TP2 – Contrôle de la complexité par régularisation

Dans le TP1, vous avez estimé le degré d'une courbe de Bézier en minimisant soit l'erreur de généralisation, soit la validation croisée. Nous nous intéressons ici à une autre approche utilisant la notion de *régularisation*. Plus précisément, nous allons tenter de contrôler la complexité d'une courbe de Bézier par « estimation écrêtée » de paramètres.

Choix de paramètres adaptés à la régularisation

L'idée sous-jacente à la régularisation d'un modèle polynomial est de choisir un degré d élevé, par exemple $d = 16$, tout en évitant un sur-apprentissage (*over-fitting*) qui nuirait à la capacité de généralisation du modèle.

Dans le cadre de l'estimation de courbes de Bézier d'extrémités P_0 et P_d fixées, on modifie le paramétrage du problème de régression : au lieu d'estimer les $d-1$ ordonnées β_i des points de contrôle[1] $P_i = (\alpha_i, \beta_i)$, où $i \in \{1, \ldots, d-1\}$, on estime les $d-1$ écarts δ_i définis par :

$$\delta_i = \beta_i - \overline{\beta}_i \tag{1}$$

où $\overline{\beta}_i$ désigne l'ordonnée de référence, dans une situation où les points de contrôle seraient alignés (cf. figure 1). Le nouveau vecteur de paramètres δ à estimer est donc défini par :

$$\delta = [\delta_1, \ldots, \delta_{d-1}] \tag{2}$$

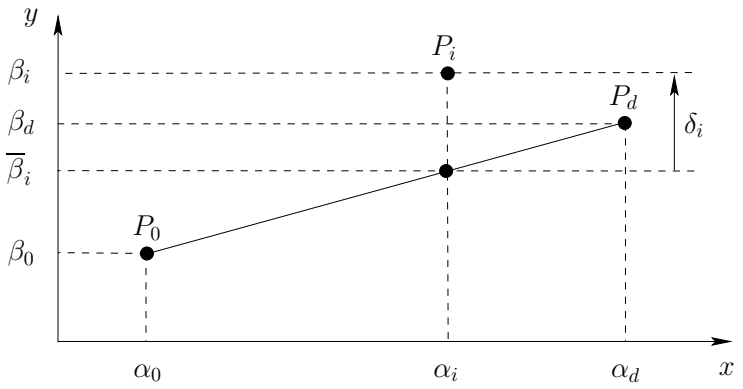


FIGURE 1 – L'écart δ_i mesure la différence entre l'ordonnée β_i du point de contrôle P_i et l'ordonnée de référence $\overline{\beta}_i$, dans une situation où les points de contrôle seraient alignés.

Lancez le script `nouveau_parametrage`, qui superpose à la courbe de Bézier du TP1 un dessin analogue à celui de la figure 1, afin de matérialiser le vecteur de paramètres δ.

1. Il est rappelé que les abscisses des points de contrôle sont uniformément réparties dans l'intervalle $[0, 1]$: $\alpha_i = i/d$.

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

Dans le TP1, l'estimation des paramètres du modèle consistait à résoudre le problème suivant :

$$\widehat{\beta} = \underset{\beta=[\beta_1,\ldots,\beta_{d-1}]}{\operatorname{argmin}} \|\mathbf{A}\,\beta^\top - \mathbf{B}\|^2 \tag{3}$$

La relation (1) liant β à δ permet de réécrire le problème (3) comme suit :

$$\widehat{\delta} = \underset{\delta=[\delta_1,\ldots,\delta_{d-1}]}{\operatorname{argmin}} \|\mathbf{A}\,\delta^\top - \mathbf{C}\|^2 \tag{4}$$

où $\mathbf{C} = \mathbf{B} - \mathbf{A}\,\overline{\beta}^\top$. Pour régulariser un modèle de grande complexité, c'est-à-dire de degré d élevé, nous pénalisons les valeurs trop élevées des écarts δ_i. Le principe de la régression écrêtée (*ridge regression*) consiste à ajouter à l'erreur d'apprentissage un terme de pénalisation :

$$\lambda \sum_{i=1}^{d-1} {\delta_i}^2 = \lambda \,\|\delta\|^2 \tag{5}$$

où $\lambda \in \mathbb{R}^+$ constitue un « hyper-paramètre ». Le problème d'estimation (4) devient alors :

$$\widehat{\delta} = \underset{\delta=[\delta_1,\ldots,\delta_{d-1}]}{\operatorname{argmin}} \|\mathbf{A}\,\delta^\top - \mathbf{C}\|^2 + \lambda\,\|\delta\|^2 = \underset{\delta\in\mathbb{R}^{1\times(d-1)}}{\operatorname{argmin}} (\mathbf{A}\,\delta^\top - \mathbf{C})^\top(\mathbf{A}\,\delta^\top - \mathbf{C}) + \lambda\,\delta\,\delta^\top \tag{6}$$

En notant $\mathbf{I}_{d-1}$ la matrice identité d'ordre $d-1$, il est aisé de trouver la solution du problème (6) :

$$\widehat{\delta}^\top = (\mathbf{A}^\top\mathbf{A} + \lambda\,\mathbf{I}_{d-1})^{-1}\mathbf{A}^\top\mathbf{C} \tag{7}$$

En vous inspirant de la fonction `moindres_carres` du TP1, écrivez la fonction `moindres_carres_ecretes`, appelée par le script `exercice_1`, qui calcule la valeur du vecteur de paramètres $\widehat{\beta}$ découlant de l'estimation (7). Attention : cette fonction doit retourner $\widehat{\beta}$, et non pas $\widehat{\delta}$.

Vérifiez que, pour $d = 16$, le choix d'une valeur très élevée de λ permet de retrouver la droite (P_0, P_d) de la figure 1. Observez l'influence de λ sur le modèle estimé.

Exercice 2 : recherche de la valeur optimale de l'hyper-paramètre λ

Dans le TP1, nous avons vu qu'en l'absence de données supplémentaires, il était possible de se servir des données d'apprentissage $\mathcal{D}_{\text{app}}$ pour tester le modèle. La validation croisée *leave-one-out* s'écrit (cf. TP1) :

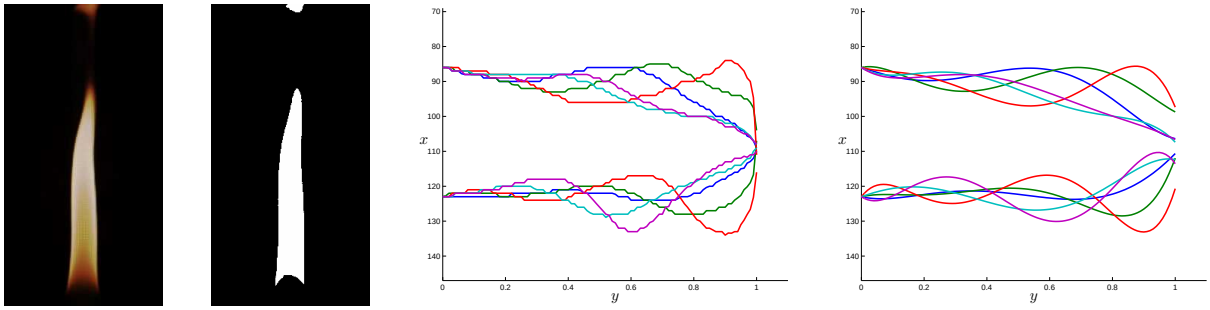
$$VC = \frac{1}{n_{\text{app}}} \sum_{j=1}^{n_{\text{app}}} \Big[y_j - f(\beta_0^*, \widehat{\beta}_j, \beta_d^*, x_j)\Big]^2 \tag{8}$$

Dans cette expression, si le vecteur de paramètres $\widehat{\beta}_j$ est estimé en minimisant un problème du type (6) pour les données d'apprentissage $\mathcal{D}_{\text{app}}\backslash\{(x_j, y_j)\}$, alors VC dépend non seulement du degré d de la courbe de Bézier, mais également de l'hyper-paramètre λ. Il est donc envisageable d'estimer la valeur optimale de λ en cherchant le minimum de VC, le degré de la courbe de Bézier étant fixé à une valeur élevée, par exemple à $d = 16$.

Dupliquez les fonctions `calcul_VC` et `estimation_d_sigma_bis` du TP1 sous les noms `calcul_VC_bis` et `estimation_lambda_sigma`. Modifiez ces deux fonctions, qui sont appelées par le script `exercice_2`, afin de montrer que la régularisation permet effectivement de contrôler la complexité du modèle (le script `exercice_2` affiche sur une même figure le modèle réel et le modèle estimé correspondant à la valeur optimale de λ). Vous constatez, en revanche, que la recherche de la valeur optimale de λ est très lente, à cause du calcul de VC (une accélération de cette recherche vous est proposée dans la partie facultative du TP).

Application : simulation réaliste d'une séquence d'images

On souhaite simuler, de la façon la plus réaliste possible, une séquence d'images d'une flamme de bougie. La figure 2 illustre les prétraitements effectués sur une séquence réelle de n images : chacune de ces images (cf. figure 2-a), soumise à un seuillage, fournit une image binaire (cf. figure 2-b), sur laquelle la silhouette de la flamme est détectée. Après normalisation, toutes ces silhouettes ont une même hauteur égale à 1. En tournant les axes d'un quart de tour vers la droite, les abscisses x sont orientées vers le bas et les ordonnées y vers la droite (cf. figure 2-c). Lancez le script `donnees`, afin de créer la matrice tridimensionnelle `bords`, de taille $m \times 2 \times n$, où $m = 101$ désigne le nombre de lignes correspondant à la hauteur de la flamme, et où $n = 10$ désigne le nombre d'images de la séquence, de telle sorte que `bords(p,1,q)` et `bords(p,2,q)`, pour $p \in \{1, \ldots, m\}$ et $q \in \{1, \ldots, n\}$, contiennent les abscisses des bords gauche et droit de la $q^{\text{ème}}$ silhouette (devenus ses bords supérieur et inférieur, après rotation d'un quart de tour), à l'ordonnée $y = (p-1)/(m-1)$. La figure 2-c montre que les silhouettes ont toutes la même base, c'est-à-dire que `bords(1,1,q)` et `bords(1,2,q)` ne dépendent pas de q (et valent, respectivement, 86 et 123).



(a) (b) (c) (d)

FIGURE 2 – (a) Une des images de la séquence réelle. (b) Détection de la silhouette par seuillage. (c) Silhouettes de la séquence réelle, après normalisation et rotation. (d) Modélisation des silhouettes par des paires de courbes de Bézier indépendantes.

Chaque bord de chaque silhouette peut être modélisé par une courbe de Bézier de degré d, entièrement définie par $d+1$ points de contrôle, dont les ordonnées $\alpha_i = i/d$, $i \in \{0, \ldots, d\}$, sont équiréparties dans l'intervalle $[0, 1]$. Les points de contrôle sont notés $P_i^q = (\beta_i^q, \alpha_i)$ pour le bord gauche (supérieur) de la $q^{\text{ème}}$ silhouette, et $Q_i^q = (\gamma_i^q, \alpha_i)$ pour son bord droit (inférieur). Les abscisses $\beta_0^q = 86$ et $\gamma_0^q = 123$ étant indépendantes de q, sont notées β_0 et γ_0. Les bords de la silhouette numéro q sont donc modélisés par les équations :

$$\begin{cases} x = \beta_0 \, B_0^d(y) + \displaystyle\sum_{i=1}^{d} \beta_i^q \, B_i^d(y) \\ x = \gamma_0 \, B_0^d(y) + \displaystyle\sum_{i=1}^{d} \gamma_i^q \, B_i^d(y) \end{cases}$$

où $y \in [0, 1]$ et où B_i^d désigne le polynôme de Bernstein de degré d.

Manifestement, modéliser la silhouette numéro q consiste à estimer les d paramètres $\beta^q = [\beta_1^q, \ldots, \beta_d^q]$ de son bord gauche et les d paramètres $\gamma^q = [\gamma_1^q, \ldots, \gamma_d^q]$ de son bord droit. La figure 2-d montre les modélisations obtenues à partir des silhouettes de la figure 2-c. Il est notable que les sommets des flammes ne sont pas fermés.

Pour que les deux courbes de Bézier se rejoignent au sommet de la flamme, il faut les coupler, c'est-à-dire faire en sorte que le point de contrôle situé au sommet de la flamme, à l'ordonnée $y = 1$, soit commun aux deux courbes. Il faut donc reformuler le problème sous la forme d'un nouveau système linéaire :

$$\mathbf{E}^q \, \mathbf{X}^q = \mathbf{F}^q \tag{9}$$

où le vecteur des inconnues vaut $\mathbf{X}^q = [\beta_1^q, \ldots, \beta_{d-1}^q, \gamma_1^q, \ldots, \gamma_d^q]^\top \in \mathbb{R}^{2d-1}$, sachant que β_d^q ne fait pas explicitement partie des inconnues puisque $\beta_d^q = \gamma_d^q$.

Exercice 3 : modélisation par deux courbes de Bézier couplées

Établissez (sur papier) les expressions de la matrice $\mathbf{E}^q$ et du vecteur $\mathbf{F}^q$ de l'équation (9). Attention : cette étape demande à être réalisée avec soin.

Écrivez la fonction `moindres_carres_bis`, appelée par le script `exercice_3`, permettant de résoudre le système linéaire (9) en moindres carrés.

Exercice 4 : simulation de silhouettes par tirages aléatoires

L'analyse précédente permet de caractériser la silhouette d'une flamme par $2\,d-1$ paramètres. Pour simuler une silhouette, on modélise les distributions des abscisses des points de contrôle par des lois normales, puis on utilise les lois estimées pour procéder au tirage aléatoire de nouveaux points de contrôle (fonction `randn`).

Écrivez la fonction `estimation_lois_n`, appelée par le script `exercice_4`, qui permet d'estimer la moyenne et l'écart-type de chacun des points de contrôle, considéré comme une variable aléatoire, à partir des réalisations empiriques que constituent les données réelles.

Écrivez la fonction `simulation`, appelée par le script `exercice_4`, qui doit simuler une silhouette de flamme par tirage aléatoire de valeurs pour les paramètres $[\beta_1^q, \ldots, \beta_{d-1}^q, \gamma_1^q, \ldots, \gamma_d^q]$, en utilisant les paramètres de lois normales précédemment estimés.

Lancez enfin le script `sequence_flammes`, qui montre un exemple d'application de l'estimation de paramètres à la synthèse d'une séquence d'images (*réalité virtuelle*).

Partie facultative : reformulation de la validation croisée

Vous avez étudié en cours la régression linéaire du type $y_j = \beta\,\mathbf{x}_j$, où $\beta \in \mathbb{R}^{1\times p}$ est le vecteur des paramètres du modèle et $\mathbf{x}_j \in \mathbb{R}^p$, pour $j \in \{1, \ldots, n_{\text{app}}\}$. En définissant :

$$\mathbf{Y} = \left[y_1, \ldots, y_{n_{\text{app}}}\right]^\top \in \mathbb{R}^{n_{\text{app}}} \quad \text{et} \quad \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\text{app}}}]^\top \in \mathbb{R}^{n_{\text{app}}\times p} \tag{10}$$

la forme matricielle de la régression linéaire s'écrit :

$$\mathbf{Y} = \mathbf{X}\,\beta^\top \tag{11}$$

Si $\widehat{y}_j = \widehat{\beta}\,\mathbf{x}_j$ désigne la prédiction du modèle appris sur l'ensemble des données d'apprentissage $\mathcal{D}_{\text{app}}$, ces prédictions s'écrivent, sous forme matricielle :

$$\widehat{\mathbf{Y}} = \mathbf{H}\,\mathbf{Y} \tag{12}$$

où $\widehat{\mathbf{Y}} = \left[\widehat{y}_1, \ldots, \widehat{y}_{n_{\text{app}}}\right]^\top \in \mathbb{R}^{n_{\text{app}}}$, et où $\mathbf{H} \in \mathbb{R}^{n_{\text{app}}\times n_{\text{app}}}$, appelée « matrice chapeau » (*hat*), a pour expression :

$$\mathbf{H} = \mathbf{X}\,(\mathbf{X}^\top\mathbf{X})^{-1}\,\mathbf{X}^\top \tag{13}$$

Il a été vu en cours que cette matrice permet de calculer autrement la validation croisée VC :

$$VC = \frac{1}{n_{\text{app}}} \sum_{j=1}^{n_{\text{app}}} \left[\frac{y_j - \widehat{y}_j}{1 - \mathbf{H}(j,j)}\right]^2 \tag{14}$$

où $\mathbf{H}(j,j)$ désigne le $j^{\text{ème}}$ élément diagonal de $\mathbf{H}$. Or, le calcul de l'expression (14) de VC est beaucoup moins coûteux que celui de (8), dans la mesure où il ne nécessite pas d'estimer les n_{app} vecteurs de paramètres $\widehat{\beta}_j$.

Exercice 5 : nouvelle estimation de l'hyper-paramètre λ

Par identification du problème générique (11) avec l'écriture $\mathbf{A}\,\beta^\top = \mathbf{B}$ du TP1, on voit que la matrice chapeau correspondant à l'estimation des paramètres d'une courbe de Bézier s'écrit :

$$\mathbf{H} = \mathbf{A}\,(\mathbf{A}^\top\mathbf{A})^{-1}\,\mathbf{A}^\top \tag{15}$$

En lieu et place des prédictions $\widehat{\mathbf{Y}} = \mathbf{H}\,\mathbf{Y}$ utilisées dans le modèle de régression linéaire simple, avec le modèle de régression écrêtée utilisé dans ce TP, on peut prédire vectoriellement les données par une expression similaire :

$$\widehat{\mathbf{Y}} = \mathbf{S}\,\mathbf{Y} \tag{16}$$

où $\mathbf{H}$ est remplacée par la « matrice de lissage » $\mathbf{S}$ (*smoothing matrix*), dont l'écriture découle de (7) :

$$\mathbf{S} = \mathbf{A}\,(\mathbf{A}^\top\mathbf{A} + \lambda\,\mathbf{I}_{d-1})^{-1}\,\mathbf{A}^\top \tag{17}$$

L'expression de VC qui se déduit de (14) est donc :

$$VC = \frac{1}{n_{\text{app}}}\sum_{j=1}^{n_{\text{app}}}\left[\frac{y_j - f(\beta_0^*, \widehat{\beta}, \beta_d^*, x_j)}{1 - \mathbf{S}(j,j)}\right]^2 \tag{18}$$

Faites une copie de la fonction `calcul_VC_bis`, de nom `calcul_VC_ter`, que vous modifierez de manière à calculer l'expression (18) de VC. Le script `exercice_5`, qui appelle cette nouvelle fonction, doit permettre d'accélérer significativement le calcul de VC. Quelle est l'ordre de grandeur de cette accélération ?