Massachusetts Institute of Technology
6.867 Machine Learning, Fall 2006

Problem Set 1 Solutions

Section A (background questions)

1. Let's begin with a little math. Let us denote by P_n the probability that n people with randomly chosen birthdays (chosen uniformly from the year) have no common birthdays. Clearly $P_1 = 1$;

$$P_{n+1} = \frac{365 - n}{365} \cdot P_n;$$

after n days have been taken, the chance that we will get one of the remaining free days is $(365 - n)/365$. If we expand this, we get the following formula:

$$\begin{aligned} P_{n+1} &= \frac{365 - n}{365} \cdot P_n \\ &= \frac{365 - n}{365} \cdot \frac{365 - (n-1)}{365} \cdot P_{n-1} \\ &\vdots \\ &= \frac{365 - n}{365} \cdot \frac{365 - (n-1)}{365} \cdot \cdots \cdot \frac{365 - 1}{365} \cdot \frac{365}{365} \\ &= \frac{365!\,/(365 - (n+1))!}{365^{n+1}}. \end{aligned}$$

We can easily write a MATLAB script to evaluate this formula (of course, we want $1 - P_n$):

```
function P =birthday_prob(n)
·   P = 1;
·   for i = 2 : n
·   ·   P = P * (365 - i)/365;
·   end
·   P = 1 - P
```

We can also give an approximate solution by just sampling (and not doing any math). This is a much slower algorithm, but it works:

```
function P =birthday_prob(n)
·   P = 0;
·   samples = 1000;
·   for i = 1 : samples
·   ·   counts = zeros(365, 1);
·   ·   collision = 0;
·   ·   for j = 1 : n
·   ·   ·   bday = randint(1, 1, 365)+1;
·   ·   ·   if counts(bday) > 0
·   ·   ·   ·   collision = 1;
·   ·   ·   ·   break
·   ·   ·   end
·   ·   ·   counts(bday) = 1;
·   ·   end
·   ·   P = P + collision;
·   end
·   P = P/samples;
```

With either script, we can quickly find the smallest group that has a 50% chance of having a common birthday:

```
>> i = 2;
>> while birthday_prob(i) < 0.5
.    i = i + 1;
>> end
>> i
i =
     22
>> birthday_prob(22)
ans =
     0.5059
```

[Yes, we have assumed for simplicity that no-one is born on a leap year and that, by extension, all people are spherical. The case of non-spherical people born in the real world is left as an exercise for the reader.]

2. In order to solve these problems, we will consider the cumulative distribution function (cdf) of the X_i and of their maximum and minimum.

$$\begin{aligned}
F_{X_i}(x) &= \text{probability that } X < x \\
&= \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1 \end{cases} \\
F_{\max\{X_i\}}(x) &= \text{probability that } \max\{X_i\} < x \\
&= \text{probability that } X_i < x \text{ for all } i \\
&= \prod_{i=1}^{n} F_{X_i}(x) \qquad \text{[because the } X_i \text{ are independent]} \\
&= \begin{cases} 0 & \text{if } x < 0 \\ x^n & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1 \end{cases} \\
1 - F_{\min\{X_i\}}(x) &= \text{probability that } \min\{X_i\} \ge x \\
&= \text{probability that } X_i \ge x \text{ for all } i \\
&= \prod_{i=1}^{n} (1 - F_{X_i}(x)) \\
&= \begin{cases} 1 & \text{if } x < 0 \\ (1-x)^n & \text{if } 0 \le x \le 1 \\ 0 & \text{if } x > 1 \end{cases}
\end{aligned}$$

By taking a derivative, we can recover the density functions of all the variables:

$$\begin{aligned}
f_{\max\{X_i\}}(x) &= \begin{cases} nx^{n-1} & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases} \\
f_{\min\{X_i\}}(x) &= \begin{cases} n(1-x)^{n-1} & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

We can finally compute the expectations! The mathematically agile will note that there are many clever ways of evaluating this last integral, but we choose to remind you of the generally applicable methods over demonstrating cute mathematics.

$$
\begin{aligned}
E[\max\{X_i\}] &= \int_0^1 x f_{\max\{X_i\}}\, dx \\
&= \int_0^1 n x^n\, dx = \boxed{\frac{n}{n+1}}. \\
E[\min\{X_i\}] &= \int_0^1 x f_{\min\{X_i\}}\, dx \\
&= \int_0^1 n x (1-x)^{n-1}\, dx \\
&= \left[-x(1-x)^n + \int (1-x)^n\, dx\right]_{x=0}^1 \qquad \text{(by parts)} \\
&= \left[-x(1-x)^n - \frac{(1-x)^{n+1}}{n+1}\right]_{x=0}^1 = \boxed{\frac{1}{n+1}}.
\end{aligned}
$$

3. We present two ways of solving this problem: the right way and the better way.

   We'll start with the better way (the way we want you to do it). Define indicator variables:

$$
I_i = \begin{cases} 0 & \text{if player } i \text{ is sick} \\ 1 & \text{otherwise.} \end{cases}
$$

   Then, $E[I_i] = P[I_i = 1] = 7/8$, and:

$$
E[I_i I_j] = P[I_i = 1 \text{ and } I_j = 1] = \frac{\binom{30}{4}}{\binom{32}{4}} = \frac{189}{248}, \qquad \text{for } i \neq j.
$$

   Without loss of generality, assume that the players are seeded so that player $2i-1$ is to play with player $2i$ for $i = 1 \ldots 16$. Then, the expected number of remaining pairs is:

$$
\begin{aligned}
E[\text{number of healthy pairs}] &= E\left[\sum_{i=1}^{16} I_{2i-1} I_{2i}\right] \\
&= \sum_{i=1}^{16} E[I_{2i-1} I_{2i}] \qquad \text{by linearity of expecation} \\
&= 16 \cdot \frac{189}{248} = \boxed{\frac{378}{31}} \approx 12.2.
\end{aligned}
$$

   Now for the right (and boring) way:

$$
\begin{aligned}
E[\text{number of healthy pairs}] &= \sum_{i=1}^{n} i P[\text{number of healthy pairs} = i] \\
&= 12 \times P[\text{number of healthy pairs} = 12] \\
&\quad + 13 \times P[\text{number of healthy pairs} = 13] \\
&\quad + 14 \times P[\text{number of healthy pairs} = 14] \\
&= 12 \times \frac{\binom{16}{4} \cdot 2^4}{\binom{32}{4}} + 13 \times \frac{\binom{16}{1}\binom{15}{2} 2^2}{\binom{32}{4}} + 14 \times \frac{\binom{16}{2}}{\binom{32}{4}} = \boxed{\frac{378}{31}}.
\end{aligned}
$$

4. (a) You should switch! This is equivalent to changing your choice to both of the doors that you did not choose. This is very counterintuitive, since it seems that switching to a single new door is the same as choosing that door to begin with, but the door that Monty opens depends on the door that you initially chose (since if we chose one of the goats to begin with, we have forced Monty to open the door to the other goat, and the last door must conceal the Corvette). Thus, you should definitely switch!

   (b)

```
≫ switchscore = 0;
≫ stayscore = 0;
≫ for i = 1 : 1000
·    doors = [1, 2, 3];
·    corvette = randsample(doors, 1);
·    myguess = randsample(doors, 1);
·    doors = find(doors ~= corvette);
·    doors = find(doors ~= myguess);
·    montyopens = randsample(doors, 1);
·    if myguess == corvette
·    ·    stayscore = stayscore + 1;
·    else
·    ·    switchscore = switchscore + 1;
·    end
≫ end
≫ [stayscore, switchscore]
ans =
     335    665
```

   (c) Our previous reasoning still holds; it doesn't matter which door you switch to; by symmetry, they both have the same probability of hiding Corvettes.

5. (a)

$$P(x_1, x_2) = \frac{1}{2\pi} \exp\left\{ -\left[\frac{(x_1 - 10)^2}{2} - (x_1 - 10)(x_2 - 5) + (x_2 - 5)^2 \right] \right\}$$

   (b)

$$\begin{aligned} cov\left(Ax, Bx\right) &= E\left[\left(Ax - E\left[Ax\right]\right)\left(Bx - E\left[Bx\right]\right)^T\right] \\ &= E\left[A\left(x - E\left[x\right]\right)\left(B\left(x - E\left[x\right]\right)\right)^T\right] \\ &= E\left[A\left(x - E\left[x\right]\right)\left(x - E\left[x\right]\right)^T B^T\right] \\ &= AE\left[\left(x - E\left[x\right]\right)\left(x - E\left[x\right]\right)^T\right] B^T \\ &= A \cdot cov\left(x\right) \cdot B^T. \end{aligned}$$

6.

$\gg v1 = [0, 0, 0, 0, 0, 1]';$
$\gg v2 = [1, 2, 3, 4, 5, 6]';$
$\gg v3 = [1, 4, 9, 16, 25, 36]';$
$\gg v4 = [1, 0, 0, 0, 0, 0]';$
$\gg u1 = v1;$
$\gg u1 = u1/\mathbf{norm}(u1);$
$\gg u2 = v2 - (v2' * u1) * u1;$
$\gg u2 = u2/\mathbf{norm}(u2);$
$\gg u3 = v3 - (v3' * u1) * u1 - (v3' * u2) * u2;$
$\gg u3 = u3/\mathbf{norm}(u3);$
$\gg u4 = v4 - (v4' * u1) * u1 - (v4' * u2) * u2 - (v4' * u3) * u3;$
$\gg u4 = u4/\mathbf{norm}(u4);$
$\gg [u1, u2, u3, u4]$

ans =

0	0.1348	−0.4040	0.9048
0	0.2697	−0.5465	−0.2842
0	0.4045	−0.4277	−0.2513
0	0.5394	−0.0475	−0.1016
0	0.6742	0.5941	0.1648
1.0000	0	0	0

If we start with vectors in a different order, we will get a different basis; for example, the first vector of the new basis will always be a unit vector in the direction of the first vector provided to us.