

Sprawozdanie - sieci Kohonena

Dominik Rafacz

02.05.2020

Oświadczenie o samodzielności pracy

Potwierdzam samodzielność powyższej pracy oraz niekorzystanie przeze mnie z niedozwolonych źródeł.
Dominik Rafacz

Przygotowanie

Załadowanie niezbędnych pakietów

```
library(MIOwAD)      # pakiet z sieciami
library(dplyr)        # transformacja danych
library(stringi)      # transformacja danych
library(ggplot2)       # wykresy
library(patchwork)    # laczenie wykresow
```

Wczytywanie danych

```
X_2d <- read.csv("../data/kohonen/hexagon.csv")
X_3d <- read.csv("../data/kohonen/cube.csv")

X_mnist <- read.csv("../data/kohonen/mnist_train.csv", header = FALSE)
X_mnist <- cbind(number = X_mnist[, 1], X_mnist[ , -1]/255)
X_mnist <- do.call(cbind, lapply(X_mnist, function(col) if(sum(col) == 0) NULL else col))

X_uci <- cbind(
  class = read.csv("../data/kohonen/ucihar_y_train.txt", header = FALSE)[[1]],
  do.call(rbind, lapply(readLines("../data/kohonen/ucihar_X_train.txt"), function(row)
    as.numeric(stri_split_regex(row, " +", omit_empty = TRUE)[[1]])))
))
```

Labortorium 7

Cel: zbudowanie dwuwymiarowych sieci Kohonena i dopasowanie ich do danych 2d i 3d

Wykonamy eksperyment, dopasowując sieci Kohonena do danych ze zbioru 2d i 3d. Będziemy dopasowywać sieci o rozmiarze 10 x 10, korzystając z dwóch możliwych funkcji sąsiedztwa – funkcji gaussowskiej i tzw. meksykańskiego kapelusza, a także z pięciu możliwych wartości parametru skali.

```
ngh_funcs <- c(
  "gauss" = gauss,
  "mexican_hat" = mexican_hat
```

```

)

scale_par <- c(
  `0.1` = 0.1,
  `0.5` = 0.5,
  `1` = 1,
  `5` = 5,
  `10` = 10
)

# trenujemy sieci dla zbioru 2d dla każdej funkcji dla każdego parametru
nets <- list(lab7 = list())
nets[["lab7"]][["2d"]] <- nlapply(ngh_funcs, function(fun)
  nlapply(scale_par, function(par)
    kohonen_network(X_2d[, 1:2], 10, 10, lambda = 10, ngh_fun = fun, scale = par)
  ))
  
# trenujemy sieci dla zbioru 3d dla każdej funkcji dla każdego parametru
nets[["lab7"]][["3d"]] <- nlapply(ngh_funcs, function(fun)
  nlapply(scale_par, function(par)
    kohonen_network(X_3d[, 1:3], 10, 10, lambda = 10, ngh_fun = fun, scale = par)
  )))

```

Teraz wygenerujemy wykresy do wizualizacji efektów treningu.

```

plots <- list(lab7 = list())
plots[["lab7"]][["2d"]] <- nlapply(names(ngh_funcs), function(fun)
  nlapply(names(scale_par), function(par)
    plot_kohonen(generate_net_plot_data(nets[["lab7"]][["2d"]][[fun]][[par]]), X_2d) +
      ggtitle("Sieć Kohonena", paste0("funkcja: ", fun, ", skala: ", par))
  )))
  
plots[["lab7"]][["3d"]] <- nlapply(names(ngh_funcs), function(fun)
  nlapply(names(scale_par), function(par) {
    dat <- generate_net_plot_data(nets[["lab7"]][["3d"]][[fun]][[par]])
    p_xy <- plot_kohonen(dat, X_3d, inp_dims_plot = c(1, 2), inp_class = 4)
    p_zy <- plot_kohonen(dat, X_3d, inp_dims_plot = c(3, 2), inp_class = 4)
    p_xz <- plot_kohonen(dat, X_3d, inp_dims_plot = c(1, 3), inp_class = 4)
    (p_xy + p_zy) /
      (p_xz + plot_spacer())
  }))

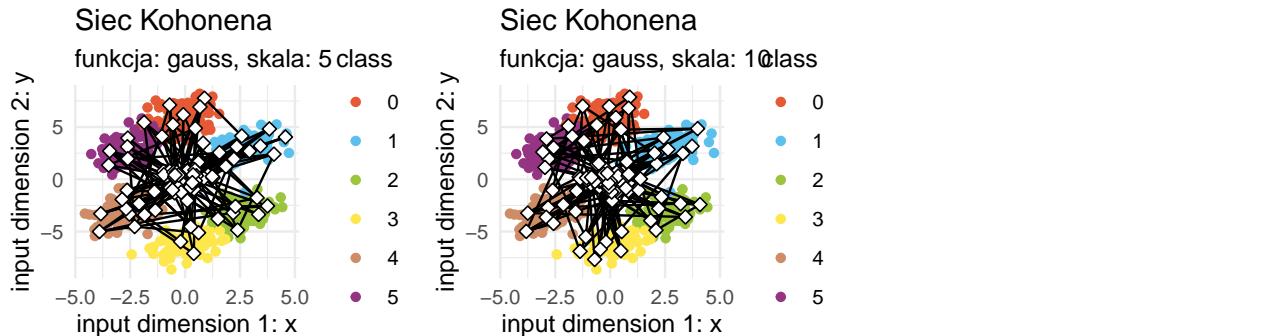
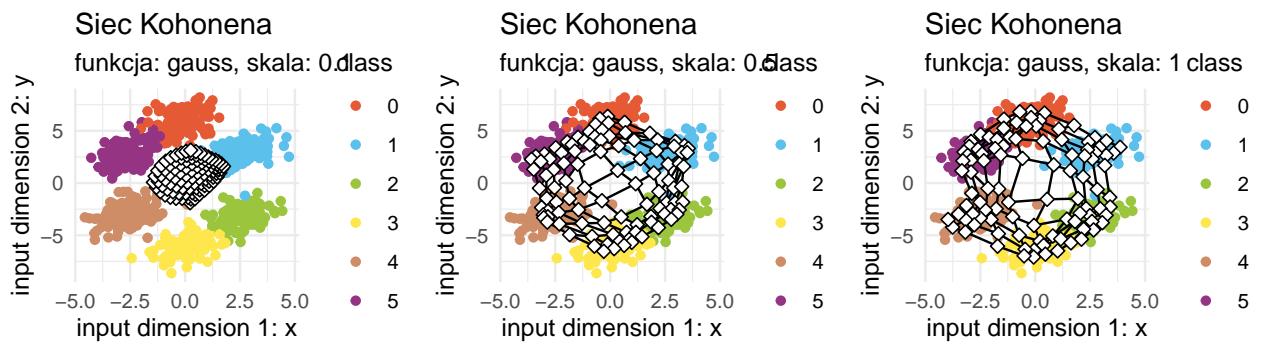
```

Przyjrzyjmy się teraz samym wykresom.

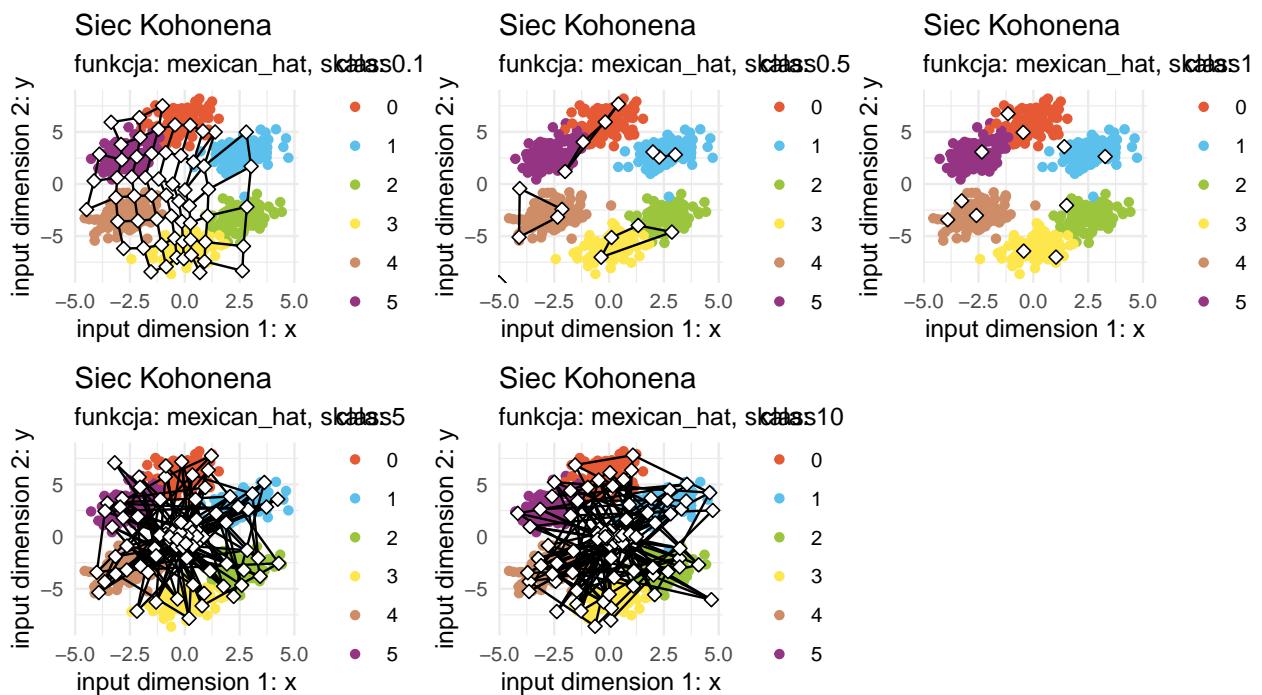
```

plots[["lab7"]][["2d"]][["gauss"]][["0.1"]] +
  plots[["lab7"]][["2d"]][["gauss"]][["0.5"]] +
  plots[["lab7"]][["2d"]][["gauss"]][["1"]] +
  plots[["lab7"]][["2d"]][["gauss"]][["5"]] +
  plots[["lab7"]][["2d"]][["gauss"]][["10"]]

```



```
plots[["lab7"]][["2d"]][["mexican_hat"]][["0.1"]]+
plots[["lab7"]][["2d"]][["mexican_hat"]][["0.5"]]+
plots[["lab7"]][["2d"]][["mexican_hat"]][["1"]]+
plots[["lab7"]][["2d"]][["mexican_hat"]][["5"]]+
plots[["lab7"]][["2d"]][["mexican_hat"]][["10"]]
```

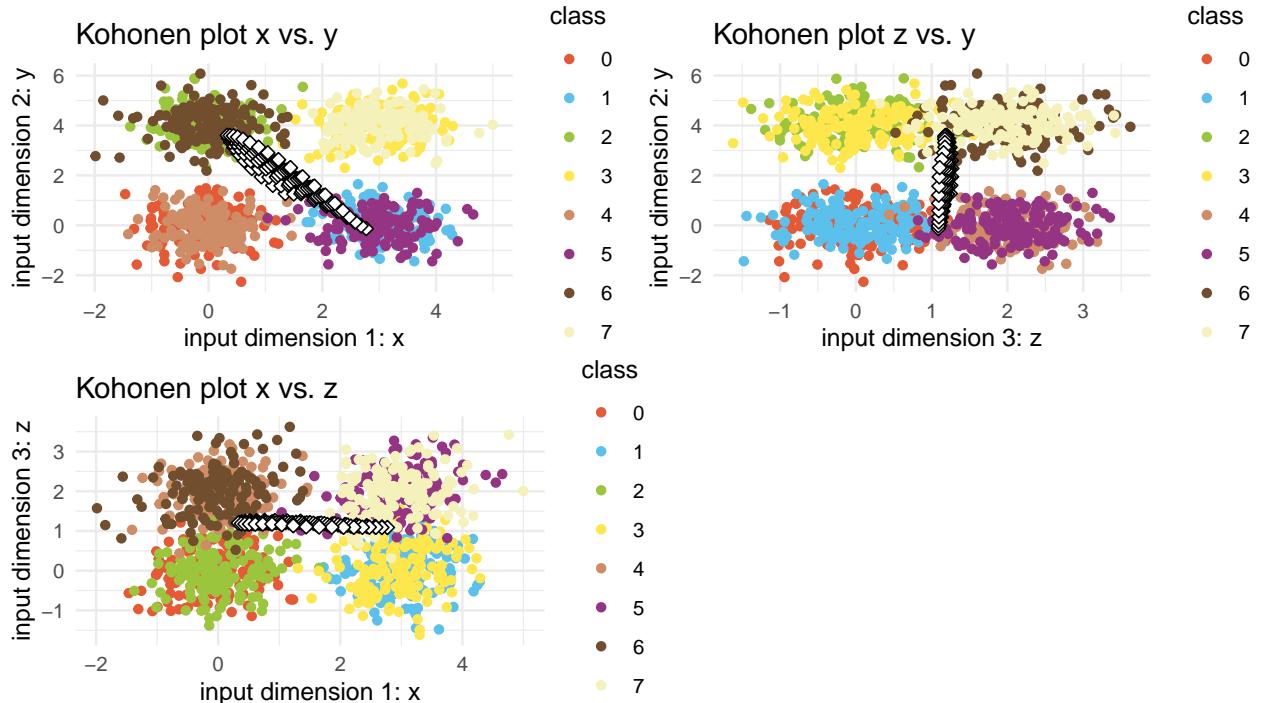


Na pierwszym zestawie wykresów widzimy sieci dopasowane do zbioru z użyciem funkcji gaussowskiej z różnym parametrem skali, na drugim funkcji meksykańskiej z tymi samymi parametrami. Możemy zauważyć:

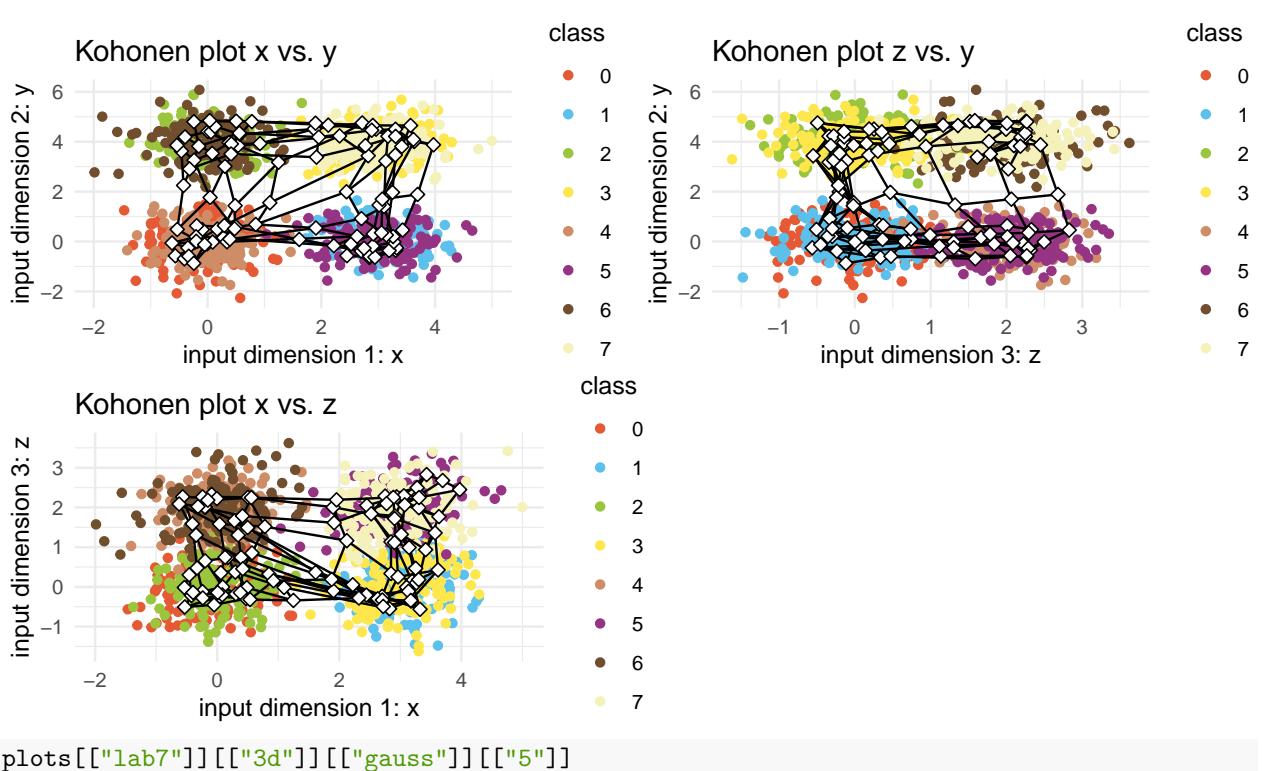
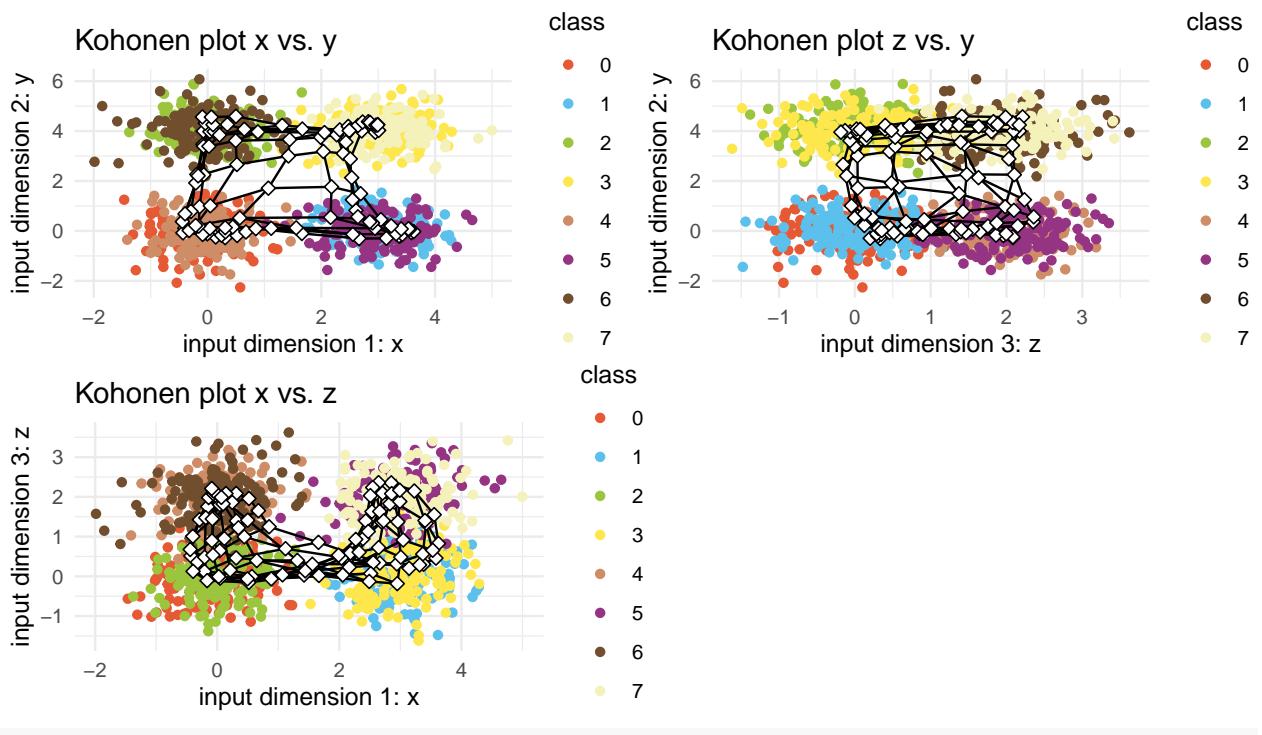
- Dla małego parametru skali, sieć nie dopasowuje się prawie w ogóle do danych.
- Bez modyfikacji skali, dopasowanie jest odpowiednie dla funkcji gaussowskiej
- Dla dużego parametru skali, relacja sąsiedztwa neuronów prawie nie ma w ogóle znaczenia i siatka jest bardzo chaotyczna.
- Funkcja meksykańska “wyrzuca” wiele neuronów poza obszar danych przy nieodpowiednim doborze parametrów.

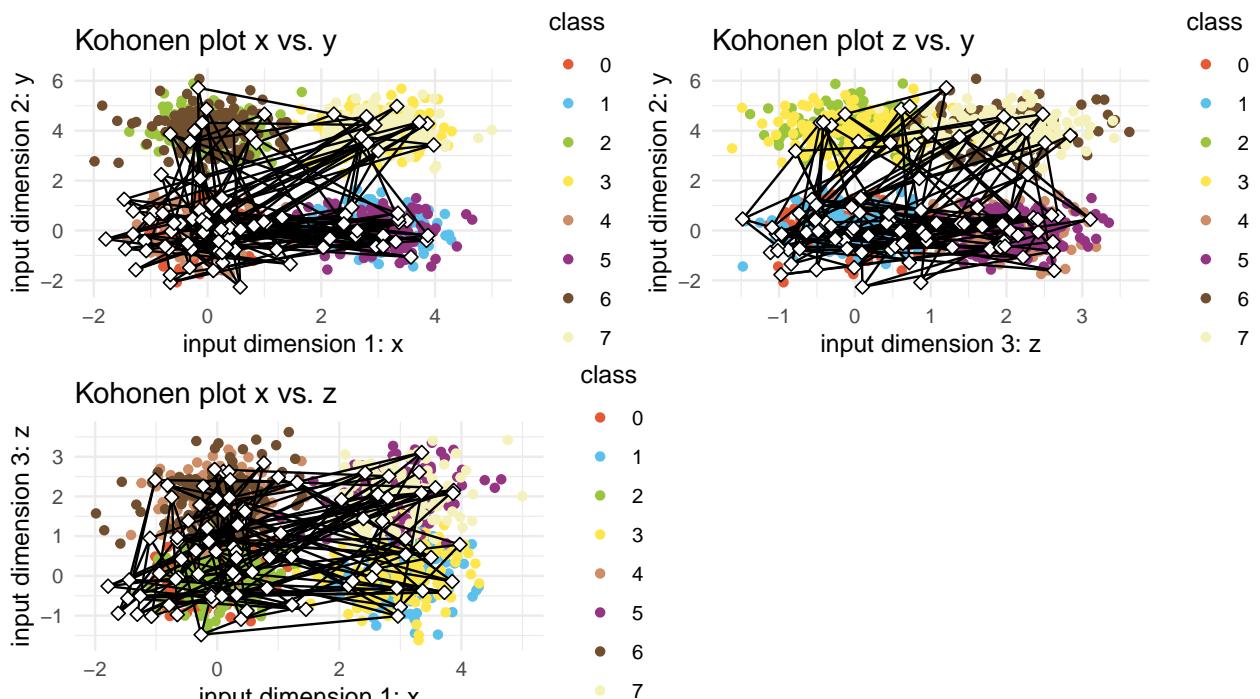
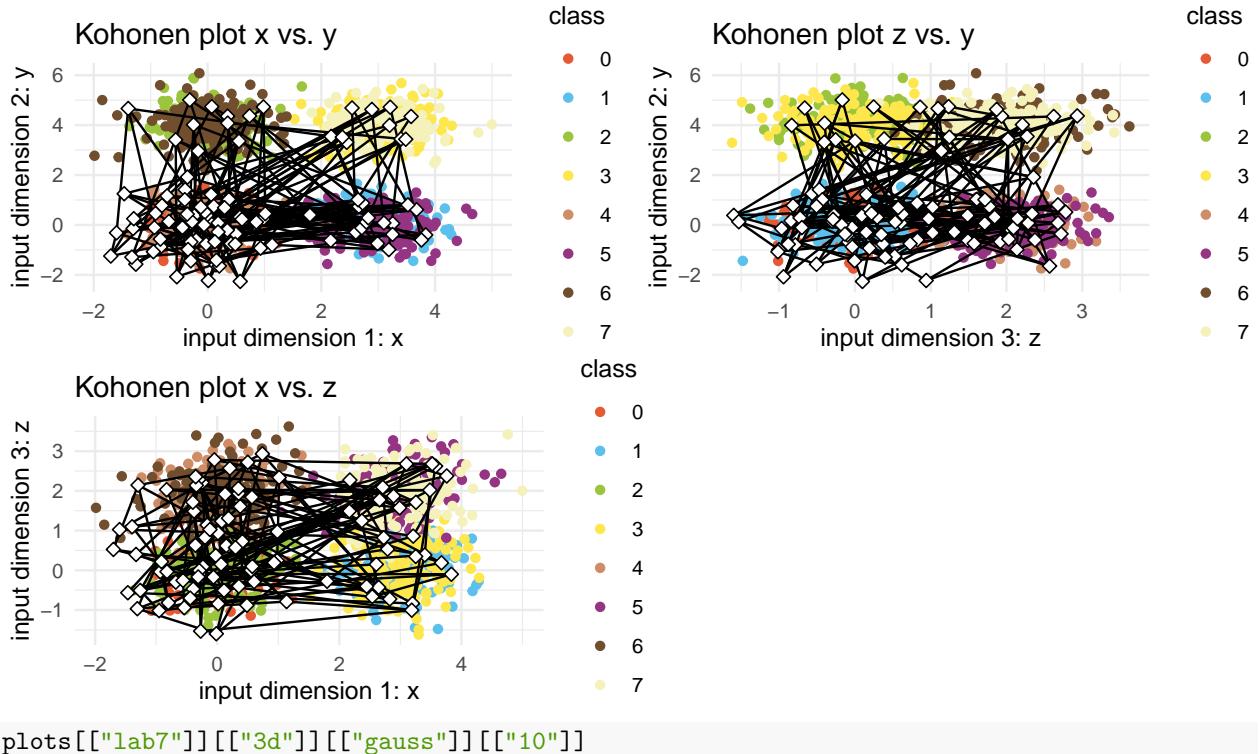
Możemy zauważać, że na dopasowaniach dla najlepszej liczby parametrów neurony klastrują się podobnie do klastrów danych.

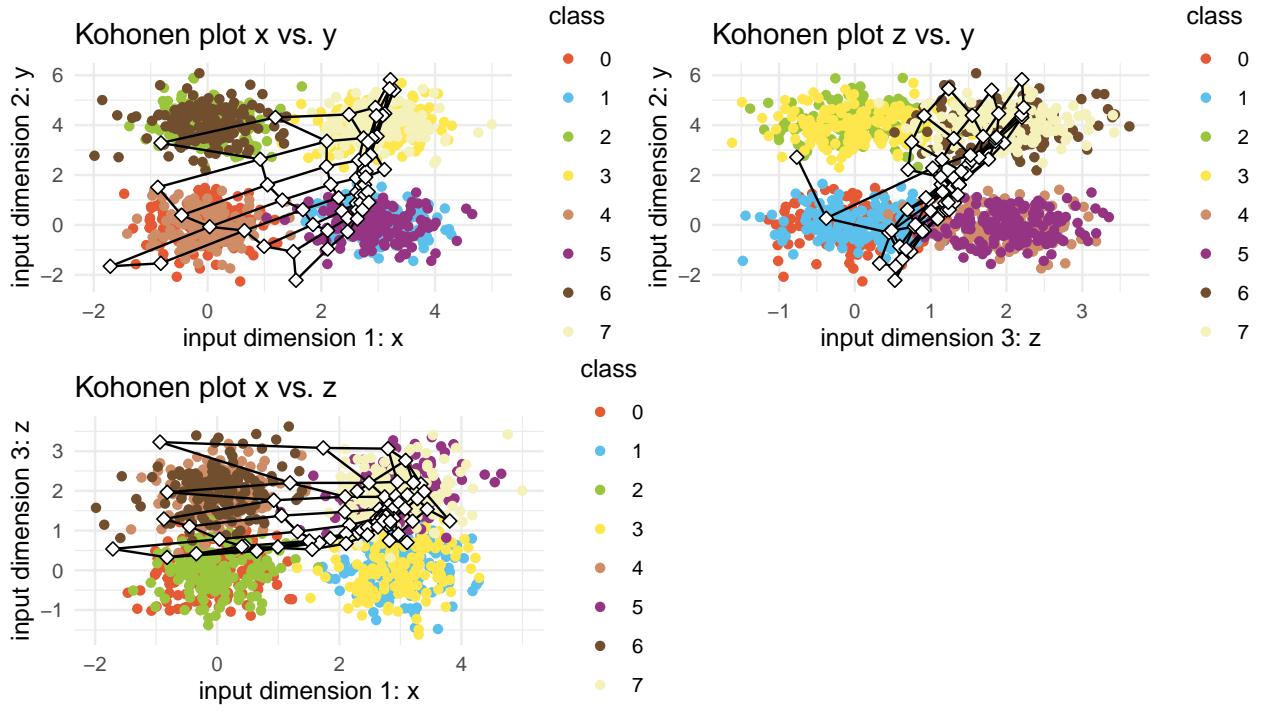
```
plots[["lab7"]][["3d"]][["gauss"]][["0.1"]]
```



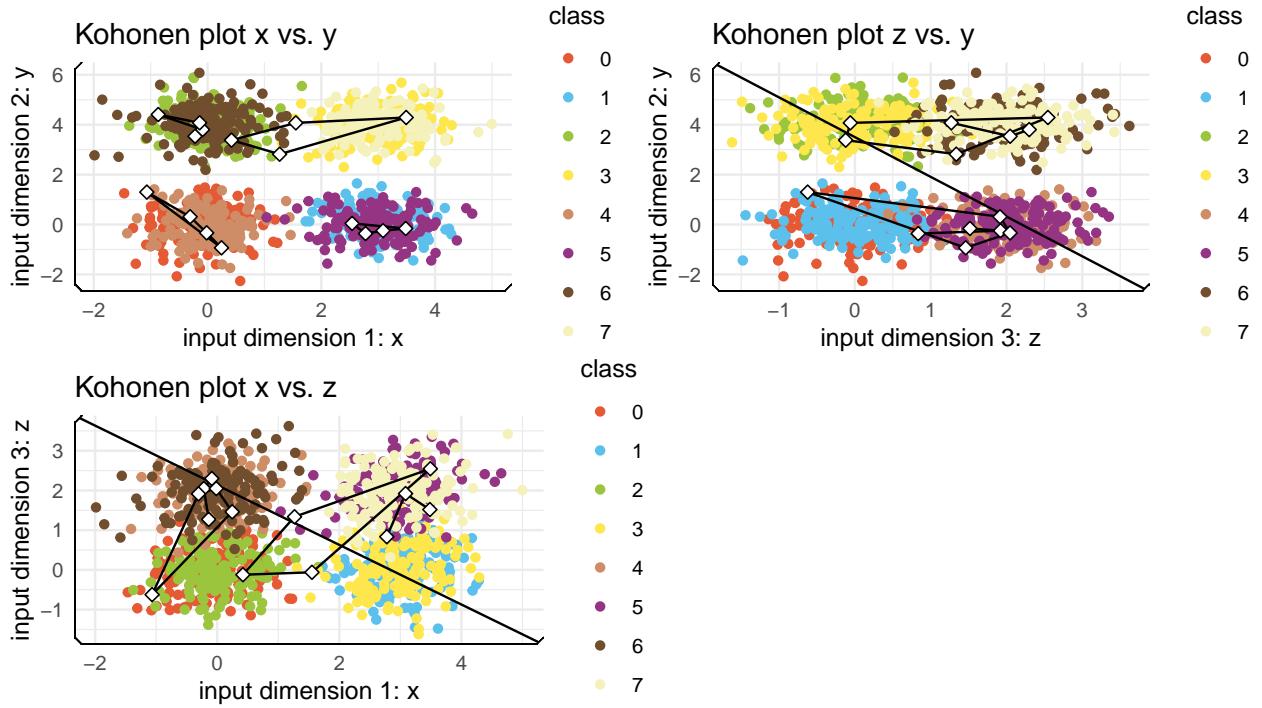
```
plots[["lab7"]][["3d"]][["gauss"]][["0.5"]]
```



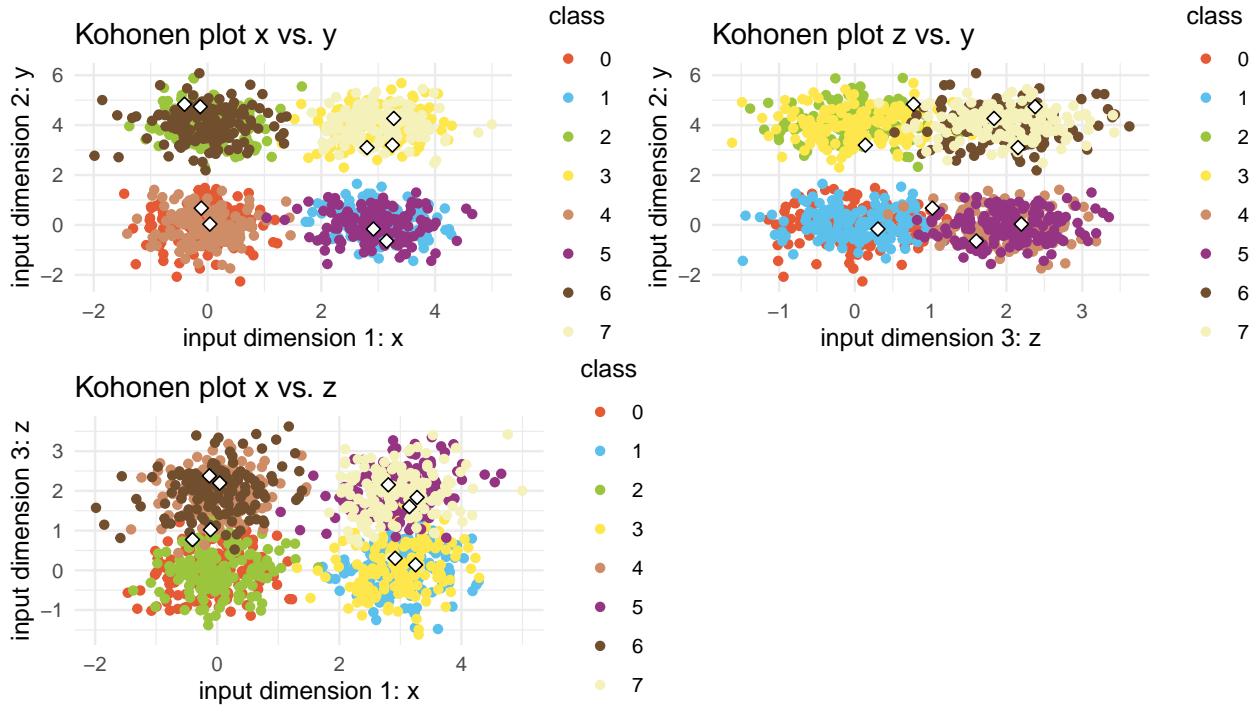




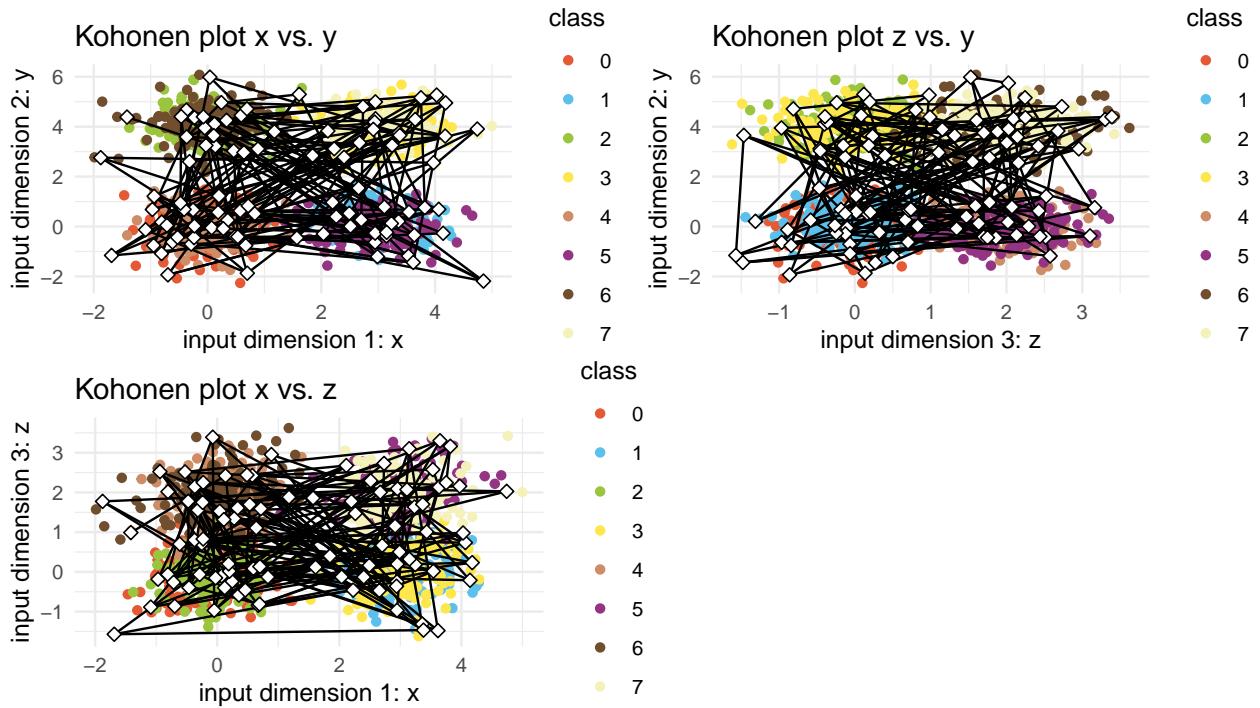
```
plots[["lab7"]][["3d"]][["mexican_hat"]][["0.5"]]
```



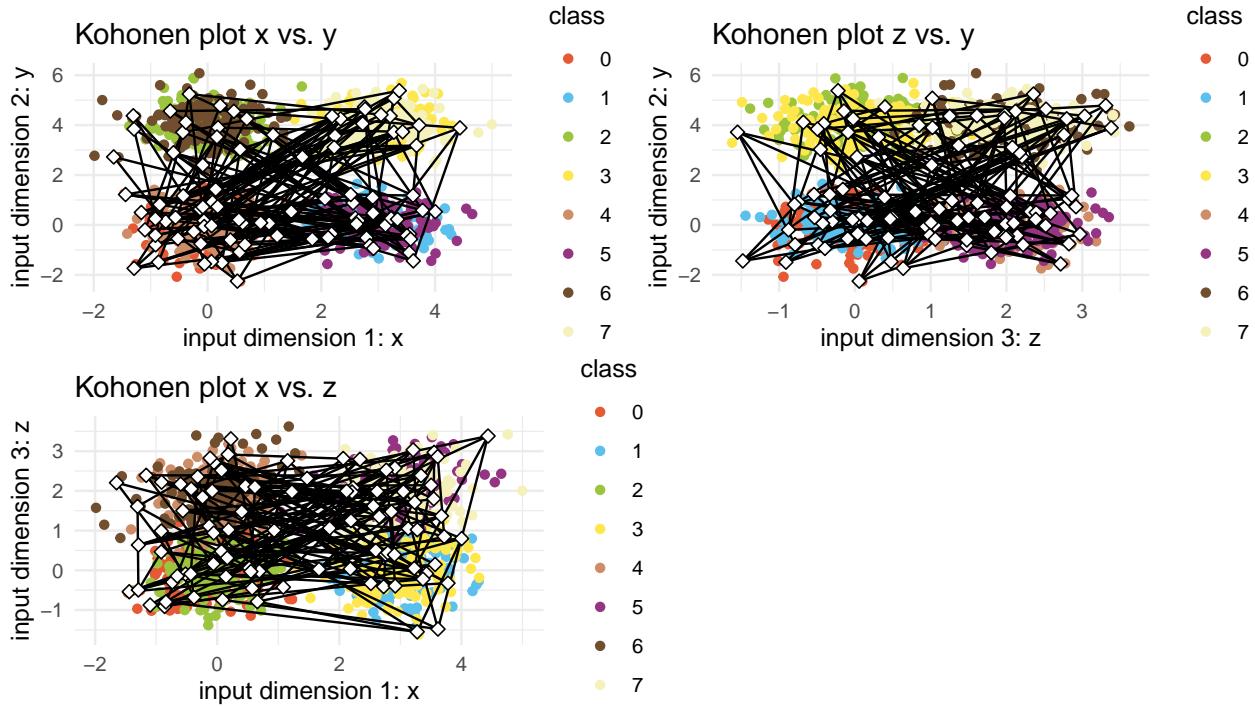
```
plots[["lab7"]][["3d"]][["mexican_hat"]][["1"]]
```



```
plots[["lab7"]][["3d"]][["mexican_hat"]][["5"]]
```



```
plots[["lab7"]][["3d"]][["mexican_hat"]][["10"]]
```



Tutaj każdy wykres to tak naprawdę zestaw trzech wykresów, prezentujących rzuty przestrzeni wejściowej na poszczególne płaszczyzny wyznaczone przez pary osi. Wykresy nie są aż tak czytelne (trudniej zobaczyć strukturę sieci), ale możemy wyciągnąć podobne wnioski jak w przypadku sieci 2d.

Labortorium 8

Cel: implementacja heksagonalnych sieci Kohonena i porównanie ich dopasowania do danych z sieciami kwadratowymi

W tym laboratorium zaimplementowałem możliwość wyboru heksagonalnej topologii sieci. Wykonałem eksperyment, dopasowując sieci Kohonena do danych ze zbioru MNIST oraz ze zbioru UCI HAR. Do każdego z tych zbiorów dopasowałem po cztery sieci, wykorzystując wszystkie kombinacje topologii (kwadratowa lub sześciokątna) i funkcji sąsiedzwa (gaussowska oraz minus druga pochodna funkcji gaussowskiej).

```

topologies = c("square", "hex") # dodatkowy parametr odpowiedzialny za topologie sieci
nets[["lab8"]]  
nets[["lab8"]][["mnist"]]  
nets[["lab8"]][["uci"]]  

  <- list(mnist = list(), uci = list())
  <- nlapply(names(ngh_funcs), function(fun) {
    nlapply(topologies, function(topology)
      kohonen_network(X_mnist[, -1], N = 17, M = 16, lambda = 10,
                      ngh_fun = ngh_funcs[[fun]],
                      scale = if (fun == "gauss") 0.6 else 5,
                      topology = topology,
                      verbose = FALSE))
  })
}

nets[["lab8"]][["uci"]]  
nets[["lab8"]][["uci"]]  

  <- nlapply(names(ngh_funcs), function(fun) {
    nlapply(topologies, function(topology)
      kohonen_network(X_uci[, -1], N = 11, M = 12, lambda = 10,
                      ngh_fun = ngh_funcs[[fun]],
                      scale = if (fun == "gauss") 0.8 else 4,
                      topology = topology,
                      verbose = FALSE))
  })
}

```

```

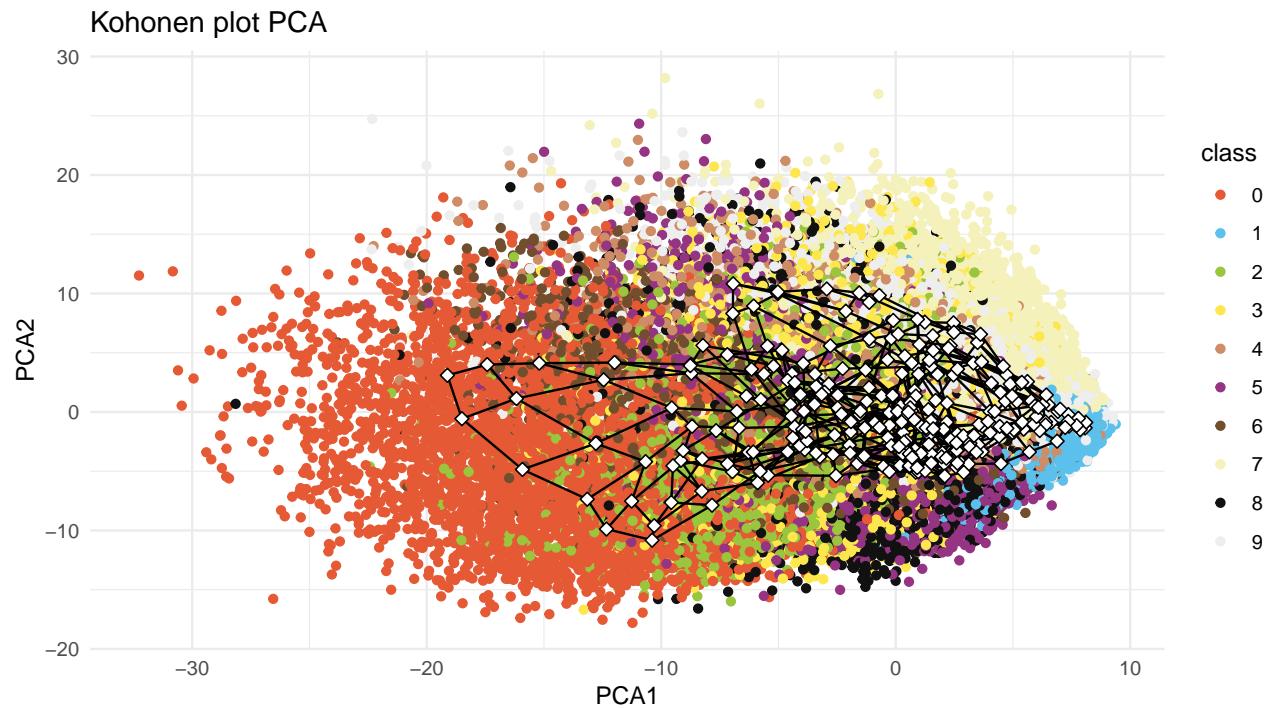
})

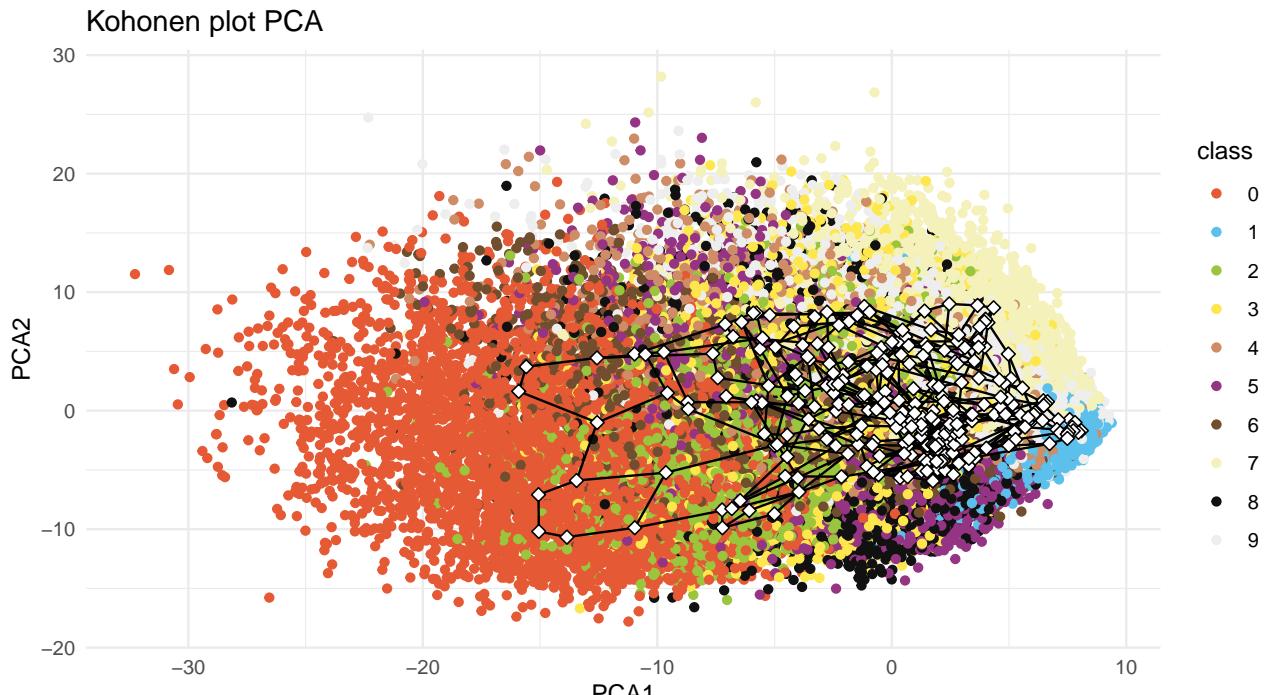
plots[["lab8"]] <- list()
plots[["lab8"]][["mnist"]] <- nlapply(names(ngh_funcs), function(fun)
  nlapply(topologies, function(topology)
    plot_kohonen_w_pca(nets[["lab8"]][["mnist"]][[fun]][[topology]], X_mnist,
                        inp_class = 1)
  ))

plots[["lab8"]][["uci"]] <- nlapply(names(ngh_funcs), function(fun)
  nlapply(topologies, function(topology)
    plot_kohonen_w_pca(nets[["lab8"]][["uci"]][[fun]][[topology]], X_uci,
                        inp_class = 1)
  ))

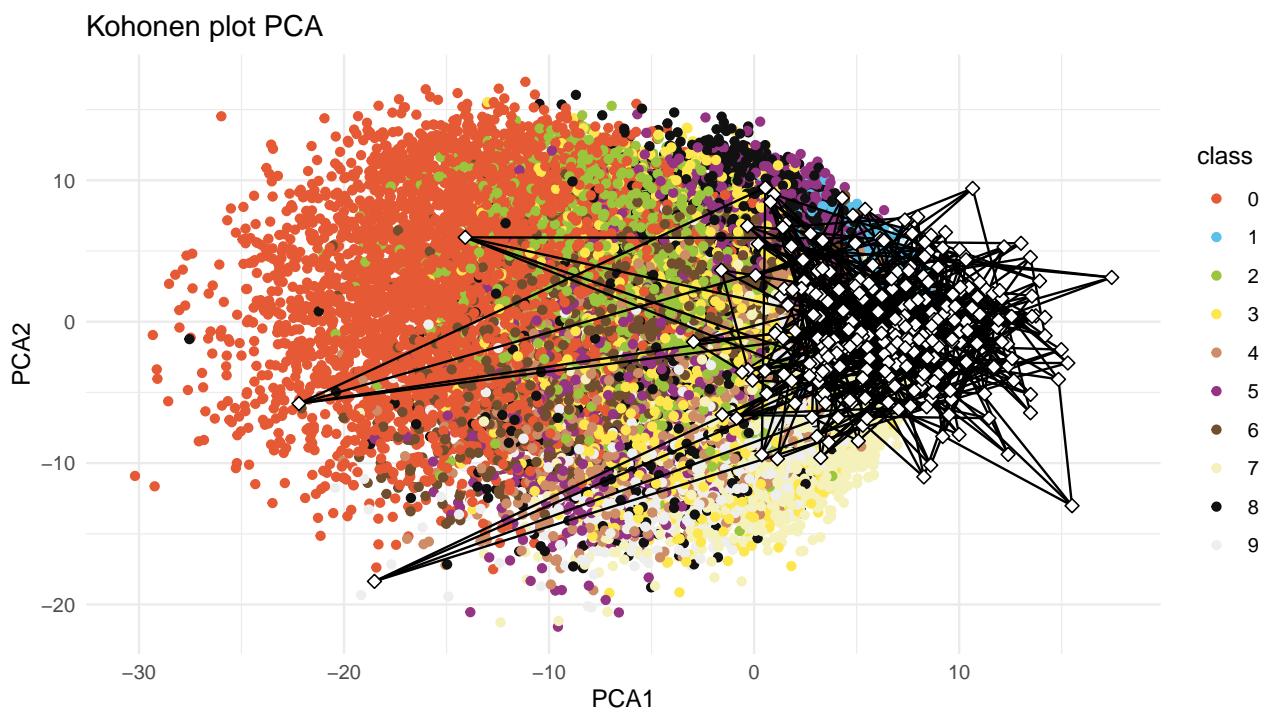
plots[["lab8"]][["mnist"]][["gauss"]][["square"]]

```

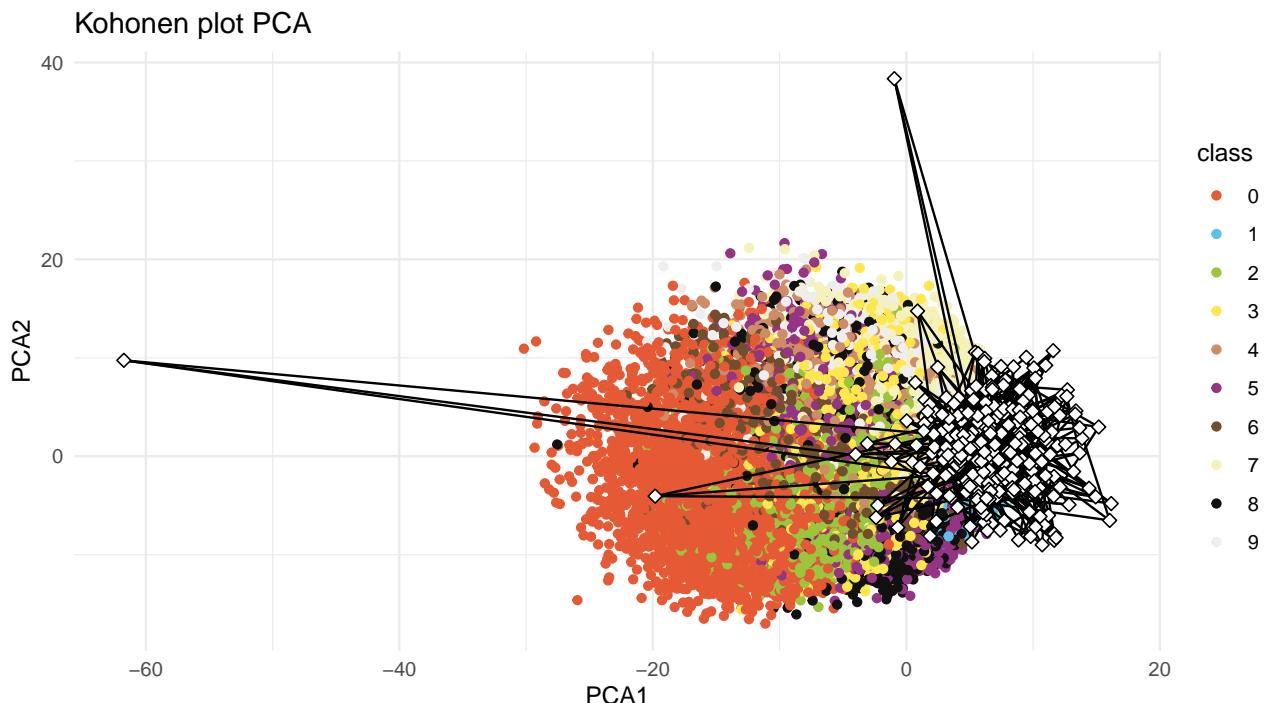




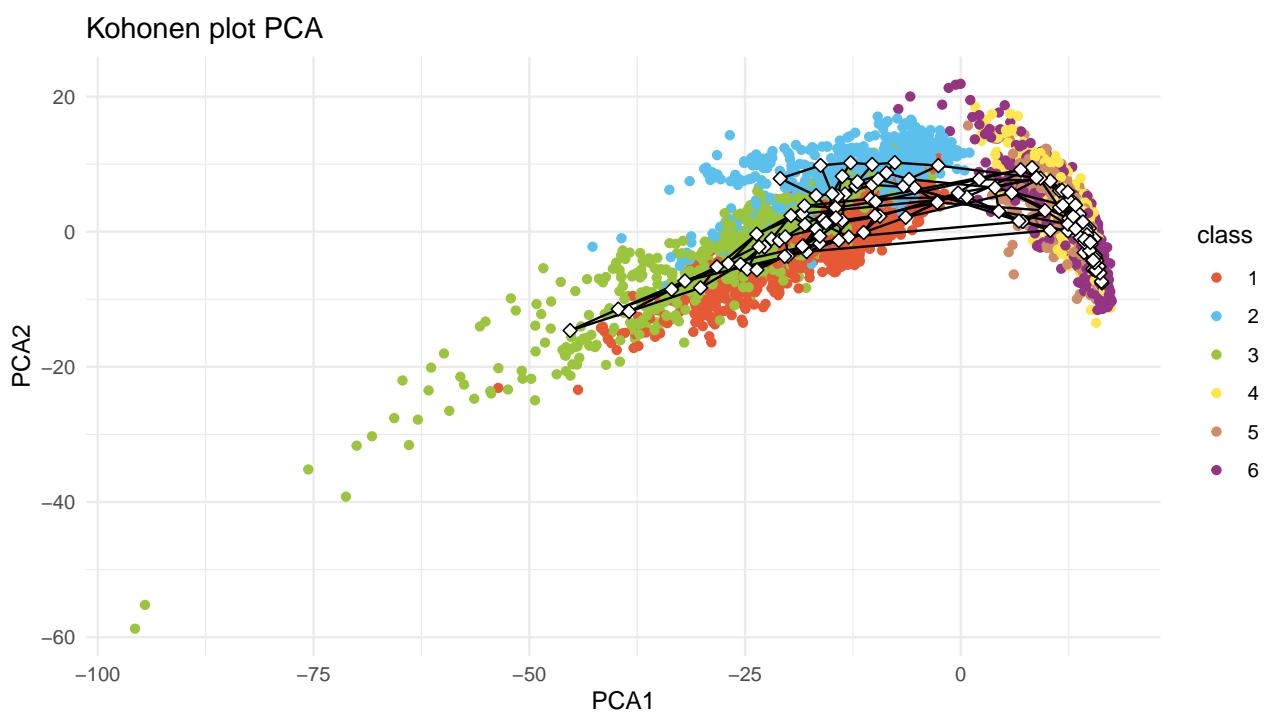
```
plots[["lab8"]][["mnist"]][["mexican_hat"]][["square"]]
```



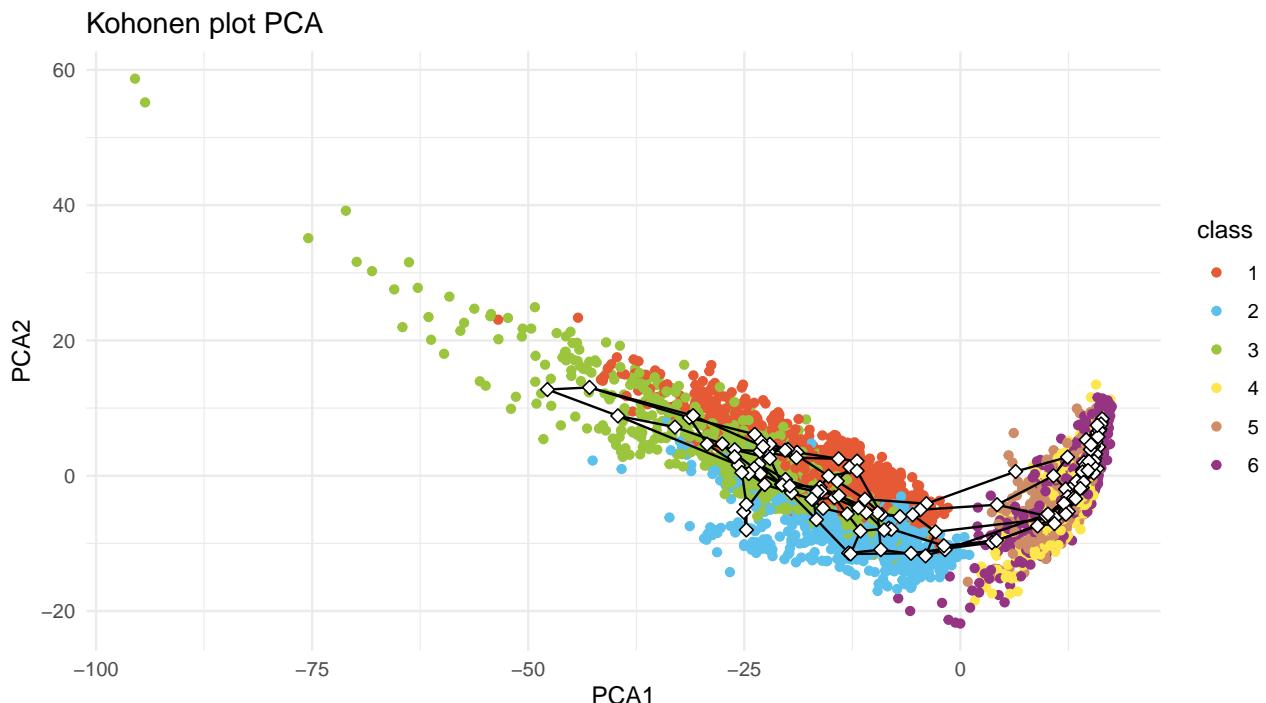
```
plots[["lab8"]][["mnist"]][["mexican_hat"]][["hex"]]
```



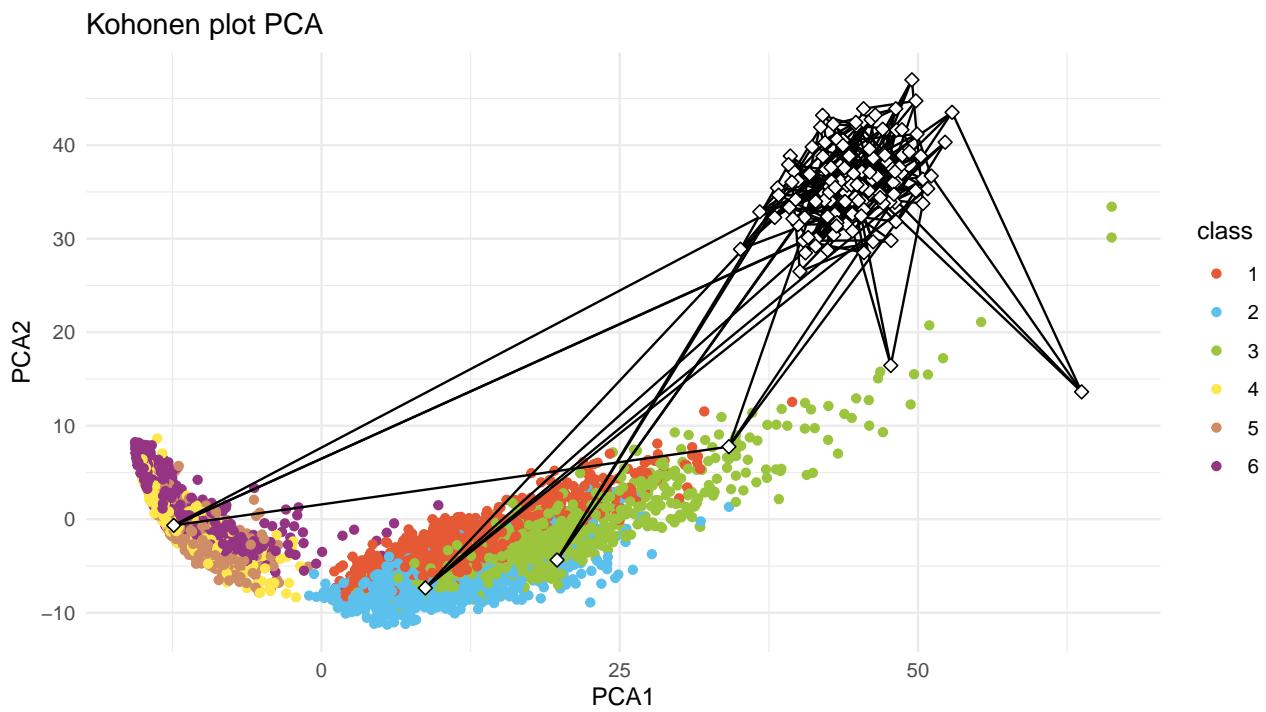
```
plots[["lab8"]][["uci"]][["gauss"]][["square"]]
```



```
plots[["lab8"]][["uci"]][["gauss"]][["hex"]]
```

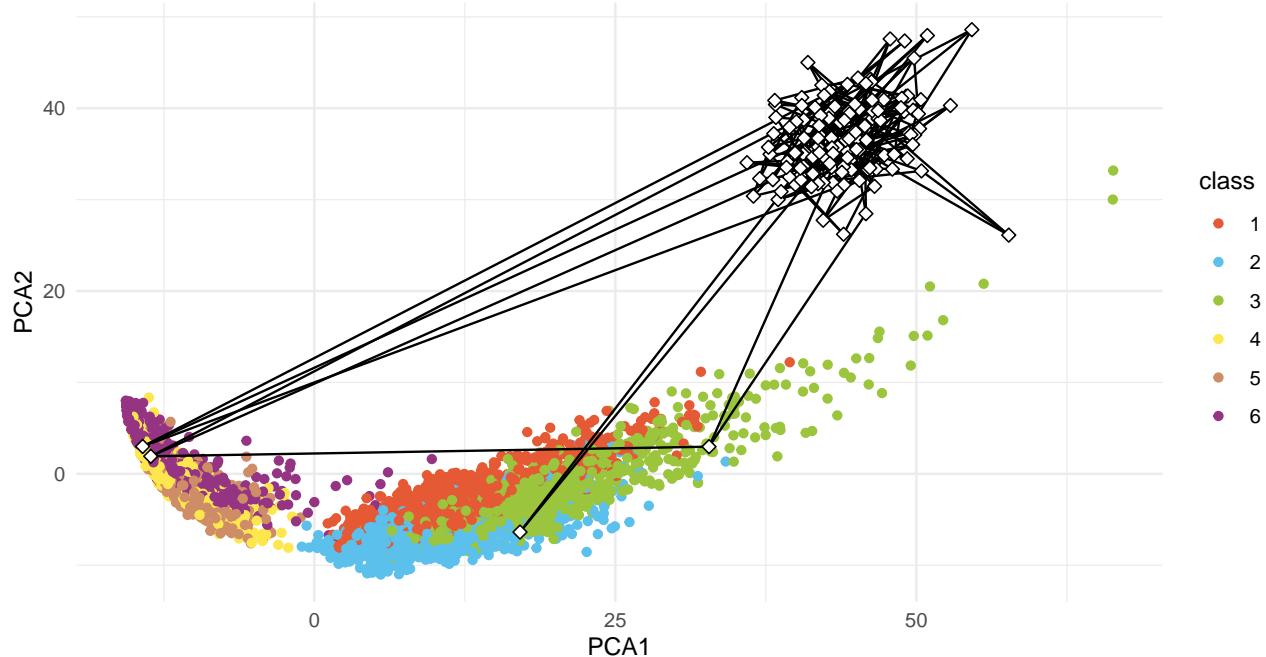


```
plots[["lab8"]][["uci"]][["mexican_hat"]][["square"]]
```



```
plots[["lab8"]][["uci"]][["mexican_hat"]][["hex"]]
```

Kohonen plot PCA



Jak widzimy na wizualizacjach, funkcja sąsiedztwa “mexican hat” nie sprawdza się najlepiej – neurony blisko obserwacji “przestrzeliwują”, ate trochę dalej są odpychane, co skutkuje w bardzo chaotycznym ich rozmieszczeniu na obu zbiorach danych, niezależnie od topologii sieci.

Jeśli chodzi o gaussowską funkcję sąsiedztwa, sprawuje się ona zdecydowanie lepiej. Jeżeli chodzi o porównanie topologii kwadratów i sześciokątów, sześciokąty dopasowują się do danych nieco lepiej, gdyż mają więcej swobody.

Na wizualizacjach możemy wyodrębnić pewne skupiska neuronów, ale trudno dostrzec wyraźne klastry, głównie dlatego, że same dane również trudno zwizualizować w ten sposób.