Méthodes de Monte-Carlo et algorithmes stochastiques – TD/TP 5

Régression, arrêt optimal et Robbins–Monro

M2 Finance – Université Gustave Eiffel

Objectifs

Cette feuille regroupe les deux thèmes de la dernière séance : **régression et arrêt optimal**, puis **approximation stochastique**. Les six exercices sont indépendants. Les exercices 1–4 sont des calculs de TD et les exercices 5–6 sont deux TP Python courts.

Exercice 1 – Espérance conditionnelle et choix des fonctions de régression

Soit $X \sim \mathcal{U}([-1,1])$ et soit ε indépendante de X, centrée, de variance $1/4$. On observe

$$Y = X^3 + \varepsilon.$$

On veut prédire Y en ne connaissant que X. On rappelle, pour tout $j \geq 0$, $\mathbb{E}(X^{2j+1}) = 0$ et $\mathbb{E}(X^{2j}) = 1/(2j+1)$.

1) Calculer $m(x) = \mathbb{E}(Y \mid X = x)$. Montrer que, pour toute fonction g telle que $g(X)$ soit de carré intégrable,
$$\mathbb{E}[(Y - g(X))^2] = \frac{1}{4} + \mathbb{E}[(X^3 - g(X))^2].$$

2) Parmi les fonctions affines $g(x) = a+bx$, déterminer la meilleure au sens de l'erreur quadratique. Justifier les deux équations qui déterminent a, b.

3) Calculer l'erreur quadratique du meilleur prédicteur affine et séparer **erreur due au bruit** et **erreur d'approximation**.

4) On ajoute la fonction x^2, donc $g(x) = a + bx + cx^2$. Sans résoudre un système 3×3, expliquer pourquoi le coefficient optimal c vaut encore 0. Proposer une fonction supplémentaire qui, elle, supprimerait l'erreur d'approximation.

Exercice 2 – Arrêt optimal sur une chaîne à deux états

Un système occupe l'état A ou B aux dates $0, 1, 2$. La matrice de transition, identique d'une date à la suivante, est

$$P = \begin{pmatrix} 3/4 & 1/4 \\ 1/2 & 1/2 \end{pmatrix} \quad \text{(lignes : état actuel ; colonnes : état suivant).}$$

Si l'on s'arrête à la date k, on reçoit le montant $h_k(X_k)$ ci-dessous. Le facteur d'actualisation entre deux dates vaut $\delta = 0{,}9$.

	$k=0$	$k=1$	$k=2$
$h_k(A)$	4	6	8
$h_k(B)$	5	3	1

On doit s'arrêter au plus tard à la date 2. On note $V_k(x)$ la valeur optimale *à la date* k, exprimée en unités monétaires de cette date, lorsque l'état est x.

1) Justifier que $V_2(A) = 8$ et $V_2(B) = 1$.
2) Calculer $C_1(x) = \delta\,\mathbb{E}[V_2(X_2) \mid X_1 = x]$, pour $x = A, B$, puis déterminer la règle optimale à la date 1 et les valeurs $V_1(A), V_1(B)$.
3) Calculer $C_0(A)$ et $C_0(B)$, puis les valeurs $V_0(A), V_0(B)$. Dans quel état faut-il s'arrêter immédiatement ?
4) Sachant $X_0 = A$, comparer $V_0(A)$ aux gains moyens des trois stratégies qui fixent dès le départ la date d'arrêt 0, 1 ou 2 (sans utiliser l'état observé aux dates intermédiaires).

Exercice 3 – Une régression Longstaff–Schwartz que l'on peut faire à la main

On dispose des **six trajectoires simulées** suivantes. À la date 1, l'état observé est X_1 ; si l'on continue jusqu'à la date 2, on reçoit H_2. Si l'on s'arrête à la date 1, on reçoit toujours 5. Tous les gains sont déjà exprimés dans la même unité temporelle.

trajectoire i	1	2	3	4	5	6
$X_1^{(i)}$	0	0	1	1	2	2
$H_2^{(i)}$	2	4	6	8	3	5

1) Pour la régression quadratique $\widehat{c}_1(x) = a + bx + cx^2$, montrer que les valeurs ajustées en $x = 0, 1, 2$ sont les moyennes des gains futurs des observations correspondantes. En déduire a, b, c.
2) Quelle règle d'arrêt obtient-on en comparant $\widehat{c}_1(X_1)$ au gain immédiat 5 ? Quel est le gain moyen *sur ces six trajectoires d'apprentissage* ?
3) Refaire la régression en n'autorisant que $a + bx$. *Indication :* par symétrie des trois groupes, la pente est $(\overline{H}_2(2) - \overline{H}_2(0))/2$. Comparer les décisions obtenues avec les deux familles.
4) *(complément)* Pourquoi le gain moyen calculé sur les six trajectoires utilisées pour ajuster la règle ne suffit-il pas pour juger sa performance ? Comment procéder correctement ?

Exercice 4 – Robbins–Monro pour une équation non linéaire

Soit $U \sim \mathcal{U}([0, 2])$. Pour $\theta \in [0, 2]$, on considère

$$F(\theta) = \mathbb{E}[(\theta - U)_+], \qquad (t)_+ = \max(t, 0).$$

On cherche le paramètre θ_* vérifiant $F(\theta_*) = 1/4$. Pour chaque valeur proposée de θ, on peut simuler un nouveau U et observer $(\theta - U)_+$.

1) Calculer explicitement $F(\theta)$ pour $\theta \in [0, 2]$ et en déduire θ_*. Vérifier que F est croissante.
2) Écrire l'algorithme de Robbins–Monro projeté sur $[0, 2]$ avec les pas $\gamma_n = 0{,}8/n$. Justifier que l'observation utilisée est **sans biais conditionnellement** à l'itéré courant.
3) Calculer les trois premiers itérés avec $\theta_0 = 1{,}5$ et les réalisations successives $U_1 = 0{,}5$, $U_2 = 1{,}6$, $U_3 = 0{,}2$. Montrer chaque correction.
4) Parmi les suites $\gamma_n = n^{-0{,}5}$, $n^{-0{,}7}$, n^{-1} et $n^{-1{,}2}$, lesquelles vérifient les deux conditions classiques $\sum \gamma_n = +\infty$ et $\sum \gamma_n^2 < +\infty$? Expliquer brièvement les deux rôles.

Exercice 5 – TP Python : apprendre et tester une politique d'arrêt

On considère le modèle simulable

$$X_{k+1} = 0{,}6X_k + 0{,}8Z_{k+1}, \quad X_0 = 0, \quad h(x) = \big(1{,}8 - (x - 0{,}8)^2\big)_+, \quad k = 0, \dots, 5,$$

avec Z_k indépendantes de loi $\mathcal{N}(0,1)$. On peut s'arrêter aux dates $1, \dots, 6$; le gain en date 0 est $0{,}96^\tau h(X_\tau)$. Le bloc suivant **simule les trajectoires** ; le travail porte sur les régressions et la construction de la règle.

```
import numpy as np
M, delta = 6, 0.96
rng_train = np.random.default_rng(2026)
rng_test = np.random.default_rng(2027)

def paths(n, rng):
    x = np.zeros((n, M+1))
    for k in range(M):
        x[:, k+1] = 0.6*x[:, k] + 0.8*rng.normal(size=n)
    return x

def reward(x):
    return np.maximum(1.8-(x-0.8)**2, 0.)

def phi(x, degree):
    return np.column_stack([x**j for j in range(degree+1)])

train = paths(8000, rng_train)
test = paths(30000, rng_test)
```

À programmer vous-même

Compléter uniquement les lignes `TODO`. Dans la phase d'apprentissage, `gain[i]` est le montant non actualisé obtenu sur la trajectoire i à la date `stop_time[i]`. Lors de la remontée au temps k, la réponse de régression est donc `delta**(stop_time-k) * gain`.

```
def fit_policy(sample, degree):
    n = len(sample)
    gain = reward(sample[:, M]).copy()
    stop_time = np.full(n, M, dtype=int)
    coeffs = {}
    for k in range(M-1, 0, -1):
        x = sample[:, k]
        y = ...                  # TODO : gain futur vu de k
        A = ...                  # TODO : fonctions de regression
        beta = ...               # TODO : np.linalg.lstsq
        c = ...                  # TODO : continuation estimee
        stop = ...               # TODO : comparaison immediate/continuation
        gain[stop] = reward(x[stop])
        stop_time[stop] = k
        coeffs[k] = beta
    return coeffs

def evaluate(sample, coeffs, degree):
    n = len(sample)
    stopped = np.zeros(n, dtype=bool)
    payoff0 = np.zeros(n)
    for k in range(1, M):
        c = ...                  # TODO : appliquer coeffs[k]
        stop = ...               # TODO : uniquement si pas deja arrete
        payoff0[stop] = (delta**k)*reward(sample[stop, k])
        stopped[stop] = True
    payoff0[~stopped] = (delta**M)*reward(sample[~stopped, M])
    return payoff0.mean()
```

1) Compléter les deux fonctions et expliquer pourquoi `fit_policy` remonte dans le temps tandis que `evaluate` l'avance.
2) Comparer sur le **même échantillon test** les degrés 1, 2 et 3. Comparer aussi aux stratégies « toujours arrêter à 1 » et « attendre jusqu'à 6 ».
3) *(complément)* Pourquoi n'est-il pas fiable de choisir le degré seulement en comparant les gains *sur l'échantillon d'apprentissage* ?

Exercice 6 – TP Python : l'effet des pas de Robbins–Monro

On reprend $F(\theta) = \mathbb{E}[(\theta - U)_+]$ de l'exercice 4. Compléter la fonction suivante pour résoudre numériquement $F(\theta) = 1/4$ sur $[0, 2]$.

```
def robbins_monro(n, gamma, seed):
    rng = np.random.default_rng(seed)
    theta = 1.5
    history = np.empty(n+1)
    history[0] = theta
    for k in range(1, n+1):
        u = rng.uniform(0., 2.)
        y = ...           # TODO : observation bruitee de F(theta)
        theta = ...       # TODO : correction et projection sur [0,2]
        history[k] = theta
    return history
```

1) Compléter les lignes `TODO`. Tester $n = 20000$ avec les pas $\gamma_k = 2/k$, $\gamma_k = 0{,}8/k^{0{,}7}$ et le pas constant $\gamma_k = 0{,}08$, en utilisant la même graine pour une comparaison plus facile.
2) Tracer les trajectoires de θ_k (échelle logarithmique en abscisse facultative) et comparer leurs **dernières valeurs** et leurs **fluctuations sur les 1000 derniers pas**. Quelle différence avec un pas constant ?

Idée à retenir

Dans ces deux TP, les diagnostics font partie du résultat : une politique doit être **évaluée hors des données d'apprentissage**, et une récurrence bruitée doit être jugée **sur sa stabilité**, pas seulement sur sa dernière valeur.