EECE5644 Fall 2019 – Exam 2

Alan Luke Jacob
NU ID: 001056590

Question 2:

1.

Question 2:

Ans] Given

$$x[n+1] = A\,x[n] + w[n]$$

where $x[n] = [h(nT), V_H(nT), a_H(nT), b(nT), V_b(nT), a_b(nT)]^T$

In the absence of model noise

$$x[n+1] = A \cdot x[n]$$

where
$$x[n+1] = \begin{bmatrix} h(nT) + V_H(nT)\Delta T + \frac{1}{2} a_H(nT)(\Delta T)^2 \\ V_H(nT) + a_H(nT) \cdot \Delta T \\ a_H(nT) \\ b(nT) + V_b(nT)\Delta T + \frac{1}{2} a_b(nT) \cdot \Delta T^2 \\ V_b(nT) + a_b(nT) \cdot \Delta T \\ a_b(nT) \end{bmatrix}$$

$$x[n+1] = A \cdot x[n]$$

$\therefore$ A matrix comes out to be

$$A = \begin{bmatrix} 1 & \Delta T & \Delta T^2/2 & 0 & 0 & 0 \\ 0 & 1 & \Delta T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta T & \Delta T^2/2 \\ 0 & 0 & 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Now $y[n] = \begin{bmatrix} h(nT) \\ b(nT) \end{bmatrix} + m[n] = C \cdot x[n] + m[n]$

$$\begin{bmatrix} h(nT) \\ b(nT) \end{bmatrix} = C \cdot \begin{bmatrix} h(nT) \\ v_H(nT) \\ a_H(nT) \\ b(nT) \\ v_b(nT) \\ a_b(nT) \end{bmatrix}$$

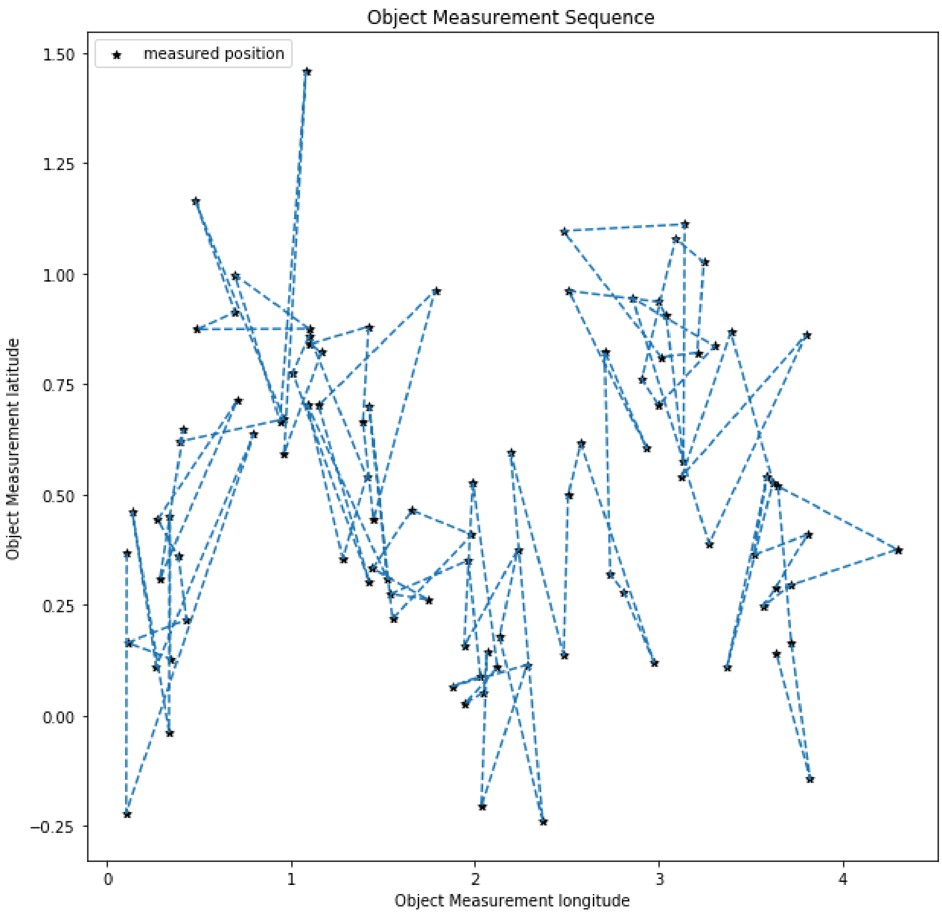
C is a transformation matrix that comes out to be,

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Substituting $\Delta T = 2$

$$A = \begin{bmatrix} 1 & 2 & 2 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Training Dataset plot using measurement values from Q2train.csv



3.

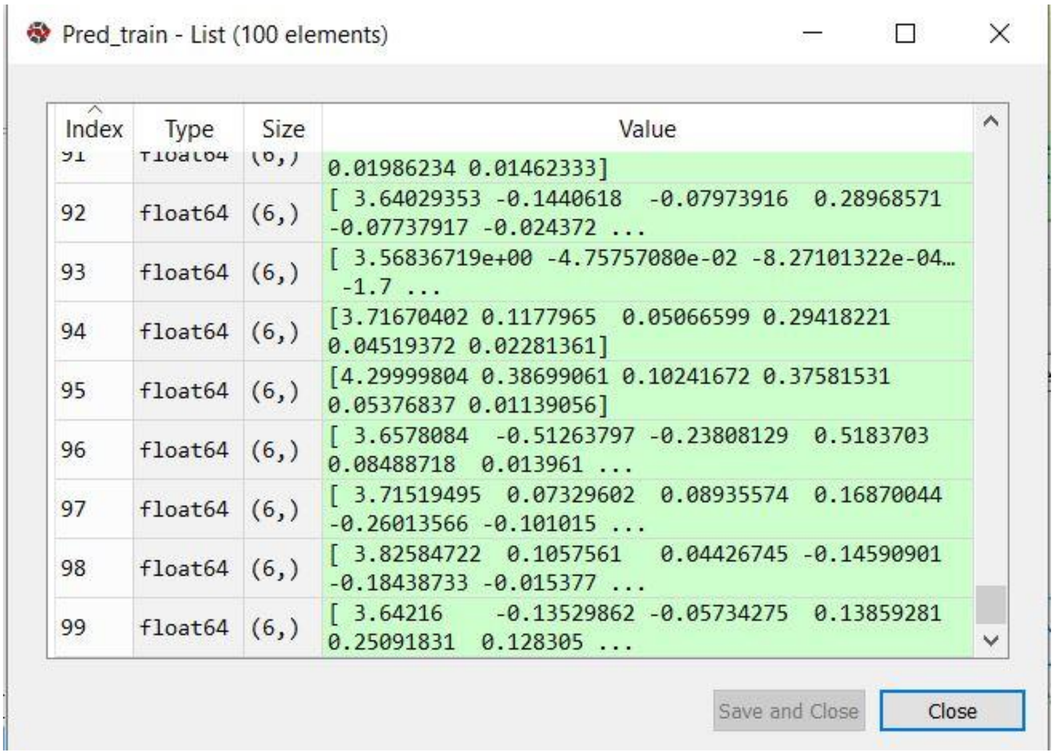
Kalman filter summary

- Model: $$\mathbf{x}_{t+1} = A\mathbf{x}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim W_t = N(\mathbf{0}, Q)$$ $$\mathbf{y}_t = C\mathbf{x}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim V_t = N(\mathbf{0}, R)$$
- Algorithm: repeat…
  - Time update: $$\hat{\mathbf{x}}_{t+1|t} = A\hat{\mathbf{x}}_{t|t}$$ $$P_{t+1|t} = AP_{t|t}A^T + Q$$
  - Measurement update:

$$\boxed{\begin{aligned} K_{t+1} &= P_{t+1|t}C^T\left(CP_{t+1|t}C^T + R\right)^{-1} \\ \hat{\mathbf{x}}_{t+1|t+1} &= \hat{\mathbf{x}}_{t+1|t} + K_{t+1}\left(\mathbf{y}_{t+1} - C\hat{\mathbf{x}}_{t+1|t}\right) \\ P_{t+1|t+1} &= P_{t+1|t} - K_{t+1}CP_{t+1|t} \end{aligned}}$$

The Kalman Filter was implemented on python by following the described sequence.

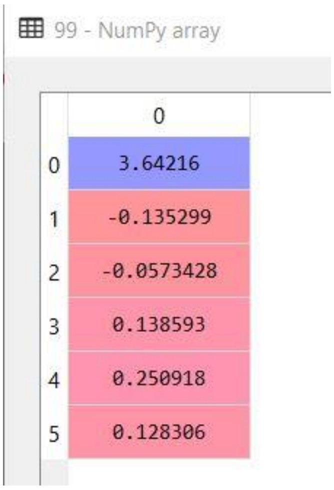
The Predicted estimate vectors xe[n] is shown for the training data in Q2train.csv



Pred_train - List (100 elements)

Index	Type	Size	Value
91	float64	(6,)	0.01986234 0.01462333]
92	float64	(6,)	[3.64029353 -0.1440618 -0.07973916 0.28968571 -0.07737917 -0.024372 ...
93	float64	(6,)	[3.56836719e+00 -4.75757080e-02 -8.27101322e-04… -1.7 ...
94	float64	(6,)	[3.71670402 0.1177965 0.05066599 0.29418221 0.04519372 0.02281361]
95	float64	(6,)	[4.29999804 0.38699061 0.10241672 0.37581531 0.05376837 0.01139056]
96	float64	(6,)	[3.6578084 -0.51263797 -0.23808129 0.5183703 0.08488718 0.013961 ...
97	float64	(6,)	[3.71519495 0.07329602 0.08935574 0.16870044 -0.26013566 -0.101015 ...
98	float64	(6,)	[3.82584722 0.1057561 0.04426745 -0.14590901 -0.18438733 -0.015377 ...
99	float64	(6,)	[3.64216 -0.13529862 -0.05734275 0.13859281 0.25091831 0.128305 ...

Save and Close Close



99 - NumPy array

	0
0	3.64216
1	-0.135299
2	-0.0573428
3	0.138593
4	0.250918
5	0.128306

It can be observed that the 0th and 3rd row value of xe[n], h=3.64216 b=0.138593 for t=199 gives the estimate of h(longitude),b(latitude) of the t=199 measured observation h=3.6398,b=0.14193. This shows how well the algorithm converges to the true estimate of object position.

Algorithm:

1. The values of the matrices A,C,Pi,Q,R were first initialized. The initial sequence xi_initial is defined by [0,0,0,0,0,0]
2. The time updates to the algorithm to the values of xhat and Phat are computed according to the formulas mentioned above.
3. Using these computed values of xhat and Phat the Kalman gain is calculated as described above.
4. The estimated vector xe[n] is then calculated. Note yt+1 is obtained from the Q2train.csv which are the noisy measurements. The P matrix is then updated.
5. Steps 2 to 4 are continuously run in a loop to obtain the estimates at different time intervals

4. The Values of estimated positions are obtained from the estimated vector xe[n] by:

Y=CX, where C is the transformation matrix described above. C is a 2*6 matrix.

Predtrain - NumPy array

	0	1
90	[illegible]	[illegible]
91	3.8077	0.406359
92	3.64029	0.289686
93	3.56837	0.245682
94	3.7167	0.294182
95	4.3	0.375815
96	3.65781	0.51837
97	3.71519	0.1687
98	3.82585	-0.145909
99	3.64216	0.138593

Format Resize Background color

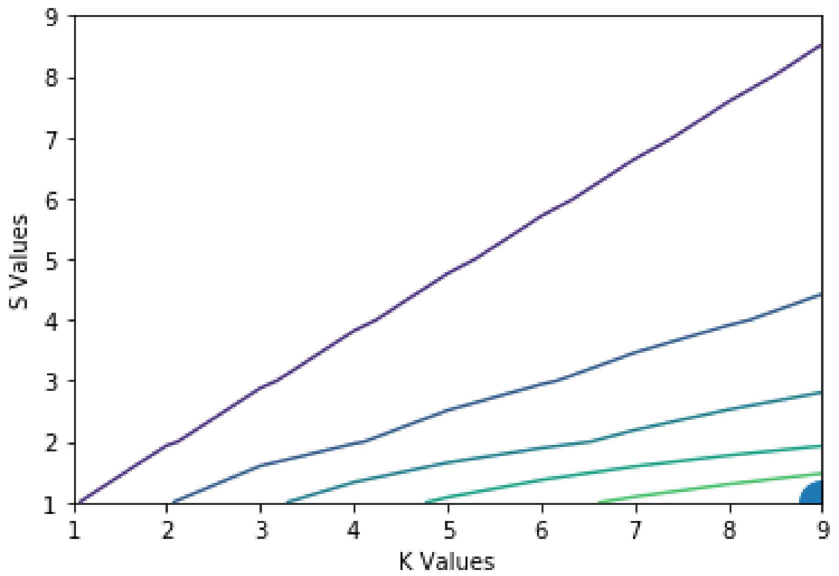
Predtest - NumPy array

	0	1
90	[illegible]	[illegible]
91	3.63133	0.115779
92	3.4184	0.322661
93	3.81899	0.0672585
94	3.43492	0.406922
95	3.76781	0.253165
96	4.06775	0.277799
97	3.81004	0.0208906
98	3.7742	-0.00321541
99	3.97544	0.139617

Format Resize Background color

5. Hyperparameter Optimization:
   The values of K, S were both varied from 1 to 9 and the corresponding cross validation score was computed.

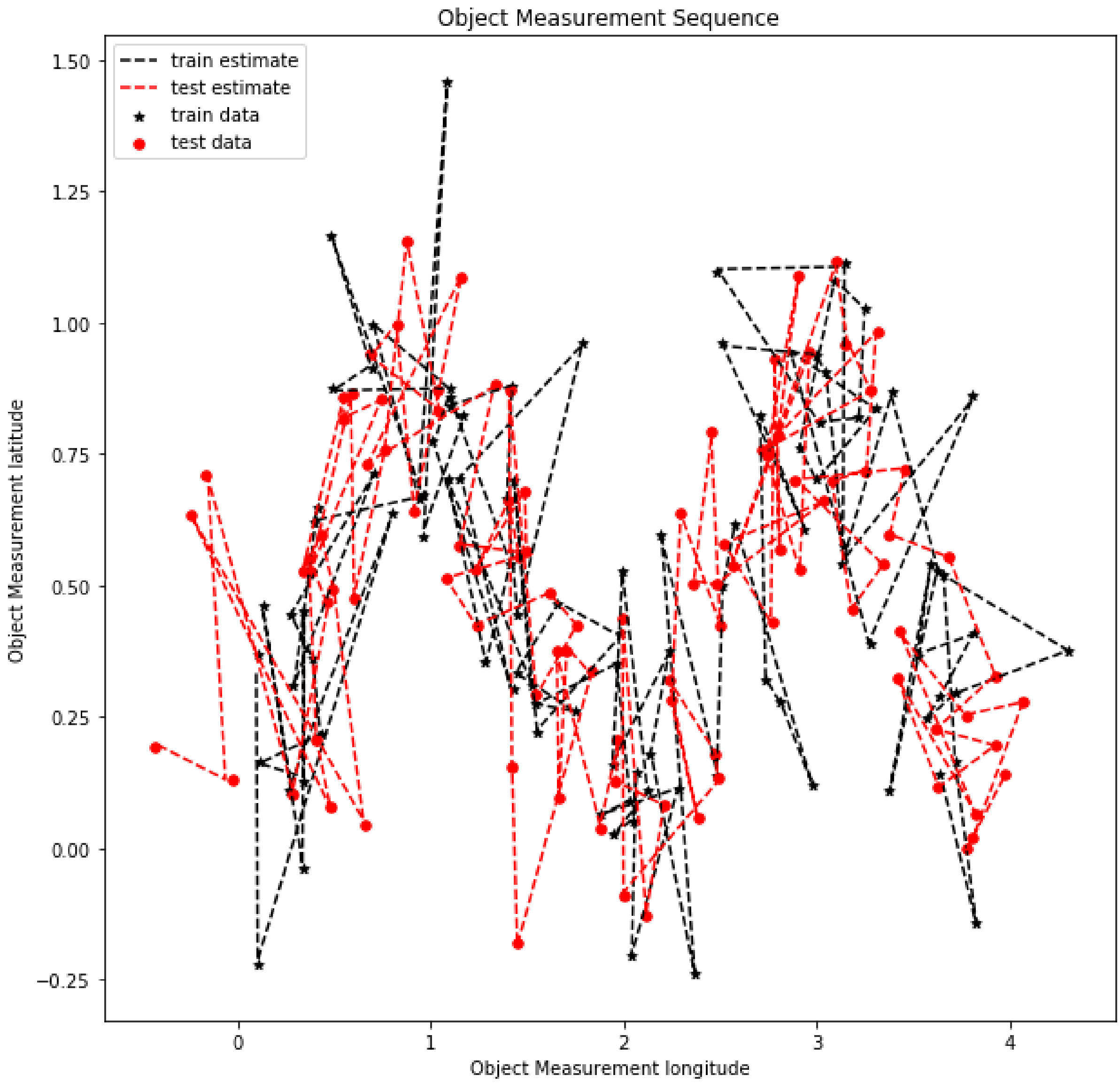
   Output:
   The optimal K, S is shown with a marker in the figure below
   The optimal pair (K, S) which gives the minimum cross validation metric is:
   9.0
   1.0
   The value of minimum cross validation metric is:
   6.532112251137055e-05



Cross Validation scores for different K, S pairs

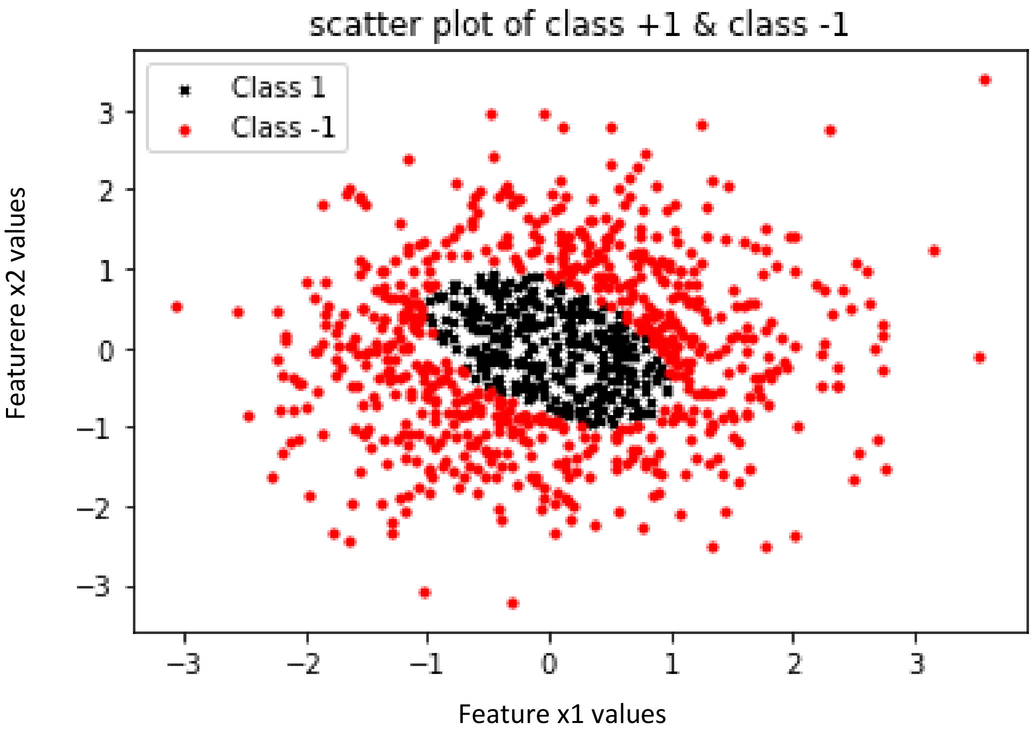
```
In [518]: scores[:,2]
Out[518]:
array([1.87079925e-03, 3.89853438e-03, 5.59983886e-03, 7.04748490e-03,
       8.30271167e-03, 9.40385596e-03, 1.03847978e-02, 1.12726647e-02,
       1.20804861e-02, 7.60271046e-04, 1.87079925e-03, 2.93258347e-03,
       3.89853438e-03, 4.78032162e-03, 5.59983886e-03, 6.35323026e-03,
       7.04748490e-03, 7.69632409e-03, 4.16712593e-04, 1.12705729e-03,
       1.87079925e-03, 2.58901277e-03, 3.26525840e-03, 3.89853438e-03,
       4.49123864e-03, 5.06092219e-03, 5.59983886e-03, 2.63909257e-04,
       7.60271046e-04, 1.31343404e-03, 1.87079925e-03, 2.41310130e-03,
       2.93258347e-03, 3.42752609e-03, 3.89853438e-03, 4.34675223e-03,
       1.82371656e-04, 5.49644389e-04, 9.78896394e-04, 1.42556211e-03,
       1.87079925e-03, 2.30628594e-03, 2.72778001e-03, 3.13350717e-03,
       3.52366027e-03, 1.33641285e-04, 4.16712593e-04, 7.60271046e-04,
       1.12705729e-03, 1.50022940e-03, 1.87079925e-03, 2.23456581e-03,
       2.58901277e-03, 2.93258347e-03, 1.02172018e-04, 3.27179774e-04,
       6.08690934e-04, 9.15854335e-04, 1.23347781e-03, 1.55348611e-03,
       1.87079925e-03, 2.18309479e-03, 2.48882054e-03, 8.06719501e-05,
       2.63909257e-04, 4.98918867e-04, 7.60271046e-04, 1.03430940e-03,
       1.31343404e-03, 1.59337964e-03, 1.87079925e-03, 2.14436226e-03,
       6.53211225e-05, 2.17481689e-04, 4.16712593e-04, 6.41927728e-04,
       8.81026330e-04, 1.12705729e-03, 1.37573478e-03, 1.62437070e-03,
       1.87079925e-03])
```

6. Plot of training and test samples along with the estimated training and test values.



Question 1:

1. Scatter Plot of entire Dataset with Class labels +1 & -1



Number of samples generated for Class +1

351

Number of samples generated for Class -1

649

2. ID3 algorithm using entropy as measure of subpopulation purity:

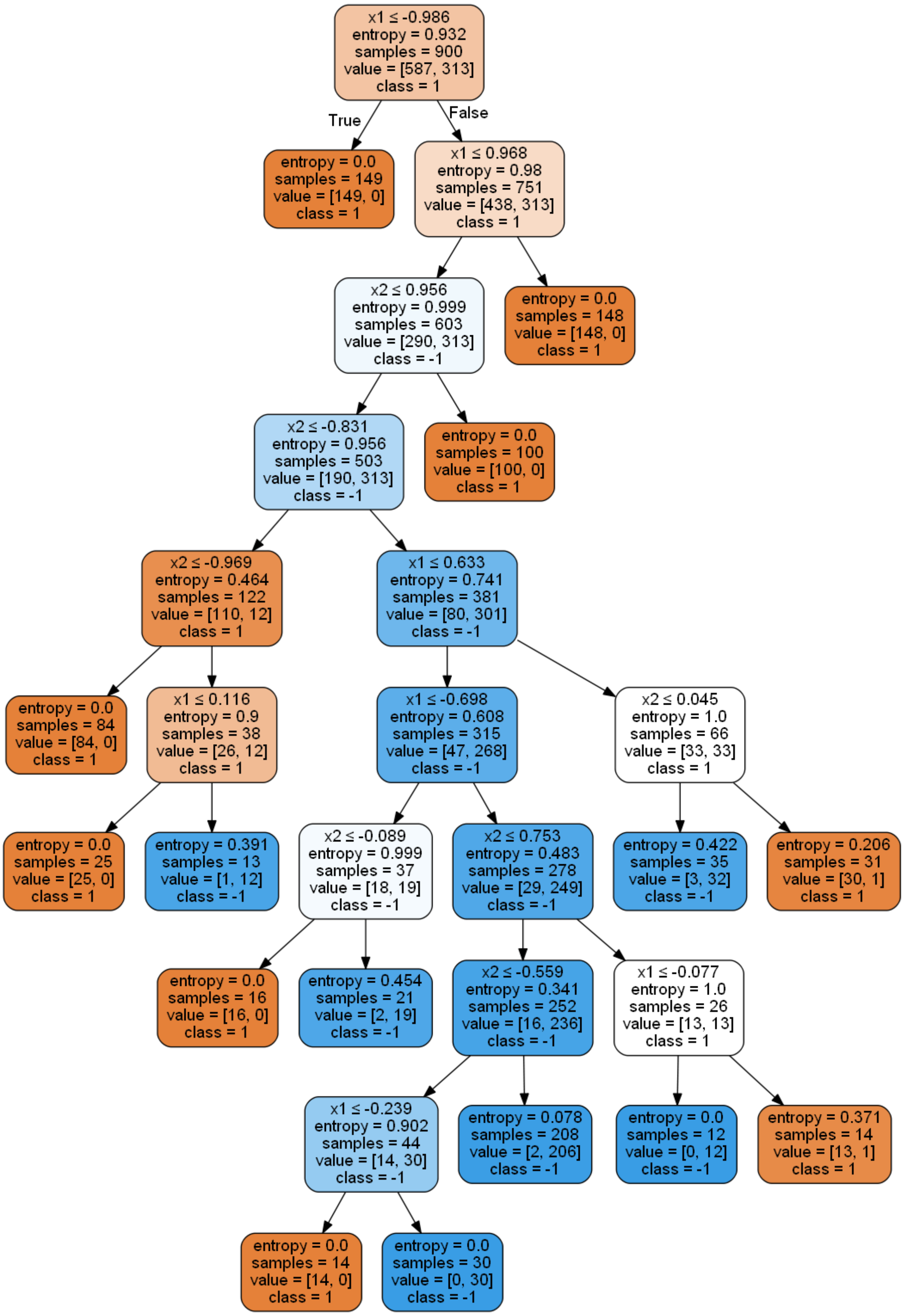
Confusion matrix:

confusion_ID3 - NumPy array

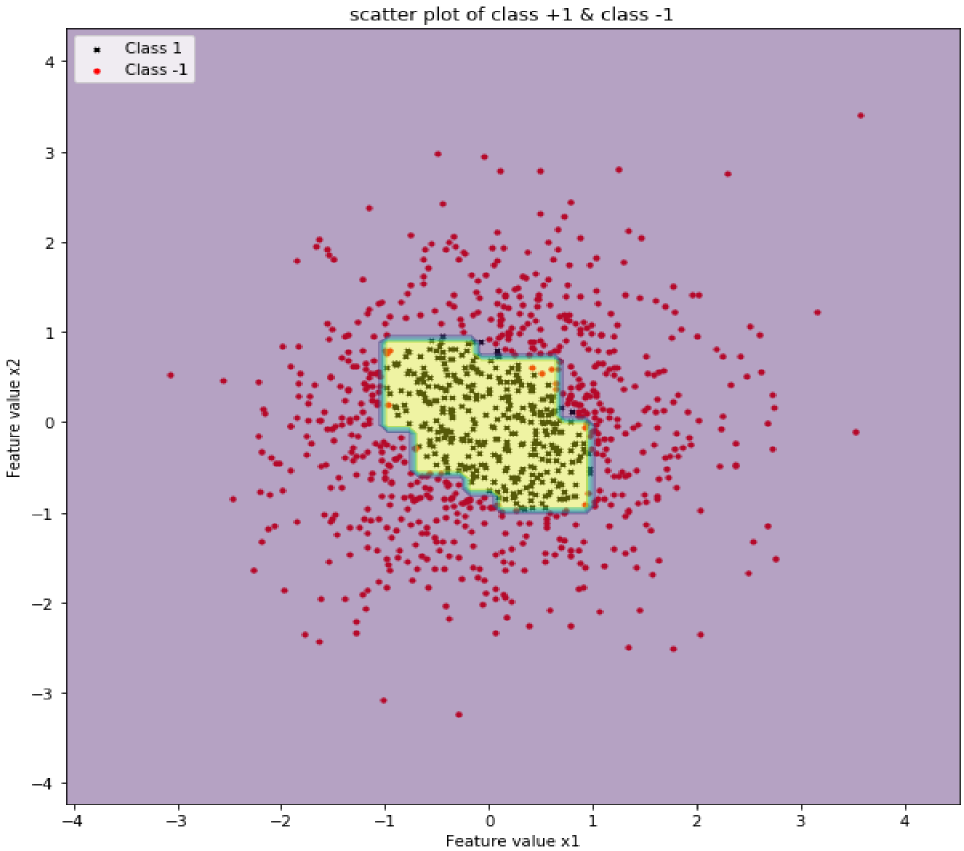
	0	1
0	60	2
1	2	36

The accuracy of the ID3 decision tree is:
Accuracy: 0.96

Tree for ID3 Algorithm:



Decision Boundary ID3: Algorithm



from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion="entropy", max_depth=11,min_impurity_decrease =0.01)

The maximum depth of the tree is defined by max_depth=11.

min_impurity_decrease defines that a node will be split if this split induces a decrease of the impurity greater than or equal to this value.

Bagging Decision tree Classifier:

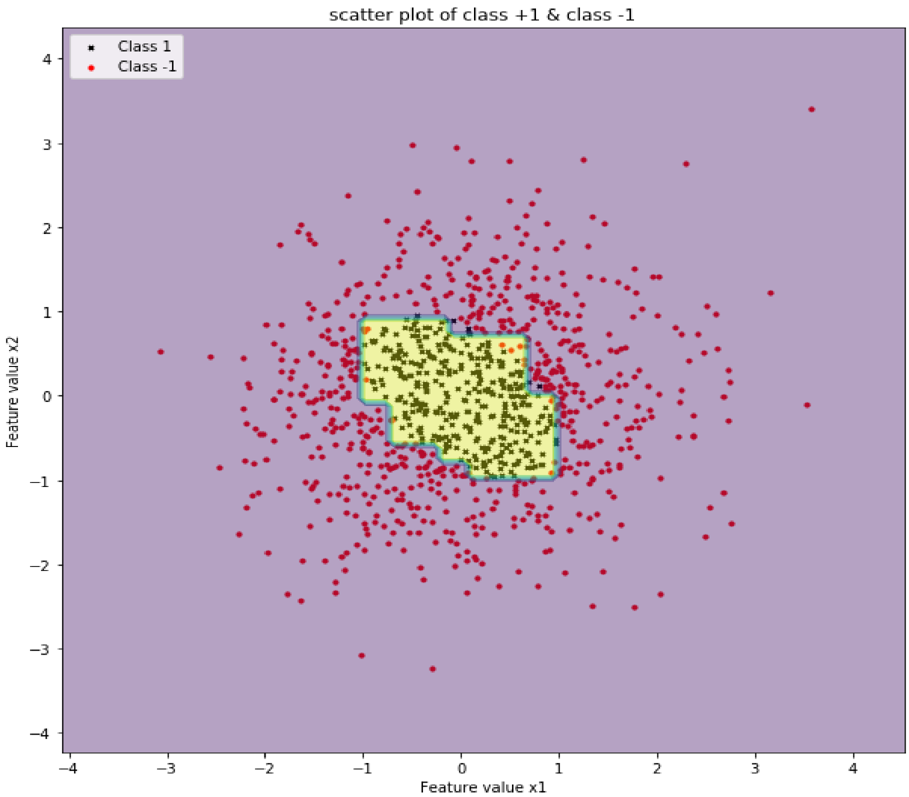
Accuracy of Bagging classifier: 0.96

Confusion matrix:

confusion_bagging - NumPy array

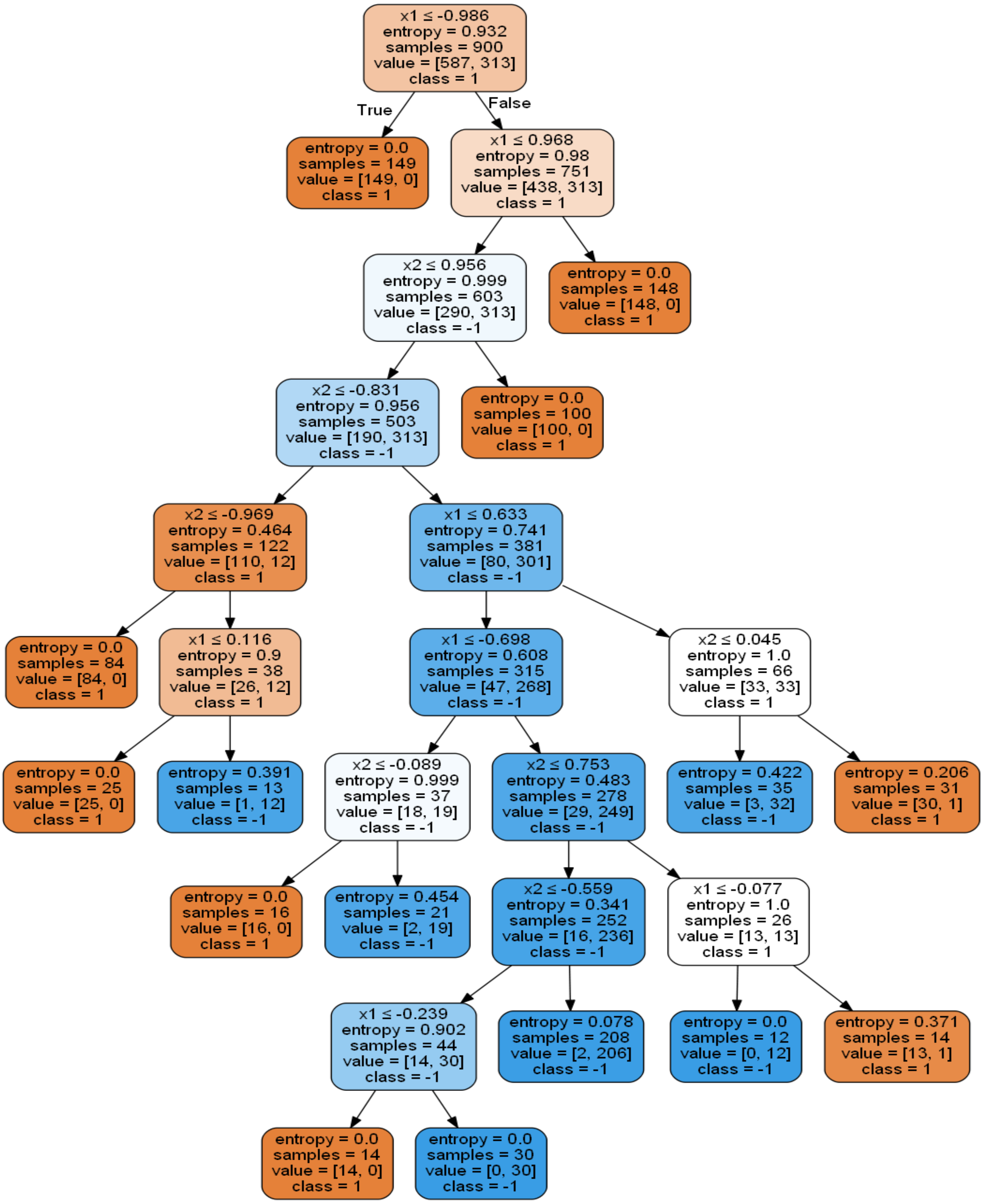
	0	1
0	60	2
1	2	36

Decision Boundary Bagging classifier:



Decision Tree for Bagging Classifier:

Note all 7 trees were observed to have the same decision tree



from sklearn.ensemble import BaggingClassifier

clfbagg = BaggingClassifier(base_estimator=clf, n_estimators=7,bootstrap=True)

The base_estimator passed is the object of the ID3 decision tree.

The number of base estimators in the ensemble is defined as 7

Bootstrap = True defines that samples are drawn with replacement from the training dataset

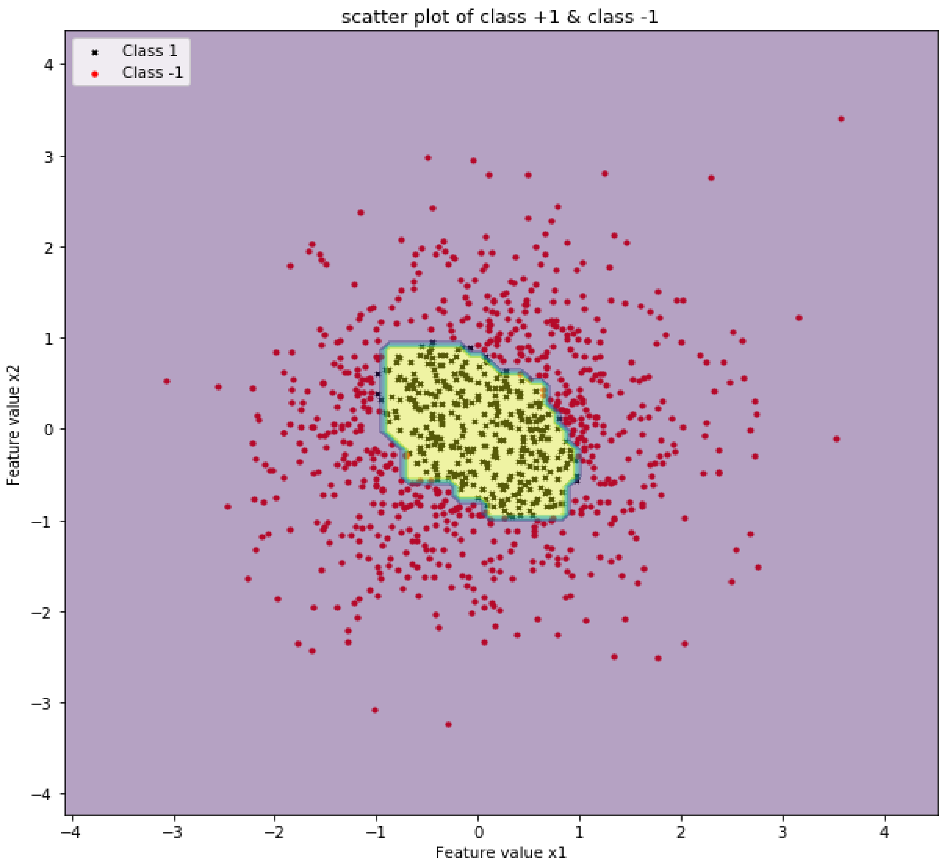
Adaboost Algorithm:

confusion_boost - NumPy array

	0	1
0	61	1
1	1	37

Accuracy of Adaboost classifier: 0.98

Decision Boundary:



The initial level weights assigned to each of the 7 estimators of the Adaboost algorithm

[0 0 0 0 0 0 0]

Final level weights assigned to the 7 estimators

[4.48863637 4.47733681 4.16177242 4.25510077 2.75899969 2.81466988

4.08100689]

The initial sample weights are equal to 1/N where N is the number of training examples. Therefore, the initial weights are 0.00111.

ADABOOST Weight Updation.

For N samples = 900 samples $(x_i, y_i)_{i=1:900}$

$x_i \in R^k$, $y_i \in -1, +1$

1. Assign the initial weights w_i for the N samples as $w_i = 1/N$, $i = 1, \dots, 900$

$$\therefore \boxed{w_i = 0.00111}$$

Now for $m=1$ to M

1. Select & extract from the pool of classifiers the classifier k_m which minimizes

$$W_e = \sum_{y_i \neq k_m(x_i)} w_i^{(m)}$$

2. Set the weight α_m of the classifier to

$$\alpha_m = \frac{1}{2} \ln\left(\frac{1-e_m}{e_m}\right)$$

where $e_m = \frac{W_e}{W}$

3. Update weights of the training example for the next iteration

If $h_m(x_i)$ is a miss, set

$$w_i^{(m+1)} = w_i^{(m)} e^{\alpha_m} = w_i^{(m)} \sqrt{\frac{1-e_m}{e_m}}$$

else

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{-\alpha_m} = w_i^{(m)} \sqrt{\frac{e_m}{1-e_m}}$$

The final output of the classifier

$$H(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right)$$

In this question $M=7$.

It is noticed that the highest weight is assigned to the first decision stump of the Adaboost classifier. Therefore, it gets more say in the final classification of the sample over the other estimators

```
from sklearn.ensemble import BaggingClassifier

clboost = AdaBoostClassifier(base_estimator=clf, n_estimators=7, learning_rate=1.0, algorithm='SAMME', random_state=None)

clboost = clboost.fit(X_train,y_train)

y_predboost=clboost.predict(X_test)
```

Classifier	Number of Training samples	Number of Test samples	Number of Misclassified samples	Accuracy
ID3	900	100	4	0.96
Bagging classifier	900	100	2	0.96
Adaboost	900	100	2	0.98

It can be noticed from the decision boundary that the Adaboost algorithm results in a smoother decision boundary and also has the smallest number of misclassified samples.

The bagging classifier classifies the samples based on a majority vote obtained from each of the 7 estimators generated which are samples on the training data with replacement so that the size of the training data set is 900 for each of them. The Bagging classifier was shown to have a decision boundary and accuracy very similar to the ID3 algorithm in this case.

ML Exam 1:

Question 3:

ML EXAM 1

Q3] MAP estimate for the parameter vector w

A3] $y = ax^3 + bx^2 + cx + d + v$

$$y = \begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} x^3 \\ x^2 \\ x \\ 1 \end{bmatrix} + v$$

where $w \sim \mathcal{N}(0, \gamma^2 I)$ $\quad v \sim N(0, \sigma^2)$

Given $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$, To estimate w

$$\hat{w}_{MAP} = \arg\max_w \ln \left[P(w / x_i, y_i)_{i=1:N} \right]$$

$$= \arg\max_w \ln \left[P\left(x_i, y_i / w \right) \cdot P(w) \right] \quad \text{By Bayes rule}$$

$$= \arg\max_w \sum_{i=1}^{N} \left[\ln \left[P(x_i, y_i / w) \cdot P(w) \right] \right]$$

$$\hat{w}_{MAP} = \arg\max_w \sum_{i=1}^{N} \left[\ln \left\{ P\left(y_i / x_i, w \right) \cdot P\left(x_i / w \right) \right] + \ln\left[P(w)\right] \right]$$

since x_i, y_i are independent of each other

$$= \arg\max_w \sum_{i=1}^{N} \left[\left\{ \ln \left[P\left(y_i / x_i, w \right) \right] + \ln \left[P\left(x_i / w \right) \right] \right\} \right] + \ln\left[P(w)\right]$$

Note $P(x_i / w)$ is constant w.r.t w & can be removed from the eq^n

$$\hat{\omega}_{MAP} = \arg\max_{\omega} \sum_{i=1}^{N} \left\{ \ln\left[P(Y_i / x_i, \omega)\right] \right\} + \ln\left[P(\omega)\right]$$

Here

$$P(y_i / x_i, w) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(y_i - w^T b(x_i))^2}{2\sigma^2}}$$

$$P(\omega) = \frac{1}{(2\pi)^{4/2} \cdot \gamma^4 |I|} e^{-\frac{\omega^T \omega}{2\gamma^2}}$$

$$= \arg\max_{\omega} \; -\frac{1}{2} \sum_{i=1}^{N} \frac{(y_i - w^T b(x_i))^2}{\sigma^2} - \frac{1}{2} \frac{\omega^T \omega}{\gamma^2}$$

Let $b_i = b(x_i)$

$$= \arg\min_{\omega} \frac{1}{\sigma^2} \sum_{i=1}^{N} [y_i - \omega^T b_i]^2 + \frac{1}{\gamma^2} \omega^T \omega$$

$$= \arg\min_{\omega} \sum_{i=1}^{N} (y_i - \omega^T b_i)(y_i - b_i^T \omega) + \frac{\sigma^2}{\gamma^2} \omega^T \omega$$

$$= \arg\min_{\omega} \sum_{i=1}^{N} y_i^2 - 2\omega^T \sum_{i=1}^{N} b_i + \omega^T \left(\sum_{i=1}^{N} b_i . b_i^T \right) \omega + \frac{\sigma^2}{\gamma^2} \omega^T \omega$$

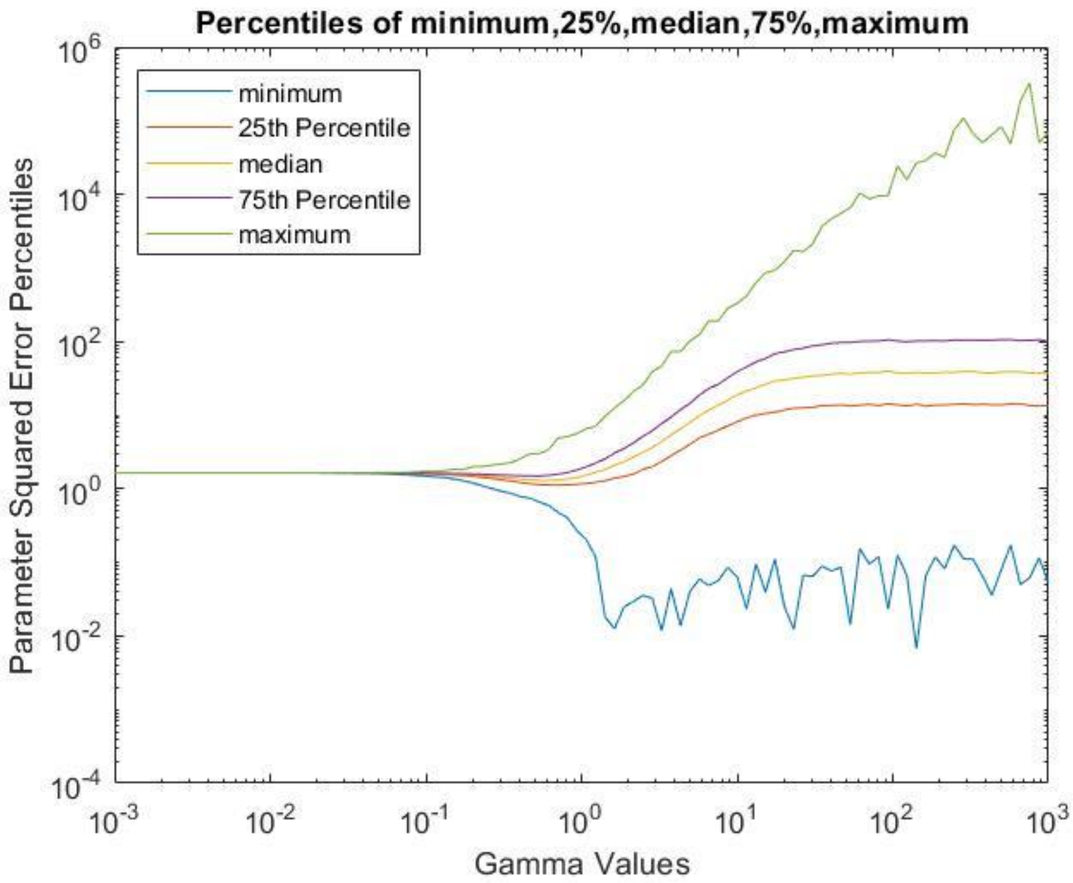
Taking $\frac{\partial}{\partial \omega}$ of the objective equation, and equating it to zero to solve for $\hat{\omega}_{MAP}$

$$-2\left(\sum_{i=1}^{N} b_i \right) + 2\left(\sum_{i} b_i b_i^T \right) \hat{\omega}_{MAP} + \frac{2\sigma^2}{\gamma^2} \hat{\omega}_{MAP} = 0$$

$$\left(\sum_{i=1}^{N} b_i . b_i^T + \frac{\sigma^2}{\gamma^2} I \right) \hat{\omega}_{MAP} = \sum_{i=1}^{N} b_i$$

$$\therefore \hat{\omega}_{MAP} = \left(\sum_{i=1}^{N} b_i . b_i^T + \frac{\sigma^2}{\gamma^2} I \right)^{-1} \left(\sum_{i=1}^{N} b_i \right)$$

Output:



References:

1. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
3. https://towardsdatascience.com/machine-learning-part-17-boosting-algorithms-adaboost-in-python-d00faac6c464
4. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier.feature_importances_
5. https://medium.com/@jaems33/understanding-kalman-filters-with-python-2310e87b8f48
6. https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.contour.html
7. [L6.8]KalmanFiltering
8. Probability, Statistics, and Random Processes for Engineers, 4th Edition, Henry Stark