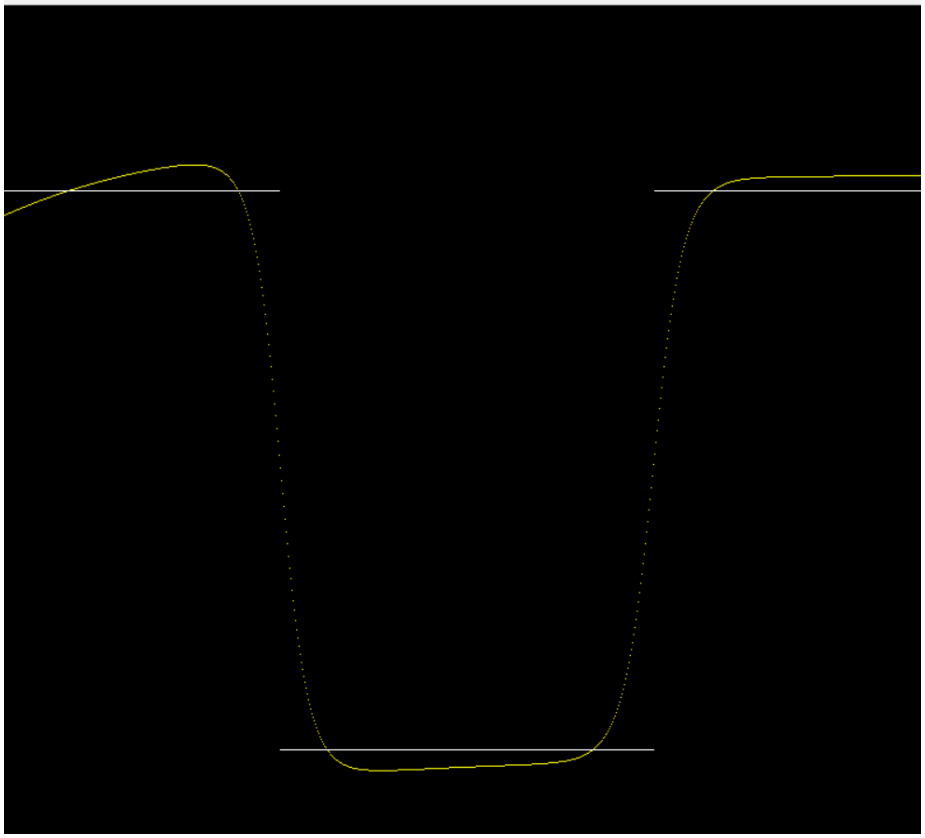
Commentaires sur le TP perceptron multi-couches

Exercice 1 : classification 1D d'une fonction pré-programmée.

On remarque qu'avec 0 couche cachée, la classification sépare les classes avec une droite (voir TP perceptron 1 neurone), ce qui n'est pas satisfaisant. Avec 1 couche cachée, on constate que la convergence commence à être relativement correcte pour 3 neurones sur la couche cachée (voir ci-dessous). Avec plus de neurones, on parvient à plus de précision en général, avec une courbe qui présente de nombreux points d'inflexion (changement de courbure) mais le gain est faible et il faut attendre plus longtemps la convergence.

Si on choisit une plus grande valeur pour le coefficient d'apprentissage, par exemple la valeur 10, les poids du réseau sont modifiés à chaque fois 10 fois plus, la convergence est donc plus rapide, mais l'inconvénient majeur est que les modifications causées par l'erreur observée sur le dernier exemple présenté ont un impact très important. C'est pour ça qu'il y a parfois un rehaussement ou un abaissement significatif de toute la courbe à l'itération suivante. Si on prend au contraire un coefficient d'apprentissage plus faible, ça bouge moins, mais la convergence est plus lente.

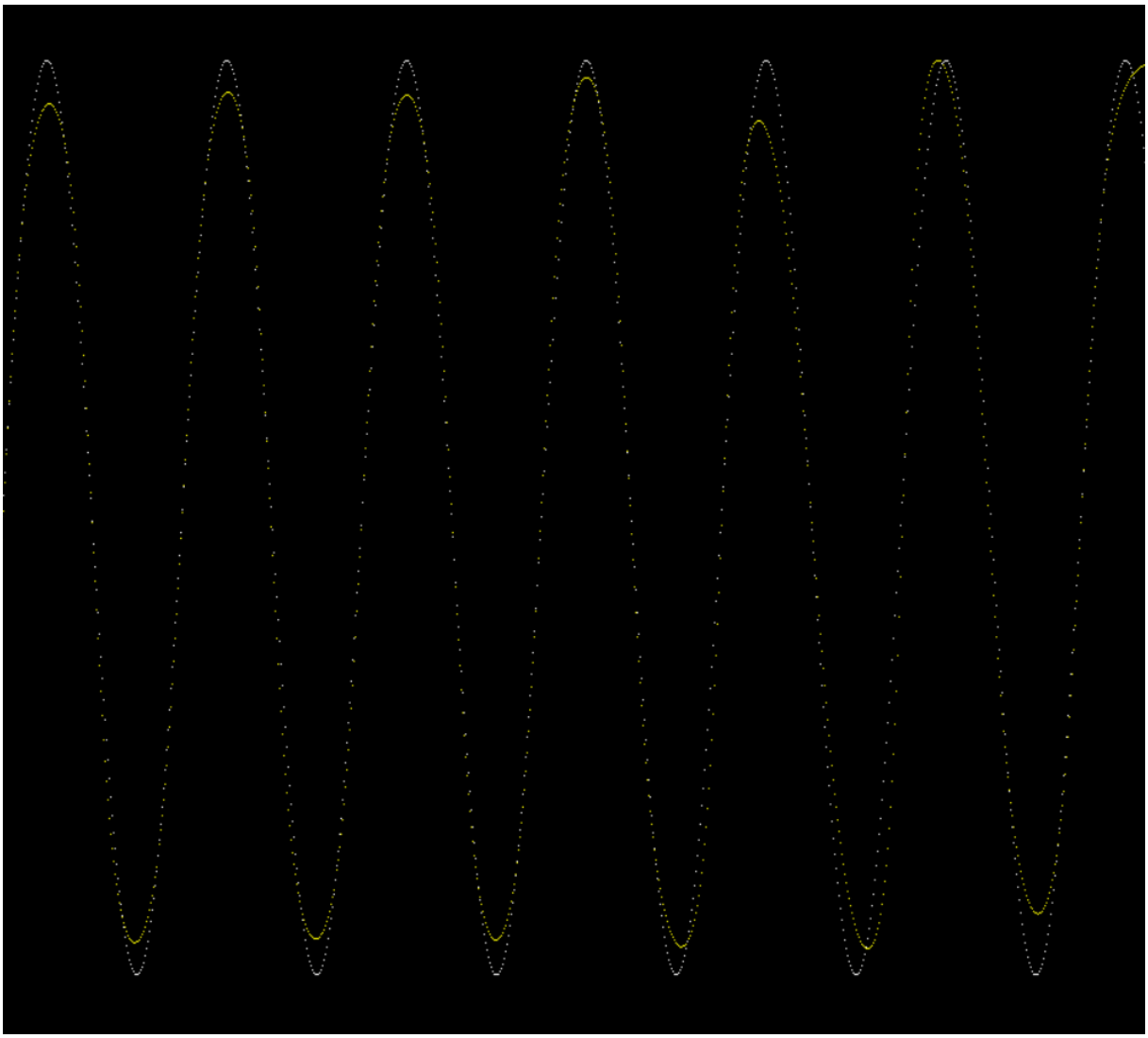


Résultat obtenu avec 1 couche cachée et 3 neurones sur la couche cachée, coefficient d'apprentissage égal à 1.

Exercice 2 : régression 1D

L'apprentissage d'une courbe fortement sinusoïdale révèle des propriétés intéressantes du perceptron. En effet, on constate grosso modo que le nombre de pics de la fonction à apprendre correspond au nombre de neurones nécessaires dans la couche cachée. Par exemple, avec 5 neurones, le perceptron arrive à apprendre 5 pics, mais pas plus, avec 10 neurones, 10 pics etc.

Avec 1 couche cachée et 13 neurones et coefficient d'apprentissage égal à 1, on arrive ainsi, de justesse, à approcher la courbe sinusoïdale avec 13 pics. Voici le résultat obtenu :



Pourquoi cette corrélation entre le nombre de neurones et le nombre de courbures ? Ce phénomène n'est pas clairement documenté dans la littérature. A priori, l'explication est que les poids de la première couche permettent de coder un segment de droite (2 poids par neurone, venant de l'entrée x et de la constante), et que les poids de la couche au-dessus permettent de faire des combinaisons de ces segments.

Avec 2 couches cachées et 6 neurones par couche cachée, on obtient des résultats similaires. Il semble même que 2 couches cachées permettent de gagner en précision, en particulier au niveau des pics

Remarque : parfois, la courbe n'apparait pas, la convergence ne se fait pas correctement, ou même le résultat finit par se dégrader. Je pense que c'est dû à des valeurs de poids trop fortes ou trop faibles par rapport aux autres qui finissent par saturer et empêcher la convergence, notamment quand le coefficient d'apprentissage est grand. On peut aussi l'interpréter comme un effet de bord dû à un surapprentissage sur certaines valeurs. Dans le polycopié, il est indiqué qu'il existe des techniques qui intègrent la valeur des poids dans l'erreur pour empêcher cette saturation, mais elles n'ont pas été utilisées ici.

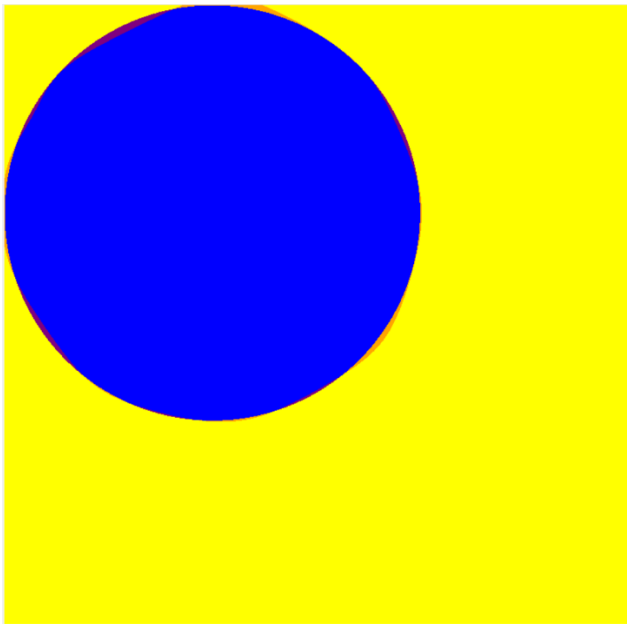
Exercice 3 : apprentissage classification 2D

3.1 On obtient rapidement de bons résultats. Par exemple, avec 2 couches cachées et 6 neurones par couche cachée, on parvient à avoir moins de 1% d'erreur de classification et un taux d'erreur de précision de l'ordre de 3%, voir résultat ci-dessous.

Première observation importante : on obtient de bons résultats, le réseau n'a aucune difficulté à apprendre à différencier les points à l'intérieur du cercle de ceux qui sont en dehors.

Deuxième observation : la convergence est lente.

Est-il possible d'avoir 0 erreur (question du TP) ? Les 1000 points servant à l'apprentissage sont répartis un peu partout dans l'image. Il y a certainement quelques points proches de la frontière du cercle, mais il en manque probablement beaucoup. Autrement dit, sur les données d'apprentissage, il est possible qu'on ait très peu, voire 0 erreur de classification. Les erreurs sur les points de l'image sont dues au fait que les points de la frontière n'étaient pas tous dans les données d'apprentissage. Et donc, on peut difficilement blâmer l'algorithme pour ces erreurs. C'était à nous d'intégrer les points difficiles à classer dans les données d'apprentissage. Il est donc normal de ne jamais parvenir à 0 erreur.



3.2 Classification avec 4 cercles

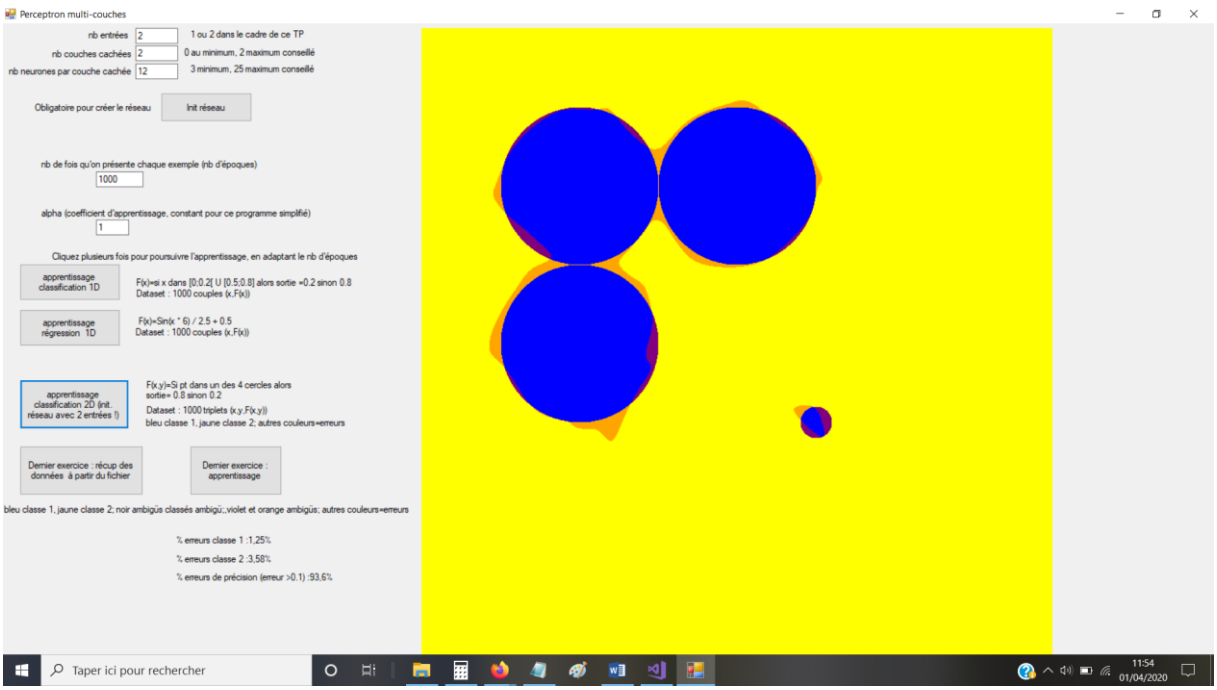
Il n'y a pas de difficulté majeure pour l'écriture du code.

On remarque qu'il faut de nombreux neurones et que ça converge lentement vers une classification qui tient de mieux en mieux compte des zones compliquées à la frontière des cercles. On remarque également que le petit cercle semble ne jamais pouvoir être classé correctement. On y parvient cependant parfois, au prix d'une architecture assez complexe et d'un très grand nombre d'itérations, par exemple 2 couches cachées, 12 neurones par couche cachée et 25000 itérations, à condition d'avoir assez de points d'apprentissage dans cette zone, ce qui est aléatoire.

Pourquoi cette difficulté avec le petit cercle ? La raison n'est pas évidente. Il semblerait que les modifications incessantes sur tous les points d'apprentissage, conduisent à un lissage important. Les

points d'apprentissage ne sont pas équilibrés, il y a beaucoup plus d'exemples de la classe hors des cercles que de la classe dans les cercles. Il faut donc attendre que les points autour du petit cercle soient proche de la valeur exigée (0,8 et pas 0,7 ou 0,75 …) de sorte que les modifications de poids deviennent faibles ou se compensent et que les modifications de poids pour les points du petit cercle puissent enfin avoir un impact.

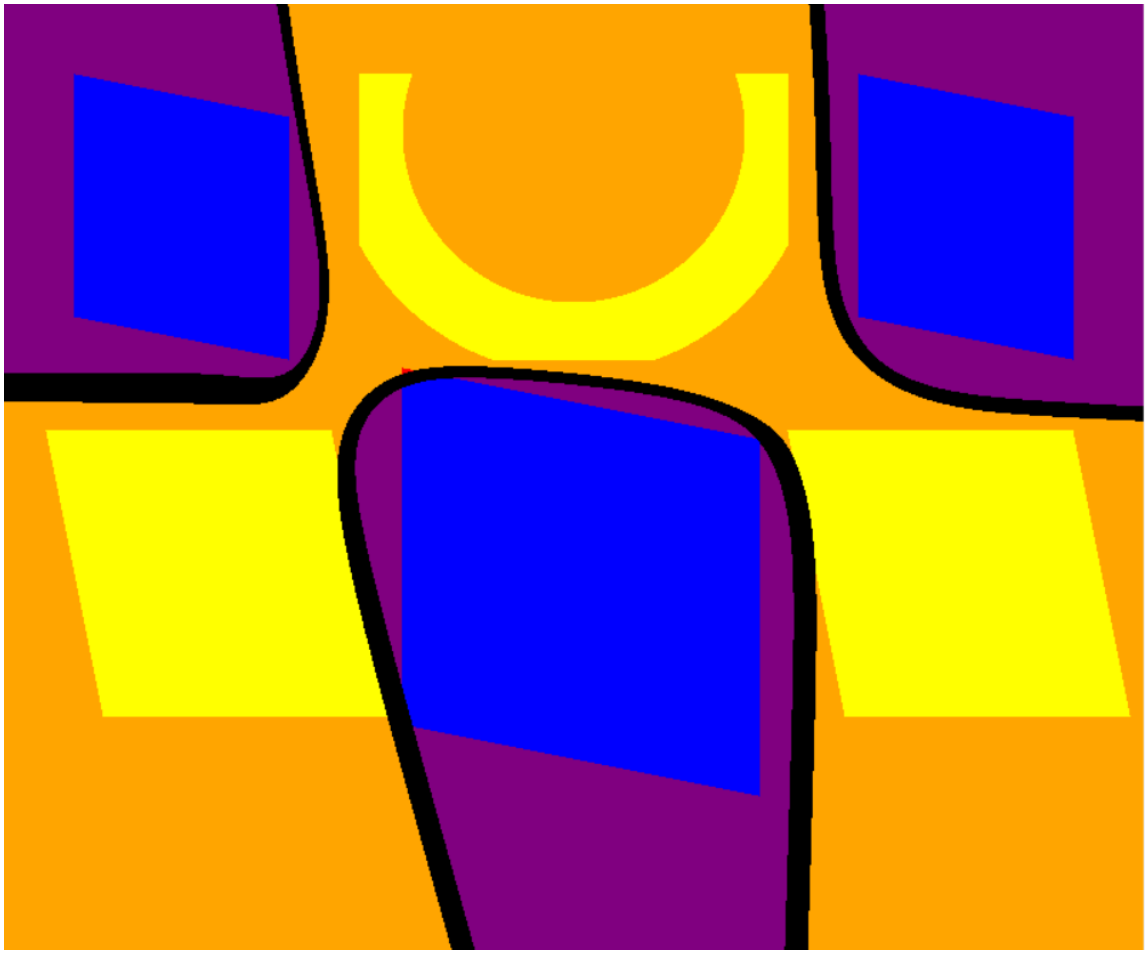
Théoriquement, on pourrait avoir une convergence bien plus rapide si on évitait de faire des modifications de poids lorsque la classification est nette et correcte (en dessous de 0,4 pour la classe 1 et au-dessus de 0,6 pour la classe 2 par exemple) sans chercher à atteindre absolument les 0,2 et 0,8. Cette option a été implémentée et testée avec succès (dans la méthode backprop de la classe reseau, on teste la différence entre la sortie désirée et la sortie obtenue et si c'est moins de 0,2 on ne modifie pas les poids. Ca converge beaucoup mieux et plus rapidement (on divise par 3 ou 4 les temps de calculs. Voici le résultat pour 2 couches cachées, 12 neurones par couche cachée, alpha =1 et environ 15000 itérations.



Exercice 4 :

Cet exercice était difficile.

Il y avait 2 fichiers. Le premier rassemblait des données d'apprentissage de 2 classes et le deuxième des données de test en intégrant une 3ème classe pour des points jugés non classables. Pour l'apprentissage, les points de la première classe étaient situés dans 3 zones distinctes correspondant à 3 parallélogrammes (un ensemble de points dans ces parallélogrammes). Pour la 2ème classe, les points étaient situés dans 3 zones distinctes correspondant à une sorte de couronne et 2 autres parallélogrammes. Voir résultat ci-dessous.



Résultats obtenus pour 2 couches cachées et 6 neurones par couche cachée. En jaune, les points de la classe 1 correctement classés, en bleu les points de la classe 2 correctement classés. Pour les tests, tous les autres points étaient classés « classe 3 » et correspondaient à une classe ambigüe. Si l'algorithme les classait en 1 ou 2, ce n'était pas considéré comme une erreur (points orange ou violet). Les points noirs correspondent à des points non classés par l'algorithme (valeurs entre 0,4 et 0,6). Il reste un peu de rouge, là ce sont de vraies erreurs de classification.

Interprétation : les résultats sont bons, les 2 classes sont bien séparées malgré leur proximité. Par ailleurs, l'algorithme laisse beaucoup de points non classés (valeurs entre 0,4 et 0,5), ce sont les points noirs. Ces points noirs sont placés à la frontière entre les classes, ce qui est intéressant. De manière générale, le perceptron fournit des valeurs incertaines pour les points à la frontière, ce qui peut être avantageusement exploité pour identifier les cas difficiles, et éventuellement adapter le processus de décision qui y serait lié.