1 Weighted Least Squares

So far we have used MLE in the context of Gaussian noise to justify the optimization formulation of regression problems, such as OLS. Let's apply this dual optimization-probability philosophy to other regression problems, such as **Weighted Least Squares**.

1.1 Optimization View

The basic idea of weighted least squares is the following: we place more emphasis on the loss contributed from certain data points over others - that is, we care more about fitting some data points over others. From an optimization perspective, the problem can be expressed as

$$\vec{w}^*_{WLS} = \arg\min_{\vec{w}\in\mathbb{R}^d} \Big(\sum_{i=1}^{n} \omega_i (y_i - \phi(x_i)^T \vec{w})^2 \Big)$$

This objective is the same as OLS, except that each term in the sum is weighted by a coefficient ω_i. As always, we can vectorize this problem:

$$\vec{w}^*_{WLS} = \arg\min_{\vec{w}\in\mathbb{R}^d} (\vec{y} - A\vec{w})^T \Omega (\vec{y} - A\vec{w})$$

Where the i'th row A is $\phi(x_i)^T$, and $\Omega \in \mathbb{R}^{nxn}$ is a diagonal matrix with $\Omega_{i,i} = \omega_i$.

We rewrite the WLS objective to an OLS objective:

$$\begin{aligned}
\vec{w}^*_{WLS} &= \arg\min_{\vec{w}\in\mathbb{R}^d} (\vec{y} - A\vec{w})^T \Omega (\vec{y} - A\vec{w}) && (1)\\
&= \arg\min_{\vec{w}\in\mathbb{R}^d} (\vec{y} - A\vec{w})^T \Omega^{1/2} \Omega^{1/2} (\vec{y} - A\vec{w}) && (2)\\
&= \arg\min_{\vec{w}\in\mathbb{R}^d} (\Omega^{1/2}\vec{y} - \Omega^{1/2}A\vec{w})^T (\Omega^{1/2}\vec{y} - \Omega^{1/2}A\vec{w}) && (3)
\end{aligned}$$

This formulation is identical to OLS except that we have scaled the data matrix and the observation vector by $\Omega^{1/2}$, and we conclude that

$$\vec{w}^*_{WLS} = \Big((\Omega^{1/2}A)^T (\Omega^{1/2}A) \Big)^{-1} \Big(\Omega^{1/2}A \Big)^T \Omega^{1/2}\vec{y} = (A^T \Omega A)^{-1} A^T \Omega \vec{y}$$

1.2 Probabilistic View

As in MLE, we assume that our observations $\vec{y}$ are noisy, but now suppose that some of the y_i's are more noisy than others. How can we take this into account in our learning algorithm so we can get

a better estimate of the weights?

Our probabilistic model looks like

$$y_i = \phi(x_i)^T \vec{w} + Z_i$$

where the Z_i's are still independent Gaussians random variables, but not necessarily identical: $Z_i \sim \mathcal{N}(0, \sigma_i^2)$. We can morph the problem into an MLE one by scaling the data to make sure all the Z_i's are identically distributed, by dividing by σ_i:

$$\frac{y_i}{\sigma_i} = \frac{\phi(x_i)^T \vec{w}}{\sigma_i} + \frac{Z_i}{\sigma_i}$$

Note that the scaled noise entries are now i.i.d:

$$\frac{Z_i}{\sigma_i} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0,1)$$

We rewrite our original problem as a scaled MLE problem:

$$\vec{w}^*_{WLS} = \arg\min_{\vec{w} \in \mathbb{R}^d} \Big(\sum_{i=1}^{n} \frac{(\frac{y_i}{\sigma_i} - \frac{\phi(x_i)^T \vec{w}}{\sigma_i})^2}{2(1)} \Big) + n \log \sqrt{2\pi}(1)$$

The MLE estimate of this scaled problem is equivalent to the WLS estimate of the original problem:

$$\vec{w}^*_{WLS} = (A^T \Sigma_z^{-\frac{1}{2}} \Sigma_z^{-\frac{1}{2}} A)^{-1} A^T \Sigma_z^{-\frac{1}{2}} \Sigma_z^{-\frac{1}{2}} \vec{y} = (A^T \Sigma_z^{-1} A)^{-1} A^T \Sigma_z^{-1} \vec{y}$$

where

$$\Sigma_z = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \sigma_n^2 \end{bmatrix}$$

Note that as long as no σ is 0, Σ_z is invertible. Note that ω_i from the optimization perspective is directly related to σ_i^2 from the probabilistic perspective: $\omega_i = \frac{1}{\sigma_i^2}$. As the variance σ_i^2 of the noise corresponding to data point i decreases, the weight ω_i increases: we are more concerned about fitting data point i because it is likely to match the true underlying denoised point. Inversely, as the variance σ_i^2 increases, the weight ω_i decreases: we are less concerned about fitting data point i because it is noisy and should not be trusted.

2 Dependent Noise

What if the Z_i's aren't independent? This usually happens for time series data. Again, we assume we have a model for how the noise behaves. The noise entries are not independent, but there is a known process.
e.g. $Z_{i+1} = rZ_i + U_i$ where $U_i \sim \mathcal{N}(0,1)$, i.i.d, $-1 \leq r \leq 1$ (so that it doesn't blow up)

or a "sliding window" example (like echo of audio) where $Z_i = \Sigma r_j U_{i-j}$, $U_i \sim \mathcal{N}(0,1)$.

In general, we can always represent the noise vector as

$$\vec{Z} = R\vec{U}$$

where $\vec{Z} \in \mathbb{R}^n$, $R \in \mathbb{R}^{nxn}$, $\vec{U} \in \mathbb{R}^n$, and $U_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(0,1)$.

$\vec{Z}$ is a **Jointly Gaussian Random Vector**. Our goal now is to derive its probability density formula.

3 Multivariate Gaussians

There are three equivalent definitions of a jointly Gaussian (JG) random vector:

1. A random vector $\vec{Z} = (Z_1, Z_2, \ldots, Z_k)^T$ is JG if there exists a base random vector $\vec{U} = (U_1, U_2, \ldots, U_l)^T$ whose components are independent standard normal random variables, a transition matrix $R \in \mathbb{R}^{kxl}$, and a mean vector $\vec{\mu} \in \mathbb{R}^k$, such that $\vec{Z} = R\vec{U} + \vec{\mu}$.
2. A random vector $\vec{Z} = (Z_1, Z_2, \ldots, Z_k)^T$ is JG if $\sum_{i=1}^k a_i Z_i$ is normally distributed for every $a = (a_1, a_2, \ldots, a_k)^T \in \mathbb{R}^k$.
3. (Non-degenerate case only) A random vector $\vec{Z} = (Z_1, Z_2, \ldots, Z_k)^T$ is JG if

$$f_{\vec{Z}}(\vec{z}) = \frac{1}{\sqrt{|\det(\Sigma)|}} \frac{1}{(\sqrt{2\pi})^k} e^{-\frac{1}{2}(\vec{Z}-\vec{\mu})^T \Sigma^{-1} (\vec{Z}-\vec{\mu})}$$

   Where $\Sigma = E[(\vec{Z}-\vec{\mu})(\vec{Z}-\vec{\mu})^T] = E[(R\vec{U})(R\vec{U})^T] = RE[\vec{U}\vec{U}^T]R^T = RIR^T = RR^T$
   Σ is also called the **covariance matrix** of $\vec{Z}$.

Note that all of these conditions are equivalent. In this note we will start by showing a proof that $(1) \implies (3)$. We will leave it as an exercise to prove the rest of the implications needed to show that the three conditions are in fact equivalent.

3.1 Proving $(1) \implies (3)$

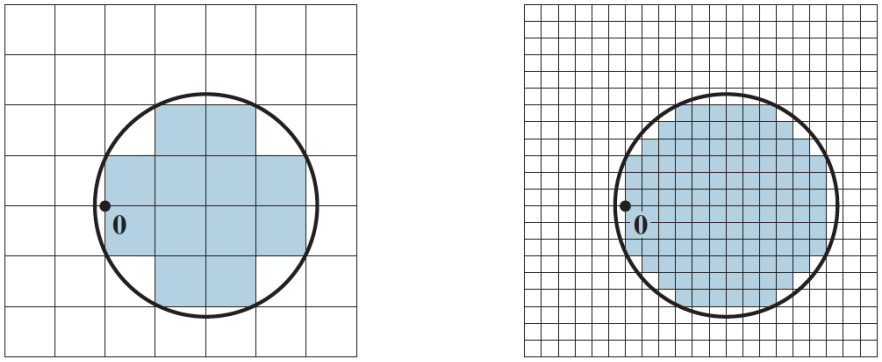
In the context of the noise problem we defined earlier, we are starting with condition (1), ie. $\vec{Z} = R\vec{U}$ (in this case $k = l = n$), and we would like to derive the probability density of $\vec{Z}$. Note that here we removed the $\vec{\mu}$ from consideration because in machine learning we always assume that the noise has a mean of 0. We leave it as an exercise for the reader to prove the case for an arbitrary $\vec{\mu}$.

We will first start by relating the probability density function of $\vec{U}$ to that of $\vec{Z}$. Denote $f_{\vec{U}}(\vec{u})$ as the probability density for $\vec{U} = \vec{u}$, and similarly denote $f_{\vec{Z}}(\vec{z})$ as the probability density for $\vec{Z} = \vec{z}$.

One may initially believe that $f_{\vec{U}}(\vec{u}) = f_{\vec{Z}}(R\vec{u})$, but this is NOT true. Remember that since there is a change of variables from $\vec{U}$ to $\vec{Z}$, we must make sure to incorporate the change of variables constant, which in this case is the absolute value of the determinant of R. Incorporating this constant, we will have the correct formula:

$$f_{\vec{U}}(\vec{u}) = |\det(R)| f_{\vec{Z}}(R\vec{u})$$

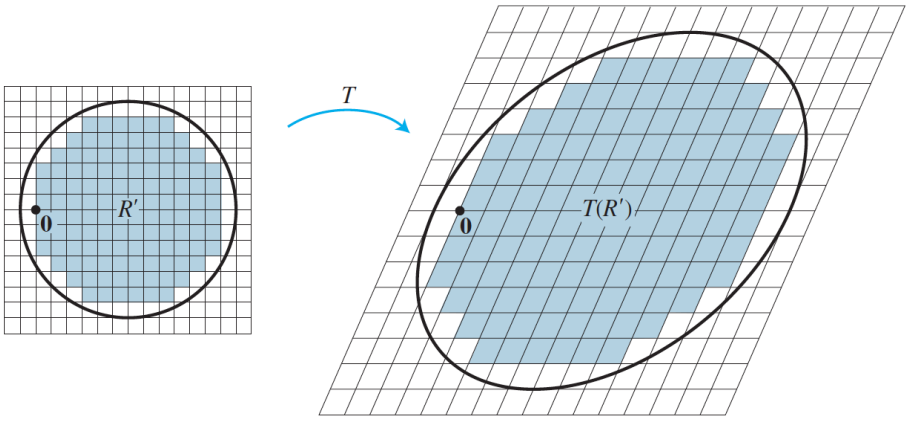
Let's see why this is true, with a simple 2D geometric explanation. Define $\vec{U}$ space to be the 2D space with axes U_1 and U_2. Now take any arbitrary region R' in $\vec{U}$ space (note that this R' is different from the matrix R that relates $\vec{U}$ to $\vec{Z}$). As shown in the diagram below, we have some off-centered circular region R' and we would like to approximate the probability that $\vec{U}$ takes a value in this region. We can do so by taking a Riemann sum of the density function $f_{\vec{U}}(.)$ over smaller and smaller squares that make up the region R':



Mathematically, we have that

$$P(\vec{U} \subseteq R') = \iint_{R'} f_{\vec{U}}(u_1, u_2)\, du_1\, du_2 \approx \sum\sum_{R'} f_{\vec{U}}(u_1, u_2)\, \Delta u_1\, \Delta u_2$$

Now, let's apply the linear transformation $\vec{Z} = R\vec{U}$, mapping the region R' in $\vec{U}$ space, to the region $T(R')$ in $\vec{Z}$ space.



The graph on the right is now $\vec{Z}$ space, the 2D space with axes Z_1 and Z_2. Assuming that the matrix R is invertible, there is a one-to-one correspondence between points in $\vec{U}$ space to points in $\vec{Z}$ space. As we can note in the diagram above, each unit square in $\vec{U}$ space maps to a parallelogram in $\vec{Z}$ space (in higher dimensions, we would use the terms **hypercube** and **parallelepiped**). Recall the

relationship between each unit hypercube and the parallelepiped it maps to:

$$\text{Area(parallelepiped)} = |\det(R)| \cdot \text{Area(hypercube)}$$

In this 2D example, if we denote the area of each unit square as $\Delta u_1 \Delta u_2$, and the area of each unit parallelepiped as ΔA, we say that

$$\Delta A = |\det(R)| \cdot \Delta u_1 \Delta u_2$$

Now let's take a Riemann sum to find the probability that $\vec{Z}$ takes a value in $T(R')$:

$$P(\vec{Z} \subseteq T(R')) = \iint_{T(R')} f_{\vec{Z}}(z_1, z_2)\, dz_1\, dz_2 \approx \sum \sum_{T(R')} f_{\vec{Z}}(\vec{z})\, \Delta A = \sum \sum_{R'} f_{\vec{Z}}(R\vec{u})\, |\det(R)| \Delta u_1 \Delta u_2$$

Note the change of variables in the last step: we sum over the squares in $\vec{U}$ space, instead of parallelograms in $\vec{R}$ space.

So far, we have shown that (for any dimension n)

$$P(\vec{U} \subseteq R') = \int \ldots \iint_{R'} f_{\vec{U}}(\vec{u})\, du_1 du_2 \ldots du_n$$

and

$$P(\vec{Z} \subseteq T(R')) = \int \ldots \iint_{R'} f_{\vec{Z}}(R\vec{u})\, |\det(R)| du_1 du_2 \ldots du_n$$

Notice that these two probabilities are equivalent! The probability that $\vec{U}$ takes value in R' must equal the probability that the transformed random vector $\vec{Z}$ takes a value in the transformed region $T(R')$.

Therefore, we can say that

$$P(\vec{U} \subseteq R') = \int \ldots \iint_{R'} f_{\vec{U}}(\vec{u})\, du_1 du_2 \ldots du_n = \int \ldots \iint_{R'} f_{\vec{Z}}(R\vec{u})\, |\det(R)| du_1 du_2 \ldots du_n = P(\vec{Z} \subseteq T(R'))$$

We conclude that

$$f_{\vec{U}}(\vec{u}) = f_{\vec{Z}}(R\vec{u})\, |\det(R)|$$

An almost identical argument will allow us to state that

$$f_{\vec{Z}}(\vec{z}) = f_{\vec{U}}(R^{-1}\vec{u})\, |\det(R^{-1})| = \frac{1}{|\det(R)|} f_{\vec{U}}(R^{-1}\vec{z})$$

Since the densities for all the U_i's are i.i.d, and $\vec{U} = R^{-1}\vec{Z}$, we can write the joint density function

of Z as

$$\begin{aligned} f_{\vec{Z}}(\vec{z}) &= \frac{1}{|\det(R)|} f_{\vec{U}}(R^{-1}\vec{z}) \\ &= \frac{1}{|\det(R)|} \prod_{i=1}^{n} f_{U_i}((R^{-1}\vec{z})_i) \\ &= \frac{1}{|\det(R)|} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}(R^{-1}\vec{z})^T(R^{-1}\vec{z})} \\ &= \frac{1}{|\det(R)|} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\vec{z}^T R^{-T} R^{-1}\vec{z}} \\ &= \frac{1}{|\det(R)|} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\vec{z}^T (RR^T)^{-1}\vec{z}} \end{aligned}$$

Note that $(RR^T)^{-1}$ is simply the covariance matrix for $\vec{Z}$:

$$Cov[\vec{Z}] = E[\vec{Z}\vec{Z}^T] = E[R\vec{U}\vec{U}^T R^T] = RE[\vec{U}\vec{U}^T]R^T = RIR^T = RR^T$$

Thus the density function of $\vec{Z}$ can be written as

$$f_{\vec{Z}}(\vec{z}) = \frac{1}{|\det(R)|} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\vec{z}^T \Sigma_Z^{-1}\vec{z}}$$

Furthermore, we know that $|\det(\Sigma_Z)| = |\det(RR^T)| = |\det(R)\cdot\det(R^T)| = |\det(R)\cdot\det(R)| = |\det(R)|^2$ and therefore

$$f_{\vec{Z}}(\vec{z}) = \frac{1}{\sqrt{\det(\Sigma_Z)}} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\vec{z}^T \Sigma_Z^{-1}\vec{z}}$$

4 MLE with Dependent Noise

Up to this point, we have been able to easily view regression problems from an optimization perspective. In the context of dependent noise however, it is much easier to view the problem from a probabilistic perspective. Note as before that:

$$\vec{y} = A\vec{w} + \vec{Z}$$

Where $\vec{Z}$ is now a jointly Gaussian random vector. That is, $\vec{Z} \sim \mathcal{N}(\vec{0}, \Sigma_Z)$, and $\vec{y} \sim \mathcal{N}(A\vec{w}, \Sigma_Z)$.

Our goal is to maximize the probability of our data over the set of possible $\vec{w}$'s:

$$\begin{aligned} w^* &= \arg\max_{\vec{w}\in\mathbb{R}^d} \frac{1}{\sqrt{|\det(\Sigma_Z)|}} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}(\vec{y}-A\vec{w})^T \Sigma_Z^{-1}(\vec{y}-A\vec{w})} && (1) \\ &= \arg\min_{\vec{w}\in\mathbb{R}^d} (\vec{y}-A\vec{w})^T \Sigma_Z^{-1}(\vec{y}-A\vec{w}) && (2) \end{aligned}$$

Notice that Σ_Z is symmetric, which means it has a good eigen structure, therefore we can take the advantage interpret this geometrically. Σ_Z can be written as

$$\Sigma_Z = Q \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \sigma_n^2 \end{bmatrix} Q^T$$

where Q is orthonormal. This means a multivariate Gaussian can be thought of having its level sets be the ellipsoid having axis given by Q and amount of stretch given by σs.

Let

$$R_Z = \Sigma_Z^{\frac{1}{2}} = Q \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \sigma_n \end{bmatrix}$$

As before, we can scale the data to morph the problem into an MLE problem with i.i.d noise variables, by premultiplying the data matrix A and the observation vector $\vec{y}$ by R_Z^{-1}.

In a very similar fashion to the independent noise problem, the MLE of the scaled dependent noise problem is $\vec{w}^* = (A^T \Sigma_Z^{-1} A)^{-1} A^T \Sigma_Z^{-1} \vec{y}$.