MANUAL
SENPY: Algorithms for Sensitivity Testing in Python

Alex Casey

Warning: The current code is version 1.2 and this manual is v1.1 and needs to be updated.

Contents

1 Introduction

Sensitivity tests are commonly used to estimate hidden (or latent) distribution parameters. For example, upon producing a batch of explosives it is assumed that each explosive specimen has a critical threshold value that, upon surpassing, will result in a positive response (detonation, a 'go'). The critical threshold values are assumed to be a continuous variable with an unknown probability density function. The difficulty of this problem arises because the critical threshold of each specimen cannot be directly measured. Consider the case of determining the yield or fracture threshold of a given material. These thresholds can be directly measured by slowing increasing the stress (or rather force) applied to the specimen with a tensile testing machine and noting the stress at which yielding or fracture occurs. An analogous test does not exist when trying to determine the critical threshold values of explosives.

When testing the sensitivity of explosives to impact or shock, one can impose a non-varying stress level (input pressure, drop-weight height, etc.) on the test article (explosive specimen) and note whether a response (detonation, 'go' or 'no-go') occurs. Unlike the fracture case, the input level cannot be slowly augmented until detonation is achieved. Additionally, if a positive response is not achieved the specimen cannot simply be re-tested at a higher input level because the specimen may have been damaged by the original test. Damage (void creation, phase change, etc.) is theorized to alter the sensitivity, or critical threshold, of the specimen – which is obviously unwanted since this is exactly the value one wishes to measure.

Thus, sensitivity tests are herein defined as methods to estimate the parameters which prescribe the probability density function of critical threshold values; where these critical thresholds cannot be directly measured (hidden, latent).

In 1989, Barry T. Neyer published a paper outlining a 'new sensitivity' test and detailed its advantages over previous methods such a Probit, Bruceton, Robbins-Monro, and Langlie [1]. Later, in 1994, Neyer published an updated version of his sensitivity test; altering the sequential design algorithm (described later) [2]. The test assumes that the distribution of critical thresholds is modelled by a normal or logistic probability density function (pdf), or can be easily transformed to make the distribution normal. Normal and logistic distributions are parameterized by two variables, μ and σ. Neyer estimates these parameters using maximum likelihood estimators. As a point of notation, $\hat{\mu}$ and $\hat{\sigma}$ are the *estimates* or *estimators* of the true parameters μ and σ. An in-depth description of the maximum likelihood estimation to sensitivity tests is attributed to Cornfiled and Mantel [3] and Golub and Grubbs [4] - with the method originating in earlier publications by Dixon and Mood [5] and by Finney [6], Bliss [7], and Fisher [8].

Once the maximum likelihood estimates (MLEs) have been calculated, Neyer proposed a sequential design in which the next stimulus level to be tested is that which maximizes the determinate of the Fisher information matrix [2]. This essentially attempts to simultaneously minimize the variances of the estimated parameters, $\hat{\mu}$ and $\hat{\sigma}$, by decreasing the Cramér-Rao lower bound (CRLB) of said parameters. Although, as will be described later, the MLE of σ is not unbiased - and therefore is not bounded by the CRLB - minimizing the CRLB will generally decrease the variance of $\hat{\sigma}$ because they are computed from the same likelihood function.

'Unique' maximum likelihood estimates will not be obtained unless the stimulus level of the smallest positive response is less than the stimulus level of the greatest non-response (no "overlap" in responses) [9]. That is, the MLE will return a value of zero for $\hat{\sigma}$ and any value between the greatest non-response level and smallest positive response level for $\hat{\mu}$. Neyer's sequential design algorithm overcomes this deficiency by using an expandable binary-search algorithm to test points until overlap is achieved.

While much of the information outlined in Neyer's papers - in terms of the statistics - was previously published, Neyer greatly impacted the field by bringing the test to the energetic materials community, comparing the test to all other known methods, creating a robust sequential design algorithm, and producing a commercial software implementation of the test.

Specific descriptions of the maximum likelihood estimators, the information matrix, and the sequential design algorithm are given below.

2 Background

2.1 Model Overview

Using 'plate' notation, the current problem framework is shown graphically in figure 1. In this figure i is the specimen number and there are n total specimens to be tested. x_i is the stimulus level at which specimen i is tested and y_i is the specimen response (binary outcome). h_i is the actual threshold of specimen i (hidden variable), and all of the specimens are assumed to have threshold values that originate from a distribution uniquely determined by two parameters, μ and σ.

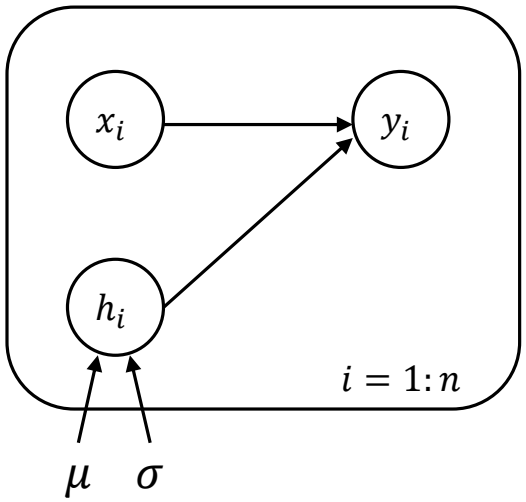


Figure 1: Graph demonstrating interaction of sensitivity model random variables assuming that the threshold distribution is parameterized by two variables, μ and σ.

Under this model the probability y_i is easily described as

$$p(y_i = 1|x_i, h_i) = \begin{cases} 1, & \text{if } x_i \geq h_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

or

$$p(y_i = 0|x_i, h_i) = \begin{cases} 1, & \text{if } x_i < h_i \\ 0, & \text{otherwise .} \end{cases} \tag{2}$$

Now, maintaining generality, it's assumed that the distribution h_i is given by a set of parameters $\tilde{\theta}$, which is written as $p(h_i|\tilde{\theta})$. Following the rules of probability, the joint distribution of y_i and h_i is

$$p(y_i, h_i|x_i, \vec{\theta}) = p(y_i|x_i, h_i)p(h_i|\vec{\theta}). \tag{3}$$

h_i can be marginalized out to produce a probability measure for y_i that is given given x_i and $\tilde{\theta}$ as

$$p(y_i|x_i, \vec{\theta}) = \int_{-\infty}^{\infty} p(y_i|x_i, h_i)p(h_i|\vec{\theta})dh_i. \tag{4}$$

Now, a 'nested' description of $p(y_i|x_i, h_i)$ could be produced, but it is easier to work with the case of $p(y_i = 1|x_i, h_i)$ which was intuitively formed and shown in equation 1. Making this substitution into equation 4 yields

$$p(y_i = 1|x_i, \vec{\theta}) = \int_{-\infty}^{\infty} p(y_i = 1|x_i, h_i)p(h_i|\vec{\theta})dh_i. \tag{5}$$

According to equation1, $p(y_i = 1)$ is equal to one when $x_i \geq h_i$ and is zero otherwise. Thus, in the integral of equation 5 the integrand is zero for all values $h_i > x_i$ so that x_i can replace ∞ as the integral upper limit.

$$p(y_i = 1|x_i, \vec{\theta}) = \int_{-\infty}^{x_i} (1)p(h_i|\vec{\theta})dh_i. \tag{6}$$

Once $p(y_i = 1)$ is known, $p(y_i = 0)$ is simply given by $1 - p(y_i = 1)$ and the likelihood function can be formed. In Neyer and preceeding works, it is assumed that h_i is described by either a normal or logistic distribution parameterized by μ and σ. If the the normal distribution assumption is made then $p(h_i) \sim N(\mu, \sigma)$ and equation 6 becomes

$$p(y_i = 1|x_i, \vec{\theta}) = \int_{-\infty}^{x_i} N(\mu, \sigma)dh_i = \Phi(x_i|\mu, \sigma) \tag{7}$$

where $\Phi(x_i|\mu, \sigma)$ is the cumulative distribution function (cdf) of a normal distribution with given parameters μ and σ. Likewise, if a logistic distribution assumption is made then equation 6 reduces to

$$p(y_i = 1|x_i, \vec{\theta}) = \int_{-\infty}^{x_i} Logistic(\mu, \sigma)dh_i = S(x_i|\mu, \sigma) \tag{8}$$

where $S(x_i|\mu, \sigma)$ is the cdf of the logistic distribution;

$$S(x_i|\mu, \sigma) = \frac{1}{1 + exp\left(\frac{-(x_i - \mu)}{\sigma}\right)}. \tag{9}$$

2.2 Derivation: Maximum Likelihood Estimators

Here, we will used the notation used by Neyer and the previous authors.

Let $L^{(i)}$ represent the stimulus level to be tested on specimen i. The standardized level is then given by $z^{(i)} = \frac{L^{(i)} - \mu}{\sigma}$, where μ and σ are the parameters that describe the assumed normal distribution of critical threshold values. Then the probability of observing a positive response in a randomly selected specimen is

$$p(z^{(i)}) = \int_{-\infty}^{z^{(i)}} f(t)dt \tag{10}$$

where,

$$f(t) = \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} \tag{11}$$

Let $y^{(i)}$ represent the observed response at stimulus level $L^{(i)}$; and let $y^{(i)} = 1$ and $y^{(i)} = 0$ for a positive and negative response ('go', 'no go'), respectively. Because the response is described by a binary variable, the likelihood function follows the form of a binomial model

$$L(\mu, \sigma) = \prod_{i=1}^{n} p(z^{(i)})^y (1 - p(z^{(i)}))^{1-y} \tag{12}$$

where n is the number of specimens tested. If the same stimulus level is tested more than once then let $N^{(i)}$ and $M^{(i)}$ represent the number of positive and negative responses observed at $L^{(i)}$, respectively. Then, the total number of test performed at level $L^{(i)}$ is equal to $(N^{(i)} + M^{(i)})$. Using this notation, equation 12 can be written as

$$L(\mu, \sigma) = \prod_{i=1}^{n} \binom{N^{(i)} + M^{(i)}}{N^{(i)}} p(z^{(i)})^{N^{(i)}} (1 - p(z^{(i)}))^{M^{(i)}} \tag{13}$$

Additionally, $(1 - p(z^{(i)}))$ is equivalent to $p(-z^{(i)})$ and can be referred to as $q^{(i)}$ so that the likelihood function is

$$L(\mu, \sigma) = \prod_{i=1}^{n} \binom{N^{(i)} + M^{(i)}}{N^{(i)}} p(z^{(i)})^{N^{(i)}} q(z^{(i)})^{M^{(i)}} \tag{14}$$

The values of μ and σ that maximize the likelihood function for the data set $\{(\mathsf{L}^{(1)}, \mathsf{N}^{(1)}, \mathsf{M}^{(1)}),\ (\mathsf{L}^{(2)}, \mathsf{N}^{(2)}, \mathsf{M}^{(2)}),\ \ldots\ (\mathsf{L}^{(\mathsf{n})}, \mathsf{N}^{(\mathsf{n})}, \mathsf{M}^{(\mathsf{n})})\}$ are the MLEs $\hat{\mu}$ and $\hat{\sigma}$. Note that in the notation of equation 14, n now represents the number of unique stimulus levels tested and not the total number of tests performed.

The parameters that maximize the likelihood function are the same as those that maximize the natural logarithm of the likelihood function because the natural logarithm is a monotonically increasing function. The log-likelihood function is often easier to work with analytically, and additionally, it is the log-likelihood function that is used in the definition of the Fisher information matrix used later. Thus, the log-likelihood function is

$$l = log(L) = \sum_{i-1}^{n} \left[log\binom{N^{(i)} + M^{(i)}}{N^{(i)}} + N^{(i)} log\left(p(z^{(i)})\right) + M^{(i)} log\left(q(z^{(i)})\right) \right]. \tag{15}$$

The first order partial derivatives of equation 15 are useful to compute the MLE, as all of the first order partial derivatives must equal zero when l is maximized. The are derived as follows:

$$\frac{\partial l}{\partial \mu} = \sum_{i=1}^{n} \left[N^{(i)} \frac{\partial \left(log(p(z^{(i)}))\right)}{\partial p(z^{(i)})} \frac{\partial p(z^{(i)})}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \mu} + M^{(i)} \frac{\partial \left(log(q(z^{(i)}))\right)}{\partial q(z^{(i)})} \frac{\partial q(z^{(i)})}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \mu} \right] \tag{16}$$

$$\frac{\partial l}{\partial \sigma} = \sum_{i=1}^{n} \left[N^{(i)} \frac{\partial \left(log(p(z^{(i)}))\right)}{\partial p(z^{(i)})} \frac{\partial p(z^{(i)})}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \sigma} + M^{(i)} \frac{\partial \left(log(q(z^{(i)}))\right)}{\partial q(z^{(i)})} \frac{\partial q(z^{(i)})}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \sigma} \right] \tag{17}$$

As seen, equations 16 and 17 are derived by the continuous, albeit tedious, application of the 'chain rule' in calculus. The partial derivatives found in these equations are formulated as

$$\frac{\partial \left(log(p(z^{(i)}))\right)}{\partial p(z^{(i)})} = \frac{1}{p(z^{(i)})} \tag{18}$$

$$\frac{\partial \left(log(q(z^{(i)}))\right)}{\partial q(z^{(i)})} = \frac{1}{q(z^{(i)})} \tag{19}$$

$$\frac{\partial p(z^{(i)})}{\partial z^{(i)}} = f(z^{(i)}) \tag{20}$$

$$\frac{\partial (z^{(i)})}{\partial z^{(i)}} = -f(z^{(i)}) \tag{21}$$

$$\frac{\partial z^{(i)}}{\partial \mu} = \frac{-1}{\sigma} \tag{22}$$

$$\frac{\partial z^{(i)}}{\partial \sigma} = \frac{-(L^{(i)} - \mu)}{\sigma^2} \tag{23}$$

Substituting equations 18 – 23 into equations 16 and 17 yields

$$\frac{\partial l}{\partial \mu} = \sum_{i=1}^{n} \left[N^{(i)} \frac{1}{p(z^{(i)})} f(z^{(i)}) \frac{-1}{\sigma} + M^{(i)} \frac{1}{q(z^{(i)})} f(z^{(i)}) \frac{1}{\sigma} \right] \tag{24}$$

$$\frac{\partial l}{\partial \sigma} = \sum_{i=1}^{n} \left[N^{(i)} \frac{1}{p(z^{(i)})} f(z^{(i)}) \frac{-(L^{(i)} - \mu)}{\sigma^2} + M^{(i)} \frac{1}{q(z^{(i)})} f(z^{(i)}) \frac{L^{(i)} - \mu}{\sigma^2} \right] \tag{25}$$

These derivatives are useful as they will almost certainly be used in any optimization routine or algorithm in order to calculate the parameter values which maximize l.

2.3 Derivation: Information Matrix

The Fisher information matrix is defined as

$$INF = -E\left[H[log(L)]\right] \tag{26}$$

where E$[\cdot]$ is the expectation operator, H$[\cdot]$ is the Hessian, and L is the likelihood function. The log-likelihood function was previously defined as l and the Hessian is a matrix of second partial derivatives. The partial derivatives are taken with respect to the parameters being estimated; in this case, μ and σ. Therefore,

$$H\left[log(L)\right] = H\left[l\right] = \begin{bmatrix} \frac{\partial^2 \mathsf{l}}{\partial \mu^2} & \frac{\partial^2 \mathsf{l}}{\partial \sigma \partial \mu} \\ \frac{\partial^2 \mathsf{l}}{\partial \mu \partial \sigma} & \frac{\partial^2 \mathsf{l}}{\partial \sigma^2} \end{bmatrix} \tag{27}$$

Let the elements of the information matrix be described as

$$INF = \begin{bmatrix} \mathsf{a}_{11} & \mathsf{a}_{12} \\ \mathsf{a}_{21} & \mathsf{a}_{22} \end{bmatrix} \tag{28}$$

and the inverse of the information matrix be

$$[INF]^{-1} = \begin{bmatrix} \mathsf{b}_{11} & \mathsf{b}_{12} \\ \mathsf{b}_{21} & \mathsf{b}_{22} \end{bmatrix} \tag{29}$$

Under this description b_{11} and b_{22} are the Cramér-Rao lower bounds (CRLBs) for $\hat{\mu}$ and $\hat{\sigma}$, respectively. However, the CRLB only applies to estimators that are unbiased. If $\hat{\mu}$ is the estimator for μ and $\hat{\sigma}$ for σ, then $\hat{\mu}$ and $\hat{\sigma}$ are unbiased estimators of μ and σ if E$[\hat{\mu}] = \mu$ and E$[\hat{\sigma}] = \sigma$. Rather than attempt to solve for $\hat{\mu}$ and $\hat{\sigma}$ when equations 24 and 25 are equal to zero and then take the expectation of those expressions, a simple simulation is run with synthetic data.

Consider the latent random variable, which represents the critical threshold values, to be given by a normal distribution $\sim$ N(μ, σ). For the simulation, let $\mu = 40$ and $\sigma = 3$. n stimulus levels are tested; stimulus levels are spaced equidistant from each other between the range of $[41, 49]$. The outcome of each test is decided randomly via a Monte-Carlo simulation. For every value of n the MLE of $\hat{\mu}$ and $\hat{\sigma}$ is calculated and this simulation is repeated 2000 times (holding n constant); the values of $\hat{\mu}$ and $\hat{\sigma}$ are averaged together in order to estimate E$[\hat{\mu}]$ and E$[\hat{\sigma}]$. The results for $\mathsf{n} = 3$ to 39 are shown in figure 2.3. This figure demonstrates that $\hat{\mu}$ is an unbiased estimator but $\hat{\sigma}$ is a biased estimator; although the bias decreases with increasing sample size n. Neyer estimated the bias of $\hat{\sigma}$ over a wide variety of designs [2] (permutations of good and poor initial guesses of the parameters) and approximated the relative bias as

$$relative\ bias = \frac{-3.5}{n}. \tag{30}$$

However, despite the biasedness of $\hat{\sigma}$, minimizing the CRLB will decrease the variance of $\hat{\sigma}$ since both the CRLB and $\hat{\sigma}$ are derived from the same likelihood function.

Moving forward, minimizing the variance of $\hat{\mu}$ and $\hat{\sigma}$, or decreasing the values of b_{11} and b_{22}, is related to maximizing the values of a_{11} and a_{22} in the information matrix (*see last paragraph of this section*). Referring back to equations 26 and 27, it can be seen that these terms are

$$a_{11} = -E\left[\frac{\partial^2 l}{\partial \mu^2}\right] \tag{31}$$

$$a_{22} = -E\left[\frac{\partial^2 l}{\partial \sigma^2}\right] \tag{32}$$

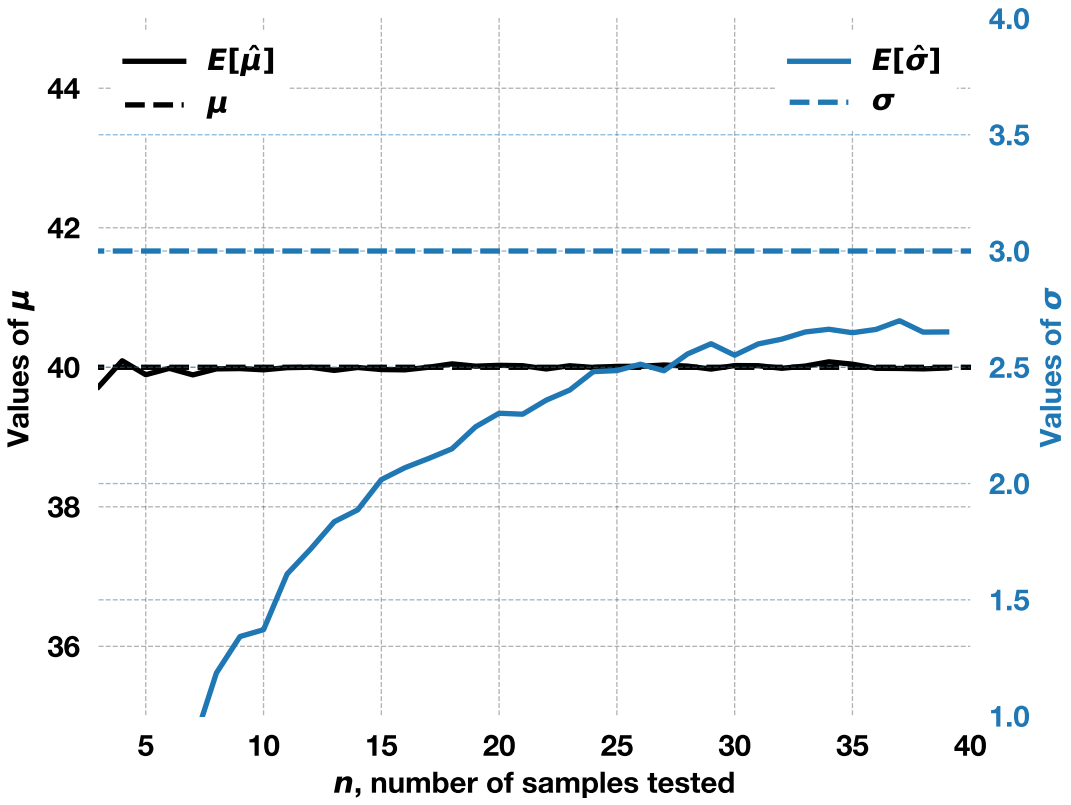


Figure 2: Expected values of estimators $\hat{\sigma}$ and $\hat{\sigma}$ for simulated data

recalling that l is the log-likelihood function given by equation 15. A derivation of the a_{11} term is provided here starting from equation 24. The end result of the a_{12} and a_{22} terms are given but an explicit derivation is not provided as these follow a similar form to the derivation of a_{11}.

$$\frac{\partial^2 l}{\partial \mu^2} = \sum_{i=1}^{n} \left[N^{(i)} \frac{-1}{\sigma} \frac{\partial \left(\frac{f(z^{(i)})}{p(z^{(i)})} \right)}{\partial \mu} + M^{(i)} \frac{-1}{\sigma} \frac{\partial \left(\frac{-f(z^{(i)})}{q(z^{(i)})} \right)}{\partial \mu} \right] \tag{33}$$

$$\frac{\partial \left(\frac{f(z^{(i)})}{p(z^{(i)})} \right)}{\partial \mu} = \frac{1}{p(z^{(i)})} \frac{\partial \left(f(z^{(i)}) \right)}{\partial \mu} + f(z^{(i)}) \frac{-1}{\left(p(z^{(i)}) \right)^2} \frac{\partial \left(p(z^{(i)}) \right)}{\partial \mu} \tag{34}$$

$$\frac{\partial \left(\frac{-f(z^{(i)})}{q(z^{(i)})} \right)}{\partial \mu} = \frac{-1}{q(z^{(i)})} \frac{\partial \left(f(z^{(i)}) \right)}{\partial \mu} + f(z^{(i)}) \frac{1}{\left(q(z^{(i)}) \right)^2} \frac{\partial \left(q(z^{(i)}) \right)}{\partial \mu} \tag{35}$$

$$\frac{\partial \left(f(z^{(i)}) \right)}{\partial \mu} = \frac{\partial f(z^{(i)})}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \mu} = z^{(i)} f(z^{(i)}) \frac{1}{\sigma} \tag{36}$$

$$\frac{\partial \left(p(z^{(i)}) \right)}{\partial \mu} = \frac{\partial \left(p(z^{(i)}) \right)}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \mu} = f(z^{(i)}) \frac{-1}{\sigma} \tag{37}$$

$$\frac{\partial \left(q(z^{(i)}) \right)}{\partial \mu} = \frac{\partial \left(q(z^{(i)}) \right)}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial \mu} = f(z^{(i)}) \frac{1}{\sigma} \tag{38}$$

Making the substitutions results in

$$\frac{\partial^2 l}{\partial \mu^2} = \sum_{i=1}^{n} \left\{ N^{(i)} \frac{-1}{\sigma} \left[\frac{1}{p(z^{(i)})} z^{(i)} f(z^{(i)}) \frac{1}{\sigma} + f(z^{(i)}) \frac{1}{\left(p(z^{(i)})\right)^2} f(z^{(i)}) \frac{1}{\sigma} \right] + M^{(i)} \frac{-1}{\sigma} \left[\frac{-1}{q(z^{(i)})} z^{(i)} f(z^{(i)}) \frac{1}{\sigma} + f(z^{(i)}) \frac{1}{\left(q(z^{(i)})\right)^2} f(z^{(i)}) \frac{1}{\sigma} \right] \right\}. \tag{39}$$

And now, after some simplification the expression becomes

$$\frac{\partial^2 l}{\partial \mu^2} = \sum_{i=1}^{n} \left\{ N^{(i)} \frac{1}{\sigma^2} \frac{1}{p(z^{(i)})} f(z^{(i)}) \left[-z^{(i)} - \frac{f(z^{(i)})}{p(z^{(i)})} \right] + M^{(i)} \frac{1}{\sigma^2} \frac{1}{q(z^{(i)})} f(z^{(i)}) \left[+z^{(i)} - \frac{f(z^{(i)})}{q(z^{(i)})} \right] \right\}. \tag{40}$$

Applying the expectation operator and negating

$$-E\left[\frac{\partial^2 l}{\partial \mu^2}\right] = \sum_{i=1}^{n} \left\{ E[N^{(i)}] \frac{1}{\sigma^2} \frac{1}{p(z^{(i)})} f(z^{(i)}) \left[+z^{(i)} + \frac{f(z^{(i)})}{p(z^{(i)})} \right] + E[M^{(i)}] \frac{1}{\sigma^2} \frac{1}{q(z^{(i)})} f(z^{(i)}) \left[-z^{(i)} + \frac{f(z^{(i)})}{q(z^{(i)})} \right] \right\}. \tag{41}$$

and noting that

$$E[N^{(i)}] = p(z^{(i)})(N^{(i)} + M^{(i)}) \tag{42}$$

$$E[M^{(i)}] = q(z^{(i)})(N^{(i)} + M^{(i)}) \tag{43}$$

after substitution the expression becomes

$$-E\left[\frac{\partial^2 l}{\partial \mu^2}\right] = \sum_{i=1}^{n} \left\{ (N^{(i)} + M^{(i)}) \frac{1}{\sigma^2} f(z^{(i)}) \left[z^{(i)} + \frac{f(z^{(i)})}{p(z^{(i)})} - z^{(i)} + \frac{f(z^{(i)})}{q(z^{(i)})} \right] \right\}. \tag{44}$$

This expression is further simplified to its final form

$$a_{11} = -E\left[\frac{\partial^2 l}{\partial \mu^2}\right] = \sum_{i=1}^{n} \left\{ (N^{(i)} + M^{(i)}) \frac{1}{\sigma^2} f(z^{(i)})^2 \left[\frac{1}{p(z^{(i)})} + \frac{1}{q(z^{(i)})} \right] \right\}. \tag{45}$$

The a_{12} (same as a_{21}) and a_{22} terms are given as

$$a_{12} = -E\left[\frac{\partial^2 l}{\partial \mu^2}\right] = \sum_{i=1}^{n} \left\{ (N^{(i)} + M^{(i)}) \frac{z^{(i)}}{\sigma^2} f(z^{(i)})^2 \left[\frac{1}{p(z^{(i)})} + \frac{1}{q(z^{(i)})} \right] \right\}. \tag{46}$$

$$a_{22} = -E\left[\frac{\partial^2 l}{\partial \mu^2}\right] = \sum_{i=1}^{n} \left\{ (N^{(i)} + M^{(i)}) \frac{\left(z^{(i)}\right)^2}{\sigma^2} f(z^{(i)})^2 \left[\frac{1}{p(z^{(i)})} + \frac{1}{q(z^{(i)})} \right] \right\}. \tag{47}$$

Next, these 'a terms' (a_{11}, a_{12}, a_{22}) are plotted in figure 2.3 for a single specimen. This plot demonstrates that a_{11} and a_{22} cannot be maximized simultaneously. However, a good amount of information about μ and σ is achieved if the test is conducted at $\mu \pm \sigma$. Quoting directly from Neyer[2],

> A D-optimal result will be obtained when the determinant of the information matrix is maximized.
>
> ...
>
> Since the off-diagonal terms [a_{12} and a_{21} here] of the matrix are typically small compared to the diagonal terms, a D-optimal test will also approximately minimize the the product of the asymptotic variances of both μ and σ.

Thus, when choosing the next stimulus level to test in a sequential design of experiments, Neyer's algorithm elects the level which maximizes the determinate of the information matrix because this relates to minimizing the variances of $\hat{\mu}$ and $\hat{\sigma}$.

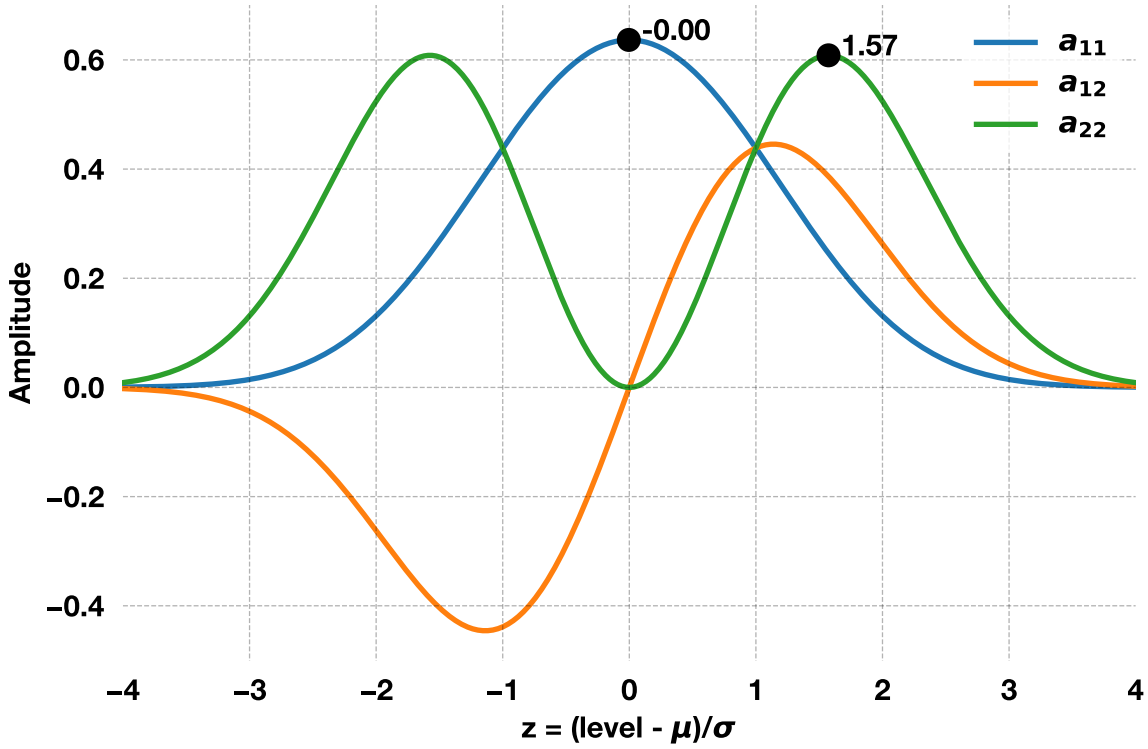


Figure 3: Values of the elements in the information matrix for a single specimen

2.4 Algorithm Flowchart

As noted, Neyer's sensitivity test chooses the stimulus level which maximizes the information matrix. However, in order to compute the information matrix, at least one point needs to be tested and estimates of μ and σ are required.

When beginning an experiment, it is possible that the experimenter has no prior knowledge about μ or σ; this scenario becomes more likely when testing a new material. Additionally, as mentioned, unique maximum likelihood estimators will only occur if there is overlap in the test outcomes. Neyer's algorithm is robust and overcomes these deficiencies and a brief summary of its is provided here.

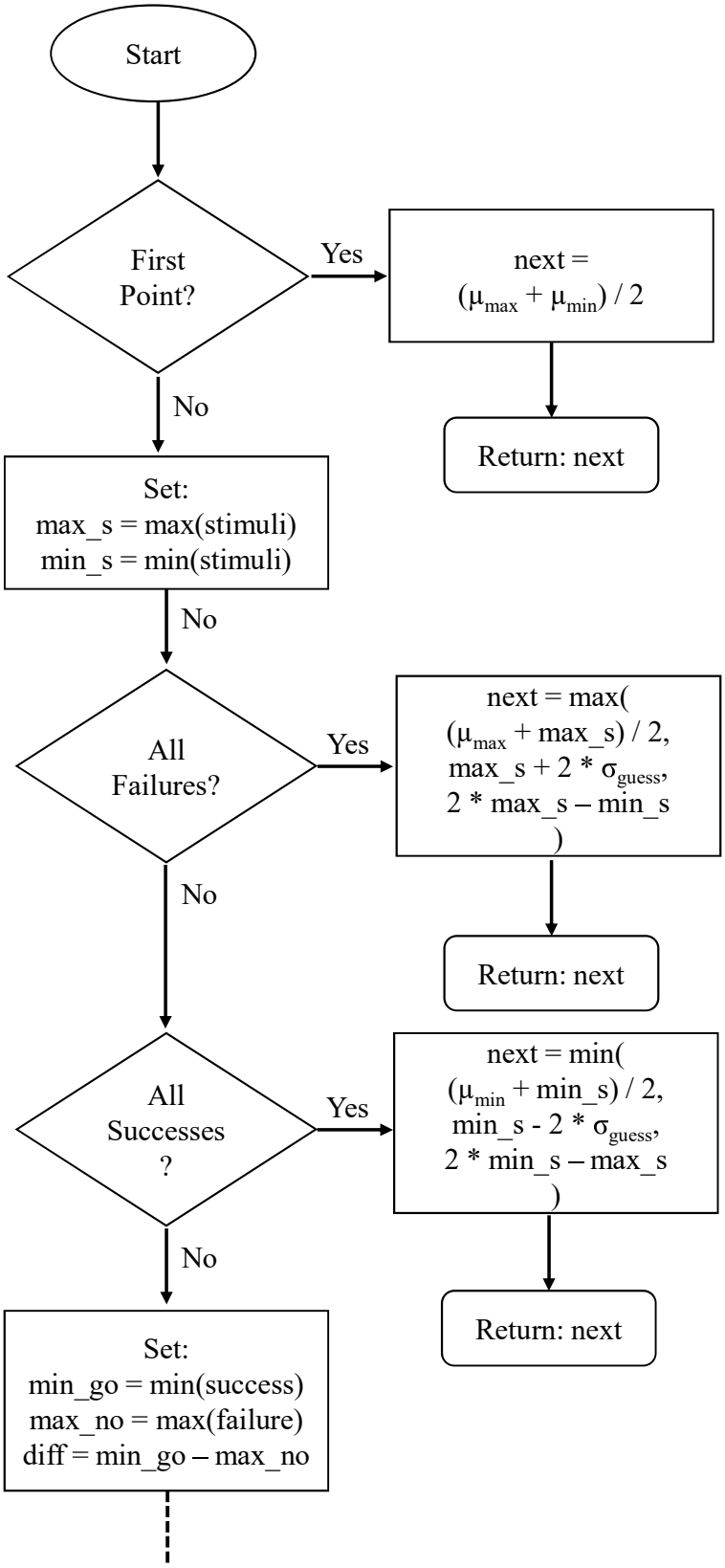
To initialize the algorithm, the user needs only to provide a guess as to the lower and upper bounds of the expected μ and a guess for σ. The algorithm begins with an binary search between the given bounds. The bounds are updated so that lower bound becomes the maximum stimulus level that produced a negative response and the upper bound is the minimum stimulus level that provides a positive response. This procedure is repeated until difference in the bounds (upper bound minus lower bound) becomes less than the guess provided for σ. It should be noted that if the algorithm only detects a single type of response, the bounds are automatically adapted to increase the search length until mixed responses are observed.

Next, the algorithm uses the midpoint between highest stimulus level returned a negative response and the lowest stimulus level that returned a positive response as an estimate for μ. This μ along with the original guessed σ, σ_{guess}, and the data taken thus far (set of stimulus levels and their respective responses) are used to maximize the determinate of the information matrix. The stimulus level that maximizes the determinate of the information matrix is the next level tested. This procedure is repeated until overlap is achieved, and on each repetition σ_{guess} is updated by $\sigma_{\text{guess}} = 0.8 * \sigma_{\text{guess}}$. It should be noted that since σ_{guess} is decreasing, it is possible that the difference ([lowest stimulus that resulted in a positive response] - [highest stimulus that resulted in a negative response]) becomes greater than σ_{guess}. In this case, the algorithm resumes a binary

search until this difference becomes less than the updated σ_{guess} or until overlap is achieved.

Once overlap is achieved, the guessed values for μ and σ are no longer used, but the maximum likelihood estimates are calculated. The algorithm performs a brief check to ensure that these estimates are not 'wild' and then uses these to determine the information matrix. Again, the stimulus level which maximizes the determinate of the information matrix is suggested as the next tested level. From this point, the algorithm continues on indefinitely until the experimenter as tested as sufficient number of specimens or until the set of specimens have been depleted.

As a visual aid, figure 4 gives a slightly adapted flowchart of Neyer's sequential design of experiments algorithm. In the original version, the maximum likelihood estimates of μ and σ are computed after the binary search has terminated regardless of whether overlap has been achieved. Without overlap, non-unique estimates are returned and $\hat{\sigma}$ should be zero. $\hat{\sigma} = 0$ is used as a decision flag: if true, then the current guesses of μ and σ are provided to the information matrix; and if false, the unique maximum likelihood estimates are passed to the information matrix (after 'wild' check, of course). However, I did not wish to rely on the MLE of σ to be zero as numerical precision or the choice of optimizer may preclude $\hat{\sigma}$ from being precisely zero. In my adapted algorithm, which is represented by the flowchart, the MLEs of μ and σ are only calculated if overlap is achieved. The occurrence of overlap is determined by simply looking at the difference between the highest negative level and lowest positive level; a quantity which is already calculated and stored. If this difference is negative, then overlap in the data is present. Without presenting proof, I believe this alteration will provide a more stable decision block between various coding languages, machine types, and choice of optimization routine.


Start
First Point?
Yes
next = (μmax + μmin) / 2
Return: next
No
Set: max_s = max(stimuli) min_s = min(stimuli)
No
All Failures?
Yes
next = max((μmax + max_s) / 2, max_s + 2 * σguess, 2 * max_s – min_s)
Return: next
No
All Successes ?
Yes
next = min((μmin + min_s) / 2, min_s - 2 * σguess, 2 * min_s – max_s)
Return: next
No
Set: min_go = min(success) max_no = max(failure) diff = min_go – max_no

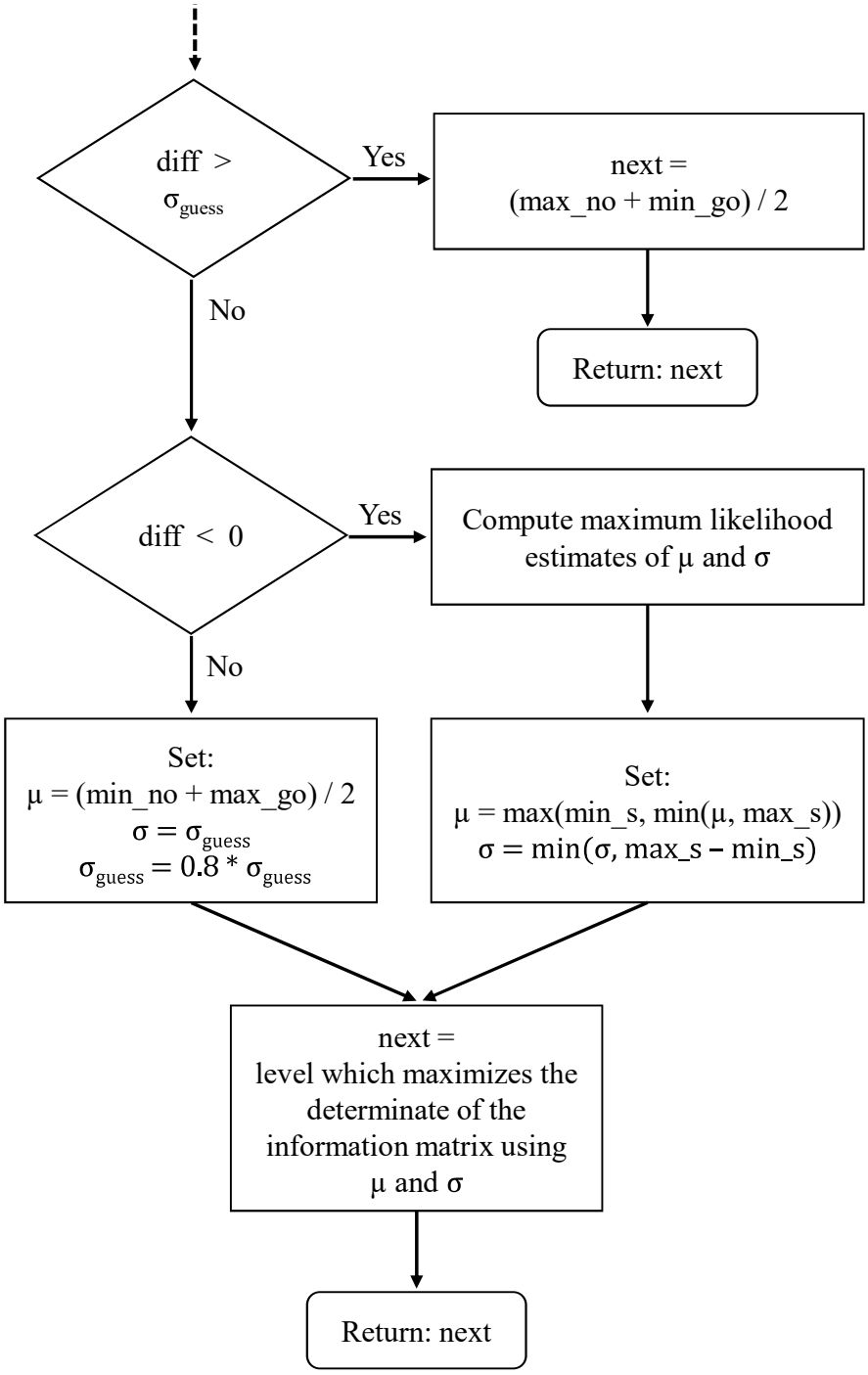


Figure 4: Flowchart of Neyer's sequential design algorithm

3 Code

3.1 Requirements

See the requirements.txt file for all required packages.

3.2 Folder Contents

The root folder is named *Neyer_alg*. This folder contains:

1) This manual.
2) A file *neyer_alg.py* which is the module that holds all of the code.
3) File *mle_example.py*. A full working example to compute the maximum likelihood estimates given a data set.
4) File *sequential_example.py*. An example of using Neyer's sequential design algorithm to suggest new test levels.
5) File *log.txt*. An example log file produced when running the sequential design algorithm.

3.3 Using the Code

The module *neyer_alg.py* contains all of the code and consists of two classes, *NeyerMLE* and *NeyerSeq*. These classes, their input options, and methods are detailed here.

3.3.1 NeyerMLE

The NeyerMLE class is used to estimate the maximum likelihood estimates $\hat{\mu}$ and $\hat{\sigma}$ given a data set. Instance initialization is given by the following signature:

```
1 est = NeyerMLE(method='BFGS', inverted=False, num_restarts=3)
```

Here, the specific object is named *est* (short for estimator), but any name can be supplied.

Inputs

- *method* [optional]. The optimization algorithm used to maximize the log likelihood function. Available choices: *BFGS* (default), *cg*, or *SLSQP*.
- *inverted* [optional]. Determines if positive responses ('go' outcomes) are more likely with a decrease in stimulus level. Default is False; that is, and increase in stimulus level will produce more positive responses.
- *num_restarts* [optional]. Number of times to run the optimization routine with different initial starting vectors. Likely not necessary. (If the hidden distribution is assumed to be logistic, the likelihood function is convex. I'm not sure if this also holds true if the hidden distribution is assumed to be normal.) At any rate, restarting the optimization increases the ability of the algorithm to find a global maximum.

Methods

- *fit(self, X, Y)*. Train the estimates $\hat{\mu}$ and $\hat{\sigma}$ on the stimulus levels X and outcomes Y. X and Y can be lists, tuples, or arrays and must be of the same length.

- *get_estimators(self)*. Returns a tuple of the estimates ($\hat{\mu}$, $\hat{\sigma}$).
- *print_estimators(self)*. Prints the estimates $\hat{\mu}$, and $\hat{\sigma}$.
- *predict_probability(self, X)*. Returns the probability of a positive response ('go') at test level X. X can be a single value or list-like.

3.3.2 NeyerSeq

The NeyerSeq class is used to run the Neyer sequential design algorithm. Instance initialization is given by the following signature:

```
seq = NeyerSeq(mu_min=None, mu_max=None, sigma_g=None, inverted=False,
               precision=8, resolution=None, lower_bound=None,
               upper_bound=None, hist=False, log_file=None)
```

Again, here, the specific object is named *seq* (sequential) but any name can be supplied.

Inputs

- *mu_min* [*required]. Initial guess for the lower bound of μ.
- *mu_max* [*required]. Initial guess for the upper bound of μ.
- *sigma_g* [*required]. Initial guess for σ.

  *If not provided during initialization, the user will be immediately prompted for these values.
- *inverted* [optional]. Determines if positive responses ('go' outcomes) are more likely with a decrease in stimulus level. Default is False; that is, and increase in stimulus level will produce more positive responses.
- *precision* [optional]. Number of decimal places to carry in intermediate calculations.
- *resolution* [optional]. Resolution (smallest step size) of the testing apparatus.
- *lower_bound* [optional]. The lower bound on the testing apparatus.
- *upper_bound* [optional]. The upper bound on the testing apparatus.
- *hist* [optional]. Boolean; flag to determine if the determinate of the information matrix versus stimulus level should be stored.
- *log_file* [optional]. Path and name of file to which output can be written. Can be used to monitor the algorithm.

Methods

- *next_pt(self)*. Return the next stimulus level at which to test suggested by the algorithm.
- *result(self, res, pt)*. Inform the algorithm of the test outcome *res* and level *pt*.
- *print_estimators(self)*. Prints the estimates $\hat{\mu}$, and $\hat{\sigma}$.
- *predict_probability(self, X)*. Returns the probability of a positive response ('go') at test level X. X can be a single value or list-like.
- *loop(self, iter=None)*. A convenient routine to cyclically call the *next_pt* method and then query the user for the result. The optional input *iter* represents the number of times to cycle through the algorithm (number of specimens to test). If *None*, the method will cycle indefinitely until the user supplies the command *'end'*.

3.4 Caveats

1) None

3.5 Contact

Bug reports and general questions can be sent to alex.casey.13 at gmail.com.

4 Minimal Working Example

The *Neyer_alg* folder contains two example files: *mle_example.py* and *sequential_example.py*. Both example files are given in their entirety below.

4.0.1 NeyerMLE

```
import numpy as np
import matplotlib.pyplot as plt
from neyer_alg import NeyerMLE

x_data = np.array([15.44,11.82,11.36,10.33,8.98,8.5,
                   8.03,7.83,7.55,7.2,6.16,5.93])
y_data = np.array([0,0,0,0,0,0,1,0,1,0,1,1])


est = NeyerMLE(inverted=True)
est.fit(x_data, y_data)
est.print_estimators()

x = np.linspace(5,16,100)
p = est.predict_probability(x)
fig, ax = plt.subplots()
ax.scatter(x_data[y_data==0], y_data[y_data==0])
ax.scatter(x_data[y_data==1], y_data[y_data==1])
ax.plot(x, p, '-k', zorder=0)

ax.set_xlabel('Attenuator Length (mm)')
ax.set_ylabel("p('go')")
ax.set_xlim(5,16)

plt.show()
```

Output:

```
mu: 7.578031199374661
sigma: 0.9003367004540552
```

This code, if run correctly, will produce figure 5.

4.0.2 NeyerSeq

```
import numpy as np
import matplotlib.pyplot as plt
from neyer_alg import NeyerSeq
from plot_settings import *

# The following dataset is from Neyer.
# Lower guess for mu: 0.6
# Upper guess for mu: 1.4
# Guess for sigma: 0.1
# Height:  Result:
# 1.00     Failure
# 1.20     Failure
# 1.40     Failure
# 1.80     Failure
# 2.60     Failure
# 4.20     Success
# 3.40     Failure
# 3.80     Failure
# 4.00     Failure
# 4.10     Failure
# 4.28     Failure
# 4.52     Failure
```

```
# 5.55    Success
# 5.24    Failure
# 6.37    Success
# 6.08    Failure
# 7.38    Success
# 7.09    Success
# 6.89    Success
# 6.74    Success
# end
# Estimated Mu: 5.39
# Estimated Sigma: 1.04

#This data set can be run with the following commands.
#User input required.

seq = NeyerSeq(0.6, 1.4, 0.1, resolution=.01,
               log_file='./log.txt')
seq.loop()
print('\n')
seq.print_estimators()
```

Output:

```
mu: 5.392185659427297
sigma: 1.041222837693301
```

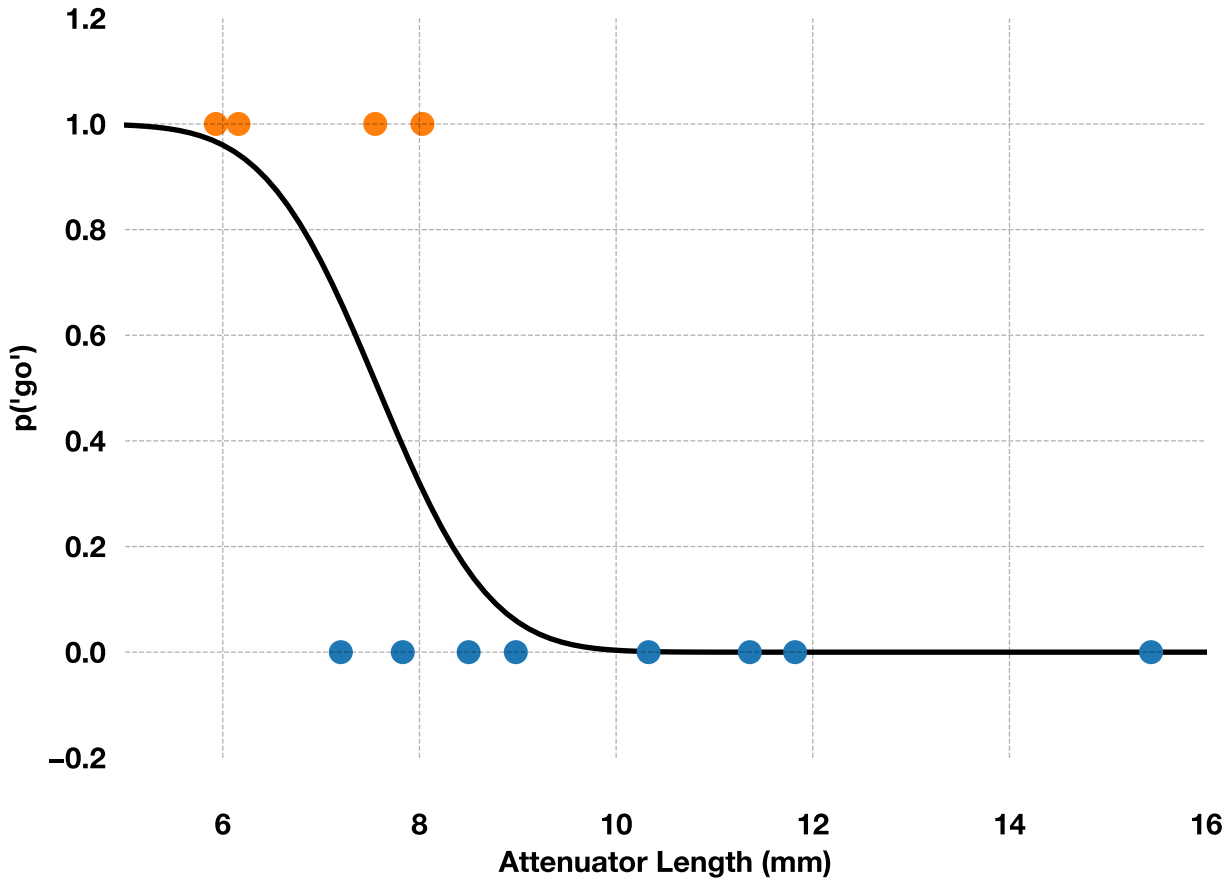


Figure 5: Test results and predictive probability of a positive response from a shock input gap test.

References

[1] B. T. Neyer, “More efficient sensitivity testing,” tech. rep., EG and G Mound Applied Technologies, Miamisburg, OH (USA), 1989.

[2] B. T. Never, “A d-optimality-based sensitivity test,” *Technometrics*, vol. 36, no. 1, pp. 61–70, 1994.

[3] J. Cornfield and N. Mantel, “Some new aspects of the application of maximum likelihood to the calculation of the dosage response curve,” *Journal of the American Statistical Association*, vol. 45, no. 250, pp. 181–210, 1950.

[4] A. Golub and F. E. Grubbs, “Analysis of sensitivity experiments when the levels of stimulus cannot be controlled,” *Journal of the American Statistical Association*, vol. 51, no. 274, pp. 257–265, 1956.

[5] W. J. Dixon and A. M. Mood, “A method for obtaining and analyzing sensitivity data,” *Journal of the American Statistical Association*, vol. 43, no. 241, pp. 109–126, 1948.

[6] D. Finney, “Probit analysis, cambridge university press,” *Cambridge, UK*, 1947.

[7] C. I. Bliss, “The calculation of the dosage-mortality curve,” *Annals of Applied Biology*, vol. 22, no. 1, pp. 134–167, 1935.

[8] C. I. Bliss, “The calculation of the dosage-mortality curve,” *Annals of Applied Biology*, vol. 22, no. 1, p. 149, 1935. Appendix by R. A. Fisher.

[9] M. J. Silvapulle, “On the existence of maximum likelihood estimators for the binomial response models,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 310–313, 1981.