## Ćwiczenie 2. Algorytm Support Vector Machines (SVM)

Celem ćwiczenia jest zbadanie zdolności uogólniania algorytmu SVM w klasyfikacji danych. Do wyznaczenia mnożników Lagrange'a oraz optymalnej funkcji decyzyjnej (hiperpłaszczyzny separującej) wykorzystana jest metoda Sequential minimal optimization (SMO). Kod źródłowy na Listingu 1 przedstawia zastosowanie funkcji `fitcsvm` (linia 18) do stworzenia klasyfikatora SVM z odpowiednimi parametrami:

- funkcja jądra: `kernel = 'gaussian'`;
- parametr funkcji jądra: `kern_par = 3`;
- metoda rozwiązania problemu programowania kwadratowego: `solver = 'SMO'`;
- ograniczenie dla mnożników Lagrange'a: `C = 0.1`.

Predykcja wyuczonego klasyfikatora SVM realizowana jest za pomocą funkcji `predict` (linia 23). Kolejne kroki listingu umożliwiają znalezienie klasyfikatora SVM o najwyższej dokładności obliczonej w procesie walidacji krzyżowej. Jest on następnie zastosowany do predykcji nowo stworzonego w zbioru testującego `x_test` (linia 35) i podania prawdopodobieństwa klasyfikacji w linii 38. Linia 42 umożliwia na samym końcu wyrysowanie krzywej ROC dla każdej z klas.

```
1  clear;
2  load('DBC');
3  X = DBC(:,1:end-1); Y = DBC(:,end);

4  % Parametry klasyfikatora SVM
5  kernel = 'gaussian'; kern_par = 3; solver = 'SMO'; C = 0.1;

6  % Wygenerowanie indeksów do CV
7  indeksy_cv = crossvalind('Kfold', Y, 10);
8  Accuracy_CV = zeros(10,1);

9  for k = 1 : 10
10     % Indeksy do walidacji i uczenia
11     cv_test_ind = (indeksy_cv == k);
12     cv_train_ind = ~cv_test_ind;

13     % Rekordy do walidacji i uczenia
14     X_Test = X(cv_test_ind,:); Y_Test = Y(cv_test_ind);
15     X_Train = X(cv_train_ind,:); Y_Train = Y(cv_train_ind);
16
17     % Uczenie i walidacja SVM:
18     SVM{k} = fitcsvm(X_Train, Y_Train, 'Standardize', true,
19                      'KernelFunction',kernel,
20                      'KernelScale', kern_par,
21                      'Solver', solver,
22                      'BoxConstraint', C);
23    Label = predict(SVM{k}, X_Test);

24    % Dokładnosc (CV) SVM
25    Accuracy_CV(k) = sum(Label == Y_Test)/length(Y_Test);
26 end

27 Avr_Accuracy = mean(Accuracy_CV);
28 std = std(Accuracy_CV);
```

Listing 1. Zastosowanie algorytmu SVM w klasyfikacji danych w aplikacji Matlab.

Zdolność uogólniania klasyfikatora SVM należy wyznaczyć na podstawie kryterium dokładności. Parametry jakie należy zmieniać w ćwiczeniu to funkcja jądra, jej parametr oraz ograniczenie dla

mnożników Lagrange'a. Ze względu na dużą liczbę symulacji, weryfikację klasyfikatora należy zrealizować według następujących wytycznych:

1. Kod na Listingu 1 nie musi być wielokrotnie uruchamiany, aby wyniki były statystycznie wiarygodne.
2. Badania należy zrealizować dla:
    a) funkcji jądra o rozkładzie Gaussa dla trzech wartości szerokości tej funkcji, np.: $\sigma = \{0.5, 5, 20\}$ oraz trzech wartości parametru regularyzacji: $C = \{0.1, 10, 1000\}$;
    b) wielomianowej funkcji jądra dla wartości wykładnika: $d = \{2, 3\}$ oraz trzech wartości parametru regularyzacji: $C = \{0.1, 10, 1000\}$; w tym wypadku należy przyjąć: `kernel = 'polynomial'`.

Model SVM, dla którego otrzymano najwyższą dokładność (uwzględniając wszystkie badane parametry) należy przetestować w klasyfikacji bez zastosowanej standaryzacji danych: w linii 18 na Listingu należy użyć instrukcji: `'Standardize', false`.

Rezultaty analizy należy przedstawić w formie tabeli prezentując wyniki uśrednionych dokładności dla wszystkich badanych parametrów.