IFT6266-H2015 **Quiz 1** Nom :
Prof : Aaron Courville Numero :

Chaque question doit être accompagnée d'une démarche !

For all questions, show your work !

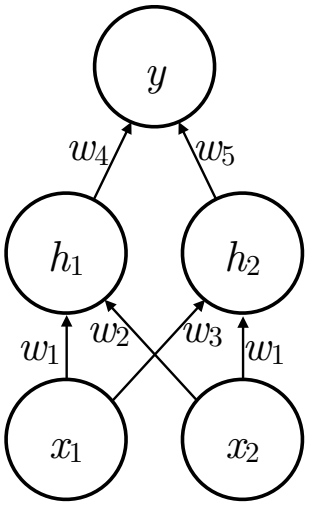
1. **Backpropagation**
   On s'intéresse à entraîner le réseau de neurones ci-bas à discriminer entre deux classes $\{0, 1\}$ via la fonction de coût d'entropie croisée (éq. 1).
   Consider training the network shown below for a binary classification $\{0, 1\}$ *task using a cross-entropy loss (eqn. 1).*

$$L = \sum_i \tilde{y}^{(i)} \log(y^{(i)}) + (1 - \tilde{y}^{(i)}) \log(1 - y^{(i)}) \tag{1}$$

$$y = \frac{1}{1 + \exp(-w_4 h_1 - w_5 h_2 - c)}$$

$$h_2 = w_3 x_1 + w_1 x_2$$

$$h_1 = \frac{1}{1 + \exp(-w_1 x_1 - w_2 x_2 - b)}$$



Initial values

$w_1 = 0.5$
$w_2 = -0.5$
$w_3 = -0.5$
$w_4 = 0.5$
$w_5 = -0.5$
$b = 0.0$
$c = 0.0$

   i En se servant de l'algorithme de rétropropagation du gradient, trouvez le gradient de la fonction de coût par rapport aux paramètres $w_1, w_2, w_3, w_4, w_5, b, c$. La réponse doit être exprimée en fonction des poids $(w_1, w_2, w_3, w_4, w_5, b, c)$, des activations (x_1, x_2, h_1, h_2, y) et de la classe $(\tilde{y}_1)$.
   Using the backpropagation algorithm, derive the gradient of the loss with respect to the parameters $w_1, w_2, w_3, w_4, w_5, b, c$*, in terms of the weight value* $(w_1, w_2, w_3, w_4, w_5, b, c)$*, the unit activations* (x_1, x_2, h_1, h_2, y) *and the label* $(\tilde{y}_1)$*.*

   ii Simulez deux itération de descente de gradient stochastique (un exemple par mise à jour des paramètres) avec *momentum*. Utilisez les hyperparamètres suivants : *learning rate* $\epsilon = 1$, *momentum parameter* $\alpha = 0.9$ et *initial velocity* $\boldsymbol{v} = \mathbf{0}$. Les données d'entraînement sont $\{\boldsymbol{x}^{(1)} = [1, -1], \tilde{y}^{(1)} = 0\}$, $\{\boldsymbol{x}^{(2)} = [0, 1], \tilde{y}^{(2)} = 1\}$. **Indiquez clairement les valeurs des paramètres et de la *velocity* après chaque mise à jour.**
   Run two iterations of stochastic gradient descent (with one example per update) with momentum. Assume learning rate $\epsilon = 1$*, momentum parameter* $\alpha = 0.9$ *and initial velocity (or momentum) state vector* $\boldsymbol{v} = \mathbf{0}$*. The training data is* $\{\boldsymbol{x}^{(1)} = [1, -1], \tilde{y}^{(1)} = 0\}$, $\{\boldsymbol{x}^{(2)} = [0, 1], \tilde{y}^{(2)} = 1\}$*.* ***Clearly indicate the parameter and velocity values after each update.***

.

.

2. **Convolutional Networks**

i Les réseaux de neurones convolutionnels (CNNs) sont largment plus performants que les MLPs conventionnels pour une tâche discriminative sur MNIST. Détaillez les trois différences principales entre les CNNs et les MLPs convenionnels et expliquez en quoi ces différences font que les CNNs sont plus performants.

A comparison of standard MLP and convolutional neural network approaches to MNIST classification indicate that CNNs have a large performance advantage. Review the 3 main differences between CNNs and standard MLPs and suggest why how these differences lead to performance improvements.

ii Soit un auto-encodeur convolutionnel (champ récepteur local, partage des poids, mais pas d'aggrégation) dans un contexte d'apprentissage non supervisé. Plus précisément, on s'intéresse à un modèle à une couche dont la k^{ieme} *feature map* est décrite par $\boldsymbol{h}_k = \sigma(\boldsymbol{W}_k * \boldsymbol{x} + \boldsymbol{b}_k)$ et la reconstruction est $\tilde{\boldsymbol{x}} = \sum_k \boldsymbol{h}_k * \tilde{\boldsymbol{W}}_k + \boldsymbol{c}$. Notez que $\sigma(\boldsymbol{x}) = 1/(1+\exp(-x))$ et que $\tilde{\boldsymbol{W}}_k$ désigne l'opération "flip" sur les deux axes appliquée à la k^{ieme} *feature map* $\boldsymbol{W}_k$.

Dans un contexte d'apprentissage non supervisé, on observe en pratique que les filtres obtenus après l'entraînement sont sensibles à des hautes fréquences spatiales (i.e. ils sont bruités et les corrélations entre les pixels voisins sont très faibles). Quelle caractéristique de l'auto-encodeur convolutionnel est responsable de ce phénomène ? Proposez une solution permettant d'obtenir des filtres plus semblables à ce qui est attendu (e.g. détecteurs de contours, etc.).

Consider the use of a convolutional autoencoder (local receptive fields, convolutional weight sharing, but without pooling) in an unsupervised learning application. Specifically, we are interested in a single layer convolutional model, with the k-th feature map given by $\boldsymbol{h}_k = \sigma(\boldsymbol{W}_k * \boldsymbol{x} + \boldsymbol{b}_k)$ *and the reconstruction given by* $\tilde{\boldsymbol{x}} = \sum_k \boldsymbol{h}_k * \tilde{\boldsymbol{W}}_k + \boldsymbol{c}$. *Where* $\sigma(\boldsymbol{x}) = 1/(1+\exp(-x))$ *and* $\tilde{\boldsymbol{W}}_k$ *is the "flip" operation over both dimensions applied to the k-th feature map* $\boldsymbol{W}_k$.

Empirically, we observe that, in the unsupervised setting, rather than getting smooth edge detectors, the trained filters often appear very high frequency (i.e. noisy, with low correlation between neighboring pixels). What feature of the convolutional autoencoder would cause this to happen ? Propose a solution that could allow one to recover more typical image features (e.g. edge detectors, etc.).

.

.

3. **CAE and Weight Decay**

i Prouvez que dans le cas d'un auto-encodeur linéaire accompagné d'une fonction de coût quadratique, la pénalité contractante est équivalente au *weight decay*. Dans votre réponse, présumez que la reconstruction de $\boldsymbol{x}$ est donnée par $\tilde{\boldsymbol{x}} = W^\top \boldsymbol{h}(\boldsymbol{x})$, où $\boldsymbol{h}(\boldsymbol{x}) = W\boldsymbol{x}$, et que la fonction de coût est $L = (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top(\boldsymbol{x} - \tilde{\boldsymbol{x}})$.

For the case of a linear autoencoder with a squared error loss function, prove that the contractive autoencoder (CAE) penalty is equivalent to weight decay. Specifically, for the linear auto-encoder of data $\boldsymbol{x}$, assume the reconstruction is $\tilde{\boldsymbol{x}} = W^\top \boldsymbol{h}(\boldsymbol{x})$, where $\boldsymbol{h}(\boldsymbol{x}) = W\boldsymbol{x}$ and the loss function is $L = (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top(\boldsymbol{x} - \tilde{\boldsymbol{x}})$.

ii En présumant l'équivalence ci-haut, décrivez si ou comment la pénalité contractante diffère du *weight decay* conventionnel lorsqu'appliquée à un auto-encodeur à une couche cachée dont les unités sont *rectified linear* (ReLus).

With the equivalence above thus established, describe if or how the CAE penalty is distinct from standard weight decay when they are both applied to an autoencoder with a single hidden layer of rectified linear units (ReLus).

.

.