TP Perceptron avec 1 neurone

Introduction : Neurone formel simple

On exploitera pour ce TP un neurone formel constitué de :

- 2 entrées réelles X et Y dont les valeurs sont comprises dans l'intervalle [0 ; 599] (taille de l'image)
- 1 entrée constante C =1
- 3 poids réels w_1, w_2 et w_3 associés respectivement aux entrées X, Y et C
- 1 fonction d'activation définie de la façon suivante :
    - On calcule somme pondérée : $somme = w_1X + w_2Y + w_3C$
    - Si somme > 0 alors sortie = 1 // seuil à 0
    - Sinon sortie = 0

Algorithme du perceptron :

Initialisation de l'ensemble des exemples d'apprentissage de la classe 1 et de la classe 2
Initialisez le coefficient d'apprentissage k ; k=1 par exemple
Initialisation des poids w_1, w_2 et w_3 *de manière aléatoire entre -100 et +100.*
seuil = 0
Répéter
    Initialisation à 0 du nombre d'erreurs de classification
    Pour chacun des couples (X,Y) de la base de données
        C=1 // Entrée constante
        Calculer la sortie de la façon suivante :
            $somme = w_1X + w_2Y + w_3C$
        Si somme > seuil alors sortie =1 sinon sortie =0
        Si sortie =1 alors classe = 1 sinon classe= 2
        Si classe = 2 et (X,Y) est un exemple de la classe 1, faire :
            $w_1 = w_1 + k\,X$
            $w_2 = w_2 + k\,Y$
            $w_3 = w_3 + k\,C$
            augmenter de 1 le nombre d'erreurs
        Si classe = 1 et (X,Y) est un exemple de la classe 2, faire :
            $w_1 = w_1 - k\,X$
            $w_2 = w_2 - k\,Y$
            $w_3 = w_3 - k\,C$
            augmenter de 1 le nombre d'erreurs
        Augmenter de 1 le nombre d'itérations
Jusqu'à ce qu'il n'y ait plus d'erreur ou qu'on ait atteint un très grand nombre d'itérations (à ajuster pour que le traitement ne dépasse pas quelques secondes).
Afficher le nombre d'erreurs et les valeurs finales des poids

1. Programmation du perceptron

Récupérez le code source C# fourni par l'enseignant permettant d'initialiser et d'afficher les exemples de la classe 1 et de la classe 2. Chaque classe est ici définie par un point central et un écart-type noté « sigma » homogène en abscisse et en ordonnée relativement aux coordonnées du centre de la classe.

A l'aide de l'algorithme ci-dessus, au clic sur un nouveau bouton, programmez l'apprentissage du perceptron afin qu'il apprenne à sortir 1 pour tous les exemples de la classe 1 et 0 pour tous les exemples de la classe 2. Pour cela, modifiez les 2 fonctions suivantes (voir code) :

private int calcule_sortie(double x, double y)
private void button1_Click(object sender, EventArgs e)

Aide : dans le code, w1, w2, w3 sont les poids à modifier, placés en attributs de Form1. Le numéro de classe de l'exemple i, est donné par ClassToLearn[i].

2. Classification des points de l'image

Qu'observez-vous au niveau du résultat ? (=> Avec un perceptron 1 neurone, la séparation est linéaire).
Une seule itération avec tous les exemples ne permet en général pas d'aboutir à un excellent résultat. Il faut donc itérer un grand nombre de fois pour faire converger et réduire le nombre d'erreurs (diagonale de la matrice de confusion avec des 100%).
Faites varier le point central de chaque classe et les écarts-types sigma pour étudier l'efficacité du perceptron lorsque les classes sont difficilement séparables.

3. Limites du perceptron

Modifiez également la valeur de k pour voir l'effet sur la vitesse d'apprentissage.

Modifiez la constante, posez C = 0 dans le programme et observez l'impact sur les limites du perceptron, en particulier lorsque les classes sont séparables mais pas par une droite qui passe par l'origine. Pour ce cas particulier, sauriez-vous démontrer l'impossible convergence de l'apprentissage ?

4. Sigmoïde

Modifiez la fonction d'activation et remplacez-la par la fonction sigmoïde (voir cours). Quel est l'impact sur le résultat ?

5. En option pour les plus rapides

Ajoutez une 3ème classe définie elle aussi par un centre et un écart-type, avec 500 points pour l'apprentissage.
Pour pouvoir partager le plan en 3 zones, il suffit d'utiliser 2 perceptrons, le premier pour apprendre à distinguer les exemples de la classe 1 de tous les autres, et le deuxième pour apprendre à distinguer les exemples de la classe 2 de tous les autres. On obtient donc 2 droites séparatrices et les exemples de la classe 3 vérifient la condition d'un ET entre 2 sorties à 0 pour les 2 premiers perceptrons. Modifiez le programme pour montrer l'apprentissage de ces 3 classes avec 2 perceptrons 1 neurone.