Lab1 : Sampling from a Gaussian Distribution

G. Dileep Kumar
Assistant Professor

June 17, 2025

1 Theoretical Concepts

A **Gaussian distribution** (also called *Normal distribution*) is defined by two parameters:

- **Mean** (μ): The center or average value.
- **Standard Deviation** (σ): Measures spread or variability.

The probability density function (PDF) of a Gaussian distribution is:

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

Sampling from this distribution means generating random values x that follow this probability distribution.

1.1 Why Sampling?

Sampling allows you to generate random data points that behave like real-world data which is normally distributed (e.g., heights, test scores).

1.2 How to Sample?

Programming languages provide built-in functions to sample from a normal distribution.

For example, in Python, `numpy.random.normal(mean, std, size)` generates samples.

2 Applications of Gaussian Sampling in Generative AI

- **Variational Autoencoders (VAEs):**
  VAEs encode input data into a latent space described by a Gaussian distribution (mean and variance). Sampling from this latent Gaussian distribution generates diverse and realistic new data after decoding.

- **Latent Space Exploration:**
  Gaussian sampling in latent spaces enables smooth interpolation and exploration of data variations, helping generate novel samples by tweaking latent vectors.
- **Diffusion Models:**
  These models progressively add Gaussian noise to data and learn to reverse the process. Sampling Gaussian noise is essential for both training and generating new samples.
- **Generative Adversarial Networks (GANs):**
  The generator network takes as input a noise vector sampled from a Gaussian (or uniform) distribution to create new data samples.
- **Data Augmentation:**
  Gaussian noise sampling can augment training datasets by creating realistic variations of data points, improving generalization of generative models.
- **Bayesian Deep Learning:**
  Gaussian sampling is used to estimate uncertainty in generative models by sampling model parameters or latent variables from Gaussian posteriors.

3 Implementation

3.1 Task

1. Generate N samples from a Gaussian distribution with a given mean μ and standard deviation σ.
2. Visualize the samples by plotting a histogram.
3. Optionally, compute the sample mean and sample standard deviation of the generated samples to verify if they approximate the true parameters μ and σ.

3.2 Python Code: Example 1

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
mean = 0        # Mean of the distribution
std_dev = 1     # Standard deviation of the distribution
N = 1000        # Number of samples to generate

# Sampling from Gaussian
samples = np.random.normal(mean, std_dev, N)
```

```python

# Compute sample statistics
sample_mean = np.mean(samples)
sample_std = np.std(samples)

print(f"Sample Mean: {sample_mean:.4f}")
print(f"Sample Std Dev: {sample_std:.4f}")

# Plotting histogram of samples
plt.hist(samples, bins=30, density=True, alpha=0.6, color='
    blue')

# Plot the true Gaussian PDF for reference
x = np.linspace(mean - 4*std_dev, mean + 4*std_dev, 1000)
pdf = (1 / (std_dev * np.sqrt(2 * np.pi))) * np.exp(- (x -
    mean)**2 / (2 * std_dev**2))
plt.plot(x, pdf, 'r', linewidth=2)

plt.title('Sampling from Gaussian Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
```

Listing 1: Sampling from a Gaussian Distribution

3.3 Multiple Parameters Example

In this example, we generate samples from two Gaussian distributions with different means and standard deviations. We visualize the samples by plotting overlaid histograms to compare how changing the mean and standard deviation affects the distribution shape.

```python
import numpy as np
import matplotlib.pyplot as plt

# Define parameters for two Gaussian distributions
params = [
    {'mean': 0, 'std_dev': 1, 'label': 'Mean=0, Std=1', '
        color': 'blue'},
    {'mean': 5, 'std_dev': 2, 'label': 'Mean=5, Std=2', '
        color': 'green'}
]

N = 1000  # Number of samples

plt.figure(figsize=(10, 6))

for p in params:
    samples = np.random.normal(p['mean'], p['std_dev'], N)
    plt.hist(samples, bins=30, density=True, alpha=0.5,
        label=p['label'], color=p['color'])

plt.title('Sampling from Different Gaussian Distributions')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.show()
```

Listing 2: Sampling from Different Gaussian Distributions