Date de remise: 17 Octobre (23h00), 2022

Instructions

- *Montrez vos traces pour toutes les questions !*
- *Utilisez LaTeX et le modèle que nous vous fournissons pour rédiger vos réponses. Vous pouvez réutiliser la plupart des raccourcis de notation, des équations et/ou des tableaux. SVP voir la politique des devoirs sur le site web du cours pour plus de détails.*
- *Soumettez ce devoir sur Gradescope. Assurez-vous que vous sélectionnez les pages contenant la réponse pour chaque question sur Gradescope. Nous ne noterons pas les questions sans les pages sélectionnées.*
- *Les auxiliaires d'enseignement pour ce devoir sont* ***Nanda Harishankar Krishna et Sarthak Mittal****.*

Question 1 (5-2-3-5)**.** (**La fonction d'activation**)

1. Considérez la fonction d'activation $h(x) = x\sigma(\beta x)$ où $\sigma(\cdot)$ définit la fonction d'activation sigmoïde $\sigma(z) = \frac{1}{1+\exp(-z)}$. Montrez que, pour un choix approprié de β, cette fonction d'activation peut rapprocher (a) la fonction d'activation linéaire, et (b) la fonction d'activation du ReLU.
2. Considérer la fonction d'activation linéaire continue $h(x)$ qui présente les pentes suivantes à différentes parties de l'entrée, en particulier, $h'(x) = \begin{cases} a & x >= \alpha \\ 0 & \beta < x < \alpha \\ -b & x <= \beta \end{cases}$ et $h(x) = 0$ pour $\beta < x < \alpha$. Supposons que vous n'ayez accès qu'aux opérations arithmétiques (l'addition, la soustraction, la multiplication et la division) ainsi que la fonction d'activation ReLU $ReLU(x) = max(0, x)$. Pouvez-vous construire $h(x)$ en utilisant ces opérations élémentaires et l'activation ReLU ? Pouvez-vous penser à une fonction d'activation populaire similaire à $h(\cdot)$?
   Conseil : Essayez de visualiser la fonction d'activation
3. Rappeler la définition de la fonction softmax $\mathcal{S}(\boldsymbol{x})_i = e^{x_i} / \sum_j e^{x_j}$, où $\boldsymbol{x} \in \mathbb{R}^d$. Montrez que c'est traduction invariante, c'est-à-dire $\mathcal{S}(\boldsymbol{x} + c) = \mathcal{S}(\boldsymbol{x})$ pour toute $c \in \mathbb{R}$.
4. Donné un i il est souvent nécessaire pour calculer $\log \mathcal{S}(\boldsymbol{x})_i = x_i - \log \sum_j e^{x_j}$. Cependant, cela peut souvent conduire à des incertitudes numériques car (a) si tous les x_j sont petits, alors il devient proche de $\log 0$ (soupassement), et b) si tous les x_j sont grands, leurs exposants peuvent entraîner un débordement. Pouvez-vous utiliser la propriété d'invariance de traduction de softmax pour le rendre numériquement stable ?
   Conseil : Pensez à $\max_j e^{x_j}$.

Answer 1. 1. On considère la fonction d'activation $h(x) = x\sigma(\beta x)$ où $\sigma(\cdot)$ définit la fonction d'activation sigmoïde $\sigma(z) = \frac{1}{1+\exp(-z)}$.
Montrons que cette fonction d'activation peut rapprocher la fonction d'activation linéaire, puis la fonction d'activation ReLU:

(a) Pour la fonction d'activation linéaire :
On pose $f(x) = x$ la fonction d'activation linéaire.
Résolvons l'équation $\frac{x}{1+\exp(-\beta x)} = x$.
Cas 1: si $x > 0$

$$\begin{aligned}\frac{x}{1+\exp(-\beta x)} &= x \\ \frac{1}{1+\exp(-\beta x)} &= 1 \\ 1+\exp(-\beta x) &= 1 \\ \exp(-\beta x) &= 0\end{aligned}$$

Donc, si $x > 0$:

$$\beta \to +\infty \iff h(x) \approx f(x)$$

Cas 2: si $x < 0$

$$\begin{aligned}\frac{x}{1+\exp(-\beta x)} &= x \\ \frac{1}{1+\exp(-\beta x)} &= 1 \\ 1+\exp(-\beta x) &= 1 \\ \exp(-\beta x) &= 0\end{aligned}$$

Donc, si $x < 0 \iff -x > 0$:

$$\beta \to -\infty \iff h(x) \approx f(x)$$

Cas 3: si $x = 0$
On a bien

$$\begin{aligned}\frac{x}{1+\exp(-\beta x)} &= x \\ \frac{0}{1+\exp(-\beta x)} &= 0\end{aligned}$$

Donc, si $x = 0$:

$$h(x) \approx f(x), \forall \beta \in \mathbb{R}$$

(b) Pour la fonction d'activation ReLU :
On pose $g(x)$ la fonction d'activation ReLU.
Résolvons l'équation $\frac{x}{1+\exp(-\beta x)} = 0$, si x < 0 et $\frac{x}{1+\exp(-\beta x)} = x$ si $x \geq 0$.
Cas 1: si $x < 0$

$$\begin{aligned}&\frac{x}{1+\exp(-\beta x)} = 0 \\ \iff &1+\exp(-\beta x) \to +\infty \\ \iff &\exp(-\beta x) \to +\infty\end{aligned}$$

Puisque $x < 0 \iff -x > 0$, cela revient à avoir :

$$\iff \beta \to +\infty$$

Donc, si $x < 0$:

$$\beta \to +\infty \iff h(x) \approx g(x)$$

Cas 2: si $x > 0$

$$\frac{x}{1 + \exp(-\beta x)} = x$$
$$\frac{1}{1 + \exp(-\beta x)} = 1$$
$$1 + \exp(-\beta x) = 1$$
$$\exp(-\beta x) = 0$$

Donc, si $x > 0$:

$$\beta \to +\infty \iff h(x) \approx g(x)$$

Cas 3: si $x = 0$
On a bien

$$\frac{x}{1 + \exp(-\beta x)} = x$$
$$\frac{0}{1 + \exp(-\beta x)} = 0$$

Donc, si $x = 0$:

$$h(x) \approx f(x), \forall \beta \in \mathbb{R}$$

Ainsi, pour un choix approprié de β, on peut rapprocher $h(x)$ à la fonction d'activation linéaire, ou à la fonction d'activation ReLU.

2. On a $h(x)$ une fonction d'activation linéaire continue tel que :

$$h'(x) = \begin{cases} a & x >= \alpha \\ 0 & \beta < x < \alpha \\ -b & x <= \beta \end{cases}$$

et $h(x) = 0$ pour $\beta < x < \alpha$. Donc, on a :

$$h(x) = \begin{cases} ax - \alpha a & x >= \alpha \\ 0 & \beta < x < \alpha \\ -bx + \beta b & x <= \beta \end{cases}$$

En effet, lorsque $x >= \alpha$, $h(x)$ est une fonction linéaire de pente a et coupe l'axe des abscisses au point $(\alpha, 0)$ et l'axe ordonnées au point $(0, -a\alpha)$.
De même, lorsque $x <= \beta$, $h(x)$ est une fonction linéaire de pente $-b$ et coupe l'axe des abscisses au point $(\beta, 0)$ et l'axe ordonnées au point $(0, b\beta)$.

On peut reconstruire $h(x)$ avec certaines opérations arithmétiques et la fonction d'activation ReLU. En effet, on a :

$$\begin{aligned} h(x) &= a \times ReLU(x-\alpha) + b \times ReLU(\beta - x) \\ &= a \times max(0, x-\alpha) + b \times max(0, \beta - x) \end{aligned}$$

Si $x <= \beta$, on a forcément $x - \alpha < 0$ et $\beta - x >= 0$, d'où :

$$\begin{aligned} h(x) &= a \times max(0, x-\alpha) + b \times max(0, \beta - x) \\ &= a \times 0 + b\beta - bx \\ &= b\beta - bx \end{aligned}$$

Si $\beta < x < \alpha$, on a forcément $x - \alpha < 0$ et $\beta - x < 0$, d'où :

$$\begin{aligned} h(x) &= a \times max(0, x-\alpha) + b \times max(0, \beta - x) \\ &= a \times 0 + b \times 0 \\ &= 0 \end{aligned}$$

Si $x >= \alpha$, on a forcément $x - \alpha >= 0$ et $\beta - x < 0$, d'où :

$$\begin{aligned} h(x) &= a \times max(0, x-\alpha) + b \times max(0, \beta - x) \\ &= ax - a\alpha + b \times 0 \\ &= ax - a\alpha \end{aligned}$$

D'où le résultat.
Une fonction d'activation populaire similaire à $h(x)$ serait la fonction marche/heaviside.

3. La fonction softmax $\mathcal{S}(x)$est une fonction d'activation communément utilisée pour la classification multiclasse. En effet, elle permet une estimation de probabilité pour chacune des classes du modèle.
La fonction s'écrit :

$$\mathcal{S}(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

. Montrons que $\mathcal{S}(\boldsymbol{x} + c) = \mathcal{S}(\boldsymbol{x})$ pour tout $c \in \mathbb{R}$:

$$\begin{aligned} & \mathcal{S}(\boldsymbol{x} + c) = \frac{e^{x_i + c}}{\sum_j e^{x_j + c}} \\ \iff & \mathcal{S}(\boldsymbol{x} + c) = \frac{e^{x_i} e^{c}}{\sum_j e^{x_j} e^{c}} \\ \iff & \mathcal{S}(\boldsymbol{x} + c) = \frac{e^{x_i}}{\sum_j e^{x_j}} \\ \iff & \mathcal{S}(\boldsymbol{x} + c) = \mathcal{S}(\boldsymbol{x}) \end{aligned}$$

D'où le résultat.

4. En utilisant la propriété d'invariance de traduction de softmax, pour le rendre numériquement stable, il suffit de considérer $c = -max_j(x_j)$.
   On a alors, avec la fonction exponentielle fonction strictement croissante positive sur $\mathbb{R}$:

$$e^{x_j - max_j(x_j)} = e^{x_j} e^{-max_j(x_j)} = \frac{e^{x_j}}{e^{max_j(x_j)}} = \frac{e^{x_j}}{max_j e^{x_j}}$$

   En effet, un tel choix permet de stabiliser numériquement, lorsqu'on considère des x_j très petits, et permet de décaler les valeurs les plus petites "vers la droite", de sorte à ce qu'elles soient plus grandes, en les divisant par $max_j e^{x_j}$.
   Lorsqu'on considère des x_j très grands, un tel choix permet de décaler les valeurs les plus grandes "vers la gauche", de sorte à ce qu'elles soient plus petites, en les divisant par $max_j e^{x_j}$.

Question 2 (10). (**Les modèles de mélange rencontrent les réseaux neuronaux**)

Considérez de modéliser certaines données $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, $\boldsymbol{x}_n \in \mathbb{R}^d$, $y_n \in \{0, 1\}$, en utilisant un mélange de modèles de régression logistique, où nous modélisons chaque étiquette binaire y_n en sélectionnant d'abord l'un des modèles de régression logistique K, basée sur la valeur d'une variable latente $z_n \sim$ Categorical$(\pi_1, ..., \pi_K)$ puis générant y_n *conditionné* sur z_n en tant que $y_n \sim$ Bernoulli$[\sigma(\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n)]$, où $\sigma(\cdot)$ est la fonction d'activation sigmoïde.

Maintenant, veuillez considérer la probabilité *marginale* de l'étiquette $y_n = 1$, étant donné $\boldsymbol{x}_n$, c.-à-d., $p(y_n = 1|\boldsymbol{x}_n)$, et montrez que cette quantité peut aussi être considérée comme la sortie d'un réseau neuronal. Spécifiez clairement ce qu'est la couche d'entrée, le(s) calque(s) caché(s), les activations, la couche de sortie.

Answer 2. La couche d'entrée est le vecteur des $\boldsymbol{x}_n$, la couche cachée est le vecteur des π_K, les activations sont la régression logistique et la fonction sigmoïde. Enfin la couche de sortie est seulement la probabilité marginale de l'étiquette $y_n = 1$ étant donné $\boldsymbol{x}_n$, $p(y_n = 1|\boldsymbol{x}_n)$.
Voici un schéma de l'architecture du modèle :

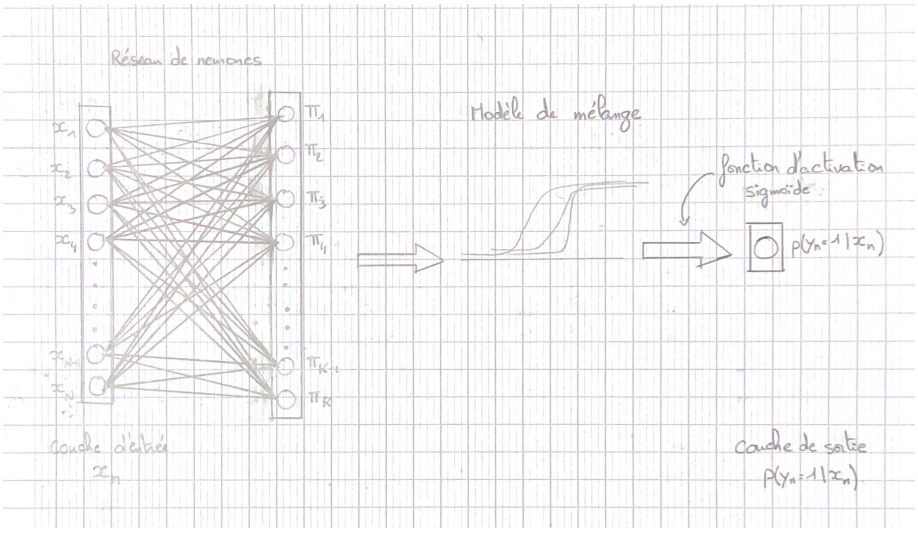

FIGURE 1 – Architecture du modèle

Question 3 (5-2-5-2-2-4)**.** (**Schémas d'optimisation**)

L'optimisation est l'un des piliers les plus importants d'apprentissage profond. Avec l'augmentation de la puissance de calcul ainsi que des schémas d'optimisation plus complexes comme Momentum et Adam, nous sommes en mesure d'optimiser des modèles d'apprentissage profond complexes à grande échelle. Cependant, tous les algorithmes que nous avons vus utilisent seulement des informations dérivées de premier ordre. Il y a mieux ?

Laissez $f(\boldsymbol{w})$ être la fonction que vous voulez minimiser. Une façon simple est de faire une descente de gradient, c'est-à-dire de mettre à jour itérativement $\boldsymbol{w}$ en $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \nabla f(\boldsymbol{w}^{(t)})$. Essayons de faire mieux maintenant.

1. Considérez l'approximation Taylor de deuxième ordre de la fonction f à $\boldsymbol{w}^{(t)}$.

$$f(\boldsymbol{w}) \approx f(\boldsymbol{w}^{(t)}) + \left\langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \right\rangle + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H} (\boldsymbol{w} - \boldsymbol{w}^{(t)})$$

   Quelle est la solution du problème d'optimisation suivant ?

$$\arg\min_{\boldsymbol{w}} \left[f(\boldsymbol{w}^{(t)}) + \left\langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \right\rangle + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H} (\boldsymbol{w} - \boldsymbol{w}^{(t)}) \right] \tag{1}$$

   Où $\boldsymbol{H}$ est la matrice hessienne (considérer défini positif), c'est-à-dire la matrice dont les éléments sont $\frac{\partial^2}{\partial w_i w_j} f(\boldsymbol{w}^{(t)})$. Ça vous dit quelque chose sur le "taux d'apprentissage" ?
2. Considérez un model perceptron multicouche composé de deux couches cachées de 512 neurones chacune. L'entrée se compose de 32 fonctionnalités, et la sortie est 10 dimensions pour représenter la probabilité de chaque classe dans un problème de classification de 10 classes. Combien de paramètres le modèle a-t-il (veuillez également inclure les termes de biais) ? Quelle est la taille de la matrice $\boldsymbol{H}$?
3. Compte tenu de l'échelle actuelle des architectures (p. ex., GPT, Dall-E, etc.), C'est facile de voir que le calcul de $\boldsymbol{H}$ est souvent intraitable. Cependant, souvent nous n'avons pas besoin de la matrice complète de $\boldsymbol{H}$, et au lieu de ça, on a juste besoin de produits Hessian-vector, c'est-à-dire $\boldsymbol{H}\boldsymbol{v}$ pour certains vecteurs $\boldsymbol{v}$. Laissez $g(\cdot)$ être la fonction de gradient de f, et considérez l'expansion de la série Taylor.

$$g(\boldsymbol{w} + \Delta\boldsymbol{w}) \approx g(\boldsymbol{w}) + \boldsymbol{H}\Delta\boldsymbol{w}$$

   Pouvons-nous utiliser cette approximation pour estimer le produit $\boldsymbol{H}\boldsymbol{v}$, donné vecteurs $\boldsymbol{v}$ quelconques. (*Conseil : Considérez* $\Delta\boldsymbol{w} = r\boldsymbol{v}$ *avec petit* r)
4. L'expansion réelle de la série Taylor de la fonction de gradient est plutôt $g(\boldsymbol{w} + \Delta\boldsymbol{w}) = g(\boldsymbol{w}) + \boldsymbol{H}\Delta\boldsymbol{w} + O(||\Delta\boldsymbol{w}||^2)$. Dans ce cas, quelle est l'estimation de l'erreur lors du calcul de $\boldsymbol{H}\boldsymbol{v}$ comme dérivé ci-dessus, où $\Delta\boldsymbol{w} = r\boldsymbol{v}$. Vous pouvez utiliser la notation $O(\cdot)$ pour décrire ce que serait l'erreur.
5. Le problème avec cette solution approximative est qu'elle est souvent sujette à des erreurs numériques. Cela est dû au fait que nous avons besoin de r pour réduire l'erreur, mais quand r est très petit alors $r\boldsymbol{v}$ perd de la précision lorsqu'il est ajouté à vw. Cependant, peut-être pouvons-nous faire mieux et obtenir une solution plus exacte plutôt qu'une solution approximative ?

Considérez l'opérateur $\mathcal{R}-$, qui est défini comme

$$\mathcal{R}_v\{f(\boldsymbol{w})\} = \frac{\partial}{\partial r} f(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0}$$

Montrez que

$$\mathcal{R}_v\{cf(\boldsymbol{w})\} = c\mathcal{R}_v\{f(\boldsymbol{w})\} \qquad \text{(Linearity under Scalar Multiplication)}$$
$$\mathcal{R}_v\{f(\boldsymbol{w})g(\boldsymbol{w})\} = \mathcal{R}_v\{f(\boldsymbol{w})\}g(\boldsymbol{w}) + f(\boldsymbol{w})\mathcal{R}_v\{g(\boldsymbol{w})\} \qquad \text{(Product Rule)}$$

6. Enfin, Utilisez l'opérateur $\mathcal{R}-$ pour dériver une équation pour calculer exactement le produit $\boldsymbol{Hv}$ à vecteur hessien.

L'algorithme que nous avons dérivé est le populaire *Newton Method*, il a été prouvé que la convergence est plus rapide que les méthodes de premier ordre sous certaines hypothèses. Non seulement avons-nous dérivé cette procédure d'optimisation, mais nous avons également découvert des moyens d'estimer de façon réaliste les produits vectoriels matriciels nécessaires.

Answer 3. 1. D'après l'approximation de Taylor de deuxième ordre, on a :

$$\nabla f(\boldsymbol{w}) \approx \langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \rangle + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)})$$

On cherche à résoudre :

$$\arg\min_{\boldsymbol{w}} \left[f(\boldsymbol{w}^{(t)}) + \langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \rangle + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)}) \right]$$
$$\iff \quad \frac{\partial}{\partial w}\left(f(\boldsymbol{w}^{(t)}) + \langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \rangle + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)}) \right) = 0$$

Or, puisque :

$$\frac{\partial}{\partial w} f(\boldsymbol{w}^{(t)} = 0$$
$$\frac{\partial}{\partial w}(\langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \rangle) = \nabla f(\boldsymbol{w}^{(t)})$$
$$\frac{\partial}{\partial w}((\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)})) = 2\boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)}) \qquad (\text{car } \nabla_x\ (\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x})$$

On a donc :

$$\frac{\partial}{\partial w}\left(f(\boldsymbol{w}^{(t)}) + \langle \boldsymbol{w} - \boldsymbol{w}^{(t)}, \nabla f(\boldsymbol{w}^{(t)}) \rangle + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{(t)})^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)}) \right) = 0$$
$$\iff \quad \frac{\partial}{\partial w} \nabla f(\boldsymbol{w}^{(t)}) + \frac{2}{2}\boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)}) = 0$$
$$\iff \quad \frac{\partial}{\partial w} \nabla f(\boldsymbol{w}^{(t)}) + \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^{(t)}) = 0$$

D'où le résultat :

$$\boldsymbol{w} = \boldsymbol{w}^{(t)} - \boldsymbol{H}^{-1}\nabla f(\boldsymbol{w}^{(t)})$$

On retrouve ainsi la formule générale de la descente du gradient. On peut alors dire que le taux d'apprentissage est égale à $\boldsymbol{H}^{-1}$.

2. On a à la première couche de pré-activation :

$$a^{(1)}(x) = W^{(1)}h^{(0)}(x) + b^{(1)}$$

avec $W^{(1)}$ de dimension 512 x 32, $h^{(0)} = x$ l'entrée de taille 32 x 1 et $b^{(1)}$ de dimension 512 x 1. $a^{(1)}$ est donc de dimension 512 x 1. L'activation est alors :

$$h^{(1)}(x) = g(a^{(1)}(x))$$

. Elle est de dimension 512 x 1 également.
On a à la deuxième couche de pré-activation :

$$a^{(2)}(x) = W^{(2)}h^{(1)}(x) + b^{(2)}$$

avec $W^{(2)}$ de dimension 512 x 512, $h^{(1)}$ de dimension 512 x 1 et $b^{(1)}$ de dimension 512 x 1. $a^{(2)}$ est donc de dimension 512 x 1. L'activation est alors :

$$h^{(2)}(x) = g(a^{(2)}(x))$$

. Elle est de dimension 512 x 1 également.
On a enfin en sortie :

$$a^{(3)}(x) = W^{(3)}h^{(2)}(x) + b^{(3)}$$

avec $W^{(3)}$ de dimension 10 x 512, $h^{(2)}$ de dimension 512 x 1 et $b^{(3)}$ de dimension 10 x 1. $a^{(3)}$ est donc de dimension 10 x 1. Une fonction softmax peut-être alors appliquée et on aurait :

$$h^{(3)}(x) = o(a^{(3)}(x)) = f(x)$$

. La sortie est ainsi de dimension 10 x 1.

Le nombre total de paramètre serait alors :

$$512 \times 32 + 512 \times 1 + 512 \times 512 + 512 \times 1 + 10 \times 512 + 10 \times 1 = 284682$$

Au final, le modèle a 284682 paramètres.
La taille de la matrice Hessienne est 32 x 32.

3. On a :

$$g(\boldsymbol{w} + \Delta\boldsymbol{w}) \approx g(\boldsymbol{w}) + \boldsymbol{H}\Delta\boldsymbol{w}$$

On pose $\Delta \boldsymbol{w} = r\boldsymbol{v}$.
On a alors :

$$g(\boldsymbol{w} + r\boldsymbol{v}) \approx g(\boldsymbol{w}) + \boldsymbol{H}r\boldsymbol{v}$$
$$g(\boldsymbol{w} + r\boldsymbol{v}) \approx g(\boldsymbol{w}) + r\boldsymbol{H}\boldsymbol{v}$$
$$\boldsymbol{H}\boldsymbol{v} \approx \frac{g(\boldsymbol{w} + r\boldsymbol{v}) - g(\boldsymbol{w})}{r}$$

On peut ainsi estimer $H\boldsymbol{v} \approx \frac{g(\boldsymbol{w}+r\boldsymbol{v})-g(\boldsymbol{w})}{r}$.

4. Sachant que l'expansion réelle de la série Taylor de la fonction de gradient est plutôt $g(\boldsymbol{w}+\Delta\boldsymbol{w}) = g(\boldsymbol{w}) + \boldsymbol{H}\Delta\boldsymbol{w} + O(||\Delta\boldsymbol{w}||^2)$, on a alors :

$$\boldsymbol{H}\boldsymbol{v} \approx \frac{g(\boldsymbol{w} + r\boldsymbol{v}) - g(\boldsymbol{w})}{r} + O(r)$$

Donc l'estimation de l'erreur lors du calcul de $H\boldsymbol{v}$ serait de $O(r)$.

5. On considère l'opérateur $\mathcal{R}-$ tel que :

$$\mathcal{R}_v\{f(\boldsymbol{w})\} = \frac{\partial}{\partial r} f(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0}$$

Montrons que :

$\mathcal{R}_v\{cf(\boldsymbol{w})\} = c\mathcal{R}_v\{f(\boldsymbol{w})\}$ (Linearity under Scalar Multiplication)
$\mathcal{R}_v\{f(\boldsymbol{w})g(\boldsymbol{w})\} = \mathcal{R}_v\{f(\boldsymbol{w})\}g(\boldsymbol{w}) + f(\boldsymbol{w})\mathcal{R}_v\{g(\boldsymbol{w})\}$ (Product Rule)

(a) Linearity under Scalar Multiplication:
Avec c une constance, on a :

$$\mathcal{R}_v\{cf(\boldsymbol{w})\} = \frac{\partial}{\partial r} cf(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0}$$
$$\mathcal{R}_v\{cf(\boldsymbol{w})\} = c\frac{\partial}{\partial r} f(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0}$$
$$\mathcal{R}_v\{cf(\boldsymbol{w})\} = c\mathcal{R}_v\{f(\boldsymbol{w})\}$$

(b) Product Rule:

$$\mathcal{R}_v\{f(\boldsymbol{w})g(\boldsymbol{w})\} = \frac{\partial}{\partial r} f(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0} g(\boldsymbol{w}) + f(\boldsymbol{w})\frac{\partial}{\partial r} g(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0}$$
$$\mathcal{R}_v\{f(\boldsymbol{w})g(\boldsymbol{w})\} = \mathcal{R}_v\{f(\boldsymbol{w})\}g(\boldsymbol{w}) + f(\boldsymbol{w})\mathcal{R}_v\{g(\boldsymbol{w})\}$$

6. On a :

$$\mathcal{R}_v\{g(\boldsymbol{w})\} = \frac{\partial}{\partial r} g(\boldsymbol{w} + r\boldsymbol{v})\Big|_{r=0}$$
$$\mathcal{R}_v\{g(\boldsymbol{w})\} = \frac{\partial}{\partial r} (g(\boldsymbol{w}) + r\boldsymbol{H}\boldsymbol{v})\Big|_{r=0}$$
$$\mathcal{R}_v\{g(\boldsymbol{w})\} = \frac{\partial}{\partial r} (r\boldsymbol{H}\boldsymbol{v})\Big|_{r=0}$$
$$\mathcal{R}_v\{g(\boldsymbol{w})\} = \boldsymbol{H}\boldsymbol{v}$$

Donc :

$$\boldsymbol{Hv} = \mathcal{R}_v\{g(\boldsymbol{w})\}$$
$$\boldsymbol{Hv} = \mathcal{R}_v\{\nabla f(\boldsymbol{w})\}$$

Question 4 (3-5-5-7)**.** (**Base de convolution**)

1. Considérez un Réseau neuronal convolutif avec l'architecture suivante

   Image → Conv-3x3 → ReLU → Conv-3x3 → MaxPool-2x2 → Conv-3x3 → ReLU → Output

   où Conv-3x3 fait référence à une couche convolutionnelle avec un noyau 3x3 taille, et MaxPool-2x2 fait référence à l'opération max-pooling sur une fenêtre noyau 2x2. Les convolutions utiliser la foulée 1 et les opérations de mutualisation utiliser la foulée 2 et n'utilisent aucun remplissage. Image de résolution en entrée (32×32), trouvez la résolution de la sortie lors du passage de cette image à travers le réseau défini ci-dessus.
2. Considérez un autre Réseau neuronal convolutif avec l'architecture.:

   Image → Conv-5x5 → ReLU → Conv-5x5 → ReLU → Global Avg. Pool → FC-128 → FC-1 (Output)

   où nous réutilisons la notation Conv-axa pour représenter les couches convolutionnelles avec la taille du noyau axa et foulée 1, et FC-b se réfère à un réseau neuronal Feedforward avec des neurones b. Cette architecture peut-elle gérer des images de taille variable ? Les Perceptron multicouches (sans couches de regroupement convolutionnelles ou globales) sont-ils capables de gérer des entrées de taille variable ? Si oui, pourquoi ; sinon, pourquoi pas ?
3. Couches de convolution *intrinsèquement* calculer une fonction linéaire (Similaire aux réseaux de transmission sans couches d'activation non linéaires !). Rappelez-vous que les fonctions d'entrée X, une fonction linéaire est n'importe quelle fonction qui peut être calculée comme, pour une matrice A.
   Considérez une couche de convolution avec le noyau W et entrez X donné par

   $$W = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \quad X = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{bmatrix}$$

   Trouvez la carte de sortie en convoquant l'entrée X avec le noyau W, **Utilisation de la foulée 2 et zéro rembourrage de taille 1**. Pouvez-vous réécrire cette fonction linéaire (c'est-à-dire, quelle est la matrice A pour cette transformation ?). Vous pouvez envisager une extension des fonctionnalités en ligne, qui représente tous les éléments de la première rangée, puis tous les éléments de la deuxième rangée, et ainsi de suite comme un vecteur aplati, par exemple $X = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{1,0} & \dots & x_{2,1} & x_{2,2} \end{bmatrix}$
4. Les convolutions transposées peuvent aider à améliorer les fonctionnalités spatialement. Considérez une couche de convolution transposée avec noyau $W = \begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \end{bmatrix}$, utilisé sur l'entrée $X = \begin{bmatrix} x_{0,0} & x_{0,1} \\ x_{1,0} & x_{1,1} \end{bmatrix}$, avec **stride = 2 et padding = 0**. La sortie est-elle toujours une transformation linéaire des caractéristiques d'entrée ?Pouvez-vous montrer ce que devrait être la matrice linéaire, compte tenu de l'expansion principale ligne pour les caractéristiques comme avant ?

Answer 4. 1. On applique la formule vue en cours :

$$o = \lfloor \frac{i + 2p - k}{s} \rfloor + 1$$

On a alors, en sortie de la première couche de convolution : $o = \lfloor \frac{32+2\times0-3"}{1} \rfloor + 1 = 30$. L'image est donc de résolution 30 x 30.
On a ensuite, en sortie de la deuxième couche de convolution (l'opération ReLU n'affectant pas la résolution), une image de résolution 28 x 28.
L'image en sortie de l'opération de MaxPooling est de résolution 14 x 14.
Enfin, l'image en sortie de la troisième couche de convolution est de taille 12 x 12. L'opération ReLU n'affectant pas les dimensions de cette dernière, la résolution de la sortie, lors du passage d'une image 32 x 32 à travers le réseau, est de 12 x 12.

2. Cette architecture permet de gérer des images de taille variable, à condition que le kernel en sortie de la deuxième couche de convolution soit de dimension 128.
Les perceptrons multicouche ne sont pas capables de gérer des entrées de taille variable. En effet, ils ne sont capables que de gérer des entrées de taille $b = 128$ ici.

3. On a :

$$W = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \quad X = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{bmatrix}$$

La dimension de la sortie est : $\lfloor \frac{3+2-3}{2} \rfloor + 1 = 2$. La résolution de l'output Y est donc 2 x 2.

$$Y = \begin{bmatrix} y_{0,0} & y_{0,1} \\ y_{1,0} & y_{1,1} \end{bmatrix}$$

En appliquant l'opération de convolution (cross-correlation), on a :

$$\begin{aligned} y_{0,0} &= w_{1,1}x_{0,0} + w_{1,2}x_{0,1} + w_{2,1}x_{1,0} + w_{2,2}x_{1,1} \\ y_{0,1} &= w_{1,1}x_{0,1} + w_{1,1}x_{0,2} + w_{2,0}x_{1,1} + w_{2,1}x_{1,2} \\ y_{1,0} &= w_{0,1}x_{1,0} + w_{0,2}x_{1,1} + w_{1,1}x_{2,0} + w_{1,2}x_{2,1} \\ y_{1,1} &= w_{0,0}x_{1,1} + w_{0,1}x_{1,2} + w_{1,0}x_{2,1} + w_{1,1}x_{2,2} \end{aligned}$$

D'où :

$$Y = \begin{bmatrix} w_{1,1}x_{0,0} + w_{1,2}x_{0,1} + w_{2,1}x_{1,0} + w_{2,2}x_{1,1} & w_{1,1}x_{0,1} + w_{1,1}x_{0,2} + w_{2,0}x_{1,1} + w_{2,1}x_{1,2} \\ w_{0,1}x_{1,0} + w_{0,2}x_{1,1} + w_{1,1}x_{2,0} + w_{1,2}x_{2,1} & w_{0,0}x_{1,1} + w_{0,1}x_{1,2} + w_{1,0}x_{2,1} + w_{1,1}x_{2,2} \end{bmatrix}$$

En posant $\boldsymbol{AX} = \boldsymbol{Y}$, avec $\boldsymbol{X}$ de dimension 9 x 1, $\boldsymbol{Y}$ de dimension 4 x 1 et $\boldsymbol{A}$ de dimension 4 x 9, on a alors :

$$A = \begin{bmatrix} w_{1,1} & w_{1,2} & 0 & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & w_{1,0} & w_{1,1} & 0 & w_{2,0} & w_{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{0,1} & w_{0,2} & 0 & w_{1,1} & w_{1,2} & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & 0 & w_{1,0} & w_{1,1} \end{bmatrix}$$

On a bien :

$$\begin{bmatrix} w_{1,1} & w_{1,2} & 0 & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & w_{1,0} & w_{1,1} & 0 & w_{2,0} & w_{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{0,1} & w_{0,2} & 0 & w_{1,1} & w_{1,2} & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & 0 & w_{1,0} & w_{1,1} \end{bmatrix} \times \begin{bmatrix} x_{0,0} \\ x_{0,1} \\ x_{0,2} \\ x_{1,0} \\ x_{1,1} \\ x_{1,2} \\ x_{2,0} \\ x_{2,1} \\ x_{2,2} \end{bmatrix} = \begin{bmatrix} w_{1,1}x_{0,0} + w_{1,2}x_{0,1} + w_{2,1}x_{1,0} + w_{2,2}x_{1,1} \\ w_{1,1}x_{0,1} + w_{1,1}x_{0,2} + w_{2,0}x_{1,1} + w_{2,1}x_{1,2} \\ w_{0,1}x_{1,0} + w_{0,2}x_{1,1} + w_{1,1}x_{2,0} + w_{1,2}x_{2,1} \\ w_{0,0}x_{1,1} + w_{0,1}x_{1,2} + w_{1,0}x_{2,1} + w_{1,1}x_{2,2} \end{bmatrix}$$

4. La sortie est toujours une transformation linéaire des caractéristiques d'entrée.
   On a bien avec $\boldsymbol{AX} = \boldsymbol{Y}$, ($\boldsymbol{X}$ est de dimension 4 x 1, $\boldsymbol{Y}$ de dimension 16 x 1 et $\boldsymbol{A}$ de dimension 16 x 4).
   La matrice linéaire $\boldsymbol{A}$ serait la suivante :

$$A = \begin{bmatrix} w_{0,0} & 0 & 0 & 0 \\ w_{0,1} & 0 & 0 & 0 \\ 0 & w_{0,0} & 0 & 0 \\ 0 & w_{0,1} & 0 & 0 \\ w_{1,0} & 0 & 0 & 0 \\ w_{1,1} & 0 & 0 & 0 \\ 0 & w_{1,0} & 0 & 0 \\ 0 & w_{1,1} & 0 & 0 \\ 0 & 0 & w_{0,0} & 0 \\ 0 & 0 & w_{0,1} & 0 \\ 0 & 0 & 0 & w_{0,0} \\ 0 & 0 & 0 & w_{0,1} \\ 0 & 0 & w_{1,0} & 0 \\ 0 & 0 & w_{1,1} & 0 \\ 0 & 0 & 0 & w_{1,0} \\ 0 & 0 & 0 & w_{1,1} \end{bmatrix}$$

De plus, on a bien l'égalité :

$$\begin{bmatrix} w_{0,0} & 0 & 0 & 0 \\ w_{0,1} & 0 & 0 & 0 \\ 0 & w_{0,0} & 0 & 0 \\ 0 & w_{0,1} & 0 & 0 \\ w_{1,0} & 0 & 0 & 0 \\ w_{1,1} & 0 & 0 & 0 \\ 0 & w_{1,0} & 0 & 0 \\ 0 & w_{1,1} & 0 & 0 \\ 0 & 0 & w_{0,0} & 0 \\ 0 & 0 & w_{0,1} & 0 \\ 0 & 0 & 0 & w_{0,0} \\ 0 & 0 & 0 & w_{0,1} \\ 0 & 0 & w_{1,0} & 0 \\ 0 & 0 & w_{1,1} & 0 \\ 0 & 0 & 0 & w_{1,0} \\ 0 & 0 & 0 & w_{1,1} \end{bmatrix} \times \begin{bmatrix} x_{0,0} \\ x_{0,1} \\ x_{1,0} \\ x_{1,1} \end{bmatrix} = Y_{flatten} = \begin{bmatrix} x_{0,0}w_{0,0} \\ x_{0,0}w_{0,1} \\ x_{0,1}w_{0,0} \\ x_{0,1}w_{0,1} \\ x_{0,0}w_{1,0} \\ x_{0,0}w_{1,1} \\ x_{0,1}w_{1,0} \\ x_{0,1}w_{1,1} \\ x_{1,0}w_{0,0} \\ x_{1,0}w_{0,1} \\ x_{1,1}w_{0,0} \\ x_{1,1}w_{0,1} \\ x_{1,0}w_{1,0} \\ x_{1,0}w_{1,1} \\ x_{1,1}w_{1,0} \\ x_{1,1}w_{1,1} \end{bmatrix}$$

et

$$Y = \begin{bmatrix} x_{0,0}w_{0,0} & x_{0,0}w_{0,1} & x_{0,1}w_{0,0} & x_{0,1}w_{0,1} \\ x_{0,0}w_{1,0} & x_{0,0}w_{1,1} & x_{0,1}w_{1,0} & x_{0,1}w_{1,1} \\ x_{1,0}w_{0,0} & x_{1,0}w_{0,1} & x_{1,1}w_{0,0} & x_{1,1}w_{0,1} \\ x_{1,0}w_{1,0} & x_{1,0}w_{1,1} & x_{1,1}w_{1,0} & x_{1,1}w_{1,1} \end{bmatrix}$$

Question 5 (3-2-3-7). (**Champ récepteur**)

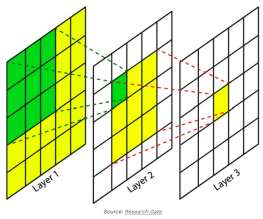

FIGURE 2 – Illustration of Receptive Field

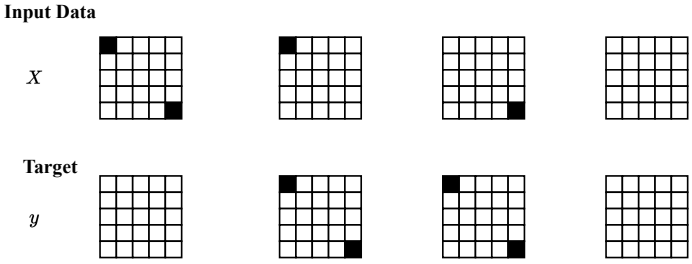

FIGURE 3 – Données illustratives pour la question 5, où X sont les images d'entrée et y sont les images de sortie de la vérité au sol. *Ceci n'est qu'une figure illustrative, les images réelles pourraient ne pas être de 5 ×5*

Le champ réceptif est défini comme la taille de la région dans l'entrée qui produit une caractéristique.. Par Exemple, Lorsque vous utilisez une couche Conv-3x3, chaque fonction est générée par une grille 3x3 dans l'entrée. Avoir deux couches Conv-3x3 successivement conduit à un champ réceptif 5x5. La figure ci-dessous en donne un exemple 2. Notez qu'il y a un champ réceptif associé à chaque fonctionnalité (par exemple, ici c'est pour la fonctionnalité de couleur jaune au Calque 3), où une fonctionnalité est une cellule dans la carte d'activation de sortie.

Considérez une architecture réseau composée uniquement de **4 Couches de convolutions, chacune avec la taille du noyau = 3, foulée = 1, zero-padding d'un pixel sur les quatre côtés, canaux intermédiaires = 16, et à un canal de sortie.**

1. Trouvez la résolution de la carte de sortie lorsque l'entrée du modèle est de taille 32×32. La taille de sortie est-elle la même que la taille d'entrée ?
2. Existe-t-il des cas où l'on pourrait envisager des extrants de la même taille que l'intrant (dans la résolution spatiale) ? Veuillez donner des exemples de tels cas.
3. Quel est le champ réceptif des caractéristiques finales obtenues à travers cette architecture, qui est, quelle est la taille de l'entrée qui correspond à une position de pixel particulière dans la sortie ?
4. Supposons qu'on vous donne des données, comme le montre la figure 3. qui est, vous avez 7×7 resolution images, où si les deux pixels de coin opposés en diagonale dans l'entrée sont les mêmes, puis la carte de sortie (de la même taille), a les coins correspondants en blanc. Si les deux coins de l'entrée sont différents, les coins correspondants de la carte de sortie sont noir. Tous les cas possibles de saisie sont décrits dans la figure 3.
   Compte tenu de l'architecture décrite ci-dessus, serait-il possible d'apprendre à bien faire avec ces données ? Autrement dit, le modèle peut-il obtenir une précision de 100% sur ces données ? Est-ce que quelque chose change si vous ajoutez une connexion résiduelle entre deux couches ?

Answer 5. 1. On a :

$$o = \lfloor \frac{i + 2p - k}{s} \rfloor + 1$$

Donc l'output est de taille 32 x 32 lorsque l'entrée du modèle est de taille 32 x 32.
La taille de sortie est la même que la taille d'entrée.

2. Oui, il existe des cas où l'on pourrait envisager des sorties de même taille que les entrées.
   Exemples :
   (a) La sélection de caractéristique (features selection). C'est un processus utilisé en apprentissage automatique et en traitement de données qui consiste à trouver un sous-sensemble de variables pertinentes dans une image.
   (b) La génération d'images à partir d'une image donnée en entrée.
3. La taille de l'entrée qui correspond à une seule position de pixel particulière dans la sortie est de 9×9.
   On souhaite calculer la taille du champ receptif de l'entrée r_0 pour une position de pixel particulière dans la sortie, ici $r_4 = 1$.
   En effet, on remarque une relation de récurrence :

$$r_{l-1} = stride\_size_l . r_l + (kernel\_size_l - stride\_size_l)$$

En posant $r_4 = 1$, on a :

$$r_3 = 1 \times 1 + (3 - 1) = 3$$
$$r_2 = 1 \times 3 + (3 - 1) = 5$$
$$r_1 = 1 \times 5 + (3 - 1) = 7$$
$$r_0 = 1 \times 7 + (3 - 1) = 9$$

D'où le résultat : $r_0 = 9$.