CSE546 HW0

Haoxin Luo

December 8, 2022

Exercise 1

After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a serious disease, and that the test is 99 percent accurate (i.e., the probability of testing positive given that you have the disease is 0.99, as is the probability of testing negative given that you don't have the disease). The good news is that this is a rare disease, striking only one in 10,000 people. What are the chances that you actually have the disease?

Let P be the event of test positive, D be the event of getting disease. By the Bayes' rule:

$$P(D) = 0.0001$$

$$P(P|D) = P(!P|!D) = 0.99$$

Therefore, $P(D|P) = \frac{P(P|D)P(D)}{P(P)} = \frac{P(P|D)P(D)}{P(P|D)P(D)+P(P|!D)P(!D)} = \frac{0.99 \cdot 0.0001}{0.99 \cdot 0.0001+0.01 \cdot 0.9999} \approx 0.0098$.

Exercise 2

For any two random variables X, Y (You may assume X and Y take on a discrete values if you find that is easier to work with).

a). We know that $E[Y|X = x] = x$, and $E[E(Y|X)] = E(X)$ so $E(Y) = E(X)$.
$Cov(X, Y) = E((X - E[X])(Y - E[Y])$
$= E((XY - YE[X] - XE[Y] + E[X]E[Y])$
$= E[XY] - E[X]E[Y]$
$= E[E[XY|X = x] - E(X)^2$
$= E[XE(Y|X)] - E[X]^2$
$= E(X^2) - E(X)^2$
Therefore, $Cov(X, Y) = E[(X - E(X)^2]$

b). X, Y are independent so $E[XY] = E[X]E[Y]$, and we know from part a that $Cov(X, Y) = E[XY] - E[X]E[Y] = 0$

Exercise 3

Let X and Y be independent random variables with PDFs given by f and g, respectively. Let h be the PDF of the random variable Z = X + Y .

a).

$F_z(z) = P(X + Y \leq z) = \int_{(x+y)\leq z} f(x)g(y)dydx = \int_{-\infty}^{\infty} f(x)[\int_{-\infty}^{z-x} f_{xy}(x,y)dy]dx$
$= \int_{-\infty}^{\infty} f(x)P(Y \leq z - x)dx$
$f_z(z) = \frac{d}{dz}F_z(z) = \frac{d}{dz}\int_{-\infty}^{\infty} F_y(z-x)f_x dx$
$h(z) = \int_{-\infty}^{\infty} f(x)g(z-x)dx$

In fact, if we draw a picture, we can also find the pdf of Z is the sum of two random variables given by the superposition integral when X and Y are independent random variables. The pdf of Z is given by the convolution integral of the marginal pdfs of X and Y.

b).

From the previous question, we know that $h(z) = \int_{-\infty}^{\infty} f(x)g(z-x)dx$, and in our case, $h(z) = \int_0^1 f(x)g(z-x)dx$. As the X and Y are all normally distributed on the interval [0,1], and Z is the sum of X and Y, so $Z \in [0, 2]$. There are two possible cases:

1. $z \in [0, 1]$, we have $z - x \geq 0$: $h(z) = \int_0^z g(z-x)dx + \int_z^1 g(z-x)dx = z$

2. $z \in [1, 2]$, we have $z - x \leq 1$: $h(z) = \int_0^{z-1} g(z-x)dx + \int_{z-1}^1 g(z-x)dx = 2 - z$

As a result:

$$h(z) = \begin{cases} 0, & \text{if } z < 0 \\ z, & \text{if } 0 \leq x \leq 1 \\ 2 - z, & \text{if } 1 < x \leq 2 \\ 1, & \text{otherwise} \end{cases}$$

Exercise 4

Let X1, X2, ..., Xn be i.i.d normally distributed random variables. Compute the following:

a).

Let $X \sim N(\mu, \sigma^2)$, Let$Y = aX + b$, then by properties of linear transformation, for any constants a and b, $aX + b \sim N(a\mu + b, a^2\sigma^2)$.

Since now, $a_1X + b \sim N(0, 1)$. Then, $a^2\sigma^2 = 1$ indicates $a = \frac{1}{\sigma}$, and $a\mu + b = 0$ indicates $b = -\frac{\mu}{\sigma}$

b). X's are all independent, so:

$E[X_1 + 2X_2] = E[X_1] + 2E[X_2] = \mu_1 + 2\mu_2 = 3\mu$

$Var[X_1 + 2X_2] = Var[X_1] + 4Var[X_2] + 0 = \sigma_1^2 + 4\sigma_2^2 = 5\sigma^2$

c).

$E[\sqrt{n}(\hat{\mu} - \mu)] = \sqrt{n}E[\hat{\mu} - \mu]$
$\sqrt{n}(E[\hat{\mu}] - E[\mu])$
$\sqrt{n}(E[\frac{1}{n}\sum X_i] - E[\mu])$
$\sqrt{n}(E[X] - \mu) = \sqrt{n}(\mu - \mu) = 0$
And $Var[\sqrt{n}(\hat{\mu} - \mu)] = nVar[\hat{\mu} - \mu]$
$nVar(\hat{\mu}) = nVar(\frac{1}{n}\sum X_i)$
$\frac{1}{n} \times n \times Var(\sum X_i) = \sigma^2$
Thus, $\sqrt{n}(\hat{\mu} - \mu) \sim N(0, \sigma^2)$

Exercise 5

a). The rank of A is dim(col(A))=2, and B is dim(col(B))=2 (we need to find the matrix in row-echelon form)

$$A \to \begin{bmatrix} 1 & 2 & 1 \\ 1 & 0 & 3 \\ 1 & 1 & 2 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 1 & 0 & 3 \\ 0 & 1 & -1 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B \to \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 1 & 1 & 2 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 2 \\ 0 & 1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

b).

The basis for A's column span is the first 2 column. A has two pivot columns, as we can obtain column 3 from 3 times column1 -column 2

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} \right\}$$

The basis for B's column span is the first 2 column. B has two pivot columns, as we can obtain column 3 from column1 + column 2.

$$\left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Exercise 6

a).

We can calucluate by:

$$\begin{bmatrix} 0 & 2 & 4 \\ 2 & 4 & 2 \\ 3 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0+2+4 \\ 2+4+2 \\ 3+3+1 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 7 \end{bmatrix}$$

b).

We know that $x = A^{-1}b$, Compute A^{-1} first and than multiply it by b, we get:

$$\begin{bmatrix} 1/8 & -5/8 & 3/4 \\ -1/4 & 3/4 & -1/2 \\ 3/8 & -3/8 & 1/4 \end{bmatrix} \times \begin{bmatrix} -2 \\ -2 \\ -4 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ -1 \end{bmatrix}$$

Therefore,

$$x = \begin{bmatrix} -2 & 1 & -1 \end{bmatrix}^T$$

Exercise 7

a).

$$f(x,y) = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix} \begin{bmatrix} A_{1,1} & A_{1,2} & ... & A_{1,n} \\ A_{2,1} & A_{2,2} & ... & A_{2,n} \\ ... & .... & ... & ... \\ A_{n,1} & A_{n,2} & ... & A_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 & y_2 & ... & y_n \end{bmatrix} \begin{bmatrix} B_{1,1} & B_{1,2} & ... & B_{1,n} \\ B_{2,1} & B_{2,2} & ... & B_{2,n} \\ ... & .... & ... & ... \\ B_{n,1} & B_{n,2} & ... & B_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} + c$$

$$f(x,y) = \begin{bmatrix} \sum_i a_{i1}x_i & ... & \sum_i a_{in}x_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} + \begin{bmatrix} \sum_i b_{i1}y_i & ... & \sum_i b_{in}y_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} + c$$

$$f(x,y) = \sum_j^n \sum_i^n a_{ij}x_i x_j + \sum_j^n \sum_i^n b_{ij}y_i x_j + c$$

b).

$$\nabla_x f(x,y) = \begin{bmatrix} \frac{d}{dy_1} f(x,y) & ... & \frac{d}{dy_n} f(x,y) \end{bmatrix} = \frac{d}{dx_k}(\sum_{i\neq k}^n \sum_{j\neq k}^n A_{ij}x_i x_j + \sum_{i\neq k}^n x_i x_k + \sum_{j\neq k}^n A_{ij}x_k x_j + A_{kk}x_k^2) +$$

$$\frac{d}{dx_k}(x_1 \sum_i^n b_{i1}y_i + ... + x_n \sum_i^n b_{in}y_i) + 0 = \sum_j^n a_{nj}x_j + \sum_i^n a_{in}x_i + \sum_i^n b_{in}y_i$$

The vector form is then:

$$\nabla_x f(x,y) = (A + A^T)x + (y^T B)^T = (A + A^T)x + B^T y$$

c).

$$\nabla_y f(x,y) = \begin{bmatrix} \frac{d}{dy_1} f(x,y) & ... & \frac{d}{dy_n} f(x,y) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n b_{1_i x_i} & \sum_{i=1}^n b_{2_i x_i} & ... & \sum_{i=1}^n b_{n_i x_i} \end{bmatrix}^T = x^T B^T$$

Exercise 8

a).

To get $diag(v)^{-1}$ = diag(w) and $g(v_i) = w_i$, since we have a diagonal matrix which only have values on its diagonal and elsewhere zeros, $g = -x$

b).

Since A is orthogonal, the transpose of A will be the same as its inverse, and $A^T A = I$. Since $x \in R^n$ a real matrix, its conjugate transpose is just the transpose, $\|x\|^2 = x^T x$

$\|Ax\|_2^2 = (Ax)^T(Ax) = x^T(A^T A)x = x^T x = \|x\|_2^2$

c).

Since B is symmetric: $B = B^T$, and B is invertible so $B^{-1} = (B^T)^{-1} = (B^{-1})^T$
$BB^{-1} = I = (B^{-1})^T B^T = (B^{-1})^T B$
$(B^{-1})^T = IB^{-1} = (B^{-1})^T BB^{-1} = B^{-1}$

From the above, we have $(B^{-1})^T = B^{-1}$, B^{-1} is also symmetric.

d).

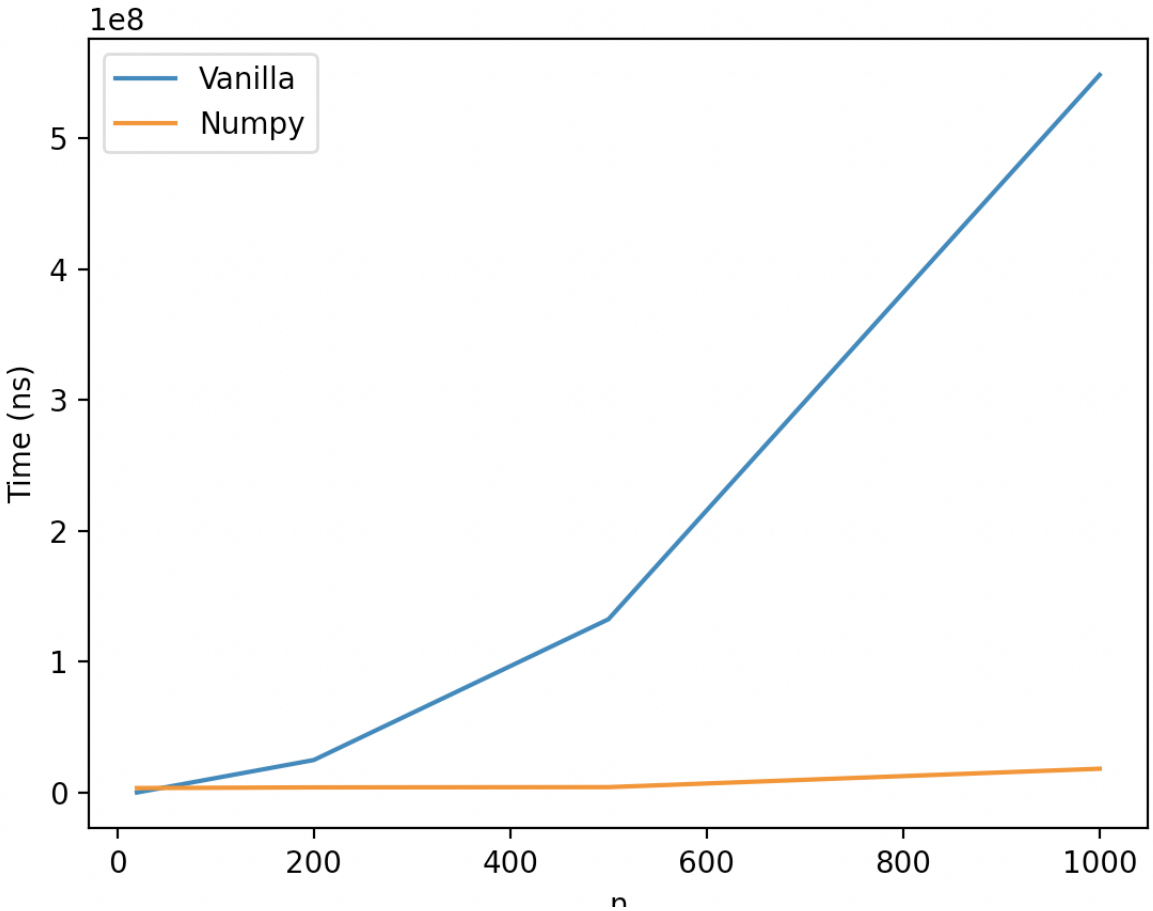
The formula for computing eigenvalue and eigenvector is $c\vec{x} = \lambda\vec{x}$, to meet the definition of a PSD matrix, we need $x^T Cx \geq 0$ for all x. Suppose x is a eigen vector of C, $\vec{x}^T c\vec{x} = \vec{x}^T \lambda\vec{x} = \vec{x}^T\vec{x}\lambda$. Then, by property of PSD, since $\vec{x}^T\vec{x} \geq 0$ its eigenvalues λ are non-negative.

Exercise 9

```python
def matrix_sum(A: Matrix, B:Matrix) -> Matrix:
    n = len(A)
    res: Matrix = [[0]*n for _ in range(n)]

    # iterate through rows
    for i in range(len(A)):
        # iterate through columns
        for j in range(len(A[0])):
            res[i][j] = A[i][j] + B[i][j]
    return res

def vector_sum(A: Vector, B:Vector) -> Vector:
    n = len(A)
    vec = [0]*n
    # iterate through one of the vector
    for i in range(len(A)):
        # sum up the corresponding elements
        vec[i] = A[i] + B[i]
    return vec
```

```python
def vanilla_solution(x: Vector, y: Vector, A: Matrix, B: Matrix) -> Vector:

    # raise NotImplementedError("Your Code Goes Here")
    return vector_sum(vanilla_matmul(matrix_sum(A, vanilla_transpose(A)), x),
    vanilla_matmul(vanilla_transpose(B), y))



@problem.tag("hw0-A")
def numpy_solution(
    x: np.ndarray, y: np.ndarray, A: np.ndarray, B: np.ndarray
) -> np.ndarray:
    """Calculates gradient of f(x, y) with respect to x using numpy arrays.
    Where $$f(x, y) = x^T A x + y^T B x + c$$

    Args:
        x (np.ndarray): a (n,) numpy array.
        y (np.ndarray): a (n,) numpy array.
        A (np.ndarray): a (n, n) numpy array.
        B (np.ndarray): a (n, n) numpy array.

    Returns:
        np.ndarray: a resulting (n, ) numpy array.

    Note:
        - Make use of numpy docs: https://numpy.org/doc/
            You will use this link a lot throughout quarter, so it might be a
    """
    # raise NotImplementedError("Your Code Goes Here")
    A_T = A.copy().transpose()
    B_T = B.copy().transpose()
    return np.dot(A + A_T, x) + np.dot(B_T, y)
```

```
(cse446) Haoxins-MacBook-Pro:homeworks coco$
(cse446) Haoxins-MacBook-Pro:homeworks coco$ /Users/coco/anaconda3/envs/cse446/b
in/python /Users/coco/Desktop/cse446/hw0-A/homeworks/vanilla_vs_numpy/vanilla_vs
_numpy.py
Time for vanilla implementation: 0.269ms
Time for numpy implementation: 3.659ms
Time for vanilla implementation: 25.071ms
Time for numpy implementation: 4.189ms
Time for vanilla implementation: 132.552ms
Time for numpy implementation: 4.371ms
Time for vanilla implementation: 548.153ms
Time for numpy implementation: 18.425ms
```



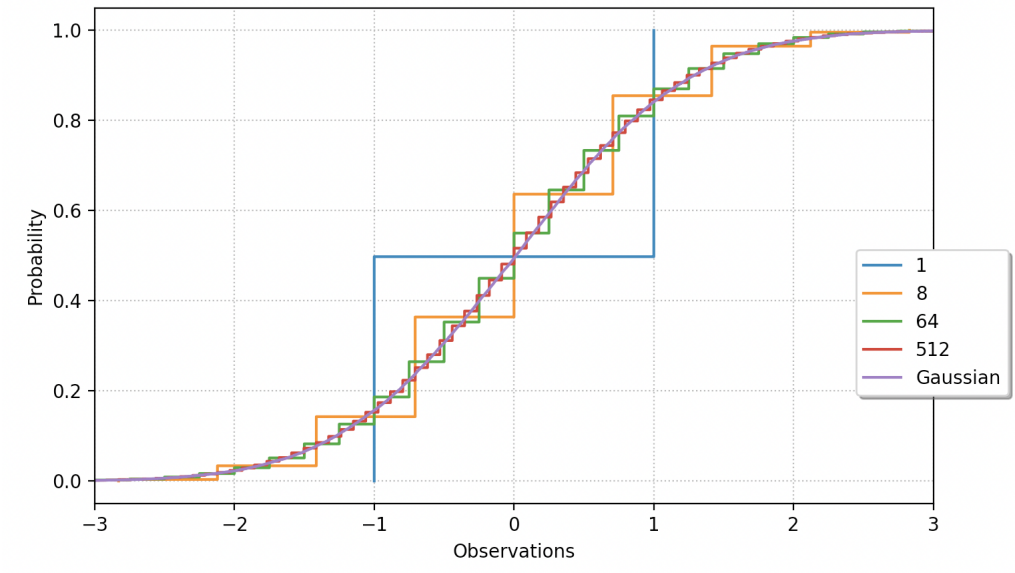
c). The difference in wall-clock time is shown from the picture above. The difference become much higher as the matrix size becomes large. The difference (3.39, 20.882, 128.181, 529.728)

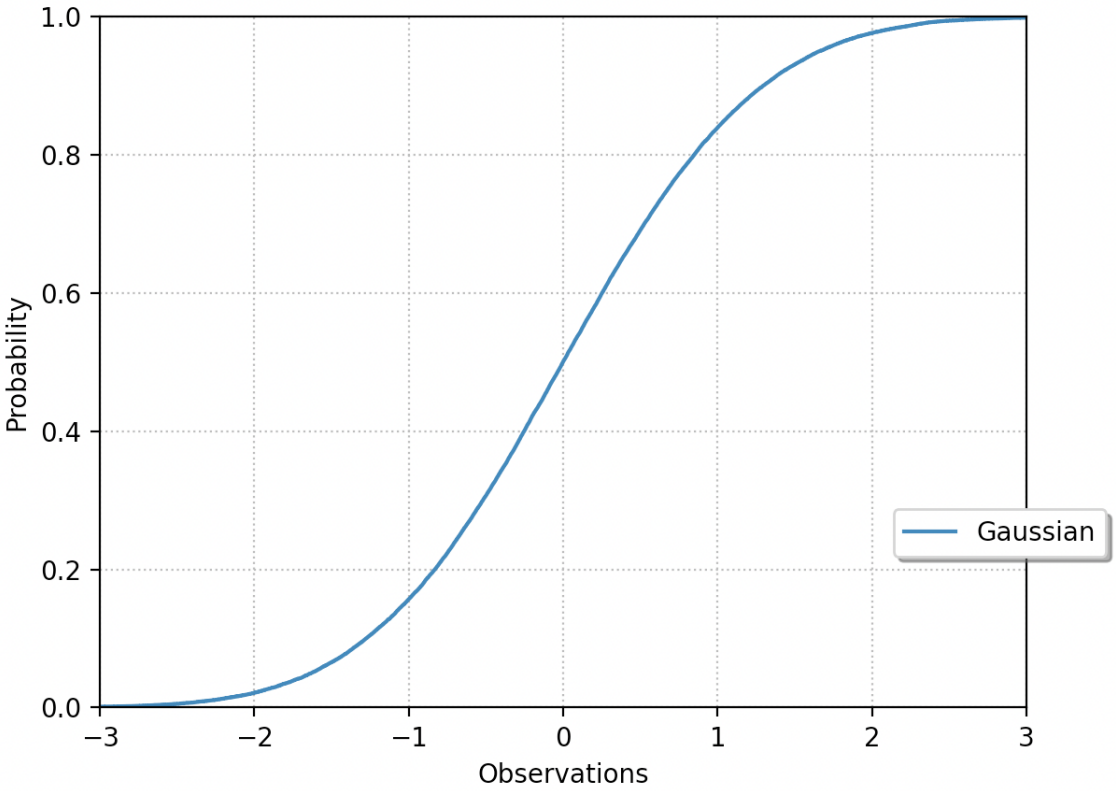
It seems that as the size of the input matrix increae, The time of vanilla goes up much higher, resulting a larger differences between the two. Operations on NumPy arrays are executed faster than Python lists because Numpy arrays are densely packed arrays of homogeneous type. But Python lists are arrays of pointers to objects, even when all of them are of the same type.

Exercise 10

a). We know that $E(\hat{F}_n(x)) = F(x)$ and $Var(\hat{F}_n(x)) = \frac{F(x)(1-F(x))}{n}$, take the derivative to get the optimal value of $\hat{F}_n(x)$, $\frac{d}{dx}\frac{F(x)(1-F(x))}{n} = 0$, we get $F(x) = 0.5$. Thus, $Var(\hat{F}_n(x)) = \frac{F(x)(1-F(x))}{n} \leq \frac{0.5-0.25}{n} = \frac{1}{4n} \leq 0, .0025^2$, solve this, we get $n = 40000$. From the generated plots, we can see that as k increases, the distribution will become more like the empirical distribution.

```python
def main():
    required_std = 0.0025
    n = int(np.ceil(1.0 / (required_std * 2))) ** 2
    ks = [1, 8, 64, 512]
    for k in ks:
        Y_k = np.sum(np.sign(np.random.randn(n, k)) * np.sqrt(1.0 / k), axis=1)
        plt.step(sorted(Y_k), np.arange(1, n + 1) / float(n), label=str(k))
        #plot_settings()


    # Plot gaussian
    Z = np.random.randn(n)
    plt.step(sorted(Z), np.arange(1, n + 1) / float(n), label="Gaussian")
    plot_settings()

@problem.tag("hw0-A", start_line=10)
def plot_settings():
    # Plotting settings
    plt.grid(which="both", linestyle="dotted")
    plt.legend(
        loc="lower right", bbox_to_anchor=(1.1, 0.2), fancybox=True, shadow=True, nco
    )
    # TODO: Look through matplotlib documentation and:
    #   - limit x axis to be between -3 and 3
    #   - Add label "Observations" on x axis
    #   - Add label "Probability" on y axis
    #   - Render the plot with plt.show() call

    #raise NotImplementedError("Your Code Goes Here")
    plt.xlim([-3,3])
    plt.ylim([0,1])
    plt.xlabel('Observations')
    plt.ylabel('Probability')
    plt.show()
```




1.0
0.8
0.6
0.4
0.2
0.0
Probability
−3
−2
−1
0
1
2
3
Observations
Gaussian