1 Title: Generation of N-dimensional normally distributed random numbers from two categories with different priors

2 Introduction

In the most of pattern recognition, decision and classification problems, the concept of training becomes popular with the advance of Machine Learning and computing power that can afford to expensive computatioal costs. Such a trainable system is a record-holder for many open problems and its powerful capability is successfully built based on the massive volume of information in a dataset. Due to the importance of data, people not only try to obtain a big quantity of data, but also concentrate a good quality of data. Often, a real-world data contains either high variance or high bias hence it is difficult to estimate true density and using such data does not necessarily result in good performance. In the circumstance, a naive assumption about the class distribution helps us synthesize data so that we can train models with a consistent dataset. Among many distributions, Normal distribution is frequently used in many literatures. This tutorial will explain how to generate binary data classes from normal distributions with prior probabilities in a perspective on numerical experiments.

It consists of the following sections:

1. Prior selection : How to set priors?
2. Normal distribution : How it work? Which is more efficient?
   (a) Central limit theorem
   (b) Inverse transform sampling method
   (c) Box-Muller transform
   (d) Ziggurat Algorithm
3. Conversion from normalized distribution
4. Extension to N-dimensional data
5. Conclusion

3 Prior Selection

Prior probability gives an idea on which class is more likely to be observed among all possible classes. For binary data classes with normal distributions, it is important to draw samples based on priors before investigating how to generate distributions. This is because in practice the number of samples contains implicit information about prior probability and also empirical prior is different from the true prior used to determine class selection where the empirical prior is calculated by only looking at the ratio of class 1 and class 2 for binary classification example.

To be specific, prior probability affects the generation of samples for binary classification in a way described in the following. First, sampling data from uniform distribution $(0, 1)$ and let the class priors be $Prob(\omega_1)$ and $Prob(\omega_2)$ respectively where the condition $Prob(\omega_1) + Prob(\omega_2) = 1$ holds. If a random sample is drawn in the range $[0, Prob(\omega_1)]$, label the sample as class 1, then, continue to generating a normal random number based on the class 1 statistics (μ, σ). Otherwise, the sample belongs to $[Prob(\omega_2), 1]$ and should be labeled as class 2, then, move onto the normal random number generation step with the class 2 statistics like the same way as we did for class 1. In summary,

$$X_i = \begin{cases} X_i^1 \sim N(\mu_1, \sigma_1) & X_i \in [0, Prob(\omega_1)] \\ X_i^2 \sim N(\mu_2, \sigma_2) & \text{otherwise} \end{cases}$$

where the superscript of X_i indicates the label of the class.

4 Normal Distribution Generation

Once you decide the class of the sample, now it is time to generate actual random number that follows the class statistics determined in the prior selection step. The normal distribution is frequetly used in many statistical models and being able to draw samples from the normal distribution lies at the heart of many probabilistic learning algorithms. The probability distribution function for the normal distribution is defined as

$$y = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\} \tag{1}$$

where μ is the distribution mean and σ is the standard deviation.

It is easy to generate an uniform distribution, but usually generation of a normal distribution requires additional steps rather than having a single step solution. Central limit theorem, inverse transform sampling method, Box-Muller transformation method and Ziggurat algorithm are examples to generate such distribution, given a source of uniform distribution. Since each of methods has its pros and cons, the optimal selection among those algorithm may differs under different conditions.

4.1 Central Limit Theorem

Centrla limit theorem states that when a sufficiently large number of samples drawn from independent random variables, the arithmetic mean of their distributions will be have a normal distribution as commonly known as a bell-shaped distribution. This implies that sampling from identical and independent uniform distributions will result in a distribution which will be normally distributed as the number of samples involved are large enough. This is shown in the following

$$\begin{aligned} \lim_{n\to\infty} \frac{X_1 + \cdots + X_n}{n} &\approx \mu \\ \lim_{n\to\infty} \frac{X_1^2 + \cdots + X_n^2}{n} &\approx \sigma^2 + \mu^2 \end{aligned} \tag{2}$$

where $\{X_1, \ldots, X_n\}$ be a random sample of size n and μ, σ represents the mean and standard deviation for a normal distribution. However, in practice

approaching a normal probabilistic distribution to a high accuracy using only central limit theorem requires an impractically large number of samples. Therefore, using this method to generate normal distribution is expensive and not plausible.

4.2 Inverse Transform Sampling Method

Inverse transform sampling method is a fundamental method in sampling for random numbers and can generate any types of probabilistic distributions given a source of uniform distributions. This method involves computing the quantile function. Which of a probabilistic distribution is the inverse of its cumulative distribution. Basically, it generates random numbers u from the uniform distribution in the interval $[0, 1]$. Then, using equations for the desired probabilistic distribution, it computes the value that satisfies $F(x) = u$. Finally, the number x drawn from F density distribution is considered as a random sample.

Random numbers that follow a normal distribution can be obtained by the following procedure. First, it starts from Poisson distribution described below.

$$p(k) = P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}, k \geq 0. \tag{3}$$

λ and k are all deterministic since λ is a given mean and k is a nth point of Poisson process. If we can simulate $X = N(1)$ where $\{N(t) : t \geq 0\}$, Poisson random number X can be obtained by using the formula

$$X = N(1) - 1 = min\{n \geq 1 : t_n > 1\} - 1 \tag{4}$$

where $t_n = X_1 + \cdots + X_n$. The relation above comes from the exponential term in Poisson distribution $e^{-\lambda}n$, that is $X_i = -\frac{1}{\lambda}\ln u_i$. Using such a relation with additivity of logarithm helps to generate a Poisson random number easily by consecutively adding X_i (equivalently, multiplying uniform random numbers u_i). Details are described as

$$\begin{aligned} X &= \min\left\{n \geq 1 : \sum_{i=1}^{n} \ln(u_i) < -\alpha\right\} - 1 \\ &= \min\left\{n \geq 1 : \ln\left(\prod_{i=1}^{n} u_i\right) < -\alpha\right\} - 1 \\ &= \min\left\{n \geq 1 : \prod_{i=1}^{n} u_i < e^{-\alpha}\right\} - 1 \end{aligned} \tag{5}$$

where the product with new u_i drawn from $U \sim \text{uniform}(0, 1)$ is repeated until we find $X = \prod_{i=1}^{n} u_i < e^{-\alpha}$. Note that for large λ in a Poisson, its distribution is approximately normal with mean of λ and variance of λ by the central limit theorem. As a result, a normal distribution can be generated once we obtain a Poisson distribution based on the samples from uniform distribution using inverse transform sampling method.

This method involves computing the quantile function of the distribution which utilizes the cumulative distribution function and then inverting that function. This is why the terminology is called "inverse" method. For a discrete

distribution, summation over all individual samples drawn from uniform distribution yields desired distributions. However, for a continuous cases, integration over probability density function is needed like the same way as we did in discrete domain, but it is impossible to obtain an analytical solution for most distributions. This shortcoming makes this method computationally inefficient in continuous domain and the alternative such as Box-Muller transform can be used.

4.3 Box-Muller Transform

The method generates a normal distribution given a source of uniform distribution. Main key of this method is to utilize the relation between Cartesian and polar coordinates. The polar form picks two samples from the interval, $[1, +1]$, and maps them to two independent samples that are normally distributed without the use of sine or cosine functions. The relation between two different domains can be characterized as few equations

$$\begin{aligned} r^2 = x_1^2 + x_2^2, \qquad & \tan\theta = \frac{x_2}{x_1} \\ x_1 = r\cos\theta, \qquad & x_2 = r\sin\theta \end{aligned} \tag{6}$$

where $\theta \in [0, 2\pi]$ and we assume $|r| \leq 1$ since we are going to use $r \sim \text{uniform}(0, 1)$ as a basic building block. Such region covers the area contained in the unit circle in polar coordinate.

Box-Muller sampling is based on the joint distribution of two independent random variables $x_1 \sim N(0, 1), x_2 \sim N(0, 1)$ in Cartesian coordinate. If we convert such distributions in polar coordinate, the joint distribution $p(x_1, x_2)$ becomes

$$\begin{aligned} p(x_1, x_2) &= p(x_1)p(x_2) \\ &= \frac{1}{2\pi}\exp\left\{-\frac{x_1^2}{2}\right\}\exp\left\{-\frac{x_2^2}{2}\right\} \\ &= \frac{1}{2\pi}\exp\left\{-\frac{x_1^2 + x_2^2}{2}\right\} \\ &= \frac{1}{2\pi}\exp\left\{-\frac{r_2}{2}\right\} \end{aligned} \tag{7}$$

From the last line of the equation, we can see that the distribution follows an exponential distribution with respect to r^2. Precisely, distributions in polar coordinate are:

$$r^2 \sim \text{exponential}\left(\frac{1}{2}\right), \qquad \theta \sim \text{uniform}\,(0, 2\pi) \tag{8}$$

Since the angle θ already follows an uniform distribution, it can be simply drawn from an uniform distribution. However, the additional work is needed to generate the exponential distribution. The connection between uniform and exponential can be formulated as follows:

$$\text{exponential}\,(\lambda) = \frac{-\log\left(\text{uniform}(0, 1)\right)}{\lambda} \tag{9}$$

Then, combining with the equation in terms of r^2 and solving for r gives,

$$r \sim \sqrt{-2\log\left(\text{uniform}(0,1)\right)} \tag{10}$$

Now all variables r and θ can be expressed in terms of uniform distributions. By back-tracking above procedure, we will be able to successively generate a normal distribution from out of uniform distributions. This way to generate normal random numbers is known as Box-Muller transformations with polar method, which makes Box-Muller transformation more efficient by avoiding direct calculations of cosine and sine functions.

4.4 Ziggurat Algorithm

Ziggurat method is the fastest algorithm for generating normal random numbers. It is more efficient than Box-Muller method but is more complicated algorithm. Due to the efficiency and speed, it has been implemented in various libraries such as GNU scientific library and MatLab.

This method partitions a target distribution into a small blocks so that the whole distribution can be described as an union of blocks which is

$$Z = \bigcup_{i=0}^{C} B_i \tag{11}$$

where the area of each B_i is a rectangle whose width extends from $x = 0$ to $x = x_i$ horizontally and whose height vertically extends from $f(x_i)$ to $f(x_{i+1})$ (details are shown in the following equation).

$$B_i = \begin{cases} 1 & \{(x,y)|0 < x < x_i, f(x_i) < y < f(x_{i+1})\} \\ 0 & \text{otherwise} \end{cases}$$

Here x_is are x-values that monotonically increases from the center to the tail for a given normal distribution.

The algorithm first choose a random block i among all possible B_is. Then, it draws a random number u_0 from uniform $(0, 1)$ and calculate z by multiplying the x-axis index i with u_0. If the randomly generated z belongs to any of partitioned blocks, z returns as a normal random number. Otherwise, it again calculates a similar step for y-axis to check whether the sample belongs to wedge or not. Since understanding of the principle of this algorithm requires background knowledge, details are out of scope for this slecture and please refer to the paper titled "The Ziggurat Method for Generating Random Variables".

The critical values such as the edges of the rectangle (x_i or $f(x_i)$) used in Ziggurat method are stored in memory, which improves efficiency in terms of computational cost. Moreover, it does not need to evaluate an exponential function all the time while Box-Muller method computes the exponential (or logarithm) every time. The ziggurat algorithm will calculate an exponential function only if samples are in the wedge. For these reason, the ziggurat algorithm is the fastest way to generate but relatively difficult to implement it is best used when large quantities of random numbers are needed.

5 Conversion from normalized distribution

Any normal distribution with μ_0 and σ_0 can be easily converted to another normal distribution with a custom mean μ_1 and standard deviation σ_1. For convenience and resource saving purpose, random numbers are generated from normalized normal distribution that follows $N(0,1)$ from a single unified source, then such numbers are converted to follow a custom normal distribution simply using the following equation.

$$\begin{aligned} Z &= \frac{X-\mu_0}{\sigma_0} \quad \text{where } Z \sim N(0,1) \text{ and } X \sim N(\mu_0, \sigma_0) \\ X &= \sigma_i Z + \mu_i \quad \text{where } Z \sim N(0,1) \text{ and } X \sim N(\mu_i, \sigma_i) \end{aligned} \tag{12}$$

6 Extension to N-dimensional data

Until now, we have studied how to generate normally distributed random number in 1-dimensional space. The same method can be extended to N-dimensional space to create multi-dimensional random vectors whose underlying distribution is normal. Compared to the previous case where we were using a scalar value for a mean and a standard deviation, N-dimensional random vectors can be obtained by repeating the single random number generation process N times, then concatenating random numbers as a column-wise vector. This is possible because all basis are orthogonal to each other and generating a single random value in a vector does not affect the generation process of other components. Note that prior selection is used to choose either class vector 1 or class vector 2. Once the class is determined by its prior drawn from uniform distribution, random number generation process is repeated N times to fill out the random vector. Each of which will follow class vector statistics, which are

$$M^1 = \begin{bmatrix} \mu_1^1 \\ \mu_2^1 \\ \vdots \\ \mu_N^1 \end{bmatrix} \quad S^1 = \begin{bmatrix} \sigma_1^1 \\ \sigma_2^1 \\ \vdots \\ \sigma_N^1 \end{bmatrix} \quad M^2 = \begin{bmatrix} \mu_1^2 \\ \mu_2^2 \\ \vdots \\ \mu_N^2 \end{bmatrix} \quad S^2 = \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \vdots \\ \sigma_N^2 \end{bmatrix} \tag{13}$$

where the superscript denotes the class label and the subscript indicates an element index. M and S denotes a mean matrix and a standard deviation matrix, respectively.

Surely, this method is no more available when more than two basis in a given multi-dimension space are correlated. The scope of this work covers a general case where all basis in the search space are orthogonal.

7 Conclusion

Now we understand how to generate N-dimensional normally distributed random numbers from two categories with different priors using samples taken from uniform distributions. The generation of random numbers with a normal distribution combined with the prior selection step discussed in the earlier section provides statistically consistent datasets, which will improve the accuracy of

trainable systems. While class selection based on the prior probabilities is relatively simple, the generation of random numbers with a normal distribution has many issues to be considered because of the trade-off between its computational cost and the accuracy. The four methods covered in this work have their own pros and cons, but practically efficiency and complexity are the most important concerns to be considered when deciding an optimal model for given task.

8 References

[1] Dong-U Lee, Wayne Luk, John Villasenor and Peter Cheung, ‘‘A Hardware Gaussian Noise G
[2] Raul Toral and Amitabha Chakrabarti, ‘‘Generation of Gaussian distributed random numbe
[3] Hassan Edrees, Brian Cheung, McCullen Sandora, David Nummey and Deian Stefan ‘‘Hardwar
[4] Central Limit Theorem, wikipedia, http://en.wikipedia.org/wiki/Central_limit_theorem
[5] Inverse Transform Sampling, wikipedia, http://en.wikipedia.org/wiki/Inverse_transform_
[6] Box-Muller Transform, wikipedia, http://en.wikipedia.org/wiki/Box_Muller_transform
[7] Box-Muller Transformation, mathworld, http://mathworld.wolfram.com/Box-MullerTransform
[8] Emiliano A. Valdez, Lecture note, http://www.math.uconn.edu/~valdez/math3634s09/Math36
[9] Karl Sigman, Lecture note, http://www.columbia.edu/~ks20/4404-Sigman/4404-Notes-ITM.pd
[10] Henrik Schoioler, Lecture note, http://www.control.auc.dk/~henrik/undervisning/DES/le
[11] http://theclevermachine.wordpress.com/2012/09/11/sampling-from-the-normal-distributio