Fondamentaux du Deep learning

Louis Thiry (INRIA Rennes)

louis.thiry@outlook.fr

Présentation

Louis Thiry

- Formation en maths appliquées a l'École Polytechnique: analyse numérique, traitement du signal et apprentissage.
- Avec Google Deep Mind: apprentissage d'une fonctionnelle pour la mécanique quantique numérique.
- Doctorat à l'ENS Paris, directeur S. Mallat
  - Modélisation mathématiques des réseaux de neurones profonds pour la classification d'images
  - Régression de l'énergie de systèmes d'atomes (molécules, solides...)
  - dispositifs de création artistes-algorithmes
- Actuellement: Postdoc ERC STUOD à l'INRIA Rennes, analyse de données satellites sur l'océan, modèles et assimilation de données pour les océans.

Vérification des installations

Au terminal:

- python –version
- jupyter notebook

En python:

- import torch
- import numpy as np
- import pandas
- import sklearn

Donnees TP: https://www.di.ens.fr/louis.thiry/

Machine Learning with Deep neural networks

- *Machine Learning*: Apprentissage automatique
- *Neural networks*: Réseaux de neurones artificiels
- *Deep*: Profond ?

Machine learning

Machine Learning

- "*Les algorithmes de Machine Learning apprennent de manière autonome à effectuer une tâche ou à réaliser des prédictions à partir de données et améliorent leurs performances au fil du temps. Une fois entraîné, l'algorithme pourra retrouver les patterns dans de nouvelles données.*" Datascientest.com

- " Champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d' "apprendre" à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. " Wikipedia.com

$\rightarrow$ Définitions imprécises et souvent anthropomorphiques.

Approximer une fonction F a partir de données.

- Une fonction F *cible* avec un ensemble de données (x^i, y^i)

$F : x =$ $\mapsto$ "cat"

- Un ensemble de fonctions paramétriques $\mathcal{F} = \{f_\theta, \theta \in \mathbb{R}^p\}$.

- Un algorithme pour trouver un bon paramètre $\theta^\star$.

  Ex: moindres carres.

Machine Learning

Notion abordées:

- La fonction F: modélisation physique / psychologique...
- Les données (x^i, y^i): probabilités, ingénierie, sociologie, bon sens.
- Un ensemble de fonctions $\mathcal{F}$: espaces de fonctions, théorie de l'approximation.
- Trouver $\theta^\star$: modélisation, optimisation.

La Régression linéaire aux moindres carres (RLMC)

Fonction F, données $\left(x^i, y^i = F(x^i)\right) \in \mathbb{R}^D \times \mathbb{R}$.

- **Important**: il faut que les données x^i viennent d'une même distribution de probabilité.
- Ensemble $\mathcal{F}$ = espace des fonction affines:

$$y = \sum_{d=1}^{D} a_d x_d + b = \langle a, x \rangle_{\mathbb{R}^D} + b$$

- Minimiser la somme des carres de l'erreur:

$$a, b = \underset{\alpha,\beta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \left(y^n - (\langle \alpha, x^n \rangle + \beta) \right)^2$$

- Condition d'optimalité ?

RLMC: condition d'optimalité

$\mathbb{R}^{D/N}$ avec les normes usuelles, on note

$$y = \begin{pmatrix} y^1 \\ \vdots \\ y^N \end{pmatrix}, X = \begin{pmatrix} 1 & x_1^1 & \cdots & x_D^1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^N & \cdots & x_D^N \end{pmatrix}, a = \begin{pmatrix} b \\ a_1 \\ \vdots \\ a_D \end{pmatrix},$$

Écrire en vectoriel:

- La relation affine entre y^n et x^n

$$y^n = \sum_{d=1}^{D} a_d x_d^n + b = \langle a, x^n \rangle_{\mathbb{R}^D} + b$$

- L'erreur quadratique

$$\frac{1}{N} \sum_{n=1}^{N} \Big(y^n - (\langle a, x^n \rangle + b) \Big)^2 \quad ?$$

RLMC: condition d'optimalité

$$y = \begin{pmatrix} y^1 \\ \vdots \\ y^N \end{pmatrix}, X = \begin{pmatrix} 1 & x_1^1 & \cdots & x_D^1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^N & \cdots & x_D^N \end{pmatrix}, a = \begin{pmatrix} b \\ a_1 \\ \vdots \\ a_D \end{pmatrix},$$

- La relation affine entre y^n et x^n

$$y = Xa$$

- L'erreur quadratique

$$\frac{1}{N}\left\|y - Xa\right\|_{2,\mathbb{R}^N}^2 = \frac{1}{N}\left\langle y - Xa, y - Xa\right\rangle_{\mathbb{R}^N}$$

Gradient d'une fonction réelle dans un système de coordonnées

Soit $f : \mathbb{R}^D \mapsto \mathbb{R}$ dérivable.
Dans un système de coordonnées cartésiennes, le gradient ∇f d'une fonction f

- est le vecteur des dérivées partielles f par rapport aux coordonnées:

$$\nabla_{(x_1,\ldots,x_D)} f = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x_1, \ldots, x_D) \\ \vdots \\ \frac{\partial f}{\partial x_D}(x_1, \ldots, x_D) \end{bmatrix}$$

- pointe dans la direction où la fonction croît le plus, sa norme correspond au taux de croissance dans cette direction.
- généralise à $\mathbb{R}^D$ la dérivée d'une fonction d'une variable.

Gradient d'une fonction réelle

Soit $f : \mathbb{R}^D \mapsto \mathbb{R}$ dérivable.

Définition générale:

Pour une variation h autour de x, la variation de f est au premier ordre une fonction linéaire de h:

$$f(x+h) = f(x) + L(h) + \underset{\|h\| \to 0}{o(h)}$$

Le gradient est le représentant de cette forme linéaire tangente:

$$f(x+h) = f(x) + \nabla f \cdot h + \underset{\|h\| \to 0}{o(h)}$$

Gradient d'une fonction

Soit $f : \mathbb{R}^N \mapsto \mathbb{R}$ dérivable.

Propriété:

Si f admet un minimum en m, **alors** son gradient s'annule en m:

$$\nabla_m f = 0_{\mathbb{R}^N}$$

On a aussi

$$f(m+h) = f(m) + \underset{\|h\| \to 0}{o(h)}$$

RLMC: condition d'optimalité

- Calculer le gradient de

$$f(a) = \frac{1}{N}\|y - Xa\|_{2,\mathbb{R}^{\mathbb{N}}}^2 = \frac{1}{N}\langle y - Xa, y - Xa\rangle_{\mathbb{R}^{\mathbb{N}}}$$

- Que donne la condition d'annulation du gradient ?

RLMC: condition d'optimalité

- Gradient de f:

$$\nabla f = \frac{2}{N}\left(X^T X a - X^T y\right)$$

- Condition d'annulation du gradient":

$$a = \left(X^T X\right)^{-1} X^T y$$

Mise en pratique: Notebook python sur les régressions linéaires.

Régression linéaire régularisée

Supposons que l'on fasse une régressions affine avec deux données et deux variables explicatives:

$$
\begin{aligned}
a_1 x_1^1 + a_2 x_2^1 + b &= y^1 \\
a_1 x_1^2 + a_2 x_2^2 + b &= y^2 \\
a_1 x_1^3 + a_2 x_2^3 + b &= y^3
\end{aligned}
$$

- Que peut-on dire de ce système ?
- Que vaudra l'erreur quadratique ?
- Cela vous parait raisonnable ?
- Qu'est ce qui pourrait résoudre ce problème ?

RLMC: Régression linéaire régularisée

Problème: on a suppose que $X^T X$ est inversible, mais si elle ne l'est pas... En particulier si N ¡ D, rg$(X^T X) = N$, la matrice n'est pas inversible.

On pose

$$f(a) = \left\| y - Xa \right\|^2 + \epsilon \|a\|^2$$

- Gradient de f ?
- Condition d'annulation du gradient" ?

Régression linéaire régularisée

On pose

$$f(a) = \left\|y - Xa\right\|^2 + \epsilon\|a\|^2$$

- Gradient de f ?

$$\nabla f = 2\left(X^T X a - X^T y\right) + 2\epsilon a$$

- Condition d'annulation du gradient ?

$$a = \left(X^T X + \epsilon I_N\right)^{-1} X^T y$$

La matrice $X^T X + \epsilon I_N$ est toujours inversible.

Enseignements

En pratique:

- Attention au sur/sous-apprentissage
- Possibilité de régulariser l'apprentissage
- Hyper-paramètres: validation croisée.
- Trouver la bonne représentation: feature engineering.

Avis d'experts:

- *"Coming up with features is difficult, time-consuming, requires expert knowledge. Applied machine learning is basically feature engineering."* Prof. Andrew Ng.
- Expose de Pierre Courtiol (42:30), Champion Kaggle.

Regression lineaire en recherche

Energie libre

$$A(r_1, \ldots, r_{N_a}) = E(r_1, \ldots, r_{N_a}) - T \times S(r_1, \ldots, r_{N_a})$$

Calculs d'energie libre: decouvert d'une nouvelle phase $C15$ dans le Fer (Marinica et al., 2012).

Entropie vibrationnelle : function des v.p. ω_j du Hessien de l'energie

$$S = k_B \sum_j \left[\ln \left(\frac{k_B T}{\hbar \omega_j} \right) + 1 \right], \quad \frac{\hbar \omega_j}{k_B T} \ll 1$$

Cout computationel: $\mathcal{O}(N_{\text{atoms}}^3)$, prohibitif !
$\rightarrow$ machine learning pour accélérer les calculs.

Etape 1: les données

Pas de donnees disponibles → C. Marinica et C. Lapointe (CEA Saclay) creent leur propres donnees.

Figure 1: Body-centered cubic Iron

- Fer BCC
- Defauts en enlevant/ajoutant 1-4 atomes
- 31,000 configurations de 1000 → 3500 atome

Etape 2: Feature engeneering

Entropie locale: représentation locale

- Angular Fourier Series (bartok2013representing)

$$\mathcal{A}_{n,l}(\mathcal{N}_i) = \sum_{j,k \in \mathcal{N}_i} f\left(r_{ij}, r_{ik}, \theta_{jik}\right)$$

- **Roto-translation invariante**: depend des distances r_ij et angles θ_{jik}
- Global descriptor: sum of local descriptors

$$A_{n,l} = \sum_i \mathcal{A}_{n,l}(\mathcal{N}_i)$$

- Rayon de $5\dot{A}$

Étape 3: Choix de la classe de fonction et du critère

- Thermodynamique classique: l'entropie est une quantité extensive.
- Feature Angular Fourier Series est extensive: somme de descripteurs locaux.
- $\rightarrow$ **La régression linéaire possède cette propriété**. Une méthode non-linéaire enlèverait cette propriété.
- Énergie: erreur quadratique appropriée $\rightarrow$ moindres carres

Étape 4: Optimisation et résultats.

Régression linéaire: bibliotheque Scikit-Learn.

Predictions avec AFS et Solid Harmonic Scattering

- Solid Harmonic Scattering: 0.48 k_B MAE
- Angular Fourier Series: 0.18 k_B MAE

Etape 5: test d'extrapolation

- Entropies : train $5 - 25k_B$, test $10 - 250k_B$
- Capacité d'extrapolation grâce a l'extensivité.
- Un franc succès pour la régression linéaire (lien du papier).

Machine Learning

Méthodologie:

1. Trouver un problème et des données
2. Choisir une classe de fonction paramétrique
3. Choisir un critère adapté au problème
4. Résoudre le problème d'optimisation
5. Analyser les résultats: sur/sous apprentissage, choix d'hyper-paramètres, sensibilité...

Chacune de ces étapes demande un savoir faire et une certain expertise $\rightarrow$ interdisciplinaire

Régression linéaire: données binaires

$x^i \in \mathbb{R}, y^i \in \{0, 1\}$

Régression linéaire: données binaires

$\rightarrow$ L'espace de fonction linéaires avec le critère des moindres carres est inadéquate aux sorties $y^i \in \{0, 1\}$.

La Régression logistique

- Idée: régression de $f(x) = P(Y = 1|X = x)$.
- Problème: $f(x) \in]0, 1[$.
- $\rightarrow$ régression linéaire sur le log odd-ratio

$$\langle a, x \rangle = \log\left(\frac{f(x)}{1 - f(x)}\right) \in \mathbb{R}$$

$$\iff f(x) = \sigma\left(\langle a, x \rangle\right)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \text{sigmoid function}$$

La Régression logistique

Critère d'optimisation:

- Moindre carres: inadapté...
- Vraissemblance des donnees

$$p(y^i|x^i) = f(x^i)^{y^i}(1 - f(x^i))^{1-y^i}$$

- Maximum de log-vraissemblance $\mathcal{L}$

$$\theta^* = \underset{\theta}{\text{argmax}}\ \mathcal{L}(\theta)$$

$$\mathcal{L}(\theta) = \text{Log}\ \left(\prod_{i=1}^{n} p(y^i|x^i)\right)$$

$$= \sum_{i=1}^{n} \langle a, y_i x_i \rangle + \log\left(\sigma(-\langle a, x_i \rangle)\right) \quad (\theta = a))$$

La Régression logistique

Trouver l'optimum:

- Gradient de $\mathcal{L}(a)$

$$\nabla \mathcal{L}(a) = \sum_{i=1}^{n} (y^i - \sigma\left(\langle a, x^i \rangle\right))x^i$$

- Annulation du gradient

$$\nabla \mathcal{L}(a) = 0 \ ?$$

$\rightarrow$ pas de forme close du type $a = ...$

- Comment trouver un point qui annule le gradient ? Y en a-t-il plusieurs ?

La Descente de Gradient

But: trouver un **point critique**, *i.e.* a^* tel que

$$\nabla F(a^*) = 0. \tag{1}$$

Algorithme de descente de gradient:

- Initialiser a_0 (zéro, aléatoire...).
- $t = 1 \ldots$:

$$a_{t+1} = a_t - \lambda_t \nabla F(a_t) \ . \qquad \text{(Gradient Descent)}$$

Stratégie naturelle pour descendre d'une montagne dans le brouillard!

λ_t: i.e. **learning rate** ou **step-size**.

- λ_t trop petit: "sur" mais très long.
- λ_t trop grand: incertain voire instable.

La Régression logistique

Convexité:

- La fonction $\theta \mapsto \mathcal{L}(\theta)$ est convexe:

- Fonction convexe: l'annulation du gradient est une condition nécessaire **et suffisante** d'optimalité.
- Solution: descente de gradient

Mise en pratique: notebook sur la régression logistique.

Fonction isotrope vs. anisotrope

Descente de gradient a pas fixe $\delta = 0.1$:

$\rightarrow$ centrer et réduire les features d'entrée facilite l'optimisation !

Différentiation automatique

$$y = f(g(h(x))) = f(g(h(w_0))) = f(g(w_1)) = f(w_2) = w_3$$

$$w_0 = x$$

$$w_1 = h(w_0)$$

$$w_2 = g(w_1)$$

$$w_3 = f(w_2) = y$$

Règle de la chaîne:

$$\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx} = \frac{df(w_2)}{dw_2}\frac{dg(w_1)}{dw_1}\frac{dh(w_0)}{dx}$$

Différentiation automatique

Deux méthodes récursives:

- Propagation directe

$$\frac{dw_i}{dx} = \frac{dw_i}{dw_{i-1}} \frac{dw_{i-1}}{dx}$$

- **Retro-propagation (Backpropagation)**

$$\frac{dy}{dw_i} = \frac{dy}{dw_{i+1}} \frac{dw_{i+1}}{dw_i}$$

Pas besoin de s'embêter avec le calcul des gradients, l'ordinateur le fait pour nous !

Mise en pratique : notebook sur la régression logistique.

Pour aller plus loin

Quelques liens:

- la famille exponentielle de loi de probas
- la regression logistique
- le papier de l'algorithme Adam