Gaussian Random Noise

The probability density for the Gaussian distribution is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ is the standard deviation. This function has its peak at the mean, and its "spread" increases with the standard deviation.

Some of the problems on the homework expect the sampled data to be corrupted with Gaussian noise. You are provided with the parameters that define the distribution in the form

$$\mathcal{N}(\mu, \sigma)$$

which we can easily implement in Python. For example, to generate N data points with a normal distribution defined by $\mathcal{N}(\mu, \sigma)$, use the following code:

```python
import numpy as np
gauss_noise = np.random.normal(mu,sigma,N)
```

Refer to the `numpy.random.normal` documentation page for more information on generating samples from a normal (Gaussian) distribution.

Vector Norms

Vector norms are non-negative scalar values associated with a vector, usually used to describe the length or magnitude of a vector. The general form for the p-norm of a vector $x = (x_1, x_2, \ldots, x_n)$ is

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

Below are some commonly used norms.

L_1 Vector Norm

The L_1 norm is calculated as the sum of the absolute vector values. This norm is also called the taxicab or Manhattan norm since it can be visualized as the distance a cab drives through the streets of Manhattan (making only right-angle turns). It is given by

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

L_2 Vector Norm

The most common norm is the L_2 or Euclidean norm. In 2-D and 3-D space, it measures as the distance of the shortest straight line that connects two points. It is computed as follows

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}$$

L_∞ Vector Norm

The L_∞ is defined by evaluating the p-norm in the limit as $p \to \infty$. It is also reffered to as the max norm because it is effectively the same as computing the maximum value of the vector. It is calculated with the equation

$$\|x\|_\infty = \max\left(|x_1|, |x_2|, \ldots, |x_n|\right)$$

To learn how to leverage the `numpy` toolbox to compute vector norms, refer to the `numpy.linalg.norm` documentation page.

Weighted Linear Least-Squares

Recall the original linear least-squares minimization problem. For a given set of data points, $[x_n, y_n]$,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and a fitting function, $\hat{y}(x)$, consisting of m linearly independent functions,

$$\hat{y}(x) = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_m f_m(x)$$

we can say the following

$$y \approx \hat{y} = Ac \quad \Longrightarrow \quad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \approx \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \ldots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \ldots & f_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \ldots & f_m(x_n) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

where c_k is the coefficient the scales $f_k(x)$, and $m < n$. The optimal solution to the minimization problem

$$min \sum_{k=1}^{n} \left(y_k - \hat{y}_k\right)^2$$

is given by the equation

$$\boxed{c = \left(A^T A\right)^{-1} A^T y}$$

The method of ordinary least-squares assumes that there is constant covariance in the errors. In other words, the weight (importance) of each data point and its effect on the best-fit curve is equal. The weighted least-squares method breaks this assumption and assigns a specific weight to each data point, essentially assuming that some measurements are more accurate than others. This is done through a diagonal weighing matrix, W,

$$W = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{bmatrix}$$

The unequal weighing on the different measurements that result from applying this weight matrix in the linear least-squares problem forces the solution to prioritize accurate curve-fitting around the data points that were assigned a higher weight. This weighted linear least-squares problem is a modified version of the original least-squares optimization problem and is represented by

$$min \sum_{k=1}^{n} w_k \left(y_k - \hat{y}_k\right)^2$$

The solution to the weighted linear least-squares problem is

$$\boxed{c = \left(A^T W A\right)^{-1} A^T W y}$$