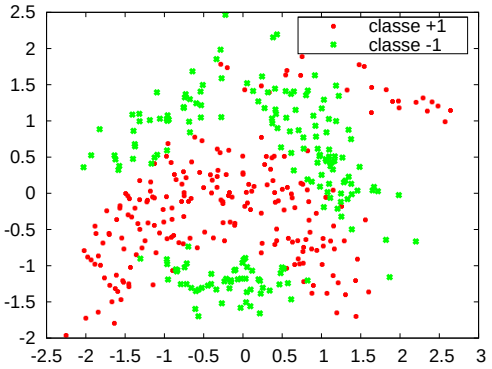
Adaboost et descente de gradient

1 Adaboost

Dans cet exercice, il est demandé de programmer l'algorithme Adaboost.

1. Télécharger l'archive banana.tar.bz2 disponible à l'url
   `http://www.lif.univ-mrs.fr/~liva/DONNEES/banana.tar.bz2`
   Cette archive contient 6 fichiers :
   – 2 fichiers pour l'apprentissage :
   `banana_train_data_1.asc` et `banana_train_labels_1.asc`
   – 2 fichiers pour le test :
   `banana_test_data_1.asc` et `banana_test_labels_1.asc`
   – et 2 fichiers qui contiennent séparément les données positives et les données négatives
   `bananaplus.asc` et `bananamoins.asc`
   Les fichiers `*data_1.asc` contiennent les descriptions des points à classer et les fichiers `*labels_1.asc` contiennent les classes correspondantes (il y a autant de lignes dans les fichiers `*data` que dans leurs fichiers `*labels` correspondant).
2. En utilisant le logiciel de votre choix (Gnuplot fait très bien l'affaire), représenter les points d'apprentissage. (Le jeu de données est constitué d'exemples dans $\mathbb{R}^2$, il est donc facile de les représenter.) Normalement, vous devez obtenir un graphique comme celui qui suit :



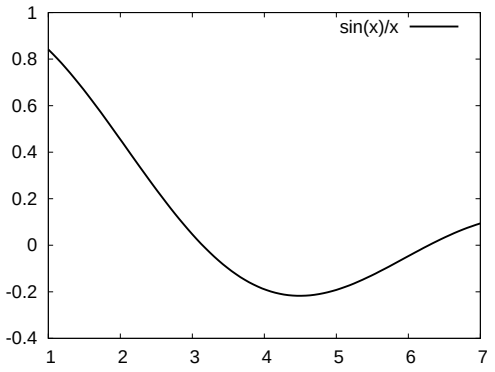
   Vous pourrez utiliser les fichiers `bananaplus.asc` et `bananamoins.asc` fournis dans l'archive.
3. Les classifieurs faibles utilisés sont des stumps, c'est-à-dire des arbres de décision à 1 niveau, comme vu en cours. Comment déterminer les seuils de ces stumps à partir des données d'apprentissage ?
4. Ecrire une fonction qui, étant donné des exemples étiquetés d'apprentissage, accompagnés de leur poids, fournit le stump qui réalise le minimum de l'erreur de classification pondérée.
5. Ecrire l'algorithme Adaboost. Le faire tourner sur les exemples d'apprentissage et mesurer la qualité du classifieur obtenu sur les données de test. Quelle est l'influence de T, le nombre d'itérations d'Adaboost, sur l'erreur de généralisation ?
6. Comment pouvez-vous optimiser votre code ?

L'algorithme Adaboost est un très bon algorithme d'apprentissage. Il peut servir de base à ce que vous mettrez en œuvre pour votre projet de robot.

2 Descente de gradient

Dans cet exercice on s'intéresse à approcher le minimum de la fonction $f(x) = \sin(x)/x$ sur l'intervalle $[1, 7]$ par une méthode de gradient.

Cette fonction a l'aspect suivant :



où l'on voit que sur l'intervalle retenu, la fonction n'admet qu'un minimum global (aux alentours de 4.5).

1. Ecrire une fonction qui calcule la valeur de f en n'importe quel point de l'intervalle de recherche.
2. Ecrire une fonction qui calcule la dérivée de f en n'importe quel point de l'intervalle de recherche.
3. Ecrire une fonction qui estime la dérivée de f en n'importe quel point de l'intervalle de recherche par :
$$f'(x) \approx \frac{f(x+\varepsilon) + f(x-\varepsilon)}{2\varepsilon},$$
pour $\varepsilon > 0$ suffisamment petit (par exemple $\varepsilon = 10^{-5}$).
4. Programmer un algorithme de descente de gradient, pour trouver le minimum de la f sur l'intervalle $[1, 7]$. L'algorithme est le suivant :
   (a) Initialiser x_0 à une valeur arbitraire de l'intervalle
   (b) Répéter tant $|f'(x_t)| > \theta$ (avec $\theta = 10^{-4}$, par exemple)
   $$x_{t+1} \leftarrow x_t - \eta f'(x_t)$$
   Quelle est l'influence de $\eta > 0$ sur la vitesse de convergence de l'algorithme ? Il est souvent conseillé de choisir une valeur de η adaptative, c'est-à-dire qui change à chaque itération. Proposer une stratégie pour choisir un pas convenable.