ENSC 2A

Intelligence Artificielle 2

TP apprentissage supervisé

avec un perceptron multicouches

Un code source C# est fourni. Il permet de tester les performances d'un perceptron multicouches, à la fois en classification et en régression. Pour l'exploiter, il faut définir dans un premier temps le réseau de neurones, en veillant à indiquer correctement le bon nombre d'entrées (1 ou 2 dans le cadre de ce TP), le bon nombre de couches cachées (typiquement entre 0 et 2, sachant que si 0 est choisi, on obtient un perceptron simple) et le nombre de neurones par couche cachée, typiquement entre 2 et 20.

Exercice 1 : Apprentissage classification 1D

1.1 Tests

Testez l'apprentissage supervisé de la classification en 1 dimension.

Entrée : 1 valeur réelle dans]0 ;1[

Sortie : 1 valeur réelle dans]0 ;1[(sortie sigmoïde)

Classification à apprendre :

Si x est dans la classe 1 définie par l'intervalle]0.3 ; 0.7[, alors la sortie doit valoir 0.2

Si x est dans la classe 2 définie par]0 ; 3] U [0.7 ; 1[, alors la sortie doit valoir 0.8

L'apprentissage est réalisé avec 1000 couples (x,classe(x)), x étant choisi aléatoirement entre 0 et 1.

Paramètres :

Nombre d'époques ou nombre de fois qu'on présente chaque exemple : pour l'apprentissage, il faut présenter chacun des exemples au moins 1 fois (tout le dataset d'apprentissage). En général, cela ne suffit pas, car la convergence est lente. Il faut donc itérer et repasser plusieurs fois toute la liste. 1 présentation du dataset = 1 époque. NB : dans ce TP, dans un souci de simplification, on ne divise par le dataset en « batch », c'est-à-dire en sous-ensembles d'apprentissage, ce qui est souvent plus efficace.

Taux d'apprentissage (alpha) : à chaque fois qu'on présente un exemple et qu'on détermine une erreur entre la sortie souhaitée et la sortie obtenue, on modifie les poids du réseau en fonction de ce paramètre. Ici, il est constant. Ce n'est pas optimal. Idéalement, il faudrait qu'il décroisse au fur et à mesure que le réseau converge vers la sortie optimale, ou déterminer la valeur la plus adaptée.

Explication du graphique : en abscisse, on a x entre 0 et 1 et en ordonnée on a la sortie également entre 0 et 1. Le graphique montre le résultat de la classification pour tous les x entre 0 et 1, avec un pas de 0.01 En blanc, c'est le résultat souhaité et en jaune le résultat obtenu par le réseau après apprentissage.

Testez la classification 1D en initialisant le réseau comme il se doit, puis en ajustant éventuellement les paramètres et en cliquant finalement (éventuellement plusieurs fois) sur le bouton « Apprentissage classification 1D ».

Une classification est estimée satisfaisante si 95% des entrées sont correctement classées (réponse du perceptron au-dessous de 0.5, on affecte à classe 1 et au-dessus de 0.5 à classe 2).

Est-il possible d'obtenir une classification satisfaisante avec 0 couche cachée ? Pourquoi ?

Avec 1 couche cachée, quel est le nombre minimal de neurones sur la couche cachée permettant d'obtenir des résultats intéressants ? (Itérez de nombreuses fois jusqu'à convergence).

1.2 Modification du code

La classification est définie par une « fonction modèle ». Voir code. Modifiez la fonction modèle de cette classification pour obtenir une autre classification :

Classification à apprendre :

> Si x est dans la classe 1 définie par l'intervalle [0 ;0.2[U [0.5 ; 0.8] alors la sortie doit valoir 0.2
>
> Si x est dans la classe 2 définie par tous les autres cas, la sortie doit valoir 0.8

Quel est l'impact sur le nombre de neurones de la couche cachée nécessaires pour obtenir une classification satisfaisante ?

Exercice 2 : Apprentissage régression 1D

2.1 Testez l'apprentissage supervisé d'une fonction en 1 dimension (régression).

Entrée : 1 valeur réelle entre 0 et 1

Sortie : 1 valeur réelle entre [0 ;1]

Fonction à apprendre : f(x) = sin(20 x) / 2.5 + 0.5

L'apprentissage est réalisé avec 1000 couples (x,f(x)), x étant choisi aléatoirement entre 0 et 1 et f(x) étant calculé de manière algébrique. En blanc, on a la courbe à apprendre et en jaune la courbe apprise grâce à un ensemble de 1000 couples (x,f(x)) avec x choisi aléatoirement entre 0 et 1.

Testez l'apprentissage en cliquant sur le bouton « Apprentissage régresion 1D » , sans oublier d'initialiser le réseau et d'ajuster les paramètres …

Avec 1 couche cachée, comment s'améliore la performance en fonction du nombre de neurones de la couche cachée (après itérations et convergence) ?

Quel est le nombre minimal de neurones sur la couche cachée permettant d'obtenir des résultats intéressants (faible écart entre la sortie attendue et la sortie calculée pour tous les x entre 0 et 1) ? Avec 2 couches cachées, qu'observez-vous ?

2.2 Modification du code

Modifiez la fonction modèle de cette classification : si on réduit ou on augmente la période du sinus, quel est l'impact sur le nombre de neurones de la couche cachée nécessaires pour obtenir une régression qui suit assez bien la courbe ?

Exercice 3 :

3.1 Nous allons tester la classification en 2 dimensions avec l'exemple proposé, où l'objectif est d'apprendre au réseau à classer les points (x,y) d'une image :

Entrée : 2 réels x, y dans l'intervalle [0,800[

Sortie : 1 réel entre 0 et 1

Classification à apprendre :

classe 1 si le point est dans un cercle, valeur associée : 0.8

classe 2 si le point est en dehors, valeur associée : 0.2

L'apprentissage est réalisé avec 1000 points choisis aléatoirement avec des abscisses et ordonnées comprises dans l'intervalle [0 ; 800[et auxquels on a associé la bonne classification (Remarque : pour l'entrée du réseau, on se ramène à des entrées entre 0 et 1 en divisant chaque abscisse et ordonnée par 800). Après apprentissage, le réseau est ensuite testé avec tous les points d'une image 800 par 800 (donc 64000 points) et on affecte alors une couleur selon la convention suivante :

Si le point est de la classe 1 et que le réseau classe ce point dans la classe 1 (en fait, la sortie calculée est supérieure à 0.5), on l'affiche en jaune.

Si le point est de la classe 2 et que le réseau classe ce point dans la classe 2 (la sortie calculée est inférieure à 0.5), on l'affiche en bleu.

Si le point est classé par le réseau en 1 et qu'il s'agit d'une erreur, on l'affiche en orange

Si le point est classé par le réseau en 2 et qu'il s'agit d'une erreur, on l'affiche en violet.

Initialisez le réseau **avec 2 entrées** (!), avec 1 ou 2 couches cachées et faites varier le nombre de neurones de la couche cachée pour observer l'impact sur le résultat de la classification (itérez jusqu'à convergence).

Est-il possible d'obtenir 0 erreur ? Pourquoi ?

3.2 Modifiez la fonction modèle de sorte que la classe 1 corresponde à l'union des points situés dans 4 cercles différents :

- cercle 1 : centre en (200,200) rayon 100
- cercle 2 : centre en (400,200), rayon 100 // cercle serré sur l'autre, pour voir l'impact
- cercle 3 : centre en (200,400), rayon 100
- cercle 4 : centre en (500,500), rayon 20 // pour voir l'impact d'un petit cercle

Est-ce que les points de tous les cercles sont correctement classés ? Quel est l'impact sur la vitesse de convergence et le nombre minimal de neurones sur la couche cachée pour obtenir des résultats satisfaisants ?

Exercice 4, pour les plus rapides.

Un ensemble d'apprentissage est fourni dans un fichier nommé datasetclassif.txt

Il s'agit de points dont les coordonnées varient entre 0 et 800. Dans le fichier, la première valeur est le numéro du point, suivi de 2 valeurs réelles correspondant aux coordonnées réelles. Dans le cadre de cet apprentissage, on connait à l'avance la classification recherchée : les 1500 premiers points appartiennent à la classe 1 et les 1500 suivants à la classe 2. L'objectif, après apprentissage, est que le perceptron apprenne à classer n'importe quel couple (x,y) pour des valeurs de x et y quelconques dans l'intervalle [0 ; 800]. Pour cela, un autre fichier vous est fourni comprenant 640 000 points (800x800) avec la classe attendue pour chacun des points, à savoir :

1 si c'est un point des régions de la classe 1

2 si c'est un point des régions de la classe 2

3 pour les points situés dans les autres régions, qu'on pourra qualifier de points ambigus.

Le fichier de vérification est intitulé datasetverif.txt

4.1 Ajoutez une fonctionnalité au programme qui permet de

- charger les points de ce fichier dans la liste des vecteurs d'entrées du perceptron,
- déterminer les sorties attendues pour chacun d'eux (par exemple 0.2 pour la classe 1 et 0.8 pour la classe 2) en fonction du rang (avant ou après 1500),
- procéder à un mélange des vecteurs d'entrée pour éviter le biais dû à l'apprentissage d'une longue liste de points de la même classe (on peut alterner par exemple)
- appeler la fonction backprop du réseau pour effectuer l'apprentissage
- comme pour l'exercice 3, tester tous les points d'une image et afficher de façon colorée les résultats de la classification, avec par exemple 5 couleurs (classe 1, classe 2, classe 3, mal classés dans classe 1, mal classés dans classe 2)

4.2 Matrice de confusion

Adaptez/ajoutez sur l'interface les informations quantitatives de la matrice de confusion :

- Nombre de points de la classe 1 bien classés
- Nombre de points de la classe 2 bien classés
- Nombre de points attribués à la classe 1 par erreur
- Nombre de points attribués à la classe 2 par erreur

4.3 Gestion des points incertains

La sortie du réseau est parfois proche de 0.5 et la classification avec un seuil à 0.5 est donc arbitraire. Identifiez tous ces cas de classification incertaine en attribuant un code couleur différent si la sortie est entre 0.4 et 0.6, et recalculez la matrice de confusion.

ANNEXE : Les fichiers en C#

Exemple de lecture d'un fichier texte.

Supposons qu'on définisse la class ClassPersonne de la manière suivante :

```
  public class ClassPersonne
    {
        private string nom;
        private int age;
        public void SetNom( string newnom) { nom = newnom;    }
        public void SetAge( int newage) {  age = newage;  }
        public string GetNom(){ return nom; }
        public int GetAge(){ return age; }
    }
```

Lecture du fichier : placement des personnes dans une liste et affichage dans un ListBox :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
// Il faut ajouter les bibliothèques d'E/S
using System.IO;
namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()  { InitializeComponent();  }

        private List<ClassPersonne> l = new List<ClassPersonne>();

        private void button1_Click(object sender, EventArgs e)
        {
            // Si le fichier n'existe pas, on ne va pas plus loin !
            if (File.Exists("c:/toto.txt") == false) return;
            // Création d'une instance de StreamReader pour permettre la
            // lecture dans le fichier dont le nom est le 1er paramètre.
            // NB : il existe un éventuel 2ème paramètre pour spécifier le
            // codage du texte
            StreamReader monStreamReader = new StreamReader("c:/toto.txt");

            // Lecture du fichier avec un while, évidemment !
            string ligne = monStreamReader.ReadLine();
            while (ligne != null)
            {
                ClassPersonne p = new ClassPersonne();
                p.SetNom( ligne );
                ligne = monStreamReader.ReadLine();
                p.SetAge( Convert.ToInt32(ligne));
                listBox1.Items.Add(p.GetNom()+" ("+ligne+")");
                ligne = monStreamReader.ReadLine();
            }
            // Fermeture du StreamReader (obligatoire)
            monStreamReader.Close();
        }
    }
}
```