TP1 – Estimation de paramètres

Ce TP vise à illustrer l'estimation de paramètres dans le cas d'une courbe de Bézier de degré variable. Une courbe de Bézier de degré d est définie par ses paramètres, en l'occurrence $d+1$ « points de contrôle ». Dans le cadre de ce TP, nous supposons que ces points de contrôle sont des points du plan $P_i = (\alpha_i, \beta_i)$, $i \in \{0, \ldots, d\}$, dont les abscisses α_i sont uniformément réparties dans l'intervalle $[0, 1]$, c'est-à-dire que $\alpha_i = i/d$, mais dont les ordonnées β_i peuvent être librement choisies dans $\mathbb{R}$. Le modèle non bruité de la courbe est défini par :

$$\left\{ \begin{array}{ccc} [0,1] & \rightarrow & \mathbb{R} \\ x & \mapsto & y = f(\beta_0, \beta, \beta_d, x) \end{array} \right. \tag{1}$$

où :

- Le vecteur $\beta = [\beta_1, \ldots, \beta_{d-1}]$ a pour coordonnées les paramètres de la courbe autres que β_0 et β_d.
- La fonction $f(\beta_0, \beta, \beta_d, x)$ est définie par :

$$f(\beta_0, \beta, \beta_d, x) = \sum_{i=0}^{d} \beta_i \, B_i^d(x) \tag{2}$$

- Les « polynômes de Bernstein » $B_i^d(x)$, $i \in \{0, \ldots, d\}$, sont définis par :

$$B_i^d(x) = \mathrm{C}_i^d \, x^i (1-x)^{d-i} \tag{3}$$

  où C_i^d est le nombre de combinaisons de i objets parmi d, calculable par la fonction `nchoosek` de Matlab.

On remarque que :

$$\left\{ \begin{array}{l} f(\beta_0, \beta, \beta_d, 0) = \beta_0 \\ f(\beta_0, \beta, \beta_d, 1) = \beta_d \end{array} \right.$$

ce qui signifie que le premier point de contrôle $P_0 = (\alpha_0, \beta_0)$ et le dernier point de contrôle $P_d = (\alpha_d, \beta_d)$ se trouvent sur la courbe de Bézier. En général, les autres points de contrôle ne se trouvent pas sur cette courbe, mais jouent seulement le rôle d'« attracteurs ».

Données d'apprentissage

Les données d'apprentissage sont produites à partir d'une courbe de Bézier dont les paramètres sont notés β_0^*, β^* et β_d^*, bruitée par un bruit blanc additif gaussien d'écart-type $\sigma = 0,5$. Ces données d'apprentissage constituent un ensemble $\mathcal{D}_{\text{app}} = \{(x_j, y_j), j \in \{1, \ldots, n_{\text{app}}\}\}$ de n_{app} points du plan, dont les abscisses x_j sont uniformément réparties dans l'intervalle $[0, 1]$ et dont les ordonnées y_j sont définies par :

$$y_j = f(\beta_0^*, \beta^*, \beta_d^*, x_j) + b_j \tag{4}$$

où :

$$b_j \sim \mathcal{N}(0, \sigma^2) \tag{5}$$

Le script `donnees_apprentissage`, qui vous est donné, appelle les fonctions `bezier` et `bezier_bruitee`. La fonction `bezier` calcule la valeur de l'ordonnée d'une courbe de Bézier selon le modèle (1)+(2)+(3). Elle utilise les opérateurs `.^` et `.*` de Matlab afin d'être valide quelle que soit la dimension du vecteur `x`. Quant à la fonction `bezier_bruitee`, elle permet de produire des données d'apprentissage selon le modèle (4)+(5), à l'aide de la fonction `randn` de Matlab.

Exercice 1 : estimation des paramètres d'une courbe de Bézier

On souhaite estimer les paramètres d'un modèle à partir des données d'apprentissage $\mathcal{D}_{\text{app}}$. Le modèle retenu est à nouveau une courbe de Bézier. On suppose connus les paramètres β_0^* et β_d^*. Les paramètres inconnus sont donc le degré d de la courbe de Bézier et le vecteur $\beta = [\beta_1, \ldots, \beta_{d-1}]$. Ce problème d'estimation étant linéaire vis-à-vis de β, mais non linéaire vis-à-vis de d, il semble raisonnable de commencer par faire une hypothèse sur la valeur du degré d et d'estimer le vecteur β.

Établissez (sur papier) les expressions de la matrice $\mathbf{A}$ et du vecteur $\mathbf{B}$ du système linéaire suivant, qui traduit le fait que la courbe de Bézier recherchée doit passer exactement par les n_{app} points (x_j, y_j) de $\mathcal{D}_{\text{app}}$:

$$\mathbf{A}\,\beta^\top = \mathbf{B} \tag{6}$$

Écrivez la fonction `moindres_carres`, appelée par le script `exercice_1`, censée retourner la solution approchée $\widehat{\beta}$ de l'équation (6), au sens des moindres carrés. Pour cela, utilisez soit l'opérateur \ de Matlab, soit la fonction `pinv` (« pseudo-inverse »).

Testez différentes valeurs du degré d entre 2 et 20. Pour les valeurs élevées de d, comprenez-vous ce que signifie la notion de « sur-apprentissage » ?

Exercice 2 : calcul de l'erreur d'apprentissage

L'erreur d'apprentissage (ou *risque empirique*) est définie comme l'écart quadratique moyen entre les données d'apprentissage y_j et les prédictions obtenues à partir du modèle appris $f(\beta_0^*, \widehat{\beta}, \beta_d^*, x_j)$.

Écrivez la fonction `erreur_apprentissage`, appelée par le script `exercice_2`, qui calcule l'erreur d'apprentissage correspondant à la valeur du degré d passée en paramètre. Il n'est pas évident de déduire de cette courbe une méthode d'estimation du degré d. Une première manière pour estimer d nécessite de faire appel à la notion de « données de test ».

Données de test

Si l'on a la possibilité de générer n_{test} nouvelles données censées correspondre au degré d recherché, ces données $\mathcal{D}_{\text{test}} = \{(x_k, y_k), k \in \{1, \ldots, n_{\text{test}}\}\}$ sont appelées « données de test », à ne pas confondre avec les données d'apprentissage $\mathcal{D}_{\text{app}} = \{(x_j, y_j), j \in \{1, \ldots, n_{\text{app}}\}\}$. En général, on choisit n_{test} grand, car le risque empirique calculé sur $\mathcal{D}_{\text{test}}$ est un estimateur non biaisé de l'erreur de généralisation.

Le script `donnees_test`, qui est similaire au script `donnees_apprentissage`, permet de produire $n_{\text{test}} = 200$ données de test (x_k, y_k) d'abscisses uniformément réparties dans l'intervalle $[0, 1]$, avec les mêmes paramètres que ceux utilisés pour la création des données d'apprentissage. Vérifiez que le script `donnees_test` produit correctement les données de test.

Exercice 3 : calcul de l'erreur de généralisation

L'erreur de généralisation (ou *risque espéré*) est définie comme l'écart quadratique moyen entre les données de test y_k et les prédictions obtenues à partir du modèle appris $f(\beta_0^*, \widehat{\beta}, \beta_d^*, x_k)$, où $\widehat{\beta}$ est estimé à partir des données d'apprentissage (et non pas à partir des données de test !).

Écrivez la fonction `erreur_generalisation`, appelée par le script `exercice_3`, qui calcule l'erreur de généralisation correspondant à la valeur du degré d passée en paramètre.

Écrivez la fonction `estimation_1_d_sigma`, également appelée par ce script, qui doit retourner une estimation $\widehat{d}$ du degré d de la courbe de Bézier égale au minimiseur de l'erreur de généralisation, et une estimation $\widehat{\sigma}$ de l'écart-type σ du bruit sur les données égale à la racine carrée du minimum de l'erreur de généralisation.

Exercice 4 : validation croisée de type *leave-one-out*

Dans le cas où l'on n'a pas la possibilité de générer des données supplémentaires, on peut se servir des données d'apprentissage $\mathcal{D}_{\text{app}}$ pour tester le modèle. Cette approche s'appelle la « validation croisée » (*cross validation*). Dans sa version *leave-one-out*, qui signifie « une [donnée] laissée de côté », la validation croisée VC, qui dépend du degré d de la courbe de Bézier, s'écrit :

$$VC = \frac{1}{n_{\text{app}}} \sum_{j=1}^{n_{\text{app}}} \Big[y_j - f(\beta_0^*, \widehat{\beta}_j, \beta_d^*, x_j) \Big]^2 \tag{7}$$

où le vecteur de paramètres $\widehat{\beta}_j$ est estimé avec les données d'apprentissage $\mathcal{D}_{\text{app}} \backslash \{(x_j, y_j)\}$. Cela signifie que chaque point (x_j, y_j) sert $n_{\text{app}} - 1$ fois de donnée d'apprentissage, et une fois de donnée de test.

Écrivez la fonction `calcul_VC`, appelée par le script `exercice_4`, qui doit calculer l'expression (7) de VC pour la valeur du degré d passée en paramètre.

Écrivez la fonction `estimation_2_d_sigma`, également appelée par ce script, qui doit retourner une estimation $\widehat{d}$ du degré d de la courbe de Bézier égale au minimiseur de VC, et une estimation $\widehat{\sigma}$ de l'écart-type σ du bruit sur les données égale à la racine carrée du minimum de VC.

Exercice 5 (facultatif) : absence de biais de l'estimation

Dans cette question, on fixe la valeur de d à 5.

On peut montrer que le vecteur de paramètres $\widehat{\beta}$ estimé en moindres carrés est distribué selon la loi normale suivante, lorsque les données d'apprentissage $\mathcal{D}_{\text{app}}$ varient :

$$\widehat{\beta} \sim \mathcal{N}(\beta^*, \sigma^2 \, (\mathbf{A}^\top \mathbf{A})^{-1})$$

Cela montre que l'estimation de paramètres, telle qu'elle a été effectuée, est non biaisée.

Écrivez un script, de nom `exercice_5`, qui tire un grand nombre de données d'apprentissage, et qui moyenne les différents vecteurs de paramètres $\widehat{\beta}$ estimés (cf. exercice 1), afin de vérifier l'absence effective de biais.