

1 Partie I (10 points)

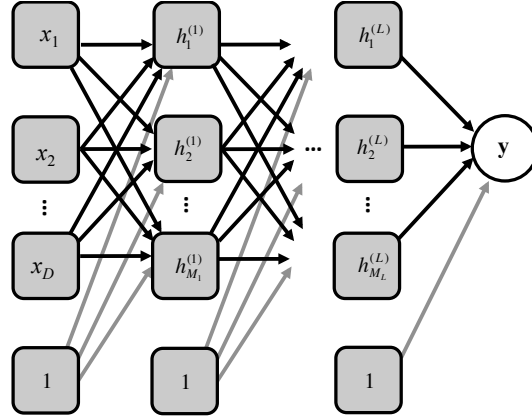


Figure 1: Un réseau de neurones.

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 un réseau de neurones avec une couche d'entrée avec $D = 100$ unités, L couches cachées, chacune avec 100 unités et un vecteur de sortie \mathbf{y} de dimension K . Vous avez $i = 1, \dots, N$ exemples, avec chaque valeur continue $y_{k,i} \in [0, 1]$, et chaque vecteur continu $\mathbf{x}_i \in \mathbb{R}^{100}$ dans un ensemble d'apprentissage. La fonction de perte est donnée par

$$L = \sum_{i=1}^N \sum_{k=1}^K (y_{k,i} - f_k(\mathbf{x}_i))^2 = \sum_{i=1}^N (\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i))^T (\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)) \quad (1)$$

où \mathbf{y}_i est un vecteur continu avec la cible qui doit être prédite. La fonction d'activation (vectorielle) de la couche finale a la forme $\mathbf{f} = [f_1, \dots, f_K]$ où chaque f_k est une sigmoïde:

$$f_k(a^{(L+1)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a_k^{(L+1)}(\mathbf{h}^{(L)}(\mathbf{x}_i)))}, \quad h_k^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a_k^{(l)}(\mathbf{x}_i))},$$

où $\mathbf{a}^{(l)} = [a_1^{(l)} \dots a_K^{(l)}]^T$ 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 $\boldsymbol{\theta}^{(l)} \hat{\mathbf{h}}^{(l-1)}$ en utilisant l'astuce de définir $\hat{\mathbf{h}}$ comme \mathbf{h} avec un 1 concaténé à la fin du vecteur.

a) (6 points) Donnez le pseudocode incluant des *calculs matriciels*—*vectoriels* détaillés pour l'algorithme de rétro-propagation pour calculer le gradient pour les paramètres de chaque couche étant donné un exemple d'entraînement.

b) (4 points) Imaginez que vous avez maintenant un jeu de données avec $N=500,000$ exemples. Expliquez comment vous utiliserez votre pseudocode pour optimiser ce réseau neuronal dans le contexte d'une expérience d'apprentissage machine correctement effectuée.

Part I – English version (10 points)

For this part, you may work in groups of 2, but you must write your own derivation and hand in your own report. If you work in a group, indicate the name of your partner in your report.

Consider a neural network with a single input layer having $D = 100$ units, L hidden layers, each having 100 units and a K dimensional output vector \mathbf{y} . You have $i = 1, \dots, N$ examples where each continuous valued $y_{k,i} \in [0, 1]$, and each continuous vector $\mathbf{x}_i \in \mathbb{R}^{100}$ is in a training set. The loss function is given by

$$L = \sum_{i=1}^N \sum_{k=1}^K (y_{k,i} - f_k(\mathbf{x}_i))^2 = \sum_{i=1}^N (\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i))^T (\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)) \quad (2)$$

where \mathbf{y}_i is a continuous vector with the target we wish to predict. The final layer has a set of activation functions encoded by the vector $\mathbf{f} = [f_1, \dots, f_K]$ where each f_k is a sigmoid function, such that

$$f_k(a^{(L+1)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a^{(L+1)}(\mathbf{h}^{(L)}(\mathbf{x}_i)))}, \quad h_k^{(l)}(\mathbf{a}^{(l)}(\mathbf{x}_i)) = \frac{1}{1 + \exp(-a_k^{(l)}(\mathbf{x}_i))},$$

where $\mathbf{a}^{(l)} = [a_1^{(l)} \dots a_K^{(l)}]^T$ is the vector resulting from the calculation of the usual preactivation function $\mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$, which could be simplified to $\boldsymbol{\theta}^{(l)}\hat{\mathbf{h}}^{(l-1)}$ using the trick of defining $\hat{\mathbf{h}}$ as \mathbf{h} with a 1 concatenated at the end.

a) (6 points) Give pseudocode including detailed *matrix-vector calculations* for the backpropagation algorithm to calculate the gradient for the parameters of each layer given a training example.

b) (4 points) Imagine that you now have a dataset with $N=500,000$ examples. Explain how you would use your pseudocode to optimize this neural network in the context of a properly performed machine learning experiment.

Partie II (20 points)

Pour cette partie, vous pouvez travailler en groupes de 2. Chacun doit soumettre son propre rapport, mais pour cette partie les rapports doivent être identiques si vous avez travaillé en groupe.

L'objectif de la partie II est de se familiariser avec la librairie Pytorch (ou Tensorflow + Keras, ou votre choix de logiciel). Vous allez expérimenter avec les données **Fashion MNIST**. Les données sont des images similaires à MNIST et consiste en 10 catégories de vêtements.

Exécutez des expériences avec différentes architectures, e.g. différents types et nombres de couches. Utilisez au moins un réseau de convolution et un réseau "fully connected". Utilisez un ensemble de validation pour sélectionner la meilleure architecture et utiliser l'ensemble de test seulement à la fin. Comparez vos résultats à l'aide de courbes d'apprentissage et de la log-vraisemblance. Vous pouvez vous référer au code disponible ici: <https://github.com/AlexPiche/INF8225/tree/master/tp2>, et aussi aux tutoriels pytorch ici: https://github.com/MaximumEntropy/pytorch_notebooks

Soumettez un rapport incluant les courbes d'apprentissage et le code de vos modèles.

Part II - English version (20 points)

For this part, you may work in groups of 2. Each person submits their own report, but for this part the reports have to be identical if you work in a group.

The objective of Part II is to familiarize yourself with the Pytorch library (or Tensorflow + Keras, or your choice of other software). You will experiment with **Fashion MNIST**. This is a dataset of images similar to MNIST, but consisting of 10 different categories of clothing.

Perform experiments with different architectures; e.g. use different types and numbers of layers. Use at least one convolutional network and one fully connected network. Use a validation set to select the best architecture, and use the test set only at the end. Compare your results based on learning curves and log-probability. You can refer to code available here: <https://github.com/AlexPiche/INF8225/tree/master/tp2>, and also to pytorch tutorials here: https://github.com/MaximumEntropy/pytorch_notebooks

Submit a report including the learning curves and code of your models.