1 Fitting a Gaussian function to data

A Gaussian function has the form

\[ f(t) = a \exp\left(-\frac{(t - \mu)^2}{\sigma^2}\right), \]

where \( t \in \mathbb{R} \) is the independent variable, \( a \in \mathbb{R} \) is called the amplitude, \( \mu \in \mathbb{R} \) is called the center, and \( \sigma \) is called the spread or width. We can assume without loss of generality that \( \sigma > 0 \). Thus, the vector of parameters is \( p = (a, \mu, \sigma) \). We are given a set of data pairs

\[(t_1, y_1), \ldots, (t_N, y_N),\]

and our goal is to fit a Gaussian function to the data. We measure the quality of a fit using the root mean squared error:

\[ E = \left(\frac{1}{N} \sum_{i=1}^{N} (f(t_i) - y_i)^2\right)^{\frac{1}{2}}. \]

(a) Explain how to use the Gauss-Newton method to compute a parameter vector \( p \) that minimizes \( E \).

(b) Fit a Gaussian function to the data in \texttt{fit_gaussian_data.m}. Explain how you chose your initial guess for the parameters. Plot the RMS error as a function of the iteration number. Plot the data together with the fitted Gaussian function. Repeat for a different, but still reasonable, initial guess for the parameters. Repeat for an unreasonable initial guess for the parameters. Comment on your results.

2 Curve smoothing

We are given a function \( F : [0, 1] \to \mathbb{R} \), and we want to find a function \( G : [0, 1] \to \mathbb{R} \) that is a smoothed version of \( F \). We will evaluate our smoothed version \( G \) of \( F \) using the following two criteria.

- The mean squared deviation from \( F \):

  \[ D = \int_0^1 (F(t) - G(t))^2 dt \]

- The mean squared curvature:

  \[ C = \int_0^1 G''(t)^2 dt. \]
The first criterion penalizes $G$ for being far away from $F$, while the second criterion penalizes $G$ for not being smooth. If we only cared about $D$, we would take $G = F$; if we only cared about $C$, we would take $G$ to be the best affine approximation of $F$ (this gives $C = 0$). However, in general we want $D$ and $C$ to both be small: that is, we have a multiobjective problem. We can approximate $F$ and $G$ by vectors $f, g \in \mathbb{R}^n$ such that

$$f_i = F(i/n) \quad \text{and} \quad g_i = G(i/n)$$

for $i = 1, \ldots, n$. You can assume that $n$ is large enough that $f$ and $g$ are good representations of $F$ and $G$. The discretized versions of our evaluation criteria are as follows.

- **The mean squared deviation from $f$:**

  $$d = \frac{1}{n} \sum_{i=1}^{n} (f_i - g_i)^2$$

- **The mean squared curvature:**

  $$c = \frac{1}{n-2} \sum_{i=2}^{n-1} \left( \frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2} \right)^2$$

In our definition of $c$, note that

$$\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}$$

is a discrete approximation of $G''(i/n)$.

(a) Explain how to choose $g$ in order to minimize $d + \mu c$ for a given $\mu \geq 0$. State any assumptions that are needed for your method to work. In particular, explain how to find $g$ in the extreme cases $\mu = 0$ and $\mu \to \infty$.

(b) The file `curve_smoothing_data.m` defines the following variables.

- $n$, the number of samples in our discretizations of $F$ and $G$
- $f$, the vector representing the function $G$

Plot the optimal tradeoff curve between $d$ and $c$. Be sure to identify critical points such as the intersection of the curve with an axis. Plot $f$ and $g$ on the same set of axes for $\mu = 0$; repeat for $\mu = \infty$, and three additional intermediate values of $\mu$.

### 3 Robust regression using the Huber penalty function

The Huber penalty function is

$$H_{\delta}(d) = \begin{cases} \frac{1}{2}d^2 & |d| \leq \delta, \\ \delta(|d| - \frac{1}{2}\delta) & \text{otherwise}, \end{cases}$$

where $\delta > 0$ is a parameter. Observe that the Huber penalty function is quadratic for small values of $d$, and linear for large values of $d$. Thus, the Huber penalty function attempts to combine the sensitivity of the squared-error loss function to small errors, and the robustness of the absolute-error loss function to large errors.
(a) Suppose you want to fit a line to given data points \((t_1, x_1), \ldots, (t_N, x_N) \in \mathbb{R}^2\). Explain how to use iteratively reweighted least squares to choose the parameters \(a\) and \(b\) in order to minimize the total Huber loss:

\[
J = \sum_{i=1}^{N} H_\delta(at_i + b - x_i).
\]

In particular, what is the weight function, and what is the update equation?

(b) Apply your method to the data defined in huber_penalty_function_data.m using \(\delta = 1\). Report your estimates of the parameters \(a\) and \(b\), and the corresponding value of the total Huber loss. Make a plot of the data, the line corresponding to your estimates of \(a\) and \(b\), and the line obtained using least-squares. Briefly comment on your results.

4 Fitting an exponential function to data
Consider the data set shown in figure 1.

![Data set](image)

**Figure 1** – a data set that can be modeled using an exponential function

The file fit_exponential_data.m defines the data points \((x_1, y_1), \ldots, (x_N, y_N)\). We want to fit an exponential function to the data:

\[
f(x) = c_1 + c_2 \exp(c_3 x).
\]

(a) Explain how to use the Gauss/Newton method to choose the parameters \(c_1\), \(c_2\), and \(c_3\) in order to minimize the sum of squared errors:

\[
\sum_{i=1}^{N} (f(x_i) - y_i)^2.
\]
(b) Apply the Gauss/Newton method to the data set. Use $c_1 = 10$, $c_2 = 5$, and $c_3 = -1$ for your initial estimates, and terminate the algorithm when $\|c^{(k)} - c^{(k-1)}\| < 1 \times 10^{-12}$, where $c^{(k)}$ is the $k$th vector of parameter estimates. Report the final parameter estimates, and the number of iterations until convergence.

c) Explain how to use the Newton’s method to choose the parameters $c_1$, $c_2$, and $c_3$ in order to minimize the sum of squared errors:

$$\sum_{i=1}^{N} (f(x_i) - y_i)^2.$$ 

d) Apply Newton’s method to the data set. Use $c_1 = 10$, $c_2 = 5$, and $c_3 = -1$ for your initial estimates, and terminate the algorithm when $\|c^{(k)} - c^{(k-1)}\| < 1 \times 10^{-12}$, where $c^{(k)}$ is the $k$th vector of parameter estimates. Report the final parameter estimates, and the number of iterations until convergence. Submit a plot showing the data and the fitted model on a single set of axes.

e) Discuss the relative performance of the Gauss/Newton method and Newton’s method. What properties of the data set are reflected in the relative performance of the two algorithms?

5 Managing a power network

Consider a power network with $m$ loads and $n$ generators. The $i$th load requests $r_i \geq 0$ units of power. If the $j$th generator operates at level $x_j \geq 0$, the amount of power delivered to the $i$th load by the $j$th generator is $G_{ij}x_j$, where $G_{ij} \geq 0$ is a known constant. Thus, the total power delivered to the $i$th load is

$$p_i = \sum_{j=1}^{n} G_{ij}x_j.$$ 

The file `power_network_management_data.m` contains the vector $r \in \mathbb{R}^m$ and the matrix $G \in \mathbb{R}^{m \times n}$.

(a) Suppose load 1 is critical, and its power request must be met exactly. Explain how to choose the vector $x \in \mathbb{R}^n$ of generator operating levels in order to minimize the sum of the squared differences between the requested and delivered powers subject to the constraint that the power request of load 1 is met exactly. Report your vector of generator operating levels, and the sum of the squared differences between the requested and delivered powers. You may ignore the constraints that the generator operating levels must be nonnegative.

(b) Suppose generator 4 fails. While the generator is being repaired, you relax the constraint that the power request of load 1 must be met exactly. Submit a plot of the trade-off curve for the squared difference between the requested and delivered power for load 1, and the sum of the squared differences between the requested and delivered powers for the other loads. You may ignore the constraints that the generator operating levels must be nonnegative.