Algorithmique des données
Régression

Charlotte Pelletier
MCF Univ. Bretagne Sud – IRISA Vannes
Basé sur le cours de Chloé Friguet (MCF UBS/IRISA).

11 mars 2020

Rappels

Rappels

- **Apprentissage supervisé** : dans les données observées, on connaît la "vraie" valeur de la variable de sortie et on cherche à comprendre/prédire le lien supposé entre les variables d'entrée et de sortie
- **Nature de la variable de sortie** (Y) ?
  - **quantitative** : **régression**
  - qualitative (à 2 ou >2 modalités) : classification (binaire / multiclasses)
- **Nature et nombre de variables d'entrée** (X) ?
  - nature : **qualitatives** et/ou **quantitatives**
  - **Une** seule variable
    - peu fréquent en pratique, mais utile pour bien comprendre ce qu'il se passe ⇒ visualisation
  - **Plusieurs** variables
    - plusieurs = de quelques dizaines à plusieurs (dizaines de) milliers ⇒ sélection de variables
    - sélection de variables, parcimonie
    - colinéarité

Modèles linéaire et logistique (1/4)

Analyse de la relation entre **Y** et toutes les variables $[\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^d]$:

- Régression linéaire (**Y** quantitative)

$$y_i \approx f_{\boldsymbol{\beta}}(\mathbf{x}_i) = f_{\boldsymbol{\beta}}(x_i^1, x_i^2 \cdots, x_i^d) = \beta_0 + \beta_1 x_i^1 + \beta_2 x_i^2 + \ldots + \beta_d x_i^d$$

- Régression logistique (**Y** binaire codée 0/1)

$$f_{\boldsymbol{\beta}}(\mathbf{x}_i) = \frac{e^{\mathbf{x}_i' \boldsymbol{\beta}}}{1 + e^{\mathbf{x}_i' \boldsymbol{\beta}}} = \mathbb{P}(Y = 1 | X = \mathbf{x}_i)$$

On cherche $\boldsymbol{\beta}$ tel que $f_{\boldsymbol{\beta}}(\mathbf{x}_i)$ est proche de y_i pour toutes les données d'apprentissage $\{\mathbf{x}_i, y_i\}_{i=1}^m$ avec $\mathbf{x}_i \in \mathbb{R}^d$

- Notation matricielle :

$$f_{\boldsymbol{\beta}}(\mathbf{X}) \approx \tilde{\mathbf{X}}\boldsymbol{\beta}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \\ y_m \end{pmatrix} \approx \begin{pmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^d \\ 1 & x_2^1 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \vdots & \\ \\ 1 & x_m^1 & x_m^2 & \dots & x_m^d \end{pmatrix} \quad \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \\ \beta_d \end{pmatrix}$$

avec $\mathbf{X} \in \mathbb{R}^{m \times d}$, $\tilde{\mathbf{X}} \in \mathbb{R}^{m \times (d+1)}$ et $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$

Coût (quadratique) **global** des erreurs

- Régression linéaire :

$$\sum_{i=1}^{m}\Big(f_{\boldsymbol{\beta}}(\mathbf{x}_i) - y_i\Big)^2$$

- Régression logistique :

$$\sum_{i=1}^{m}\Big[y_i log\Big(f_{\boldsymbol{\beta}}(\mathbf{x}_i)\Big) + (1 - y_i) log\Big(1 - f_{\boldsymbol{\beta}}(\mathbf{x}_i)\Big)\Big]$$

Objectif : minimiser le coût global des erreurs :

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big(\mathbf{J}(\boldsymbol{\beta}) \Big)$$

- Régression linéaire : solution explicite (Moindres Carrés Ordinaires) - si $S = (\mathbf{X}'\mathbf{X})$ est inversible

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big(\mathbf{J}(\boldsymbol{\beta}) \Big) \quad = \quad \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \, ||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}|| = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

- Régression logistique : pas de solution analytique explicite, besoin d'algorithmes d'optimisation itératifs type descente de gradient (et variantes)

Itération k de l'algo. de descente du gradient - rég. logistique

$$\beta_j^{(k)} \quad := \quad \beta_j^{(k-1)} - \alpha \frac{1}{m} \sum_{i=1}^{m} \Big(f_{\boldsymbol{\beta}^{(k-1)}}(\mathbf{x}_i) - y_i \Big) x_i^j$$

Remarque : $\forall i, x_i^0 = 1$

Compromis biais-variance

Ouvrez le Jupyter Notebook `CM06_polynom.ypynb`.

Comment mesurer la qualité de l’ajustement ?

- en fonction de la qualité des prédictions
- en fonction des conséquences des actions (les prédictions pouvant être vues comme un type d’action particulier)
- la qualité doit pouvoir être mesurée sur une échelle positive ou négative (par exemple, une fonction de coût)

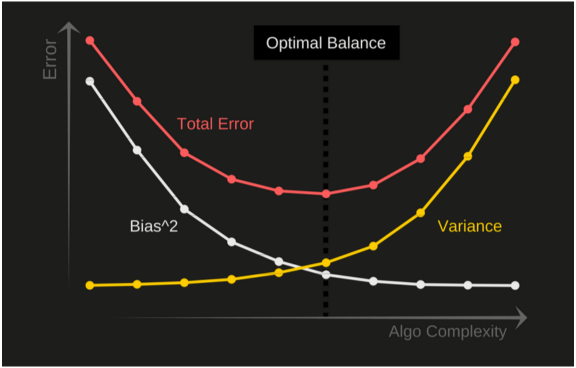
Qualité de l'ajustement

- **Généralisation** : propriété importante de l'apprentissage
  - La généralisation représente la capacité du modèle à pouvoir effectuer des prédictions robustes sur des nouvelles données.
- **Sur/sous-apprentissage** = modèle qui ne donne pas de bons résultats de généralisation

➜ Compromis nécessaire entre biais (sous-ajustement) et variance (sur-ajustement)

Décomposition biais-variance

Erreur total = Biais^2 + Variance + erreur



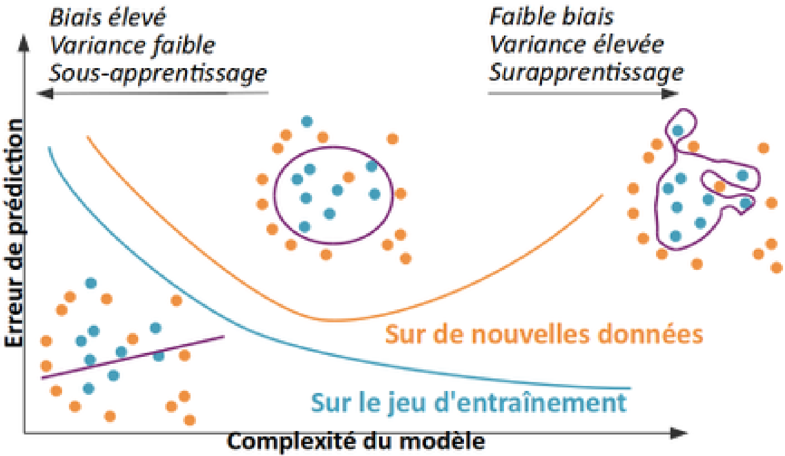
Source : `https://elitedatascience.com/bias-variance-tradeoff`

Par exemple, l'erreur des moindres carrés

$$\mathbb{E}\Big[(Y-\hat{Y})^2\Big] \quad = \quad \overset{\sigma^2}{\mathbf{V}(Y)} + \overset{\text{Variance}}{\mathbf{V}(\hat{Y})} + \overset{\text{Biais}^2}{\Big[Y-\mathbb{E}(\hat{Y})\Big]^2}$$

[Démonstration en cours]

- Bien sélectionner un modèle
  - Modèle complexe (à haute variance) $\Rightarrow$ phénomène sous-jacent mal représenté, modèle trop dépendant aux données d'apprentissage et au bruit (fluctuations aléatoires, non représentatives du phénomène)
  - Modèle simple (biais) $\Rightarrow$ complexité du phénomène non capturée, modèle trop généraliste pour fournir des prédictions précises
  - ➜ on cherche un compromis !
- Comment bien choisir un modèle ? [CM08]
  - échantillons d'apprentissage : pour construire le modèle
  - échatillons de validation : pour choisir la valeur de ses hyperparamètres
  - échantillons test : pour évaluer ses performances en terme de prédiction sur des nouvelles données



Source : openclassroom

Régularisation

Régularisation

- Objectif : ajouter de l'information pour éviter le sur-apprentissage en pénalisant la complexité du modèle

Régularisation

- Objectif : ajouter de l'information pour éviter le sur-apprentissage en pénalisant la complexité du modèle
- Solution : on garde toutes les variables candidates dans le modèle mais on ajoute une norme sur les paramètres dans la fonction coût
  - Norme $\mathcal{L}_1$: $||\boldsymbol{\beta}||_1 = \sum_j |\beta_j|$
  - Norme $\mathcal{L}_2$: $||\boldsymbol{\beta}||_2^2 = \sum_j \beta_j^2$
- Conséquences :
  - on contrôle les valeurs de certains paramètres, le modèle est donc plus simple et plus facilement généralisable.
  - le modèle sera plus performant puisque on diminue (l'espérance de) l'erreur de prédiction.

En pratique

On modifie le problème d'optimisation en ajoutant un terme de pénalisation : maximisation de la vraisemblance des données tout en ayant une valeur acceptable pour le terme de pénalisation

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big(\mathbf{J}(\boldsymbol{\beta}) - \lambda \mathcal{R}(\boldsymbol{\beta}) \Big)$$

- $\mathcal{R}(\boldsymbol{\beta})$: terme de pénalisation (fonction de $\boldsymbol{\beta}$ positive)
- $\lambda > 0$: poids accordé à la pénalisation

Pénalisation ridge

Pénalisation "*ridge*" (*shrinkage* $\sim$ rétrécissement) = on force les coefficients à prendre de petites valeurs $\Rightarrow$ régularisation $\mathcal{L}_2$

- Régression linéaire :

$$\mathbf{J}(\boldsymbol{\beta}, \lambda) = \frac{1}{2m} \sum_{i=1}^{m} \Big(f_{\boldsymbol{\beta}}(\mathbf{x}_i) - y_i \Big)^2 + \frac{\lambda}{2m} \sum_{j=1}^{d} \beta_j^2$$

Solution explicite : $\boldsymbol{\beta}^* = \Big[(\mathbf{X}'\mathbf{X}) + \lambda \mathbb{I} \Big]^{-1} \mathbf{X}'\mathbf{Y}$

- Régression logistique :

$$\mathbf{J}(\boldsymbol{\beta}, \lambda) = -\frac{1}{m} \sum_{i=1}^{m} \Big[y_i log\Big(f_{\boldsymbol{\beta}}(\mathbf{x}_i) \Big) + (1 - y_i) log\Big(1 - f_{\boldsymbol{\beta}}(\mathbf{x}_i) \Big) \Big] + \frac{\lambda}{2m} \sum_{j=1}^{d} \beta_j^2$$

$\sim$ *weight-decay* (algorithme de descente de gradient stochastique)

Pénalisation ridge

Itération k de l'algorithme de descente du gradient avec régularisation

$$
\begin{aligned}
\beta_0^{(k)} &:= \beta_0^{(k-1)} - \frac{\alpha}{m}\sum_{i=1}^{m}\left(f_{\boldsymbol{\beta}^{(k-1)}}(\mathbf{x}_i) - y_i\right) \\
\beta_j^{(k)} &:= \beta_j^{(k-1)} - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(f_{\boldsymbol{\beta}^{(k-1)}}(\mathbf{x}_i) - y_i\right)x_i^j + \frac{\lambda}{m}\beta_j^{(k-1}\right] \\
&= \beta_j^{(k-1)}\left(1 - \frac{\alpha\lambda}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(f_{\boldsymbol{\beta}^{(k-1)}}(\mathbf{x}_i) - y_i\right)x_i^j
\end{aligned}
$$

Pénalisation *LASSO*

Pénalisation *LASSO* (*Least Absolute Shrinkage and Selection Operation*) = on force les coefficients à prendre de petites valeurs ⇒ régularisation $\mathcal{L}_1$

- Régression linéaire :

$$\mathbf{J}(\boldsymbol{\beta}, \lambda) = \frac{1}{2m} \sum_{i=1}^{m} \Big(f_betav(\mathbf{x}_i) - y_i \Big)^2 + \frac{\lambda}{2m} \sum_{j=1}^{d} |\beta_j|$$

- Régression logistique :

$$\mathbf{J}(\boldsymbol{\beta}, \lambda) = -\frac{1}{m} \sum_{i=1}^{m} [y_i log \Big(f_{\boldsymbol{\beta}}(\mathbf{x}_i) \Big) + (1 - y_i) log \Big(1 - f_{\boldsymbol{\beta}}(\mathbf{x}_i) \Big) \Big] + \frac{\lambda}{2m} \sum_{j=1}^{d} |\beta_j|$$

Remarques

- Cas de la constante :
  - on ne régularise pas β_0 (le biais)
- Les variables X doivent être centrées et réduites afin de limiter l'influence des variables à forte variance (tout en gardant $\forall i, x_i^0 = 1$)

Remarques

Remarques sur la pénalisation LASSO (uniquement)

- Pas d'algorithme de calcul direct des coefficients $\Rightarrow$ utilisation d'approches itératives partant de $\forall j,\ \beta_j = 0$
- Effet LASSO
  - coefficients à 0 $\Rightarrow$ variables exclues du modèle
  - sélection de variable (par exemple sélection d'une des variables dans un groupe de variables corrélées)
- LASSO permet d'avoir au maximum m coefficients non nuls - Cas $m < d$?

Coefficient de régularisation

Remarques

- Rôle de λ :
  - $\lambda \mapsto +\infty$: tous les coefficients $\beta \mapsto 0$
  - $\lambda = 0$: pas de régularisation
- Choix de λ par validation croisée (minimisation de l'erreur de prédiction)

Elasticnet

Combinaison des régressions *ridge* et *LASSO*

- Régression linéaire :

$$\mathbf{J}(\boldsymbol{\beta}, \lambda_1, \lambda_2) = \frac{1}{2m} \sum_{i=1}^{m} \Big(f_b etav(\mathbf{x}_i) - y_i \Big)^2 + \frac{\lambda_1}{2m} \sum_{j=1}^{d} |\beta_j| + \frac{\lambda_2}{2m} \sum_{j=1}^{d} \beta_j^2$$

- Régression logistique :

$$\begin{aligned} \mathbf{J}(\boldsymbol{\beta}, \lambda_1, \lambda_2) &= -\frac{1}{m} \sum_{i=1}^{m} \Big[y_i log\Big(f_{\boldsymbol{\beta}}(\mathbf{x}_i) \Big) + (1 - y_i) log\Big(1 - f_{\boldsymbol{\beta}}(\mathbf{x}_i) \Big) \Big] \\ &\quad + \frac{\lambda_1}{2m} \sum_{j=1}^{d} |\beta_j| + \frac{\lambda_2}{2m} \sum_{j=1}^{d} \beta_j^2 \end{aligned}$$

Elasticnet

Autre paramétrisation possible

- Régression linéaire :

$$\mathbf{J}(\boldsymbol{\beta}, \lambda, \alpha) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_betav(\mathbf{x}_i) - y_i\right)^2 + \lambda\left[\frac{\alpha}{2m}\sum_{j=1}^{d}|\beta_j| + \frac{1-\alpha}{2m}\sum_{j=1}^{d}\beta_j^2\right]$$

- Régression logistique :

$$\begin{aligned}\mathbf{J}(\beta, \lambda, \alpha) &= -\frac{1}{m}\sum_{i=1}^{m}\left[y_i log\left(f_{\boldsymbol{\beta}}(\mathbf{x}_i)\right) + (1-y_i)log\left(1 - f_{\boldsymbol{\beta}}(\mathbf{x}_i)\right)\right] \\ &\quad + \lambda\left[\frac{\alpha}{2m}\sum_{j=1}^{d}|\beta_j| + \frac{1-\alpha}{2m}\sum_{j=1}^{d}\beta_j^2\right]\end{aligned}$$

Elasticnet

Remarques

- Sélection de variable (coefficient = 0) – comme LASSO
- Groupe de variables corrélées : partage des poids – comme Ridge
- Estimation des coefficient par optimisation (*Coordinate descent algorithm*)
- Choix de λ_1 et λ_2 : procédure en deux étapes