Machine Learning 2: Assignment 1

Vid Hanžel, Christoph Royer

March 2025

1 Pseudorandom Number Generation

1.1 Inversion Method

1.1.1 Finding inverse CDFs

a)

$$f\left(\cdot|a,b\right) : x \in \rightarrow c_{\chi[a,b]}(x) a, b \in \mathbb{R}, a < b$$

Calculate normalising constant:

$$\begin{aligned} 1 &= \int_{\infty}^{\infty} f(x)dx \\ &= c\int_a^b 1dx \\ &= c(b-a) \\ \Rightarrow c &= \frac{1}{b-a} \end{aligned}$$

$$f(\cdot|a,b) : x \in \mathbb{R} \rightarrow \frac{1}{b-a}\chi_{[a,b]}(x)$$

Create CDF:

$$\begin{aligned} F_f(x \mid a,b) &= \int_{-\infty}^{x} f(z \mid a)dz \\ &= \int_{-\infty}^{x} \frac{1}{b-a}\chi_{[a;b]}(z)dz \end{aligned}$$

Piecewise solutions for the integral:

$$\begin{aligned} \forall x < a : \int_{-\infty}^{x} \chi_{[a,b]}(x)dx &= 0 \\ \forall x > b : \int_{-\infty}^{x} \chi_{[a,b]}(x)dx &= \int_a^b 1dx \\ \forall x, a \leq x \leq b : \int_{-\infty}^{x} \chi_{[a,b]}(x)dx &= \int_a^x 1dx = x - a \end{aligned}$$

Final CDF:

$$F_f(x \mid a,b) = \int_{-\infty}^{x} f(z \mid a,b)dz = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{else} \end{cases}$$

Invert CDF in range $a \leq x \leq b$:

$$\begin{aligned} &\text{if } a \leq x \leq b: \\ &\qquad y = \frac{x-a}{b-a} \\ &\qquad y(b-a) = x-a \\ &\qquad y(b-a)+a = x \end{aligned}$$

$$\begin{aligned} a \leq & x \leq b \\ a \leq & y(b-a)+a \leq b \\ 0 \leq & y(b-a) \leq b-a \\ 0 \leq & y \leq 1 \end{aligned}$$

Final inverted CDF:

$$F_f^{-1}(y \mid a,b) = y(b-a)+a \quad \forall 0 \leq y \leq 1$$

b)

$$g(\cdot \mid \lambda) : x \in \mathbb{R} \to ce^{-\lambda x}\chi_{[0,\infty]}(x), \lambda \in (0,\infty)$$

Calculate normalising constant:

$$\begin{aligned} \int_{-\infty}^{\infty} ce^{-ax}\chi_{[0,\infty)}(x)dx = 1 \\ \int_{0}^{\infty} ce^{-\lambda x}dx = 1 \\ c = 1/\left(\int_0^\infty e^{-\lambda x}dx\right) \\ c = 1/\left(\frac{e^{-\lambda x}}{-x}\bigg|_{x=0}^{\infty}\right) \\ c = 1/\left(\frac{1}{\lambda}\right) \\ c = \lambda \end{aligned}$$

$$g(x \mid \lambda) = \lambda e^{-\lambda x}\chi_{[0,\infty)}(x)$$

Create CDF:

$$F_g\left(x \middle| \lambda\right) = \int_{-\infty}^{x} \lambda e^{-\lambda z}\chi_{[0,\infty}(z)dz$$

If $x \geq 0$:

$$\begin{aligned} F_g\left(x \middle| \lambda\right) &= \int_0^x \lambda e^{-\lambda z} dz \\ &= \lambda \left(\frac{e^{-\lambda x}}{-\lambda} - \frac{e^{-\lambda \cdot 0}}{-\lambda} \right) \\ &= e^0 - e^{-\lambda x} \\ &= 1 - e^{-\lambda x} \end{aligned}$$

If $x < 0$:

$$\int_{-\infty}^x \lambda_e e^{-\lambda z} x_{[0;\infty}(2) dz = 0$$

Final CDF:

$$F_g(x \mid \lambda) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

Invert CDF in range $x \geq 0$:

$$\begin{aligned} y &= 1 - e^{-\lambda x} \\ 1 - y &= e^{-\lambda x} \\ \ln(1-y) &= -\lambda x \\ \frac{\ln(1-y)}{-\lambda} &= x \end{aligned}$$

$$\begin{aligned} \frac{\ln(1-y)}{-\lambda} &\geq 0 \\ \ln(1-y) &\geq 0 \\ 1 - y &\geq 1 \\ 0 &\geq y \end{aligned}$$

$$\ln(1-y) \Rightarrow y \leq 1$$

Final inverted CDF:

$$F_g^{-1}(y \mid \lambda) = \frac{\ln(1-y)}{-\lambda} \quad \forall 0 \leq y \leq 1$$

c)

$$h(\cdot \mid \alpha, \beta) : x \in \mathbb{R} \to c\frac{1}{x\log(\beta/\alpha)}\chi_{[\alpha,\beta]}(x) \quad \alpha, \beta \in (0,\infty), \alpha < \beta$$

Calculate normalising constant:

$$\begin{aligned}
\int_{-\infty}^{\infty} c\frac{1}{x\log(\beta/x)}x_{[x,\beta]}(x)dx &= 1 \\
c\int_{\alpha}^{\beta} \frac{1}{x\log(\beta/\alpha)}dx &= 1 \\
\frac{c}{\log(\beta/\alpha)}\int_{\alpha}^{\beta} \frac{1}{x}dx &= 1 \\
\frac{c}{\log(\beta/\alpha)}\left(\log(\beta) - \log(\alpha)\right) &= 1 \\
\frac{c}{\log(\beta/\alpha)}\left(\log\left(\frac{\beta}{\alpha}\right)\right) &= 1 \\
c &= 1
\end{aligned}$$

$$h(x \mid \alpha, \beta) = \frac{1}{x\log(\beta/\alpha)} \quad \chi_{[\alpha,\beta]}(x)$$

Create CDF:

$$F_h(x \mid \alpha, \beta) = \int_{-\infty}^{x} \frac{1}{z\log(\beta/\alpha)}\chi_{[\alpha,\beta]}(z)dz$$

Piecewise definitions:

$$\forall x < \alpha : \quad \int_{-\infty}^{x} \frac{1}{z \log(\beta/\alpha)} \chi_{[\alpha,\beta]}(z) dz = 0$$

$$\begin{aligned}
\forall x > \beta : \quad & \int_{-\infty}^{x} \frac{1}{z \log(\beta/\alpha)} \chi_{[\alpha,\beta]}(z) dz \\
&= \int_{\alpha}^{\beta} \frac{1}{z \log(\beta/\alpha)} dz = 1 \qquad \text{(see above)} \\
\forall x \text{s.t. } \alpha \leq x \leq \beta : & \int_{-\infty}^{x} \frac{1}{z \log(\beta/\alpha)} \chi_{[\alpha,\beta]}(z) dz \\
&= \int_{\alpha}^{x} \frac{1}{z \log(\beta/\alpha)} dz \\
&= \frac{1}{\log(\beta/\alpha)} (\log(x) - \log(\alpha)) \\
&= \frac{\log(x/\alpha)}{\log(\beta/\alpha)}
\end{aligned}$$

Final CDF:

$$F_h(x \mid \alpha, \beta) = \begin{cases} 0 & \text{if } x < \alpha \\ \frac{\log(x/\alpha)}{\log(\beta/\alpha)} & \text{if } \alpha \leq x \leq \beta \\ 1 & \text{else} \end{cases}$$

Invert CDF in range $\alpha \leq x \leq \beta$:

$$\begin{aligned}
y &= \frac{\log(x/\alpha)}{\log(\beta/\alpha)} \\
y \log(\beta/\alpha) &= \log(x/\alpha) \\
e^{(y \log(\beta/\alpha))} &= \frac{x}{\alpha} \\
\alpha e^{(y \log(\beta/\alpha))} &= x
\end{aligned}$$

$$\begin{aligned}
&\alpha \leq \alpha e^{(y \log(\beta/\alpha))} \leq \beta \\
&1 \leq e^{(y \log(\beta/\alpha))} \leq \beta/\alpha \\
&0 \leq y \log(\beta/\alpha) \leq \log(\beta/\alpha) \\
&0 \leq y \leq 1
\end{aligned}$$

Final inverted CDF:

$$F_h^{-1}(y \mid \alpha, \beta) = \alpha e^{(y \log(\beta/\alpha))} \quad \forall 0 \leq y \leq 1$$

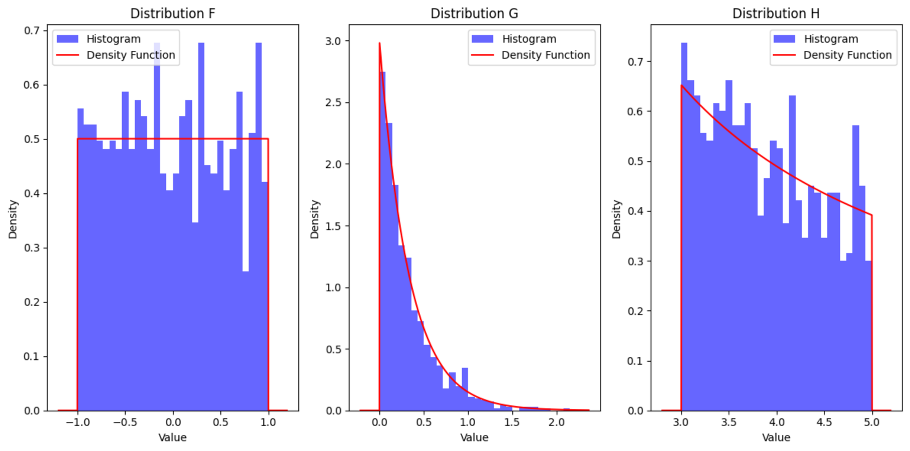

Figure 1: Distributions with 10^3 samples

1.1.2 Implementation

For the implementation, see `inversion_method.py`.

1.1.3 Draw samples

You can see the histograms after drawing samples 10^3, 10^4 and 10^5 times in Figure 1, Figure 2 and Figure 3 respectively.

1.1.4 Sample the standard normal distribution

You can see the adaptation of the inversion method to the standard normal distribution in Algorithm 1. It is not efficient, since it has to iteratively approximate the inverse of the cdf for each data sample. Depending on the chosen accuracy ϵ, this can take many iterations per sample.

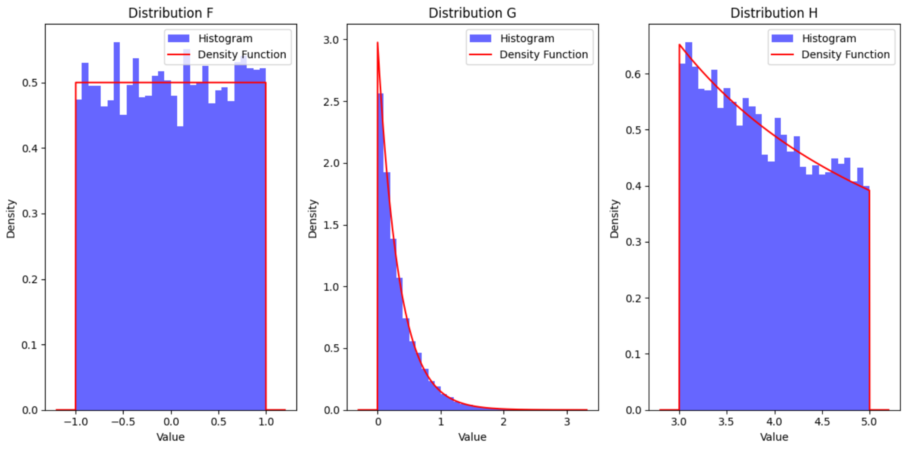

Figure 2: Distributions with 10^4 samples

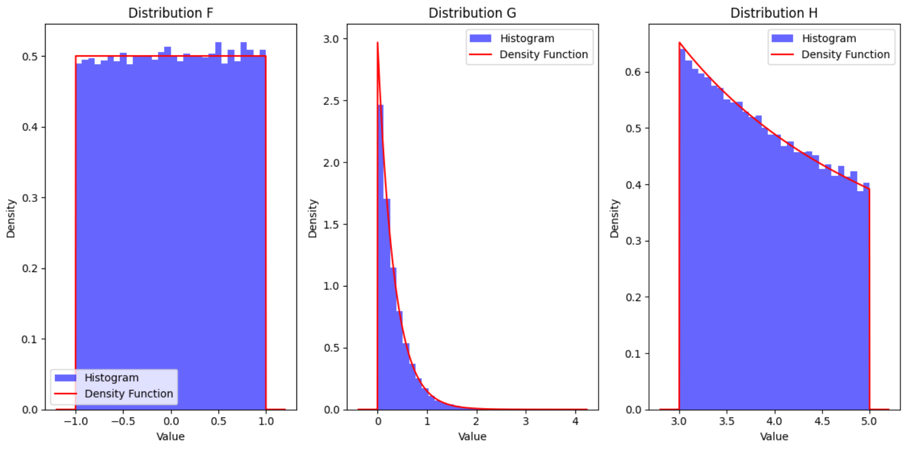

Figure 3: Distributions with 10^5 samples

```
Data: $u_k \overset{\text{iid}}{\sim} U(0,1)$, $cdf\_standard\_normal$, acceptable accuracy $\epsilon$
Result: $x_k \overset{\text{iid}}{\sim} N(0,1)$
for $k = 1$ to $n$ do
    $x_k \leftarrow 0$
    while $u_k > cdf\_standard\_normal(x_k)$ do
        $x_k \leftarrow x_k + 0.5$
    end
    $\Delta_x \leftarrow 0.5$
    while $|u_k - cdf\_standard\_normal(x_k)| > \epsilon$ do
        while $u_k > cdf\_standard\_normal(x_k - \Delta_x)$ do
            $\Delta_x \leftarrow \frac{\Delta_x}{2}$
        end
        $x_k \leftarrow x_k - \Delta_x$
    end
end
```
Algorithm 1: An algorithm to sample from the standard normal distribution

1.2 Box-Muller method

1.2.1 Proof

Lemma 1 (Box-Muller) *Let $U, V \sim U(0,1)$ be stochastically independent random variables. Define*

$$R = \sqrt{-2\log U}, \quad W = 2\pi V.$$

Then

$$X = R\cos W, \quad Y = R\sin W$$

are stochastically independent, standard normally distributed random variables.

$$U, V \sim U(0,1)$$
$$f_U(u) = \begin{cases} 1; 0 \leq u \leq 1 \\ 0;\ \text{else} \end{cases}$$
$$f_V(v) = \begin{cases} 1; 0 \leq v \leq 1 \\ 0;\ \text{else} \end{cases}$$

$$W = 2\pi V$$
$$V = \frac{W}{2\pi}$$
$$f_W(w) = f_v\left(\frac{w}{2\pi}\right)\left|\frac{d}{dw}\frac{1}{2\pi}\cdot w\right| = \frac{1}{2\pi}\begin{cases} 1; 0 \leq \frac{w}{2\pi} \leq 1 \\ 0;\ \text{else} \end{cases}$$
$$= \begin{cases} \frac{1}{2\pi}; 0 \leq w \leq 2\pi \\ 0;\ \text{else} \end{cases}$$

$$R = \sqrt{-2\log(U)}/^2$$

$$R^2 = -2\log(U)/ \ : -2$$

$$\frac{R^2}{-2} = \log(U)$$

$$U = e^{-\frac{R^2}{2}}$$

$$f_R(r) = f_U\left(e^{-\frac{r^2}{2}}\right) \cdot \left|\tfrac{d}{dr} e^{-\frac{r^2}{2}}\right| = \begin{cases} re^{-\frac{r^2}{2}}; 0 \le e^{-\frac{r^2}{2}} \le 1 \\ 0; \text{ else} \end{cases}$$

$$0 \le e^{-\frac{r^2}{2}} \le 1/\log$$

$$-\infty < -\frac{r^2}{2} \le 0/ : -\frac{1}{2}$$

$$+\infty > r^2 \geqslant 0$$

$$-\infty < r < \infty \implies r \in \mathbb{R}$$

$$f_R(r) = -r \cdot e^{-\frac{r^2}{2}}$$

$$f_{R,W}(r,w) = f_R(r) \cdot f_W(w) = \begin{cases} \frac{r \cdot e^{-\frac{r^2}{2}}}{2\pi}; & 0 \le w \le 2\pi, r \in \mathbb{R} \\ 0; & \text{else} \end{cases}$$

$$X = R\cos(W) \quad Y = R\sin(W)$$

$$X^2 + Y^2 = R^2\cos^2(\omega) + R^2\sin^2(\omega) =$$

$$= R^2 \underbrace{(\cos^2(\omega) + \sin^2(\omega))}_{1} = R^2$$

$$R = \sqrt{X^2 + Y^2}$$

$$\frac{Y}{X} = \frac{R \cdot \sin(W)}{R \cdot \cos(W)} = \tan(w)$$

$$W = \arctan\left(\frac{y}{x}\right)$$

$$\frac{\partial R}{\partial x} = \frac{x}{\sqrt{x^2+y^2}}, \quad \frac{\partial R}{\partial y} = \frac{y}{\sqrt{x^2+y^2}}$$

$$\frac{\partial W}{\partial x} = -\frac{y}{x^2+y^2}, \quad \frac{\partial W}{\partial y} = \frac{x}{x^2+y^2}$$

$$f_{X,Y}(x,y) = f_{\mathrm{R,W}}\left(\sqrt{x^2+y^2}, \arctan\left(\frac{y}{x}\right)\right) \begin{vmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ -\frac{y}{x^2+y^2} & \frac{x}{x^2+y^2} \end{vmatrix}$$

$$= \begin{cases} \frac{\sqrt{x^2+y^2}\cdot e^{-\frac{x^2+y^2}{2}}}{2\pi\cdot\sqrt{x^2+y^2}}; 0 \leq \arctan\left(\frac{y}{x}\right) \leq 2\pi, \sqrt{x^2+y^2} \in \mathbb{R} \implies x, y \in \mathbb{R} \\ 0; \text{else} \end{cases}$$

$$= \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}} = \frac{1}{2\pi} \exp\left(-\frac{1}{2}(x^2+y^2)\right)$$

With this we prove that the Box-Muller generates samples from the 2D Standard normal distribution

1.2.2 Box-Muller implementation

a) For the implementation, see `box_muller_method.py`

b) Visualization of 1000 samples using the implemented method.

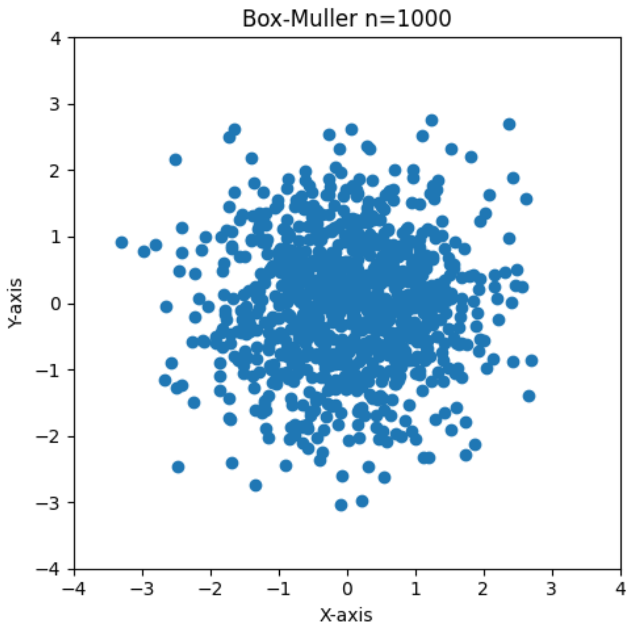

Figure 4: Box-Muller scheme with sample size of n=1000

c) Maximum Likelihood Estimators of Marginals with n=1000

$$\mathcal{N}_j(\mu_j, \Sigma_j), \quad j = 1, 2$$

$$\mu_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} \frac{1}{100} & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 5 & 6 \\ 6 & 9 \end{bmatrix}$$

Estimates Normal 1:

$$\hat{\mu}_1 = \begin{bmatrix} -1.0009 \\ 0.9569 \end{bmatrix} \quad \hat{\sigma}_1^2 = \begin{bmatrix} 0.0101 \\ 0.9388 \end{bmatrix}$$

Estimates Normal 2:

$$\hat{\mu}_2 = \begin{bmatrix} -0.0198 \\ -0.0816 \end{bmatrix} \quad \hat{\sigma}_2^2 = \begin{bmatrix} 5.0333 \\ 9.0176 \end{bmatrix}$$

The estimated marginal expected values and variances closely approximate the true means and variances, with small deviations due to sampling randomness. If we increase the number of generated samples the error gets even smaller.

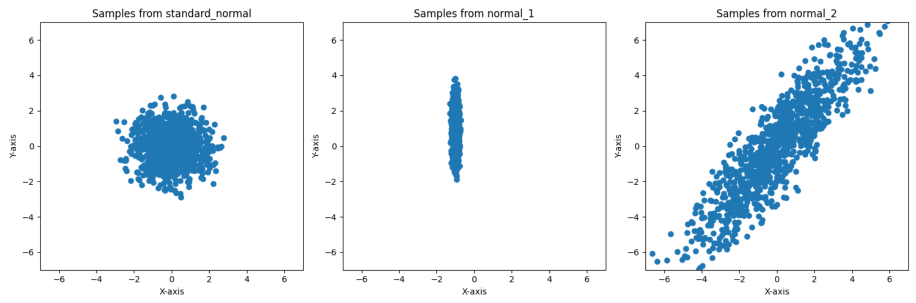

Figure 5: Plots of the 2D standard normal distribution, Normal 1 and Normal 2

1.2.3 Marsaglia-Bray

a)

$U, V \sim U(0,1)$
$\bar{U}, \bar{V} \sim U(-1,1)$
We reject the points outside of the unit circle and then we can express $\bar{u}$ and $\bar{v}$ with s and θ
$s = \bar{u}^2 + \bar{v}^2$
$s < 1$
$(\bar{u}, \bar{v}) = (\sqrt{s}\cos\theta, \sqrt{s}\sin\theta)$

$$f_{\bar{U}}(\bar{u}) = \begin{cases} \frac{1}{2}, & \text{if } -1 \leq \bar{u} \leq 1, \\ 0, & \text{else} \end{cases}$$

$$f_{\bar{V}}(\bar{v}) = \begin{cases} \frac{1}{2}, & \text{if } -1 \leq \bar{v} \leq 1, \\ 0, & \text{else} \end{cases}$$

The PDF of $(\bar{U}, \bar{V})$ gives us uniformly distributed points on a [-1,1]x[-1,1] square and is:

$$f_{\bar{U},\bar{V}}(u,v) = f_{\bar{U}}(u) \cdot f_{\bar{V}}(v) = \left\{ \begin{array}{l} \frac{1}{4}; -1 < u, v < 1 \\ 0;\ \text{else} \end{array} \right.$$

We then show that also (S, θ) are uniformly distributed

$$\theta = \arctan\left(\frac{\bar{v}}{\bar{u}}\right)$$

$$\frac{\bar{v}}{\sqrt{s}} = \sin\theta$$

$$\frac{\bar{u}}{\sqrt{s}} = \cos\theta$$

$$f_{S,\theta}(s,\theta) = f_{\bar{U},\bar{V}}(\sqrt{s}\cdot\cos\theta, \sqrt{s}\sin\theta) \cdot \left| \begin{array}{cc} \frac{\partial s}{\partial \bar{u}} & \frac{\partial s}{\partial \bar{v}} \\ \frac{\partial \theta}{\partial \bar{u}} & \frac{\partial \theta}{\partial \bar{v}} \end{array} \right|$$

$$= \frac{1}{4} \cdot \left| \begin{array}{cc} 2\bar{u} & 2\bar{v} \\ -\frac{\bar{v}}{\bar{u}^2+\bar{v}^2} & \frac{\bar{u}}{\bar{u}^2+\bar{v}^2} \end{array} \right| = \frac{1}{4}\left(\frac{2\bar{u}^2}{\bar{u}^2+\bar{v}^2} + \frac{2\bar{v}^2}{\bar{u}^2+\bar{v}^2}\right) = \frac{1}{2}$$

We then normalize the PDF of S and θ to the definition intervals $S \in (0,1)$ and $\theta \in (0, 2\pi)$ on the unit disk.

$$\int_0^1 \int_0^{2\pi} c \cdot f_{S,\theta}(s,\theta) ds d\theta = 1$$

$$\frac{c}{2}\int_0^1 \int_0^{2\pi} ds d\theta = \frac{c}{2} 2\pi = c\pi$$

$$c = \frac{1}{\pi}$$

$$f_{S,\theta\ \text{norm}}\ (s,\theta) = \frac{1}{2\pi}$$

We can see that now S is uniformly distributed between 0 and 1, corresponding to the U from the original Box–Muller method, and θ corresponds to W from the original algorithm. The cosine and sine are replaced by $\bar{u}/\sqrt{s}$ and $\bar{v}/\sqrt{s}$ respectively thus avoiding calculating trigonometric functions entirely.

b) For the implementation, see `box_muller_method.py`

c) Maximum Likelihood Estimators of Marginals (Marsaglia-Bray Scheme) Let the samples be drawn from the distributions

$$\mathcal{N}_j(\mu_j, \Sigma_j), \quad j = 1, 2$$

$$\mu_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} \frac{1}{100} & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 5 & 6 \\ 6 & 9 \end{bmatrix}$$

Estimates Normal 1:

$$\hat{\mu}_1 = \begin{bmatrix} -0.9932 \\ 0.9329 \end{bmatrix} \quad \hat{\sigma}_1^2 = \begin{bmatrix} 0.0095 \\ 1.0145 \end{bmatrix}$$

Estimates Normal 2:

$$\hat{\mu}_2 = \begin{bmatrix} 0.1516 \\ 0.0919 \end{bmatrix} \quad \hat{\sigma}_2^2 = \begin{bmatrix} 4.7652 \\ 8.2434 \end{bmatrix}$$

The estimates here have a larger error especially for Normal 2 compared to the Box-Muller method

2 Monte-Carlo estimation

2.1 Proof

The law of large numbers states that for a distribution X with expected value μ, the expected value can be approximated using i.i.d. samples from the distribution as follows:

$$\begin{aligned}
\mu &= E(X) \\
x_i &\overset{iid}{\sim} X, i = 1, ..., n \\
\bar{X_n} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\
\bar{X_n} &\to \mu \text{ as } n \to \infty
\end{aligned}$$

Substituting X with $f(Y)$ gives the required proof for the Monte-Carlo estimation:

$$\begin{aligned}
X &= f(Y) \\
\mu = E(X) &= E(f(Y)) \\
y_i &\overset{iid}{\sim} Y, i = 1, ..., n \\
x_i = f(y_i) &\overset{iid}{\sim} f(Y), i = 1, ..., n \\
\bar{X_n} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}\sum_{i=1}^{n} f(y_i) &= M_n \\
M_n &\to \mu \text{ as } n \to \infty
\end{aligned}$$

This shows that M_n gives a good approximation for $E(f(Y))$ as long as the samples y_i are identically and independently distributed.

2.2 Implementation

For the implementation, see `monte_carlo_method.py`.

2.3 Draw samples

We can see from the array Σ that X_1 and X_2 are independent. Thus we can calculate the exact expectation and variance this way:

$$\begin{aligned}
X_1 &\sim N(2, 2) & \mathbb{E}[X_1] &= 2 & \text{Var}[X_1] &= 2^2 = 4 \\
X_2 &\sim N(3, 1) & \mathbb{E}[X_2] &= 3 & \text{Var}[X_2] &= 1^2 = 1
\end{aligned}$$

$$\begin{aligned}\mathbb{E}[f(X_1, X_2)] &= \mathbb{E}[2X_1 + X_2] \\ &= 2\mathbb{E}[X_1] + \mathbb{E}[X_2] \\ &= 2 \cdot 2 + 3 \\ &= 7\end{aligned}$$

$$\begin{aligned}\operatorname{Var}[f(X_1, X_2)] &= \operatorname{Var}[2X_1 + X_2] \\ &= 4\operatorname{Var}[X_2] + \operatorname{Var}[X_2] - \operatorname{cov}[X_1, X_2] \\ &= 4 \cdot 4 + 1 - 0 \\ &= 17\end{aligned}$$

Running `monte_carlo_method.py` with different sample sizes gives the following results:

10^3 :

```
expected values:
 > box muller: 7.3085981
 > marsaglia bray: 7.0336766
variances:
 > box muller: 16.3357503
 > marsaglia bray: 16.8717241
```

10^4 :

```
expected values:
 > box muller: 7.0059634
 > marsaglia bray: 7.0354796
variances:
 > box muller: 17.2427512
 > marsaglia bray: 17.2470814
```

10^5 :

```
expected values:
 > box muller: 7.0173744
 > marsaglia bray: 7.0066446
variances:
 > box muller: 16.8803746
 > marsaglia bray: 16.9953122
```

One can see that the estimated values are close to the true expected value and variance. They get increasingly more accurate as n grows.

2.4 Variance reduction

2.4.1 Derivation

$$T = Y + a(Z - E[Z])$$

$$\begin{aligned}
\operatorname{Var}[T] &= \operatorname{Var}[Y + a(Z - E[Z])] \\
&= \operatorname{Var}[Y] + \operatorname{Var}[a(Z - E[Z])] + 2\operatorname{cov}[Y, a(Z - E[Z])] \\
&= \operatorname{Var}[Y] + a^2 \operatorname{Var}[Z] + 2a \operatorname{cov}[Y, Z]
\end{aligned}$$

Using the following equivalences:

$$\begin{aligned}
\operatorname{Var}[a(Z - E[Z])] &= a^2 \operatorname{Var}[Z - E[z]] \\
&= a^2(\operatorname{Var}[Z] + \operatorname{Var}[-E[z]] - \operatorname{Cov}[Z, -E[Z]]) \\
&- a^2(\operatorname{Var}[Z] - \operatorname{cov}[Z, -E[Z]]) \\
&= a^2(\operatorname{Var}[Z] - (E[-ZE[Z]] - E[Z]E[-E[Z]])) \\
&= a^2(\operatorname{Var}[Z] - (-E[Z]E[Z] + E[Z]E[Z])) \\
&= a^2 \operatorname{Var}[Z]
\end{aligned}$$

$$\begin{aligned}
\operatorname{cov}[Y, a(Z - E[Z])] &= E\left[Ya(Z - E[Z])\right] - E[Y]E[a(Z - E[Z])] \\
&= aE[YZ - YE[Z]] - aE[Y]E[Z - E[Z]] \\
&= a(E[YZ] - E[Y]E[Z]) - aE[Y](E[Z] - E[Z]) \\
&= a \operatorname{cov}[Y, Z]
\end{aligned}$$

Getting the minimum using the derivative:

$$\begin{aligned}
\frac{d \operatorname{Var}[T]}{da} = 2a \operatorname{Var}[Z] + 2\operatorname{cov}[Y, Z] = 0 \\
2a \operatorname{Var}[Z] = -2\operatorname{cov}[Y, Z] \\
a = -\frac{\operatorname{cov}[Y, Z]}{\operatorname{Var}[Z]}
\end{aligned}$$

Substituting in the original formula:

$$\begin{aligned}
\mathrm{Var}[T] &= \mathrm{Var}[Y] + \frac{\mathrm{cov}[Y,Z]^2}{\mathrm{Var}[Z]^2}\mathrm{Var}[Z] - 2\frac{\mathrm{cov}[Y,Z]}{\mathrm{Var}[Z]}\mathrm{cov}[Y,Z] \\
&= \mathrm{Var}[Y] + \frac{\mathrm{Cov}[Y,Z]^2}{\mathrm{Var}[Z]} - \frac{2\,\mathrm{Cov}[Y,Z]^2}{\mathrm{Var}[Z]} \\
&= \mathrm{Var}[Y] - \frac{\mathrm{Cov}[Y,Z]^2}{\mathrm{Var}[Z]} \\
&= \mathrm{Var}[Y]\left(1 - \frac{\mathrm{Cov}[Y,Z]^2}{\mathrm{Var}[Y]\,\mathrm{Var}[Z]}\right) \\
&= \mathrm{Var}[Y]\left(1 - \mathrm{Corr}[Y,Z]\right)
\end{aligned}$$

So we are looking for a distribution Z that has high correlation with Y.

2.4.2 Implementation

To implement the variance reduction, we first calculated a few properties of the distributions:

$$\begin{aligned}
\mathbb{E}[Z] &= \mathbb{E}[h(U)] \\
&= \int_0^2 \frac{x}{2}dx + \int_{1/2}^1 \frac{1-x}{2}dx \\
&= \left[\frac{x^2}{4}\right]_{x=0}^{1/2} + \left[\frac{x}{2} - \frac{x^2}{4}\right]_{1/2}^1 \\
&= \frac{1}{16} + \frac{1}{2} - \frac{1}{4} - \frac{1}{4} + \frac{1}{16} = \frac{1}{8}
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}[Z] &= \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 \\
&= \int_0^1 h(x)dx - \left(\frac{1}{8}\right)^2 \\
&= \int_0^{1/2} \left(\frac{x}{2}\right)^2 dx + \int_{1/2}^1 \left(\frac{1-x}{2}\right)^2 dx - \frac{1}{64} \\
&= \int_0^{1/2} \frac{x^2}{4}dx + \int_{1/2}^1 \frac{1-2x+x^2}{4}dx - \frac{1}{64} \\
&= \frac{1}{96} + \frac{1}{4} - \frac{1}{8} - \frac{1}{4} + \frac{1}{8} + \frac{1}{12} - \frac{1}{96} - \frac{1}{64} \\
&= \frac{1}{48} - \frac{1}{64}
\end{aligned}$$

The exact expected value would be:

$$\begin{aligned}
\mathbb{E}[g(U)] &= \mathbb{E}[U(1-U)] \\
&= \mathbb{E}\left[U - U^2\right] \\
&= \mathbb{E}[U] - \mathbb{E}\left[U^2\right] \\
&= \int_0^1 x dx - \int_0^1 x^2 dx \\
&= \frac{1}{2} - \frac{1}{3} = \frac{1}{6}
\end{aligned}$$

For the implementation, see `monte_carlo_method.py`.

2.4.3 Draw samples

We can observe that the monte carlo estimation with variance reduction approximates the expected value better than the vanilla monte carlo, especially with smaller sample sizes:

10^2 **samples:**

```
expected value
 > exact: 0.1666667
 > vanilla monte-carlo: 0.1688122
 > control variate monte-carlo: 0.1661726
```

10^3 **samples:**

```
expected value
 > exact: 0.1666667
 > vanilla monte-carlo: 0.1668933
 > control variate monte-carlo: 0.1668957
```

10^5 **samples:**

```
expected value
 > exact: 0.1666667
 > vanilla monte-carlo: 0.1667755
 > control variate monte-carlo: 0.1666999
```

3 Bayesian image denoising

3.1 Kernel density estimation

3.1.1 Proof

We want to prove that the following is a probability density function:

$$f_h(x) = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{h}\right)^{d} k\left(\frac{x-x_j}{h}\right)$$

For this to hold, we need to prove two things:

$$\forall x \in \mathbb{R}^d : f_h(x) \geq 0$$
$$\int_{\mathbb{R}^d} f_n(x) = 1$$

Proving nonnegativity using nonnegativity of k:

$$\begin{aligned}
k(u) &\geq 0 && \forall u \in \mathbb{R}^d \\
k\left(\frac{x-x_j}{h}\right) &\geq 0 && u = \frac{x-x_j}{h} \\
\sum_{j=1}^{n} k\left(\frac{x-x_j}{h}\right) &\geq 0 && \\
\sum_{j=1}^{n}\left(\frac{1}{h}\right)^{d} k\left(\frac{x-x_j}{h}\right) &\geq 0 && h > 0 \\
\frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{h}\right)^{d} k\left(\frac{x-x_j}{h}\right) &\geq 0 && n > 0 \\
f_h(x) &\geq 0 && \forall x \in \mathbb{R}^d
\end{aligned}$$

Proving integral of f is 1 using that integral of k is 1:

$$\begin{aligned}
\int_{\mathbb{R}^d} f_h(x) &= \int_{\mathbb{R}^d} \frac{1}{n} \sum_{j=1}^{n} \left(\frac{1}{h}\right)^d k\left(\frac{x - x_j}{h}\right) dx \\
&= \frac{1}{n} \int_{\mathbb{R}^d} \sum_{j=1}^{n} \left(\frac{1}{h}\right)^d k\left(\frac{x - x_i}{h}\right) dx \\
&= \frac{1}{n} \sum_{j=1}^{n} \int_{\mathbb{R}_d} \left(\frac{1}{h}\right)^d k\left(\frac{x - x_j}{h}\right) dx \\
&= \frac{1}{h} \sum_{j=1}^{n} \int_{-\infty}^{\infty} \frac{1}{h} \cdots \int_{-\infty}^{\infty} \frac{1}{h} \int_{-\infty}^{\infty} \frac{1}{h} k\left(\frac{x - x_j}{h}\right) dx_{(1)} dx_{(2)} \cdots dx_{(d)} \\
&= \frac{1}{n} \sum_{j=1}^{n} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(u) du_{(1)} du_{(2)} du_{(d)} \\
&= \frac{1}{n} \sum_{j=1}^{n} 1 \\
&= 1
\end{aligned}$$

Repeatedly using the substitution method for each dimension:

$$\int g'(x) f(g(x)) dx = \int f(u) du$$

$$\frac{d}{dx} \frac{x - x_j}{h} = \frac{1}{h}$$

3.1.2 Impact of bandwidth

You can see the visualization of the bandwidth impact in Figure 6. The larger the bandwidth, the bigger the area that the KDE smooths over.

In the limit $h \to 0^+$, the KDE becomes a dirac-delta function with spikes at each datapoint, each with an area of $\frac{1}{n}$.

In the case that $h \to \infty$, the KDE becomes a constant function where $\forall x, y \in \mathbb{R}^d : f_h(x) = f_h(y)$.

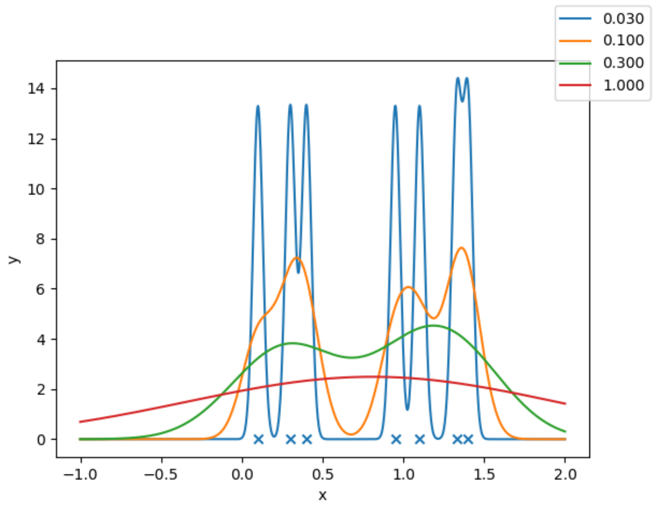

Figure 6: The impact of different bandwidths on the resulting KDE

3.2 Denoising MNIST data

3.2.1 Pen and paper proof

$$Y = X + Z \quad Z \sim \mathcal{N}\left(0, \sigma^2\right)$$

$$f_{Y/X=x}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(\left(-\frac{1}{2\sigma^2}\right)(y-x)^2\right)$$

$$P(Y \leq y \cap X = x) = P(Z \leq y - x \cap X = x)$$

$$\overset{Z \perp\!\!\!\perp X}{=} P(X = x) \cdot P(Z \leq y - x)$$

$$P(Y \leqslant y \mid X = x) = \frac{P(Y \leqslant y \cap X = x)}{P(X = x)} =$$

$$= \frac{P(X = x) \cdot P(Z \leq y - x)}{P(X = x)} =$$

$$= P(Z \leq y - x) \Rightarrow P(Z + x \leq y)$$

$Y \mid X = x$ has the same distribution as $Z + x$ and since Z is a standard normal distribution $\mathcal{N}(0, \sigma)$ to which we then add a constant x. We are just shifting the mean by x which means that $Y \mid X = x$ is equal to $\mathcal{N}(x, \sigma)$ thus it's PDF is:

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

which is the some as the given PDF for $d = 1$ TODO add

3.2.2 Pen and paper

Here we verify analytically that the log of the PDF of $f_{Y/X=x}(y)$ is the same as the given formula.

$$f_{Y/X=x}(y) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \cdot \exp\left(-\frac{\|y-x\|_2^2}{2\sigma^2}\right)$$

$$\log\left(f_{Y|X=x}(y) =\right.$$

$$= \log\left(\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \cdot \exp\left(-\frac{\|y-x\|_2^2}{2\sigma^2}\right)\right)$$

$$= \log\left(\left(2\pi\sigma^2\right)^{-\frac{d}{2}}\right) + \log\left(\exp\left(-\frac{\|y-x\|_2^2}{2\sigma^2}\right)\right.$$

$$= -\frac{d}{2}\log\left(2\pi\sigma^2\right) - \frac{\|y-x\|_2^2}{2\sigma^2}$$

Here we verify analytically that the log of $p_h(x)$ is the same as the given formula.

$$p_h(x) = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{h}\right)^d \frac{1}{\sqrt{(2\pi)^d}} \exp\left(-\frac{1}{2h^2}\left\|x-x_j\right\|_2^2\right)$$

$$\log((p_h(x)) = \log\left(\frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{h}\right)^d \frac{1}{\sqrt{(2\pi)^d}} \exp\left(-\frac{1}{2h^2}\left\|x-x_j\right\|_2^2\right)\right).$$

$$= \underbrace{-\log(n) - d\log(h) - \frac{d}{2}\log(2\pi)}_{\text{constant w.r.t j}} + \log\left(\sum_{j=1}^{n}\exp\left(-\frac{\left\|x-x_j\right\|_2^2}{2h^2}\right)\right)$$

$$= \log\left(\exp\left(-\log(n) - d\log(h) - \frac{d}{2}\log(2\pi)\right)\right) + \log\left(\sum_{j=1}^{n}\exp\left(-\frac{\left\|x-x_j\right\|_2^2}{2h^2}\right)\right)$$

$$= \log\left(\sum_{j=1}^{n}\exp\left(-\log(n) - d\log(h) - \frac{d}{2}\log(2\pi) - \frac{\left\|x-x_j\right\|_2^2}{2h^2}\right)\right)$$

3.2.3 Implementation

The functions *log_gaussian_kde()* and *log_likelihood()* are implemented in the file `bayesian_denoising.py`.

3.2.4 Application to sign language MNIST

The method is effective across all noise levels. The most notable improvement occurs when $\sigma = 0.25$, where the hands in the original image are barely visible due to high noise, but become clearly recognizable after denoising.

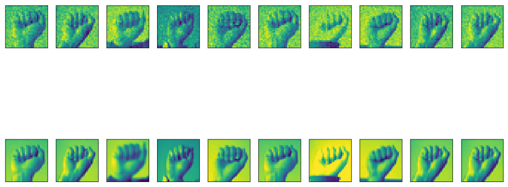
Figure 7: Noise level: $\sigma = 0.05$

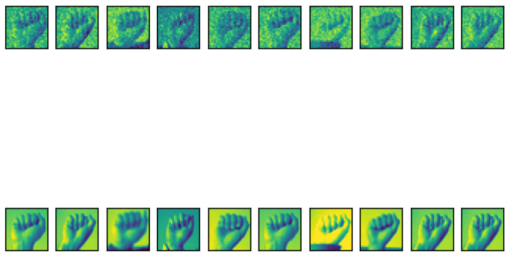
Figure 8: Noise level: $\sigma = 0.1$

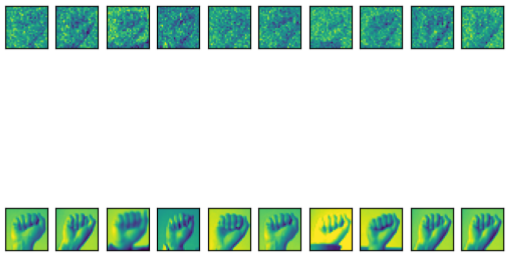
Figure 9: Noise level: $\sigma = 0.25$