Stochastic Simulation
Professor: Prof. Fabio Nobile
Assistant: Eliott Van Dieren

Week of 2025-12-15 to 2025-12-19

Lab 14 of Thursday 18th December 2025

Exercice 1.

Consider a simple PERT network problem where

$$z(\theta) = \mathbb{E}[\max\{\theta X_1 + X_2, (1-\theta)X_3\}],$$

with $0 < \theta < 1$. The random variables X_1, X_2, and X_3 follow Erlang(2) distributions with the following properties with means 1,2, and 3, respectively. We consider the optimization problem

$$\min_{\theta \in [0,1]} z(\theta). \tag{1.1}$$

1) Verify that $z'(\theta) = \mathbb{E}[\frac{d}{d\theta}\max\{\theta X_1 + X_2, (1-\theta)X_3\}]$. Write pen and paper the estimators IPA and LR to approximate $z'(\theta)$.

2) Estimate $z'(\theta)$ using IPA and LR with 10^5 samples and for a grid $\theta \in \{0.1, 0.2, \dots, 0.9\}$. For each value of θ estimate the standard deviation of the estimators of $z'(\theta)$ and plot them as a function of θ.

3) Implement the stochastic gradient descent method to minimize $z(\theta)$. Use a step size of your choice and compute the approximated optimal $\widehat{\theta}^*$. Use both

   - SGD with a decreasing step size $\tau_k \propto \frac{1}{k}$ and a fixed sample size to estimate z' at each iteration;
   - SGD with a fixed step size τ and a geometrically increasing sample size to estimate z' at each iteration.
   - Using 0.6253 as a reference solution, plot the error $|\theta_k - \theta^*|$ as a function of the number of gradient evaluations.

Solution

1) We are given the function

$$z(\theta) = \mathbb{E}[\psi(\theta, X)], \quad X = (X_1, X_2, X_3),$$

where

$$\psi(\theta, x) = \max\{\theta x_1 + x_2, (1-\theta)x_3\}$$

To verify that $z'(\theta) = \mathbb{E}\Big[\frac{d}{d\theta}\max\{\theta X_1 + X_2, (1-\theta)X_3\}\Big]$, we notice that, given any $\theta_0 \in (0,1)$, $\theta \mapsto \psi(\theta, x)$ is differentiable in θ_0 for all $x \in \mathbb{R}^3 \setminus P$ where $P = \{x \in \mathbb{R}^3 : \theta_0 x_1 + x_2 =$

$(1-\theta_0)x_3\}$ which is a plane in $\mathbb{R}^3$ and, hence, it is a zero measure set. This means that $\theta \mapsto \psi(\theta, x)$ is almost everywhere differentiable in θ_0. Furthermore,

$$|\psi(\theta_1, x) - \psi(\theta_2, x)| \leqslant |\theta_1 - \theta_2|(x_1 + x_3)$$

and $\mathbb{E}[|X_1 + X_3|] < \infty$ since the Erlang(2) distribution has finite first moment. This entails that we can exchange the derivative and the expectations, and the IPA estimator for z' is well defined.

The derivative of the maximum function is defined piecewise as:

$$\frac{d}{d\theta}\max\{\theta x_1 + x_2, (1-\theta)x_3\} = x_1 \mathbb{1}_{\{\theta x_1 + x_2 > (1-\theta)x_3\}} - x_3 \mathbb{1}_{\{\theta x_1 + x_2 < (1-\theta)x_3\}}$$

Thus,

$$z'(\theta) = \mathbb{E}\big[X_1 \mathbb{1}_{\{\theta X_1 + X_2 > (1-\theta)X_3\}}\big] - \mathbb{E}\big[X_3 \mathbb{1}_{\{\theta X_1 + X_2 < (1-\theta)X_3\}}\big].$$

The IPA Estimator is then a Monte Carlo approximation of the above.

In order to compute the Likelihood Ratio (LR) estimator, let us introduce the changes of variables $\tilde{X}_1 = \theta X_1$ and $\tilde{X}_3 = (1-\theta)X_3$. Note that the probability density functions of $\tilde{X}_1$ and $\tilde{X}_3$ are

$$f^1_\theta(\tilde{x}) = \frac{1}{\theta} f\Big(\frac{\tilde{x}}{\theta}, 2, 1\Big), \quad f^3_\theta(\tilde{x}) = \frac{1}{(1-\theta)} f\Big(\frac{\tilde{x}}{(1-\theta)}; 2, 3\Big)$$

respectively, where $f(x; k, m)$ is the density of the Erlang(k) distribution with mean m. Let us note that, with $\beta = k/m$

$$\begin{aligned}
\frac{d}{d\theta}\log\Big(\frac{1}{\eta(\theta)} f\eta(\theta)\Big(\frac{\tilde{x}}{\eta(\theta)}; k, m\Big)\Big) &= \frac{d}{d\theta}\Big(-\log(\eta(\theta)) + \log\Big(f\Big(\frac{\tilde{x}}{\eta(\theta)}, k, m\Big)\Big)\Big) \\
&= \frac{d}{d\theta}\Big(-\log(\eta(\theta)) + \log\Big(\frac{\beta^k(\tilde{x}/\eta(\theta))^{k-1}e^{-\beta\tilde{x}/\theta}}{(k-1)!}\Big)\Big) \\
&= \frac{d}{d\theta}\Big(-\log(\eta(\theta)) + \Big((1-k)\log(\eta(\theta)) - \frac{\beta\tilde{x}}{\eta(\theta)}\Big)\Big) \\
&= \frac{d}{d\theta}\Big(-k\log(\eta(\theta)) - \frac{\beta\tilde{x}}{\eta(\theta)}\Big),
\end{aligned} \tag{1.2}$$

so that

$$\frac{d}{d\theta} f^1_\theta(\tilde{x}) = \frac{2\tilde{x}}{\theta^2} - \frac{2}{\theta}, \quad \frac{d}{d\theta} f^3_\theta(\tilde{x}) = \frac{2}{(1-\theta)} - \frac{2\tilde{x}}{3(1-\theta)^2} \tag{1.3}$$

Finally,

$$z'(\theta) = \mathbb{E}\Big[\max\{\tilde{X}_1 + X_2, \tilde{X}_3\}\frac{d}{d\theta}\big(\log(f^1_\theta(\tilde{X}_1)) + \log(f^3_\theta(\tilde{X}_3))\big)\Big],$$

and the LR estimator is a Monte Carlo approximation of the above.

2) The results are shown in Figure 1. In this case IPA outperforms the LR method, especially near the boundary of the interval $(0, 1)$. Indeed, the latter relies on the computation of the derivatives in (1.3) which explode as $\theta \to 0^+$ and $\theta \to 1^-$.

3) The results are shown in Figure 2. Note that on the x axis we plotted the total number of derivative evaluations to fairly compare the two methods in terms of their computational cost. In this case, they show a similar performance.

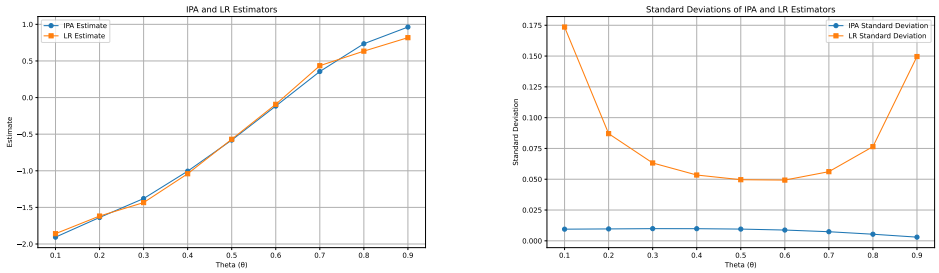

Figure 1: Left: IPA and LR estimates. Right: IPA and LR standard deviations.

Python code:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy.stats import gamma

# Define the parameters of the problem
np.random.seed(42)


def sample_erlang(k, mean, size):
    scale = mean / k
    return gamma.rvs(a=k, scale=scale, size=size)


def pdf(x, k, mean):
    scale = mean / k
    return gamma.pdf(x, a=k, scale=scale)


def dpdf(x, k, mean):
    beta = k / mean
    coeff = (beta**k) / scipy.special.gamma(k)
    term1 = (k - 1) * x ** (k - 2) * np.exp(-beta * x)
    term2 = beta * x ** (k - 1) * np.exp(-beta * x)
    return coeff * (term1 - term2)


def der_score(x_1, x_3, theta):
    dlogf1 = 2 * x_1 / (theta**2) - 2 / theta
    dlogf3 = -(2 * x_3 / (3 * (1 - theta) ** 2) - 2 / (1 - theta))
    return dlogf1 + dlogf3

# Number of samples for estimation
n_samples = int(1e5)

# Define the grid of \theta values
theta_grid = np.linspace(0.1, 0.9, 9)

# Placeholder to store results
ipa_estimates = []
lr_estimates = []
std_ipa = []
std_lr = []

# Simulation for IPA and LR estimators
```

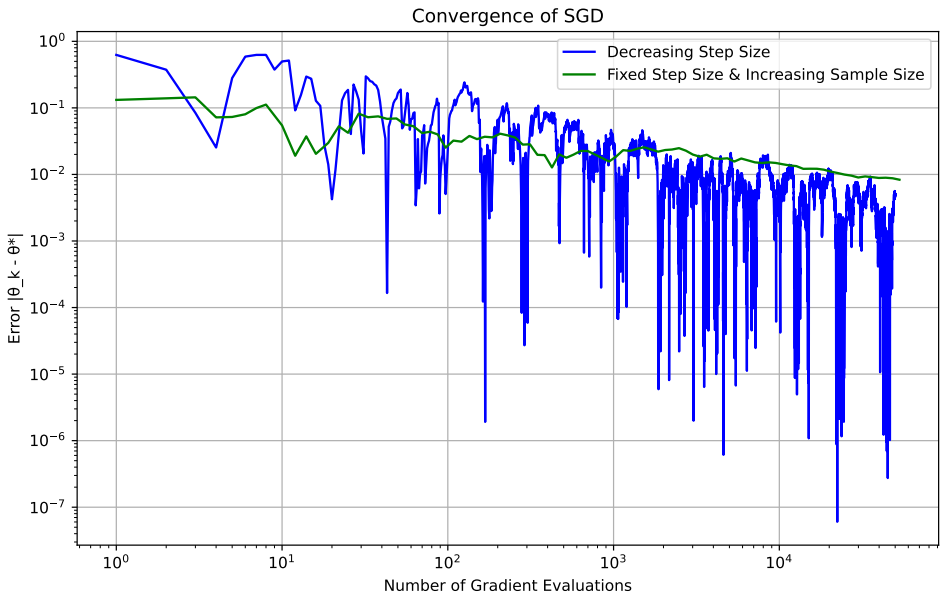

Figure 2: SGD convergence in the cases of decreasing stepsize with fixed samples size and fixed step size and increasing sample size.

```
for theta in theta_grid:
    X1 = sample_erlang(2, 1, n_samples)
    X2 = sample_erlang(2, 2, n_samples)
    X3 = sample_erlang(2, 3, n_samples)

    # Evaluate the max functions
    path1 = theta * X1 + X2
    path2 = (1 - theta) * X3
    max_path = np.maximum(path1, path2)

    # IPA estimator
    d_max_path = np.where(path1 > path2, X1, -X3)
    ipa_estimates.append(np.mean(d_max_path))
    std_ipa.append(np.std(d_max_path) / np.sqrt(n_samples))

    # LR estimator
    tX1 = theta * X1
    tX3 = (1 - theta) * X3
    lr_grad = max_path * der_score(tX1, tX3, theta)
    lr_estimates.append(np.mean(lr_grad))
    std_lr.append(np.std(lr_grad) / np.sqrt(n_samples))

# Plot estimates
plt.figure(figsize=(10, 6))
plt.plot(theta_grid, ipa_estimates, label="IPA Estimate", marker="o")
plt.plot(theta_grid, lr_estimates, label="LR Estimate", marker="s")
plt.xlabel("Theta (\u03B8)")
plt.ylabel("Estimate")
plt.title("IPA and LR Estimators")
plt.legend()
```

```
plt.grid()
plt.show()

# Plot standard deviations
plt.figure(figsize=(10, 6))
plt.plot(theta_grid, std_ipa, label="IPA Standard Deviation", marker="o")
plt.plot(theta_grid, std_lr, label="LR Standard Deviation", marker="s")
plt.xlabel("Theta (\u03B8)")
plt.ylabel("Standard Deviation")
plt.title("Standard Deviations of IPA and LR Estimators")
plt.legend()
plt.grid()
plt.show()


# Stochastic Gradient Descent (SGD)
true_theta_star = 0.6253

# SGD with decreasing step size
sgd_steps_decreasing = []
theta_sgd_decreasing = 0.5  # Initial theta
errors_decreasing = []

max_iter_decreasing = 50000
n_samples = int(1.0e0)
for k in range(max_iter_decreasing):
    X1 = sample_erlang(2, 1, n_samples)
    X2 = sample_erlang(2, 2, n_samples)
    X3 = sample_erlang(2, 3, n_samples)

    path1 = theta_sgd_decreasing * X1 + X2
    path2 = (1 - theta_sgd_decreasing) * X3

    d_max_path = np.where(path1 > path2, X1, -X3)
    grad_estimate = np.mean(d_max_path)

    step_size = 1 / (k + 1)  # Decreasing step size
    theta_sgd_decreasing -= step_size * grad_estimate
    theta_sgd_decreasing = max(
        0, min(1, theta_sgd_decreasing)
    )  # Projection to keep theta in [0, 1]

    sgd_steps_decreasing.append(theta_sgd_decreasing)
    errors_decreasing.append(abs(theta_sgd_decreasing - true_theta_star))

# SGD with fixed step size and geometrically increasing sample size
sgd_steps_fixed = []
theta_sgd_fixed = 0.5  # Initial theta
errors_fixed = []
sample_size = 1000  # Initial sample size
step_size_fixed = 0.01

delta = 1.1  # Multiplication factor at each iteration
max_iter_geometric = int(
    1 + np.log(1 + (delta - 1) * max_iter_decreasing) / np.log(delta)
)  # To keep the total number of samples equal
num_grad_evals_geometric = np.zeros(max_iter_geometric)
for k in range(max_iter_geometric):
    sample_size = int(n_samples * delta**k)  # Geometrically increasing sample size
    num_grad_evals_geometric[k] = sample_size
    X1 = sample_erlang(2, 1, sample_size)
    X2 = sample_erlang(2, 2, sample_size)
    X3 = sample_erlang(2, 3, sample_size)
```

```
    path1 = theta_sgd_fixed * X1 + X2
    path2 = (1 - theta_sgd_fixed) * X3

    d_max_path = np.where(path1 > path2, X1, -X3)
    grad_estimate = np.mean(d_max_path)

    theta_sgd_fixed -= step_size_fixed * grad_estimate
    theta_sgd_fixed = max(0, min(1, theta_sgd_fixed))  # Keep theta in [0, 1]

    sgd_steps_fixed.append(theta_sgd_fixed)
    errors_fixed.append(abs(theta_sgd_fixed - true_theta_star))

# Plot SGD progress for fixed step size and increasing sample size
num_grad_evals_decreasing = np.arange(1, max_iter_decreasing + 1) * n_samples
num_grad_evals_geometric = np.cumsum(num_grad_evals_geometric)
plt.figure(figsize=(10, 6))
plt.plot(
    num_grad_evals_decreasing,
    errors_decreasing,
    label="Decreasing Step Size",
    color="b",
)
plt.plot(
    num_grad_evals_geometric,
    errors_fixed,
    label="Fixed Step Size & Increasing Sample Size",
    color="g",
)
plt.xlabel("Number of Gradient Evaluations")
plt.ylabel("Error |\u03B8_k - \u03B8*|")
plt.title("Convergence of SGD")
plt.yscale("log")
plt.xscale("log")
plt.grid()
plt.legend()
plt.show()
```