

Every Model Learned by Gradient Descent Is Approximately a Kernel Machine

Pedro Domingos, 2020

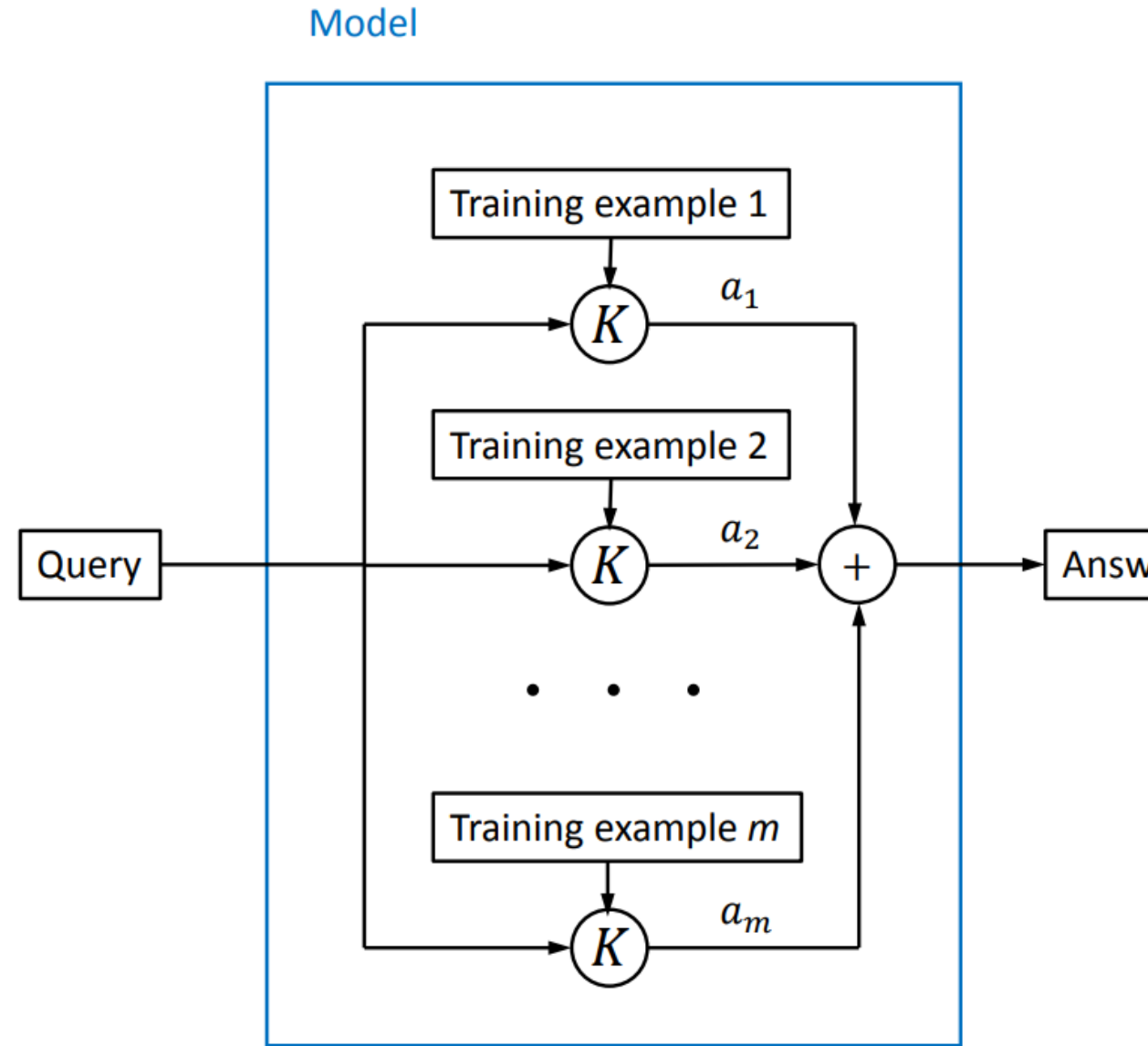
Aperçu

- Les réseaux profonds découvrent de nouvelles représentations
- Claim : ce sont justes des kernel machines (machines à noyaux) avec un noyau particulier
- Vrai pour tous les modèles appris par descente de gradient
- Les poids du réseau sont une superposition des exemples d'entraînement
- L'architecture incorpore de la connaissance

Kernel Machines

$$y = g\left(\sum_i a_i K(x, x_i) + b\right)$$

- y Sortie du modèle
- $g(\cdot)$ Fonction non linéaire
- a_i, b Paramètres appris
- $K(\dots, \dots)$ Noyau
- x Donnée à tester
- x_i Données d'entraînement

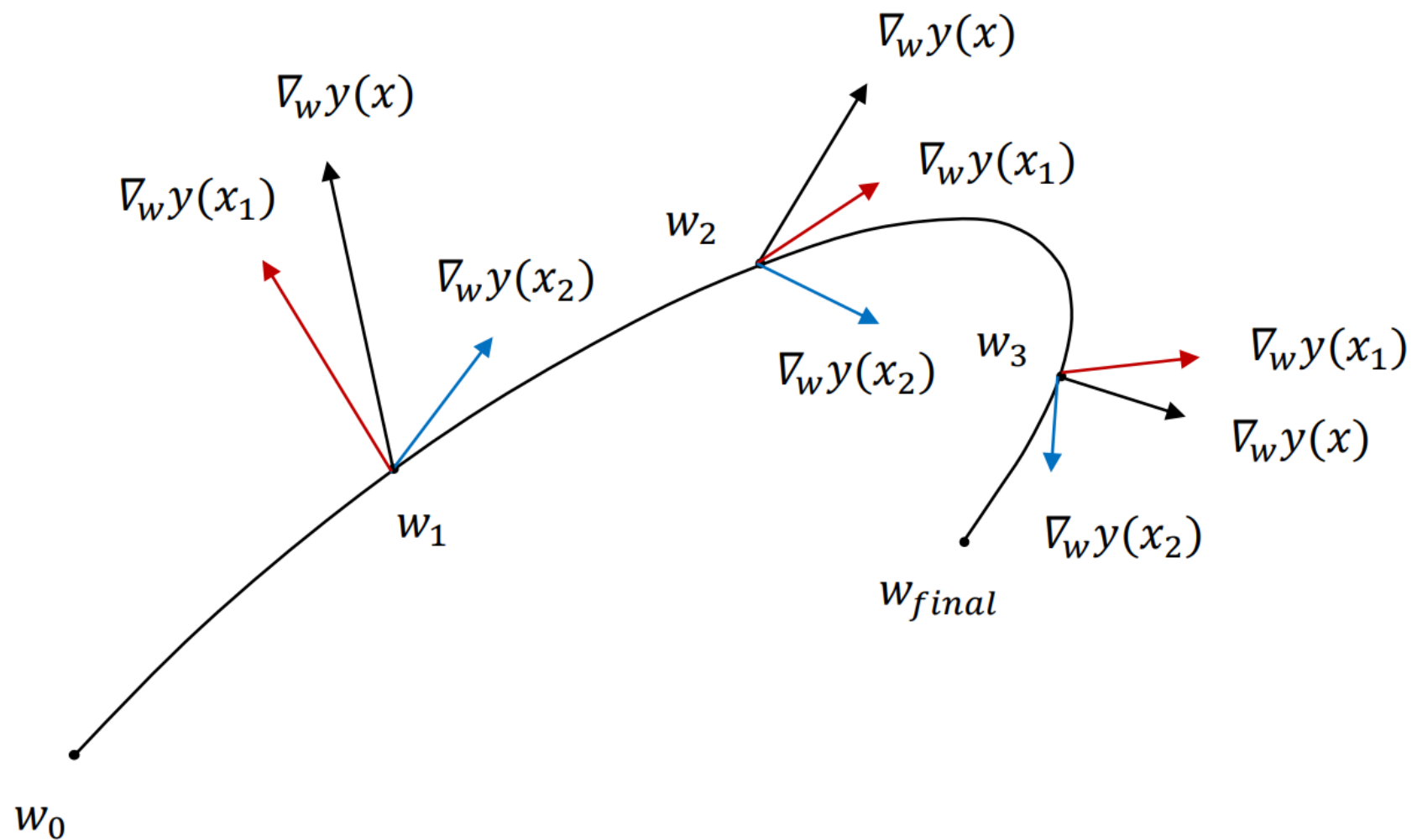


Path Kernels

$$K^{PK}(x, x') = \int_{c(t)} \nabla_w y(x) \cdot \nabla_w y(x') dt$$
$$K^{NTK}(x, x') = \nabla_w y(x) \cdot \nabla_w y(x')$$

- | | |
|---------------------|---|
| • $K(\dots, \dots)$ | Path Kernel |
| • x, x' | Données |
| • t | Temps |
| • w | Les poids |
| • $c(t)$ | Chemin pris par w pendant la descente de gradient |
| • $y(\cdot)$ | Modèle |

Path Kernel



Théorème

Soit le modèle $y = f_w(x)$, avec f une fonction différentiable par w , appris avec le training set $\{(x_i, y_i^*)\}_{i=1}^m$ par descente de gradient avec une fonction de coût différentiable $L = \sum_i L(y_i^*, y_i)$ et le learning rate ϵ . Alors :

$$\lim_{\epsilon \rightarrow 0} y = \sum_{i=1}^m a_i K(x, x_i) + b$$

avec $K^{PK}(x, x_i)$ le path kernel associé à $f_w(x)$ et au chemin pris par les paramètres pendant la descente de gradient; a_i la moyenne des $\frac{\partial L}{\partial y_i}$ le long du chemin pondéré par le Tangent Kernel correspondant, et b le modèle initial.

Preuve 1/3

- Descente de gradient :

$$w_{s+1} = w_s - \epsilon \nabla_w L(w_s)$$

- Gradient Flow :

$$\frac{dw(t)}{dt} = -\nabla_w L(w(t))$$

- Chain rule :

$$\frac{dy}{dt} = \sum_{j=1}^d \frac{\partial y}{\partial w_j} \frac{\partial w_j}{\partial t}$$

- On remplace :

$$\frac{dy}{dt} = \sum_{j=1}^d \frac{\partial y}{\partial w_j} \left(-\frac{\partial L}{\partial w_j} \right)$$

Preuve 2/3

- Additivité + chain rule :

$$\frac{dy}{dt} = \sum_{j=1}^d \frac{\partial y}{\partial w_j} \left(- \sum_{i=1}^m \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial w_j} \right)$$

- Réarrangement :

$$\frac{dy}{dt} = - \sum_{i=1}^m \frac{\partial L}{\partial y_i} \sum_{j=1}^d \frac{\partial y}{\partial w_j} \frac{\partial y_i}{\partial w_j}$$

- NTK definition :

$$\frac{dy}{dt} = - \sum_{i=1}^m L' K^{NTK}(x, x_i)$$

- Intégration :

$$\lim_{\epsilon \rightarrow 0} y = y_0 - \int_{c(t)} \sum_{i=1}^m L' K^{NTK}(x, x_i)$$

Preuve 3/3

- Réarrangement :

$$\lim_{\epsilon \rightarrow 0} y = y_0 - \sum_{i=1}^m L' \int_{c(t)} K^{NTK}(x, x_i)$$

- Définition Path Kernel :

$$\lim_{\epsilon \rightarrow 0} y = y_0 - \sum_{i=1}^m L' K^{PK}(x, x_i)$$

- Finalement :

$$\lim_{\epsilon \rightarrow 0} y = \sum_{i=1}^m a_i K^{PK}(x, x_i) + b$$

Remarques

- a_i et b dépendent de x
- Suppose un learning rate suffisamment petit
- Ajouter un terme régularisation change juste le b
- Preuve pour Batch Gradient Descent