

M2 MPRI - Programmation Probabiliste - WEB-PPL

Christine Tasson

8 décembre 2022

Webppl est un langage de programmation probabiliste basé sur Javascript. Pour programmer en ligne utiliser le site <http://webppl.org/> avec la documentation <https://webppl.readthedocs.io/en/master/>.

1 Premiers exemples.

Exercice 1.

Calculer la variable aléatoire associée au programme ci-dessous

```
1 var p = 0.1
2 var model = function() {
3   var x = sample(Uniform({a:0, b:1}))
4   return (x <= p)
5 }
6 var dist = Infer(model)
7 viz.auto(dist)
```

Exercice 2.

Implémenter en webppl puis, calculer la variable aléatoire engendrée par l'algorithme suivant :

- Lancer successivement deux fois une pièce biaisée de paramètres p
- Retourner 1 si la pièce a renvoyé pile puis face
- Retourner 0 si la pièce a renvoyé face puis pile
- dans les autres cas, recommencer

Expliquer le résultat.

Exercice 3. *Distribution uniforme sur un ensemble fini*

Pour modéliser une variable aléatoire uniforme sur $\{0, \dots, m\}$, on peut utiliser la décomposition binaire.

- Utiliser la variable aléatoire $\sum_{i=0}^{n-1} \mathcal{B}(0.5)2^i$ pour modéliser une distribution uniforme sur $\{0, \dots, 2^{n-1}\}$ en webppl.
- Utiliser du conditionnement pour obtenir une uniforme sur $\{0, \dots, m\}$.

2 Loi de Bayes

Exercice 4. *Faux Négatifs*

En 2020, le rendement diagnostique du frottis nasopharyngé-PCR Covid-19 pour une population dont 10% de la population est infectée par le Covid-19.

	Infected+	Infected-		Infected+	Infected-
Test+	56	9	Test+	83	9
Test-	44	891	Test-	17	891
Total	100	900	Total	100	900

- En utilisant la formule de Bayes, calculer le taux de faux négatifs pour les test PCR (Test-sachant que Infected+) dans les deux cas décrits ci-dessus.
- Écrire un programme probabiliste pour simuler ce résultat.

Exercice 5. *Militaires et Cannabis*

Une générale demande à un militaire s'il fume. Il lance une pièce

- si la pièce tombe sur face, alors il répond la vérité
- si la pièce tombe sur pile, alors il relance la pièce :
 - si la pièce tombe sur face, alors il répond oui
 - si la pièce tombe sur pile, alors il répond non

La générale qui ne sait pas combien de fois la pièce a été lancée. Avec 160 oui parmi 200, la proportion de fumeurs est de $p = 60\%$.

- Approcher la probabilité de fumer sachant que la réponse est oui.
- On a observé que le nombre de oui sachant le paramètre p suit une loi binomiale. Approcher la densité de probabilité du paramètre p représentant le nombre de fumeurs.

Exercice 6.

Écrire un modèle probabiliste permettant de trouver la droite la plus probable passant par les points $\{(0, 0), (1, 1), (2, 4), (3, 6)\}$

Références

- [1] G. Grimmett and D. Stirzaker, *One Thousand Exercises in Probability*, Oxford, 2001
- [2] Paul Hurst and Royer F. Cook and Douglas A. Ramsay *Assesing the prevalence of illicit drug use in the army*. U.S. Army, Research Institute for the Behavioral and Social Sciences, 1975
- [3] IOANNIS KOKKINAKIS, KEVIN SELBY, BERNARD FAVRAT, BLAISE GENTON et JACQUES CORNUZ *Performance du frottis nasopharyngé-PCR pour le diagnostic du Covid-19* Rev. Med Suisse 2020

Solution 1.

C'est une variable aléatoire discrète dont le support à deux valeurs. C'est donc une Bernoulli de paramètre p .

Solution 2.

On génère une pièce équilibrée à partir de deux pièces biaisées de même paramètre p .

```

1  var model=function() {
2  var x = sample(Bernoulli({p:0.2}))
3  var y = sample(Bernoulli({p:0.2}))
4  condition ((x & !y) | (!x & y))
5  return x
6  }
7  var dist = Infer(model)
8  viz.auto(dist)

```

Solution 3.

On utilise $\sum_{i=0}^{n-1} \text{coin}(0.5)2^i$ pour calculer une distribution uniforme sur $\{0, \dots, 2^n - 1\}$ que l'on restreint à $\{0, \dots, m\}$ par rejet.

```

1
2  var unif2 = function (n) {
3    var x = sample( Bernoulli( {p:0.5} ));
4    if (n == 0) {return x;}
5    else {return 2*unif2(n-1) + x;}}
6
7
8  var unif = function (m) {
9    var n = Math.ceil(Math.log(m)/Math.log(2));
10   var x = unif2(n);
11   if (x <= m){return x} else {return unif(m)}}
12
13
14  var unif = function (m) {
15    var n = Math.ceil(Math.log(m)/Math.log(2));
16    var x = unif2(n);
17    condition(x<=m)}
18
19  var dist = Infer(unif)

```

Solution 4.

```

1  var test = function() {
2    var Mpos = sample(Bernoulli({p:0.1}));
3    var Tneg = function(M){
4      if (M){
5        return sample(Bernoulli({p:0.17}));
6      }
7      else {
8        var y = 891/900
9        return sample(Bernoulli({ p:y })))
10     }
11   }
12   condition(Tneg(Mpos))
13   return Mpos
14 }
15 // Define and plot associated distribution
16 viz(Infer( {model:test} ))

```

Solution 5.

```
1  var canabis = function(){
2  var smoke = sample( Bernoulli( {p:0.6}));
3  var coin = sample( Bernoulli({ p:0.5 }));
4  var x = coin ? true : smoke
5  condition(x)
6  return smoke
7  }
8  viz(Infer(canabis))
9
10 var model = function(){
11   var p = sample(Uniform({a:0, b:1}))
12   var answer = function(){
13     var smoke = sample( Bernoulli( {p:p}));
14     var coin = sample( Bernoulli({ p:0.5 }));
15     var x = coin ? true : smoke
16     return x
17   }
18   condition(sum(repeat(200,answer))==160)
19   return p
20 }
21 viz(Infer(model))
22
23 var model = function(){
24   var p = sample(Uniform({a:0, b:1}))
25   var answers = repeat(200,function(){
26     var smoke = sample(Bernoulli( {p:p}));
27     var coin = sample(Bernoulli( {p:0.5}));
28     return coin ? true : smoke
29   })
30   // Compute proportion of Yes
31   var YP = sum(answers)/200
32   // Score with respect to a binomial of bias YP
33   // compared with data
34   observe(Binomial({n:200, p:YP}),160)
35   return p
36 }
37 // Infer with Importance Sampling
38 viz(Infer({method: 'SMC', particles:2000, model:model}))
```

Solution 6.

```
1  ///////////////////////////////////
2  // Linear Regression
3
4  var xs = [0, 1, 2, 3];
5  var ys = [0, 1, 4, 6];
6
7  var model = function() {
8    var m = gaussian(0, 2);
9    var b = gaussian(0, 2);
10    var sigma = gamma(1, 1);
11
12    var f = function(x) {
13      return m * x + b;
14    };
15
16    map2(
17      function(x, y) {
18        observe(Gaussian({mu: f(x), sigma: sigma}), y);
19      },
20      xs,
21      ys);
22
23    return {b:m};
```

```
24 }  
25  
26 viz.auto(Infer({method: 'MCMC', samples: 10000}, model));  
27  
28  
29  
30 var xs = [0, 1, 2, 3];  
31 var ys = [0, 1, 4, 6];  
32 var df = function(x){return 2*x-0;5}  
33 viz.line([0,3],[df(0),df(3)])  
34 viz.scatter(xs,ys)
```
