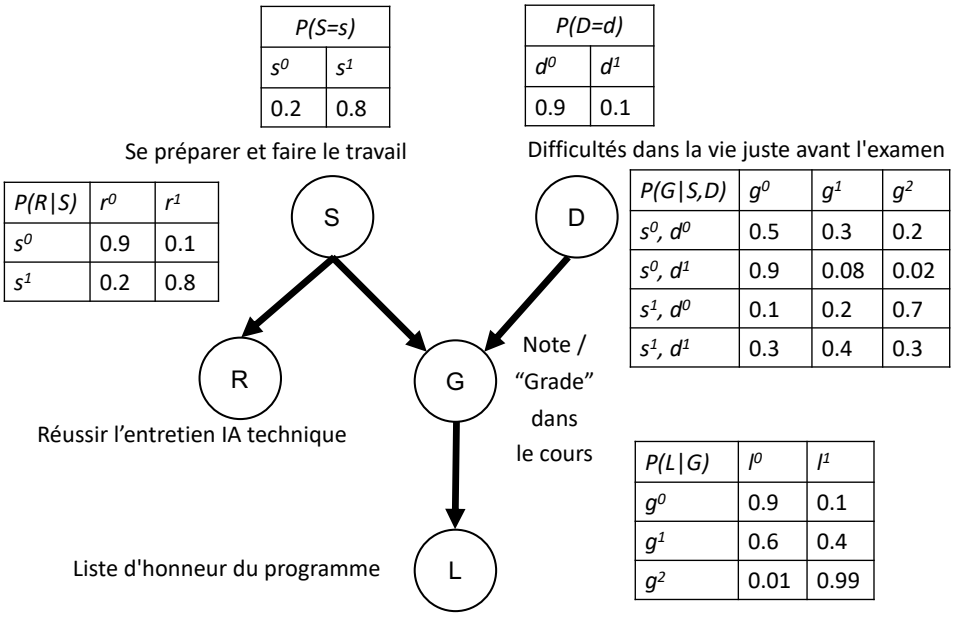
INF8225 TP1 H25

Date limite : Partie 1 et 2 – 20h30 le 6 février 2025, Partie 3 – 20h30 le 20 février 2025

Remettez votre fichier Colab en 2 formats: **.pdf** ET **.ipynb** (jupyter notebook format). Lien vers le Colab: `https://colab.research.google.com/drive/18wBEYJRamoyapvGntxqFqpaOPLX1swpI#scrollTo=cpoOrxiaHV43`

1 Partie 1 (16 points)

L'objectif de la partie 1 du travail pratique est de permettre à l'étudiant de se familiariser avec les réseaux Bayésiens et la librairie Numpy. Considérons le réseau Bayésien ci-dessous.



P(S=s)	
s^0	s^1
0.2	0.8

P(D=d)	
d^0	d^1
0.9	0.1

P(R|S)	r^0	r^1
s^0	0.9	0.1
s^1	0.2	0.8

P(G|S,D)	g^0	g^1	g^2
s^0, d^0	0.5	0.3	0.2
s^0, d^1	0.9	0.08	0.02
s^1, d^0	0.1	0.2	0.7
s^1, d^1	0.3	0.4	0.3

P(L|G)	l^0	l^1
g^0	0.9	0.1
g^1	0.6	0.4
g^2	0.01	0.99

Figure 1: Un modèle simple pour les notes à un examen (G) et sa relation avec les étudiants qui se préparent aux examens et font correctement le travail pour les devoirs (S), les étudiants qui ont des difficultés dans la vie juste avant l'examen final (D), les étudiants qui réussissent bien à un entretien technique pour un emploi axé sur le sujet du cours (R), et des étudiants qui se retrouvent sur une sorte de palmarès de leur programme (L).

1.1 Description des tâches

1.1.1 À l'aide de ces tables de probabilité conditionnelles, calculez les requêtes ci-dessous. Dans les cas où l'on compare un calcul non interventionnel à un calcul interventionnel, commentez sur l'interprétation physique des deux situations et les résultats obtenus à partir de vos modèles.

a) $P(G) = [P(G = g^0), P(G = g^1), P(G = g^2)]$

b) $P(G|R = r^1)$

c) $P(G|R = r^0)$

d) $P(G|R = r^1, S = s^0)$

e) $P(G|R = r^0, S = s^0)$

f) $P(R|D = d^1)$

g) $P(R|D = d^0)$

h) $P(R|D = d^1, G = g^2)$

i) $P(R|D = d^0, G = g^2)$

j) $P(R|D = d^1, L = l^1)$

k) $P(R|D = d^0, L = l^1)$

l) $P(R|do(G = g^2))$

m) $P(R|G = g^2)$

n) $P(R)$

o) $P(G|do(L = l^1))$

p) $P(G = g^1|L = l^1)$

2 Partie 2 (20 points)

L'objectif de la partie 2 du travail pratique est de permettre à l'étudiant de se familiariser avec l'apprentissage automatique via la régression logistique. Nous allons donc résoudre un problème de classification d'images en utilisant l'approche de descente du gradient (gradient descent) pour optimiser la log-vraisemblance négative (negative log-likelihood) comme fonction de perte. Nous réaliserons des expériences en utilisant l'ensemble de données de Fashion MNIST.

L'algorithme à implémenter est une variation de descente de gradient qui s'appelle l'algorithme de descente de gradient stochastique par mini-ensemble (mini-batch stochastic gradient descent). Votre objectif est d'écrire un programme en Python pour optimiser les paramètres d'un modèle étant donné un ensemble de données d'apprentissage, en utilisant un ensemble de validation pour déterminer quand arrêter l'optimisation, et finalement de montrer la performance sur l'ensemble du test.

2.1 Théorie: la régression logistique et le calcul du gradient

Il est possible d'encoder l'information concernant l'étiquetage avec des vecteurs multinomiaux (one-hot vectors), c.-à-d. un vecteur de zéros avec un seul 1 pour indiquer quand la classe $C = k$ dans la dimension k. Par exemple, le vecteur $\mathbf{y} = [0, 1, 0, \cdots, 0]^T$ représente la deuxième classe. Les caractéristiques (features) sont données par des vecteurs $\mathbf{x}_i \in \mathbb{R}^D$. En définissant les paramètres de notre modèle comme : $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_K]^T$ et $\mathbf{b} = [b_1, b_2, \cdots b_K]^T$ et la fonction softmax comme fonction de sortie, on peut exprimer notre modèle sous la forme :

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\mathbf{y}^T\mathbf{W}\mathbf{x} + \mathbf{y}^T\mathbf{b})}{\sum_{\mathbf{y}_k \in \mathcal{Y}} \exp(\mathbf{y}_k^T\mathbf{W}\mathbf{x} + \mathbf{y}_k^T\mathbf{b})} \tag{1}$$

L'ensemble de données consiste de n paires (label, input) de la forme $\mathcal{D} := (\tilde{\mathbf{y}}_i, \tilde{\mathbf{x}}_i)_{i=1}^n$, où nous utilisons l'astuce de redéfinir $\tilde{\mathbf{x}}_i = [\tilde{\mathbf{x}}_i^T 1]^T$ et nous redéfinissions la matrice de paramètres $\boldsymbol{\theta} \in \mathbb{R}^{K\times(D+1)}$ (voir des notes de cours pour la relation entre $\boldsymbol{\theta}$ et $\mathbf{W}$). Notre fonction de perte, la log-vraisemblance négative des données selon notre modèle est définie comme:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) := -\log \prod_{i=1}^{N} P(\tilde{\mathbf{y}}_i|\tilde{\mathbf{x}}_i; \boldsymbol{\theta}) \tag{2}$$

Pour cette partie du TP, nous avons calculé pour vous le gradient de la fonction de perte par rapport aux paramètres du modèle:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = -\sum_{i=1}^{N} \frac{\partial}{\partial\boldsymbol{\theta}} \left\{ \log \left(\frac{\exp(\tilde{\mathbf{y}}_i^T\boldsymbol{\theta}\tilde{\mathbf{x}}_i)}{\sum_{\mathbf{y}_k\in\mathcal{Y}} \exp(\mathbf{y}_k^T\boldsymbol{\theta}\tilde{\mathbf{x}}_i)} \right) \right\} \tag{3}$$

$$= -\sum_{i=1}^{N} \left(\tilde{\mathbf{y}}_i\tilde{\mathbf{x}}_i^T - \sum_{\mathbf{y}_k\in\mathcal{Y}} P(\mathbf{y}_k|\tilde{\mathbf{x}}_i, \boldsymbol{\theta})\mathbf{y}_k\tilde{\mathbf{x}}_i^T \right) \tag{4}$$

$$= \sum_{i=1}^{N} \hat{\mathbf{p}}_i\tilde{\mathbf{x}}_i^T - \sum_{i=1}^{N} \tilde{\mathbf{y}}_i\tilde{\mathbf{x}}_i^T \tag{5}$$

où $\hat{\mathbf{p}}_i$ est un vecteur de probabilités produit par le modèle pour l'exemple $\tilde{\mathbf{x}}_i$ et $\tilde{\mathbf{y}}_i$ est le vrai *label* pour ce même exemple.

Finalement, il reste à discuter de l'évaluation du modèle. Pour la tâche d'intérêt, qui est une instance du problème de classification, il existe plusieurs métriques pour mesurer les performances du modèle la précision de classification, l'erreur de classification, le taux de faux/vrai positifs/négatifs, etc. Habituellement dans le contexte de l'apprentissage automatique, la précision est la plus commune.

La précision est définie comme le rapport du nombre d'échantillons bien classés sur le nombre total d'échantillons à classer:

$$\tau_{acc} := \frac{|\mathcal{C}|}{|\mathcal{D}|}$$

où l'ensemble des échantillons bien classés $\mathcal{C}$ est:

$$\mathcal{C} := \{(\mathbf{x}, \mathbf{y}) \in \mathcal{D} \,|\, \arg\max_k P(\cdot|\tilde{\mathbf{x}}_i; \boldsymbol{\theta})_k = \arg\max_k \tilde{y}_{i,k}\}$$

En mots, il s'agit du sous-ensemble d'échantillons pour lesquels la classe la plus probable selon notre modèle correspond à la vraie classe.

2.2 Description des tâches

2.2.1 Code à compléter

On vous demande de compléter l'extrait de code dans le fichier Colab pour résoudre ce problème. Vous devez utiliser la librairie PyTorch dans cette partie du TP: https://pytorch.org/docs/stable/index.html. Mettez à jour les paramètres de votre modèle avec la descente par *mini-batch*. Exécutez des expériences avec trois différents ensembles: un ensemble d'apprentissages avec 90% des exemples (choisis au hasard), un ensemble de validation avec 10%. Utilisez uniquement l'ensemble de test pour obtenir votre meilleur résultat une fois que vous pensez avoir obtenu votre meilleure stratégie pour entraîner le modèle.

IMPORTANT

L'objectif du TP est de vous faire implémenter les mathématiques de la descente de gradient à la main. **Il est donc interdit d'utiliser les capacités de construction de modèles ou de différentiation automatique de pytorch – par exemple, aucun appels à torch.nn, torch.autograd ou à la méthode .backward()**. L'objectif est d'implémenter un modèle de classification logistique ainsi que son entainement en utilisant uniquement des opérations matricielles de base fournies par PyTorch e.g. torch.sum(), torch.matmul(), etc.

2.2.2 Rapport à rédiger

Présentez vos résultats dans un rapport. Ce rapport devrait inclure:

- **Recherche d'hyperparamètres:** Faites une recherche d'hyperparamètres pour différents taux d'apprentissage, e.g. 0.1, 0.01, 0.001, et différentes tailles de mini-batch, e.g. 1, 20, 200, 1000 pour des modèles entrainés avec SGD. Présentez dans un tableau la précision finale du modèle, sur l'*ensemble de validation*, pour ces différentes combinaisons d'hyperparamètres.
- **Analyse du meilleur modèle:** Pour votre meilleur modèle, présentez deux figures montrant la progression de son apprentissage sur l'*ensembe d'entrainement et l'ensemble de validation*. La première figure montrant les courbes de log-vraisemblance négative moyenne après chaque epoch, la deuxième montrant la précision du modèle après chaque epoch. Finalement donnez la précision finale sur l'ensemble de test.
- Lire l'article de recherche - Adam: a method for stochastic optimization. Kingma, D., & Ba, J. (2015). International Conference on Learning Representation (ICLR). https://arxiv.org/pdf/1412.6980.pdf. Implémentez Adam, répétez les deux étapes précédentes (recherche d'hyperparamètres et analyse du meilleur modèle) cette fois en utilisat Adam, et comparez les performances finales avec votre meilleur modèle SGD.

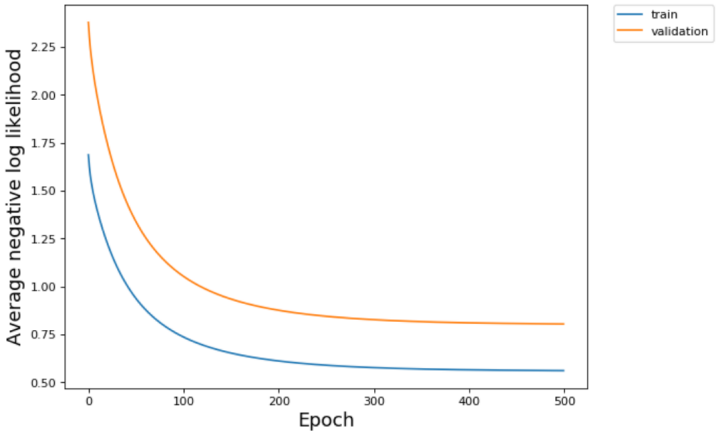


Figure 2: Exemple de courbes d'apprentissage.

3 Partie III (20 points)

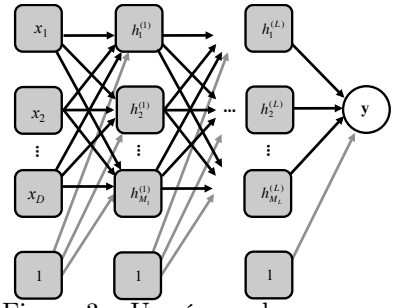


Figure 3: Un réseau de neurones.

Pour cette partie, vous pouvez travailler en groupes de 2, mais il faut écrire sa propre dérivation et soumettre son propre rapport. Si vous travaillez avec un partenaire, il faut indiquer leur nom dans votre rapport.

Considérons maintenant un réseau de neurones avec une couche d'entrée avec $D = 784$ unités, L couches cachées, chacune avec 300 unités et un vecteur de sortie $\mathbf{y}$ de dimension K. Vous avez $i = 1, .., N$ exemples dans un ensemble d'apprentissage, où chaque $\mathbf{x}_i \in \mathbb{R}^{784}$ est un vecteur de caractéristiques (features). $\mathbf{y}$ est un vecteur du type *one-hot* – un vecteur de zéros avec un seul 1 pour indiquer que la classe $C = k$ dans la dimension k. Par exemple, le vecteur $\mathbf{y} = [0, 1, 0, \cdots, 0]^T$ représente la deuxième classe. La fonction de perte est donnée par

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{k,i} \log(f_k(\mathbf{x}_i)) \tag{6}$$

La fonction d'activation de la couche finale a la forme $\mathbf{f} = [f_1, ..., f_K]$ donné par la fonction d'activation softmax:

$$f_k(\mathbf{a}^{(L+1)}(\mathbf{x}_i)) = \frac{\exp(a_k^{(L+1)})}{\sum_{c=1}^{K} \exp(a_c^{(L+1)})},$$

et les couches cachées utilisent une fonction d'activation de type ReLU:

$$\mathbf{h}^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \text{ReLU}(\mathbf{a}^{(l)}(\mathbf{x}_i) = \max\left(0,\ \mathbf{a}^{(l)}(\mathbf{x}_i)\right) \tag{7}$$

où $\mathbf{a}^{(l)}$ est le vecteur résultant du calcul de la préactivation habituelle $\mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$, qui pourrait être simplifiée à $\boldsymbol{\theta}^{(l)}\tilde{\mathbf{h}}^{(l-1)}$ en utilisant l'astuce de définir $\tilde{\mathbf{h}}$ comme $\mathbf{h}$ avec un 1 concaténé à la fin du vecteur.

a) (5 points) Donnez le pseudocode incluant des *calculs matriciels—vectoriels*[1] détaillés pour l'algorithme de rétropropagation pour calculer le gradient pour les paramètres de chaque couche **étant donné un exemple d'entraînement**.

b) (15 points) Implémentez l'optimisation basée sur le gradient de ce réseau en Pytorch. Utilisez le code squelette dans le fichier Colab comme point de départ et implémentez les mathématiques de l'algorithme de rétropropagation que vous avez décrit à la question précédente. Utilisez encore l'ensemble de données de Fashion MNIST (voir Partie 2). Comparez vos gradients et votre optimisation avec le même modèle optimisé avec Autograd. Lequel est le plus rapide ? Si vous rencontrez des difficultés pour implémenter votre modèle qui n'utilise pas la différenciation automatique avant la date limite, vous pouvez uniquement remettre la version autodiff et une version partiellement implémentée avec des gradients calculés manuellement pour les notes partielles. Utilisez le modèle le plus rapide pour les expériences suivantes. **Comparez différents modèles ayant différentes largeurs (nombre d'unités) et profondeurs (nombre de couches)**. Proposez et réalisez vos propres expériences. Ici encore, n'utilisez l'ensemble de test que pour votre expérience finale lorsque vous pensez avoir obtenu votre meilleur modèle.

[1] Voir The Matrix Cookbook: `https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`