Introduction to Deep Learning M2177.0043
Seoul National University

Homework #**1**
Donghak Lee 65

INSTRUCTIONS

- Anything that is received after the deadline will be considered to be late and we do not receive late homeworks. We do however ignore your lowest homework grade.
- Answers to every theory questions need to be submitted electronically on ETL. Only PDF generated from LaTex is accepted.
- Make sure you prepare the answers to each question separately. This helps us dispatch the problems to different graders.
- Collaboration on solving the homework is allowed. Discussions are encouraged but you should think about the problems on your own.
- If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution.

1 Learning LaTex

1.1

Donghak	Lee
Computer Science and Engineering	2017-12751

1.2

2 Inverse transform sampling

2.1 Probability integral transform

By properties of pdf, $F_X(x)$ is invertible function with range (0, 1). So $\forall y \in (0,1)$,

$$F_Y(y) = P(Y \leq y) = P(F_X(X) \leq y) = P(X \leq F_X^{-1}(y)) = F_X(F_X^{-1}(y)) = y$$

$$f_Y(y) = F'_Y(y) = 1$$

Therefore, random variable Y is uniformly distributed in [0, 1].

Introduction to Deep Learning M2177.0043
Seoul National University

Homework #**1**
Donghak Lee

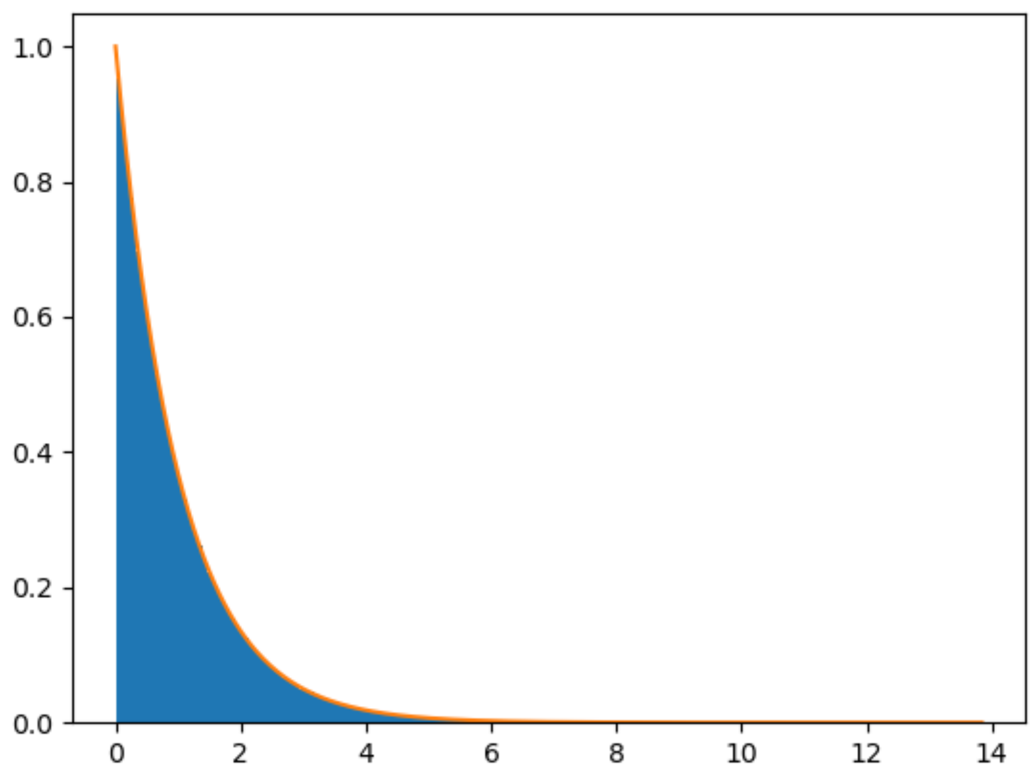
2.2 Inverse transform sampling

Because random variable U is uniformly distributed in [0, 1], pdf $f_U(u) = 1$. So,

$$P(F_X^{-1}(U) \leq x) = P(F_X(F_X^{-1}(U)) \leq F_X(x)) = P(U \leq F_X(x)) = F_X(x)$$

Therefore, cdf of $F_X^{-1}(U)$ is $F_X(x)$

2.3




```
import numpy as np
import matplotlib.pyplot as plt
samples = np.random.exponential(1, 1000000)
y, x, p = plt.hist(samples, bins=500, density=True)
fx = np.linspace(min(x), max(x), 500)
fy = np.exp(-fx)
plt.plot(fx, fy)
plt.show()
```

⇒ didn't sample from uniform (−10)

Introduction to Deep Learning M2177.0043
Seoul National University

Homework #**1**
Donghak Lee

---

3 Optimal hedge ratio

Assume that I own x shares of stock B, variance of total stock is

$$V(A+xB) = V(A) + x^2V(B) + 2xCov(A,B) = \sigma_A^2 + x^2\sigma_B^2 + 2x\sigma_A\sigma_B\rho$$

To minimize variance of total stock,

$$\frac{dV}{dx} = 2x\sigma_B^2 + 2\sigma_A\sigma_B\rho = 0$$

check $\frac{d^2}{dx^2}$ (-5)

$$\Rightarrow x = -\frac{\sigma_A\rho}{\sigma_B}$$

Therefore, the number of share I have to sell n is

$$\begin{cases} n = x & (\rho > 0) \\ n = x + \frac{\sigma_A\rho}{\sigma_B} & (\rho \leq 0) \end{cases}$$

4 Eigenvalues

$$X = \begin{pmatrix} 1 & c & \cdots & c \\ c & 1 & \cdots & c \\ \cdots & \cdots & \cdots & \cdots \\ c & c & \cdots & 1 \end{pmatrix} = \begin{pmatrix} c & c & \cdots & c \\ c & c & \cdots & c \\ \cdots & \cdots & \cdots & \cdots \\ c & c & \cdots & c \end{pmatrix} + \begin{pmatrix} 1-c & 0 & \cdots & 0 \\ 0 & 1-c & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1-c \end{pmatrix} = C + (1-c)I$$

$$Rank(C) = 1 \Rightarrow dim(N(C)) = p - 1$$

$X - \lambda I = C$, so there are p-1 eigenvalues $\lambda_i = 1 - c$.
Since $Tr(X) = p$, the last eigenvalue $\lambda_p = p - (1-c)(p-1) = 1 + (p-1)c$, which have eigenvector $x_p = (1, 1, \cdots, 1)^T$

5 Multivariate normal

5.1

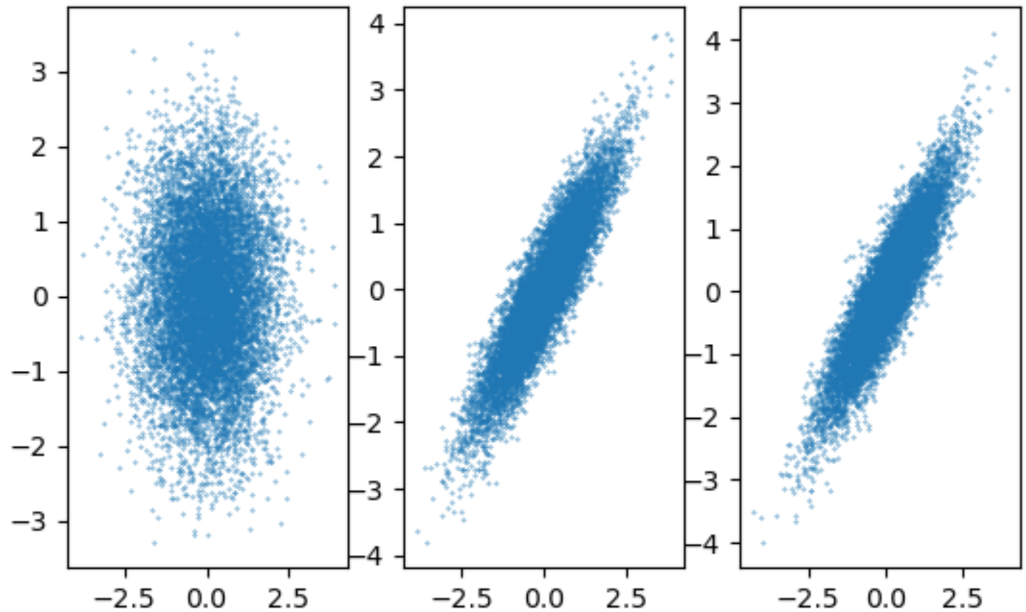
Σ is symmetric, so it can be decomposed to $\Sigma = QDQ^T = SS^T (S = QD^{\frac{1}{2}})$.
Let $X = SZ + \mu$,

$$E(X) = \mu + SE(Z) = \mu$$

$$V(X) = E((X-\mu)(X-\mu)^T) = E(SZZ^TS^T) = SE(ZZ^T)S^T = SIS^T = SS^T = \Sigma$$

Therefore, $X \sim N(\mu, \Sigma)$

5.2




```python
import numpy as np
import matplotlib.pyplot as plt
import math
np.random.seed(1337)
Z = np.random.normal(0, 1, (2, 10000))
sigma = np.array([[math.sqrt(1.9/2), math.sqrt(0.1/2)],
[math.sqrt(1.9/2), -math.sqrt(0.1/2)]])
X = np.matmul(sigma, Z)
mean = [0.0, 0.0]
cov = [[1.0, 0.9],[0.9, 1.0]]
MN = np.random.multivariate_normal(mean, cov, 10000).T
plt.subplot(1, 3, 1)
plt.scatter(Z[0], Z[1], s=0.1)
plt.subplot(1, 3, 2)
plt.scatter(X[0], X[1], s=0.1)
plt.subplot(1, 3, 3)
plt.scatter(MN[0], MN[1], s=0.1)

plt.show()
```

Just use np.linalg package

Introduction to Deep Learning M2177.0043
Seoul National University

Homework #**1**
Donghak Lee

---

6 Optimization with positivity constraints

Let x_i = i-th element of vector x and $v = x - x^*$. For $C = R^n_+$, v that minimize $\nabla f(x^*)^T v$ is

$$v_i = \begin{cases} -\nabla f(x^*)_i & (x_i^* \neq 0) \\ Max(-\nabla f(x^*)_i, 0) & (x_i^* = 0) \end{cases}$$

Therefore, simplified local minimum solution x^* is

No proof (-20)

$$\begin{cases} \nabla f(x^*)_i = 0 & (x_i^* \neq 0) \\ \nabla f(x^*)_i \geq 0 & (x_i^* = 0) \end{cases}$$