Université Versailles
Saint-Quentin-en-Yvelines

Couche softmax

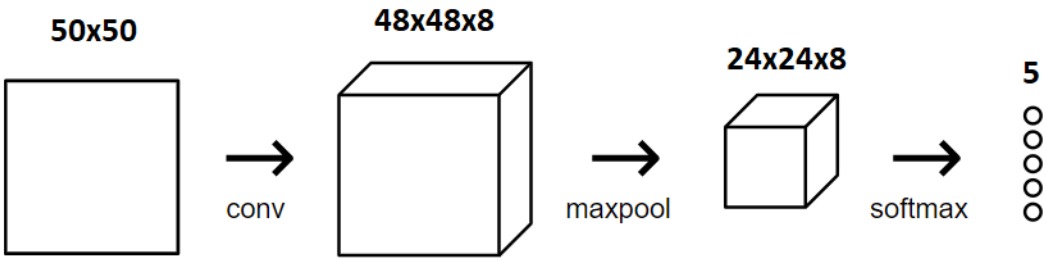
Tuteur :**Hugo Bolloré**

Softmax

Pour compléter notre CNN, nous devons lui donner la capacité de faire des prédictions. Nous allons le faire en utilisant la couche finale standard pour un problème de classification multiclasse : la couche Softmax, une couche entièrement connectée (dense) qui utilise la fonction Softmax comme son activation.

Usage de la couche

Nous utiliserons une couche softmax avec 5 nœuds, chaque nœud représente une plante, comme couche finale dans notre CNN. Chaque nœud de la couche sera connecté à chaque entrée. Une fois la transformation softmax appliquée, le chiffre représenté par le nœud avec la probabilité la plus élevée sera la sortie du CNN.



softmax va nous aider à quantifier notre degré de certitude quant à notre prédiction, ce qui est utile lors de la formation et de l'évaluation de notre CNN. Plus précisément, l'utilisation de softmax nous permet d'utiliser la perte d'entropie croisée, qui prend en compte notre certitude de chaque prédiction.

Calcul perte d'entropie croisée

$$L = -\ln(p_c)$$

où c est la bonne classe (dans notre cas, la bonne plantes),p_c est la probabilité prédite pour la classe c. Comme toujours, une perte plus faible est meilleure.
Par exemple, dans le meilleur des cas, nous aurions :

$$p_c = 1, L = \ln(1) = 0$$

Entrainement

L'entraînement d'un réseau de neurones se compose généralement de deux phases :

- Une phase **forward**, où l'entrée est entièrement passée à travers le réseau.
- Une phase **backward**, où les gradients sont rétropropagés **(backprop)** et les poids sont mis à jour.

Nous suivrons ce modèle pour former notre CNN. Nous utiliserons également deux grandes idées spécifiques à la mise en œuvre :

- Pendant la phase avant, chaque couche mettra en cache toutes les données (telles que les entrées, les valeurs intermédiaires, etc.) dont elle aura besoin pour la phase arrière. Cela signifie que toute phase arrière doit être précédée d'une phase avant correspondante.
- Pendant la phase de retour, chaque couche**recevra un gradient** et également **renverra un gradient**. Il recevra le gradient de perte par rapport à ses sorties$\left(\frac{\partial \mathbf{L}}{\partial \mathbf{out}}\right)$ et renvoie le gradient de perte par rapport à ses entrées $\left(\frac{\partial \mathbf{L}}{\partial \mathbf{in}}\right)$.

Backprop: Softmax

La première chose que nous devons calculer est l'entrée de la phase arrière de la couche Softmax, $\left(\frac{\partial \mathbf{L}}{\partial \mathbf{out_s}}\right)$,où out_s est la sortie de la couche Softmax : un vecteur de 5 probabilités

$$\frac{\partial L}{\partial out_s} = \begin{cases} 0 & \text{si } i \neq c \\ -\frac{1}{p_i} & \text{si } i = c \end{cases}$$

c est la classe correcte

Nous mettons ici en cache 3 choses qui seront utiles pour l'implementation de la phase arrière **(backward)** :

- La forme de l'entrée avant de l'aplatir (24x24x8).
- L'entrée après l'avoir aplatie.
- Le totale, qui est les valeurs passées à l'activation softmax.

Avec cela , nous pouvons commencer à dériver les gradients pour la phase backprop. Nous avons déjà dérivé l'entrée de la phase arrière Softmax : $\frac{\partial \mathbf{L}}{\partial \mathbf{out_s}}$.

Calcul gradient de out_c:

$$out_c = \frac{e^{t_c}}{\sum_i e^{t_i}}$$

Maintenant, considérons une classe k telle que k $\neq c$
On peut réécrire out_s comme :

$$out_s(c) = e^{t_c} S^{-1}$$

Règle de chaîne à dériver :

$$\frac{\partial out_s(c)}{\partial t_k} = \frac{\partial out_s(c)}{\partial S}\left(\frac{\partial S}{\partial t_k}\right)$$

$$= -e^{t_c} S^{-2}(\frac{\partial S}{\partial t_k})$$

$$= -e^{t_c} S^{-2}(e^{t_k})$$
$$= \frac{-e^{t_c} e^{t_k}}{S^2}$$

On commence par chercher pour c en recherchant un gradient non nul dans d_L_d_out. Une fois que nous avons trouvé cela, nous calculons le gradient $\frac{\partial \mathbf{out_s(i)}}{\partial \mathbf{t}}$ d_L_d_out en utilisant les résultats que nous avons obtenus ci-dessus :

$$\frac{\partial out_s(k)}{\partial t} = \begin{cases} \frac{-e^{t_c} e^{t_k}}{S^2} & \text{si } k \neq c \\ \frac{-e^{t_c}(S - e^{t_c})}{S^2} & \text{si } k \neq c \end{cases}$$

Nous voulons finalement les gradients de perte par rapport aux poids, aux biais et aux entrées :

- Nous allons utiliser le gradient de poids, $\frac{\partial \mathbf{L}}{\partial \mathbf{w}}$ pour mettre à jour les poids de notre couche.
- Nous allons utiliser le gradient de poids, $\frac{\partial \mathbf{L}}{\partial \mathbf{b}}$ pour mettre à jour les biases de notre couche.
- Nous allons retourner le gradient d'entrée, $\frac{\partial \mathbf{L}}{\partial \mathbf{input'}}$ à partir de notre *backprop*() méthode afin que la couche suivante puisse l'utiliser. C'est le gradient de retour

Pour calculer ces 3 gradients de perte, nous devons d'abord dériver 3 autres résultats : les gradients des totaux par rapport aux poids, aux biais et aux entrées. L'équation pertinente ici est :

$$t = w * input + b$$

$$\frac{\partial t}{\partial w} = input$$
$$\frac{\partial t}{\partial b} = 1$$
$$\frac{\partial t}{\partial input} = w$$

Tout assembler :

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial w}$$
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial b}$$
$$\frac{\partial L}{\partial input} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial b}$$