Apprentissage statistique et réseaux de neurones

Chapitre 1: Introduction.

Frédéric Richard
frederic.richard@univ-amu.fr

Master Mathématiques appliquées, statistique (1ère année),
Parcours Data Science

2025

Outline

Classification d'images.

- Problème : assigner une catégorie à une image.
- Applications :
  - Imagerie médicale : détecter la présence de pathologie (dépistage) ou évaluer le stade d'une maladie (diagnostic).
  - Sécurité, biométrie : reconnaissance de visages, d'empreintes, etc.
  - Reconnaissance de caractères manuscrits, recherche par le contenu (internet),...

Imagenet et le défi ILSVRC.

- ImageNet :
  - Base de données d'images naturelles sur internet annotées pour la recherche en vision par ordinateur.
  - Annotation : présence / absence d'un objet (tigre, chat,...), participative.
  - Plus de 14 millions d'images et environ 21000 classes.
- ImageNet Large Scale Visual Recognition Challenge (2010-17) :
  - But : prédire classes des objets présents dans les images.
  - Progrés énormes apportés par les réseaux de neurones convolutifs profonds (DCNN).
  - avant 2012 : taux d'erreur à 25 %,
  - 2012 (AlexNet, 60 millions de paramètres) : 16 %,
  - 2014 (GoogleNet, 6 millions, plus profond) : ¡7 %,
  - 2015 (ResNet, très profond, 152 couches) : ¡3,6 %,
  - possibilité de dépasser capacités humaines.

Classification.

- Etant donnée l'observation de variables $(x_j)_{j=1}^p$ dans $\mathbb{R}^p$ pour un individu, déterminer la classe y dans $\{1, \cdots, K\}$ de l'individu.
- Une image : ensemble de niveaux de gris $I_{m,n}$ indexés par des positions (pixels) (m, n).
- Une image de taille $L \times M$ peut se transformer en un vecteur de taille $p = L\,M$:

$$x_j = I_{l,m}, \quad \text{pour} \quad j = (l-1)\,M + m.$$

- Variables de régression : niveaux de gris de l'image.

Régression logistique.

- Observations : $(x_i, Y_i)_{i=1}^n$, $x_i \in \mathrm{R}^p$, $Y_i \in \{-1, 1\}$.
- Hyp. : Y_i v.a. indépendantes de loi de Bernoulli vérifiant $\mathbb{P}(Y_i = 1) = p_i(\theta)$ et $\mathbb{P}(Y_i = -1) = 1 - p_i(\theta)$ pour $\theta = (\theta_j)_{j=0}^p$ dans $\mathbb{R}^{p+1}$ et

$$p_i(\theta) = \sigma\left(\theta_0 + \sum_{j=1}^{p} \theta_j x_i^{(j)}\right) \text{ avec } \sigma(t) = \frac{1}{1+e^{-t}}.$$

- Vraisemblance du modèle :

$$\mathcal{L}(\theta; y_1, \cdots, y_n) = \prod_{i=1}^{n} p_i(\theta)^{\frac{y_i+1}{2}} (1 - p_i(\theta))^{\frac{1-y_i}{2}}.$$

- L'EMV de θ minimise la fonction de perte logistique :

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i \langle \theta, \tilde{x}_i \rangle}\right) \text{ avec } \tilde{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}.$$

Lien avec les réseaux de neurones.

- x_i : couche d'entrée du réseau.
- $\theta_0 + \sum_{j=1}^{p} \theta_j x_i^{(j)}$: couche dense.
- $\sigma(t) = \frac{1}{1+e^{-t}}$: activation.
- $z_i = \sigma(\theta_0 + \sum_{j=1}^{p} \theta_j x_i^{(j)})$: couche de sortie du réseau.
- Apprentissage par minimisation de la fonction de perte logistique :

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i \langle \theta, \tilde{x}_i \rangle}\right) \quad \text{avec } \tilde{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}.$$

Régression multinomiale.

- Observations : $(x_i, Y_i)_{i=1}^n$ variable $(\in \mathrm{R}^p)$ / classe $(\in \{1, \cdots, K\})$.
- On suppose que les Y_i sont des v.a. indépendantes de lois de probabilité (modèle soft-max) :

$$\mathbb{P}(Y_i = k) = p_i^{(k)}(\theta) = \frac{\exp(\langle \theta^{(k)}, \tilde{x}_i \rangle)}{\sum_{m=1}^K \exp(\langle \theta^{(m)}, \tilde{x}_i \rangle)}.$$

où les paramètres $\theta^{(k)}$ associés à chaque classe k sont dans $\mathbb{R}^{p+1}$.

- L'EMV de $\theta = (\theta^{(k)})_{k=1}^K$ réalise un minimum du critère d'entropie croisée (cross entropy) :

$$S(\theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}_{y_i=k} \log(p_i^{(k)}(\theta))$$

Outline

Apprentissage statistique.

- On dispose d'observations $(x_i, y_i)_{i=1}^n$. On suppose que ces observations sont distribuées selon la même loi de distribution inconnue P que des variables (X, Y) à valeurs dans $\mathcal{X} \times \mathcal{Y}$.
- L'objectif est de prédire Y à partir de X en utilisant une famille $\mathcal{F}$ de fonctions paramétrées par θ : pour $f_\theta \in \mathcal{F}$, la prévision de Y est donnée par $\widehat{Y} = f_\theta(X)$.
- On définit une fonction de coût ℓ qui mesure l'erreur de prévision de Y par $\widehat{Y}$.
- On cherche θ minimisant le risque théorique :

$$\mathbb{E}(\ell(f_\theta(X), Y)) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f_\theta(x), y) dP(x, y).$$

- P étant inconnue, on minimise le risque empirique :

$$E_n(\ell(f_\theta(x), y)) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i).$$

Exemple de la régression logistique.

- Observations $(x_i, y_i)_{i=1}^n$ dans $\mathcal{X} \times \mathcal{Y}$, $\mathcal{X} = \mathbb{R}^p$ et $\mathcal{Y} = \{-1, 1\}$.
- Famille de fonctions :

$$\mathcal{F} = \{x \to \theta_0 + \sum_{j=1} \theta_j x^{(j)}, \theta \in \mathbb{R}^{p+1}\}.$$

- Fonction de coût :

$$\ell(y, z) = \log(1 + \exp(-yz)).$$

- Risque empirique : fonction de perte logistique

$$E_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \langle \tilde{x}_i, \theta \rangle)) \text{ avec } \tilde{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}.$$

Problème du sur-ajustement (over-fitting).

- En pratique, on cherche $\widehat{\theta} \in \arg\min_\theta E_n(\ell(f_\theta(x), y))$.
- Erreur d'apprentissage : $E_n(\ell(f_{\widehat{\theta}}(x), y))$.
  - Évalue la qualité de l'approximation des y_i par les x_i au moyen des fonctions de $\mathcal{F}$. Si trop élevée, mauvaise approximation par $\mathcal{F}$.
  - Mais, ne quantifie pas la capacité de prévision de Y par X.
- Erreur de généralisation : $\mathbb{E}(\ell(f_{\widehat{\theta}}(X), Y))$.
  - Non évaluable (car P inconnue),
  - Peut être estimée avec d'autres exemples que ceux de l'apprentissage.
- Phénomène de sur-ajustement :
  - Erreur d'apprentissage faible/erreur de généralisation élevée.
  - Origine : $p >> n$, trop de paramètres par rapport au nombre d'exemples d'apprentissage.

Outline

Descente de gradient.

- L'apprentissage donne lieu au problème d'optimisation :

$$\widehat{\theta} \in \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} C_i(\theta) \quad \text{avec} \quad C_i(\theta) = \ell(f_\theta(x_i), y_i)).$$

- Si C_i est différentiable, on peut approcher $\widehat{\theta}$ avec un algorithme de descente de gradient :
  - Itération $t+1$:

$$\theta(t+1) = \theta(t) - \rho_t \frac{1}{n} \sum_{i=1}^{n} \nabla C_i(\theta(t)).$$

  - Convergence vers un minimum local sous certaines conditions.
  - Coût calculatoire augmente avec le nombre d'exemples n.

Descente de gradient stochastique.

- Itération $t+1$: $\theta(t+1) = \theta(t) - \rho_t \nabla C_{i_t}(\theta(t))$, où i_t est un indice tiré aléatoirement dans $\{1, \cdots, n\}$.
- Avantages :
  - réduit les temps de calcul,
  - peut s'extraire de minima locaux peu profonds,
  - peut atténuer les effets de sur-ajustement.
- Variante : gradient stochastique par mini-lots (batch).
  Époque $t+1$:
  - Initialisation : $\tilde{\theta}(1) = \theta(t)$ puis pour $u = 1, \cdots, L-1$,

$$\tilde{\theta}(u+1) = \tilde{\theta}(u) - \rho_t \frac{1}{m} \sum_{i \in B_u(t)} \nabla C_i(\tilde{\theta}(u)),$$

    où, à t fixé, les $B_u(t)$ sont des ensembles disjoints de m indices tirés aléatoirement dans $\{1, \cdots, n\}$.
  - $\theta(t+1) = \tilde{\theta}(L)$.

Outline

Test

- Après optimisation, on estime habituellement le risque théorique $\mathbb{E}(\ell(f_{\widehat{\theta}}(X), Y))$ par

$$\tilde{E}_m(\ell(f_{\widehat{\theta}}(\tilde{x}), \tilde{y})) = \frac{1}{m} \sum_{i=1}^{m} \ell(f_{\widehat{\theta}}(\tilde{x}_i), \tilde{y}_i).$$

  au moyen de nouveaux exemples $(\tilde{x}_i, \tilde{y}_i)_{i=1}^m$.
- On peut comparer différentes familles paramétriques $\mathcal{F}$ à partir de cette estimation.
- Dans le cas de la classification, on peut aussi évaluer le modèle en comptant le nombre d'erreurs de classification faites sur les données de test avec le modèle $f_{\widehat{\theta}}$.
- Pour faire ces évaluations, il faut suffisamment d'exemples. Si ce n'est pas le cas, on peut faire des validations croisées.

Validation

- Pendant l'optimisation, on peut également suivre l'évolution au cours des itérations (ou époques) d'une estimation du risque théorique.
- Cela permet de déceler un sur-ajustement.
- Cela permet de fixer manuellement des hyper-paramètres (taille de batch, learning rate, etc.).
- Pour éviter un biais d'estimation, il faut veiller à ce que les exemples de validation soient différents des exemples d'apprentissage et de test.

Outline

- Librairies python pour l'apprentissage de réseaux de neurones : Keras, Pytorch.
- "PyTorch is an optimized tensor library for deep learning using GPUs and CPUs"
- Classe de base = "Tensor" (tenseur, tableau multidimensionnel)
  - dimension $d = 2$: matrice.
  - extension en dimension $d > 2$.
- Différents modules :
  - Module **torch.nn** (pour Neural Networks) permet de définir des réseaux de neurones.
  - Module **torch.optim** permet de faire l'apprentissage de réseaux de neurones.