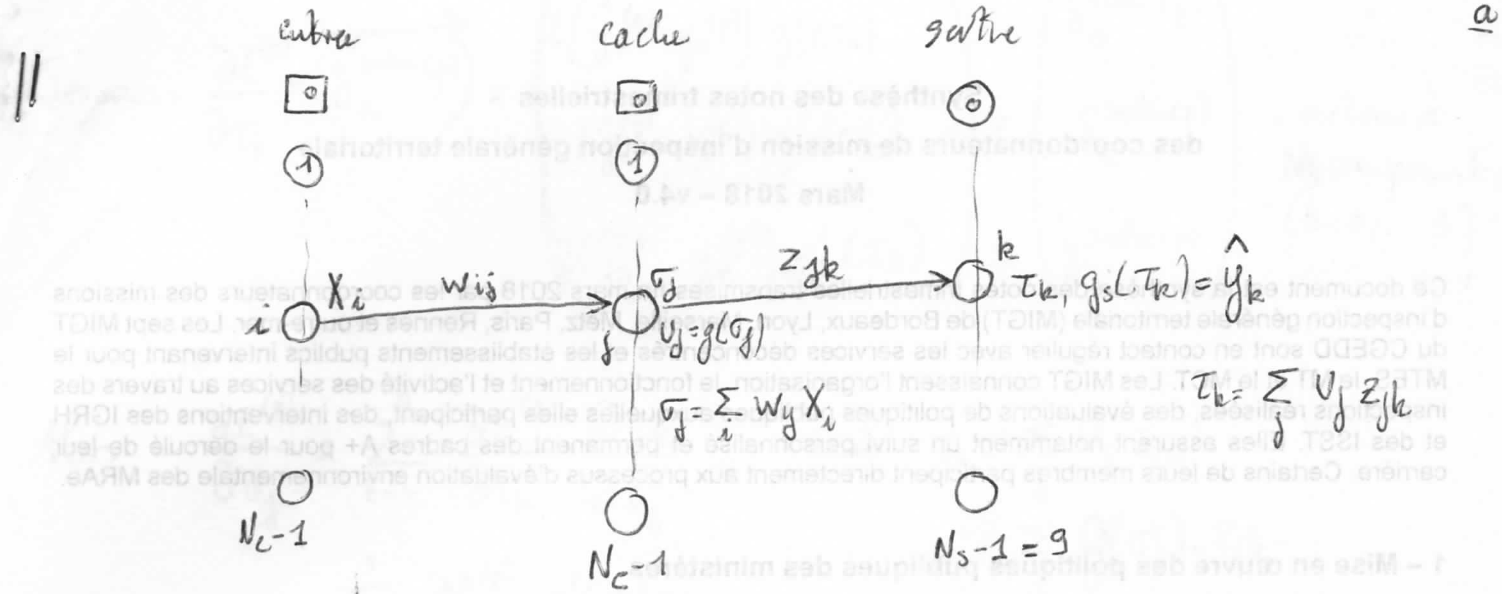
a



$j, g(x)$ $\quad$ $k, g_s(x)$

$$E^{(p)} = \sum_{k=0}^{9} \left(\hat{y}_k^{(p)} - y_k^{(p)}\right)^2$$ = erreur sur l'elt p de l'ensem

$$E = \frac{1}{P} \sum_{p=0}^{P-1} \sum_{k=0}^{9} \left(\underbrace{\hat{y}_k^{(p)}}_{\text{prediction}} - \underbrace{y_k^{p}}_{\text{true}}\right)^2 = \frac{1}{P} \sum E^{(p)}$$

$\hat{y}_k^{(p)}$ = prediction pour cas p d'être chiffre k

$y_k^{(p)}$ = vraie valeur de probabilité
= 0 si k ≠ true digit
= 1 si k = true digit

$$\frac{\partial E}{\partial x} = \frac{2}{P} \sum_{p=0}^{P-1} \sum_{k=0}^{9} \left(\hat{y}_k^{(p)} - y_k^{p}\right) \times \frac{\partial g_s(\tau_k)}{\partial x}$$

Si $x = \tau_k$: $$\frac{\partial E}{\partial \tau_k} = \frac{2}{P} \sum_{p=0}^{P-1} \left(\hat{y}_k^{(p)} - y_k^{(p)}\right) g_s'(\tau_k)$$ = vecteur à k composantes

Si $x = z_{jk}$: $$\frac{\partial E}{\partial z_{jk}} = \frac{2}{P} \sum_{p=0}^{P-1} \left(\hat{y}_k^{(p)} - y_k^{(p)}\right) g_s'(\tau_k) \frac{\partial \tau_k}{\partial z_{jk}}$$

$$= \frac{2}{P} \sum_{p=0}^{P-1} \left(\hat{y}_k^{(p)} - y_k^{(p)}\right) \times g_s'(\tau_k) \times U_j$$

Si $x = U_j$: $$\frac{\partial E}{\partial U_j} = \frac{2}{P} \sum_{p=0}^{P-1} \sum_{k=0}^{9} \left(\hat{y}_k^{(p)} - y_k^{(p)}\right) g_s'(\tau_k) \frac{\partial \tau_k}{\partial U_j}$$

$$= \frac{2}{P} \sum_{p=0}^{P-1} \sum_{k=0}^{9} \left(\hat{y}_k^{(p)} - y_k^{(p)}\right) g_s(\tau_k) \times z_{jk}$$

Donc : $\frac{\partial E^{(p)}}{\partial \tau_k} = \overrightarrow{(\delta_k^{\text{sortie}(p)})} = \begin{pmatrix} 2(\hat{y}_0^{(p)} - y_0^{(p)})\, g_s'(\tau_0) \\ \vdots \\ 2(\hat{y}_k^{(p)} - y_k^{(p)})\, g_s'(\tau_k) \\ \vdots \\ 2(\hat{y}_g^{(p)} - y_g^{(p)})\, g_s'(\tau_g) \end{pmatrix} = \begin{pmatrix} \delta_0^{\text{sortie}(p)} \\ \vdots \\ \delta_k^{\text{sortie}(p)} \\ \vdots \\ \delta_g^{\text{sortie}(p)} \end{pmatrix}$ = vecteur à N_S composantes $(k = 0, \dots, \underset{N_s - 1}{g})$

Alors $\frac{\partial E^{(p)}}{\partial \sigma_j} = \sum_{k=0}^{g} \frac{\partial E^{(p)}}{\partial \tau_k} \frac{\partial \tau_k}{\partial \sigma_j}$ mais $\tau_k = \sum_j U_j z_{jk}$

$= \sum_j g(\sigma_j) \cdot z_{jk}$

$= \sum_{k=0}^{g} \frac{\partial E^{(p)}}{\partial \tau_k} \times g'(\sigma_j) z_{jk}$ $\Leftarrow$ donc $\frac{\partial \tau_k}{\partial \sigma_j} = g'(\sigma_j) z_{jk}$

$= \sum_{k=0}^{g} [g'(\sigma_j) z_{jk}] \times \frac{\partial E^{(p)}}{\partial \tau_k}$ = vecteur à N_c composantes. $(j = 0, \dots, N_c - 1)$

$= \delta_j^{\text{caché}(p)}$

$$\overrightarrow{\delta}^{\text{caché}(p)} = \begin{pmatrix} \delta_0^{\text{caché}(p)} \\ \vdots \\ \delta_j^{\text{caché}(p)} \\ \vdots \\ \delta_{N_c-1}^{\text{caché}(p)} \end{pmatrix} = \begin{pmatrix} g'(\sigma_0) z_{00} & \dots & g'(\sigma_0) z_{0k} & \dots & g'(\sigma_0) z_{0g} \\ g'(\sigma_j) z_{j0} & \dots & g'(\sigma_j) z_{jk} & \dots & g'(\sigma_j) z_{jg} \\ g'(\sigma_{N_c-1}) z_{N_c-1/0} & \dots & g'(\sigma_{N_c-1}) z_{N_c-1/k} & \dots & g'(\sigma_{N_c-1}) z_{N_c-1/g} \end{pmatrix} \begin{pmatrix} \delta_0^{\text{sortie}(p)} \\ \vdots \\ \delta_k^{\text{sortie}(p)} \\ \vdots \\ \delta_g^{\text{sortie}(p)} \end{pmatrix}$$

$N_c - 1$: nb neurones cachée

$g = N_c - 1$: nb neurones de sortie

rq : si g = logistique, $g'(\sigma_j) = g(U_j) \times [1 - g(U_j)]$

g_s = logistique, $g_s'(\tau_k) = g_s(\hat{y}_k) \times [1 - g_s(\hat{y}_k)]$

Méthode:

1) calculer toutes les prédictions: (phase forward) 1 échantillon = $(\vec{x}^{(p)}, \vec{y}^{(p)})$

$p = 0, \dots, P-1$ (pour tous les échantillons du jeu de données), faire:

a) $X_i^{(p)}$ = données de la couche d'entrée, $i = 1, \dots N_{entrée}-1$ et $X_0^{(p)} = 1{,}0$ (neurone de biais)

b) calculer $\sigma_j^{(p)} = \sum_{i=0}^{N_e-1} w_{ij} X_i^{(p)}$ et $U_j^{(p)} = g(\sigma_j^{(p)})$ ← f. activation cachée

$j = 1, \dots N_c - 1$

$\sigma_0^{(p)} = 1.0$ (neurone biais)

c) calculer $\tau_k^{(p)} = \sum_{j=0}^{N_c-1} z_{jk} U_j^{(p)}$ et $\hat{y}_k^{(p)} = g_s(\tau_k^{(p)})$ ← f. activation sortie

d) Conserver les valeurs de: $X_i^{(p)}, U_j^{(p)}, \hat{y}_k^{(p)}$

2) Calculer les erreurs pour $p = 0, \dots, P-1$ (i.e. chaque échantillon).

a) en couche de sortie: $\delta_k^{(p)\,sortie} = 2 g_s'(\tau_k)\left(\hat{y}_k^{(p)} - y_k^{(p)}\right) = 2 g_s(\hat{y}_k^{(p)}) \times \left[1 - g_s(\hat{y}_k^{(p)})\right] \times \left[\hat{y}_k^{(p)} - y_k^{(p)}\right]$

si f logistique

(prédiction) (vraie valeur voir (a))

$k = 0, \dots, N_{sortie}-1$

pour chaque $p = 0, \dots, P-1$.

b) en couche cachée: $\delta_j^{(p)\,cachée} = g'(\sigma_j) \times \sum_{k=0}^{N_s-1} z_{jk}\, \delta_k^{(p)\,sortie} = g(U_j) \times [1 - g(U_j)] \sum_{k=0}^{N_s-1} z_{jk}\, \delta_k^{(p)\,sortie}$ ← si f logistique

$j = 1, \dots, N_{caché}-1$

↳ on ne prend pas $j=0$ car le neurone caché $j=0$ est celui de biais donc pas d'erreur.

3) calculer les gradients et les corrections:

a) pour $p = 0, \dots P-1$, calculer: $\frac{\partial E^{(p)}}{\partial z_{jk}} = \delta_k^{(p)\,sortie} \cdot U_j$

et $\frac{\partial E^{(p)}}{\partial w_{ij}} = X_i \times \delta_j^{caché\,(p)}$ ← voir page d)

b) d'où les gradients totaux moyennés: $\frac{\partial E}{\partial z_{jk}} = \frac{1}{P} \sum_{p=0}^{P-1} \frac{\partial E^{(p)}}{\partial z_{jk}} = \frac{1}{P} \sum_{p=0}^{P-1} \delta_k^{(p)\,sortie} \times U_j$

$\frac{\partial E}{\partial w_{ij}} = \frac{1}{P} \sum_{p=0}^{P-1} \frac{\partial E^{(p)}}{\partial w_{ij}} = \frac{1}{P} \sum_{p=0}^{P-1} \delta_j^{(p)\,caché} \times X_i$

c) d'où les corrections: $\Delta z_{jk} = -\alpha\, \partial E / \partial z_{jk}$ et $\Delta w_{ij} = -\alpha\, \partial E / \partial w_{ij}$, à ajouter à w_{ij} et z_{jk}.

On a : $\dfrac{\partial E^{(p)}}{\partial w_{ij}} = 2 \sum_k \left(\hat{y}_k^{(p)} - y_k^{(p)} \right) \dfrac{\partial y_k^{(p)}}{\partial w_{ij}}$

$$\frac{\partial y_k}{\partial w_{ij}} = \frac{\partial g_s(\tau_k)}{\partial w_{ij}} = g_s'(\tau_k) \frac{\partial \tau_k}{\partial w_{ij}}$$

$$\frac{\partial \tau_k}{\partial w_{ij}} = \partial \left[\sum_j z_{jk} U_j \right] / \partial w_{ij}$$

$$= \sum_j z_{jk} \frac{\partial U_j}{\partial w_{ij}} = \sum_j z_{jk} \frac{\partial g(\sigma_j)}{\partial w_{ij}}$$

$$= \sum_j z_{jk}\, g'(\sigma_j) \frac{\partial \sigma_j}{\partial w_{ij}}$$

$$\hookrightarrow \partial \left(\sum_i w_{ij} X_i \right) / \partial w_{ij}$$

$$= z_{jk}\, g'(\sigma_j)\, X_i$$

$$\Rightarrow \frac{\partial E^{(p)}}{\partial w_{ij}} = 2 \sum_k \left(\hat{y}_k^{(p)} - y_k^{(p)} \right) g_s'(\tau_k)\, z_{jk}\, g'(\sigma_j)\, X_i$$

$$= 2 X_i \sum_k g'(\sigma_j)\, z_{jk} \times \left(\hat{y}_k^{(p)} - y_k^{(p)} \right) g_s'(\tau_k)$$

$$= \cancel{2}\, X_i \cdot \delta_j^{\text{caché}(p)}$$

↓ n° couche –

Couche sortie $\delta_d^m = (\hat{y}_d - y_d^m)\, g_o'(a_d^m)$ $\qquad \hat{y} = g_o(a_1^m)$

indice du cas dans l'échantillon (N elt au total)

Couche cachée k. $\delta_j^k = o_j^k (1 - o_j^k) \sum_{\ell=1}^{r^{k+1}} w_{j\ell}^{k+1} \delta_\ell^{k+1}$

Si 1 couche $\delta_j^{m-1} = \underbrace{o_j^{m-1} (1 - o_j^{m-1})}_{g'(a_j^{m-1})} \sum_{\ell=1}^{r^m} w_{j\ell} \delta_\ell^m$

nb nodes en couche (k+1)

$$\delta_j^k = \frac{\partial E}{\partial a_j^k} = \sum_{\ell=1}^{r^{k+1}} \frac{\partial E}{\partial a_\ell^{k+1}} \cdot \frac{\partial a_\ell^{k+1}}{\partial a_j^k}$$

($\ell \neq 0$ car node biais n'interviennent pas)

$$= \sum_{\ell=1}^{r^{k+1}} \delta_\ell^{k+1} \frac{\partial a_\ell^{k+1}}{\partial a_j^k}$$

mais $a_\ell^{k+1} = \sum_{j=1}^{r^k} w_{j\ell}^{k+1} g(a_j^k) \Rightarrow \frac{\partial a_\ell^{k+1}}{\partial a_j^k} = w_{j\ell}^{k+1} g'(a_j^k)$

w_{ij}^k = poids du node j de la couche k, dendrites i

o_i^k = output du node i de la couche k $\Rightarrow o_i^k = g(a_i^k)$

a_i^k = Σ pondérée $w_{ji}^k g(a_j^{k+1})$

1) forward: $\forall d=1,\dots,N$ $(\vec{x}_d, y_d) \mapsto \hat{y}_d,\ a_j^k,\ O_j^k = g(a_j^k),\ j=0,\dots,m$

indice de couche — indice de couche

indice de node — input | out — hidden

2) backward $\forall d=1,\dots,N$:

faire calculer $\dfrac{\partial E_d}{\partial w_{ij}^k}$ pour chaque poids w_{ij}^k ← couche $= m, m-1, \dots, 0$ dans cet ordre.

dendrite i — node j de couche k

$\delta_j^k O_i^k$

pour cela: a) calcul erreur couche sortie: $\delta_1^m = g'(a_1^m)(\hat{y}_d - y_d)$

couche sortie — 1 noeud en sortie

b) calcul erreur couche cachée $k = m-1, m-2, \dots$ première couche cachée

$$\delta_j^k = g'(a_j^k) \sum_{l=1}^{r^{k+1}} w_{jl}^{k+1}\, \delta_l^{k+1}$$

couche — noeud — r^{k+1} - nb nodes couche $k+1$

$l \geq 0$ car on ne prend pas node biais ($l=0$)

c) $\dfrac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k\, O_i^{k-1}$

3) Combiner les gradients individuels:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^{N} \frac{\partial}{\partial w_{ij}^k}\left(\frac{1}{2}(\hat{y}_d - y_d)^2\right) = \frac{1}{N}\sum_{d=1}^{N} \frac{\partial E_d}{\partial w_{ij}^k}$$

alors $\Delta w_{ij}^k = -\alpha \dfrac{\partial E}{\partial w_{ij}^k}$.

$\boxed{1}\ O^0$ $\qquad\qquad$ $\boxed{1}\ O^0$

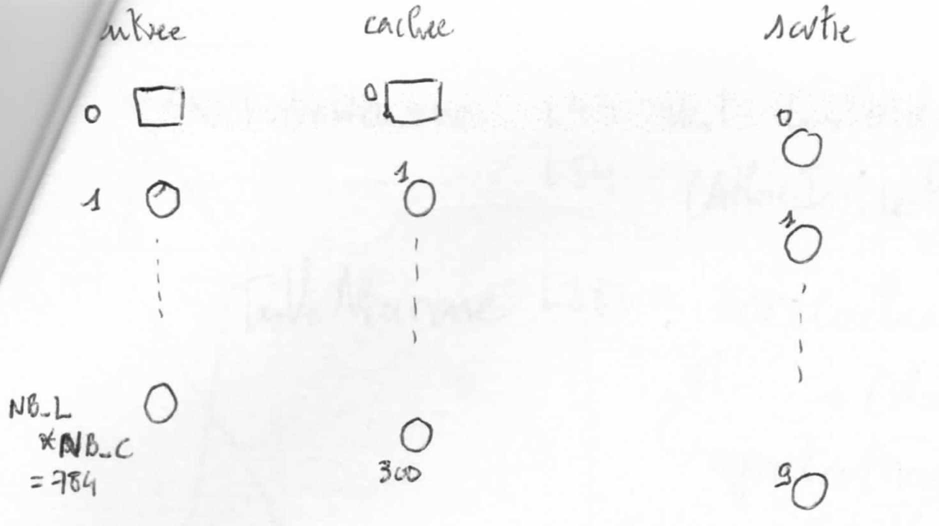
O^1 $\qquad\qquad$ O^1

O^2 $\qquad\qquad$ O^2

$O^i\ (\delta_i^k, a_i^k, O_i^k) \xrightarrow{w_{ij}^{k+1}} O^j\ (\delta_j^{k+1}, a_j^{k+1}, O_j^{k+1})$

O^{r_k} $\qquad\qquad$ $O^{r_{k+1}}$

niv k $\qquad\qquad$ niv $(k+1)$



NB_NEURONES_C_ENTREE = 28×28+1 = 785

NB_C_CACHEE = 301

NB_SORTIE = 10

Pour l'échantillon k^{smp} (= 0, … 60000-1)

Entrée : (TabDonnéeMNIST [smp / TBLOC][smp % TBLOC]. Image [ligne] [col] . /255) 1 à 784

TabDon… [——][——]

1.0 ↦ 0

Neurone entrée :

0 → 1.0

1 … NB_COL+1 … : (ligne 1)/255.0 (ligne 2)/255.0 …

NB_L * NB_COL : (ligne NB_LIGNE)/255.0

TabXCachée $\equiv \vec{x}_{smp}$.

Cachée : LePMC . TabOutputCoucheCachee [-][-][0] = 1.0

" " [-][ ][j] = $O_j^{cachée}$ $j = 1, \ldots,$ LePMC . CoucheCachee . NbNeurone − 1 = 300

Sortie : LePMC . TabOutputCoucheSortie [-][-][j] = O_j^{sortie} $j = 0, \ldots$ LePMC . ~ − 1 = 9

En sortie : LePMC → TabErreurSortie [-][-][j] = g'(TabOutputCSortie[ ][ ][j]) × $\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$ $j = 0, \ldots, 9$

- TestNeurone. L43 init LesPoids[1/2][i][j] = 0,5 et non 0.00005
  L54 (AllocInitLesPoids)

- TestNeurone L283 TabXCachee[j * NB_COLONNES + k] = (double) ~ Image[j][k]
  ajouter neurone de biais couche entrée
  - TabXCachee a 1 case de plus
  - T~~~ [0] = 1.0
  - et T~~~ [1 + j * NB_COLONNE + k] = ~ Image[j][k] / 255
  normalisation à [0,1]
  L520/L333

- TestNeurone L288 et 295 LePMC. TabOutputCoucheCachee
  ajouter un neurone de biais = 1.0 en case [0] et tout décalé d'1
  j de 1 à 300 [j] -> [1+j]

- TestNeurone. ApprentissageMNIST_PMC L324-
  revoir tous les facteurs correctifs
  (379? 386? 410? 416? 489??

- MNIST_Test.h <- voir les constantes

(1) neurone de biais
(2) normaliser 0-255 -> 0.0/1.0
(3) facteurs correctifs gradient / retro propag.

Traitement avec la fonction de coût logarithmique et la fonction d'activation Softmax en sortie.

La fonction softmax est adaptée comme fonction d'activation de la couche de sortie car elle permet de normaliser les productions $\tau_0, \dots, \tau_9$ des neurones de sortie afin d'obtenir des probabilités $\hat{y}_0, \hat{y}_1, \dots \hat{y}_9$ dont la somme vaut 1.

On a:

$$\hat{y}_k = \frac{e^{\tau_k}}{\sum_{i=0}^{9} e^{\tau_i}} = \hat{y}_k(\tau_0, \dots \tau_9)$$

(vq: on n'a plus seul $\hat{y}_k = g_s(\tau_k)$)

La fonction Softmax a ceci de particulier qu'elle prend en donnée d'entrée non seulement la production τ_k du neurone de sortie k, mais aussi celle de tous les autres neurones $(\tau_0, \dots, \tau_9)$. Elle diffère en cela notablement des autres fonctions d'activation (par exemple la fonction sigmoïde), qui ne prennent en donnée d'entrée, lorsqu'elles sont appliquées à un neurone k, que la seule valeur τ_k produite par le neurone de sortie k

La fonction de coût logarithmique est mathématiquement construite pour pénaliser les erreurs de prédictions, en s'appuyant sur le comportement de la fonction Log. (voir livres pour justifications avec les mains).

On a:

$$C = -\sum_{p=0}^{P-1} \sum_{k=0}^{9} \underbrace{y_k^{(p)}}_{\text{vraie valeur}} \text{Log}(\underbrace{\hat{y}_k^{(p)}}_{\text{prédiction pour l'échantillon } p \text{ d'être le chiffre } k}) = \sum_{p=0}^{P-1} C^{(p)}$$

$y_k^{(p)} = 1$ si k = vrai chiffre
$\phantom{y_k^{(p)}} = 0$ si $k \neq$ vrai chiffre

avec $C^{(p)}$ = coût logarithmique de l'échantillon p = $-\sum_{k=0}^{9} y_k^{(p)} \text{Log}(\hat{y}_k^{(p)})$

On note que si le chiffre vrai est k (alors $y_k^{(p)} = 1$ et $y_{j \neq k}^{(p)} = 0$) et si la prédiction est bonne (alors $\hat{y}_k^{(p)} \to 1$), le seul terme non nul $(-y_k^{(p)} \text{Log}(\hat{y}_k^{(p)})) \to 0$ car $\text{Log}\, x \to 0$ qd $x \to 1$)

En revanche, si la prédiction est mauvaise (alors $\hat{y}_k^{(p)} \to 0$ et $\hat{y}_j^{(p)} = 1$ pour une valeur de $j \neq k$), alors le seul terme non nul dans $C^{(p)}$, qui est $(-y_k^{(p)} \text{Log}\, \hat{y}_k^{(p)}) \to +\infty$ (car $\text{Log}\, x \to -\infty$ qd $x \to 0$)

Dérivées partielles de la fonction de coût (pour un échantillon):

$$\frac{\partial C^{(p)}}{\partial x} = \sum_{k=0}^{9} -\left(y_k^{(p)} / \hat{y}_k^{(p)}\right) \times \frac{\partial \hat{y}_k^{(p)}}{\partial x} .$$

(On omettra l'indice (p) dans la suite des calculs)

$$\hat{y}_k = \text{Softmax}(\tau_0, \tau_k, \tau_9) = e^{\tau_k} / \left(\sum_{\ell=0}^{9} e^{\tau_\ell}\right)$$

On calcule les dérivées partielles de $\hat{y}_k$ par rapport à τ_q, $q=0,\dots,9$.

- si $q=k$, τ_k apparaît 2 fois dans softmax (au numérateur et au dénominateur).

$$\frac{\partial \hat{y}_k}{\partial \tau_k} = \left(e^{\tau_k} \times \left(\sum_\ell e^{\tau_\ell}\right) - e^{\tau_k} \times e^{\tau_k}\right) / \left(\sum_\ell e^{\tau_\ell}\right)^2$$

$$= \left(e^{\tau_k} \times \left(e^{\tau_k} + \sum_{\ell \neq k} e^{\tau_\ell}\right) - e^{\tau_k} \times e^{\tau_k}\right) / \left(\sum_\ell e^{\tau_\ell}\right)^2$$

$$= e^{\tau_k} \times \left(\sum_{\ell \neq k} e^{\tau_\ell}\right) / \left(\sum_\ell e^{\tau_\ell}\right)^2$$

$$= \frac{e^{\tau_k}}{\left(\sum_\ell e^{\tau_\ell}\right)} \times \frac{\left(\sum_\ell e^{\tau_\ell} - e^{\tau_k}\right)}{\sum_\ell e^{\tau_\ell}} = \hat{y}_k \times (1 - \hat{y}_k)$$

- si $q \neq k$, τ_q apparaît une seule fois dans softmax (au dénominateur)

$$\frac{\partial \hat{y}_k}{\partial \tau_q} = \frac{e^{\tau_k} \times (-e^{\tau_q})}{\left(\sum_\ell e^{\tau_\ell}\right)^2} = \frac{e^{\tau_k}}{\left(\sum_\ell e^{\tau_\ell}\right)} \times \frac{(-e^{\tau_q})}{\left(\sum_\ell e^{\tau_\ell}\right)} = -\hat{y}_k \times \hat{y}_q .$$

Ceci permet d'obtenir les dérivées partielles suivantes:

$$\frac{\partial C^{(p)}}{\partial \tau_q} = \sum_{k=0}^{9} -\left(y_k / \hat{y}_k\right) \frac{\partial \hat{y}_k}{\partial \tau_q} = \sum_{k \neq q} -\frac{y_k}{\cancel{\hat{y}_k}} \times \left(-\cancel{\hat{y}_k} \cdot \hat{y}_q\right) - \frac{y_q}{\cancel{\hat{y}_q}} \times \cancel{\hat{y}_q} (1 - \hat{y}_q)$$

$$= \hat{y}_q \sum_{k \neq q} y_k - y_q (1 - \hat{y}_q) = \hat{y}_q \underbrace{\sum_{k=0}^{9} y_k}_{=1} - y_q = \hat{y}_q - y_q , \quad q = 0,\dots,9.$$

Donc $\boxed{\frac{\partial C^{(p)}}{\partial \tau_k} = \hat{y}_k - y_k}$, vecteur à 10 composantes

$$\boxed{\frac{\partial C^{(p)}}{\partial z_{jk}}} = \frac{\partial C^{(p)}}{\partial \tau_k} \times \frac{\partial \tau_k}{\partial z_{jk}}$$

avec: $\tau_k = \sum_j U_j \cdot z_{jk}$

$$\boxed{= (\hat{y}_k - y_k) \times U_j}$$

$$\boxed{\frac{\partial C^{(p)}}{\partial U_j}} = \sum_k \frac{\partial C^{(p)}}{\partial \tau_k} \times \underbrace{\frac{\partial \tau_k}{\partial U_j}}_{= z_{jk}} = \boxed{\sum_k (\hat{y}_k - y_k) \times z_{jk}}$$

Posons : $\frac{\partial C^{(p)}}{\partial \tau_k} = \overrightarrow{\left(\delta_k^{\text{sortie}(p)} \right)} = \begin{pmatrix} \hat{y}_0^{(p)} - y_0^{(p)} \\ \vdots \\ \hat{y}_9^{(p)} - y_9^{(p)} \end{pmatrix} = \begin{pmatrix} \delta_0^{\text{sortie}(p)} \\ \vdots \\ \delta_9^{\text{sortie}(p)} \end{pmatrix}$ = vecteur à N_s composantes ($k = 0, \ldots, 9$) ($9 = N_s - 1$).

Alors : $\frac{\partial C^{(p)}}{\partial \sigma_j} = \sum_{k=0}^{9} \frac{\partial C^{(p)}}{\partial \tau_k} \times \frac{\partial \tau_k}{\partial \sigma_j}$

mais $\tau_k = \sum_j U_j z_{jk} = \sum_j g(\sigma_j) z_{jk}$

g : fonction d'activation de la couche ~~gam~~ cachée, qui prend le seul σ_j pour calculer U_j.

donc : $\frac{\partial \tau_k}{\partial \sigma_j} = g'(\sigma_j) \times z_{jk}$.

$$= \sum_{k=0}^{9} \left(\hat{y}_k^{(p)} - y_k^{(p)} \right) \times g'(\sigma_j) \times z_{jk}$$

$$= \sum_{k=0}^{9} \left[g'(\sigma_j) . z_{jk} \right] \times \delta_k^{\text{sortie}(p)}$$

$$= \delta_j^{\text{caché}(p)}.$$

$$\vec{\delta}^{\,\text{caché}(p)} = \begin{pmatrix} \delta_0^{\text{caché}(p)} \\ \vdots \\ \delta_j^{\text{caché}(p)} \\ \vdots \\ \delta_{N_c-1}^{\text{caché}(p)} \end{pmatrix} = \begin{pmatrix} g'(\sigma_0) z_{00} & \cdots & g'(\sigma_0) z_{09} \\ g'(\sigma_j) z_{j0} \cdots & g'(\sigma_j) z_{jk} \cdots & g'(\sigma_j) z_{j9} \\ \vdots & & \\ \vdots & & g'(\sigma_{N_c-1}) z_{N_c-1\,9} \end{pmatrix} \begin{pmatrix} \delta_0^{\text{sortie}(p)} \\ \vdots \\ \delta_k^{\text{sortie}(p)} \\ \vdots \\ \delta_9^{\text{sortie}(p)} \end{pmatrix}$$

N_c = nb neurones cachés

10 neurones de sortie

~~Ex~~ Ex : si g = sigmoïde, $g'(\sigma_j) = g(U_j) \times \left[1 - g(U_j) \right]$.

---

rq : $\frac{\partial C^{(p)}}{\partial w_{ij}} = \frac{\partial C^{(p)}}{\partial U_j} \times \frac{\partial U_j}{\partial w_{ij}} = \sum_k (\hat{y}_k - y_k) z_{jk} \times g'(\sigma_j) \times \underbrace{\frac{\partial \sigma_j}{\partial w_{ij}}}_{= X_i}$

$$= X_i \times \sum_k \left[g'(\sigma_j) z_{jk} \times (\hat{y}_k - y_k) \right]$$

$$= X_i \times \delta_j^{\text{caché}(p)}$$

...éthode :

1) calculer ttes prédictions :

a) ↳ si $X_i^{(p)}$ = donnée du neurone d'entrée i pour l'échantillon p.
(rq : $X_0^{(p)} = 1$ car neurone de biais)

alors

b) ↳ $\sigma_j^{(p)} = \sum_i W_{ij} X_i^{(p)}$ et $U_j^{(p)} = g(\sigma_j^{(p)})$
↳ activation couche cachée

c) ↳ $\tau_k^{(p)} = \sum_j Z_{jk} U_j^{(p)}$ et $\hat{y}_k^{(p)} = g_s(\tau_0^{(p)}, \dots, \tau_9^{(p)})$
↳ softmax

d) ↳ conserver les valeurs de : $X_i^{(p)}$,

2) calculer toutes les erreurs :

a) couche sortie : $\delta_k^{sortie(p)} = \frac{\partial C^{(p)}}{\partial \tau_k} = \hat{y}_k^{(p)} - y_k^{(p)}$.

↑ probabilité produite par softmax

↳ vraie valeur = 1 si k = vrai chiffre
0 si $k \neq$ vrai chiffre

b) couche cachée : $\delta_j^{caché(p)} = g'(\sigma_j) \times \sum_k Z_{jk} \, \delta_k^{sortie(p)}$

↓

$= g(U_j) \times [1 - g(U_j)]$ si g = sigmoïde

3) calculer les gradients et les corrections

a) pour $p = 0, \dots P-1$ calculer : $\frac{\partial C^{(p)}}{\partial z_{jk}} = \delta_k^{sortie(p)} \times U_j$

et $\frac{\partial C^{(p)}}{\partial w_{ij}} = X_i \times \delta_j^{caché(p)}$

b) d'où les gradients cumulés, totaux :

$$\frac{\partial C}{\partial z_{jk}} = \sum_{p=0}^{P-1} \frac{\partial C^{(p)}}{\partial z_{jk}} = \sum_{p=0}^{P-1} \delta_k^{sortie(p)} \times U_j$$

$$\frac{\partial C}{\partial w_{ij}} = \sum_p \frac{\partial C^{(p)}}{\partial w_{ij}} = \sum_p \delta_j^{caché(p)} \times X_i$$

c) d'où les corrections :

$$\Delta z_{jk} = -\alpha \, \partial C / \partial z_{jk} \quad \text{et} \quad \Delta w_{ij} = -\alpha \, \partial C / \partial w_{ij}$$

à ajouter à w_{ij} et z_{jk}.

Finalement, changer E en C et la fonction d'activation de sortie en Softmax change peu de choses