```
while True: #Покоординатный спуск
    iDelta = 0
    while iDelta < ordinateCount:
        ww = calcCutoffDistance(classesCount, instancesMax, ordinateCount,
            Target, Opposite,vectorWeightsCurr, argClasses)
        countCutOffPrev = ww[0]
        distanceCutOffPrev = ww[1]
        deltaMultiPrev = 0
        weightsOld = vectorWeightsCurr[iDelta]  #Проверка направления спуска
        deltaMultiCurr = calcDescentDirection(classesCount, instancesMax,
        ordinateCount,Target,Opposite,vectorWeightsCurr,argClasses)
        while True:
            deltaMultiCurr = deltaMultiCurr * 2
            vectorWeightsCurr[iDelta]=weightsOld+deltaMultiCurr+deltaMultiPrev
            ww=calcCutoffDistance(classesCount,instancesMax,ordinateCount,Target,
            Opposite,vectorWeightsCurr,argClasses)
            countCutOffCurr = ww[0]
            distanceCutOffCurr = ww[1]
            if countCutOffCurr > countCutOffPrev:
                countCutOffPrev = countCutOffCurr
                distanceCutOffPrev = distanceCutOffCurr
                deltaMultiPrev += deltaMultiCurr
                deltaMultiCurr=calcDescentDirection(classesCount,instancesMax,
                ordinateCount,Target,Opposite,vectorWeightsCurr,argClasses)
            elif countCutOffCurr < countCutOffPrev:
                deltaCutoffDistance[0][iDelta] = vectorWeightsCurr[iDelta] -
                deltaMultiCurr - weightsOld
                deltaCutoffDistance[1][iDelta] = countCutOffPrev
                vectorWeightsCurr[iDelta] = weightsOld
                break
            elif distanceCutOffCurr > distanceCutOffPrev:
                deltaCutoffDistance[0][iDelta] = vectorWeightsCurr[iDelta] -
                deltaMultiCurr - weightsOld
                deltaCutoffDistance[1][iDelta] = countCutOffPrev
                vectorWeightsCurr[iDelta] = weightsOld
                break
            else:
                countCutOffPrev = countCutOffCurr
                distanceCutOffPrev = distanceCutOffCurr
        iDelta += 1
    iDelta = 1
    maxCutoff = deltaCutoffDistance[1][0]
    maxCutoffIndex = 0
    while iDelta < ordinateCount:#
        if maxCutoff < deltaCutoffDistance[1][iDelta]:
            maxCutoff = deltaCutoffDistance[1][iDelta]
            maxCutoffIndex = iDelta
        iDelta += 1
    iDelta = 0
    condCycle = 0
    minCutoffDistance = abs(deltaCutoffDistance[0][maxCutoffIndex])
    maxCutoffIndex = ordinateCount
    while iDelta < ordinateCount:
        condCycle += abs(deltaCutoffDistance[0][iDelta])
        if maxCutoff == deltaCutoffDistance[1][iDelta]:
            if minCutoffDistance >= abs(deltaCutoffDistance[0][iDelta]):
                maxCutoffIndex = iDelta
        iDelta += 1
    vectorWeightsCurr[maxCutoffIndex] += deltaCutoffDistance[0][maxCutoffIndex]
    if condCycle == 0:
        break
contrastingWeights(classesCount, instancesMax, ordinateCount, Target, Opposite,
    vectorWeightsCurr, argClasses)
valueDoorstep = calcBiasDoorstep(classesCount, instancesMax, ordinateCount, Target,
    Opposite, vectorWeightsCurr, argClasses)
print(vectorWeightsCurr)
```