CS 3430: S19: SciComp with Py
Assignment 6
Exponential Models in Psychology, Social Science, Public Health, and Botany

Vladimir Kulyukin
Department of Computer Science
Utah State University

February 16, 2019

Learning Objectives

1. Ebbinghaus Model of Forgetting
2. Spread of Epidemics
3. Spread of News
4. Logistic Growth
5. 2D Plotting

Problem 1: Ebbinghaus Retention Model (1 pt)

After taking a class students rarely retain all the material covered in that class. Some retain more, some - less. Let $r(t)$ be the percentage of the material that the student can recall t weeks later. The psychologist Hermann Ebbinghaus (`https://en.wikipedia.org/wiki/Hermann_Ebbinghaus`) found that this percentage of retention can be modeled as $r(t) = (100 - a)e^{-\lambda t} + a$, where λ and a are student specific and such that λ is a positive constant, typically $0 < \lambda < 1$, and $0 < a < 100$.

Write the function `percent_retention_model(lmbda, a)` that takes two constants `lmbda` and `a` and returns a function representation of the Ebbinghaus retention model.

Use your implementation of `percent_retention_model(lmbda, a)` to implement the function `plot_retention(lmbda, a, tl, tu)` that takes four constants such that the first two, `lmbda` and `a`, are the same as in `percent_retention_model`,

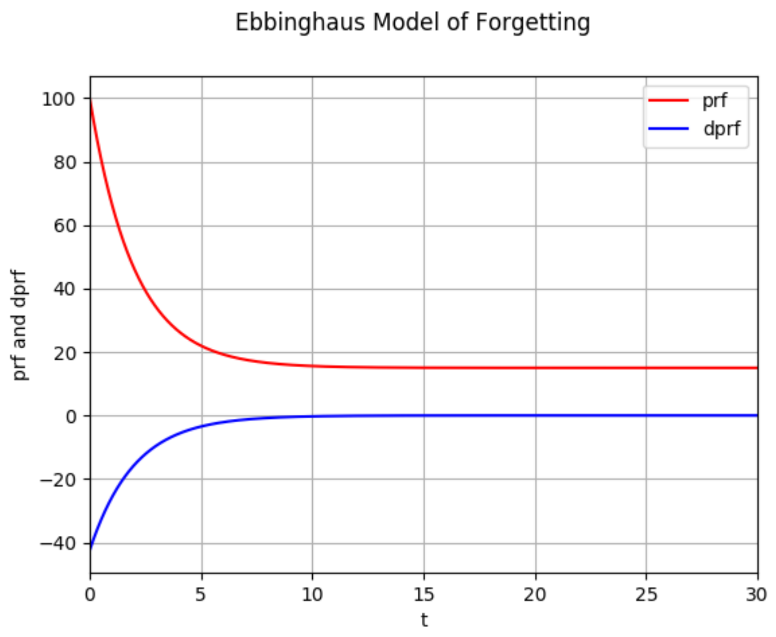

Figure 1: Ebbinghaus Model of Forgetting.

and the last two, `tl` and `tu`, are the lower and upper bound of the x-axis interval on which the plots of the model returned by `percent_retention_model` and its derivative are plotted. The title of the plot should be ”Ebbinghaus Model of Forgetting,” the x-axis label should be ”t,” and the y-axis label should be ”prf and dprf,” which stand for ”percent retention function” and ”derivative of percent retention function,” respectively. The prf curve should be plotted red, the dprf curve – blue.

Figure 1 is the plot generated with the following call in the Python shell.

```
>>> plot_retention(make_const(0.5), make_const(15.0),
                   make_const(0.0), make_const(30))
```

Figure 2 is the plot generated with the following call in the Python shell.

```
>>> plot_retention(make_const(0.25), make_const(25.0),
                   make_const(0.0), make_const(30))
```

Let’s find out how much material a student whose $\lambda = 0.5$ and $a = 15$ will retain after 2 weeks.

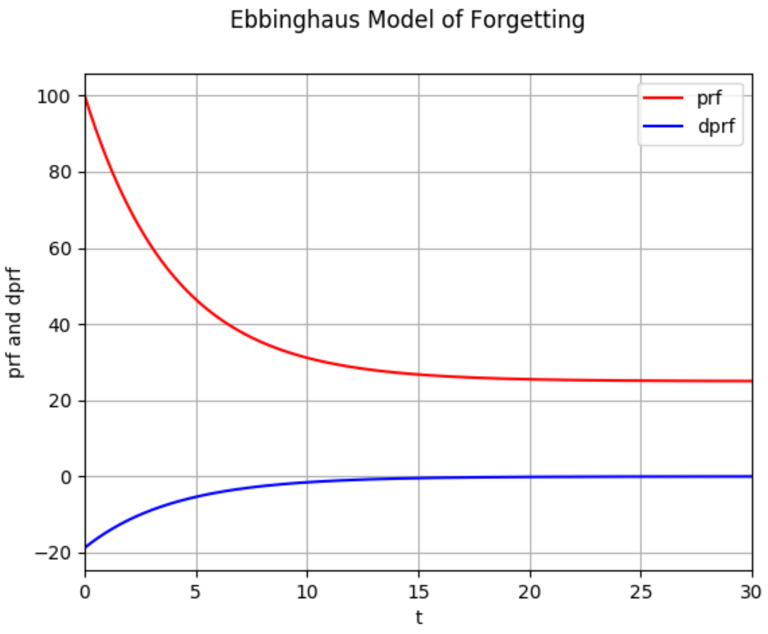

Figure 2: Ebbinghaus Model of Forgetting.

```
>>> prm = percent_retention_model(make_const(0.5), make_const(15.0))
>>> prmf = tof(prm)
>>> prmf(2)
46.2697524996
```

Save your code in `hw06_s19.py`.

Problem 2: Spread of Diseases (1 pt)

Suppose the department of public health in the city of X is monitoring the spread of a contagious disease (e.g., a strain of flu). Let X's population be P. At week t_0 the deparment records that p_0 cases have been reported. At week t_1, such that $t_0 < t_1$, the department records that p_1 cases have been reported.

Write the function `spread_of_disease_model(p, t0, p0, t1, p1)` that takes five constants where the first constant is the size of the population of X and the other four constants are the values of t_0, p_0, t_1, and p_1, as described in the previous paragraph. This function returns a function representation of the spread model for the disease determined by the given parameters.

Use your implementation of `spread_of_disease_model` to implement the function `plot_spread_of_disease(p, t0, p0, t1, p1, tl, tu)` whose first four parameters are the same as the parameters of `spread_of_disease_model` and the last two, `tl` and `tu`, are the lower and upper bound of the x-axis interval on which the plots of the model returned by `spread_of_disease_model` and its derivative are plotted. The title of the plot should be "Spread of Disease," the x-axis label should be "t," and the y-axis label should be "sdf and dsdf," which stand for "spread of disease function" and "derivative of spread of disease function," respectively. The sdf curve should be plotted red, the dsdf curve – blue.

Figure 3 is the plot generated with the following call in the Python shell.

```
>>> plot_spread_of_disease(make_const(500000),
                           make_const(0.0),
                           make_const(200.0),
                           make_const(1.0),
                           make_const(500.0),
                           make_const(0.0),
                           make_const(7.0))
```

Save your code in `hw06_s19.py`.

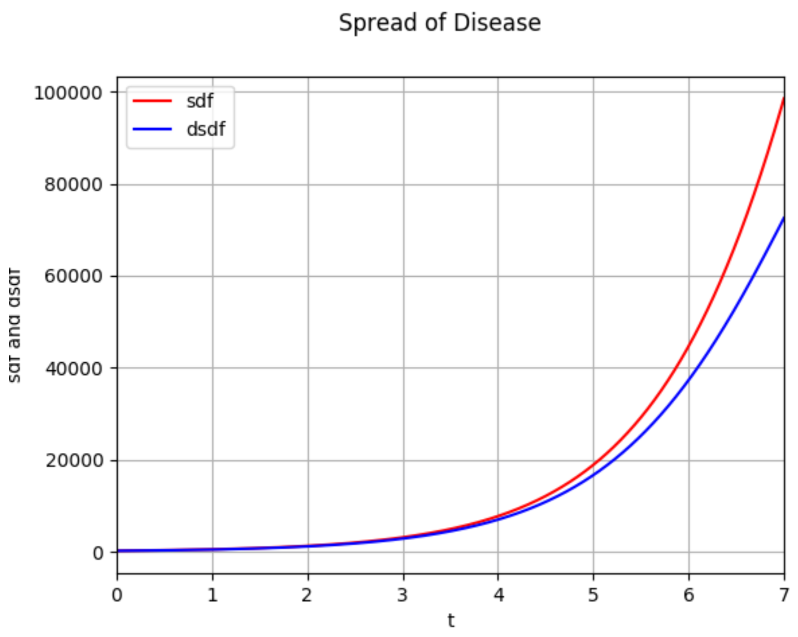

Figure 3: Spread of Disease.

Problem 3: Plant Growth (2 pts)

Suppose a botanist monitors the growth of a plant X. The botanist records that at day t_0 the plant's height is h_0 cm and at day t_1, such that $t_0 < t_1$, the plant's height is h_1 cm. Over time, after watching many plants X grow and recording their growth stages, the botanist observes that at maturity a typical plant X reaches the height of m cm.

Write the function `plant_growth_model(m, t0, h0, t1, h1)` that takes five constants where the first constant is the height of a typical plant X at maturity and the other four constants are the values of t_0, h_0, t_1, and h_1, as described in the previous paragraph. This function returns a function representation of the plant growth model for a typical plant X determined by the given parameters.

Use your implementation of `plant_growth_model` to implement the function `plot_plant_growth(m, t0, h0, t1, h1, tl, tu)` whose first five parameters are the same as the parameters of `plant_growth_model` and the last two, `tl` and `tu`, are the lower and upper bound of the x-axis interval on which the plots of the model returned by `plant_growth_model` and its derivative are plotted. The title of the plot should be "Plant Growth," the x-axis label should be "t," and the y-axis label should be "pgf and dpgf," which stand for "plant growth function" and "derivative of plant growth function," respectively. The pgf curve should be plotted red, the dpgf curve – blue.

Figure 4 is the plot generated with the following call in the Python shell.

```
>>> plot_plant_growth(make_const(55.0),
                      make_const(9.0),
                      make_const(8.0),
                      make_const(25.0),
                      make_const(48.0),
                      make_const(9.0),
                      make_const(50.0))
```

Save your code in `hw06_s19.py`.

Problem 4: Spread of News (1 pt)

A news item is broadcast by a mass media outlet to a potential audience of P people.

Write the function `spread_of_news_model(p, k)` that takes two constants that characterize the spread of news among a given population, as we discussed in class, and returns a function representation of the spread of news model determined by these parameters.

Use your implementation of `spread_of_news_model` to implement the function `plot_spread_of_news(p, k, tl, tu)` whose first two parameters are the

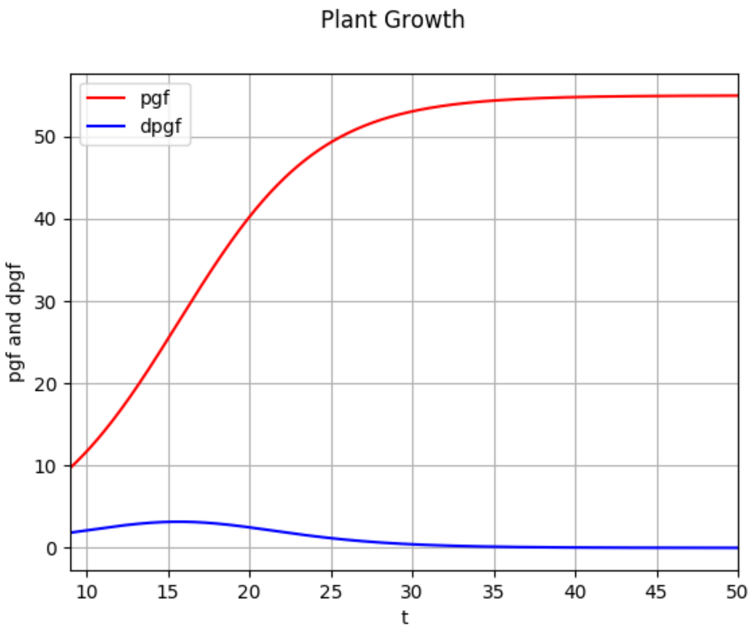

Figure 4: Plant Growth.

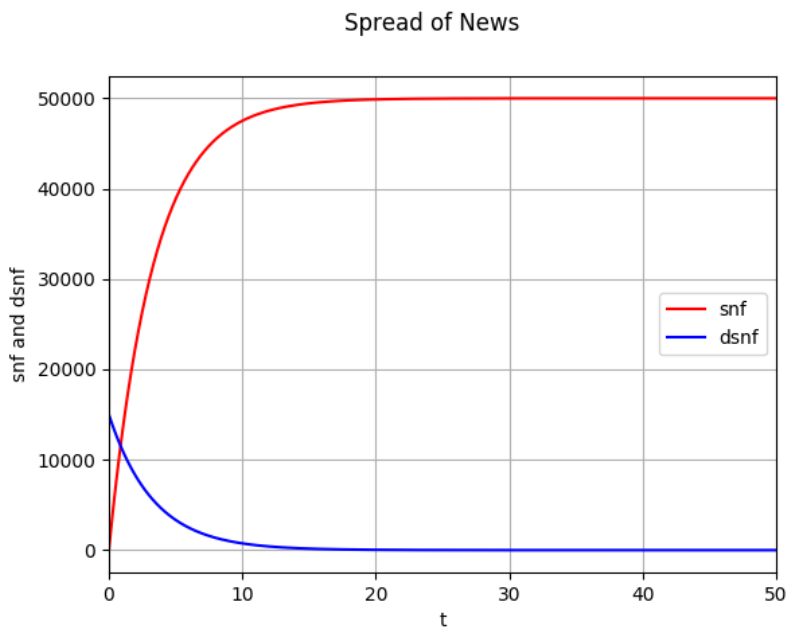

Figure 5: Spread of News.

same as the parameters of `spread_of_news_model` and the last two, `tl` and `tu`, are the lower and upper bound of the x-axis interval on which the plots of the model returned by `spread_of_news_model` and its derivative are plotted. The title of the plot should be "Spread of News," the x-axis label should be "t," (time in days) and the y-axis label should be "snf and dsnf," which stand for "spread of news function" and "derivative of spread of news function," respectively. The snf curve should be plotted red, the dsnf curve – blue.

Figure 5 is the plot generated with the following call in the Python shell.

```
>>> plot_spread_of_news(make_const(50000),
                        make_const(0.3),
                        make_const(0.0),
                        make_const(50.0))
```

Let's find out at what rate the news is spreading initially.

```
>>> sn = spread_of_news_model(make_const(50000),
                              make_const(0.3))
>>> dsn = deriv(sn)
>>> dsnf = tof(dsn)
>>> dsnf(0)
```

```
15000.0
```

So, initially the rate of outreach is 15,000 people per day. What happens at day 10?

```
>>> sn = spread_of_news_model(make_const(50000),
                              make_const(0.3))
>> dsn = deriv(sn)
>>> dsnf = tof(dsn)
>>> dsnf(10)
746.806025518
```

The rate of outreach drops to 747 people. How about day 30?

```
>>> sn = spread_of_news_model(make_const(50000),
                              make_const(0.3))
>> dsn = deriv(sn)
>>> dsnf = tof(dsn)
>>> dsnf(30)
1.8511470613
```

The rate of outreach is less than 2 people per day. The media outlet had better look for new topics.

A side note for CS seniors and grad students in this class. If you are looking for a project, here is something to think about. Suppose that you take a bunch of corporate media clips, similar to the news collage we watched in lecture 11 (here is the link `https://www.youtube.com/watch?v=qjUvfZj-Fm0` in case you didn't attend), push each of them through an ASR engine to obtain its transcript, and then model the distribution of corporate cliches such as "bombshell," "walls are closing in," "tipping point," "beginning of the end," etc. Once you have reliable cliche distribution models, you can use them to classify news clips as "character assassination," "fear mongering to obtain funding," etc.

Save your code in `hw06_s19.py`.

What to Submit

Submit your code in `hw06_s19.py`. You should also submit all the files needed to run your code. The easiest and safest thing for you to do is to zip your whole directory into `hw06.zip` and upload it to Canvas.

Do not change the names of the files that were given to you or the names of the functions you are asked to implement. The weekly unit tests that I write for the grader depend on these names remaining the same.

Happy Hacking!