```
while nn >= 2:
    countCutOffPrev = 0
    qq = 0
    while qq < wlsl.countClasses-1:
        ww = 0
        while ww < wlsl.countInstancesEachClassTraining[qq]:
            ee = qq + 1
            while ee < wlsl.countClasses:
                rr = 0
                while rr < wlsl.countInstancesEachClassTraining[ee]:
                    tt = 0
                    while tt < wlsl.sizeVector:
                        #первоначальное приближение вектора весов
                        wlsl.currentWeights[tt] =
                        wlsl.inputsClassTraining[qq][ww][tt] -
                        wlsl.inputsClassTraining[ee][rr][tt]
                        tt += 1
                    #Инициализировать столбец «значение скалярного произведения»
                    wlsl.initColScalarMul(wlsl.currentWeights)
                    #Определить целевую и противоположную категории
                    mm = wlsl.getMinMaxScalarMul()
                    opCat = int(mm[0])
                    taCat = int(mm[1])
                    #Определить максимальное значение скалярного произведения в НЕ
                    целевых категориях
                    noTargMax = wlsl.calcNoTarMax(taCat)
                    #print(noTargMax)
                    #Определить количество отсечённых экземпляров в целевой категории
                    countCutOffTarget = wlsl.calcCutOffSignTarget(taCat, noTargMax)
                    #Определить минимальное значение скалярного произведения в НЕ
                    противоположных категориях
                    noOppoMin = wlsl.calcNoOppMin(opCat)
                    #print(noOppoMin)
                    #Определить количество отсечённых экземпляров в противоположной
                    категории
                    countCufOffOpposit = wlsl.calcCutOffSignOpposit(opCat, noOppoMin)
                    countCutOffCurrent = countCutOffTarget + countCufOffOpposit
                    if countCutOffPrev < countCutOffCurrent:
                        countCutOffPrev = countCutOffCurrent
                        wlsl.previousWeights = wlsl.currentWeights.copy()
                        countCutOffRight = countCutOffTarget
                        categoryRight = taCat
                        maxNoRight = noTargMax
                        countCufOffLeft = countCufOffOpposit
                        categoryLeft = opCat
                        minNoLeft = noOppoMin
                        wlsl.gradientDescentScanning(categoryRight, categoryLeft,
                        countCutOffPrev)
                    rr += 1
                    #ff += 1
                ee += 1
            #print(ww)
            ww += 1
        qq += 1
    #Инициализировать столбец «значение скалярного произведения»
    wlsl.initColScalarMul(wlsl.previousWeights)
    #становить в 1 столбец «признак отсеченности» в целевой категории и вернуть
    значение порога справа.
    thresholdRight = wlsl.setColCutOffSignTarget(categoryRight, maxNoRight)
    thresholdRight = (thresholdRight // 2) + (maxNoRight // 2)
    #Сортировать указанную категорию по возрастанию «признака отсечённости»
    wlsl.sortCategoryCutOff(categoryRight)
    #Установить в 2 столбец «признак отсеченности» в противоположной категории и
    вернуть значение порога слева.
    thresholdLeft = wlsl.setColCutOffSignOpposit(categoryLeft, minNoLeft)
    thresholdLeft = (thresholdLeft //2) + (minNoLeft // 2)
    #Сортировать указанную категорию по возрастанию «признака отсечённости»
    wlsl.sortCategoryCutOff(categoryLeft)
    wlsl.countInstancesEachClassTraining[categoryLeft] -= countCufOffLeft
    wlsl.countInstancesEachClassTraining[categoryRight] -= countCutOffRight
    nn = 0
    rr = 0
```

```
65        while rr < wlsl.countClasses:
66            if wlsl.countInstancesEachClassTraining[rr] > 0:
67                nn += 1
68            rr += 1
69
```