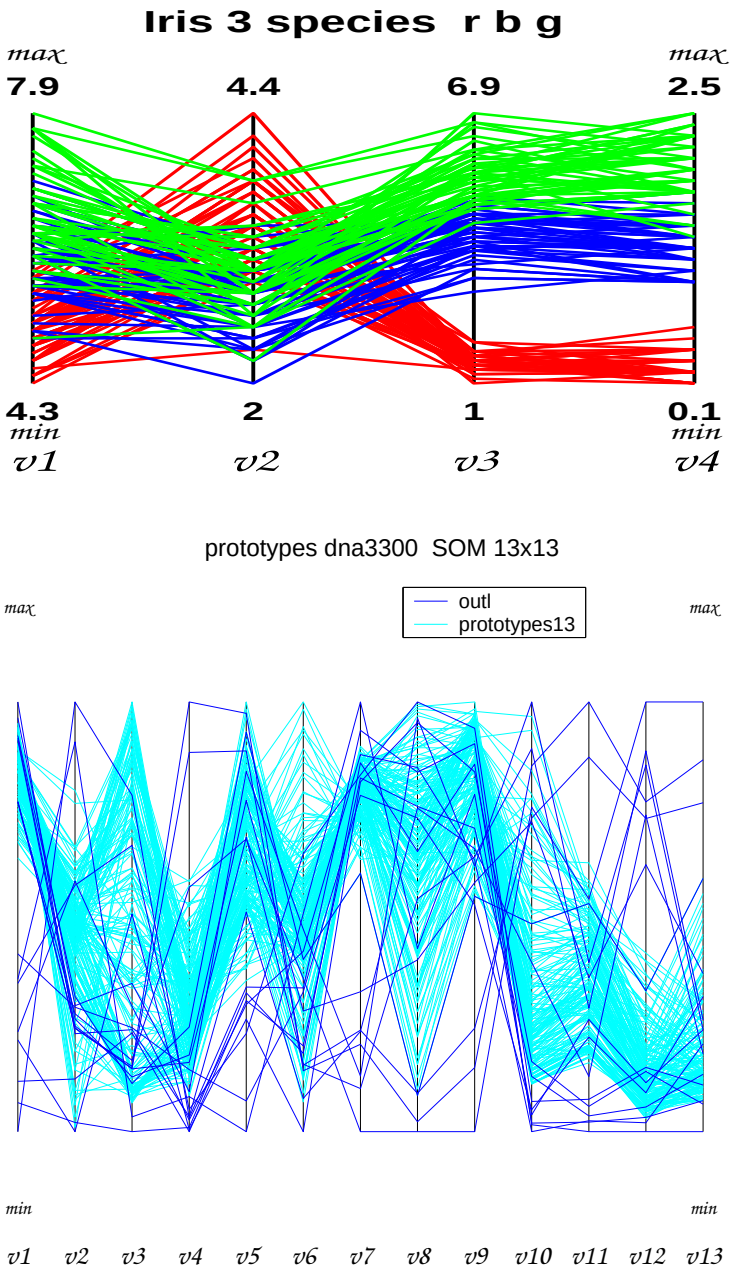
Na rysunku 10.4 obserwujemy zjawisko *skręcania się* siatki powiązań między punktami. Obraz taki jest dość częstym zjawiskiem przy stosowaniu odwzorowań Sammona. Wynika on z trudności otrzymania obrazu dwuwymiarowego który ma pokazać odległości między punktami w przestrzeni wielowymiarowej, a więc w przestrzeni o nieporównywalnie bogatszych możliwościach.

Czasami pomaga wystartowanie z innej mapy, np. uzyskanej z innego przybliżenia początkowego przy uczeniu mapy, lub innego układu neuronów na mapie.

A oto skrypty, za pomocą których wykonano rysunek 10.4: Skrypty te wykonywano na danych iris.data znajdujących się w pakiecie somtb2. Każdy wektor danych ma tam przypisaną długą etykietę (np. 'Iris Versicolor'). W skrypcie etykiety poszczególnych kwiatów (wektorów danych) uległy skróceniu do dwu-literowych.

```
%% -------------------- job_samm.m --------------------------------
% make the data, shorten labels,  make the SOM
sD = som_read_data('iris.data'); sD = som_normalize(sD,'var');
sD = som_label(sD, 'replace'. [1:50], 'Se');
sD = som_label(sD, 'replace'. [51:100], 'Ve');
sD = som_label(sD, 'replace'. [101:150], 'Vi');
sM = som_make(sD); sM = som_autolabel(sM,sD,'vote');
%sM = som_make(sD); sM = som_autolabel(sM,sD,'vote');
% visualization  of codebook vectors by Sammon projection
% Ps=sammon(sM,3);
som_grid(sM,'Coord',sammon(sM,3),'marker','none',...
         'Label',sM.labels,'labelcolor','k');
%------------- end job_samm.m ---------------------------------------
```

10.4 Wykresy typu 'parallel coordinate plot'



Rysunek 10.5: *Parallel coordinate plot,* ***góra****: dane 'Iris', r(red)- Setosa, b(blue) - Versicolor, g(green) - Virginica;* ***dół****: dane 'geny', pokazano odstające obserwacje na tle* 13×13 *prototypów, pliki plik paraIris.eps i para13ss.eps*

Wykresy te zostały sporządzone funkcją para.m, modyfikacją analogicznej funkcji znajdującej się w pakiecie CSTool in Matlab [8] (CS – Computational Statistics).

10.5 GTM, Generative Topographic Mapping

10.5.1 Wprowadzenie

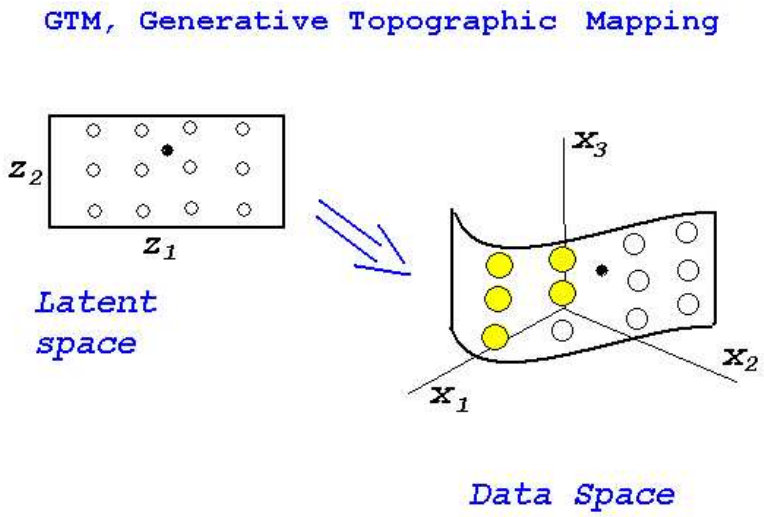
Metoda map Kohonena, mimo iż szeroko stosowana w praktyce, spotyka się z następującymi krytycznymi uwagami (por. [6], str. 243):

1. Brak funkcji kosztu.
2. Brak ogólnego dowodu o zbieżności stosowanego algorytmu uczenia.
3. Brak teoretycznych podstaw co do wyboru parametrów η i $\mathcal{N}_c$ potrzebnych przy trenowaniu sieci.
4. Brak modelu opisującego gęstość rozkładu rozważanych danych.

Metoda GTM, zaproponowana przez Bishopa i Svensena ([5], [6]) nie podlega tym krytycznym uwagom.

Idea metody GTM (Generative Topographic Mapping) jest przedstawiona na rysunku 10.6.

Obserwowane przez nas dane, czyli zmienne (na rysunku zmienne x_1, x_2, x_3) powstały jako wynik generowania przez zmienne ukryte (na rysunku zmienne z_1, z_2) z dodatkiem pewnego błędu powstałego przy mierzeniu tych cech, ewntl. dodatkowego błędu wynikającego z nieuwzględnienia wszystkich czynników ukrytych (pominięcia niektórych czynników mających znaczny udział przy generowaniu obserwowanych zmiennych).



Rysunek 10.6: *Idea metody GTM. Obserwowane przez nas dane '*$\mathbf{x}$*' – ulokowane w przestrzeni danych* $\mathcal{D}$ *(*DATA SPACE*) o* d *wymiarach – zostały wygenerowane przez nieobserwowane przez nas zmienne '*$\mathbf{z}$*' tworzące tzw. przestrzeń zmiennych ukrytych* $\mathcal{L}$ *(*LATENT SPACE*) o* q *wymiarach,* $q < d$*. Na rysunku* $d = 3$ *a* $q = 2$*. Punkty* $\mathbf{x}$*, z dokładnością do błędu pomiaru, leżą na* q*-wymiarowej nieliniowej rozmaitości zanurzonej w* R^d*. Naszym celem jest znalezienie odwzorowania* $\mathbf{z} \rightarrow \mathbf{x}$*. plik sub10/gtmIdea1reduced.ps*

Tak więc mamy przestrzeń zmiennach obserwowanych $\mathcal{D}$ – DATA SPACE i przestrzeń zmiennych ukrytych $\mathcal{L}$ – LATENT SPACE – co pokazano na rysunku. Punkty należące do przestrzeni danych oznaczamy symbolem $\mathbf{x}$ – są one d–wymiarowe: $\mathbf{x} \in R^d$. Punkty należące do przestrzeni zmiennych ukrytych oznaczamy symbolem $\mathbf{z}$; są one q–wymiarowe: $\mathbf{z} \in R^q$, przy czym na ogół $q << d$.

Najbardziej interesujące są przypadki, gdy $q = 1$ lub $q = 2$ (te przypadki są zaimplementowane w Netlabie).

Metoda GTM konstruuje odwzorowanie

$$\mathbf{y} = \mathbf{y}(\mathbf{z}; \mathbf{W}, \sigma) \tag{10.5}$$

mapujące punkty z przestrzeni $\mathcal{L}$ na punkty w przestrzeni $\mathcal{D}$. Odwzorowanie (10.5) jest charakteryzowane parametrami $\mathbf{W}$ i σ [1]; wynikają one z modelu probabilistycznego opisującego rozkłady gęstości $p(\mathbf{z})$, $\mathbf{z} \in \mathcal{L}$, $p(\mathbf{x}|\mathbf{z}, \mathbf{W})$, $\mathbf{x} \in \mathcal{D}$, oraz $p(\mathbf{x}|\mathbf{W}, \sigma)$ i pewnych dodatkowych założeń, przyjętych przez autorów metody.

10.5.2 Model probabilistyczny

Korzystamy z modelu Bayesowskiego opisującego 'przyczyny' i 'skutki'. Przyczyną są wartości 'z' zmiennych ukrytych. Skutkami są obserwowane wartości 'x'.

Pytanie: jeśli zaobserwowaliśmy daną wartość 'x', to jak możemy dotrzeć do wartości 'z' które były 'przyczyną' zaobserwowanej danej wartości 'x'.

Odpowiadamy na to pytanie, wyznaczając odpowiednie prawdopodobieństwa (funkcje gęstości) *à priori* i *à posteriori*.

Latent space $\mathcal{L}$, $dim = q$

$p(\mathbf{z})$ gęstość prawdopodobieństwa
Przyjmując r. dyskretny z M punktów:
$p(\mathbf{z}) = \frac{1}{M} \sum_{j=1}^{M} \delta(\mathbf{z} - \mathbf{z}_j)$, (*)
gdzie $\delta(.)$ jest funkcją Diraca

Data space $\mathcal{D}$, $dim = d$

$p(\mathbf{x}|\mathbf{z}, \mathbf{W}, \sigma) \sim \mathcal{N}(\mathbf{y}(\mathbf{z}; \mathbf{W}), \sigma^2)$ – r. warunkowy
Znając p($\mathbf{z}$), możemy wycałkować (integrating out) zmienną 'z' i otrzymać rozkład bezwarunkowy
$p(\mathbf{x}; \mathbf{W}, \sigma) = \int p(\mathbf{x}|\mathbf{z}, \mathbf{W}, \sigma) p(\mathbf{z}) d\mathbf{z}$
Dla $p(\mathbf{z}) = \frac{1}{M} \sum_{j=1}^{M} \delta(z - z_j)$, znając $\mathbf{z}_1, .., \mathbf{z}_M$, daje to
$p(\mathbf{x}; \mathbf{W}, \sigma) = \frac{1}{M} \sum_{j=1}^{M} p(\mathbf{x}|\mathbf{z}_j, \mathbf{W}, \sigma)$ (**)
Rozkład (**) może być uważany jako rozkład mieszaniny o M składnikach gaussowskich z izotropową wariancją.

Znając estymatory $\mathbf{W}$ i σ potrafimy obliczyć responsibilities $R_{jn} = P(j|\mathbf{x}_n, \mathbf{W}, \sigma)$ czyli p-stwa á posteriori

Estymacja parametrów met. najw. wiarogodności

Mając N punktów próbkowych $\mathbf{x}_1, \ldots, \mathbf{x}_N$ możemy napisać *funkcję wiarogodności* (complete Likelihood):
$\mathcal{L}(\mathbf{W}, \sigma) = \sum_{n=1}^{N} \ln \{\frac{1}{M} \sum_{j=1}^{M} p(\mathbf{x}_n|\mathbf{z}_j, \mathbf{W}, \sigma)\}$
Stosujemy algorytm EM dla mieszanin

[1] odwzorowanie (10.5) może posiadać dalsze parametry, wynikające ze specyfikacji przyjętego modelu

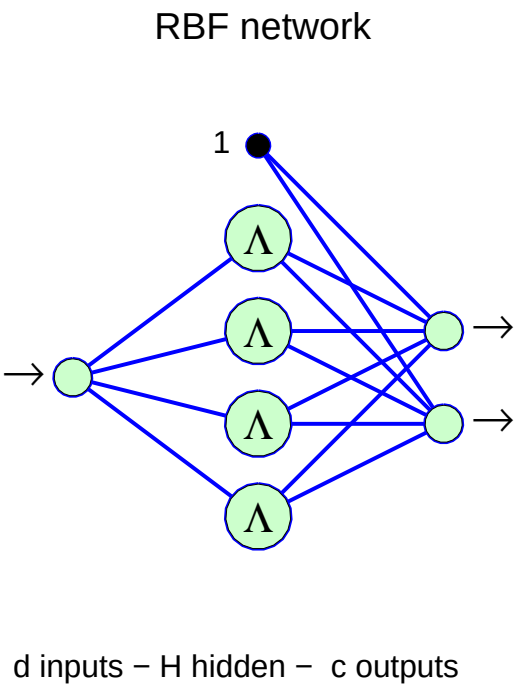
Aby powiedzieć coś bardziej detalicznego o algorytmie, należy wyspecyfikować kształt funkcji mapującej $\mathbf{z} \to \mathbf{x}$. Funkcja ta będzie oznaczana jako $\mathbf{y}(\mathbf{z}, \mathbf{W})$.

Funkcja mapująca

$\mathbf{y}(\mathbf{z}; \mathbf{W}) = \mathbf{W} \cdot \underline{\mathbf{G}}(\mathbf{z})$,

gdzie $\mathbf{W} = \mathbf{W}_{d\times(H+1)} = \begin{bmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_d^T \end{bmatrix}$, $\quad \mathbf{w}_k = [w_{k1}, \ldots, w_{k,H+1}]^T$, $k = 1, \ldots, d$

$\underline{\mathbf{G}}(\mathbf{z}) = [G_1(\mathbf{z}), \ldots, G_H(\mathbf{z}), \{G_0 \equiv 1\}]^T$ są H ustalonymi radialnymi funkcjami bazowymi + $(G_0 \equiv 1)$



Rysunek 10.7: *Sieć RBF służąca do odwzorowania punktów* $\mathbf{z} \in \mathcal{L}$ *na punkty* $\mathbf{x} \in \mathcal{D}$. *Na wejściu mamy wartości* $\mathbf{z}$, *na wyjściu – wartości* $\mathbf{y}(\mathbf{z}; \mathbf{W})$. *Na rysunku przyjęto* $q = 1, H = 4, D = 2$. *plik sub10/rbfnet2.eps*

Rysunek 10.7 pokazuje przykładową sieć RBF służącą do odwzorowania punktów $\mathbf{z}$ leżących w przestrzeni zmiennych ukrytych LATENT SPACE – na punkty $\mathbf{x}$ leżące w przestrzeni obserwowanych danych $\mathcal{D}$.

Architektura sieci:

q inputs$\to$ H hidden$\to$ d outputs

Przyjęto: $d = 1,\ H = 4,\ d = 2$.

Warstwa wejściowa i ukryta zawierają po jednym dodatkowym elemencie, odpowiadającym obciążeniu (bias-owi) funkcji wagowych.

Z założenia przestrzeń $\mathcal{L}$ jest reprezentowana przez zbiór $\{\mathbf{z}_j\}$ zawierający M wartości dyskretnych. W takiej sytuacji warstwa ukryta sieci RBF - przy znanych wagach $\mathbf{W}$ - oblicza aktywacje dla każdego punktu $\mathbf{z}_j$:

	$j=1$	$j=2$	$\ldots$	$j=M$
$h=0$	$G_0=1$	$G_0=1$	$\ldots$	$G_0=1$
$h=1$	$G_1(\mathbf{z}_1)$	$G_1(\mathbf{z}_2)$	$\ldots$	$G_1(\mathbf{z}_M)$
$h=2$	$G_2(\mathbf{z}_1)$	$G_2(\mathbf{z}_2)$	$\ldots$	$G_2(\mathbf{z}_M)$
$h=3$	$G_3(\mathbf{z}_1)$	$G_3(\mathbf{z}_2)$	$\ldots$	$G_3(\mathbf{z}_M)$
$h=4$	$G_4(\mathbf{z}_1)$	$G_4(\mathbf{z}_4)$	$\ldots$	$G_4(\mathbf{z}_M)$

10.5.3 Algorytm GTM zrealizowany w Netlabie

Podstawowe kroki algorytmu

Algorytm GTM zrealizowany w Netlabie wykonuje następujące kroki:

A Utworzenie podstawowej sieci GTM
B Utworzenie pod-sieci RBFNET i GMMNET
C Nadanie wartości początkowych sieci GTM
D Wyznaczanie wag algorytmem EM, funkcja GTMEM
E Wyznaczanie gęstości prawdopodobieństwa i ich charakterystyk: funkcje GTMPROB, GTMPOST, GTMMEAN, GTMLMODE.

Omówimy teraz po krótce te podstawowe kroki:

.....

ad **A**). Utworzenie podstawowej sieci GTM. Sieć GTM jest tworzona za pomocą funkcji:
`net=gtm(dim_latent, nlatent, dim_data,ncentres, rbfun);` [2]

Znaczenia parametrów funkcji GTM:

dim_latent:	q – wymiar $\mathcal{L}$;
nlatent:	M – liczba punktów dyskretnych w $\mathcal{L}$,
dim_data:	d – wymiar przestrzeni $\mathcal{D}$;
ncentres:	H – liczba centrów dla sieci RBF;
rbfun:	rodzaj aktywacji neuronów warstwy ukrytej sieci RBF

Pola sieci GTM utworzonej przez funkcję `net`:

type	'gtm'
nin	dimension of data space d
dim_latent:	q – wymiar $\mathcal{L}$
rbfnet	RBF network data structure
gmmnet	GMM data structure
X	sample of latent points from $\mathcal{L}$, na razie puste (=[])

Jak widać, dwa pola sieci (struktury) GTM zawierają odrębne sieci typu RBF i GMM. Zostały one utworzone za pomocą rozkazów

```
net.rbfnet = rbf(dim_latent, ncentres, dim_data, rbfunc, 'linear');
```

oraz

```
net.gmmnet = gmm(dim_data, nlatent, 'spherical');
```

Sieć `net.rbfnet` tworzy pomost między przestrzeniami $\mathcal{L}$ i $\mathcal{D}$. Sieć `net.gmmnet` (= `mix`) operuje w przestrzeni $\mathcal{D}$. Opisuje ona rozkład danych jako mieszaninę M składników Gaussowskich o sferycznej macierzy kowariancji. Rozkład ten jest wykorzystywany przez algorytm EM przy znajdowaniu estymatorów parametrów $\mathbf{W}$ i σ.

.....

[2] Pomijamy tu dalsze parametry potrzebne przy określaniu tzw. hiperparametrów modelu Bayesowskiego, którego to modelu nie omawiamy

ad **B**). Obsada pod-sieci RBFNET i GMMNET, funkcja

`mix = gtmfwd(net)`. Zadaniem funkcji jest przekazanie aktualnej zawartości pola `net.X` do pod-sieci GMM. Dzieje się to za pomocą dwóch instrukcji:

```
net.gmmnet.centres = rbffwd(net.rbfnet, net.X);
mix = net.gmmnet;
```

Pole `net.X` zawiera dane próbkowe $\mathbf{z}_j,\ j = 1, \ldots, M$ określające rozkład $p(\mathbf{z})$ w przestrzeni $\mathcal{L}$. Dane te są określane oddzielnie za pomocą funkcji `net=gtminit(...)` opisanej w następnym punkcie (C).

.....

ad **C**). Nadanie wartości początkowych sieci GTM Służy temu funkcja GTMINIT. Zadaniem tej funkcji jest dostarczenie do już utworzonej sieci `net` danych uczących (wektorów danych należących do przestrzeni $\mathcal{D}$ oraz utworzenie punktów $\mathbf{z}_i \in \mathcal{L}$, oraz wybranie z nich centrów do sieci RBF.

Nagłówek funkcji GTMINIT:
`net = gtminit(net, options, data, sampType, varargin);`

Znaczenie parametrów:
net – utworzona poprzednio sieć typu 'gtm',
options – jak w Netlabie, options(7) – patrz niżej,
data – dane uczące, rozmiaru $N \times d$,
sampType – typ siatki, jaką ma tworzyć zbiór $\{\mathbf{z}_i\} \in \mathcal{L}$. Typ może być:
'regular': wtedy trzeba zadeklarować rozmiary siatki w parametrze varargin, np. dla q=2: na miejsce 'varargin' można podstawić [10, 8], [8,4]
'uniform': punkty $\{\mathbf{z}_i\}$ są wybierane losowo (random) $\in [-1, 1]^q$
'Gaussian': punkty $\{\mathbf{z}_i\}$ są wybierane z $[(N(0, 1)/2]^q$. Aby określić automatycznie (tj. na podstawie próbki latent points parametr sigma funkcji Gaussowskiej, należy ustawić options(7)=1.

Parametr 'vargin' jest określany tylko wtedy, gdy `sampType`='regular' (należy wtedy podać `l_samplesize` – the number of latent points oraz `rbf_samp_size` – the number of RBF centres); w przeciwnym przypadku centra są określane przez funkcję RBFSETBF wywoływaną we wnętrzu funkcji.

Liczba wygenerowanych punktów próbkowych z $\mathcal{L}$ jest zapamiętana jako zmienna `nlatent` w polu `net.gmmnet.ncentres`.

Liczba centrów sieci RBF jest zapamiętana jako zmienna `nhidden`
w polu `net.rbfnet.nhidden`.

.....

ad **D**). Wyznaczanie wag algorytmem EM, funkcja GTMEM

O algorytmie EM napisano kilka książek i mnóstwo prac ... .

Algorytm ten jest zaimplementowany jako funkcja:
`[net, options, errlog] = gtmem(net, data, options)`.
Wagi są zapamiętane w `net.rbfnet.w2` i `net.rbfnet.b2`.

Błąd jest zapamiętany w options(8) jako ujemny logarytm z wiarogodności:
`-sum(log(gmmprob(mix, x)));`.

.....

ad **E**). Wyznaczanie gęstości prawdopodobieństwa i ich charakterystyk. Są to funkcje:

GTMPROB – oblicza $p(\mathbf{x}|\mathbf{z}, \mathbf{W}, \sigma)$,
GTMPOST – oblicza *responsibilities* $R_{jx} = p(j|\mathbf{x}, \mathbf{W}, \sigma)$
GTMMEAN – oblicza *posterior mean*, GTMLMODE – oblicza *posterior mode*.

Potrzebne przy wizualizacji punktów w przestrzeni $\mathcal{L}$. Naszym celem jest odwzorowanie obserwowanych (w $\mathcal{D}$) punktów $\mathbf{x}_n$ na odpowiednie punkty 'z' w $\mathcal{L}$. Jednak – posługując się teorią Bayesowską, otrzymujemy cały rozkład $p(\mathbf{z}|\mathbf{x}_n)$. Potrzebujemy jakiejś sumarycznej informacji. Może ona zostać dostarczona przez wartość oczekiwaną lub wartość modalną rozkładu $p(.|.)$.
Wartość oczekiwana $< \mathbf{z}|\mathbf{x}_n, \tilde{\mathbf{W}}, \tilde{\sigma} >$ może być myląca.
Wartość modalna jest obliczana jako $j_{max} = arg\max_j R_{jn}$.

Dla praktyka

W przedstawionym algorytmie użytkownik określa przede wszystki **liczbę** i **typ** funkcji bazowych potrzebnych do konstrukcji sieci RBF. Parametry tej sieci kontrolują gładkość funkcji mapującej $\mathcal{L} \to \mathcal{D}$. W szczególności, przy Gaussowskiej funkcji bazowej, ważny jest stosunek σ/s, gdzie s oznacza odległości (spacing) między sample points in $\mathcal{L}$. Powiększenie M, czyli liczby latent space sample points, może tylko polepszyć jakość modelu, np. przyczynić się do dokładniejszego obliczenia rozkładu $p(\mathbf{x}|\mathbf{W}, \sigma)$ – lecz wzrasta wtedy złożoność i koszt obliczeń.

Ważny jest punkt startowy. Propozycja: Rozpocząć szukanie $\mathcal{L}$ poprzez zastąpienie całej przestrzeni $\mathcal{D}$ podprzestrzenią liniową rozpiętą na q pierwszych składowych głównych.

Wtedy określamy błąd

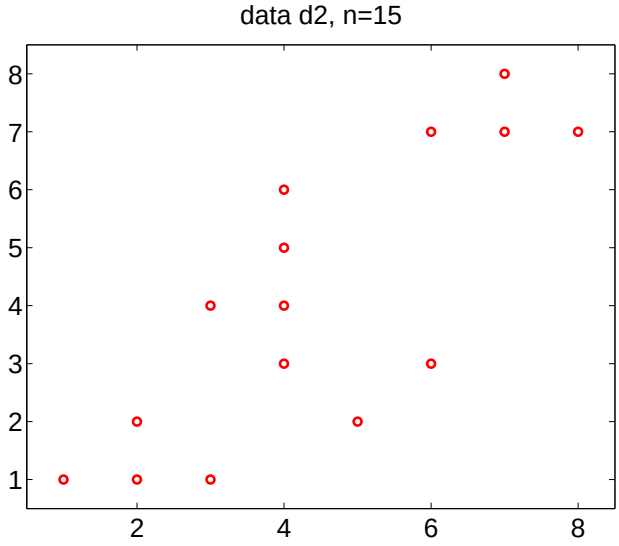
$$E = \frac{1}{2}\sum_{j=1}^{M} || \underbrace{\mathbf{W}\underline{G}(\mathbf{z}_j)}_{\mathbf{y}(\mathbf{z}_j;\mathbf{W})} - \mathbf{U}\mathbf{z}_j ||,$$

gdzie $\mathbf{U}$ zawiera q pierwszych, odpowiednio przeskalowanych wektorów własnych macierzy kowariancji obserwowanych danych.

10.5.4 Przykłady zastosowań

Przykład 1. Dane 'd2' z sekcji 7

Dane 'd2' są zawarte w tablicy wymiaru 15×2, tj. $N = 15$ osobników, charakteryzowanych przez $d = 2$ cechy.



Rysunek 10.8: *Dane d2, n=15, d=2. plik sub10/gtmMy1d2.eps*

Dalsza analiza jest wzorowana na module DEMGTM1 pakietu Netlab. Dane chcemy odwzorować na prostą ($q = 1$.

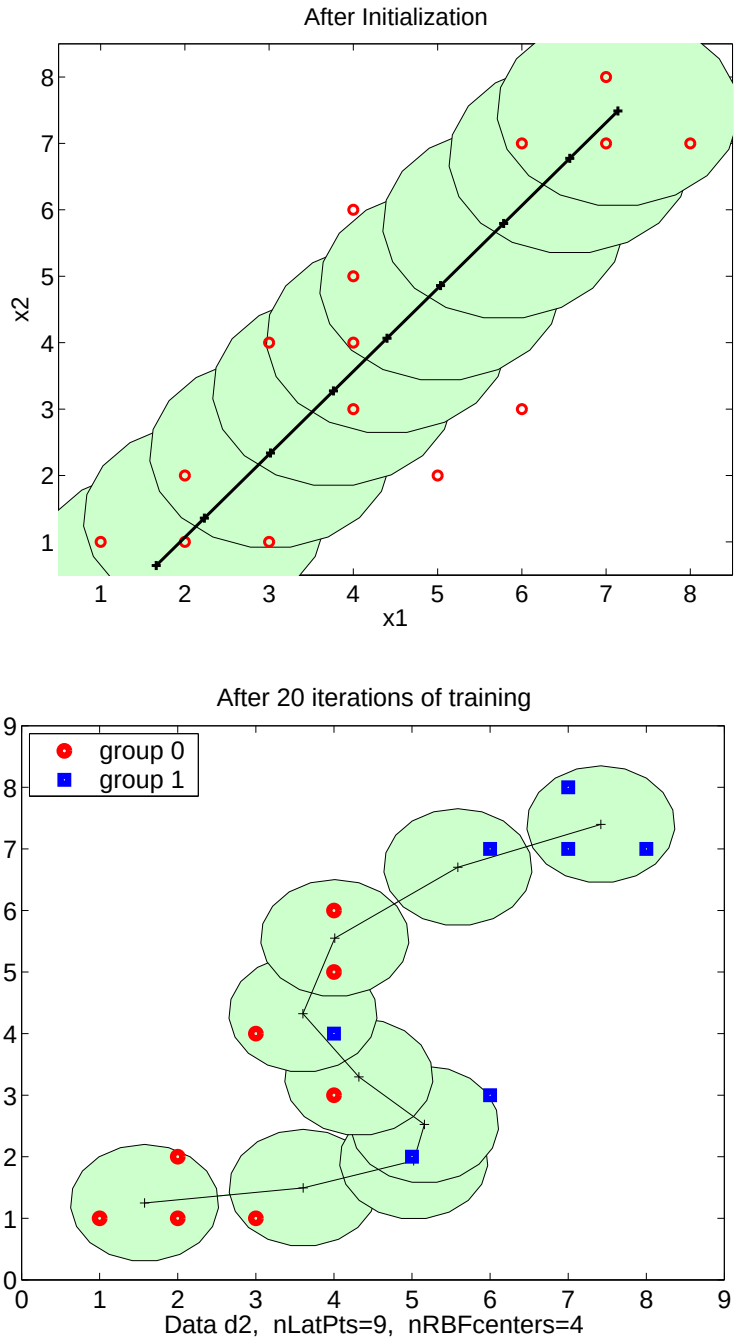
Natomiast funkcję gęstości rozkładu w $\mathcal{D}$ przybliżamy mieszaniną $M = 9$ rozkładów Gaussowskich.

Rezultat aproksymacji jest pokazany na rysunku 10.9.

Zauważmy, że znalezione odwzorowanie nie odzwierciedla zróżnicowania między grupami danych.

A oto fragmenty skryptu wykonującego obliczenia (pierwowzór: DEMGTM1):

```
%%Generate and plot (along with the data) an initial GTM model
          % Preliminaries:
numLatentPoints = 9; % = nb of mix.centers
numRbfCentres = 4;
net = gtm(1, numLatentPoints, 2, numRbfCentres, 'gaussian');
iters=20; pauz=0.4;
options = zeros(1, 18); options(7) = 1;
xcir = [0:(2*pi)/(20-1):2*pi]'; % Generate a unit circle figure
unitC = [sin(xcir) cos(xcir)];    % to be used for plotting
          % Now initialize the network and plot it
net = gtminit(net, options, data, 'regular', numLatentPoints,numRbfCentres);
mix = gtmfwd(net);
h1=figure(1); plot(mix.centres(:,1), mix.centres(:,2), 'g');
hold on;
for i=1:numLatentPoints
   c = 2*unitC*sqrt(mix.covars(1)) + [ones(20,1)*mix.centres(i,1) ...
           ones(20,1)*mix.centres(i,2)];
   fill(c(:,1), c(:,2), [0.8 1 0.8]);
end
plot(data(:,1), data(:,2), 'ro');
plot(mix.centres(:,1), mix.centres(:,2), 'k+');
plot(mix.centres(:,1), mix.centres(:,2), 'k');
hold off
           % Train the GTM and plot it along with the data
options = foptions;
options(1) = -1; % Turn off all warning messages
options(14)=iters;
for j = 1:iters
   [net, options] = gtmem(net, data, options);
   mix = gtmfwd(net);
end;
h2=figure(2);
plot(mix.centres(:,1), mix.centres(:,2), 'g');
hold on;
for i=1:numLatentPoints % tyle jest mix.centers
   c = 2*unitC*sqrt(mix.covars(1)) + [ones(20,1)*mix.centres(i,1) ...
           ones(20,1)*mix.centres(i,2)];
   fill(c(:,1), c(:,2), [0.8 1.0 0.8]);
   end
plot(data(:,1), data(:,2), 'ro');
plot(mix.centres(:,1), mix.centres(:,2), '-k+');
axis([data_min-0.5*2 data_max+0.5*2 data_min-0.5*2 data_max+0.5*2]);
title(['After ', int2str(iters),' iterations of training'])
hold off
```



Rysunek 10.9: *Dane d2, n=15, d=2, aproksymowane mieszaniną Gaussowską o M=9 składnikach.* ***Góra:*** *Punkty danych i initializacja modelu GTM/GMM dla dim_latent=1.* ***Dół:*** *Stan po zakończeniu iteracji i osiągnięta aproksymacja za pomocą mieszaniny 9-cio składnikowej. pliki sub10/gtmMy1a.eps i sub10/gtmMy1b.eps*

10.5.5 Przykład 2 – dane irys

Dane 'iris' są cztero-wymiarowe, tak więc $d = 4$. Chcemy odwzorować te dane na płaszczyźnie, czyli przyjmujemy $q = 2$. Wszystkich punktów danych jest $n = 150$.

Zakładamy regularną strukturę punktów próbkowych $\mathbf{z}_j \in \mathcal{L}$. Przyjmujemy, że punkty próbkowe mają leżeć na siatce [12 12], a centra sieci RBF na siatce [4 4]. Tak więc deklarujemy:

```
data=iris; dim_data = size(data,2);
dim_latent = 2; latent_shape = [12 12]; % wymiar i ksztalt punktów próbkowych
nlatent = prod(latent_shape); % liczba punktow probkowych
rbf_shape=[4 4]; n_rbfc = prod(rbf_shape); % ksztalt i liczba centrow sieci rbf
```

Następnie tworzymy sieć RBF i inicjujemy jej wartości:

```
net = gtm(dim_latent, nlatent, dim_data, n_rbfc, 'gaussian', 0.1);
options = foptions;
options(1) = -1;
options(7) = 1; % Set width factor of RBF
net = gtminit(net, options, data, 'regular', latent_shape, rbf_shape);
```

Teraz trenujemy utworzoną sieć 'net'. Celem trenowania jest wyznaczenie parametrów $\mathbf{W}$ i σ. Następnie wyznaczamy wartości oczekiwane (means) i wartości modalne (modes) rozkładów a posteriori $p(j|\mathbf{x}_i, \mathbf{W})$ – dla każdego obserwowanego wektora danych $\mathbf{x}_i$:

```
options = foptions;
options(14) = 50; % nb of iterations
options(1) = 0; % without printing error after each iteration, altern.  =1
[net, options] = gtmem(net, data, options);
means = gtmlmean(net, data); modes = gtmlmode(net, data);
```

Wyznaczone w ten sposób wartości średnie (lub modalne) mogą być wykreślone, np.:

```
plot(means(:,1), means(:,2),'.r');
```

W ten sposób otrzymujemy wizualizację obserwowanych wektorów danych (punktów w $\mathcal{D}$) na płaszczyźnie. Wizualizacja ta pokazuje *topologię* punktów danych – *nie* pokazuje jednak ich *odległości*.

Aby pokazać odległości – stosujemy tzw. 'Magnification factor', wynikający z analizy odwzorowania $\mathbf{y}(\mathbf{z}; \mathbf{W})$ metodami geometrii różniczkowej:

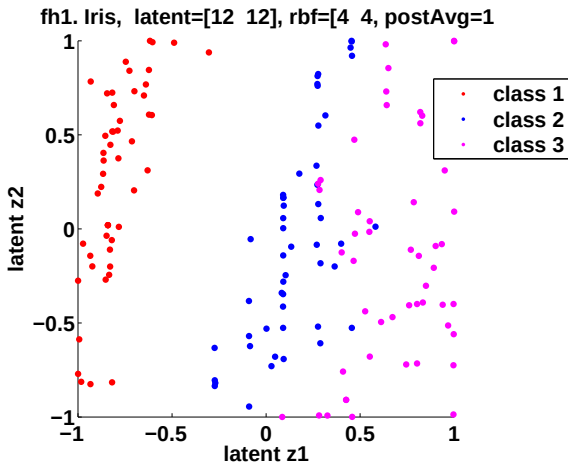
```
mags = gtmmag(net, net.X); % Reshape into grid form
Mags = reshape(mags, fliplr(latent_shape));
imagesc(net.X(:, 1), net.X(:,2), Mags);
colormap(summer); colorbar % cool, 1-bone, summer
```

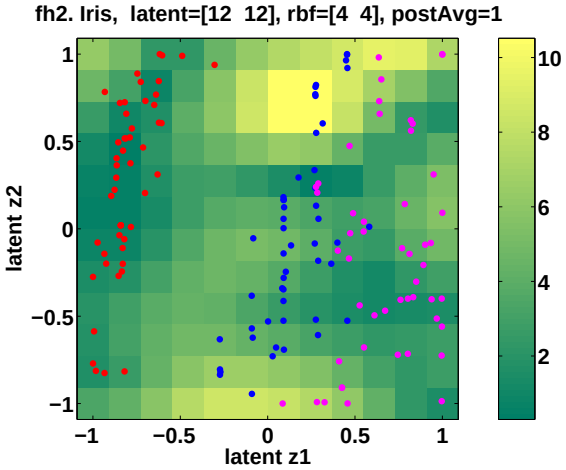
Sporządziliśmy w ten sposób rysunki, które są pokazane dalej. Przy palecie `colormap summer` ciemniejsze kolory sugerują bliskość punktów leżących w zacienionych obszarach.

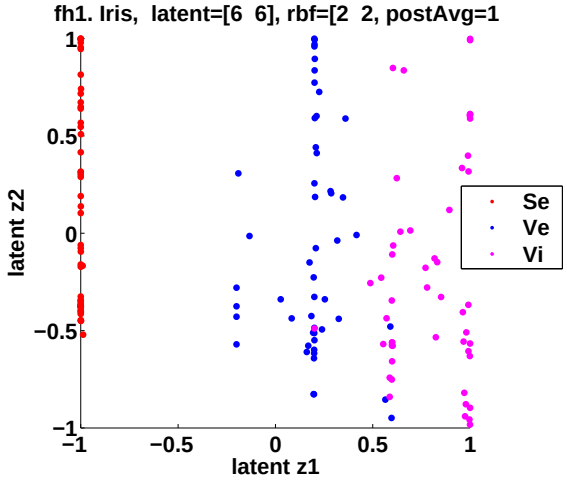
Pokazujemy 3 pary rysuków dla danych 'iris' sporządzone dla następujących wartości `latent_shape` i `rbf_shape` :

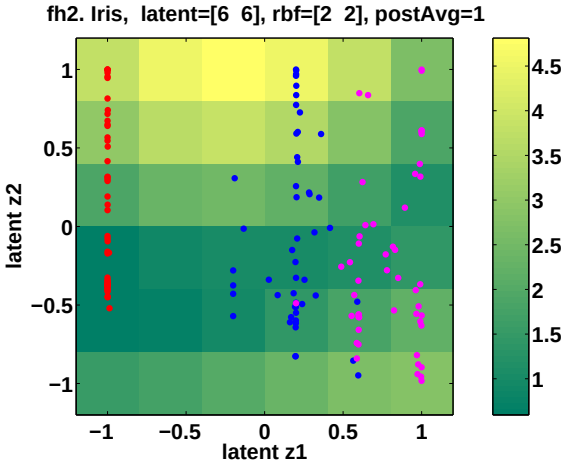
[12 12], [4 4], [6 6], [2 2] i [20 20], [5 5].

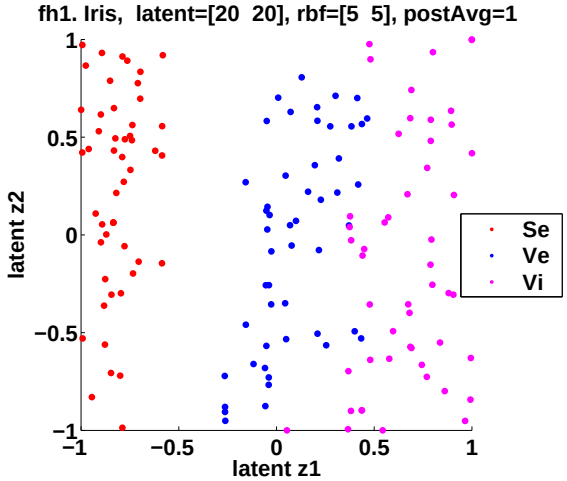
Lewa strona rysunków pokazuje tylko wizualizację punktów indywidualnych z zaznaczonymi gatunkami irysa; prawa strona pokazuje również zacieniowanie wynikające z zastosowania techniki 'Magnification Factor'.

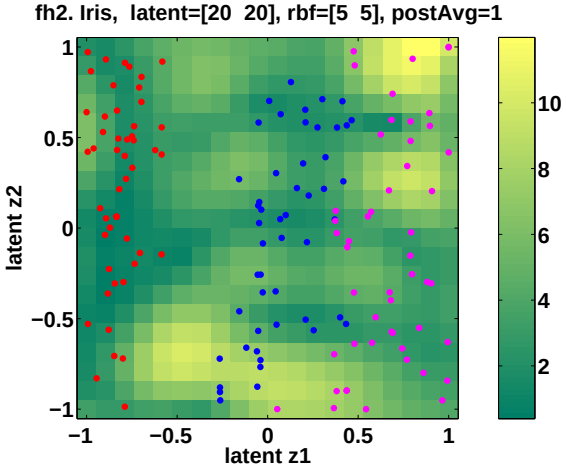

fh1. Iris, latent=[12 12], rbf=[4 4, postAvg=1
class 1
class 2
class 3
latent z1
latent z2


fh2. Iris, latent=[12 12], rbf=[4 4], postAvg=1
latent z1
latent z2


fh1. Iris, latent=[6 6], rbf=[2 2, postAvg=1
Se
Ve
Vi
latent z1
latent z2

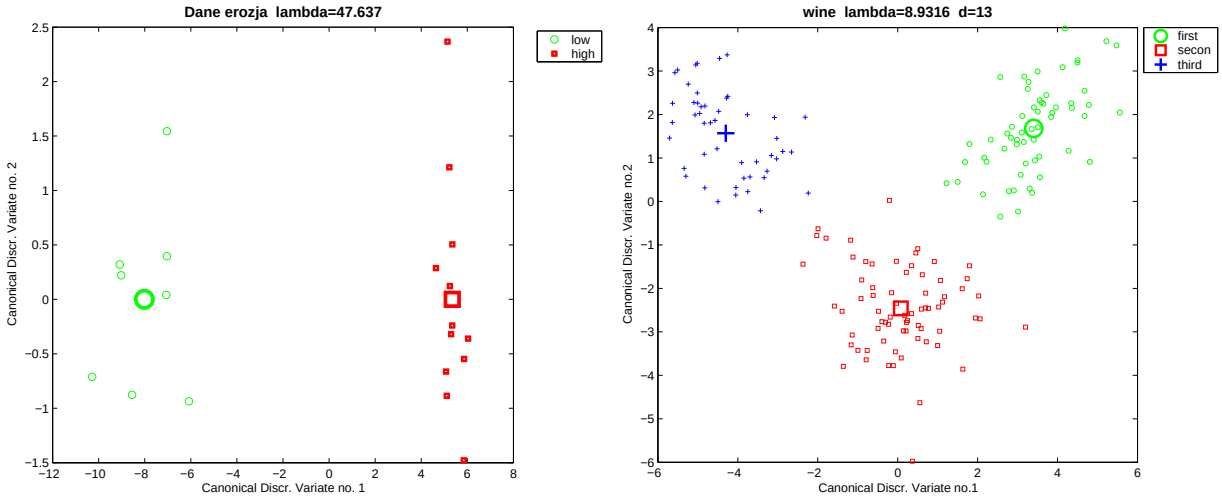

fh2. Iris, latent=[6 6], rbf=[2 2], postAvg=1
latent z1
latent z2


fh1. Iris, latent=[20 20], rbf=[5 5], postAvg=1
Se
Ve
Vi
latent z1
latent z2


fh2. Iris, latent=[20 20], rbf=[5 5], postAvg=1
latent z1
latent z2

10.6 Kanoniczne funkcje dyskryminacyjne

Kanoniczne funkcje dyskryminacyjne zostały opisane w sekcji 7 'Notatek'. Teoria jest opisana np. w książce Lachenbrucha [7].



Rysunek 10.10: *Lewa: Wizualizacja 2 grup na przykładzie danych 'lowHi' przedstawiających niski i bardzo wysoki poziom erozji, plik cand2.eps. Prawa: Wizualizacja 3 grup na przykładzie danych 'wine', plik gtmpara5/candwine.eps*

Literatura

[1] Juha Vesanto, SOM-based data visualization methods. *Intelligent Data Analysis*, 3 (2) 1999, 111–126.

[2] Juha Vesanto, J. Himberg, E. Alhoniemi, J. Parhankangas, *SOM Toolbox for Matlab 5*. Som Toolbox Team, Helsinki University of Technology, Finland, Libella Oy, Espoo 2000, 1–54. http://www.cis.hut.fi/projects/somtoolbox/

[3] S. Osowski, *Sieci neuronowe w ujęciu algorytmicznym.* WNT W-wa 1996.

[4] Sammon, J.W.Jr. (1969). A nonliner mapping for data structure analysis. *IEEE Trans. on Computers*, C-18(5), 401–409, May 1969.

[5] Johan Frederic Marcus Svensen, GTM: The Generative Topographic Mapping. PhD dissertation. Aston Iniversity, Birmingham, Report NCRG/98/024, April 1998.

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

[7] Lachenbruch P.A., *Discriminant Analysis*. Haffner Press, McMillan P.C. Inc., 1975.

[8] Wendy and Angel Martinez, *Computational Statistics Handbook with MATLAB*. CRC Press, 2001.