

# TD : descente de gradient

## 1 Exercices introductifs (non évalués)

- Comprendre et implémenter le code suivant et observer le comportement de l'algorithme en faisant varier la valeur du pas de gradient (`step_size`)

```
def optimise_univariate(step_size=0.4,max_epochs=30):  
    """  
    Optimizes  $f(x) = (x-3)^2$  (strictly convex)  
    """  
    x = 0  
    for e in range(max_epochs):  
        x -= step_size * 2*(x-3) #  $f'(x) = 2(x-3)$   
        obj = (x-3)**2  
        print(x,obj)  
    return x
```

- Comprendre et implémenter le code suivant et observer le comportement de l'algorithme en faisant varier la valeur du pas de gradient (`step_size`)

```
def optimise_bivariate(step_size=0.4,max_epochs=30):  
    """  
    Optimizes  $f(x_1,x_2) = x_1^2+x_2^2+2x_1+8x_2$  (strictly convex)  
    """  
    (x1,x2) = (0,0)  
    for e in range(max_epochs):  
        obj = x1**2+x2**2+2*x1+8*x2  
        print((x1,x2),obj)  
        x1,x2 = (x1-step_size*(2*x1+2),x2-step_size*(2*x2+8))  
  
    obj = x1**2+x2**2+2*x1+8*x2  
    print((x1,x2),obj)  
    return (x1,x2)
```

- Lire le code de la classe `LogisticModel` dans le fichier `numericGD.py`. Vérifiez que vous comprenez l'implémentation de la méthode `train`

## 2 Exercices notés (à faire en binome)

Télécharger le corpus `Sequoia` (v6.0) et travailler avec le fichier `sequoia-corpus.np_conll`

1. Faire une fonction `split(nomfichier)` qui lit un corpus et écrit un corpus d'entraînement, un corpus de développement et un corpus de test.

2. Faire une fonction `read_corpus(nomfichier)` qui lit les données d'un corpus à partir d'un nom de fichier et qui renvoie un jeu de données d'entraînement pour votre classifieur.
3. Faire une classe `AvgPerceptron` avec au moins une fonction appelée `train(self,dataset,step_size=0.1,max_epochs)` qui implémente une fonction d'entraînement d'un perceptron moyenné multiclasse pour prédire les tags des mots de votre corpus ainsi qu'une fonction `test(self,dataset)` qui renvoie l'exactitude de votre classifieur sur le jeu de données passé en paramètres.

**Contrainte** Vous utiliserez la classe `SparseWeightVector.py` comme support pour implémenter le vecteur de poids du perceptron. Vous pouvez consulter le fichier `Multiclass.py` comme source d'inspiration.

### 3 Procédure de notation

On attend le code qui répond aux exercices (1,2,3) ci-dessus enregistré dans un fichier nommé `exo1-nombinomes.py`. La procédure de notation sera partiellement automatisée et exécutera (au moins) la séquence d'instructions suivantes.

```
split('sequoia-corpus.np_conll')
trainc = read_corpus('sequoia-corpus.np_conll.train')
devc    = read_corpus('sequoia-corpus.np_conll.dev')
testc   = read_corpus('sequoia-corpus.np_conll.test')
p = AvgPerceptron()
p.train(trainc,devc)
print(p.test(testc))
```