Derivatives Involving Normal CDF and PDF Analytically and with Sympy

Fan Wang

2020-09-02

Contents

1 Differentiation with Normal Distribution

Go to the **RMD**, **PDF**, or **HTML** version of this file. Go back to Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
import sympy as sym
import scipy.stats as stats
import pandas as pd
import pprint
```

1.1 Derivatives and the Normal Distribution

1.1.1 Normal CDF and PDF and Standard Normal

Before we differentiate, important to note that while the CDF of normal distribution, and the CDF of the z-scored rescaled normal distribution are the same, this is not true for PDF. Let Φ and ϕ represent the standard normal (mean 0, sd =1) distribution, and F and f denote the CDF and pdf or some other normal distribution.

While we have the equality below:

$$\Phi\left(z = \frac{x-\mu}{\sigma}\right) = F(x;\mu,\sigma)$$

The two terms below are NOT equal:

$$\phi\left(z = \frac{x-\mu}{\sigma}\right) \neq f(x;\mu,\sigma)$$

We show this equality in CDF and inequality in PDF below:

```
# Mean and standard deviations
fl_mu = 5
fl_sd = 1.314

# Define an array of x-values
ar_x = np.linspace(fl_mu-3*fl_sd, fl_mu+3*fl_sd, 5)
ar_z = (ar_x - fl_mu)/fl_sd

# PDFs
ar_fx_pdf = stats.norm.pdf(ar_x, fl_mu, fl_sd)
ar_fz_pdf = stats.norm.pdf(ar_z, 0, 1)

# CDFs
ar_fx_cdf = stats.norm.cdf(ar_x, fl_mu, fl_sd)
ar_fz_cdf = stats.norm.cdf(ar_z, 0, 1)

# Combine
mt_pdf_cdf = np.column_stack((ar_x, ar_z,
    ar_fx_pdf, ar_fz_pdf, ar_fx_cdf, ar_fz_cdf)).astype(float)
ar_colnames = ['x', 'z', 'pdf_x', 'pdf_zneqx', 'cdf_x', 'cdf_z']
ar_rownames = ['x=' + str(fl_x) for fl_x in ar_x]
df_pdf_cdf = pd.DataFrame(data=np.around(mt_pdf_cdf, 4),
                          index=ar_rownames,
                          columns=ar_colnames)

# Show derivatives of CDF
print(df_pdf_cdf)
```

```
##                          x    z   pdf_x  pdf_zneqx   cdf_x   cdf_z
## x=1.0579999999999998  1.058 -3.0  0.0034     0.0044  0.0013  0.0013
## x=3.029               3.029 -1.5  0.0986     0.1295  0.0668  0.0668
## x=5.0                 5.000  0.0  0.3036     0.3989  0.5000  0.5000
## x=6.971               6.971  1.5  0.0986     0.1295  0.9332  0.9332
## x=8.942               8.942  3.0  0.0034     0.0044  0.9987  0.9987
```

1.1.2 Derivatives of the Normal PDF with Respect to X

The normal PDF function is:

$$f\left(x;\mu,\sigma\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\cdot\exp\left(-\frac{1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

For the standard normal distribution, with $\mu=0$ and $\sigma=1$, we have:

$$\phi\left(z=\frac{x-0}{1}\right)=\frac{1}{\sqrt{2\pi}}\cdot\exp\left(-\frac{x^2}{2}\right)$$

Note again that $f(x)\neq\phi(x)$ due to the first term below with the extra $\frac{1}{\sigma}$ for $f(x)$:

$$\phi\left(z=\frac{x-\mu}{\sigma}\right)=\frac{1}{\sqrt{2\pi}}\cdot\exp\left(-\frac{1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right)\neq\frac{1}{\sqrt{2\pi\sigma^2}}\cdot\exp\left(-\frac{1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Test this function and see if we obtain the same PDF from our equation and from scipy.stats.norm, as well as from sympy.

```
# using mean and sd and array of x from last block
# Evaluate pdf using scipy stats
ar_f_x_pkg = stats.norm.pdf(ar_x, fl_mu, fl_sd)

# Evaluate using numpy
ar_f_x_fml = (1/np.sqrt(2*np.pi*(fl_sd**2)))*np.exp((-1/2)*(((ar_x - fl_mu)**2)/(fl_sd**2)))
ar_f_z_fml = (1/np.sqrt(2*np.pi))*np.exp((-1/2)*((ar_x-fl_mu)/fl_sd)**2)

# Evaluate using sympy
x, mu, sd = sym.symbols('x mu sd')
sy_f_x = (1/sym.sqrt(2*sym.pi*(sd**2)))*sym.exp((-1/2)*(((x - mu)**2)/(sd**2)))
ar_f_x_sym = [sy_f_x.subs([(x, fl_x), (mu, fl_mu), (sd, fl_sd)]).evalf() for fl_x in ar_x]

# Compare
mt_pdf = np.column_stack((ar_f_x_pkg, ar_f_x_sym, ar_f_x_fml, ar_f_z_fml)).astype(float)
ar_colnames = ['stats.norm.pdf', 'symbolic', 'formula_fx', 'formula_fz_neq_fx']
df_pdf = pd.DataFrame(data=np.round(mt_pdf, 4),
                      index=ar_rownames,
                      columns=ar_colnames)
print(df_pdf)
```

```
##                       stats.norm.pdf  symbolic  formula_fx  formula_fz_neq_fx
## x=1.0579999999999998          0.0034    0.0034      0.0034             0.0044
## x=3.029                       0.0986    0.0986      0.0986             0.1295
## x=5.0                         0.3036    0.3036      0.3036             0.3989
## x=6.971                       0.0986    0.0986      0.0986             0.1295
## x=8.942                       0.0034    0.0034      0.0034             0.0044
```

What is the derivative of the normal pdf with respect to x?

$$\begin{aligned}\frac{df\left(x;\mu,\sigma\right)}{dx} &= \frac{1}{\sqrt{2\pi\sigma^2}}\cdot\exp\left(-\frac{1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right)\cdot\left(-\frac{2}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)\right)\cdot\frac{1}{\sigma} \\ &= -f\left(x;\mu,\sigma\right)\cdot\left(\frac{x-\mu}{\sigma^2}\right)\end{aligned}$$

Let's check our differentiation using sympy.

```
# using mean and sd and array of x from prior blocks
# Take derivative
sy_df_dx = sym.diff(sy_f_x, x)
# Derivative given mu and sd
sy_df_dx = sy_df_dx.subs([(mu, fl_mu), (sd, fl_sd)])
sy_df_dx = sym.simplify(sy_df_dx)
# Evaluate symbol
ar_df_dx_sym = [sy_df_dx.subs([(x, fl_x)]).evalf() for fl_x in ar_x]

# Evaluate using numpy with formula
ar_df_dx_fml = -1*ar_f_z_fml*((ar_x - fl_mu)/(fl_sd**2))

# Evaluate with derivative definition
ar_epsilon = np.array([1e-1, 1e-2, 1e-4, 1e-8])
mt_df_dx = np.empty((ar_x.shape[0], ar_epsilon.shape[0]))
```

```
ar_pdf_x = stats.norm.pdf(ar_x, fl_mu, fl_sd)
for it_eps_ctr, fl_epsilon in enumerate(ar_epsilon):
  ar_pdf_x_plus_epsilon = stats.norm.pdf(ar_x+fl_epsilon, fl_mu, fl_sd)
  mt_df_dx[:, it_eps_ctr] = (ar_pdf_x_plus_epsilon - ar_pdf_x)/(fl_epsilon)

# Combine and print
mt_df_dx = np.column_stack((ar_df_dx_sym, ar_df_dx_fml, mt_df_dx)).astype(float)
ar_colnames = ['symbolic.df_dx', 'formula_df_dx'] + [ 'diff=' + str(fl_eps) for fl_eps in ar_epsilon]
df_df_dx = pd.DataFrame(data=np.around(mt_df_dx, 4),
                        index=ar_rownames,
                        columns=ar_colnames)
print(df_df_dx)
```

```
##                       symbolic.df_dx  formula_df_dx  ...  diff=0.0001  diff=1e-08
## x=1.0579999999999998          0.0077         0.0101  ...       0.0077      0.0077
## x=3.029                       0.1125         0.1479  ...       0.1125      0.1125
## x=5.0                         0.0000        -0.0000  ...      -0.0000      0.0000
## x=6.971                      -0.1125        -0.1479  ...      -0.1125     -0.1125
## x=8.942                      -0.0077        -0.0101  ...      -0.0077     -0.0077
##
## [5 rows x 6 columns]
```

1.1.3 Derivatives of the Standard Normal CDF with Respect to X

Rather than the PDF, we can also take derivative of the normal CDF with respect to x. By definition, the cumulative distribution is the integral below the probability density function curve. By definition, the derivative of the CDF is the PDF in the case of the standard normal distribution.

Note that below, we are looking at Φ and ϕ, not F and f, hence, when we use pdf functions from the computer, we can not use the pdf function for x, but the pdf for z.

$$\frac{d\Phi\left(z=\frac{x-\mu}{\sigma}\right)}{dx} = \frac{1}{\sigma}\cdot\phi\left(z=\frac{x-\mu}{\sigma}\right)$$

```
# Find CDF by taking difference using scipy CDF function
ar_epsilon = np.array([1e-6, 1e-9, 1e-12])
mt_dCDF_dx_dif = np.empty((ar_epsilon.shape[0]+1, ar_x.shape[0]))
ar_cdf_x = stats.norm.cdf(ar_x, fl_mu, fl_sd)
for it_eps_ctr, fl_epsilon in enumerate(ar_epsilon):
  ar_cdf_x_plus_epsilon = stats.norm.cdf(ar_x+fl_epsilon, fl_mu, fl_sd)
  ar_dCDF_dx_dif = (ar_cdf_x_plus_epsilon - ar_cdf_x)/(fl_epsilon)
  mt_dCDF_dx_dif[it_eps_ctr, ] = ar_dCDF_dx_dif

# Derivative formula, note we use ar_f_z_fml, not ar_f_x_fml
ar_dCDF_dx_fml = (1/fl_sd)*ar_f_z_fml
mt_dCDF_dx_dif[ar_epsilon.shape[0],] = ar_dCDF_dx_fml
mt_dCDF_dx_dif = np.transpose(mt_dCDF_dx_dif)

# To panda
ar_colnames = ['diff=' + str(fl_eps) for fl_eps in ar_epsilon] + ['formula']
ar_rownames = ['x=' + str(fl_x) for fl_x in ar_x]
df_dCDF_dx_dif = pd.DataFrame(data=np.round(mt_dCDF_dx_dif, 3),
                              index=ar_rownames,
                              columns=ar_colnames)
```

```
# Show derivatives of CDF
print(df_dCDF_dx_dif)
```

```
##                       diff=1e-06  diff=1e-09  diff=1e-12  formula
## x=1.0579999999999998       0.003       0.003       0.003    0.003
## x=3.029                    0.099       0.099       0.099    0.099
## x=5.0                      0.304       0.304       0.304    0.304
## x=6.971                    0.099       0.099       0.099    0.099
## x=8.942                    0.003       0.003       0.003    0.003
```

1.1.4 Derivative of Function of Normal CDF with Respect to X

$$\frac{d\left(A \cdot x \cdot \Phi\left(\frac{x-\mu}{\sigma}\right)\right)}{dx} = \frac{A \cdot x}{\sigma} \cdot \phi\left(\frac{x-\mu}{\sigma}\right) + A \cdot \Phi\left(\frac{x-\mu}{\sigma}\right)$$

1.1.5 Derivatives of the Standard Normal CDF with Respect to Mean and SD

We also have the derivative of the normal distribution with respect to its mean and standard deviation. We have:

$$\frac{d\Phi\left(z=\frac{x-\mu}{\sigma}\right)}{d\mu} = -\frac{1}{\sigma} \cdot \phi\left(z=\frac{x-\mu}{\sigma}\right)$$

And we have:

$$\frac{d\Phi\left(z=\frac{x-\mu}{\sigma}\right)}{d\sigma} = (-1) \cdot \phi\left(z=\frac{x-\mu}{\sigma}\right) \cdot \frac{(x-\mu)}{\sigma^2}$$

Note the similarity of the expression above to the earlier expression for $\frac{df(x)}{dx}$, where f is the pdf of a normal, the only difference is whether the pdf of the standard normal and the pdf of the normal is used.

```
# Find CDF by taking difference using scipy CDF function
ar_epsilon = np.array([1e-6, 1e-9, 1e-12])
mt_dCDF_dx_dif_sd = np.empty((ar_epsilon.shape[0]+1, ar_x.shape[0]))
ar_cdf_x = stats.norm.cdf(ar_x, fl_mu, fl_sd)
for it_eps_ctr, fl_epsilon in enumerate(ar_epsilon):
  # now add to the fl_sd part
  ar_cdf_x_plus_epsilon = stats.norm.cdf(ar_x, fl_mu, fl_sd+fl_epsilon)
  ar_dCDF_dx_dif_sd = (ar_cdf_x_plus_epsilon - ar_cdf_x)/(fl_epsilon)
  mt_dCDF_dx_dif_sd[it_eps_ctr, ] = ar_dCDF_dx_dif_sd

# Derivative formula, note we use ar_f_z_fml, not ar_f_x_fml
ar_dCDF_dx_fml_sd = -((ar_x-fl_mu)/fl_sd**2)*ar_f_z_fml
mt_dCDF_dx_dif_sd[ar_epsilon.shape[0],] = ar_dCDF_dx_fml_sd
mt_dCDF_dx_dif_sd = np.transpose(mt_dCDF_dx_dif_sd)

# To panda
ar_colnames = ['sd_diff=' + str(fl_eps) for fl_eps in ar_epsilon] + ['formula_dPHI_dSD']
ar_rownames = ['x=' + str(fl_x) for fl_x in ar_x]
df_dCDF_dx_dif_sd = pd.DataFrame(data=np.round(mt_dCDF_dx_dif_sd, 3),
                                 index=ar_rownames,
                                 columns=ar_colnames)

# Show derivatives of CDF
print(df_dCDF_dx_dif_sd)
```

```
##                        sd_diff=1e-06  ...  formula_dPHI_dSD
## x=1.0579999999999998           0.010  ...             0.010
## x=3.029                        0.148  ...             0.148
## x=5.0                          0.000  ...            -0.000
## x=6.971                       -0.148  ...            -0.148
## x=8.942                       -0.010  ...            -0.010
##
## [5 rows x 4 columns]
```

1.2 An Utility Maximization Problem with a Single Choice and Normal CDF

Below we have an equation, we want to take its derivative with respect to N. Φ and ϕ both appear in the following equation.

$$g(N) = \left\{ \begin{array}{c} (Y - p \cdot N) + \rho \cdot (Y - p \cdot N)^2 \\ +\gamma \cdot \hat{A} \cdot N^{\beta} \\ +\lambda \cdot \left(\hat{A} \cdot N^{\beta} - \mu \right) \cdot \Phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \\ +\lambda \cdot \sigma \cdot \phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \end{array} \right\}$$

This is an equation from the paper You Are What Your Parents Expect: Height and Local Reference Points, this equation captures relative trade-offs between nutritional and alternative consumption choices, taking expectation of relative height into consideration. This equation contains both the standard normal CDF and PDF (see Mills ratio).

1.2.1 Derivative with CDF and PDF

We first take derivative of the function with respect to N, taking into proper consideration what we derived earlier regarding derivatives of PDF and CDF.

The derivative is:

$$\frac{dg(N)}{dN} = \left\{ \begin{array}{c} -p - 2 \cdot p \cdot \rho \cdot (Y - p \cdot N) \\ +\beta \cdot \gamma \cdot \hat{A} \cdot N^{\beta - 1} \\ +\beta \cdot \lambda \cdot \hat{A} \cdot N^{\beta - 1} \cdot \left(\Phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \right) \\ +\lambda \cdot \left(\hat{A} \cdot N^{\beta} - \mu \right) \cdot \left(\phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \right) \cdot \frac{\beta \cdot \hat{A} \cdot N^{\beta - 1}}{\sigma} \\ -\lambda \cdot \sigma \cdot \phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \cdot \frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \cdot \frac{\beta \cdot \hat{A} \cdot N^{\beta - 1}}{\sigma} \end{array} \right\}$$

Note that the last two terms cancel out, and we have:

$$\begin{aligned} \frac{dg(N)}{dN} &= \left\{ \begin{array}{c} -p - 2 \cdot p \cdot \rho \cdot (Y - p \cdot N) \\ +\beta \cdot \gamma \cdot \hat{A} \cdot N^{\beta - 1} \\ +\beta \cdot \lambda \cdot \hat{A} \cdot N^{\beta - 1} \cdot \left(\Phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \right) \end{array} \right\} \\ &= \left\{ \begin{array}{c} -p - 2 \cdot p \cdot \rho \cdot (Y - p \cdot N) \\ +\beta \cdot \hat{A} \cdot N^{\beta - 1} \cdot \left(\gamma + \lambda \cdot \Phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \right) \end{array} \right\} \end{aligned}$$

At optimal choices, we can set the FOC equal to zero, and have:

$$p + 2 \cdot \rho \cdot p \cdot Y - 2 \cdot \rho \cdot p^2 \cdot N = \beta \cdot \hat{A} \cdot N^{\beta - 1} \cdot \left(\gamma + \lambda \cdot \Phi \left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma} \right) \right)$$

For more or less complicated functions, it is potentially easy to make mistakes during differentiation. As we did before, we will check by computing $\lim_{\epsilon \to 0} \frac{g(N+\epsilon)-g(N)}{\epsilon}$, following the definition of derivative. We try multiple values of ϵ as before. This confirms the results we just derived.

```
# Check if the derivative derived is correct
# Various parameters
fl_rho = -0.0473
fl_beta = 0.0725
fl_A = np.exp(4.1435)
fl_gamma = 0.0325
fl_lambda = -0.0257
fl_mu = 77
fl_sigma = 1.5

# Price and income
fl_P = 53
fl_Y = 449

# An array of N choices
ar_N = np.linspace(1, 30, 5)

# Evaluate formula components
fc_c = lambda N: fl_Y - fl_P * N
fc_mpn = lambda N: fl_beta*fl_A*(N**(fl_beta-1))
fc_health = lambda N: fl_A*(N**fl_beta)
fc_cdf_health = lambda H: stats.norm.cdf((H - fl_mu)/fl_sigma, 0, 1)
fc_pdf_health = lambda H: stats.norm.pdf((H - fl_mu)/fl_sigma, 0, 1)

# Evaluate FOC formula explicitly
ar_FOC = (2*fl_rho*(fl_P**2)*ar_N - 2*fl_rho*fl_P*fl_Y - fl_P)
ar_FOC = ar_FOC + fc_mpn(ar_N)*(fl_gamma + fl_lambda*fc_cdf_health(fc_health(ar_N)))

# The Utility function
def g_N_eval(N):
  ar_health = fc_health(N)
  g_N = (fc_c(N) + fl_rho*(fc_c(N)**2)) + fl_gamma*ar_health
  g_N = g_N + fl_lambda*(ar_health - fl_mu)*fc_cdf_health(ar_health)
  g_N = g_N + fl_lambda*fl_sigma*fc_pdf_health(ar_health)
  return g_N

# Compute Utility and Directly take differences.
ar_epsilon = np.array([1e-6, 1e-9, 1e-12])
mt_dG_dN_dif = np.empty((ar_N.shape[0], ar_epsilon.shape[0]))
ar_g_N = g_N_eval(ar_N)
for it_eps_ctr, fl_epsilon in enumerate(ar_epsilon):
  ar_g_N_plus_epsilon = g_N_eval(ar_N+fl_epsilon)
  ar_dG_dN_dif = (ar_g_N_plus_epsilon - ar_g_N)/(fl_epsilon)
  mt_dG_dN_dif[:, it_eps_ctr] = ar_dG_dN_dif

# To panda
mt_dG_dN = np.column_stack((ar_FOC, mt_dG_dN_dif)).astype(float)
ar_colnames = ['formula_dG_dN'] + [ 'diff=' + str(fl_eps) for fl_eps in ar_epsilon]
ar_rownames = ['N=' + str(fl_N) for fl_N in ar_N]
df_dG_dN = pd.DataFrame(data=np.around(mt_dG_dN, 4),
```

```
                        index=ar_rownames,
                        columns=ar_colnames)
print(df_dG_dN)
```

```
##          formula_dG_dN  diff=1e-06  diff=1e-09  diff=1e-12
## N=1.0         1932.6133   1932.6132   1932.6144   1932.6762
## N=8.25           5.9330      5.9328      5.9330      5.9304
## N=15.5       -1920.6331  -1920.6333  -1920.6327  -1919.0338
## N=22.75      -3847.1909  -3847.1910  -3847.1881  -3845.3436
## N=30.0       -5773.7444  -5773.7446  -5773.7270  -5762.5584
```

1.2.2 Components of First Order Condition and CDF

An important expression is:

$$\gamma + \lambda \cdot \Phi\left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma}\right)$$

We have MPNonH and MPHonU

The term MPHonU is positive if both γ and λ are positive. And if $\gamma - \lambda > 0$

$$\frac{d\Phi\left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma}\right)}{d\sigma} = (-1) \cdot \phi\left(\frac{\hat{A} \cdot N^{\beta} - \mu}{\sigma}\right) \cdot \frac{\left(\hat{A} \cdot N^{\beta} - \mu\right)}{\sigma^2}$$

which is positive when $\hat{A} \cdot N^{\beta} < \mu$, and negative otherwise.