Least squares, gradient descent
Earlier, we looked at the **logistic model** for population growth,

\[ y' = ry(1 - y/K). \]

The **Gompertz model** instead uses the equation

\[ y' = -ay \log(y/K) \]

where \( a \) and \( K \) are constants.

- Used to describe growth of tumors (\( y(t) = \) tumor size)
- Growth rate per \( y \) (i.e. \( a \log(y/K) \)) non-linear in \( y \)

The solution to this ODE is

\[ y(t) = K \exp(-ae^{-bt}) \] (M)

Suppose we have an experiment measuring the tumor size at \( N \) times:

\[ (\hat{t}_k, \hat{y}_k), \quad k = 1, \ldots, N. \] (D)
The goal: estimate parameters $K$, $a$ and $b$ by fitting (M) to the data (D).

(data here generated from exact solution plus some noise...)

\[ y(t) = K \exp(-ae^{-bt}) \]  
\[ (\hat{t}_k, \hat{y}_k), \quad k = 1, \ldots, N. \]
Gompertz model

\begin{equation}
y(t) = K \exp(-ae^{-bt}) \tag{M}
\end{equation}

\begin{equation}
(\hat{t}_k, \hat{y}_k), \quad k = 1, \cdots, N. \tag{D}
\end{equation}

To define the ‘best fit’, consider the least-squared error

\[ E(r) = \sum_{k=1}^{N} (y(\hat{t}_k; r) - \hat{y}_k)^2 \]

where \( r = (r_1, r_2) \) denotes the parameters \((a, b)\).

(The \( r \) vector is more convenient for computation).

- The function \( E \) should have a minimum at \( r^* = (a^*, b^*) \) - these are the best fit parameters
- Since the number of parameter is small, \( E \) cannot be made zero
- How do we find this minimum?
Now, ignoring the context, what we really want is to find a minimum of $E(r)$, \( E = \text{some function of } r = (r_1, r_2) \)

### Key facts (calculus!)

- **At a minimum of $E$, the gradient is zero:**
  
  \[ r^* \text{ is a minimum of } E \implies \nabla E = 0 \text{ at } r^* \]

  where \( \nabla E = (\frac{\partial E}{\partial r_1}, \frac{\partial E}{\partial r_2}) \).

- **At any point $r$, the function $E$ is decreasing fastest in the $-\nabla E$ direction.**

The latter holds since

\[
\text{(rate of change of } E \text{ in dir. } v) = v \cdot \nabla E
\]

which is most negative when \( v = -\nabla E \).

We call $v$ the direction of **steepest descent**. This fact suggests that we can ‘follow’ the gradient down to look for a min. of $E$...
Least squares: gradient descent

To search for a possible minimum $x^*$ of the function

$$F(x), \quad F : \mathbb{R}^n \to \mathbb{R}$$

we can iteratively ‘follow the gradient.’

**Algorithm sketch: gradient descent**

Pick a point $x_k$. Then,

- Calculate the steepest descent direction

$$v = -\nabla F(x_k).$$

- Search along $v$, i.e. along the half-line

$$q(\alpha) = x_k + \alpha v$$

to find a new point with a smaller $F$ value.

- Set this to be the ‘next’ point $x_{k+1}$ and repeat.

- This generates a sequence $x_0, x_1, \ldots$, of points with

$$F(x_0) > F(x_1) > F(x_2) > \cdots \to F(x^*) \quad (\text{if all goes well...})$$

- When to stop? When $|F(x_{k+1}) - F(x_k)|$ is small enough, or $|x_{k+1} - x_k|...$
Gradient descent - backtracking

A scheme is required to search for a minimum along the (half) line

\[ q(\alpha) = x_k + \alpha v, \quad \alpha > 0 \quad (L) \]

This step (the **line search**) can be done in several ways.

**Line search strategy: Backtracking**

After identifying the line (L),

- Guess \( \alpha = 1 \)
- If \( F(q(\alpha)) \) is less than \( F(x_k) \) by enough, accept the step
- If not, set \( \alpha \) to \( \alpha / 2 \) and repeat.

- Since \( f \) is decreasing along \( v \) close to \( x_k \), backtracking will always find an \( \alpha \)
- Simplest ‘enough’ criterion: just use

\[ F(q(\alpha)) < F(x_k). \]

(Better choices \( \rightarrow \) more progress per step).

More sophisticated approaches exist... but this will at least guarantee progress!
Gradient descent: example

function: $F(x, y) = (x - 1)^2 + 4y^2$

gradient: $\nabla F = (2(x - 1), 8y)$

Gradient descent steps ($x = (x, y)$):

$$x_{k+1} = x_k - \alpha_k \nabla F(x_k)$$

stopping condition: $\|x_{k+1} - x_k\| < tol$

```python
def func(x):  # F
    return (x[0] - 1)**2 + 4*x[1]**2

def dfunc(x):  # gradient of F
    dx = 2*(x[0] - 1)
    dy = 8*(x[1])
    return np.array((dx, dy))
```

With $tol = 10^{-3}$, completes in 10 steps (not great, but works!).
Gradient descent: limitations

**Issue 1:** Local minima can be a problem...
- We can only hope for a ‘local minimum’ - gradient descent can get stuck in local valleys
- Finding the global minimum of $F$ is harder!
- A good initial guess helps, but requires prior knowledge of $F$.

**Issue 2:** The gradient gives limited information...
- $-\nabla F$ is not always the best direction to search for a minimum...
- Can be slow, give bad information when far from the minimum, etc.
- Many ways to improve it by using a better searching scheme!

Some examples for reference: Levenberg-Marquardt, Gauss-Newton, BFGS...
Gradient descent: applied to least squares

Now back to non-linear least squares... we need to minimize

\[ E(r) = \sum_{k=1}^{N} (y(\hat{t}_k; r) - \hat{y}_k)^2 \]

where \( r = (r_1, r_2) \) (the parameters to fit). To find \( \nabla E \), compute

\[ \frac{\partial}{\partial r_j} \left[ (y(\hat{t}_k, r) - \hat{y}_k)^2 \right] = 2(y(\hat{t}_k, r) - \hat{y}_k) \cdot \frac{\partial f}{\partial r_j}(\hat{t}_k, r) \]

by the chain rule. Then

\[ \frac{\partial E}{\partial r_j} = 2 \sum_{k=1}^{N} \left[ (y(\hat{t}_k, r) - \hat{y}_k) \frac{\partial y}{\partial r_j}(\hat{t}_k, r) \right] \text{ for } j = 1, 2 \]

We then guess initial parameters \( r_0 = (a_0, b_0) \).

The gradient descent loop computes

\[ \mathbf{v}_k = -\nabla E \text{ at } r_k \]

\[ \alpha_k = \text{result of the search step} \]

\[ r_{k+1} = r_k + \alpha_k \mathbf{v}_k \]

Note that convergence to the ‘right’ minimum may require a good \( r_0 \).
Gradient descent: applied to least squares

\[ y(t) = K \exp(-ae^{-bt}) \quad \text{(M)} \]

\[ (\hat{t}_k, \hat{y}_k), \quad k = 1, \ldots, N. \quad \text{(D)} \]

(data here generated from \((a, b) = (3, 2)\) some noise...)

Result: \((a, b) \approx (2.980, 1.979)\), after 43 iterations.
For numerics and analysis, it’s important to put the problem in the right form!

Consider, for example, a model for cell growth with a ‘rapid growth’ term

$$\frac{dp}{dt} = rp(1 - \frac{p}{k}) + sp^2, \quad p(0) = p_0.$$ 

modeling a population $p(t)$ of cells that an grow abnormally fast.

- This problem has three parameters (plus an initial condition $p_0$).
- Understanding qualitative behavior means exploring a 4d space!

We can reduce the number of parameters using non-dimensionalization.

- The idea: rescaling a variable by a constant ($p = C\hat{p}$) does not change how the system behaves (just ‘changing the units’)
- The process: scale all the variables, plug into the model equations.
  Then choose scales that simplify the eqs. (and/or that are meaningful).
Aside: Non-dimensionalization

Model equation:

\[
\frac{dp}{dt} = rp(1 - \frac{p}{K}) + sp^2, \quad p(0) = p_0.
\]

\(p(t)\) is the population (number of cells) and \(t = \) time in minutes.

There are two variables to scale here. Define ‘non-dimensional’ variables

\[
\hat{p} = \frac{p}{P}, \quad \hat{t} = \frac{t}{T}.
\]

where \(P_0\) (cells) and \(T_0\) (minutes) are constants (‘scales’) to be found.

Plug into the model to get

\[
\frac{P}{T} \frac{d\hat{p}}{d\hat{t}} = rP\hat{p}(1 - \frac{P}{K}\hat{p}) + sP^2\hat{p}^2, \quad \hat{p}(0) = \frac{p_0}{P}.
\]

Now multiply through by \(T/P\) to get a non-dimnensional equation (all terms have no units):

\[
\frac{d\hat{p}}{d\hat{t}} = rT\hat{p}(1 - \frac{P}{K}\hat{p}) + sTP\hat{p}^2, \quad \hat{p}(0) = \frac{p_0}{P}.
\]

Now simplifying the logistic term to cancel out \(r\), \(K\) suggests the scales

\[
T = \frac{1}{r}, \quad P = K.
\]
Aside: Non-dimensionalization

Now simplifying the logistic term to cancel out \( r, K \) suggests the scales

\[
T = \frac{1}{r}, \quad P = K.
\]

This gives

\[
\frac{d\hat{p}}{d\hat{t}} = \hat{p}(1 - \hat{p}) + \frac{sK}{r} \hat{p}^2, \quad \hat{p}(0) = \frac{p_0}{K}.
\]

Finally, we see the underlying parameters that matter, which are

\[
\sigma = \frac{sK}{r}, \quad \hat{p}_0 = \frac{p_0}{K}.
\]

Relabeling \( y = \hat{p} \) and \( \tau = \hat{t} \) we then have an ODE for \( y(\tau) \),

\[
\frac{dy}{d\tau} = y(1 - y) + \sigma y^2, \quad y(0) = y_0.
\]

which has a single parameter \( \sigma \) (plus initial condition).

As a sanity check, \( \sigma \) should be non-dimensional (no units):

\[
\sigma = s \frac{K}{r} \sim \frac{1}{\text{cells}} \frac{\text{cells}}{\text{cells} \cdot t} \frac{1}{t}.
\]
Non-dimensionalization

In summary, we converted the dimensional equation

\[
\frac{dp}{dt} = rp \left(1 - \frac{p}{K}\right) + sp^2, \quad p(0) = p_0. \tag{D}
\]

into non-dimensional form

\[
\frac{dy}{d\tau} = y \left(1 - y\right) + \sigma y^2, \quad y(0) = y_0 \tag{ND}
\]

with \(y = p/K\) and \(\tau = rt\) and \(\sigma = sK/r\). This tells us:

- The key parameter is \(\sigma\), a ratio of the rapid to base growth rates.
- The behavior of this ODE really depends on one parameter, not three.
- For numerics, we should use (ND): it leads to simpler code, and we can more easily use properties of the equation to improve the solver.
- For example - \(p\) can asymptote at a time \(t^*\) due to the rapid growth. It is easier to estimate this with (ND), since only \(\sigma\) is involved.

Remember: the code may need to know how to convert back e.g,

\[
\text{times } \tau = 1, 2, 3, \cdots \iff \text{times } t = 1/r, 2/r, \cdots \text{ minutes}.
\]
Example:
parameter estimation for ODEs
Here’s a real example of this sort of problem, where we ‘fit’ an ODE model to experimental data.

In fluid dynamics, we need to measure the surface tension $\gamma$ of a fluid—how much energy per area is required to ‘stretch’ the surface of a fluid.

One method to do so is a pendant drop experiment, where you create a droplet that hangs from a source (like a drop from a faucet).

Then, we fit a theoretical model to this data using least-squares, where $\gamma$, the surface tension, is one of the parameters.

Surface tension in a round drop is described by the **Young-Laplace** formula:

$$\text{pressure from s.t.} = \Delta p = -\gamma \left( \frac{1}{R_1} + \frac{1}{R_2} \right)$$

where $R_1, R_2$ are the radii of curvature.

Gravity creates **hydrostatic pressure** proportional to depth:

$$\text{pressure from gravity} = -Cz$$

(e.g. the weight of water creating pressure in deep ocean).

Now we assume the drop is circular around the vertical axis (cylindrical symmetry) and consider coordinates $(r, z)$ (cylindrical).
... we obtain the following system of ODEs:

\[
\begin{align*}
\frac{d\phi}{ds} &= \frac{2}{R_0} - \frac{B}{R_0^2} z - \frac{\sin \phi}{r} \\
\frac{dz}{ds} &= \cos \phi \\
\frac{dr}{ds} &= \sin \phi
\end{align*}
\]

- The independent variable \( s \) is the arc length.
- The parameters are \( R_0 \) (a base ‘radius’) and \( B = \rho g R_0^2 / \gamma \), the \textbf{Bond number}.
- Given \( B, R_0 \) we can compute surface tension!

The ODE only has an analytical solution with no gravity - just a spherical drop:

\[
\phi = \frac{s}{R_0}, \\
r = R_0 \sin(\phi), \quad z = R_0(\cos(\phi) - 1),
\]

\begin{itemize}
\item \( \rho = \frac{\pi}{11} \)
\item \( s = R_0 \pi \)
\end{itemize}
Letting \( y = (\phi, z, r) \) we have a system of the form

\[
y'(s) = F(y)
\]

subject to the initial conditions

\[
z = r = \phi = 0 \text{ at } s = 0.
\]

In python:

```python
def pendant(s, y, b, r0):
    dp = 2/r0 - b*y[1]/r0**2 - sin(y[0])/r
    dz = cos(y[0])
    dr = sin(y[0])
    return np.array([dp, dz, dr])
```

We can then solve it using an ode solver...

```python
ic = np.zeros(3) # initial condition
b = 1
r0 = 1
s, y = ode_solver(lambda s,y: pendant(s,y,b,r0), [0, 10], ic, h)
```

Assume that the ODE solver takes a function \( f(s,y) \)... The lambda function lets us ‘capture’ \( b \) and \( r_0 \). This makes pendant into a function of \( (s,y) \) only.
Next, we need to know where to stop.

- The pendant shape ends where the drop is attached to the device.
- Assume this occurs when $\phi = \pi/2$
- $\phi = \pi/2$ happens once before, too
- $\implies$ stop at second point with $\phi = \pi/2$

Now let’s assume we have measured data

$$(\hat{s}_k, \hat{r}_k, \hat{z}_k), \quad k = 0, \cdots N$$

where $s_k = kh$ for a uniform spacing $h$, stopping with the criterion above.

Let $r(s; B, R_0)$ (etc.) be the numerical solution. Then the LS error is

$$E(B, R_0) = \sum_{k=0}^{N} \left[ (r(\hat{s}_k; B, R_0) - \hat{r}_k)^2 + (z(\hat{s}_k; B, R_0) - \hat{z}_k)^2 \right]^{1/2}$$
Finally, we have stated a least-squares problem - to minimize

$$E(B, R_0) = \sum_{k=0}^{N} \left[ (r(\hat{s}_k; B, R_0) - \hat{r}_k)^2 + (z(\hat{s}_k; B, R_0) - \hat{z}_k)^2 \right]$$

the least squares error between the measured and numerical solutions.

This can be solved using gradient descent. Finally, the surface tension is computed from $B$ and $R_0$ using

$$\gamma = \frac{B}{\rho g R_0^2}.$$

All that is left is to apply gradient descent. We need:

- A function to compute $E(B, R_0)$ (which calls the ODE solver)
- A function to compute $\nabla E$ (more work required!)
For our fluid droplet example - in dimensional form,

\[
\gamma \left( \frac{d\phi}{ds} + \frac{\sin \phi}{r} \right) = 2\gamma R_0 - \rho g z, \quad \frac{dz}{ds} = \cos \phi, \quad \frac{dr}{ds} = \sin \phi
\]

where \(2\gamma R_0\) is a ‘reference pressure’ \(\rho = \text{fluid density}\).

We can use \(R_0\), the droplet ‘radius’ as a scale for all length variables,

\[
s = R_0 \hat{s}, \quad z = R_0 \hat{z}, \quad r = R_0 \hat{r},
\]

and \(\phi\) is already non-dimensional, leading to

\[
\frac{\gamma}{R_0} \left( \frac{d\phi}{d\hat{s}} + \frac{\sin \phi}{\hat{r}} \right) = 2\gamma R_0 - \frac{\rho g}{R} \hat{z}.
\]

Multiply by the factor on the left to get the non-dimensional form

\[
\frac{d\phi}{d\hat{s}} + \frac{\sin \phi}{\hat{r}} = 2 - \frac{\rho g R_0^2}{\gamma}.
\]
Non-dimensionalization

We then get the non-dimensional system

\[
\frac{d\phi}{ds} + \frac{\sin \phi}{r} = 2 - Bz
\]

\[
\frac{d\hat{z}}{d\hat{s}} = \cos \phi, \quad \frac{d\hat{r}}{d\hat{s}} = \sin \phi
\]

where

\[
B = \frac{\rho g R_0^2}{\gamma} \quad (\text{‘Bond number’}).
\]

The Bond number can be interpreted as

\[
B = \frac{\text{strength of gravity forces}}{\text{strength of surface tension forces}}.
\]

The non-dimensionalization tells us that:

- The system’s behavior really depends only on \( B \)
  e.g. a water droplet and a liquid gallium (like mercury) droplet - about 7 times as dense and 7 times higher surface tension - behave the same way.
- We really only need one parameter for the ODE