Contrôle périodique
INF8225 IA : techniques probabilistes et d'apprentissage

Toute documentation sur papier est autorisée.

All forms of documentation on paper are allowed.

Question	Points
Q1	20
Q2	12
Q3	10
Total	42

Hiver 2024, le 1 mars 12h45 à 15h35.

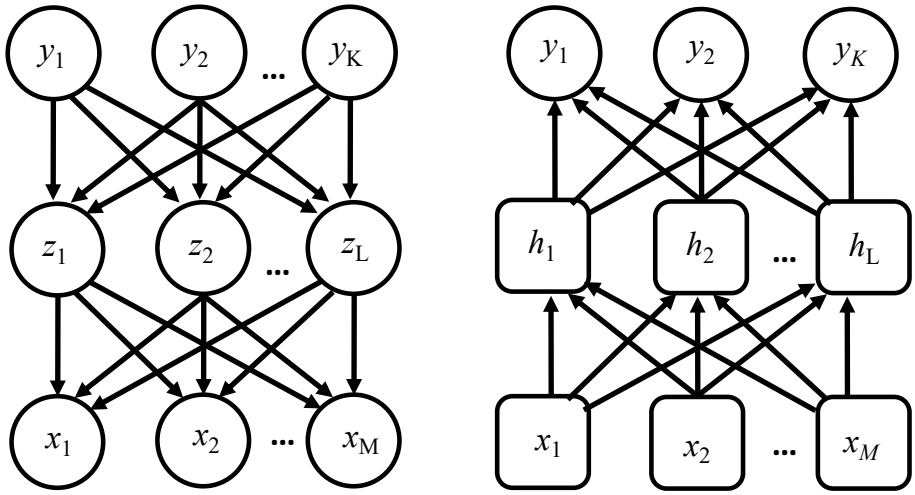

FIGURE 1 – (À gauche) Un réseau bayésien où toutes les variables sont binaires et toutes les probabilités sont stockées dans des tableaux. (À droite) Un réseau neuronal standard qui prendra des entrées constituées de caractéristiques binaires où chaque élément de $\mathbf{x} = [x_1, \ldots, x_M]$ vaut zéro ou un, et le réseau modélise K variables binaires de sorti $y_1 \ldots y_K$, qui pourrait également être exprimé sous forme de vecteur $\mathbf{y} = [y_1, \ldots, y_K]$.

Question 1 (version française) (20 points)

a) Écrivez une équation pour le modèle probabiliste que représente chaque réseau. Assurez-vous que la factorisation et les compositions de fonctions ressortent clairement de vos équations et assurez-vous que les paramètres de votre modèle sont clairement indiqués.

b) Fournissez une expression du nombre de paramètres pour chaque modèle en fonction de K, L et M et expliquez comment vous l'avez obtenue.

c) Si vous aviez des données d'entraînement sous la forme de paires N $\{\mathbf{x}, \mathbf{y}\}_{i=1\ldots N}$. Quelle serait votre fonction objectif pour effectuer l'apprentissage dans chaque modèle ?

d) Si vous disposiez de données pour seulement $\mathbf{x}_{i=1\ldots N}$, pourriez-vous toujours effectuer un apprentissage avec ces deux modèles ? Si oui, comment pourriez-vous procéder ?

e) Comment calculeriez-vous $P(y_1|\tilde{\mathbf{x}})$ et $P(z_1|\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ pour le modèle de gauche ?

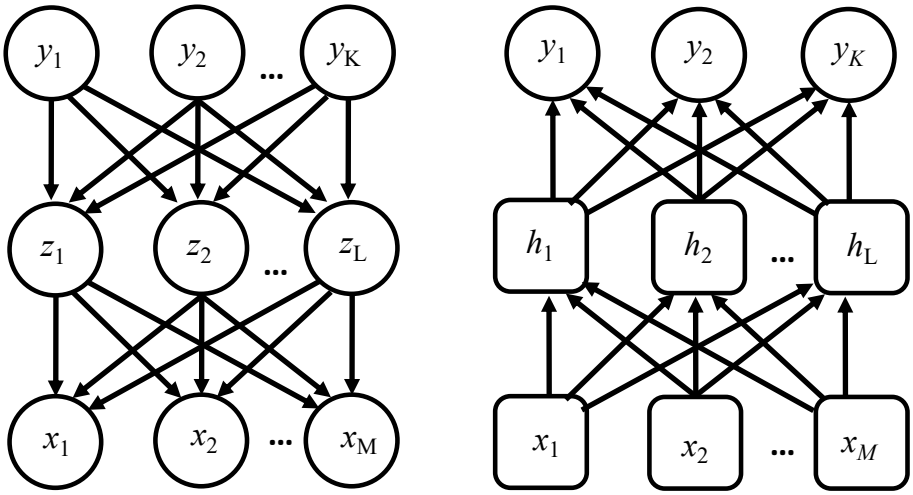

FIGURE 2 – (Left) A Bayesian Network where all variables are binary and all probabilities are stored in tables. (Right) A standard neural network that will take inputs consisting of binary features where each element of $\mathbf{x} = [x_1, \ldots, x_M]$ is either zero or one, and the network outputs K binary variables $y_1 \ldots y_K$, which could also be expressed as a vector $\mathbf{y} = [y_1, \ldots, y_K]$.

Question 1 (English version) (20 points)

a) Write an equation for the probability model that each network represents. Make sure that the factorization and the function compositions are clear from your equations and make sure that the parameters of your model are clearly indicated.

$$P\left(x_1, x_2 \ldots, x_M, z_1, \ldots, z_L, y_1, \ldots y_k\right) = \prod_{k=1}^{k} P\left(y_k\right) \prod_{l=1}^{L} P\left(z_l \mid y_1, \ldots y_k\right) \prod_{m=1}^{M} P\left(x_m \mid z_l, \ldots, z_L\right) \text{(Bayes Net on left)}$$

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{k} P\left(y_i \mid \mathbf{x}\right), P\left(y_i \mid \bar{x}\right) = \text{Beroulli}(y_i, f(h(x))) = \text{Bern}\left(y_i, f[\mathbf{W}\mathbf{h}(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{b}]\right) \quad \text{(Neural Net on right)}$$

One could also use sigmoids in place of the way in which we have written the Bernoulli distributions.

b) Provide an expression for the number of parameters for each model as a function of K, L and M and explain how you obtained this.

For the Bayesian Network the unconditional probabilities each have 2 parameters, so there are $2K$, for the layer of zs there are $L \cdot 2^{K+1}$, for the layer of xs we have $M \cdot 2^{L+1}$. So in total we have : $2K + L \cdot 2^{K+1} + M \cdot 2^{L+1}$ parameters. See the previous practice midterms for similar examples of this technique for determining model complexity and the discussions we had about the previous midterms.

For the Neural Network we just have the parameters associated with the two linear transformations : 1) $\mathbf{W}\mathbf{x} + \mathbf{b}$ and 2) $\mathbf{W}\mathbf{h} + \mathbf{b}$, this gives us $M \cdot L + L + L \cdot K + K$ parameters.

c) If you had training data in the form of N pairs $\{\mathbf{x}, \mathbf{y}\}_{i=1\ldots N}$. What would be your objective function to perform learning in each model ?

$-\sum_{i=1}^{N} \log\left(P\left(\tilde{\mathbf{y}}_i \mid \tilde{\mathbf{x}}_i\right)\right)$ (For the Neural Net on right.)

$$-\sum_{i=1}^{N} \log P\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i\right) = -\sum_{i=1}^{N} \log \left[\sum_{\mathbf{z}_i} P\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i, \mathbf{z}_i\right)\right] = -\sum_{i=1}^{N} \log \left[\sum_{z_1}\sum_{z_2}\dots\sum_{z_L} P\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i, \mathbf{z}_i\right)\right] \text{ (Bayes Net on left)}$$

d) If you had data for only $\mathbf{x}_{i=1\dots N}$, could you still perform learning with both of these models ? If so, how might you do it ?

For the Bayesian Network, one can optimize the marginal likelihood. As discussed in class, when random variables are unobserved, one must marginalize over them to obtain a single scale quantity for our objective function to be minimized/maximized.

$-\sum_{i=1}^{N} \log p\left(\tilde{\mathbf{x}}_i\right) = \sum_{i=1}^{N} \log \sum_{y_1}\dots\sum_{y_N}\sum_{z_1}\dots\sum_{z_N} p(\tilde{\mathbf{x}}, \mathbf{y}, \mathbf{z})$

For the Neural Network, one could answer No, it is not possible, or define the output to be the same as the input and train an autoencoder (potentially with noise).

e) How would you compute $P(y_1|\tilde{\mathbf{x}})$ and $P(z_1|\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ for the model on the left ?

We need to remember to marginalize over both the zs and all the other ys that are not y_1.

$$P\left(y_1 \mid \bar{x}\right) = \frac{\sum_{z_1}\cdots\sum_{z_l}\overbrace{\sum_{y_2}\cdots\sum_{y_N}}^{\text{Attention}} p(y_1,\cdots y_K, z_1,\dots z_L, x_1,\dots x_M)}{\sum_{z_1}\cdots\sum_{z_L}\sum_{y_1}\underbrace{\sum_{y_1}\cdots\sum_{y_N}}_{\text{Attention}} p(\{y\}_{1\dots K}\{z\}_{1\dots L}, \{x\}_{1\dots M})}$$

We have similar marginalizations for the second quantity. This is really just more generalized version of what we did for TP1 part 1.

$$P(z_1|\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \frac{\sum_{z_2}\cdots\sum_{z_N} p(\mathbf{z}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})}{\sum_{z_1}\cdots\sum_{z_N} P(\mathbf{z}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})}$$

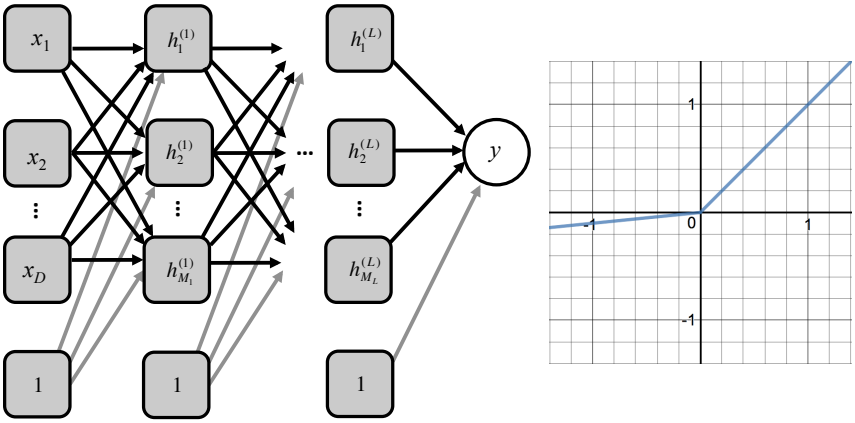

FIGURE 3 – (Left) A neural network. (Right) The LeakyReLU activation function.

Question 2 (version française) (12 points)

a) (8 points) Si vous aviez besoin de faire une prédiction continue et unidimensionnelle à valeur réelle y avec un réseau tel que celui illustré dans la figure ci-dessus :
i) Comment définiriez-vous l'activation finale avant la perte ?
ii) Comment écririez-vous un modèle de probabilité conditionnelle pour votre problème de prédiction ?
iii) Comment écririez-vous la fonction de perte sur un ensemble de données composé d'entrées $\mathbf{x}_{i=1\ldots N}$ et de sorties $y_{i=1\ldots N}$?
iv) Si vous deviez faire une prédiction pour un vecteur continu bidimensionnel de valeur $\mathbf{y}$, comment définiriez-vous le nouveau modèle et la probabilité conditionnelle ? (Fournissez une équation)

b) (2 points) Expliquez pourquoi l'utilisation de la fonction Leaky ReLU pourrait présenter des avantages en termes de propagation de gradient lors de la rétro-propagation par rapport aux ReLU (Rectified Linear Units) classiques. Notez que les fonctions d'activation de la couche cachée ReLU $h_k^{(l)}$ peuvent être spécifiées mathématiquement comme :

$$h_k^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \begin{cases} a_k^{(l)}(\mathbf{x}_i) & a_k^{(l)}(\mathbf{x}_i) > 0 \\ c \cdot a_k^{(l)}(\mathbf{x}_i) & a_k^{(l)}(\mathbf{x}_i) \leq 0 \end{cases}, \tag{1}$$

c) (2 points) L'algorithme d'Adam utilise un calcul de la forme suivante :
$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$. Appliquez cette fonction de manière récursive, définissez $m_{t-3} = 0$ et fournissez une équation qui exprime m_t comme une fonction *simplifiée* de β_1, g_t, g_{t-1} et g_{t-2}.

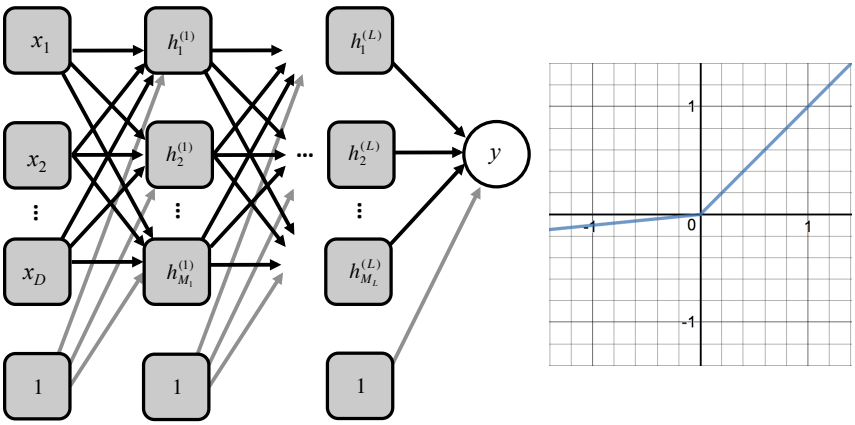

FIGURE 4 – (Left) A neural network. (Right) The LeakyReLU activation function.

Question 2 (English version) (12 points)

a) (8 points) If you needed to make a continuous, single dimensional real valued prediction y with a network such as the one illustrated in the figure above :
i) How would you define the final activation prior to the loss ?
ii) How would you write a conditional probability model for your prediction problem ?
iii) How would you the write the loss function over a dataset consisting of inputs $\mathbf{x}_{i=1...N}$ and outputs $y_{i=1...N}$?
iv) If you needed to make a prediction for a two dimensional continuous vector valued $\mathbf{y}$, how would you define the new model and conditional probability ? (Provide an equation)

i) It would be recommended to use either no activation function at all (so that it is just a linear prediction from the usual pre-activation) or a scaled form of the tanh function after making sure that the minimial and maximal values have been identified. If we use a sigmoid or a ReLU then we cannot predict negative values.

ii) $P(y|\mathbf{x}) = \mathcal{N}(y; f(\mathbf{x}), \sigma^2)$

iii) $-\sum_{i=1}^{N} \log P(y|\mathbf{x}_i) \rightarrow \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$
(Using the Normal or Gaussian distribution leads to a squared error loss)

iv) Simply use a two dimensional Gaussian, i.e. with a two dimensional mean and a covariance matrix, and create two output units connected to the rest of the network with no activation function or a tanh depending on your choice above. This can be written as : $P([y_1\ y_2]^T|\mathbf{x}) = \mathcal{N}(\mathbf{y}; f(\mathbf{x}), \mathbf{\Sigma})$.

b) (2 points) Explain why the use of the Leaky ReLU function might have advantages in terms of gradient propagation during back propagation compared to regular ReLUs (Rectified Linear Units). Note that ReLU hidden layer activation functions $h_k^{(l)}$ can be specified mathematically as :

$$h_k^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \begin{cases} a_k^{(l)}(\mathbf{x}_i) & a_k^{(l)}(\mathbf{x}_i) > 0 \\ c \cdot a_k^{(l)}(\mathbf{x}_i) & a_k^{(l)}(\mathbf{x}_i) \leq 0 \end{cases}, \tag{2}$$

In back propagation the gradient computed at the ouptut of the network is modified by repeated multiplications of the form $\Delta_l = \mathbf{D}_l \mathbf{W}_{(l+1)}^T \Delta_{(l+1)}$, where the $\mathbf{D}$ matrices are diagonal and contain the derivative of the activation functions evaluated at the pre-activation inputs. If we use a leaky ReLU then this matrix will contain ones and cs versus a regular ReLU which would contain ones and zeros (like a noisy identity matrix). This can allow gradient to flow a bit more easily backwards in the model.

c) (2 points) The Adam algorithm uses a computation of the following form :
$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$. Apply this function recursively, set $m_{t-3} = 0$ and provide an equation that expresses m_t as a *simplified* function of β_1, g_t, g_{t-1} and g_{t-2}.

$m_t = \beta_1 \cdot [\beta_1 \cdot m_{t-2} + (1 - \beta_1) \cdot g_{t-1}] + (1 - \beta_1) \cdot g_t$
$m_t = \beta_1 \cdot [\beta_1 \cdot [\beta_1 \cdot m_{t-3} + (1 - \beta_1) \cdot g_{t-2}] + (1 - \beta_1) \cdot g_{t-1}] + (1 - \beta_1) \cdot g_t$
$m_t = (1 - \beta_1)[\beta_1^2 \cdot g_{t-2} + \beta_1 g_{t-1} + g_t]$

Question 3 (French/English version) (10 points)

1) FR : (Vrai ou Faux) Il existe une solution globalement optimale pour les paramètres d'un réseau neuronal une seule couche d'entrée connectée à une seule unité de sortie et utilisant une fonction d'activation sigmoïdale pour un problème de classification binaire. EN : (True or False) There exists a globally optimal solution for the parameters of a neural network with a single input layer connected to a single output unit and using a sigmoidal activation function for a binary classification problem.

2) FR : (Vrai ou Faux) La régression polynomiale peut fournir une fonction non linéaire qui est estimée via la solution à un problème d'estimation linéaire. EN : (True or False) Polynomial regression can provide a nonlinear function that is estimated via the solution to a linear estimation problem.

3) FR : (Vrai ou Faux) La porte d'entrée dans un RNN LSTM utilise une fonction d'activation tanh. EN : (True or False) The input gate in an LSTM RNN uses a tanh activation function.

4) FR : (Vrai ou Faux) L'algorithme Adam n'a pas d'hyperparamètres. EN : (True or False) The Adam algorithm has no hyperparameters.

5) FR : (Vrai ou Faux) Adam consiste à estimer le premier et le deuxième moment du gradient. EN : (True or False) Adam involves estimating the first and second moment of the gradient.

6) FR : (Vrai ou Faux) Selon l'approche de modélisation de Pearl une intervention sur une variable X du modèle n'aura d'influence que sur les parents de X. EN : (True or False) According to Pearl's modeling approach, an intervention on a variable X in a model will only have an influence on the parents of X.

7) FR : (Vrai ou Faux) On peut apprendre les ResNets avec plus de 1000 couches. EN : (True or False) ResNets can be trained with more than 1000 layers.

8) FR : (Vrai ou Faux) Un modèle de Markov caché est un type de modèle génératif. EN : (True or False) A hidden Markov model is a type of generative model.

9) FR : (Vrai ou Faux) Si l'on voulait apprendre un réseau neuronal multicouche avec un ensemble de données pour faire des prédictions à l'avenir sur un ensemble de test inconnus, la meilleure stratégie est toujours d'utiliser le modèle le plus profond qui a la plus haute performance sur les données d'apprentissage. EN : (True or False) If one is fitting a multilayer neural network model to data to make future predictions on an unknown test set, the best strategy is always to use deepest model that has the highest training set performance.

10) FR : Pour une séquence de longueur n et une représentation de la dimensionnalité d, la complexité de l'autoattention dans un « transformer » est donnée par : EN : For a sequence of length n and a representation of dimensionality d, the per layer complexity of self attention in a transformer model is given by :
A. $O(1)$
B. $O(n)$
C. $O(n \cdot d)$
D. $O(n^2 \cdot d)$

Solution : V V F F V F V V F D