Back propagat° algo & auto-encoders

"AI" = perfs tirées par 1 tech: DNN + algo apprentissage supervisé (deep learning)
+ cs reco patterns (image, texte, sons, ...)

avatars : → deep reinforc° learning (Alpha go)
→ generative models (GAN, generative adversarial netw.)

- sources - livre "deep-learning" Goodfellow, Bengio, Courville MIT Press en ligne
  - cours Stanford en ligne (Youtube)
  - cours Berkeley : Ali Ghodsi (Youtube)
  - Wikipedia
  - (, Lebesgue Days)

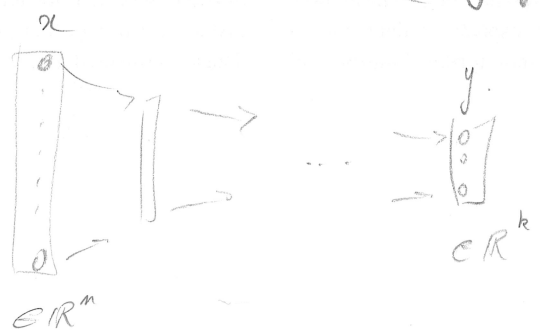
  ↪ pas toujours bien formalisé/mathématisé; bcp de "recettes" ... et de résultats impressionnants

- appren. supervisé :

  essayer d'approcher/d'apprendre $f : \mathbb{R}^n \longrightarrow \mathbb{R}^k$
  $x \longmapsto y = f(x)$

  à partir de (bcp) d'exemples $(x^i, y^i = f(x^i))$ $i = 1, \dots N$

NN :



- $\min_w \mathbb{E}\left(\|Y - f_w(X)\|^2\right)$ → erreur moyenne. où $(X,Y) \sim P_{X,Y}$ loi des données

  on a des tirages iid (x^i, y^i) $i = 1 \dots N$

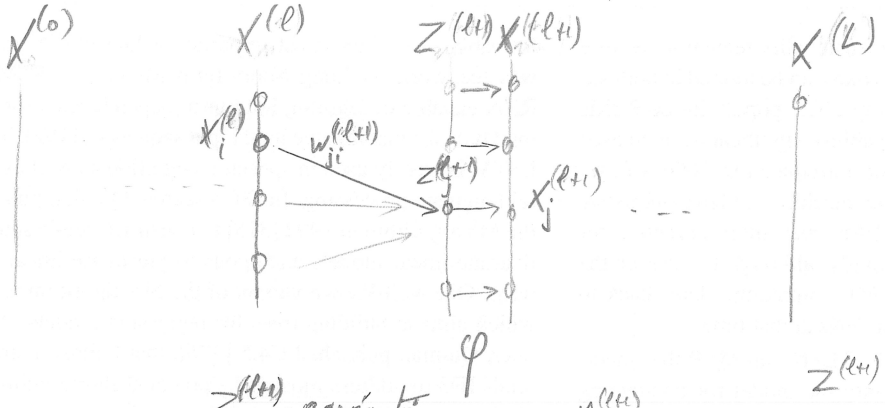
  Remplace $\mathbb{E}(\text{---})$ par $\frac{1}{N} \sum_i \|y^i - f_w(x^i)\|^2 = E(w)$ = erreur.

  Puis gradient descat: $\nabla_w E$ et $w(t+1) \leftarrow w(t) - \alpha \nabla_w E(w(t))$

- en fait: procéder par mini-batches, puis reshuffling.
  jusqu'à CV sur le dataset.

- on va voir calcul grad pour 1 donnée (x, y).

  on off



noter inversion w_{ji} va de i vers j d'où terme W^T dans les équations.

$Z^{(l+1)}$ = agrégation

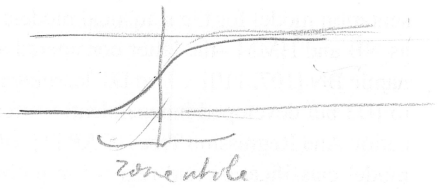
$$X^{(l+1)} = \varphi\left(W^{(l+1)T} \cdot X^{(l)}\right)$$

φ : ponctuel ; $W^{(l+1)T} \cdot X^{(l)}$: matriciel

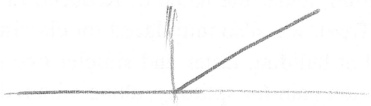
$$\begin{bmatrix} X^{(l+1)} \end{bmatrix} = \varphi\left(\overbrace{\begin{bmatrix} W^{(l+1)T} \end{bmatrix} \cdot \begin{bmatrix} X^{(l)} \end{bmatrix}}^{Z^{(l+1)}} \right)$$

φ = fonction d'activation (non linéaire, saille)

- sigmoïde (fct. logistique) $\varphi(x) = \dfrac{1}{1+e^{-x}}$, $\varphi'(x) = \varphi(x)(1-\varphi(x))$



- Relu $\varphi(x) = \max(0, x)$



- Soft. + $\varphi(x) = \log(1+e^x)$

Erreur sur 1 donnée pour l'approximation de f.

Soit $f_w : X^{(0)} \longmapsto X^{(L)}$

on a la donnée x et sortie désirée y

mesure écart de $f(x)$ à y : $E_w = \frac{1}{2} \| f_w(x) - y \|^2$ quadratique

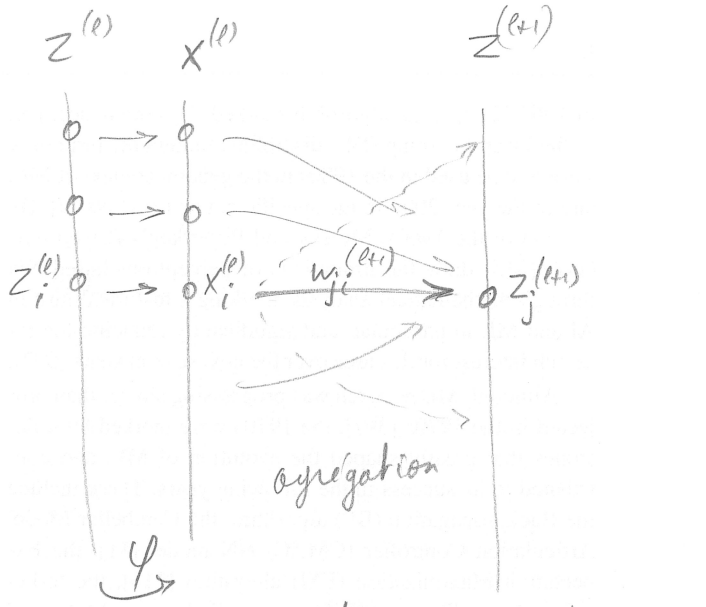
↳ ajuster les poids $w^{(n)}$ pour réduire l'erreur.

pb optimisat° numérique : $\min_w E_w$

résolu par descente de gradient :

$$W(t+1) = W(t) - \alpha \underbrace{\nabla_W E_w(t)}_{\text{à calculer}}$$

(Rétro) Propagation du gradient



- On souhaite calculer les valeurs pour tout $\ell = 0 \dots L$

$$\nabla_{z^{(\ell)}} E = \begin{bmatrix} \square \end{bmatrix} \frac{\partial E}{\partial z_j^{(\ell)}}$$

pour en tirer

$$\frac{\partial E}{\partial w_{ji}^{(\ell+1)}} = \frac{\partial E}{\partial z_j^{(\ell+1)}} \cdot \frac{\partial z_j^{(\ell+1)}}{\partial w_{ji}^{(\ell+1)}} = \frac{\partial E}{\partial z_j^{(\ell+1)}} \cdot x_i^{(\ell)}$$

note : w_{ji} va de i vers j (d'où W^T dans les éq°)

↳ matriciellement : $\frac{\partial E}{\partial W^{(\ell+1)}} = \nabla_{z^{(\ell+1)}} E \cdot [x^{(\ell)}]^T$

- récurrence arrière : E dépend de $z^{(\ell)}$ via $z^{(\ell+1)}$

$$\frac{\partial E}{\partial z_i^{(\ell)}} = \sum_j \frac{\partial E}{\partial z_j^{(\ell+1)}} \cdot \frac{\partial z_j^{(\ell+1)}}{\partial z_i^{(\ell)}} = \sum_j \frac{\partial E}{\partial z_j^{(\ell+1)}} \cdot \underbrace{\frac{\partial z_j^{(\ell+1)}}{\partial x_i^{(\ell)}}}_{w_{ji}^{(\ell+1)}} \cdot \underbrace{\frac{\partial x_i^{(\ell)}}{\partial z_i^{(\ell)}}}_{\varphi'(z_i^{(\ell)})}$$

$$= \begin{bmatrix} \square \end{bmatrix} W_{\cdot,i}^{(\ell+1)} \cdot \begin{bmatrix} \square \end{bmatrix} \nabla_{z^{(\ell+1)}} E \cdot \square \, \varphi'(z_i^{(\ell)})$$

⚠ vient atténuer le gradient si on prend la sigmoïde, hors zone utile.

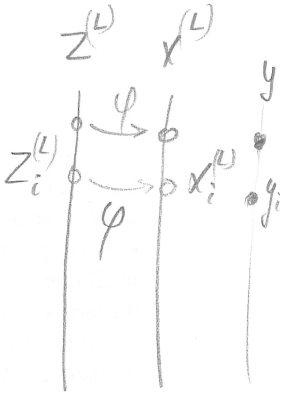
↳ matriciellement :

$$\nabla_{z^{(\ell)}} E = \left([W^{(\ell+1)}]^T \cdot \nabla_{z^{(\ell+1)}} E \right) .* \varphi'(z^{(\ell)})$$

$\square \cdot \begin{bmatrix} \square \end{bmatrix} \quad .* \begin{bmatrix} \square \end{bmatrix}$

↳ produit terme à terme. non linéarité

Passage de σ à RELU a permis de propager le gradient + profond ⇒ deep neural nets.



- Initialisation

$$E = \frac{1}{2}\|f_w(x) - y\|^2 = \frac{1}{2}\|x^{(L)}(x) - y\|^2$$

$$= \frac{1}{2}\sum_i \left(x_i^{(L)}(x) - y_i\right)^2$$

$$\frac{\partial E}{\partial z_i^{(L)}} = \frac{\partial E}{\partial x_i^{(L)}} \cdot \frac{\partial x_i^{(L)}}{\partial z_i^{(L)}} = \left(x_i^{(L)} - y_i\right) \cdot \varphi'\left(z_i^{(L)}\right)$$

matriciellement : $\nabla_{z^{(L)}} E = \left(x^{(L)} - y\right) \cdot * \varphi'\left(z^{(L)}\right)$

↳ produit terme à terme

Remarques

1) Algo : forward = calculer les $x^{(l)}$ $z^{(l)}$, backward = repropagation gradient en arrière.

2) on prend souvent $x_i^{(l+1)} = \varphi\left(\sum_j w_{ji}\, x_j^{(l)} + b_i^{(l+1)}\right)$

↳ biais, change seuil de réact° du neuron

$\Leftrightarrow$ rajouter ds $x^{(l)}$ un "neurone" toujours à 1

3) on ne procède pas exemple par exemple pour régler les poids, mais par "mini batches" : paires $(x(k), y(k))$ d'exemples, $k = 1 \ldots K$

On prend alors $E = \frac{1}{2m}\sum_k \|f_w(x(k)) - y(k)\|^2$

4) Le signal d'erreur incorpore souvent un terme de régularisation pour garder les poids petits.

$$E = \frac{1}{2}\|x^{(L)}(x) - y\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L} \|W^{(l)}\|^2$$

↳ rajoute 1 terme linéaire $\frac{\partial E}{\partial w^{(l+1)}} = \text{——} + \lambda \cdot W^{(l)}$

L'évolution des poids devient

$$W(t+1) = W(t) + \beta\Big[\underbrace{-\alpha \nabla_W E_W(t)}_{\text{gradient}} \underbrace{+\alpha\lambda W(t)}_{\text{régul.}}\Big] + \underbrace{(1-\beta)\,\Delta W(t)}_{\text{inertie dans la descente - évite de rester piégé ds min locaux}}$$

4) La profondeur de propag. repose sur RELU
↳ ~ pb linéaire + seuils d'activation

- Bcp de réglages "à la main", expérimentaux
- On connaît des struct. de rsx qui marchent bien pour la reco de patterns en image (rsx conv.) et en parole (short-term long term)
- Les 1ères couches sont "standard", multi-optimal°, et pré-entraînables → on a la plus qu'à entraîner les couches sups pr 1 appli donnée.

ex image = convolutions locales + décimation.
(conv. locale, invariance translat°, rotat°)

- Les dern. couches sont totalement connectées.
↳ elles portent une descript° compacte des images en "features".

- La mise en corresp des features de haut niveau permet d'associer texte/son/image dans un même domaine de représentation → interrogat° d'images.