TME 4 :Perceptron

1 Implémentation du perceptron

Implémentez la fonction `hinge(datax,datay,w)`, coût du perceptron, et `hinge_grad(datax,datay,w)` son gradient. (indication : utilisez `np.sign` et `np.maximum`).
Complétez le code suivant pour implémenter le perceptron. Commencez par coder la fonction `predict`, puis la fonction `fit`.

```
1 HINGE = Fonction(hinge, hinge_grad,0)
2
3 class Perceptron:
4     def __init__(self,loss=HINGE,max_iter=100,eps=0.01):
5         self.max_iter, self.eps = max_iter,eps
6         self.w = None
7         self.loss = loss
8     def fit(self,datax,datay):
9         pass
10    def predict(self,datax):
11        pass
12    def score(self,datax,datay):
13        pass
```

Testez sur un exemple simple (type deux gaussiennes). Tracez la trajectoire de l'apprentissage dans l'espace des poids et les frontières obtenues dans l'espace d'exemple.
Modifier vos fonctions afin de permettre la prise en compte d'un biais.
(bonus) Modifier vos fonctions afin de permettre une descente de gradient stochastique.
(bonus) Implémentez les fonctions `mse` et `mse_g`, moindres carrés et son gradient.

2 Données USPS

Reprenez les données USPS. La fonction ci-dessous permet d'afficher une donnée :

```
1 def show_usps(data):
2     plt.imshow(data.reshape((16,16)),interpolation="nearest",cmap="gray")
```

Sur quelques exemples de problèmes 2 classes (6 vs 9, 1 vs 8 par exemple), entrainez votre perceptron et visualisez la matrice de poids obtenue. Que remarquez vous ?

3 Données 2D et projection

Testez votre perceptron sur les autres données artificielles. Que remarquez vous ? Est-ce normal ?
Codez une fonction de projection polynomiale des données comme vu en TD. Faites les expériences et tracer les frontières.
Codez une fonction de projection gaussienne et expérimentez. Vaut-il mieux beaucoup de points ou peu de points pour la base de projection ?