INF8225 - Intelligence artificielle : techniques probabilistes et d'apprentissage

Hiver 2019

TP No. [1]

Groupe [2]

[1923715] – [BETTACHE Lyes Heythem]

[03-02-2019]

Partie I : réseaux Bayésiens

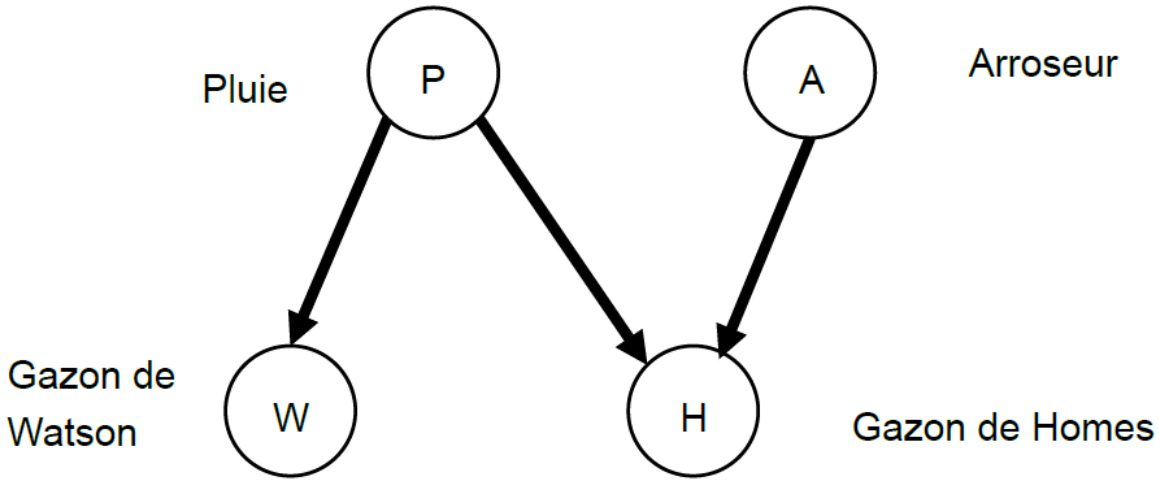


Figure 1: « The Wetgrass Network »

$$P(P,A,W,H) = P(W\ /\ P)P(H\ /\ \ P,A)P(P)P(A$$

$$P(V_1,V_2,\ldots,V_n) = \prod_{i=1}^{n} P(V_i \mid \text{parents}(V_i))$$

Pour plus du détaille sur les calculs numériques voir le fichier **'réseaux Bayésiens'**

Question a

$P(w = 1)$?

$$P(w) = \sum_{P,A,H} P(P,A,W,H) = \sum_{p,a,h} P(W\ /\ P = p)P(H = h\ /\ \ P = p, A = a)P(P = p)P(A = a)$$
$$= \sum_{p} P(W\ /\ P = p)P(P = p) \sum_{a} P(A = a) \sum_{h} P(H = h\ /\ \ P = p, A = a)$$

On remarque que Le nœud W est indépendant des nœuds H , A étant donné le nœud P

$$P(w) = \sum_{P} P(W\ /\ P = p)P(P = p)$$

$$P(w=1) = \sum_{P} P(W=1 \,/\, P=p)P(P=p)$$

Résultat (Voir le code) :

```
Pr(Watson = 1) = 0.36000000000000004
```

Question b

$P(W=1 \,/\, H=1)$?

$$P(W=1 \,/\, H=1) = \frac{P(W=1, H=1)}{P(H=1)}$$

$P(H=1)$?

$$P(H) = \sum_{p,a,w} P(P,A,W,H) = \sum_{p,a,w} P(W=w \,/\, P=p)P(H \,/\, P=p, A=a)P(P=p)P(A=a)$$

$$= \sum_{p} P(P=p) \sum_{a} P(H \,/\, P=p, A=a)P(A=a) \sum_{w} P(W=w \,/\, P=p)$$

On remarque que Le nœud H est indépendant du nœud W étant donné le nœud P et A

$$P(H) = \sum_{P,a} P(H \,/\, P=p, A=a)P(P=p)P(A=a)$$

$$P(H=1) = \sum_{P,a} P(H=1 \,/\, P=p, A=a)P(P=p)P(A=a)$$

$P(W=1, H=1)$?

$$P(W,H) = \sum_{p,a} P(W \,/\, P=p)P(H \,/\, P=p, A=a)P(P=p)P(A=a)$$

$$P(W=1, H=1) = \sum_{p,a} P(W=1 \,/\, P=p)P(H=1 \,/\, P=p, A=a)P(P=p)P(A=a)$$

$P(W = 1 \,/\, H = 1)$?

$$P(W = 1 \,/\, H = 1) = \frac{\sum_{p,a} P(W = 1 \,/\, P = p)P(H = 1 \,/\, P = p, A = a)P(P = p)P(A = a)}{\sum_{P,a} P(H = 1 \,/\, P = p, A = a)P(P = p)P(A = a)}$$

Résultat (Voir le code) :

```
b) Pr(W = 1/H = 1) = 0.788235294117647
```

Question c

$P(W = 1 \,/\, H = 1, A = 0)$?

$$P(W = 1 \,/\, H = 1, A = 0) = \frac{P(W = 1, H = 1, A = 0)}{P(H = 1, A = 0)}$$

$P(W = 1, H = 1, A = 0)$?

$$P(W = 1, H = 1, A = 0) = \sum_{p} P(W = 1 \,/\, P = p)P(H = 1 \,/\, P = p, A = 0)P(P = p)P(A = 0)$$

$P(H = 1, A = 0)$?

$$P(H = 1, A = 0) = \sum_{p,w} P(W = w \,/\, P = p)P(H = 1 \,/\, P = p, A = 0)P(P = p)P(A = 0)$$

$P(W = 1 \,/\, H = 1, A = 0)$?

$$P(W = 1 \,/\, H = 1, A = 0) = \frac{\sum_{p} P(W = 1 \,/\, P = p)P(H = 1 \,/\, P = p, A = 0)P(P = p)P(A = 0)}{\sum_{p,w} P(W = w \,/\, P = p)P(H = 1 \,/\, P = p, A = 0)P(P = p)P(A = 0)}$$

Résultat (Voir le code) :

```
c) Pr(W = 1/H = 1,A = 0) = 0.025
```

Question d

$P(W = 1 \,/\, A = 0)$?

$$P(W = 1 \,/\, A = 0) = \frac{P(W = 1, A = 0)}{P(A = 0)}$$

$P(W = 1, A = 0)$?

$$P(W = 1, A = 0) = \sum_{p,h} P(W = 1 \,/\, P = p) P(H = h \,/\, P = p, A = 0) P(P = p) P(A = 0)$$

$P(A = 0)$?

$$P(A = 0) = \sum_{p,h,w} P(W = w \,/\, P = p) P(H = h \,/\, P = p, A = 0) P(P = p) P(A = 0)$$

$$P(W = 1 \,/\, A = 0) = \frac{\sum_{p,h} P(W = 1 \,/\, P = p) P(H = h \,/\, P = p, A = 0) P(P = p) P(A = 0)}{\sum_{p,h,w} P(W = w \,/\, P = p) P(H = h \,/\, P = p, A = 0) P(P = p) P(A = 0)}$$

Résultat (Voir le code) :

```
d) Pr(W = 1/A = 0) = 0.07111111111111111
```

Question e (dans l'énoncé on a déjà la valeur de Pr(W = 1/P = 1)=1)

$P(W = 1 \,/\, P = 1)$?

$$P(W = 1 \,/\, P = 1) = \frac{P(W = 1, P = 1)}{P(P = 1)}$$

$P(W = 1, P = 1)$?

$$P(W = 1, P = 1) = \sum_{a,h} P(W = 1 \,/\, P = 1) P(H = h \,/\, P = p, A = a) P(P = p) P(A = a)$$

$P(P = 1)$?

$$P(P=1) = \sum_{a,h,w} P(W=w \,/\, P=1)P(H=h \,/\, P=1, A=a)P(P=1)P(A=a)$$

$$P(W=1 \,/\, P=1) = \frac{\sum_{a,h} P(W=1 \,/\, P=1)P(H=h \,/\, P=p, A=a)P(P=p)P(A=a)}{\sum_{a,h,w} P(W=w \,/\, P=1)P(H=h \,/\, P=1, A=a)P(P=1)P(A=a)}$$

Résultat (Voir le code) :

```
e) Pr(W = 1/P = 1) = 1.0
```

Partie 2 La régression logistique et le calcul du gradient

Partie 2-a

a) Montrez les résultats pour différents taux d'apprentissage, e.g. 0.1, 0.01, 0.001, et différentes tailles de minibatch, e.g. 1, 20, 200, 1000.

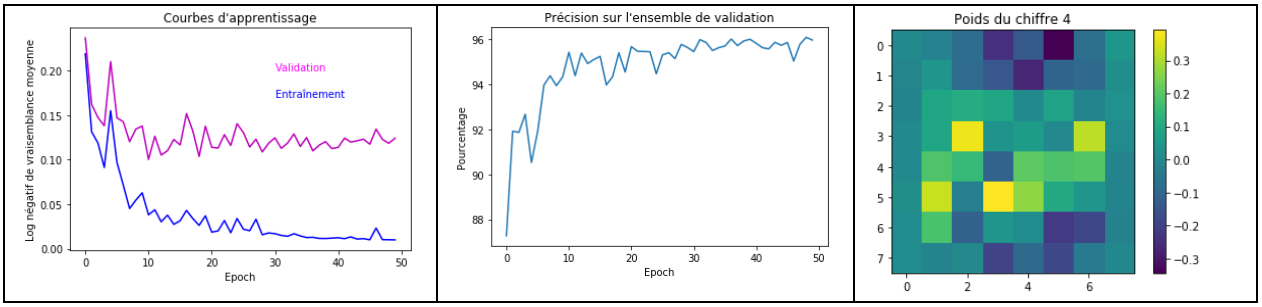
Les pages suivantes, présentés les résultats obtenus avec certaines combinaisons de taux d'apprentissage et de tailles de mini-batch lors de l'exécution d'un code répliquant une descente stochastique de gradients par mini-batches avec la fonction de régression logistique.

Expérience 1 : taux d'apprentissage = 0.001 , élément/mini-batch = 1

```
La précision obtenue =  95.52401577730538

Pour un lr =  0.001

Avec une taille de mini-batch =  1
```

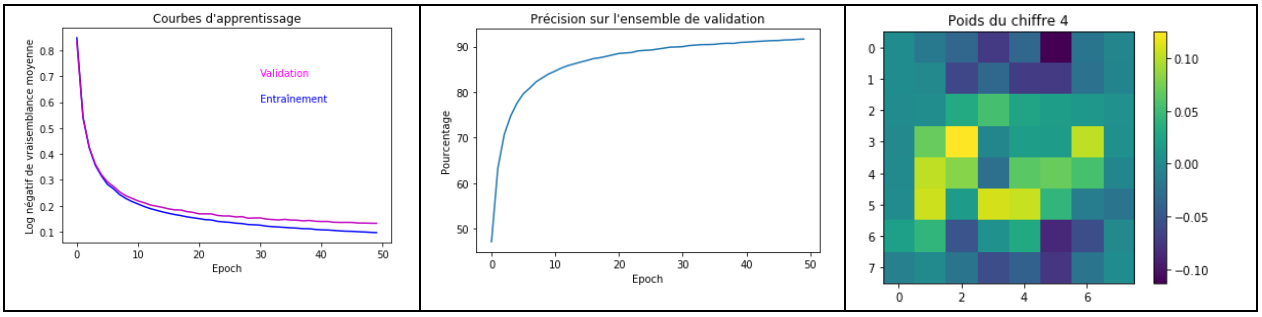


Expérience 2 : taux d'apprentissage = 0.001 , élément/mini-batch=20

```
La précision obtenue =  91.61272956485556

Pour un lr =  0.001

Avec une taille de mini-batch =  20
```

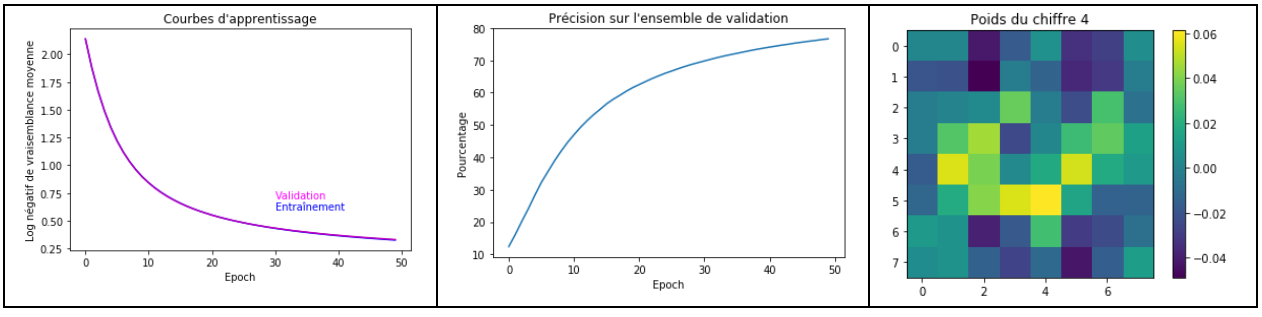


Expérience 3 : taux d'apprentissage = 0.001 , élément/mini-batch=200

```
La précision obtenue =  76.81875430500654

Pour un lr =  0.001

Avec une taille de mini-batch =  200
```

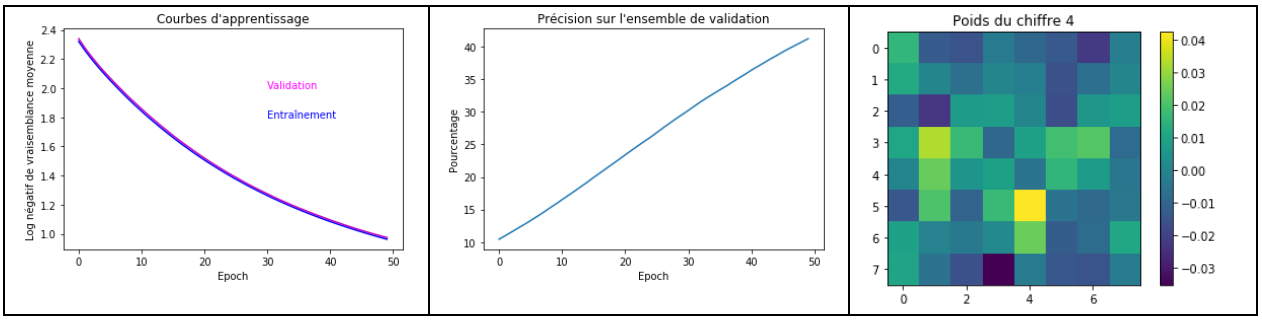


Expérience 4 : taux d'apprentissage = 0.001 , élément/mini-batch = 1000

```
La précision obtenue =  40.77929510400232

Pour un lr =  0.001

Avec une taille de mini-batch =  1000
```

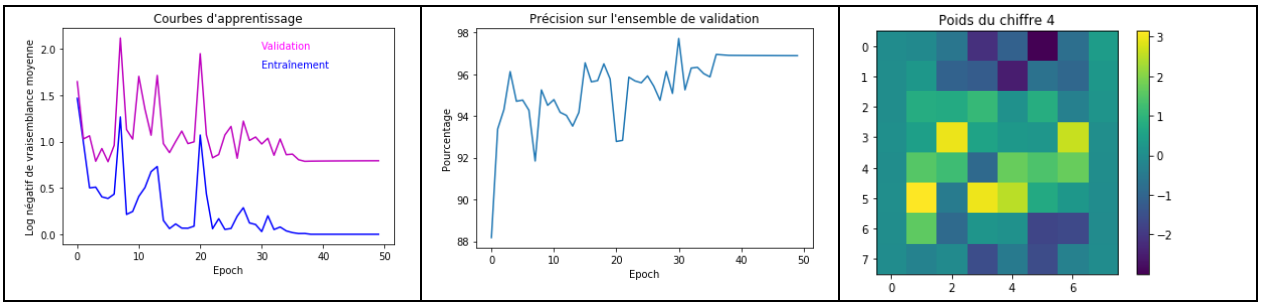


Expérience 5 : taux d'apprentissage = 0.01 , élément/mini-batch= 1

```
La précision obtenue =  95.42194885062183

Pour un lr =  0.01

Avec une taille de mini-batch =  1
```

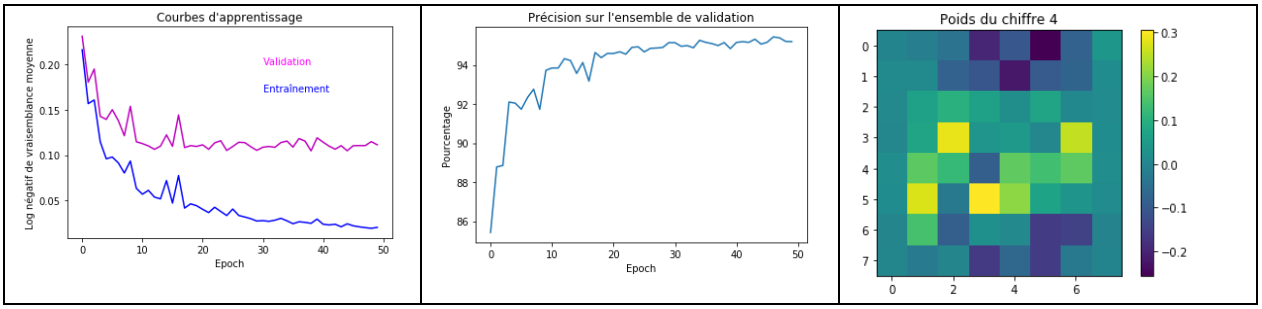


Expérience 6 : taux d'apprentissage = 0.01 , élément/mini-batch =20

```
La précision obtenue =  95.61926379863827

Pour un lr =  0.01

Avec une taille de mini-batch =  20
```

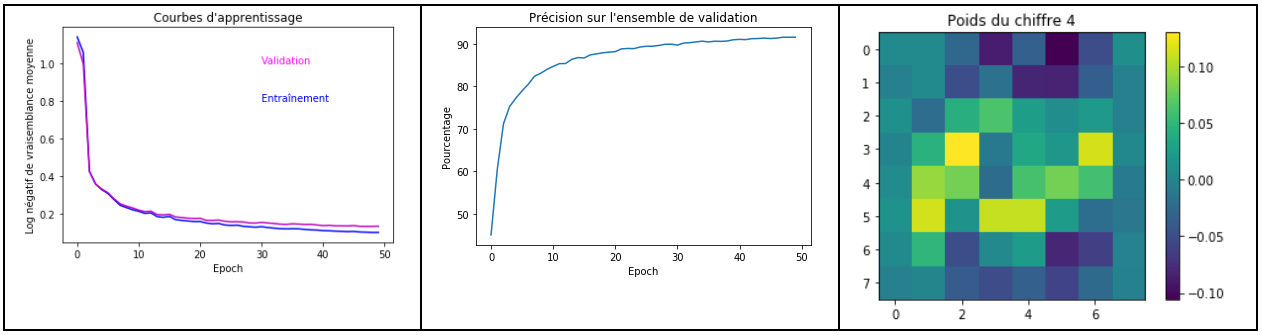


Expérience 7 : taux d'apprentissage = 0.01 , élément/mini-batch =200

```
La précision obtenue =  91.84000079618745

Pour un lr =  0.01

Avec une taille de mini-batch =  200
```

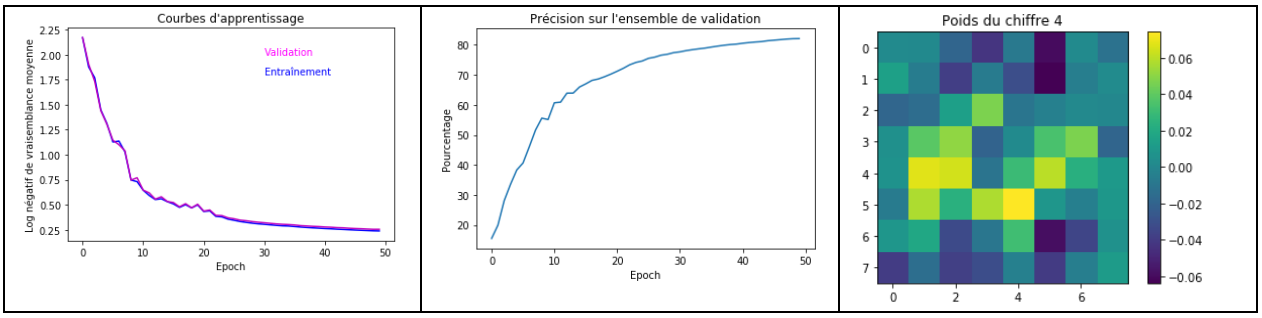


Expérience 8 : taux d'apprentissage = 0.01 , élément/mini-batch =1000

```
La précision obtenue =  82.4959160359478

Pour un lr =  0.01

Avec une taille de mini-batch =  1000
```

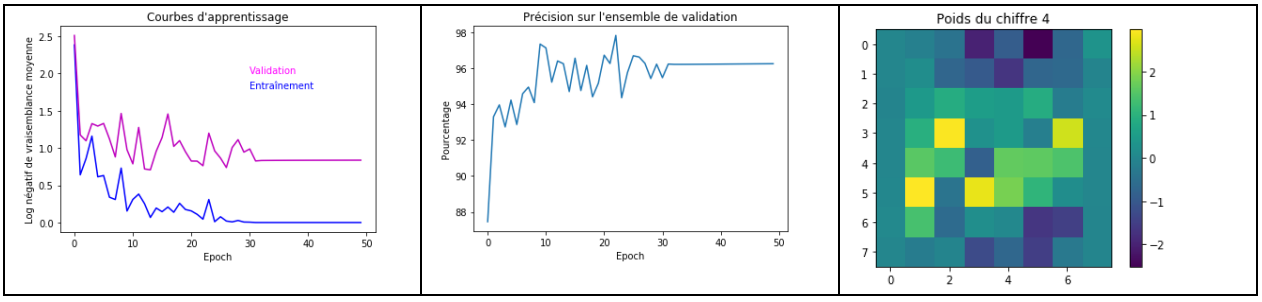


Expérience 9 : taux d'apprentissage = 0.1 , élément/mini-batch =1

```
La précision obtenue =  96.01143339831215

Pour un lr =  0.01

Avec une taille de mini-batch =  1
```

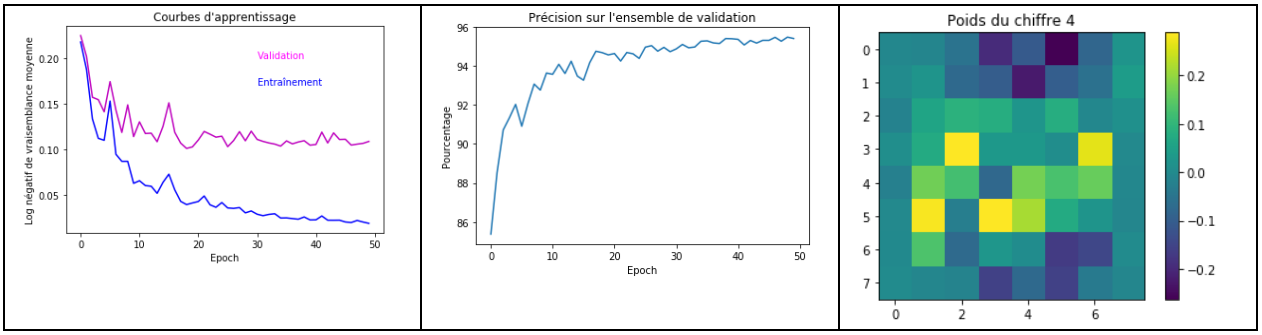


Expérience 10 : taux d'apprentissage = 0.1 , élément/mini-batch = 20

```
La précision obtenue =  95.27038478442759

Pour un lr =  0.01

Avec une taille de mini-batch =  20
```

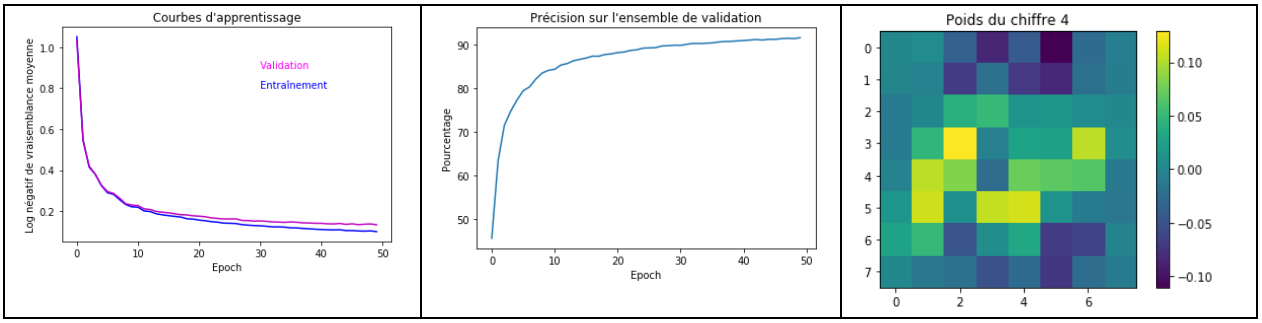


Expérience 11 : taux d'apprentissage = 0.1 , élément/mini-batch = 200

```
La précision obtenue =  91.32347846763156

Pour un lr =  0.01

Avec une taille de mini-batch =  200
```

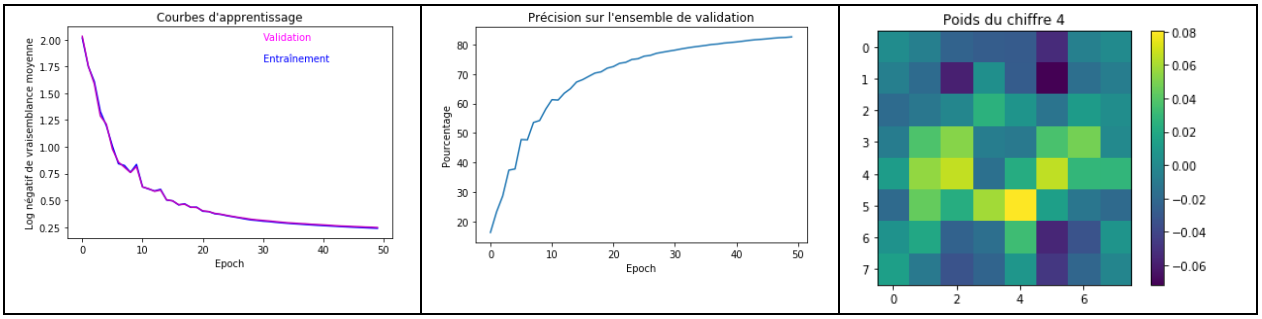


Expérience 12 : taux d'apprentissage = 0.1 , élément/mini-batch =1000

```
La précision obtenue =  82.51866540445253

Pour un lr =  0.01

Avec une taille de mini-batch =  1000
```



Interprétation:

La première figure présente la précision obtenue sur l'ensemble de test et les paramètres de l'expérience.

La seconde figure présente :

1- Le graphe des pertes observées par log négatif (où la courbe en bleu présente les pertes sur l'ensemble de d'entraînement et la courbe en magenta présente les pertes sur l'ensemble de validation),
2- La précision (accuracy) sur l'ensemble de validation
3- Les poids appris pour le chiffre 4 (on peut aussi présenter les autres)

Conclusion :
On remarque que pour un taux d'apprentissage plus élevé donne en moyenne une meilleure précision sur l'ensemble de test. Par contre, les courbes présentent beaucoup plus de bruit car suite à chaque ajustement des gradients sur une mini-batch les poids sont augmentés ou diminués avec un plus grand impact.

Et on remarque aussi lorsque les mini-batches sont très petites donnes en moyenne une meilleure précision sur l'ensemble de test. Et on remarque que les pertes de l'ensemble de validation sont beaucoup moins bien cohérentes avec celles de l'ensemble d'entraînement,

Partie 2-b

b) Ajouter 8 dimensions supplémentaires à la fin de chaque exemple dans l'ensemble d'entraînement, l'ensemble de validation et l'ensemble de test. Remplissez ces dimensions avec des valeurs aléatoires en utilisant une distribution aléatoire uniforme entre la plus petite valeur et la plus grande valeur parmi toutes les dimensions de l'ensemble de données d'origine.

```
X = digits.data
x_uni= np.random.uniform(np.min(X), np.max(X), (1797,8))

X=np.hstack((X,x_uni))

X_train, X_test, y_train, y_test = train_test_split(X, y_one_hot, test_size=0.3, random_state=42)
X_test, X_validation, y_test, y_validation = train_test_split(X_test, y_test, test_size=0.5, random_state=42)
y = digits.target

y_one_hot = np.zeros((y.shape[0], len(np.unique(y))))
y_one_hot[np.arange(y.shape[0]), y] = 1  # one hot target or shape Nx
W = np.random.normal(0, 0.01, (len(np.unique(y)), X.shape[1]))  # Weights of shape KxL
```

La régularisation du types Elastic Net à votre modèle.

Fonction loss :

$$loss = loss + \lambda_1 \|W\|^2 + \lambda_2 \||W|\|$$

$$\sum_{i=1}^{N} \text{-log}(p(y_i \mid x_i; W, b)) + \lambda_1 \sum_{j=1}^{M} (w_j{}^2) + \lambda_2 \sum_{j=1}^{M} (|w_j|)$$

Fonction Gradient :

$$\frac{\delta}{\delta\theta} log \prod_{i=1}^{N} P(\tilde{y}_i|\tilde{x}_i;\theta) = \sum_{i=1}^{N} \frac{\delta}{\delta\theta}\left\{log\left(\frac{1}{Z(\tilde{x}_i)}\, exp(\tilde{y}_i{}^T \theta \tilde{x}_i)\right\} + \frac{\delta}{\delta\theta}\{\lambda_1\|W\|^2 + \lambda_2\||W|\|\}\right.$$

$$= \sum_{i=1}^{N}\left(\tilde{y}_i\tilde{x}_i{}^T - \sum_{y_k \in Y} P(\tilde{y}_i|\tilde{x}_i;\theta) y_k\, \tilde{x}_i{}^T\right) + \frac{\delta}{\delta\theta}\{\lambda_1\|W\|^2 + \lambda_2\||W|\|\}$$

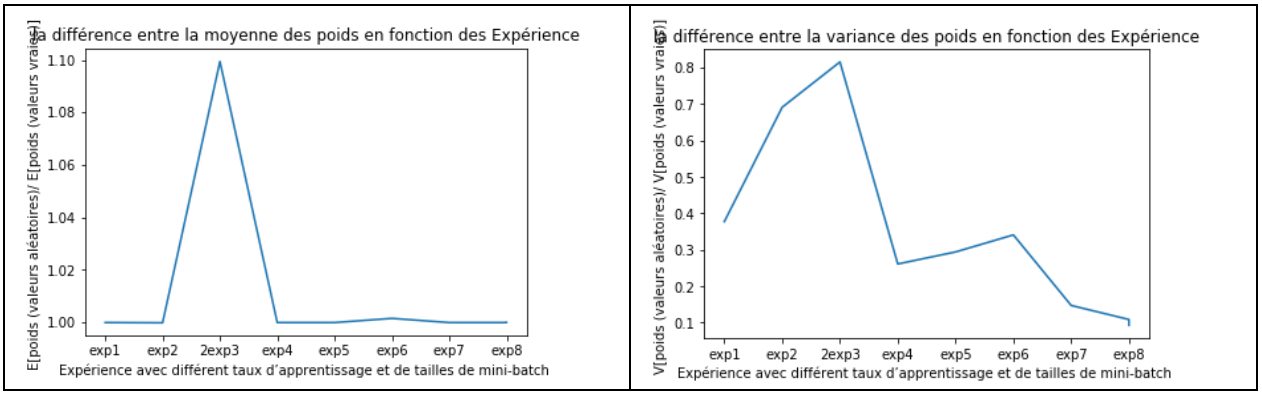
$$= \sum_{i=1}^{N}(\tilde{y}_i\tilde{x}_i{}^T) - \sum_{i=1}^{N}(\hat{y}_i\tilde{x}_i{}^T) + 2\lambda_1\|W\| + \lambda_2 signe(W)$$

$$\textbf{gradient} = \sum_{i=1}^{N}(\tilde{y}_i\tilde{x}_i{}^T) - \sum_{i=1}^{N}(\hat{y}_i\tilde{x}_i{}^T) + 2\lambda_1\|W\| + \lambda_2 signe(W)$$

Le tableau suivant, présenté les coefficients de fonction de perte (landa1 et landa2) optimal obtenus avec certaines combinaisons de taux d'apprentissage et de tailles de mini-batch lors de l'exécution d'un code répliquant une descente stochastique de gradients par mini-batches avec la fonction de régression logistique.

Exp1 La précision obtenue = 93.36786479714118 Pour un lr = 0.001 Avec une taille de mini-batch = 20 best landa1 = 0.1 best landa2 = 0.3	**Exp2** La précision obtenue = 92.23422489214774 Pour un lr = 0.001 Avec une taille de mini-batch = 200 best landa1 = 0.2 best landa2 = 0.6
Exp3 La précision obtenue = 91.01475857056847 Pour un lr = 0.001 Avec une taille de mini-batch = 1000 best landa1 = 0.5 best landa2 = 0.8	**Exp4** La précision obtenue = 94.5521078862512 Pour un lr = 0.01 Avec une taille de mini-batch = 20 best landa1 = 0.1 best landa2 = 0.1
Exp5 La précision obtenue = 92.93304735729834 Pour un lr = 0.01 Avec une taille de mini-batch = 200 best landa1 = 0.1 best landa2 = 0.3	**Exp6** La précision obtenue = 92.58149794989127 Pour un lr = 0.01 Avec une taille de mini-batch = 1000 best landa1 = 0.2 best landa2 = 0.2
Exp7 La précision obtenue = 89.58274396808868 Pour un lr = 0.1 Avec une taille de mini-batch = 20 best landa1 = 0.1 best landa2 = 0.1	**Exp8** La précision obtenue = 95.6678775012582 Pour un lr = 0.1 Avec une taille de mini-batch = 200 best landa1 = 0.1 best landa2 = 0.1
Exp9 La précision obtenue = 94.69744235127128 Pour un lr = 0.1 Avec une taille de mini-batch = 1000 best landa1 = 0.1 best landa2 = 0.2	

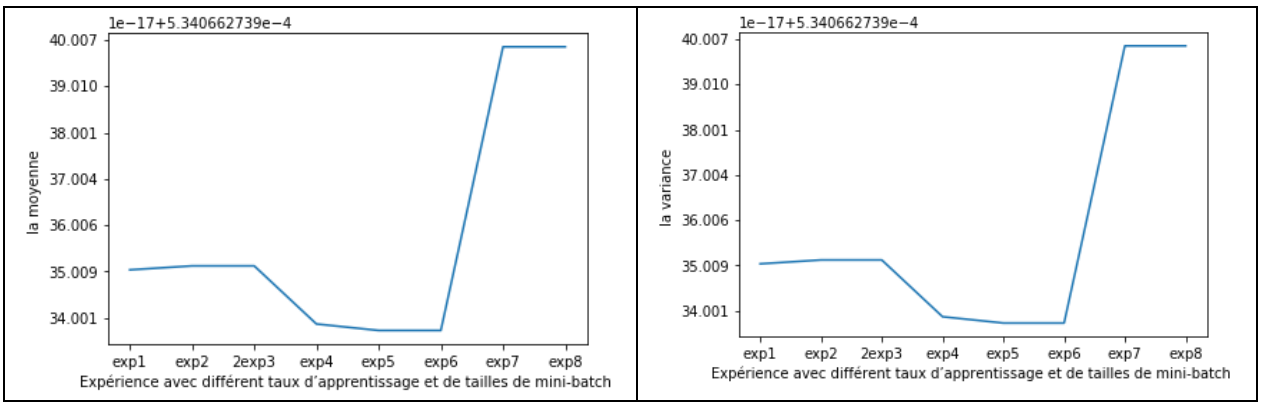
La moyenne et la variance des poids de modèle associés à ces dimensions contenant des valeurs aléatoires par rapport à la moyenne et la variance des poids associés aux dimensions contenant des vraies données



On remarque que pour un coefficient de fonction de perte (landa1) élevé donne une différence de moyenne des poids élevé, donc l'augmentation des coefficients (landa1) implique l'augmentation de la moyenne des poids de modèle associés à ces dimensions contenant des valeurs aléatoires.

On remarque aussi que pour des coefficients de fonction de perte (landa1 et landa2) élevé donne une différence de variance des poids élevé, donc l'augmentation des coefficients (landa1 et landa2) implique l'augmentation de la variance des poids de modèle associés à ces dimensions contenant des valeurs aléatoires.

Moyennes et variances d'un modèle appris sans régularisation



D'après les figures précédentes on remarque que les moyennes et les variances d'un modèle appris sans régularisation sont très petites par rapport un modèle appris avec régularisation

Pour ramènent à zéro les poids associés aux dimensions aléatoires ajoutées, il suffit juste que les coefficients de fonction de perte (landa1 et landa2) convergent vers zéro (sont très petites).