Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Vinayak Trivedi
Roll Number: 160790
Date: November 2, 2018

QUESTION 1

For a randomly chosen α_n, co-ordinate descent is $\alpha_n = \alpha_n + \delta_*$. $\delta_* = argmax\mathbf{f}(\alpha + \delta\mathbf{e}_n) = argmax_\delta(\alpha + \delta\mathbf{e}_n)^T\check{1} - 0.5 * (\alpha + \delta\mathbf{e}_n)^T\mathbf{G}(\alpha + \delta\mathbf{e}_n)$

Simplifying, we get $\implies$ arg $\max_\delta \alpha^T\check{1} + \delta - 0.5 * \alpha^T\mathbf{G}\alpha - \delta\alpha^T\mathbf{G}\mathbf{e}_n - 0.5 * \delta^2\mathbf{e}_n^T\mathbf{G}\mathbf{e}_n$

Differentiating above equation with respect to scalar δ and setting to zero, we get -
$1 - \alpha^T\mathbf{G}\mathbf{e}_n - \delta\mathbf{e}_n^T\mathbf{G}\mathbf{e}_n = 0$

$\implies \delta_* = \frac{1-\alpha^T\mathbf{G}\mathbf{e}_n}{\mathbf{e}_n^T\mathbf{G}\mathbf{e}_n} = \frac{1}{\mathbf{x}_n^T\mathbf{x}_n}(1 - \alpha^T\mathbf{G}\mathbf{e}_n)$.

Now, we know that each α_n belongs in range [0,C], thus in order to keep this constraint intact, value of final δ_* will be calculated as (**Eqn 1.1**)

$$\delta_*^{final} = \begin{cases} C - \alpha_n \text{ if } \delta_* + \alpha_n > C \\ -\alpha_n \text{ if } \delta_* + \alpha_n < 0 \\ \frac{1}{\mathbf{x}_n^T\mathbf{x}_n}(1 - \alpha^T\mathbf{G}\mathbf{e}_n) \text{ otherwise} \end{cases}$$

The **Co-ordinate ascent Algorithm** is thus as follows -

- Initialise the α vector to be α_0
- Choose a random index i in 1 to N, in order to update α_i.
- Calculate the value of δ_*^{final} from equation 1.1, using the i-th training example and old value of α vector.
- Update the value of $\alpha_i = \alpha_i + \delta_*^{final}$
- If the value of f(α) starts decreasing at updated α vector or α vector stops changing, then break. Else, go to step 2 Again.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Vinayak Trivedi
Roll Number: 160790
Date: November 2, 2018

QUESTION 2

Since the total sum of squared distances between all pairs of input points is a constant (C_2^n pairs), thus minimizing the sum of squared distances between pairs of points within the same cluster, is equivalent to maximizing the sum of squared distances between pairs of points which are not in the same cluster. This is primarily because total sum is constant and a pair of two points can be either in the same cluster or in the different clusters, there is nothing in between these two options. Hence pairs of points in the same cluster and pairs of points in different clusters actually cover the entire pairs of points available, whose sum is constant. Thus minimizing one will surely maximize the other.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Vinayak Trivedi
Roll Number: 160790
Date: November 2, 2018

QUESTION 3

In this problem setting, we can say that each x_n^{obs} is an input which is related to a latent variable $z_n = x_n^{miss}$. We will calculate the probability distribution to estimate value of x_n^{miss} in the E-step of EM algorithm and using this distribution of x_n^{miss}, we will maximize the CLL in M-step to update the parameters.

Now, we know that each x_n is gaussian with mean μ and covariance Σ, and that each x_n is $\begin{bmatrix} x_n^{obs} \\ x_n^{miss} \end{bmatrix}$, thus $P(x_n^{miss}|x_n^{obs})$ will also be Gaussian whose mean and covariance matrix can be determined by current values of μ and Σ and known indices where the data is missing in x_n. (From Section 4.3.1 of MLAPP). Thus, we can say that $P(x_n^{miss}|x_n^{obs}) = \mathcal{N}(x_n^{miss}|\mu_{miss}^{cond}, \Sigma_{miss}^{cond})$. In particular, $\mu_{miss}^{cond} = \mu_{miss} + \Sigma_{miss,obs}\Sigma_{obs,obs}^{-1}(\mathbf{x}_n^{obs} - \mu_n^{obs})$, and
$\Sigma_{miss}^{cond} = \Sigma_{miss,miss} - \Sigma_{miss,obs}\Sigma_{obs,obs}^{-1}\Sigma_{miss,obs}$. This will conclude our M step of EM.

Now, we need to maximize $\mathrm{E}(\sum_{n=1}^{N} \log \mathrm{P}(x_n^{obs}, x_n^{miss}))$ which is equal to $\mathrm{E}(\sum_{n=1}^{N} \log \mathrm{P}(\mathbf{x}_n))$. Now, since we know x_n follows gaussian with mean μ and Covariance Σ, we can write the probability $\mathrm{P}(\mathbf{x}_n)$ easily and taking log, we get -

$\arg\max_\theta -\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^T\Sigma^{-1}(x_n-\mu) - \frac{ND}{2}log2\pi - \frac{N}{2}log|\Sigma|$. Transforming above equation using formula listed in Maths refresher slides, we get the CLL as -

$$\boxed{CLL = \frac{-1}{2}trace(\Sigma^{-1}\sum_{n=1}^{N}(x_n-\mu)(x_n-\mu)^T) - \frac{ND}{2}log2\pi - \frac{N}{2}log|\Sigma|}$$

Now, $\mathrm{E}(x_n) = \begin{bmatrix} x_n^{obs} \\ E(x_n^{miss}) \end{bmatrix}$. $E(x_n^{miss}) = \mu_{miss}^{cond}$ as derived above from probability distribution of x_n^{miss}. Thus, $\mathrm{E}(x_n) = \begin{bmatrix} x_n^{obs} \\ \mu_{miss}^{cond} \end{bmatrix}$.

Now $\mathrm{E}(x_n x_n^T) = \mathrm{E}\begin{bmatrix} x_n^{obs}x_n^{obs^T} & x_n^{obs}x_n^{miss^T} \\ x_n^{miss}x_n^{obs^T} & x_n^{miss}x_n^{miss^T} \end{bmatrix} = \begin{bmatrix} x_n^{obs}x_n^{obs^T} & x_n^{obs}\mathbf{E}(x_n^{miss})^T \\ \mathbf{E}(x_n^{miss})x_n^{obs^T} & \mathbf{E}(x_n^{miss}x_n^{miss^T}) \end{bmatrix}$

Now, $\mathbf{E}(x_n^{miss}) = \mu_{miss}^{cond}$, $\mathbf{E}(x_n^{miss})^T = (\mu_{miss}^{cond})^T$ and $\mathbf{E}(x_n^{miss}x_n^{miss^T}) = \mathbf{E}(x_{miss})\mathbf{E}(x_{miss})^T + \Sigma_{miss}^{cond}$ which is equal to $\mu_{miss}^{cond}(\mu_{miss}^{cond})^T + \Sigma_{miss}^{cond}$.

thus, now we have calculated all unknown parameters in $\mathrm{E}(x_n x_n^T)$, let us write $\mathrm{E}[x_n]$ and $\mathrm{E}[x_n x_n^T]$ together.

$$\boxed{E(x_n) = \begin{bmatrix} x_n^{obs} \\ \mu_{miss}^{cond} \end{bmatrix}. \text{ and } E(x_n x_n^T) = \begin{bmatrix} x_n^{obs}x_n^{obs^T} & x_n^{obs}(\mu_{miss}^{cond})^T \\ \mu_{miss}^{cond}x_n^{obs^T} & \mu_{miss}^{cond}(\mu_{miss}^{cond})^T + \Sigma_{miss}^{cond} \end{bmatrix} \textbf{ Eqn 2.1}}$$

Now, we can state expected CLL as -

$$\boxed{ECLL = \frac{-1}{2}trace(\Sigma^{-1}\sum_{n=1}^{N}\mathbf{E}[(x_n-\mu)(x_n-\mu)^T)] - \frac{ND}{2}log2\pi - \frac{N}{2}log|\Sigma|}$$

Now, taking partial derivative of above ECLL with respect to μ and Σ and setting them to zero, we get (**Eqn 2.2**) -

$$\boxed{\mu_t = \frac{1}{N}\sum_{n=1}^{N}\mathbf{E}[x_n] \text{ and } \Sigma_t = \frac{1}{N}\sum_{n=1}^{N}\mathbf{E}[x_n x_n^T] - \mu_t \mu_t^T}.$$

Thus the overall **Expectation Maximisation** algorithm can be stated as -

- Initialise the μ and Σ to be μ_0 and Σ_0
- **E step** - Calculate the posterior distribution of x_n^{miss} given x_n^{obs} using the current values of μ and Σ and known indices where data is missing, using the procedure explained in first half of this answer. (It will be Gaussian whose parameters can be estimated by using current μ and Σ).
- Using the above calculated distribution of x_n^{miss}, calculate $\mathrm{E}[x_{miss}^T]$ and $\mathrm{E}[x_n^{miss} x_n^{miss^T}]$ as shown in above derivations, and using these values, calculate $\mathrm{E}[x_n]$ and $\mathrm{E}[x_n x_n^T]$ for all data points using the equation 2.1
- **M step** - Update the values of current parameters, μ and Σ using the equation 2.2.
- Use these new values of parameters and go back to step 2, where you again estimate the new distribution of x_n^{miss} (basically its new mean and covariance) and continue so on.
- When the parameters μ and Σ stop changing much, stop the procedure. (Converged)

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Vinayak Trivedi
Roll Number: 160790
Date: November 2, 2018

QUESTION 4

Let each x_n is related to a latent variable y_n where values of y_n for $n \in (1, 2, ..N)$ are exactly known and remaining to be estimated. Let us define y_{nk} to be 1 when $y_n = k$. In the E-step, we compute the Expectation of y_{nk} and use it to maximize the ECLL in E-step.
Now, the CLL = log(P(X,Y)). The ECLL can be written exactly like a GMM(Gaussian Mixture Model) (directly taken from slides).

$$\boxed{ECLL = \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbf{E}[y_{nk}][log(\pi_k) + log(\mathcal{N}(x_n|\mu_k, \Sigma_k))]}$$

Differentiating above ECLL to update the model parameters, the new values for parameters can be found exactly like a GMM (note that this problem would have been exactly GMM if all the y_n were unknown, since modelling the data as combination of K gaussians and doing Generative Classification with unknown labels is exactly same). Thus if we denote $\mathbf{N}_k = \sum_{n=1}^{N} \mathbf{E}[y_{nk}^t]$ then at t-th iteration, parameters can be found by equations (directly taken from slides)-

$$\boxed{\pi_k^t = \frac{\mathbf{N}_k}{N} \ , \ \mu_k^t = \frac{1}{\mathbf{N}_k}\sum_{n=1}^{N} \mathbf{E}[y_{nk}]x_n \text{ and } \Sigma_k^t = \frac{1}{\mathbf{N}_k}\sum_{n=1}^{N} \mathbf{E}[y_{nk}](x_n - \mu_k^t)(x_n - \mu_k^t)^T \ \textbf{Eqn 4.1}}.$$

In the E-step, when we calculate the expected value of y_{nk} then we don't touch the values of y_{nk} where $n \in (1, 2...N)$, and for other remaining indices, we update the $\mathbf{E}[y_{nk}]$ exactly like a GMM as follows (directly taken from slides) -

$$\boxed{\mathbf{E}[y_{nk}^t] = \frac{\pi_k^{t-1}\mathcal{N}(x_n|\mu_k^{t-1}, \Sigma_k^{t-1})}{\sum_{l=1}^{K} \pi_l^{t-1}\mathcal{N}(x_n|\mu_l^{t-1}, \Sigma_l^{t-1})} \text{ if n } \in (N+1, N+2..., N+M) \ \textbf{Eqn 4.2}}.$$

$$\boxed{\mathbf{E}[y_{nk}^t] = y_{nk}^{t-1} = y_{nk}^t \text{ if n } \in (1, 2..., N) \ \ \textbf{Eqn 4.3}}$$

Using above equations, we can formualte the **EM Algo** for this problem as follows -

- For $n \in (1, 2...N)$, initialize for every n, $y_{nk} = 1$ if $y_n = k$, and zero to all other indices of k.
- Initialize the model parameters μ_k and Σ_k and π_k.
- **E-step** - Calculate expectation of y_{nk} according to the equations 4.2 and 4.3, depending upon the value of n.
- **M step** - Using expected values of y_{nk} and equation 4.1, calculate the updated model parameters μ_k and Σ_k and π_k.
- If not converged, go to step 3, and calculate the new Expected value of y_{nk} using updated model parameters and continue like this until converge.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Vinayak Trivedi
Roll Number: 160790
Date: November 2, 2018

QUESTION 5

(a) The model proposed in problem basically first tries to find which class a particular data point belongs to, and then predicts the output according to weight vector of that class. This is useful when the data is having different linear dependence with input in different regions, for e.g. in 2-D, data is following y=2x+3 in one region, y=5x+1 in other and so on. That is data is combination of different linear models. Standard Probabilistic Linear Regression can't be used because it models same weight vector for all points, but here data has different weight vectors according to clusters(classes).

(b) The given problem is similar to GMM. When we estimate Z, z_n for each y_n will be $z_n = \arg\max_{k\in(1,2..K)} log(P(y_n, z_n = k)) = \arg\max_{k\in(1,2..K)} log(P(z_n = k)) + log(P(y_n|z_n = k))$. Thus,on simplification it gives,

$$z_n = \arg\max_{k\in(1,2..K)} log(\pi_k) - \frac{\beta}{2}(y_n - \mathbf{w}_k^T x_n)^2 \textbf{ Eqn 5.1}$$

Next, when we fix z_n for each y_n in first step, we need to maximize log(P(Y,Z)) and find updated global parameters. Now, $\log(P(Y,Z)) = \sum_{n=1}^{N} log(\Pi_{k=1}^{K} P(y_n, z_n = k)^{z_{nk}})$, where z_{nk} is a binary value, which is 1 if and only if $z_n = k$ and zero in all other cases. Thus, we can write, in second step of ALT-OPT, our objective to maximize is -

$$\Theta_{opt} = \arg\max_{\Theta} \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[log(\pi_k) - \frac{\beta}{2}(y_n - \mathbf{w}_k^T x_n)^2] \textbf{ Eqn 5.2}.$$

Now, maximizing the above objective with respect to each π_k, we get $\pi_k = \frac{\sum_{n=1}^{N} z_{nk}}{N} = \frac{N_k}{N}$ where N_k is number of points which have $z_n = k$. Similarly maximising with respect to each $\mathbf{w}_k$, we get
$\sum_{n=1}^{N} z_{nk}(y_n - w_k^T x_n)x_n = 0$

$\implies \mathbf{w}_k = (\sum_{n=1}^{N} z_{nk} y_n x_n)(\sum_{n=1}^{N} z_{nk} x_n x_n^T)^{-1}$. Thus, we get the following update equations for global parameters in second step of ALT-OPT,

$$\pi_k = \frac{\sum_{n=1}^{N} z_{nk}}{N} \text{ and } \mathbf{w}_k = (\sum_{n=1}^{N} z_{nk} x_n x_n^T)^{-1}(\sum_{n=1}^{N} z_{nk} y_n x_n) \textbf{ Eqn 5.3}.$$

The **ALT-OPT Algo** is

- Initialise each π_k and $\mathbf{w}_k$.
- **Update Z** - For each data point, calculate the value of z_n from Eqn 5.1.
- **Update global params**- For calculated $z_n, z_{nk} = 1$ only if $z_n = k$, otherwise 0. Then update the values of global parameters, π_k and $\mathbf{w}_k$, according to **Eqn 5.3.**
- Again go to step 2 with updated values of parameters and calculate new z_n for every point, and continue so on till not converged.

As can be seen by equation 5.1, when value of π_k is same for all k and equal to 1/K, then equation simply reduces to $z_n = \arg\min_{k\in(1,2..K)}(y_n - \mathbf{w}_k^T x_n)^2$. The update to each z_n done in this way simply means that each point will be assigned the cluster for which its value is closest to $w^T x_n$. This basically assigning a data point to a cluster, where the current weight vector best models its data value.

Part(c)- In the EM algortihm, we will not assign binary values to z_{nk}, but instead a probability distribution of $P(z_{nk}|y_n)$ in the E step, and then maximise the Expected CLL in M step. Clearly the CLL remains the same as we computed above, hence one can say that the Expected CLL is -

$$\boxed{ECLL = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[log(\pi_k) - \frac{\beta}{2}(y_n - \mathbf{w}_k^T x_n)^2]}.$$

Again, maximizing this ECLL with respect to π_k and $\mathbf{w}_k$ as done above gives-

$$\boxed{\pi_k = \frac{\sum_{n=1}^{N}\mathbf{E}[z_{nk}]}{N} \text{ and } \mathbf{w}_k = (\sum_{n=1}^{N}\mathbf{E}[z_{nk}]x_n x_n^T)^{-1}(\sum_{n=1}^{N}\mathbf{E}[z_{nk}]y_n x_n) \textbf{ Eqn 5.4}}.$$

Now, calculating $\mathbf{E}[z_{nk}]$ in the E-step, $\mathbf{E}[z_{nk}] = 0 * P(z_{nk} = 0|y_n) + 1 * P(z_{nk} = 1|y_n)$. Simplifying above expression, and normalising $\mathbf{E}[z_{nk}]$, we will obtain the following expression for it (similar to GMM as done in class)-

$$\boxed{\mathbf{E}[z_{nk}] = \frac{\pi_k exp\left(\frac{-\beta}{2}(y_n - \mathbf{w}_k^T x_n)^2\right)}{\sum_{l=1}^{K}\pi_l exp\left(\frac{-\beta}{2}(y_n - \mathbf{w}_l^T x_n)^2\right)} \textbf{ Eqn 5.5}}.$$

The **EM-Algo** is thus-

- Initialise each π_k and $\mathbf{w}_k$.
- **E-step** Find the Expectation of z_{nk} according to the equation 5.5.
- **M-step** Using the expected value of z_{nk}, update the global parameters π_k and $\mathbf{w}_k$ according to equation 5.4.
- Go to step 2 again with updated values of parameters and continue so on, if not converged.

Now, the equation 5.5 can be re-written as

$$\mathbf{E}[z_{nk}] = \frac{1}{1 + \sum_{l\neq k}\frac{\pi_l}{\pi_k} exp\left(\frac{-\beta}{2}[(y_n - \mathbf{w}_l^T x_n)^2 - (y_n - \mathbf{w}_k^T x_n)^2]\right)} \tag{1}$$

Let for index m, the value of $(y_n - \mathbf{w}_m^T x_n)^2$ is greatest among all others. Then, we can easily see that in above equation, when k is not equal to m, then $\frac{-\beta}{2}[(y_n - \mathbf{w}_m^T x_n)^2 - (y_n - \mathbf{w}_k^T x_n)^2]$ will tend to positive infinity when β tends to infinity. This will make the value of $\mathbf{E}[z_{nk}]$ tend to zero. Thus when β tends to infinity then expected value of z_{nk} becomes zero for all k except for one which has the value of $(y_n - \mathbf{w}_m^T x_n)^2$ maximum. Thus, the Algo tends to ALT-OPT where we deterministically choose only one value of z_n for which a function has maximum value, and discard all other values for z_n. This is exactly what our EM will do when β tends infinity, and hence becomes ALT-OPT eventually.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Vinayak Trivedi
Roll Number: 160790
Date: November 2, 2018

QUESTION
6

Programming Problem 1

part a) The plots for each value of hyperparameter are attached below. From the plots, it is observed that the accuracy of model on test data keeps on decreasing as we increase the value of hyperparameter. The RMSE error for each case is -

- For 0.1, RMSE = 0.0325776702936
- For 1, RMSE = 0.170303903442
- For 10, RMSE = 0.609267159654
- For 100, RMSE = 0.911085805277

part b) The plots for each value of landmarks are attached below. For different values of landmark,we see that accuracy on test-data keeps on increasing as we increase the number of landmark points, primarily due to the fact that data is being projected to higher and higher dimensions.RMSE error for each landmark is -

- For L=2, RMSE = 0.973379001578
- For L=5, RMSE = 0.838323134386
- For L=10, RMSE = 0.406972042266
- For L=20, RMSE = 0.139462732102
- For L=50, RMSE =0.0764018436474
- For L=100, RMSE = 0.0696165350274

Best value of L among those tested seems to be 100, because of least error on test-data.

Programming part 2

Part a) Plot is attached. The transformation used was (x,y) to (sqrt($x^2 + y^2$),0).

Part b) Plots for 10 different trials are attached below. We can see that when the chosen landmark point is close to the centre, then we get good clustering, and if it is towards outside, we get a bad clustering. This is because we are projecting the data in 1 dimension, based on kernel value with chosen landmark point. The kernel chosen is RBF, which basically gives large values if distance between two points is less and small if distance is large. Thus, data is actually being projected in 1-D according to their distances from the chosen landmark point. If point chosen is close to center, then points of cluster 1 will have very less distance and hence large 1-D value, and opposite for points of cluster 2. Thus, they become linearly separable and hence clustered perfectly by K-means. Contrary to it, if we choose the landmark point far away, then some points of cluster 1 are close to it, some are far. Similarly some points of cluster 2 become close to it and some become far. Thus when we transform them in 1-D, their distances don't follow a pattern, and are kind-of intermixed or randomized. Thus, the points are not linearly separable in 1-D and hence are not clustered by K-means in a nice manner.

Figure 1: Kernel Ridge regression plot with hyper-parameter 0.1

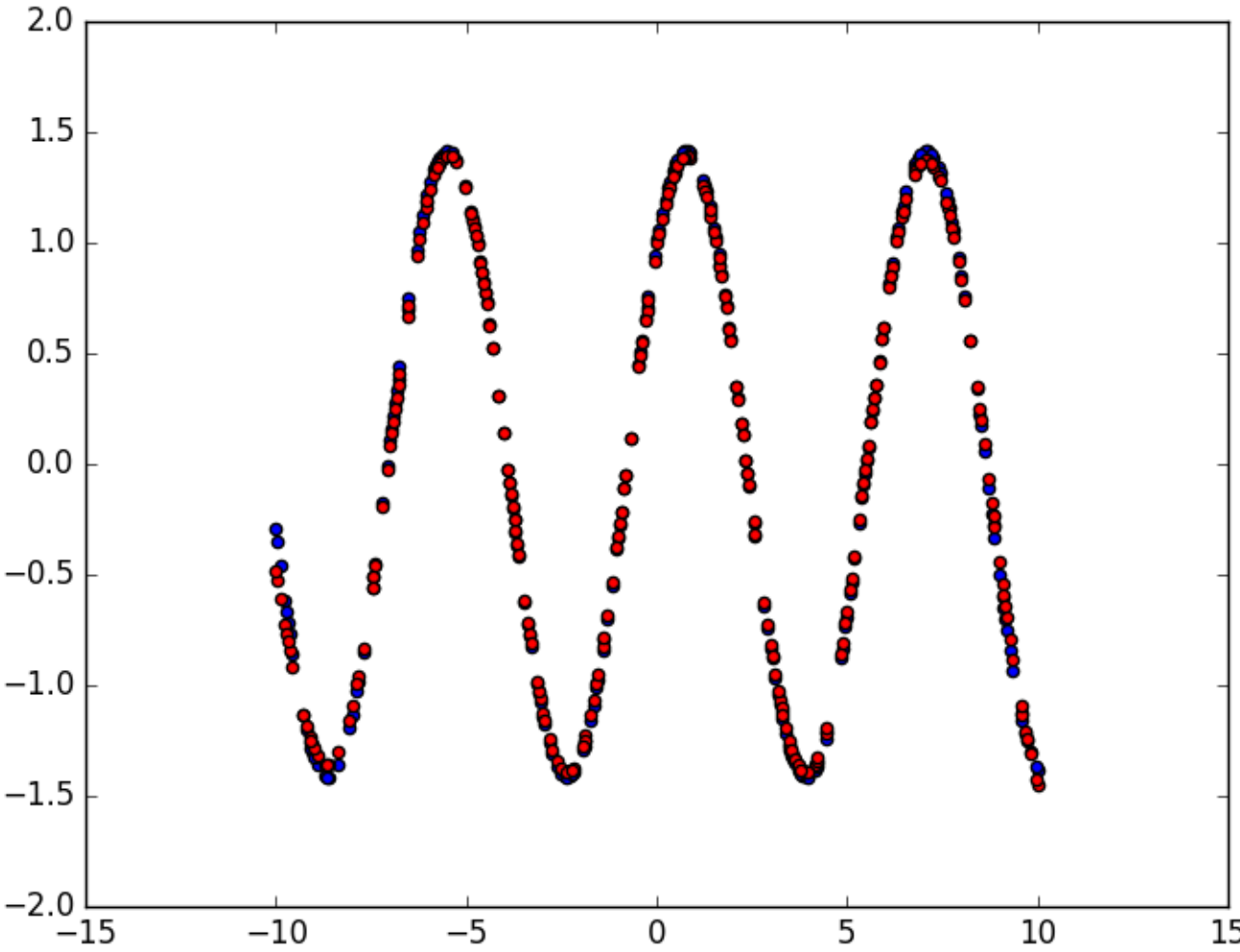

Figure 2: Kernel Ridge regression plot with hyper-parameter 1

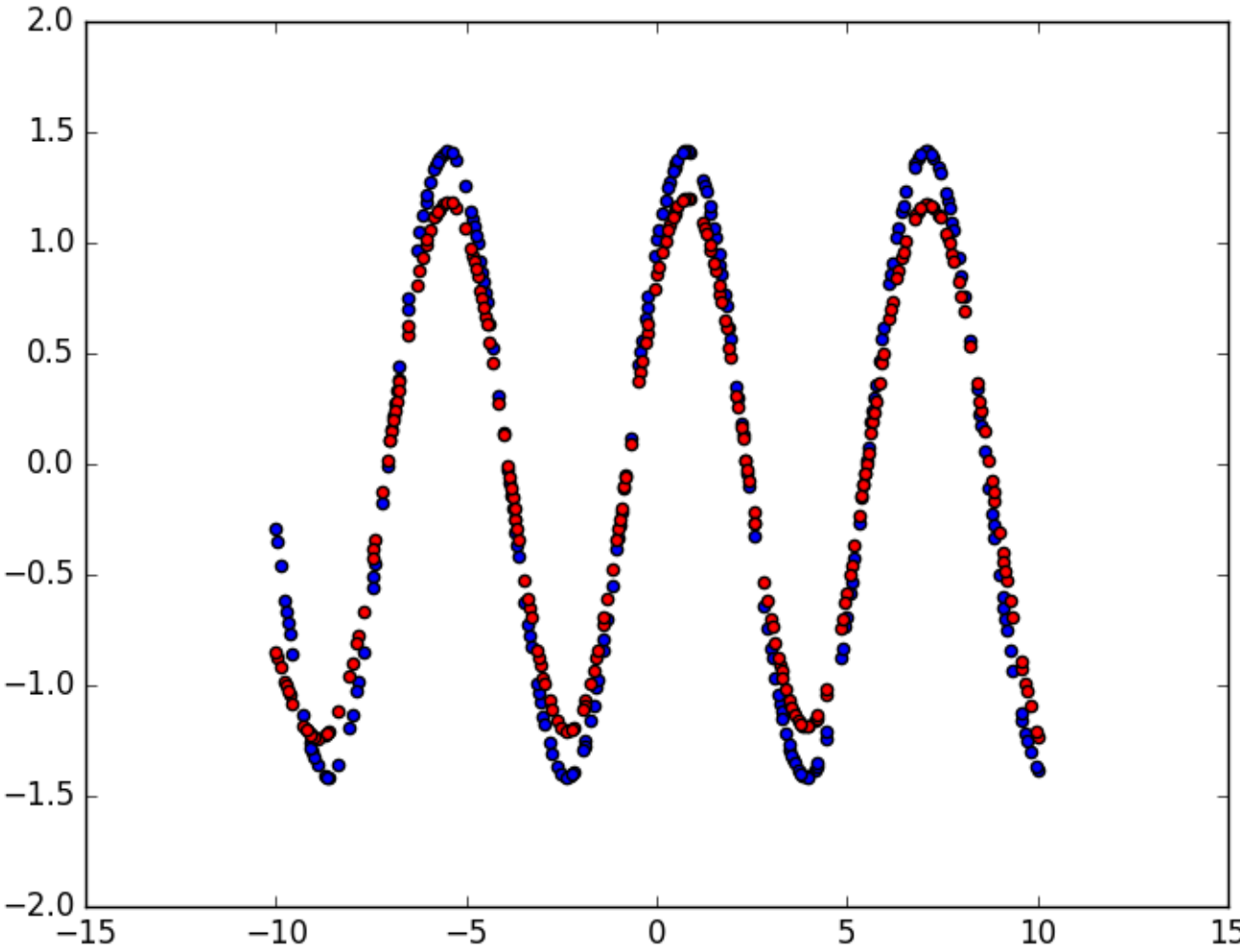

Figure 3: Kernel Ridge regression plot with hyper-parameter 10

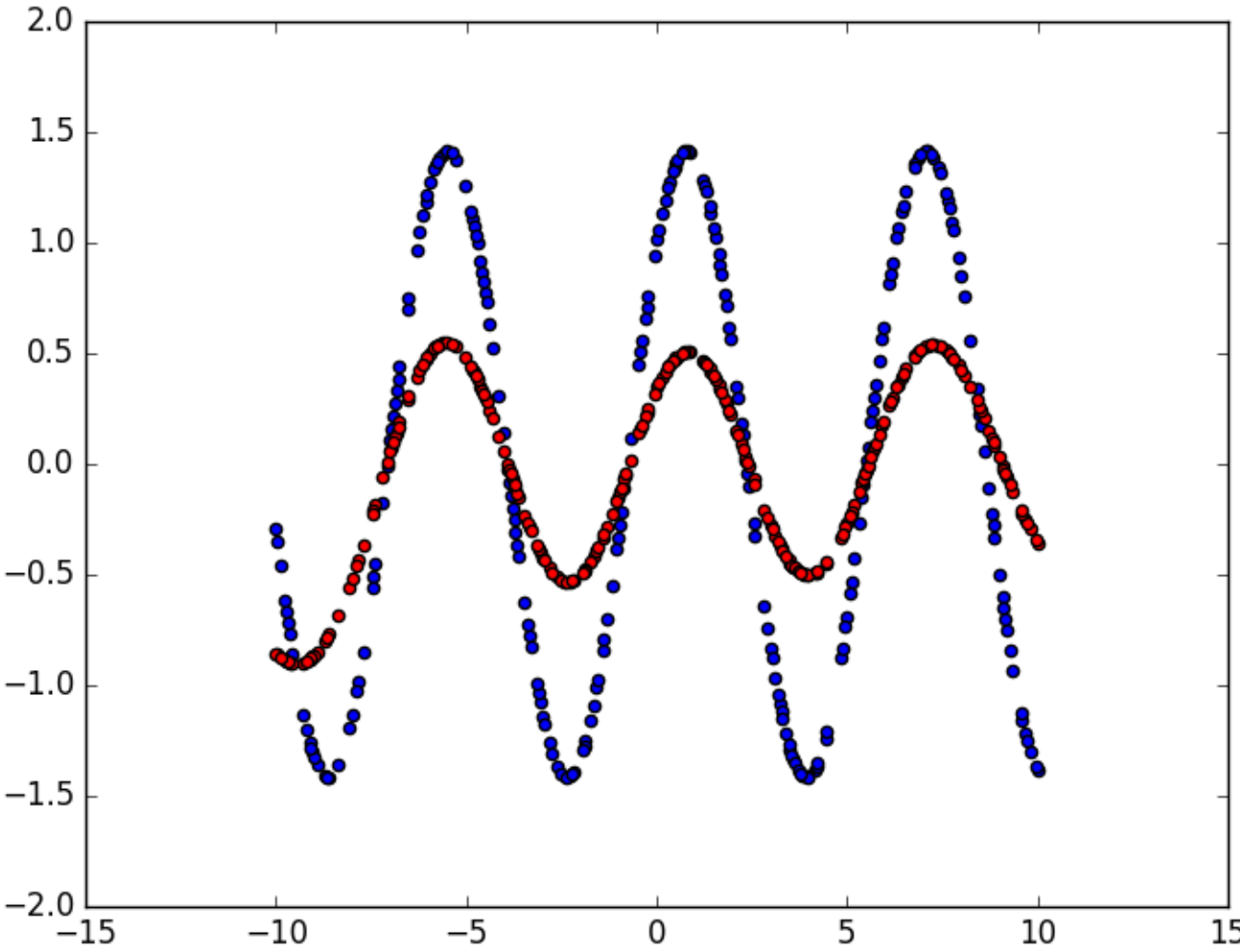

Figure 4: Kernel Ridge regression plot with hyper-parameter 100

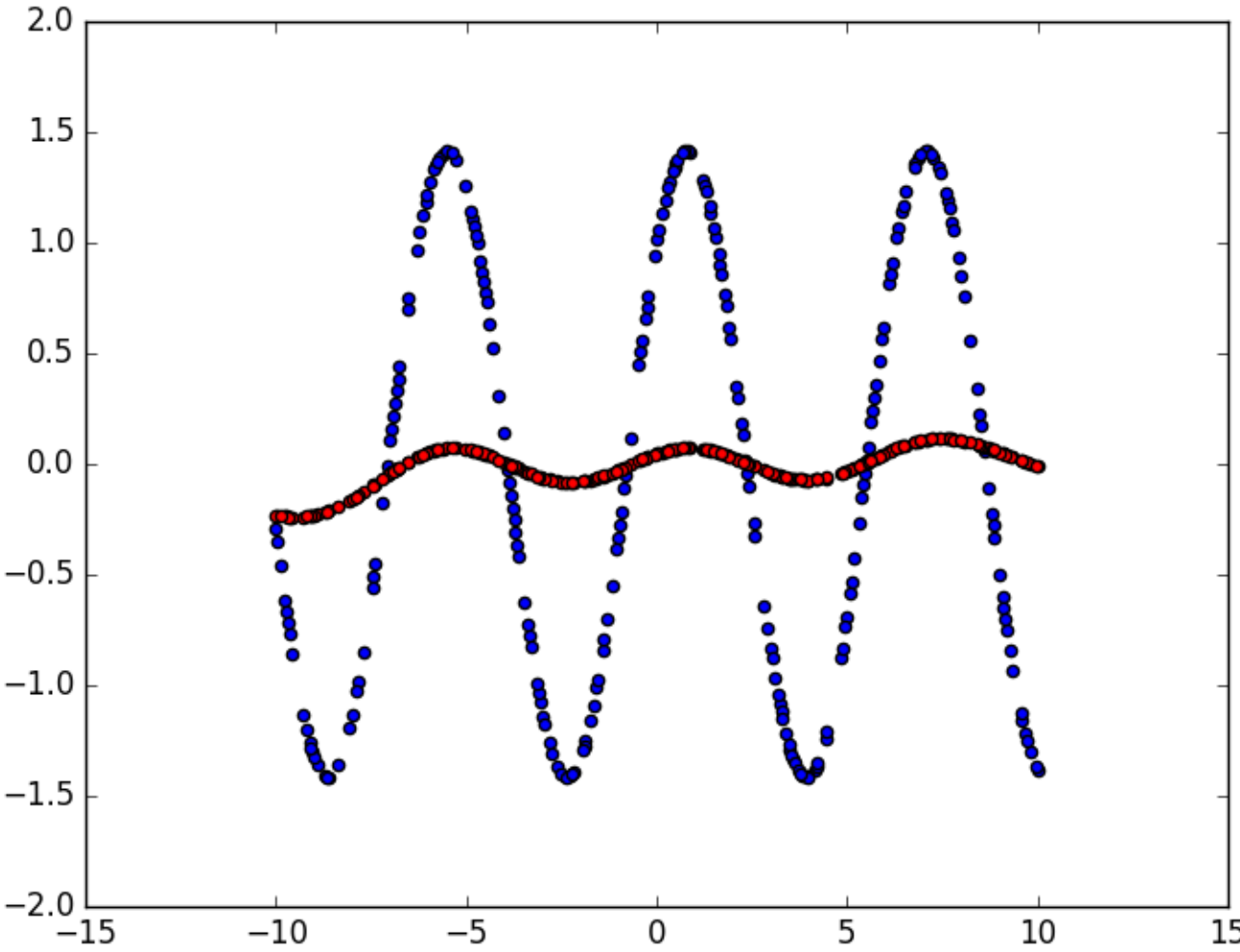

Figure 5: Landmark Ridge regression plot with 2 Landmark points

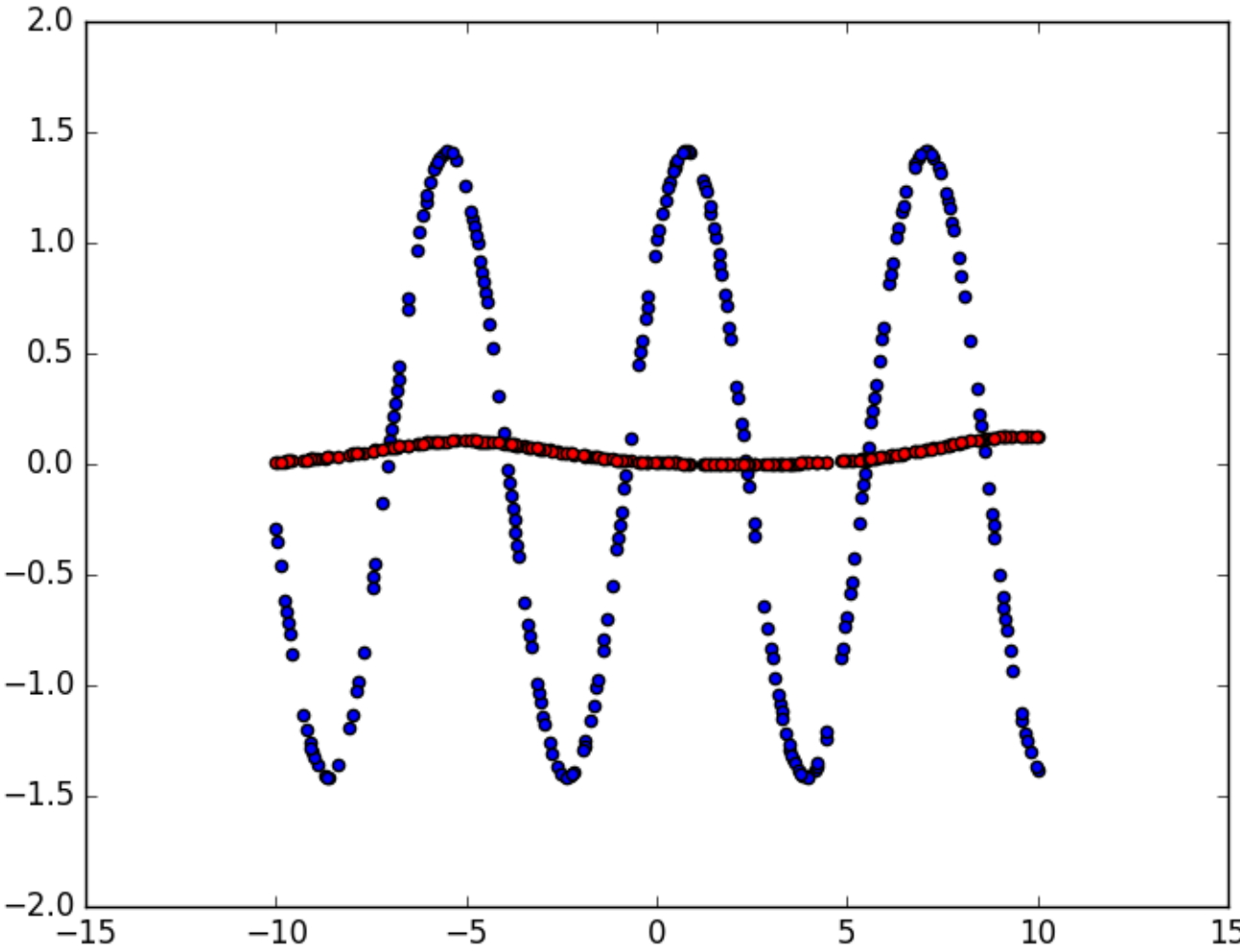

Figure 6: Landmark Ridge regression plot with 5 Landmark points

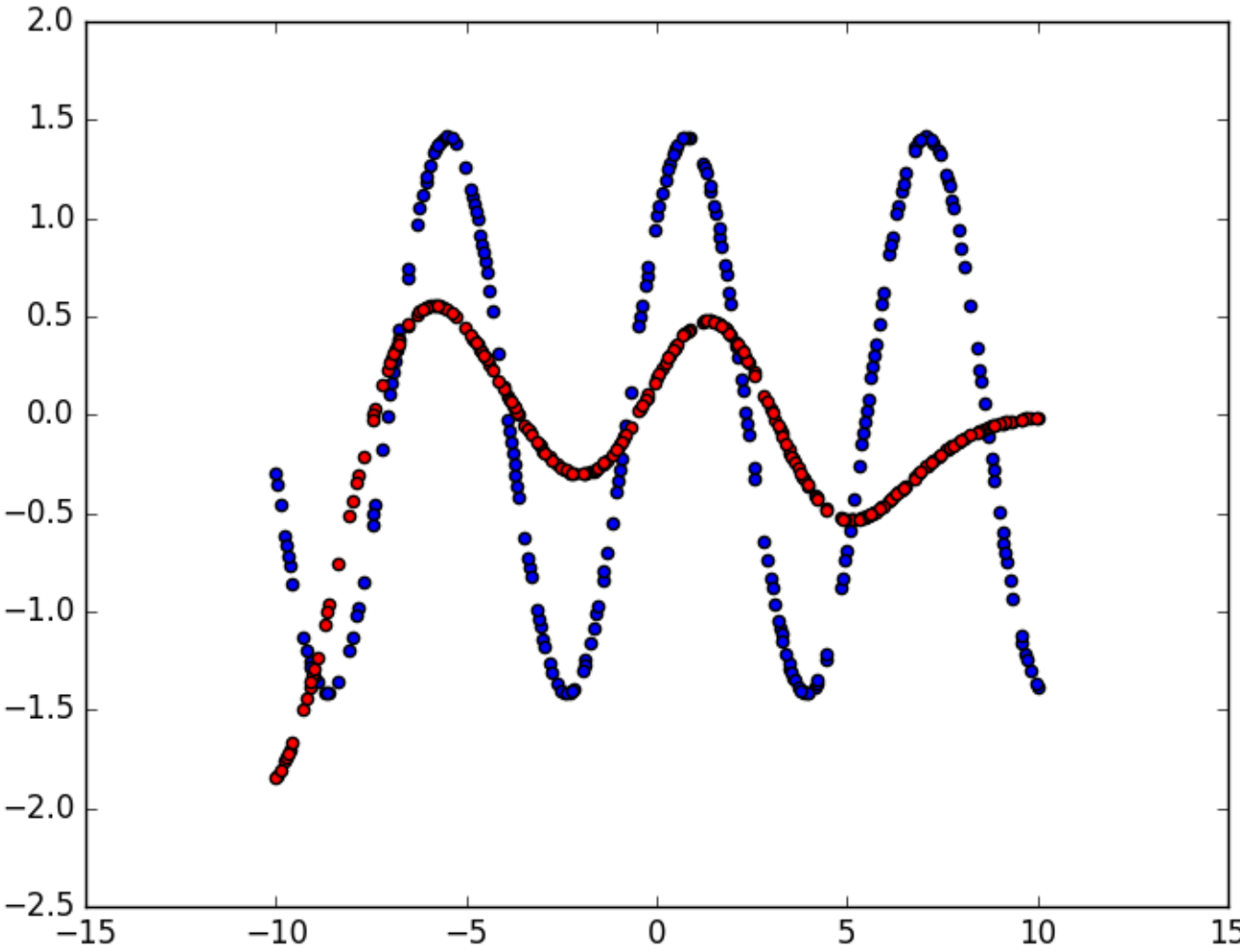

Figure 7: Landmark Ridge regression plot with 10 Landmark points

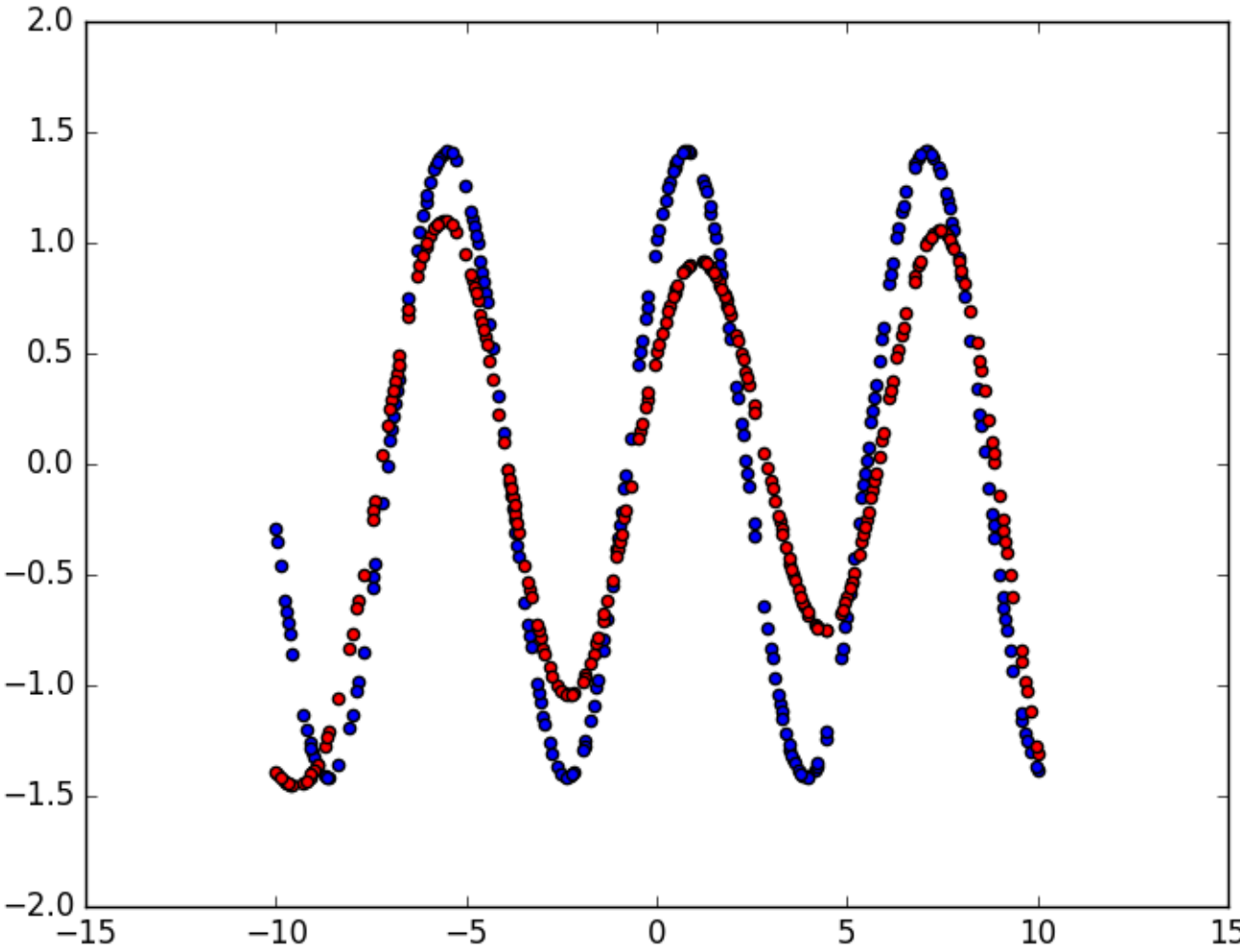

Figure 8: Landmark Ridge regression plot with 20 Landmark points

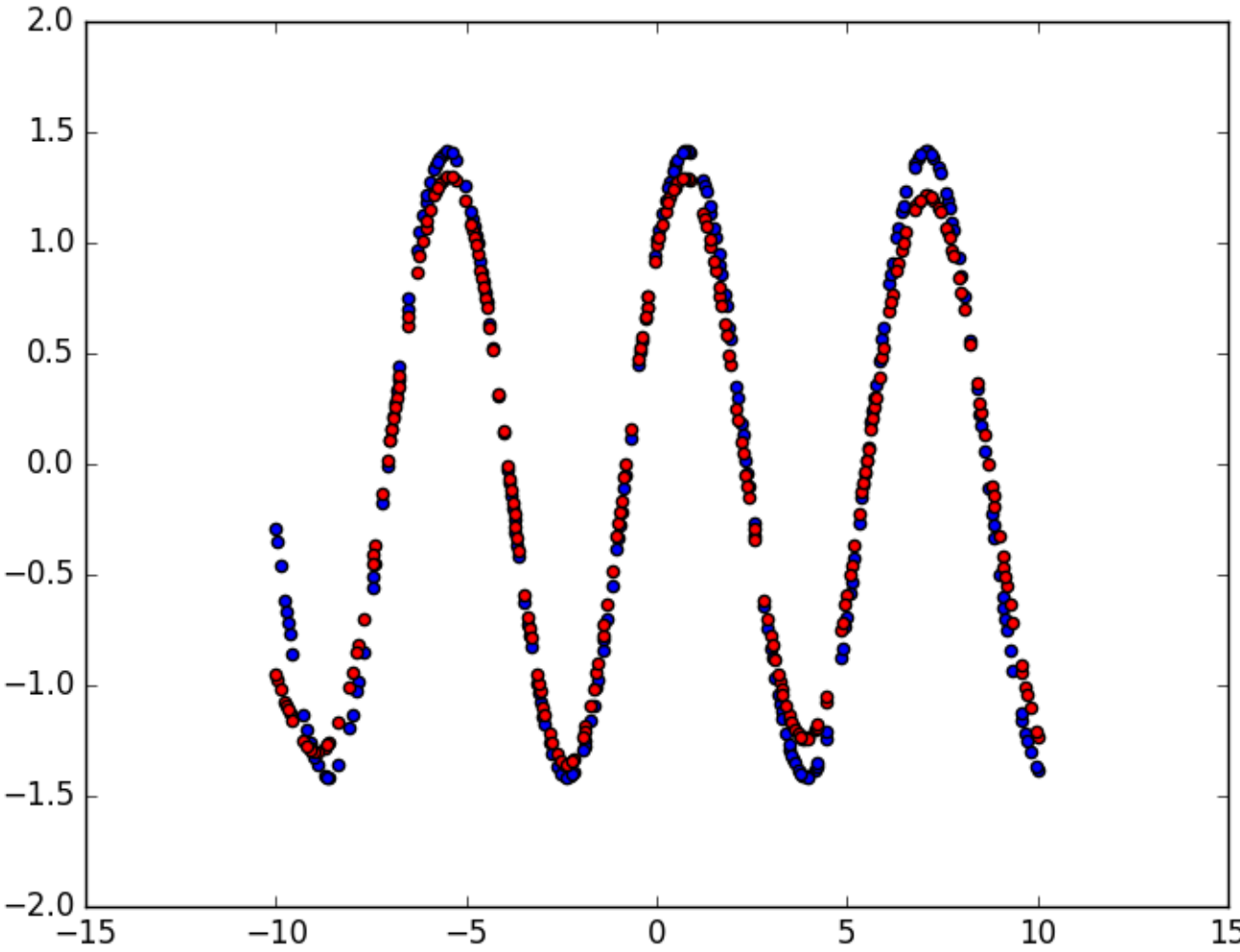

Figure 9: Landmark Ridge regression plot with 50 Landmark points

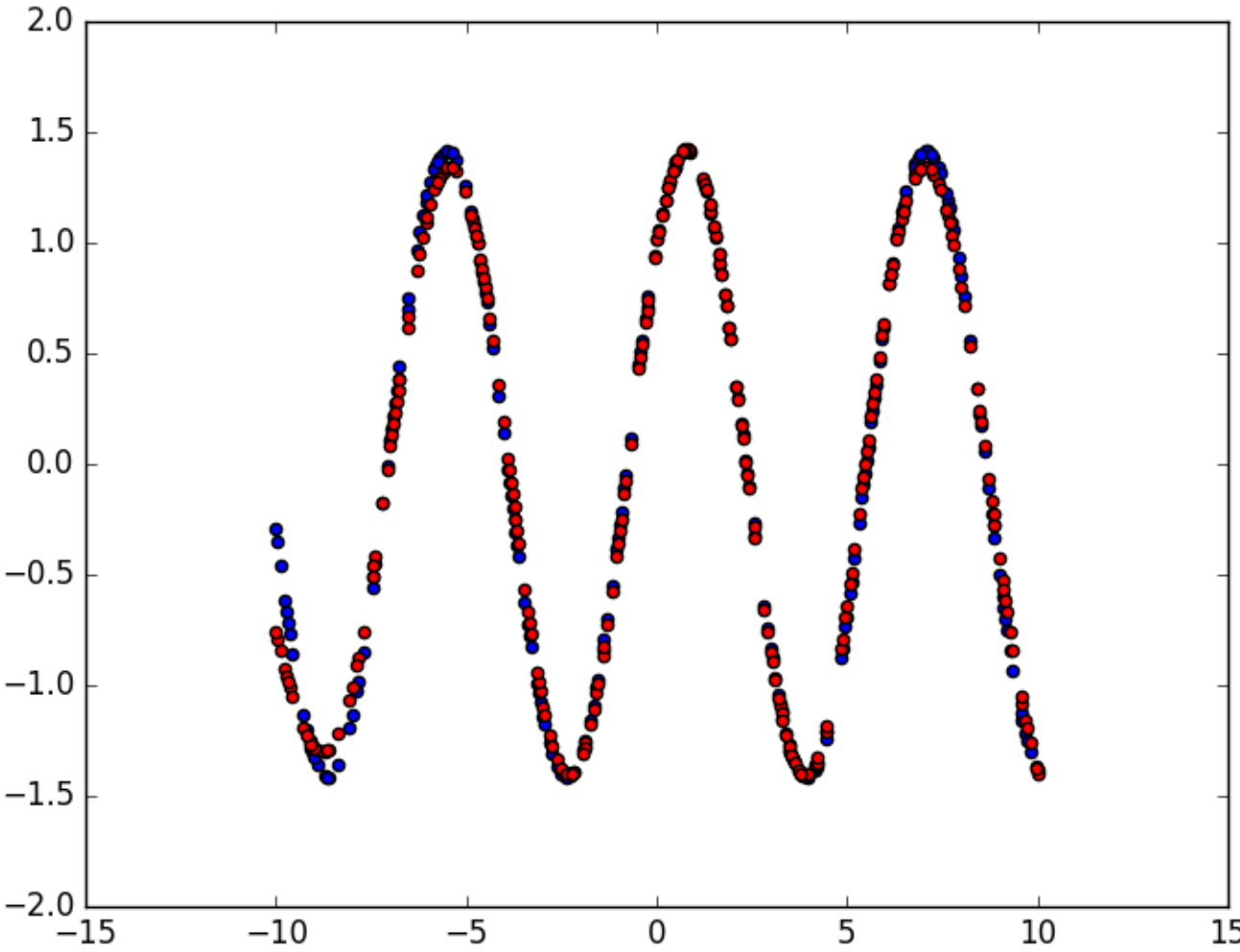

Figure 10: Landmark Ridge regression plot with 100 Landmark points

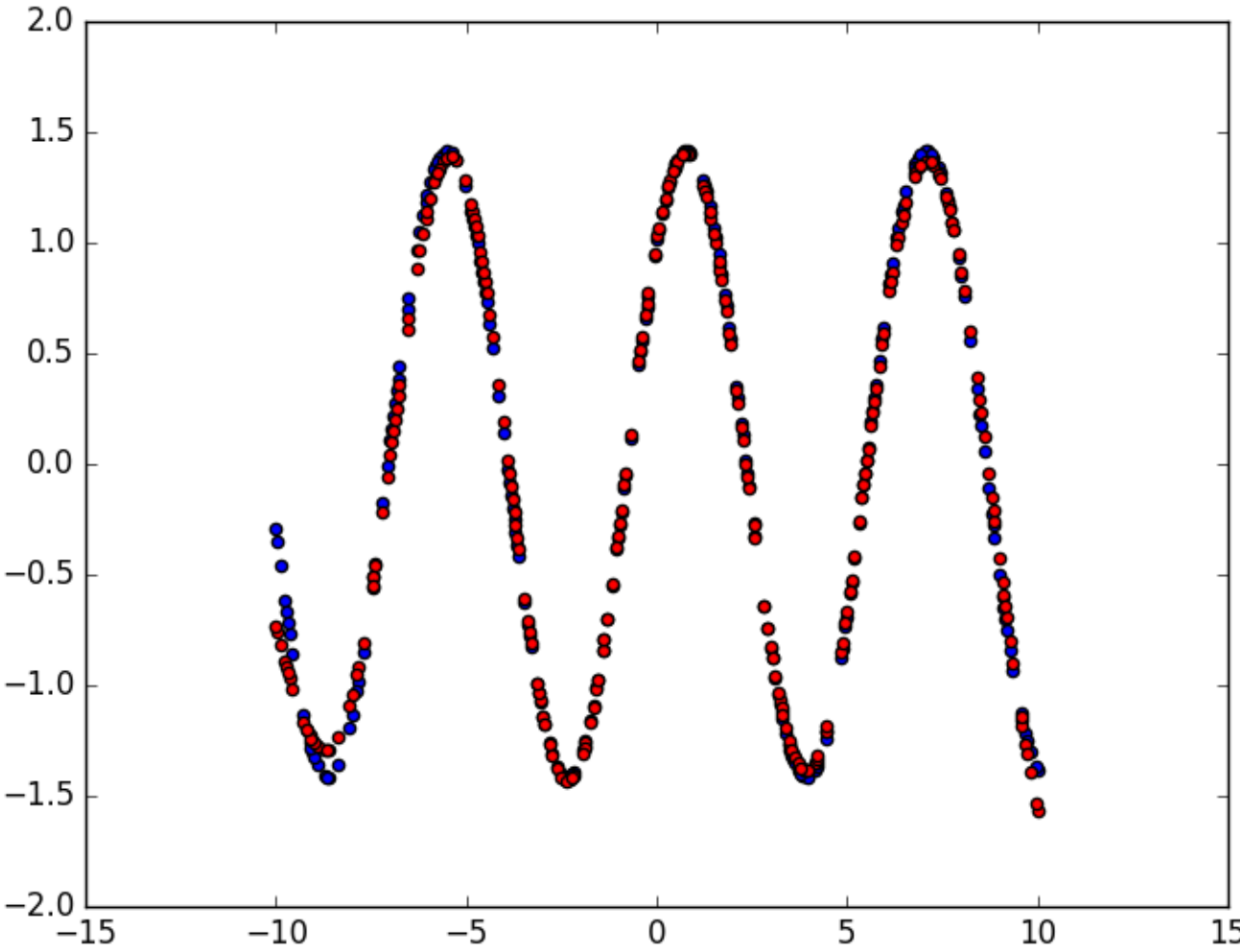

Figure 11: Clustering Plot showing two clusters with different colors with feature transformation described above.

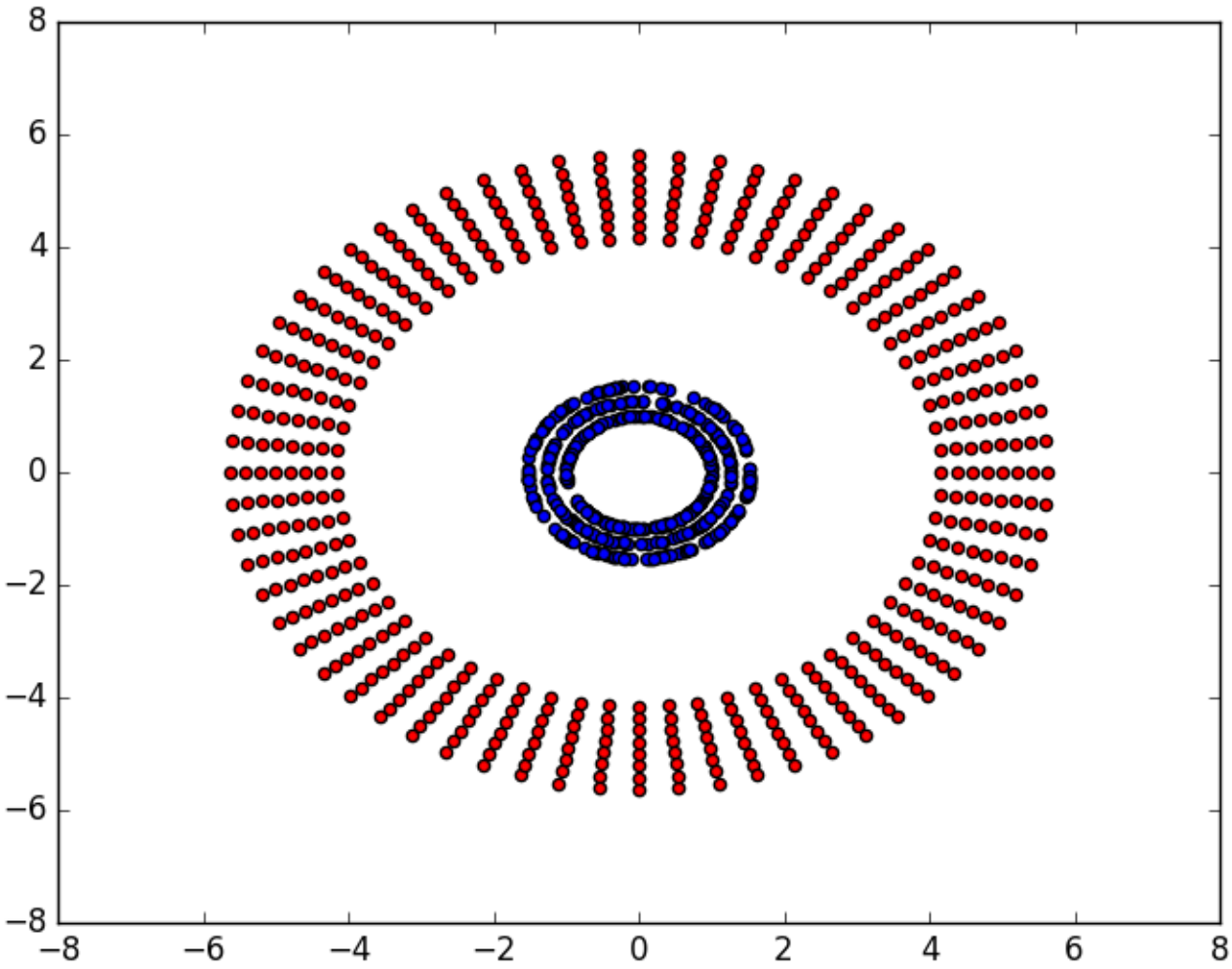

Figure 12: Clustering plot with single random landmark point- Trial Number 1

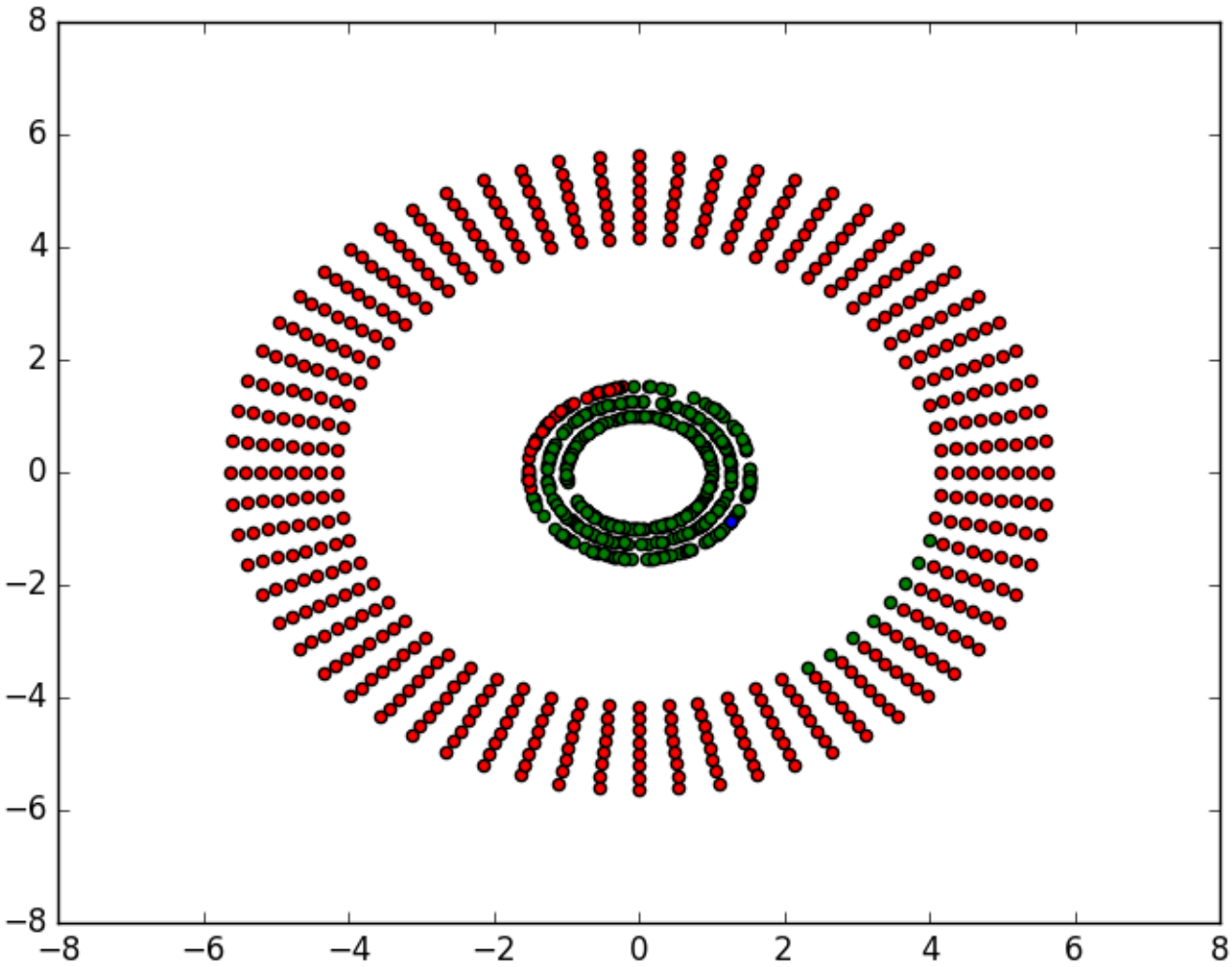

Figure 13: Clustering plot with single random landmark point- Trial Number 2

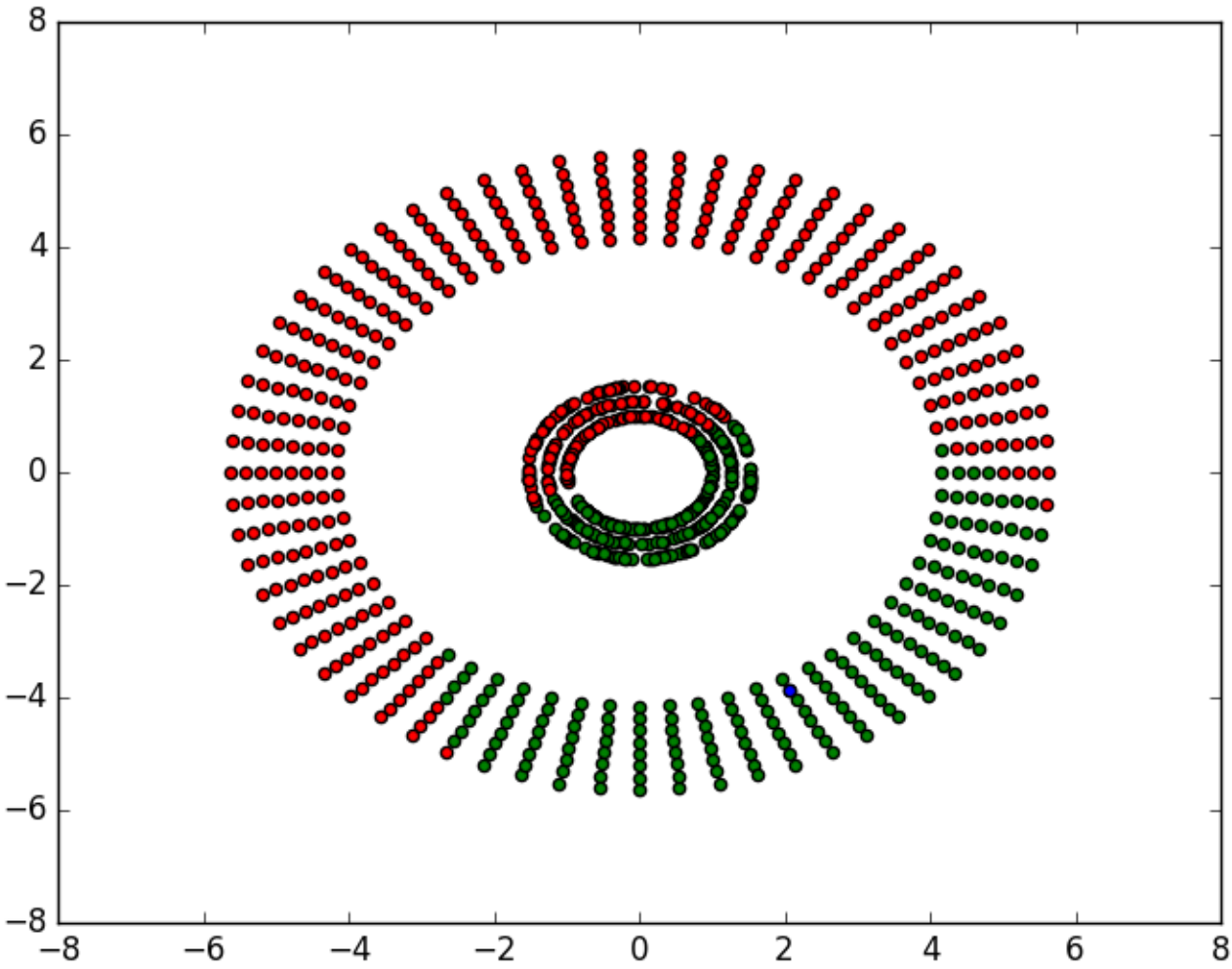

Figure 14: Clustering plot with single random landmark point- Trial Number 3

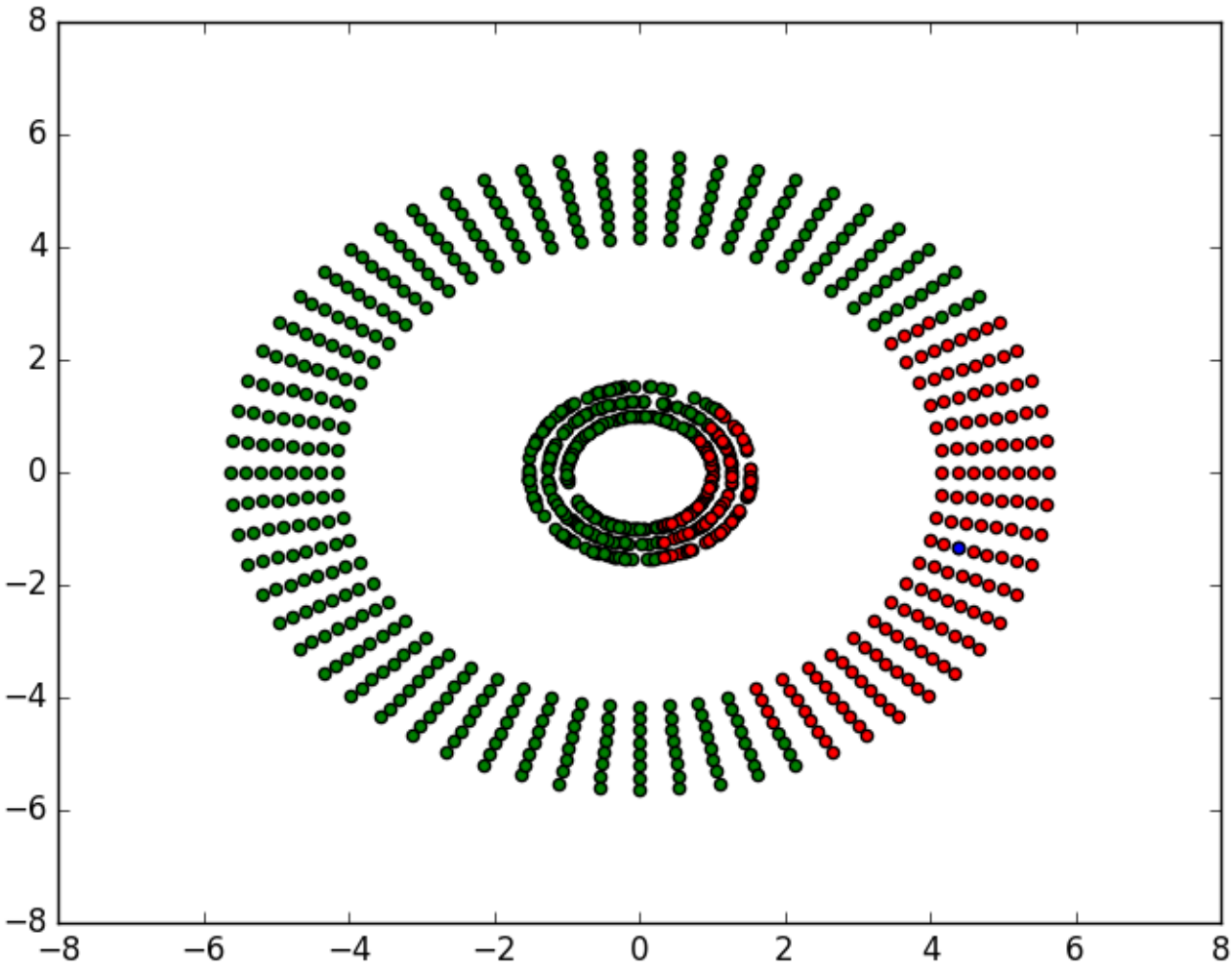

Figure 15: Clustering plot with single random landmark point- Trial Number 4

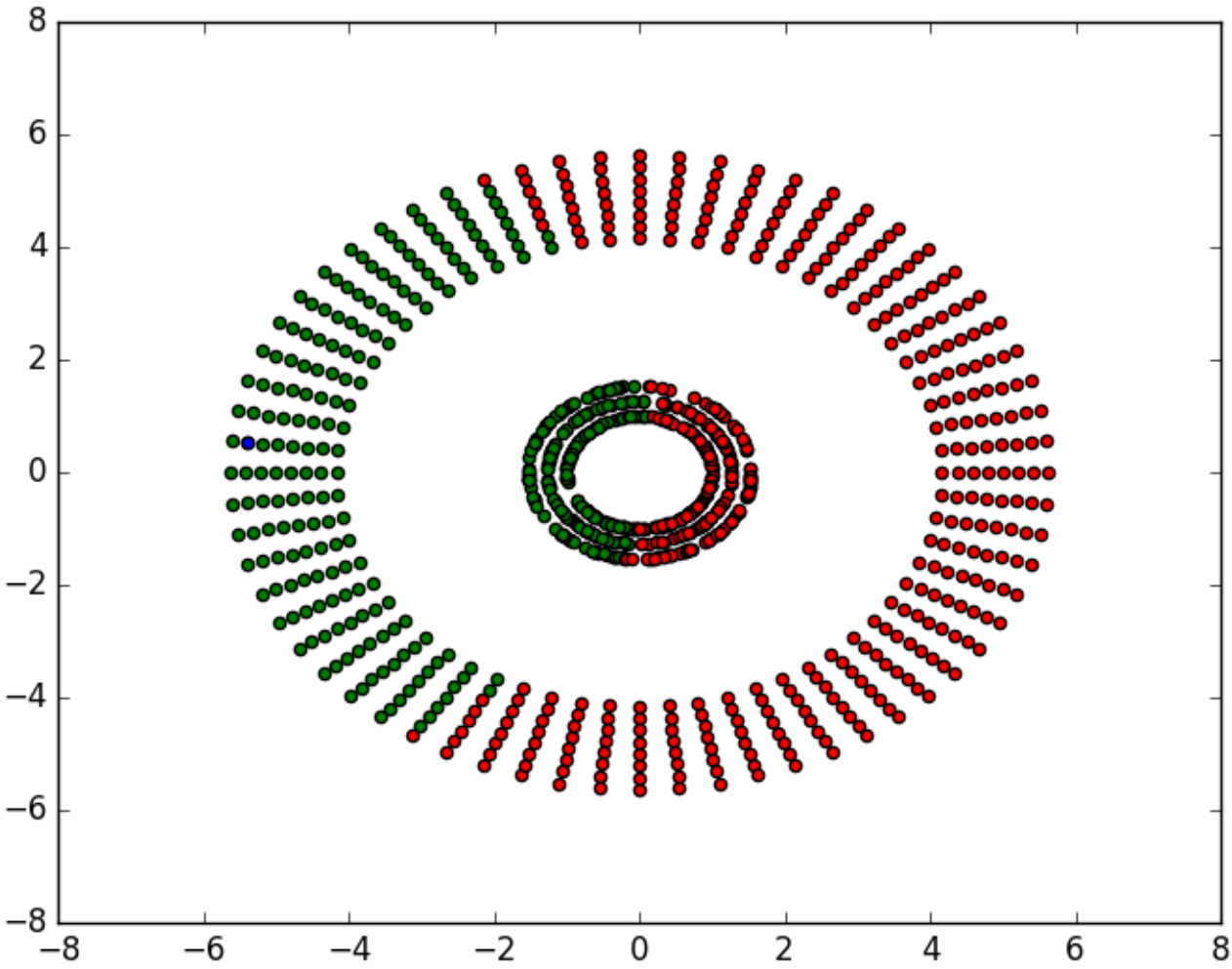

Figure 16: Clustering plot with single random landmark point- Trial Number 5

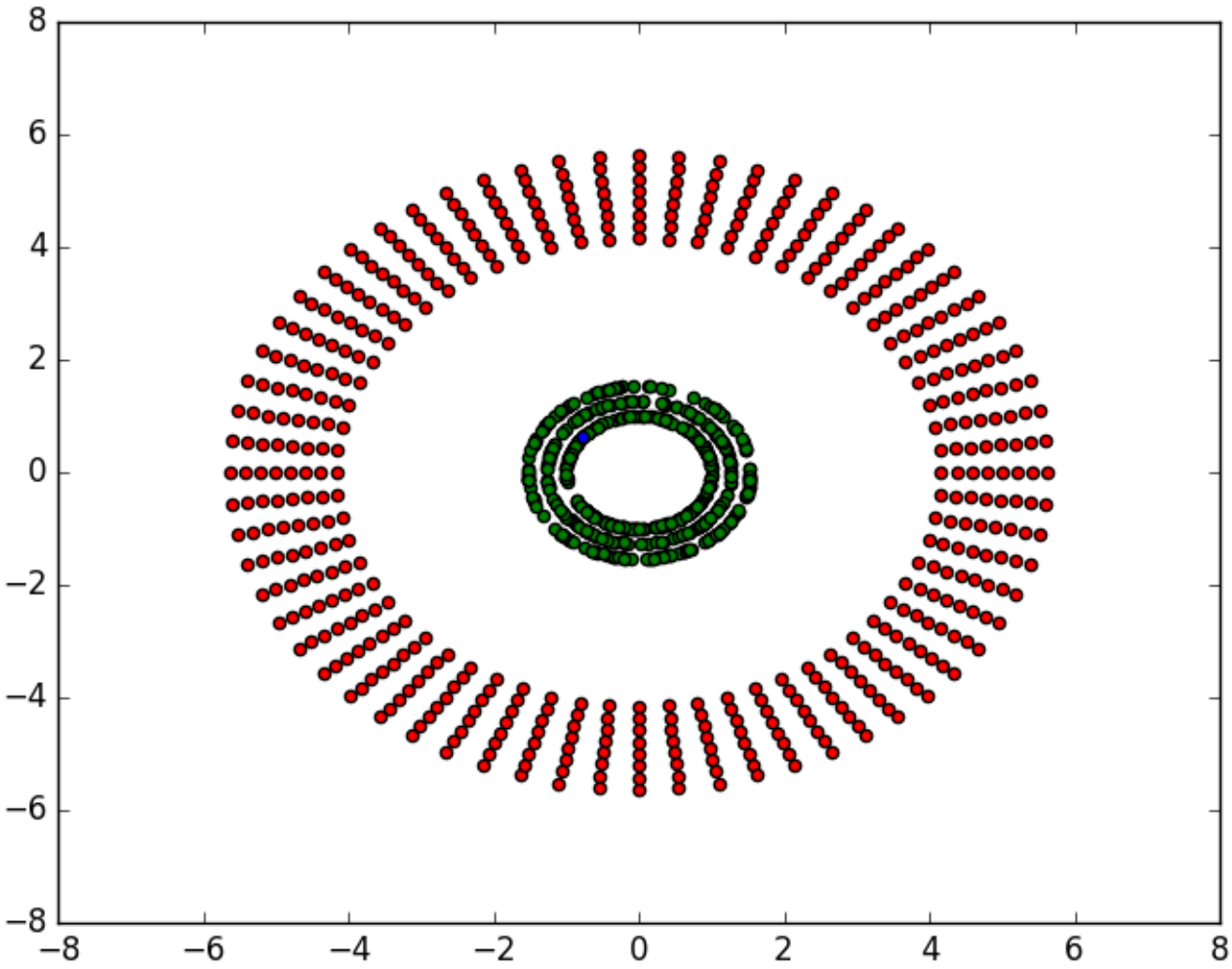

Figure 17: Clustering plot with single random landmark point- Trial Number 6

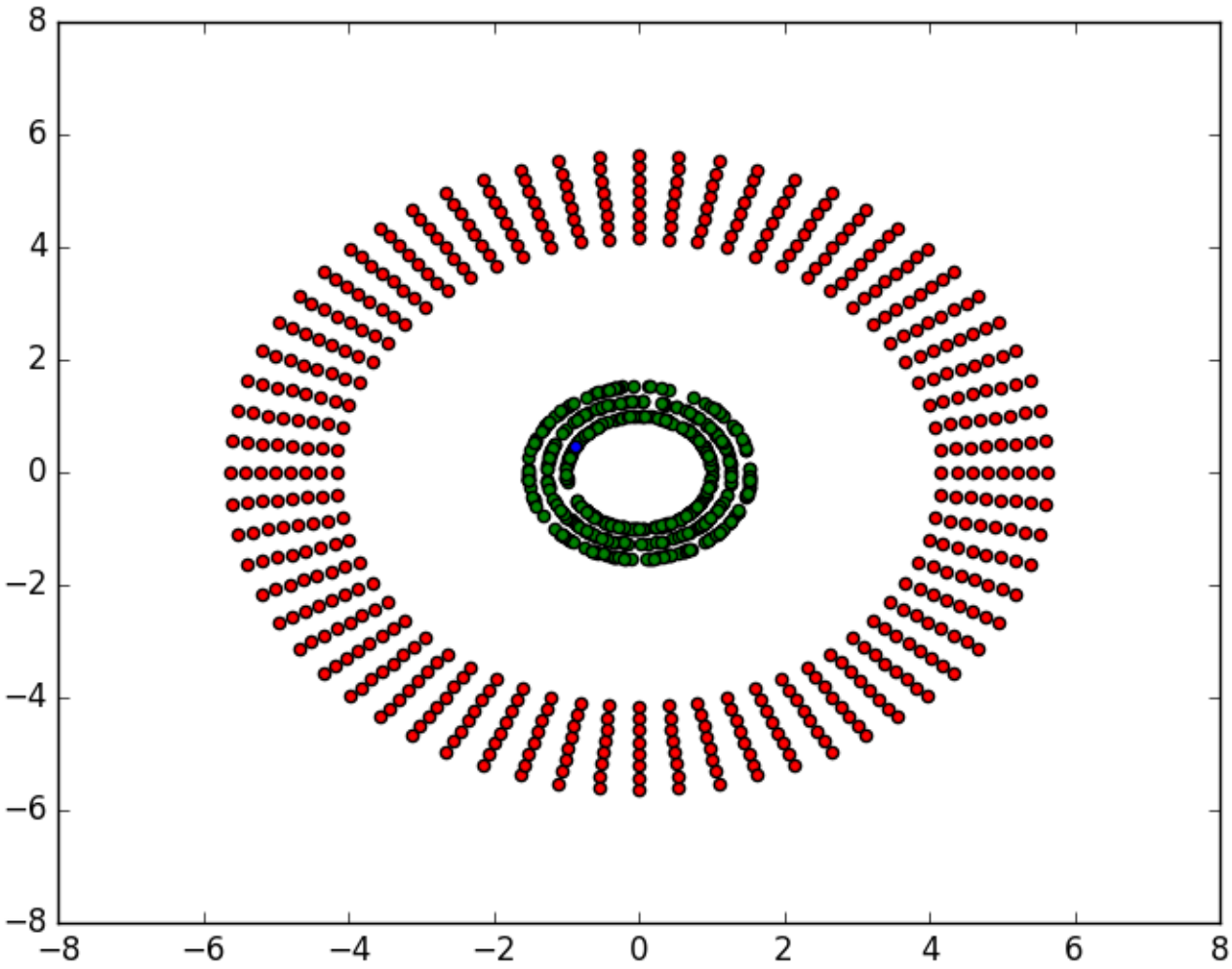

Figure 18: Clustering plot with single random landmark point- Trial Number 7

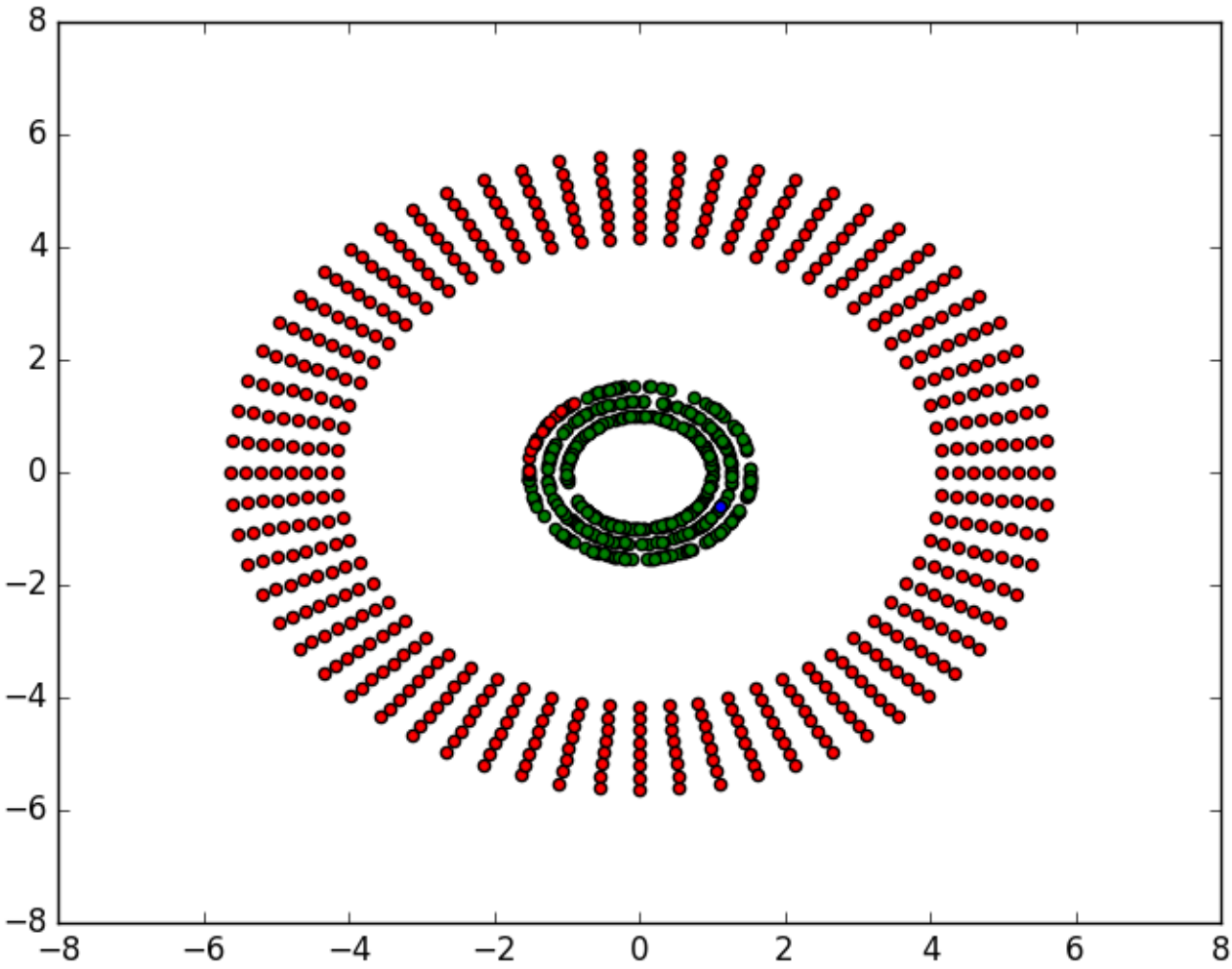

Figure 19: Clustering plot with single random landmark point- Trial Number 8

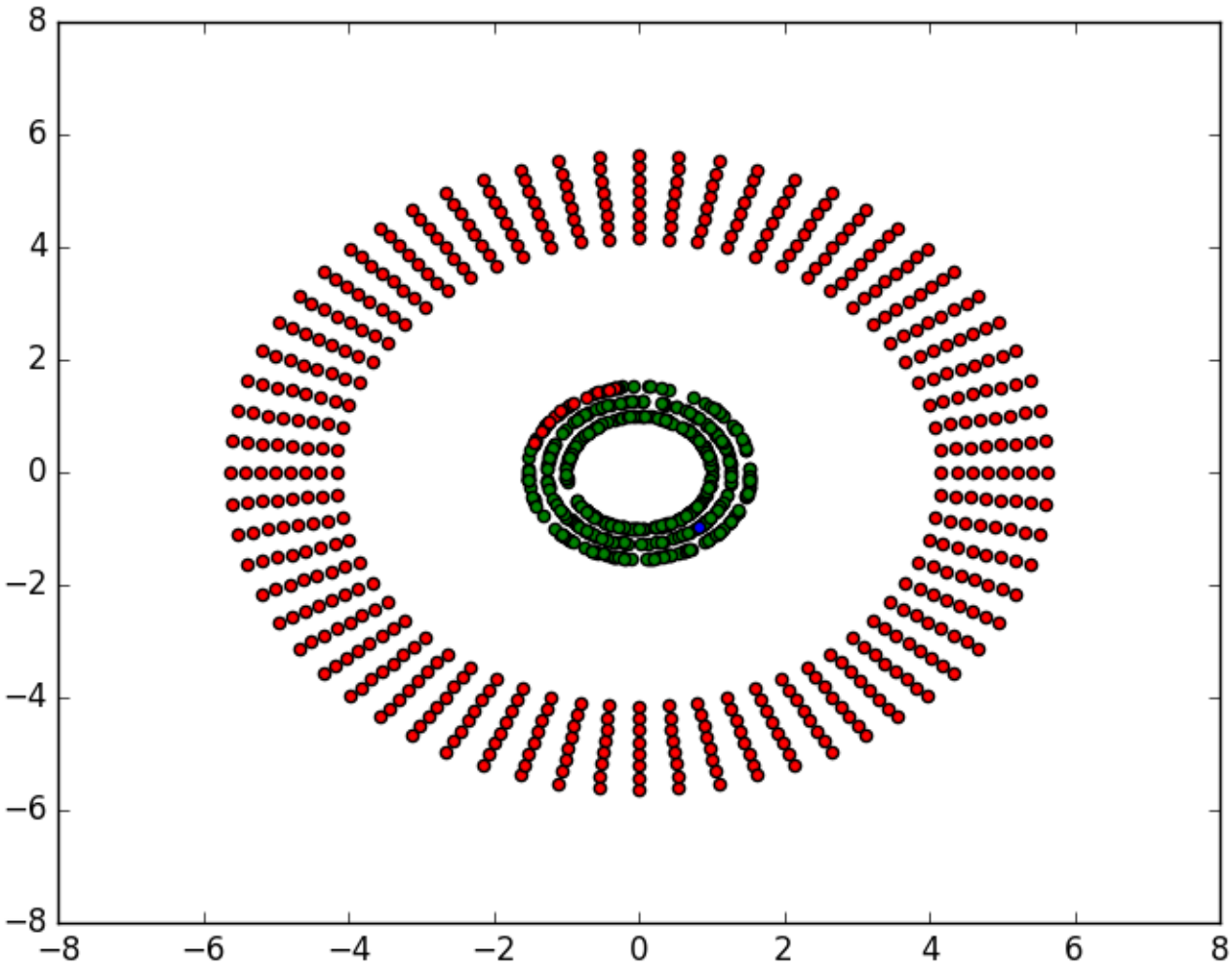

Figure 20: Clustering plot with single random landmark point- Trial Number 9

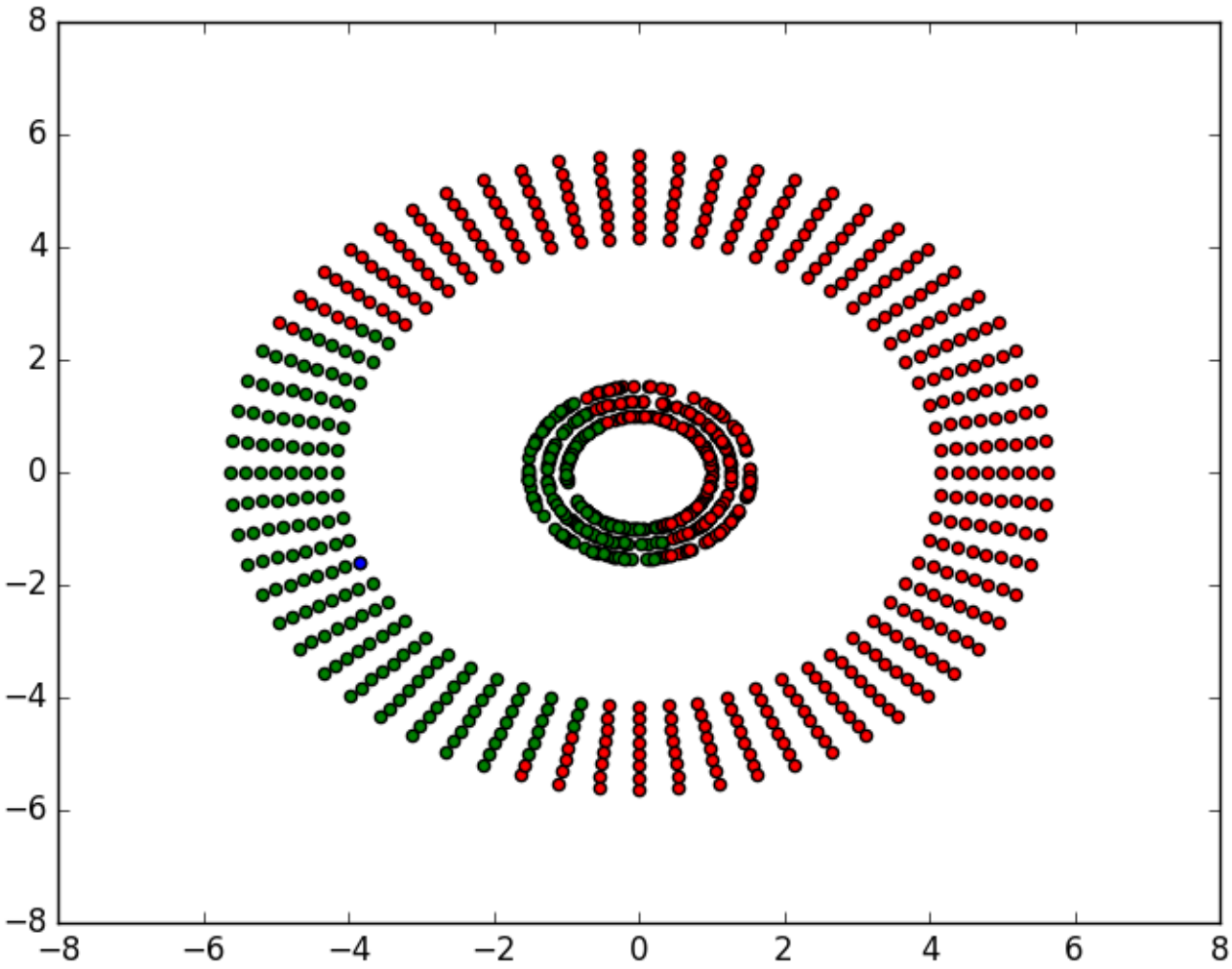

Figure 21: Clustering plot with single random landmark point- Trial Number 10