M2 MPRI - Programmation Probabiliste - WEB-PPL

Christine Tasson

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

```
var p = 0.1;
var model = function(){
    var x = sample(Uniform({a:0, b:1}));
    return (x <= p);
  };
var dist = Infer(model);
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 donné
- 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 $\{0, \ldots, m\}$, on peut utiliser la décomposition binaire.

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

2 Loi de Bayes

Exercice 4. *Faux Négatifs*

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

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

	Infected+	Infected-
Test+	83	9
Test-	17	891
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 Canabis*

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 a 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.

```
var model=function(){
    var x = sample(Bernoulli({p:0.2}));
    var y = sample(Bernoulli({p:0.2}));
    condition ((x & !y)|(!x & y));
    return x;
  };
var dist = Infer(model);
viz.auto(dist);
```

Solution 3.

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

```

var unif2 = function (n) {
    var x = sample( Bernoulli( {p:0.5} ));
    if (n == 0) {return x;}
    else {return 2*unif2(n-1) + x;}
  };


var unif = function (m) {
    var n = Math.ceil(Math.log(m)/Math.log(2));
    var x = unif2(n);
    if (x <= m){return x;} else {return unif(m);}
  };


var unif = function (m) {
    var n = Math.ceil(Math.log(m)/Math.log(2));
    var x = unif2(n);
    condition(x<=m);
    return x;
  };

var dist = Infer({model:function(){unif(10);}});

viz.auto(dist)
```

Solution 4.

```
var test = function(){
    var Mpos = sample(Bernoulli({p:0.1}));
    var Tneg = function(M){
        if (M){
            return sample(Bernoulli({p:0.17}));
          }
        else {
            var y = 891/900
            return sample(Bernoulli({ p:y }))
          }
      }
    condition(Tneg(Mpos))
    return Mpos
```

```
  }
// Define and plot associated distribution
viz(Infer( {model:test} ))
```

Solution 5.

```
var canabis = function(){
    var smoke = sample( Bernoulli( {p:0.6}));
    var coin = sample( Bernoulli({ p:0.5 }));
    var x = coin ? true : smoke;
    condition(x);
    return smoke;
  }
viz(Infer(canabis));

var model = function(){
    var p = sample(Uniform({a:0, b:1}));
    var answer = function(){
        var smoke = sample( Bernoulli( {p:p}));
        var coin = sample( Bernoulli({ p:0.5 }));
        var x = coin ? true : smoke;
        return x;
      }
    condition(sum(repeat(200,answer))==160);
    return p;
  }
viz(Infer(model))

var model = function(){
    var p = sample(Uniform({a:0, b:1}));
    var answers = repeat(200,
    function(){
        var smoke = sample(Bernoulli( {p:p}));
        var coin = sample(Bernoulli( {p:0.5}));
        return coin ? true : smoke;
      })
    // Compute proportion of Yes
    var YP = sum(answers)/200;
    // Score with respect to a binomial of bias YP
    // compared with data
    observe(Binomial({n:200, p:YP}),160);
    return p;
  }
// Infer with Importance Sampling
viz(Infer({method: 'SMC', particles:2000, model:model}));
```

Solution 6.

```
////////////////////
// Linear Regression

var xs = [0, 1, 2, 3];
var ys = [0, 1, 4, 6];

var model = function() {
    var m = gaussian(0, 2);
    var b = gaussian(0, 2);
    var sigma = gamma(1, 1);

    var f = function(x) {
        return m * x + b;
      };

    map2(
    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)
```
