Travaux pratiques 3 : Régression

Le but de ce TP est de traiter du problème suivant : étant donnés n points (x_i, y_i) dans un plan (les données), comment trouver une fonction ou une courbe f dont le graphe approxime bien de ces points ?

En équations, on a

$$y_i = f(x_i) + \epsilon_i$$

où ϵ_i est l'erreur que l'on fait en approximant y_i par $f(x_i)$. On aimerait déterminer une fonction f qui minimise l'erreur totale.

Ainsi posé, le problème n'est pas bien défini. En effet il existe une infinité de courbes passant exactement par ces n points, et ayant par conséquent une erreur nulle. Il faut donc donner d'autres informations concernant la courbe f : par exemple définir ce que l'on entend par "approximer" et aussi préciser les propriétés de f. Typiquement, on aimerait choisir f parmi une famille de fonctions (par exemple des fonctions polynomiales de degré donné, périodiques, gaussiennes, etc...). On appelle ce problème *régression* ou *fitting*. Matlab fournit un certain nombre d'outils puissants pour l'analyse de données, que nous allons introduire dans ce TP.

1 Importer des données

Matlab permet d'importer des données contenues dans des fichiers de données (*.dat). Pour ceci, cliquez sur *Home* → *Import Data* et choisissez le fichier de données. Alternativement, vous pouvez choisir le répertoire contenant les données dans la fenêtre *Current folder* et double cliquer sur le fichier correspondant. Un assistant s'ouvre alors, qui permet de créer dans le Workspace des vecteurs ou des matrices contenant ces données.

Testez l'importation de données en téléchargeant le fichier "data1.dat" sur Moodle et en l'important dans une variable matricielle de taille $n \times 2$ appellée `data1`, où n est le nombre de mesures. Notez que pour importer le fichier de données comme variable matricielle, vous devez choisir l'option *Numeric matrix* au lieu de l'option par défaut *Column vectors*.

2 Régression linéaire : la méthode des moindres carrés

2.1 Moindre carrés

Définissons une quantité mesurant dans quelle mesure une fonction approche bien notre ensemble de points. Étant donné les n points (x_i, y_i), on va chercher une fonction f qui minimise

$$\Omega(f) = \sum_{i=1}^{n} (y_i - f(x_i))^2 \geq 0, \tag{1}$$

c'est-à-dire la somme des carrés des distances entre les données et les valeurs de f. Il est clair que si aucune spécification n'est requise sur f, alors il existe une infinité de fonctions annulant Ω pour un nombre fini de points donnés. Il est donc indispensable de définir le type de fonctions approximant nos données.

2.2 Régression par droites affines

Considérons d'abord le cas où l'on demande que la courbe f soit une droite, $f(x) = ax+b$, $a, b \in \mathbb{R}$. Dans ce cas, on peut faire ce calcul explicitement. On réécrit d'abord (1)

$$\Omega(a, b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2. \tag{2}$$

Pour minimiser la fonction $\Omega(a, b)$ on doit trouver les paramètres a et b tels que

$$\frac{\partial \Omega}{\partial a} = \frac{\partial \Omega}{\partial b} = 0. \tag{3}$$

En calculant ces derivées explicitement, on obtient le système d'équations suivant :

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix}. \tag{4}$$

Exercice 1 Régression affine. Commencez par vérifier la formule (4).

1. Ecrivez une fonction `RegressionAffine` prenant comme paramètres d'entrée une matrice $n \times 2$ contenant les points (x_i, y_i) et retournant les paramètres a et b résolvant le système d'équations (4).
2. Cette fonction doit dessiner sur un même graphe la droite de régression déterminée par (a, b) ainsi que les données de départ.
   Indications :
   — N'oubliez pas d'inclure la commande `hold on` pour pouvoir dessiner plusieurs objets sur le même graphique.
   — Pour dessiner les données, fournissez l'argument supplémentaire `'ob'` à la commande `plot`, de manière à dessiner un cercle bleu pour chaque point de donnée.
   — Pour dessiner la droite, utilisez l'argument supplémentaire `'-r'` de manière à lier les points par une ligne rouge.
3. Une fois la fonction programmée, téléchargez les fichiers de données "data1.dat", "data2.dat", "data3.dat", "data4.dat" et "data5.dat" sur Moodle, importez-les dans Matlab et utilisez `RegressionAffine` sur ces données dans votre script. Que constatez-vous en examinant vos graphiques ?
4. Pour déterminer le minimum global d'une fonction, il n'est en général pas suffisant de calculer simplement les points critiques (c'est-à-dire un point où la dérivée s'annule). Un point critique peut aussi correspondre à un maximum local ou global, à un minimum local, à un point d'inflexion, ou à un point-selle. Dans notre cas, pourquoi n'est-il pas nécessaire d'être plus prudent et de vérifier que le point critique que l'on a trouvé est bien un minimum ?
5. Il existe une façon plus simple d'implementer la fonction `RegressionAffine`. En effet, Matlab implémente la méthode des moindre carrés pour la régression linéaire par l'opération `[\]`, déjà utilisée au TP1. On rappelle que `X \Y` est une matrice A telle que $XA = Y$. Dans le cas d'une régression, ce système n'est pas résoluble exactement (il n'y a que deux variables indépendentes, a et b, et une contrainte pour chaque paire (x_i, y_i)). Au lieu de retourner une erreur, Matlab retourne la meilleure approximation selon la méthode des moindre carrés.

Implémentez une fonction `RegressionAffine2` faisant le même travail que `RegressionAffine`, mais utilisant `[\]`. Vérifiez que vous obtenez les mêmes droites de régression sur les données.

Indication : Une petite difficulté est qu'il faut transformer notre régression affine en un problème linéaire. Ceci est possible en ajoutant une colonne à la matrice X ne comportant que des 1s. Quelle forme a la matrice A ?

2.3 Régression par cercles

Il arrive que des problèmes apparemment non linéaires puissent être néanmoins linéarisés, comme on va le voir dans l'exemple suivant.

Dans les fichiers "data4.dat" et "data5.dat", on a vu des dispositions de points ressemblant plus à des ellipses ou à des cercles qu'à une droite. Notre prochain but est de développer une méthode linéaire de régression pour trouver un cercle approximant au mieux un nuage de points.

Un cercle est déterminé par son centre de coordonnées (a, b) et son rayon r. Un point P de coordonnées (x, y) est sur le cercle si seulement si $(x-a)^2 + (y-b)^2 = r^2$, ou de manière équivalente, si

$$\Phi(a, b, r, x, y) := ((x-a)^2 + (y-b)^2 - r^2)^2 \tag{5}$$

s'annule. On remarque que la fonction Φ est positive, différentiable et qu'elle s'annule sur les points du cercle, il est donc naturel d'essayer de la minimiser par rapport aux paramètres (a, b, r).

Si on a n points de coordonnées (x_i, y_i) pour i variant de 1 à n, on pose

$$\begin{aligned}
\Omega(a, b, r) &= \sum_{i=1}^{n} \Phi(a, b, r, x_i, y_i) && (6)\\
&= \sum_{i=1}^{n} ((x_i - a)^2 + (y_i - b)^2 - r^2)^2 && (7)\\
&= \sum_{i=1}^{n} (x_i^2 - 2\,x_i\,a + a^2 + y_i^2 - 2\,y_i\,b + b^2 - r^2)^2 && (8)\\
&= \sum_{i=1}^{n} (x_i^2 - 2\,x_i\,a + y_i^2 - 2\,y_i\,b + a^2 + b^2 - r^2)^2\ . && (9)
\end{aligned}$$

L'astuce : On pose $u = a^2 + b^2 - r^2$ et on considère une nouvelle fonction en les trois variables indépendantes a, b et u définie par

$$\Omega(a, b, u) = \sum_{i=1}^{n} (x_i^2 - 2\,x_i\,a + y_i^2 - 2\,y_i\,b + u)^2 \tag{10}$$

On détermine a, b et u en minimisant cette fonction, et obtient le système linéaire

$$\frac{\partial \Omega}{\partial a} = 0, \qquad \frac{\partial \Omega}{\partial b} = 0, \qquad \frac{\partial \Omega}{\partial u} = 0 \tag{11}$$

que l'on résout comme précédemment.

Exercice 2 Régression circulaire.

1. Écrivez une fonction `RegressionCirculaire` qui prend comme paramètres d'entrée une matrice $n \times 2$ contenant les points (x_i, y_i) et retourne les paramètres a, b et r décrivant le cercle.
2. Faîtes en sorte que `RegressionCirculaire` dessine sur un même graphe le cercle de régression ainsi que les données de départ.
3. Testez `RegressionCirculaire` dans votre script sur les données contenues dans les fichiers "data4.dat" et "data5.dat".

Lorsque l'on demande que la fonction f soit un polynôme en x et y de degré donné, on peut procéder de manière similaire aux deux exemples précédents. Par exemple, au lieu d'utiliser des cercles comme dans l'exemple simple ci-dessus, il est parfois judicieux d'utiliser des coniques (ellipse, parabole, hyperbole), dont l'équation est donnée par un polynôme général de degré 2 :

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \ , A, B, C, D, E, F \in \mathbb{R} \ . \tag{12}$$

3 Régression non-linéaire : `nlinfit`

Pour les cas où la fonction f dans (1) est quelconque, Matlab fournit un outil de régression puissant : `nlinfit`. Il s'agit de trouver parmi une famille de fonctions $\{f_a\}$, où a est un paramètre, celle approchant le mieux les données. On va implémenter le cas où la famille $\{f_a\}$ est une famille de courbes gaussiennes caractérisées par leur moyenne, écart-type et amplitude.

3.1 Régression gaussiennes sur des données

Si l'on effectue un grand nombre de mesures d'une certaine quantité et si les mesures sont indépendantes les unes des autres, alors ces valeurs se distribuent autour de la valeur moyenne selon la *loi normale* ou *gaussienne*, dont la fonction de répartition est donnée par

$$P(x) = A \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right). \tag{13}$$

μ est la moyenne de la gaussienne, σ est l'écart-type (correspondant à l'"étalement" de la gaussienne), et $A := P(\mu)$ est l'amplitude (voir figure 3.1).

Supposons que l'on ait notre ensemble de mesures $\{m_k\}_{k=1,\dots,N}$, donné par un ensemble de N points distribués le long de l'axe des x. On va commencer par construire un *histogramme* de la façon suivante. On choisit d'abord un intervalle I assez grand pour contenir toutes nos données, c'est-à-dire tous les points $\{m_k\}$. On découpe I en n intervalles ("bins") $\{I_i\}_{i=1,\dots,n}$. On note x_i pour le centre de l'intervalle I_i et on compte le nombre y_i de points dans chaque bin I_i. On obtient ainsi un ensemble de points $\{(x_i, y_i)\}_{i=1,\dots,n}$ dans le plan, sur lequel on peut appliquer une régression.

Comme on s'attend à ce que la distribution de nos points suive une loi normale, on va appliquer une régression par rapport à une famille de courbes gaussiennes

$$f_{A,\mu,\sigma}(x) = A \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right) \tag{14}$$

parametrée par A, μ et σ. C'est typiquement un cas où l'on peut utiliser la fonction `nlinfit`.

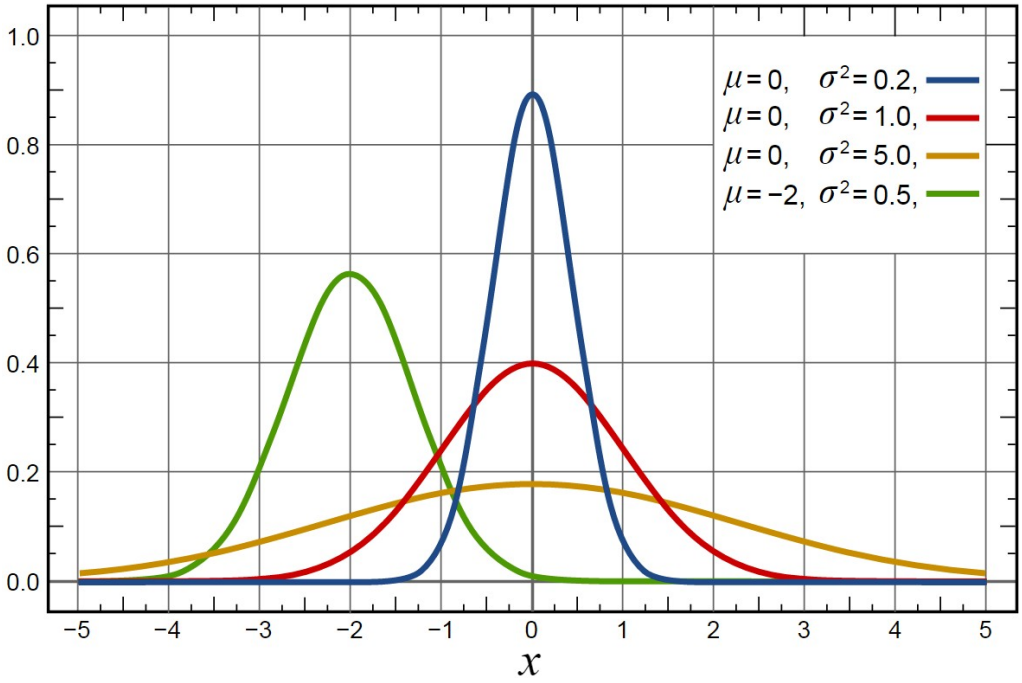

FIGURE 1 – Quelques courbes gaussiennes, dont les amplitudes ont été normalisées à $A = \frac{1}{\sigma\sqrt{2\pi}}$.

Exercice 3 Régression gaussienne. Le fichier "datag.dat" contient environ 10^4 mesures prises au LEP de la masse de la particule W. Téléchargez-le et importez-le dans Matlab.

1. A partir de ces données, construisez un histogramme `h` en utilisant la fonction `histogram` décrite dans la Section 3.2 ci-dessous.
2. `histogram` nous fournit un vecteur `h.BinEdges` contenant les points de séparation des bins. Construisez un vecteur `CentresBins` contenant le centre de chaque bin. Plottez le vecteur de valeurs de l'histogramme, `h.Values`, en fonction de `CentresBins` et comparez à l'histogramme pour vérifier que tout fonctionne.
3. Créez une fonction `FamGauss` qui admet comme variables un vecteur tridimensionel `a` et une variable réelle `x`. `FamGauss` doit retourner la valeur au point `x` de la gaussienne d'amplitude `a(1)`, de moyenne `a(2)` et d'écart-type `a(3)`. (Comme précédemment, `FamGauss` doit être définie dans votre script, pas dans un fichier séparé.)
4. Effectuez une régression par rapport à la famille de fonctions gaussiennes `FamGauss` sur le nuage de points produit par l'histogramme. Utilisez pour ceci la fonction `nlinfit` décrite dans la Section 3.3. Donnez l'amplitude, la moyenne et l'écart-type de la courbe de régression.
5. Plottez cette gaussienne et vérifiez qu'elle approxime bien l'histogramme. Pour un résultat esthétique, ajoutez les trois paramètres supplémentaires `'k', 'Linewidth', 2` à la fonction `plot` (pour dessiner la courbe gaussienne en noir avec une largeur de trait de deux points).
6. Répétez l'opération avec un nombre de bins différent. Est-ce que les paramètres de la courbe de régression ont changé ? Si oui, lesquels et pourquoi ?

3.2 La fonction `histogram`

Matlab fournit la fonction `histogram`, qui permet de construire un histogramme à partir de données. `histogram` s'utilise très simplement de la manière suivante :

```
h = histogram(m)
```

où `m` est le vecteur contenant les données.

La variable retournée `h` est une variable de type histogram, qui contient toutes les informations à propos de l'histogramme. Par exemple, le nombre de mesures dans chaque bins est contenu dans le vecteur `h.Values` (de taille n). Les séparations entre les bins sont contenues dans un vecteur `h.BinEdges` (de taille $n+1$). Le nombre, la taille et le placement des bins est effectué automatiquement, mais peut être ajusté manuellement si nécessaire. Par exemple, on peut créer un histogramme avec 50 bins par la commande

```
h = histogram(m,50)
```

Pour plus d'informations, cf. `help histogram`.

3.3 Utilisation de `nlinfit`

La procédure pour passer la famille de fonctions $\{f_a\}$ comme un paramètre dans `nlinfit` est la suivante.

1. On crée une nouvelle fonction Matlab (par exemple `FuncFit`, enregistrée dans le fichier `FuncFit.m`), qui va encoder notre famille de fonctions. `FuncFit` prend comme entrée le paramètre a paramétrisant la famille de fonctions f_a et la variable x (dans cet ordre !). Elle retourne $f_a(x)$ en utilisant la valeur de a passée en paramètre. En d'autre termes, on considère la famille de fonctions paramétrée par a comme une unique fonction dépendant de a et de x. a et x peuvent bien sûr être des variables vectorielles.

   Par exemple, si la famille de fonctions à minimiser est $\{f_{A,B,\omega} | A, B, \omega \in \mathbb{R}\}$ et

   $$f_{A,B,\omega}(x) = A\sin(\omega x) + B\cos(\omega x), \tag{15}$$

   on écrit une fonction qui prend comme entrée le vecteur de paramètres (A, B, ω) ainsi que la variable x, et qui retourne $f_{A,B,\omega}(x)$.

2. Dans le script principal, on utilise la fonction `nlinfit` de la façon suivante :

   ```
   pf = nlinfit(x,y,@FuncFit,p0).
   ```

   — (x, y) sont deux vecteurs contenant les deux coordonnées des points sur lesquels on aimerait effectuer la régression.
   — `FuncFit` est la famille de fonctions par rapport à laquelle on aimerait effectuer la régression, encodée comme expliqué plus haut.
   — `p0` contient une valeur de départ des paramètres, sur lequel l'algorithme se base pour trouver la valeur optimale des paramètres. Il est nécessaire de fournir ces valeurs initiales. Idéalement, le résultat devrait être indépendant de celles-ci, dans la pratique, ça n'est pas toujours le cas...
   — La variable retournée `pf` contient les valeurs des paramètres qui minimisent (1).

   Pour plus d'informations, voir `help nlinfit`.