Laissez des traces de votre démarche pour toutes les questions !
For all questions, show your work !

1. **(4 points) Rétropropagation** / ***Backpropagation***
   Soit un réseau de neurones composé d'une couche d'entrée, d'une couche cachée sigmoïde et d'une couche de sortie *softmax*. La fonction de coût est l'entropie croisée. On dispose de plusieurs exemples d'entraînement et les mises à jour de paramètres se font toujours après avoir observé l'ensemble d'entraînement en entier. On emploie une taille de pas (*learning rate*) suffisamment petit, mais suffisamment grand pour que le réseau apprenne. On s'arrête lorsque la mise à jour des paramètres laisse les poids inchangés. Pour les questions qui suivent, répondez par oui ou par non, et accompagnez votre réponse d'une brève explication (une phrase suffit).

   (a) Est-ce que pour la situation décrite ci-haut on a la garantie que le vecteur de poids se rapproche, à chaque mise à jour des paramètres, du vecteur de poids associé au minimum de la fonction de coût (ou à un des vecteurs, s'il en existe plusieurs) ?

   (b) Est-ce que pour la situation décrite ci-haut on a la garantie que la fonction de coût diminue après chaque mise à jour des paramètres ?

   (c) Est-ce que pour la situation décrite ci-haut on a la garantie de converger vers un vecteur de poids qui donne lieu à la plus petite fonction de coût possible ?

   (d) On prédit la classe d'une image en choisissant l'index du neurone de sortie qui donne la plus grosse probabilité lorsqu'on présente cette image au réseau. Est-ce que pour la situation décrite ci-haut on a la garantie que le nombre d'erreurs de classification du réseau sur l'ensemble d'entraînement (en considérant la stratégie de classification décrite précédemment) n'augmente jamais ?

   Consider training a network consisting of input units, a hidden layer of logistic units, and a softmax output layer. The objective function is the cross-entropy loss. We have many training cases and we always compute our weight update based on the entire training set, using the error backpropagation algorithm. We use a learning rate that's small enough for all practical purposes, but not so small that the network doesn't learn. We stop when the weight update becomes zero. For each of the following questions, answer yes or no, and explain very briefly (one short sentence can be enough).

   (a) Does this training set-up guarantee that the weight vector gets closer, on every step, to the weight vector that gives the smallest possible loss value (or to one such weight vector if there are multiple) ?

   (b) Does this training set-up guarantee that the loss gets better (i.e. smaller) on every step ?

   (c) Does this training set-up guarantee that eventually we'll reach a weight vector that gives the smallest possible loss value ?

   (d) We use our network to classify images by simply seeing which of the 10 output units gets the largest probability when the network is presented with the image, and declaring the number of that output unit to be the network's guessed label. Does our training set-up guarantee that the number of mistakes that the network makes on training data (by following this classification strategy) never increases ?

2. **(6 points) Représentations invariantes et équivariantes** / ***Invariant and Equivariant Representations***
   (a) Quelle est la différence entre une représentation invariante et une représentation équivariante ? Expliquez en deux ou trois phrases.
   (b) Pouvez-vous donner en exemple une représentation qui soit à la fois invariante et équivariante ? Si oui, décrivez cet exemple.
   (c) La fonction d'activation *maxout* est-elle plus invariante ou équivariante ? Expliquez votre réponse.

   (a) What is the difference between an invariant representation and an equivariant representation ? Explain in 2-3 sentences.
   (b) Can you imagine a representation that is both invariant and equivariant ? If so, describe an example of such a feature.
   (c) Is the maxout unit best described as a invariant or an equivariant feature ? Explain your answer.

3. **(12 points) Optimisation et régularisation** / ***Optimization and Regularization***
   (a) Comparez et contrastez (en spécifiant les différences, avantages et désavantages) des algorithmes d'optimisation suivants : Adagrad, RMSprop et Adadelta.
   (b) En quoi le *momentum* habituel diffère-t-il du *Nesterov momentum* ?
   (c) Pourquoi le *early-stopping* prévient-il le sur-apprentissage ? Pensez-vous que c'est vrai dans un contexte de régression linéaire ?

   (a) Compare and contrast (providing specific differences, advantages and disadvantages) the following optimization algorithms : Adagrad, RMSprop, Adadelta.
   (b) What are the differences between standard Momentum and Nesterov Momentum ?
   (c) Explain why early-stopping prevents overfitting ? Would you expect it to work in the context of linear regression ?

4. **(12 points) Machine de Boltzmann restreinte et machine de Boltzmann profonde / *Restricted Boltzmann Machine and Deep Boltzmann Machine***
Soit un modèle graphique probabiliste non dirigé défini pour les vecteurs binaires $\boldsymbol{x} \in \{0,1\}^l$,$\boldsymbol{y} \in \{0,1\}^m$,$\boldsymbol{z} \in \{0,1\}^n$. La distribution jointe d'un tel modèle est $P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$ pour la fonction d'énergie suivante :

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = -\boldsymbol{x}^\top W \boldsymbol{y} - \boldsymbol{x}^\top U \boldsymbol{z} - \boldsymbol{y}^\top V \boldsymbol{z} - a\boldsymbol{x}^\top \boldsymbol{x} - b\boldsymbol{y}^\top \boldsymbol{y} - c\boldsymbol{z}^\top \boldsymbol{z} \tag{1}$$

où a, b, c sont des constantes de décalage scalaires et W, U, Z sont des matrices de poids avec les dimensions appropriées (par exemple, W est une matrice de taille $l \times m$).

(a) Dérivez les équations des distributions conditionnelles $P(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$, $P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$ et $P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y})$. Laissez des traces de votre démarche pour une seule distribution conditionnelle ; pour les autres, la réponse suffit.

(b) Décrivez une méthode efficace pour échantillonner de la distribution jointe. Soyez précis dans votre description et indiquez pourquoi votre algorithme est efficace.

(c) Considérons $\boldsymbol{x}$ comme une variable visible ou observée. La distribution postérieure $P(\boldsymbol{z}, \boldsymbol{y} \mid \boldsymbol{x})$ est-elle calculable de façon traitable, comme pour une RBM, ou intraitable, comme pour une DBM ? Expliquez votre réponse.

(d) Ce modèle peut-il être entraîné par l'algorithme de *Contrastive Divergence* (CD) ? Si c'est le cas, décrivez comment on pourrait appliquer l'algorithme CD à ce modèle, et sinon, expliquer pourquoi. Dans les deux cas, quelques phrases devraient suffire.

Consider an undirected probabilistic model over three binary random vectors : $\boldsymbol{x} \in \{0,1\}^l$,$\boldsymbol{y} \in \{0,1\}^m$,$\boldsymbol{z} \in \{0,1\}^n$. *As an undirected graphical model, the joint probability distribution is given by* $P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$ *following energy function :*

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = -\boldsymbol{x}^\top W \boldsymbol{y} - \boldsymbol{x}^\top U \boldsymbol{z} - \boldsymbol{y}^\top V \boldsymbol{z} - a\boldsymbol{x}^\top \boldsymbol{x} - b\boldsymbol{y}^\top \boldsymbol{y} - c\boldsymbol{z}^\top \boldsymbol{z} \tag{2}$$

where a, b, c *are each scalar offsets and* W, U, Z *are weight matrices of the appropriate size (e.g.* W *is an* $l \times m$ *matrix).*

(a) Derive the conditional distributions $P(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$, $P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$ *and* $P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y})$. *You only need to show your work for one of these.*

(b) Describe an efficient method of sampling from the joint probability distribution. Be specific in your description and indicate why your algorithm is efficient.

(c) Let us consider $\boldsymbol{x}$ *as the visible or observed variable. Is the posterior distribution(i.e.* $P(\boldsymbol{z}, \boldsymbol{y} \mid \boldsymbol{x})$*) tractable to compute, like the RBM, or is it intractable, like the DBM ? Explain your answer.*

(d) Is this model amenable to learning via the Contrastive Divergence (CD) algorithm ? If so, provide a description of how one could apply CD to this model, if not, explain why. In either case, a few sentences should suffice.

5. **(4 points) Autoencodeur variationnel (VAE)** / ***Variational Autoencoder (VAE)***

   (a) L'autoencodeur variationnel représente la borne inférieure variationnelle habituelle comme une combinaison d'un terme de reconstuction et d'un terme de régularisation. En partant de la borne inférieure suivante, obtenez ces deux termes et spécifiez lequel est le terme de reconstruction et lequel est le terme de régularisation :

$$\mathcal{L}(q) = \int q(\boldsymbol{z} \mid \boldsymbol{x}) \log \left(\frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{x})} \right) \, d\boldsymbol{z} \tag{3}$$

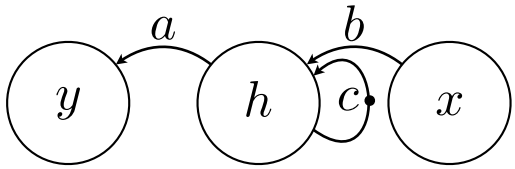
   (b) L'autoencodeur variationnel et la machine de Boltzmann profonde font tous deux appel à l'inférence variationnelle approximative. En classe, on a souligné que pour un de ces deux modèles, l'inférence variationnelle est faite de façon paramétrique tandis que pour l'autre elle est faite de façon non paramétrique. Expliquez ce qu'on entend par cela, et spécifiez quel modèle fait appel à quel type d'inférence (paramétrique ou non paramétrique).

(a) The Variational Autoencoder represents the standard variational lower bound as the combination of a reconstruction term and a regularization term. Starting with the variational lower bound below, derive these two terms, specifying which is the reconstruction term and which is the regularization term.

$$\mathcal{L}(q) = \int q(\boldsymbol{z} \mid \boldsymbol{x}) \log \left(\frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{x})} \right) \, d\boldsymbol{z} \tag{4}$$

(b) Both the Variational Autoencoder and the Deep Boltzmann Machine use variational approximate inference. In class, we discussed that one was a parametric application of variational inference and the other uses a more standard non-parametric application of variational inference. Explain what was meant by this and specify which method (VAE or DBM) corresponds to what kind of variational inference (parametric or non-parametric).

6. **(12 points) Rétropropagation dans le temps** / ***Backpropagation Through Time***
Soit le réseau de neurones récurrent ci-bas. Le flèche avec un point indiquent un délai d'un pas de temps (la connexion associée au poids c). Les flèches sans point ne sont pas soumises à un délai temporel (la connexion associée au poids a et b). Les unités y et h ont toutes deux une fonction d'activation sigmoïde.



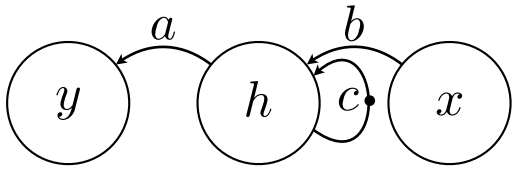
(a) Dessinez le graphe computationnel déroulé de ce RNN pour trois pas de temps **(de** $t = 1$ **à** $t = 3$**)**. Indiquez les activations cachées de la façon suivante : $h_1, h_2, \ldots$ et étiquetez toutes les connexions et leurs poids correspondants.

(b) Soit un exemple d'entraînement composé de la paire d'entrée/sortie $\boldsymbol{x}, \hat{\boldsymbol{y}}$ où $\hat{\boldsymbol{y}} \in [0,1]^3$, et soit la sortie du modèle y **(de** $t = 1$ **à** $t = 3$**)**. On emploie l'erreur quadratique comme fonction de coût : $E = \sum_{t=1}^{3} e_t$ où $e_t = \frac{1}{2}(y_t - \hat{y}_t)^2$. Est-ce que la valeur de h_t influence uniquement la sortie au temps t ? Sinon, pour chaque h_t (pour $t = 1$ à $t = 3$), spécifiez quelle sortie il influence.

(c) Dans ce contexte, calculez la dérivée de la fonction de coût par rapport au paramètre b, c'est-à-dire $\frac{\partial C}{\partial b}$.

Consider the Recurrent Neural Network shown below. The arrow with the dot indicates delays by one time step (i.e. connection associated with weight c). Arrows without dots indicate no time delay (i.e. connection associated with weights a and b). Units y and h both have a logistic sigmoid activation functions, x is the input (with no activation function). There are no biases. So for example, the activation of h at a time t is given by $h_t = \text{sigmoid}(ch_{t-1} + bx)$. *Assume that* $h_0 = d$ *(a constant). Recall that :*

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{5}$$



(a) Plot the computational graph for this RNN, i.e. unroll the RNN in time for 3 time steps **(from** $t = 1$ **to** $t = 3$**)**. *Indicate hidden unit activations with subscripts that denote time (e.g.* $h_0, h_1, \ldots$ *) and label all connections with their corresponding weight labels.*

(b) Consider a training data point consisting of input/output pairs $(\boldsymbol{x}, \hat{\boldsymbol{y}})$ *where* $\hat{\boldsymbol{y}} \in [0,1]^3$. *Using a squared error objective function, we have* $E = \sum_{t=1}^{3} e_t$ *where* $e_t = \frac{1}{2}(y_t - \hat{y}_t)^2$. *Does the value of* h_t *affect only the output at time t (i.e. the value of* y_t*) ? If not, for each* h_t *(from* $t = 1$ *to* $t = 3$*) specify which output it affects.*

(c) For the setup above, compute the gradient necessary to update the parameter b given the data point, i.e. compute $\frac{\partial E}{\partial b}$.