Advanced Machine Learning

Summer Semester 2019

Jonathon Luiten <luiten@vision.rwth-aachen.de>
Ömer Sali <sali@vision.rwth-aachen.de>

Exercise 1: Sampling and Digit Recognition

due before Thursday 27th of June 08:00 am

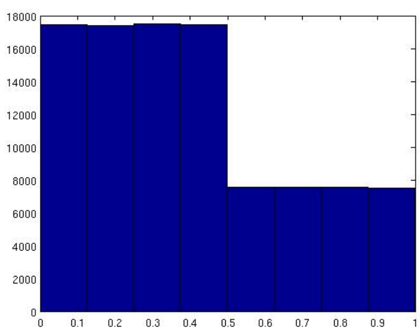
Turn in your solutions to the RWTH Moodle. This exercise is non-compulsory

Remember:

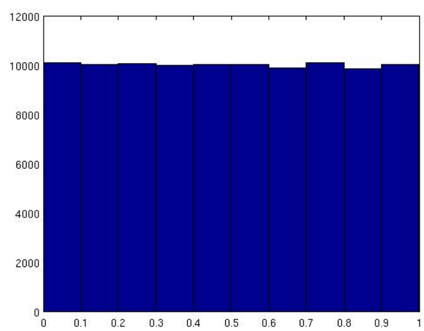
- feel free to form groups to work on the exercises, and discuss the problems and the code
- put all group members' names in your submission
- use the .zip, .tar.gz, or .tar.bzip format to zip your source code
- please use one separate code file for each sub-problem, question1a.m, question1b.m, ...
  Even if this means that you copy some code, it makes things much clearer.

Question 1: Sampling Warming Up

Lets start with a simple sampling example. Suppose that `badRand()` produces random samples on the interval $[0, 1]$, but not quite as well as rand() does: 70% of the time, `badRand()` produces a value that's uniformly distributed between 0 and 0.5; the other 30% of the time, it produces a uniformly distributed value between 0.5 and 1.0. The probability density for `badRand()` is shown in figure 1(a).



(a) before



(b) after

a) Adjust the samples drawn from the `badRand()` such that the resulting distribution is uniform on $[0, 1]$. As in the right-hand side of figure 1(b).

   For creating `badRand()` use the following matlab function:

```
function x = badRand()
    r = rand;
    if(r < 0.7)
        x = rand / 2;
    else
        x = 0.5 + rand/2;
    end
end
```

Question 2: Rejection Sampling, Transformation method

Suppose we want to generate points between 0 and 1 where the probability density of selection x is e.g. $f_X(x) = 2x^2$. In the lecture you've learned a number of approaches, that could be used for this task.

a) As first lets recap the *rejection sampling* algorithm:

   1. Draw x from an uniform distribution on $[0, 1]$;

2. Draw another value s from an uniform distribution on $[0, 2]$;
3. If $s < f_X(x)$, keep x; otherwise reject and start again by step 1;

Implement the *rejection sampling* algorithm using Matlab. For generating an uniform random number use the Matlab function `rand()`.

b) Since we use a very simple pdf $f_X(x)$ we can also use *transformation method* for generating samples. The corresponding cumulative distribution:

$$F(x) = \int_{\infty}^{x} f_X(z)dz$$

can easily be computed. To draw samples from this pdf, we just need to invert the cumulative distribution function:

$$u \sim Uniform(0,1) \rightarrow F^{-1}(u) \sim f(x)$$

Implement this method for generating samples of the proposed pdf $f_X(x)$ using Matlab.

Question 3: Importance Sampling

We will now turn to *Importance Sampling*. Remember, this is not a method to generate samples from the distribution directly, but a method to estimate the expection. It can be viewed as a generalization of the uniform sampling method. In this exercise you should compute the expectation of the following function:

$$f(x) = \begin{Bmatrix} \frac{1}{100}\sqrt{x} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{Bmatrix}$$

under the following target distribution:

$$p(x) = w_1 \mathcal{N}(x|\mu_1, \sigma_1) + w_2 \mathcal{N}(x|\mu_2, \sigma_2)$$

using as proposal distribution:

$$g(x) = \mathcal{N}(x|\mu, \sigma)$$

Implement importance sampling in Matlab using following values for the parameters:

$$\mu = 3,\ \mu_1 = 0,\ \mu_2 = 10,\ \sigma_1 = 2,\ \sigma_2 = 2,\ \sigma = 10,\ w_1 = 0.3,\ w_2 = 0.7$$

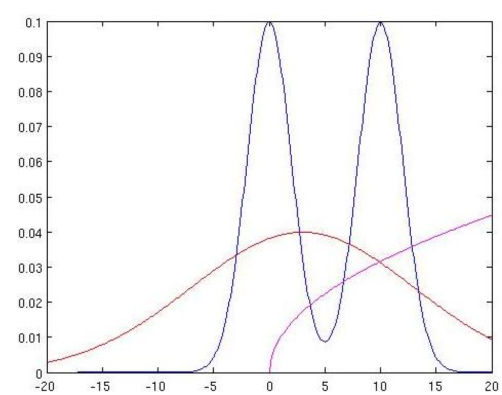


Figure 1: Blue: target distribution Red: proposal distribution Magenta: proposed function

Question 4: Metropolis Hasting (MH)

The aim of this exercise is to illustrate the *Metropolis Hasting (MH)* algorithm by applying it to a simple model.

a) Given is the following target distribution:

$$p(x) = \frac{1}{\sqrt{(2\pi)}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$$

Implement the MH algorithm to generate samples from the proposed target distribution using as proposal distribution $q(x|x')$ the random walk:

$$x \sim Uniform(x' - \frac{h}{2}, x' + \frac{h}{2}).$$

Let $\mu = 0$, $\sigma^2 = 1$, $h = 1$.

- Use your implementation to generate 10000 MH-updates.
- Make a histogram of the sampled values and compare to the true distribution.

b) As next step use your MH implementation for sampling from a mixture of 1D Gaussian as given in Question 3:

$$p(x) = w_1\mathcal{N}(x|\mu_1, \sigma_1) + w_2\mathcal{N}(x|\mu_2, \sigma_2)$$

using the proposal distribution:

$$g(x) = \mathcal{N}(x|\mu, \sigma)$$

- Use the paramater values that are given in Question (3).
- Use your implementation to generate 10000 MH-updates.
- Make a histogram of the sampled values and compare to the true distribution.

Hint: Omit the first 2000 samples.