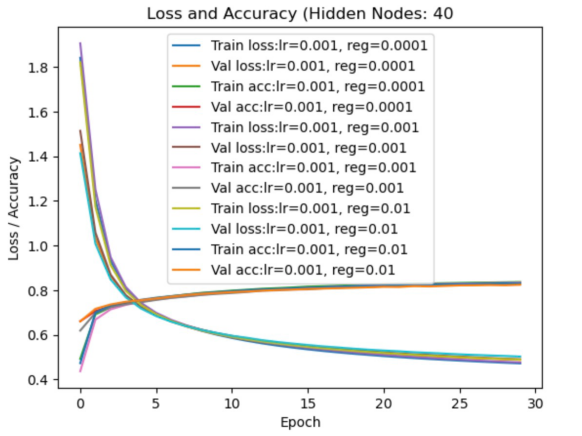
EE559-Homework#7-Lei Lei

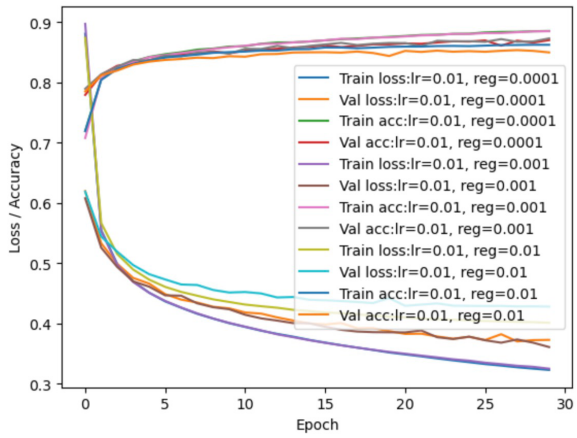
1. **Hyper-parameter Optimization for MLP on FMNIST**

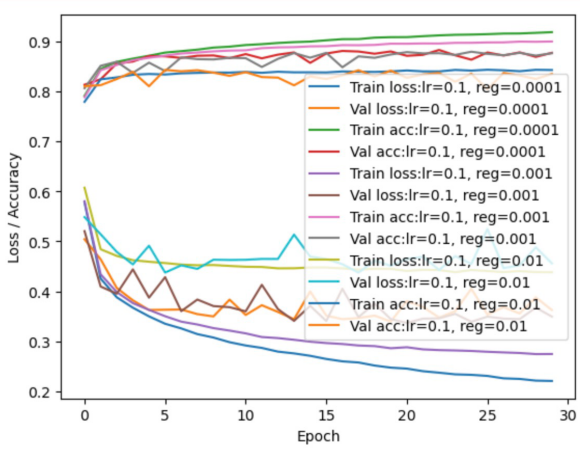
(a) The batch size is 64, and the mean is 10.510799455642701, the std is 3.460327396677144

(b) For each value of hidden nodes, produce learning curves (include train/val and do for both loss/accuracy plots).

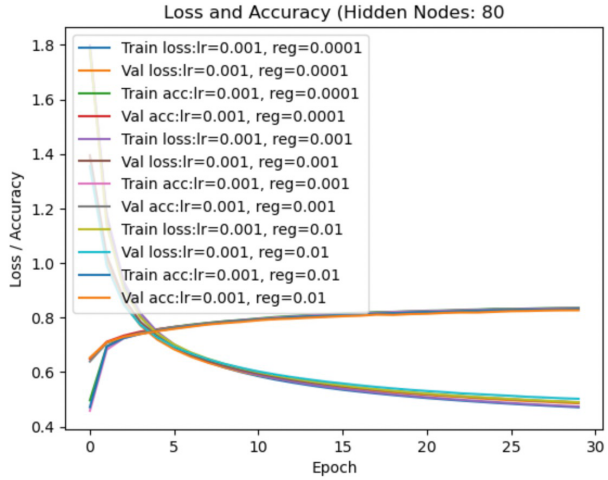
Hidden nodes = 40

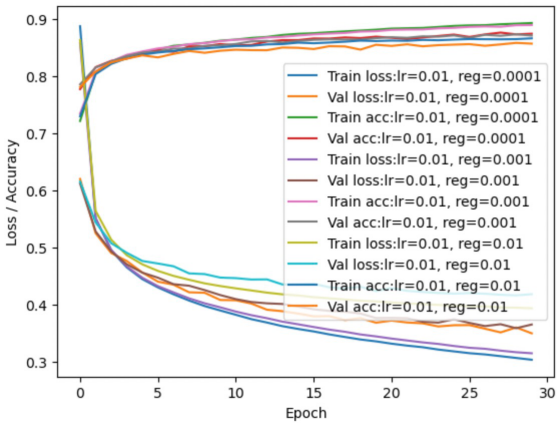


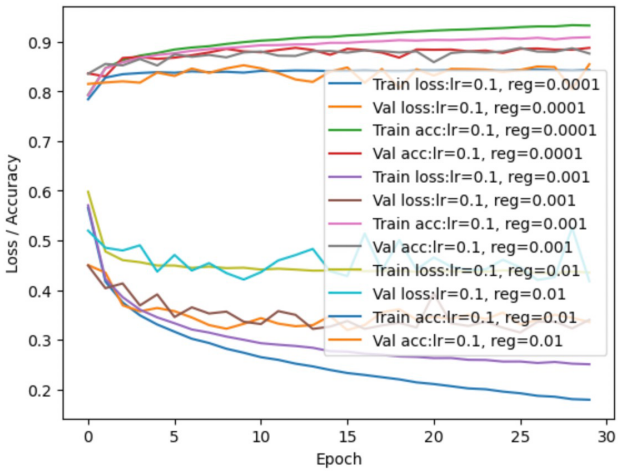




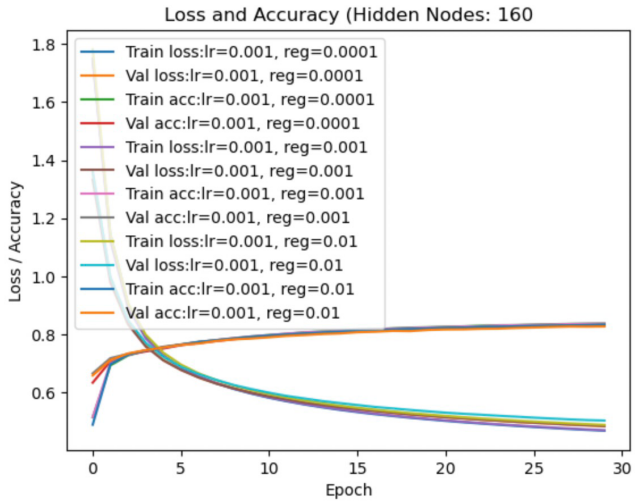
Hidden nodes = 80

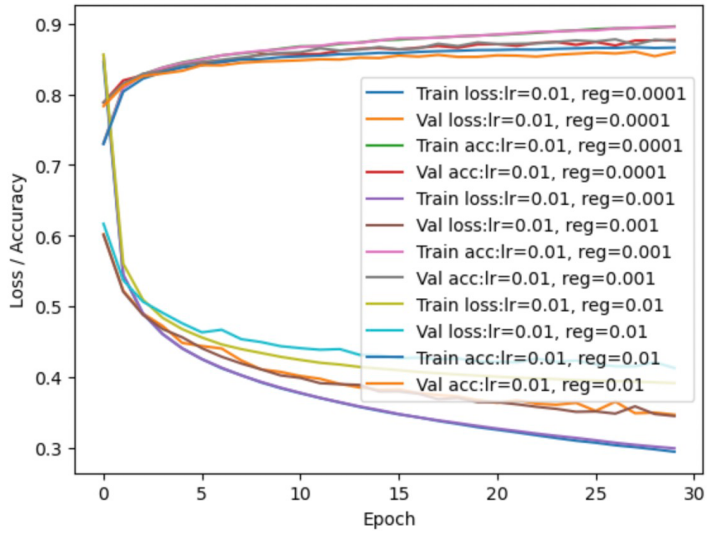


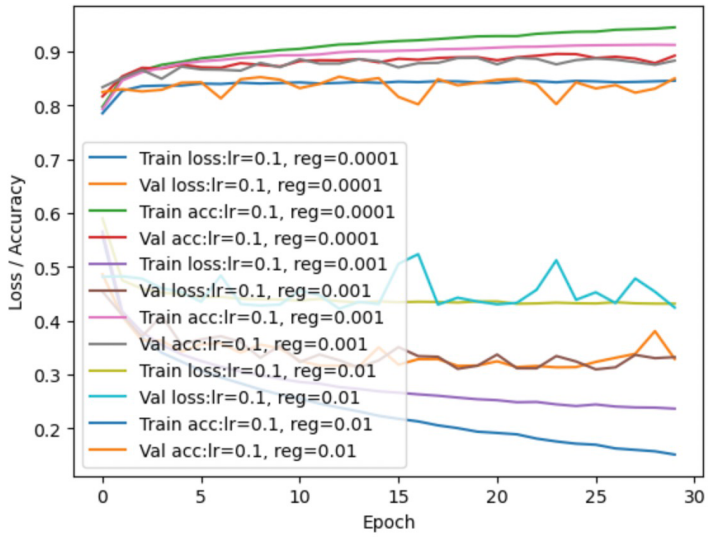




Hidden nodes = 160







The best hyper-parameters are hidden nodes = 160, learning rate = 0.01, weight decay rate = 0.001.

```
Best: 0.881729 using {'module__n_hidden': 160, 'optimizer__lr': 0.01, 'optimizer__weight_decay': 0.001}
```

(c) For the best hyper-parameters found in part (b), run 5 training runs out to 100 epochs. Report the best accuracy (over epochs) on val for each run - this is 5 numbers. Compute, mean, max, and std deviation for these 5 values.

```
The best accuracy (over epochs) on val for each run is [0.8916666666666667, 0.8924166666666666, 0.8909166666666667,
0.8925, 0.8920833333333333]
The mean, max, and std deviation for these 5 values is 0.8919166666666666 0.8925 0.0005797509043641774
```

(d) Take best model from part c (highest val accuracy) and evaluate on test. Report the test accuracy. Report the number of trainable parameters and all hyper-parameters used to obtain this final best model.

Test accuracy: 0.8838

The number of trainable parameters is 4： batchsize, hidden nodes, learning rate, weight decay.

All hyper-parameters is 14:

batch size : 16, 32, 64, 128

Hidden nodes: 40, 48, 80, 160

Learning rate: 0.001,0.01, 0.1

Weight decay: $1e{-}4,1e{-}3,1e{-}2$

2. **RBF network for 2D function approximation based on data**

(a) (i) $\alpha = \left((\Delta x_1 \Delta x_2)/M\right)^{1/2}, \Delta x_1 = 2 \; and \; \Delta x_2 = 2, so \; \alpha = 2/M^{(1/2)}$
(ii) $\gamma_d = 1/(2 * \sigma^2), \sigma = 5\alpha, so \; \gamma_d = 1/(2 * 25 * \alpha^2) = 1/(50 * \alpha^2). Insert \; the \; equation \; of \; \alpha \; we \; can \; get \; \gamma_d = M/200.$

(b) For comparison to the below systems, compute the RMSE of a trivial system that always outputs the sample mean value *y* on the training-set data.

The RMSE equals to 3.2035150890062902.

(c) (ii) The gamma range is [1.5e-01 1.5e+00 1.5e+01 1.5e+02 1.5e+03 1.5e+04], and the best gamma is 15.0 by using M = 3000.

(iii)Two tables

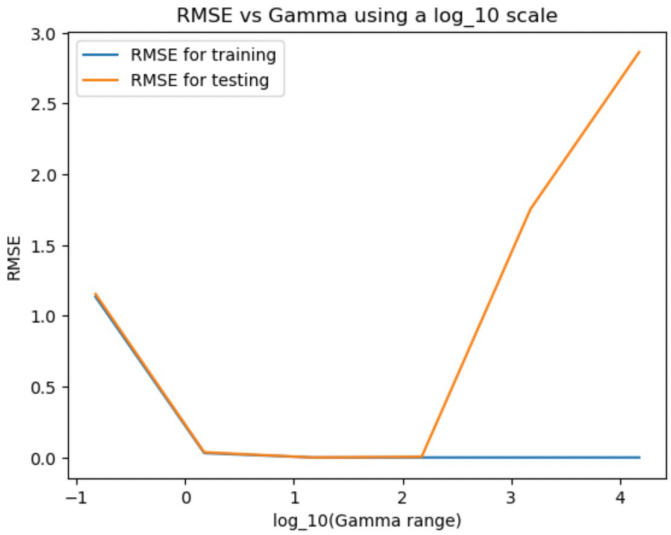
Gamma	1.5e-01	1.5e+00	1.5e+01	1.5e+02	1.5e+03	1.5e+04
Train mean	1.13475822e+00	3.11643033e-02	3.05638622e-08	3.58646810e-12	3.23542139e-14	2.06211531e-14
Val mean	1.15403078e+00	3.66475529e-02	7.73597943e-07	5.21484373e-03	1.75331839e+00	2.86271151e+00

Therefore, the best mean is 7.73597943e-07, and the gamma = 15.0

Gamma	1.5e-01	1.5e+00	1.5e+01	1.5e+02	1.5e+03	1.5e+04

Train std	7.33129502e-02	3.26677428e-03	7.91081453e-09	5.94505790e-13	5.27441228e-15	3.43880324e-16
Val std	7.89945707e-02	5.67047695e-03	5.96283883e-07	3.39567285e-03	3.53033549e-01	4.21701206e-02

(iv)



(d)(ii) The gamma range [3.0e-02 3.0e-01 3.0e+00 3.0e+01 3.0e+02 3.0e+03], and the best gamma is 30.0 while M = 600.

(iii)Two tables:
Mean values:

M\Gamma_d	0.01gamma_d	0.1gamma_d	1gamma_d	10gamma_d	100gamma_d	1000gamma_d
30\ 0.15	4.44080402e+00	1.95377118e+00	1.61148995e+00	1.60435959e+00	1.71059315e+00	2.83699548e+00
60 \ 0.3	2.32321219e+00	1.64184933e+00	1.19648210e+00	8.37349842e-01	1.31558598e+00	2.78517231e+00
100 \ 0.5	3.34377336e+00	1.45976893e+00	6.24635057e-01	2.23584693e-01	1.08798621e+00	2.83303000e+00
300 \ 1.5	3.35234670e+00	1.20668983e+00	4.53877066e-02	5.75480131e-03	1.01493664e+00	2.81418989e+00

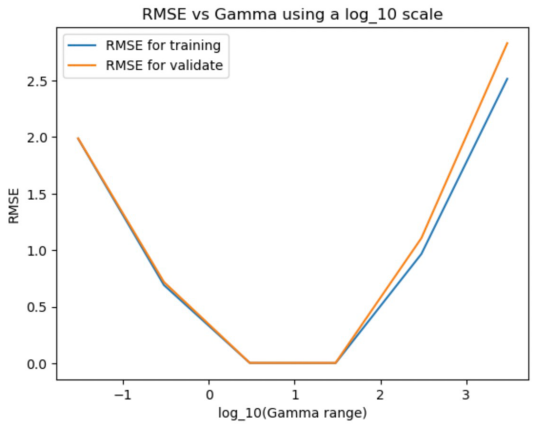
600 \ 3	2.28566428e+00	7.30729809e-01	3.41404257e-03	2.97683835e-03	1.04051149e+00	2.81379265e+00

Therefore, the best mean is 2.97683835e-03, and the gamma = 30.0

Std values:

M\Gamma_d	0.01gamma_d	0.1gamma_d	1gamma_d	10gamma_d	100gamma_d	1000gamma_d
30\ 0.15	1.70213297e+00	6.18803269e-02	8.47780020e-02	3.27101955e-02	1.23407094e-01	4.85061256e-02
60 \ 0.3	1.04444245e-01	7.89385778e-02	1.22902386e-01	6.58137238e-02	2.27517650e-01	4.97531895e-02
100 \ 0.5	7.92319193e-01	5.41196716e-02	3.16149611e-02	1.03746207e-01	1.54592931e-01	5.17729150e-02
300 \ 1.5	7.52317193e-01	1.01550063e-02	2.20893959e-03	1.01218304e-03	1.07575213e-01	8.15113919e-02
600 \ 3	4.77663186e-01	1.83054271e-02	4.17104497e-04	8.32066505e-04	1.72934319e-01	3.20417258e-02

(iv)



(v) What is the smallest value of M or K (and its associated γ) that would give RMSE at least a factor of 10 lower than the trivial system of (b)?

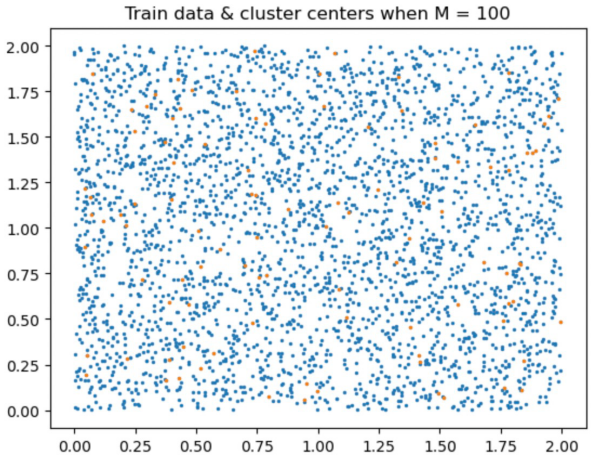
```
The smallest M is 100, the gamma = 5.0
```

What factor reduction in number of hidden units (dimensionality of the expanded feature space) from the original M=3000 in part (c) does this RCC model represent?

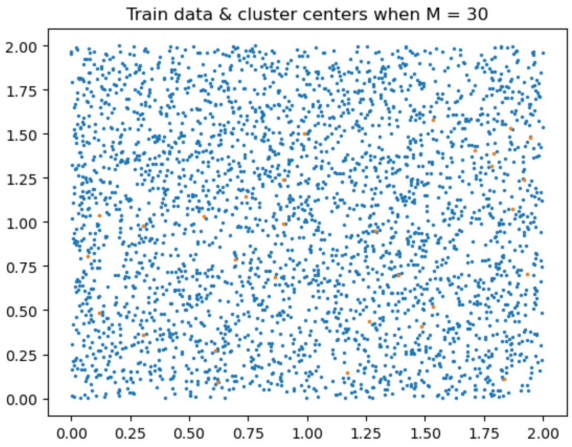
3000 / 100 = 30

(vi) Plot in original 2D feature space, the training data points x& and the cluster centers μ' for your best values of hyperparameters.

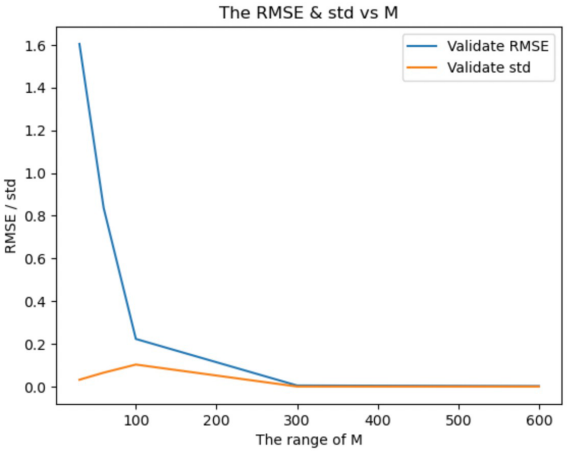
Then repeat the plots for your RCC model



And again for your lowest-complexity-model (M=30 or K=30).



(vii) Plot the validation error and its standard deviation vs. the second hyperparameter (*M* for (d), K for (e))



(e) Use K-means clustering to choose basis function centers for a given K
(ii)The gamma range is [3.0e-02 3.0e-01 3.0e+00 3.0e+01 3.0e+02 3.0e+03].
The best gamma is 30.0, the K = 600.

(iii)Two Tables:
Mean values:

K\Gamma a_d	0.01gamma _d	0.1gamma_ d	1gamma_d	10gamm a_d	100gamma_ d	1000gamma _d
30\ 0.15	3.16923859 e+00	2.33095153 e+00	1.51877461 e+00	1.63575 688e+00	1.73887304 e+00	2.86843057 e+00
60 \ 0.3	2.34332994 e+00	1.79674960 e+00	1.16123748 e+00	8.21192 046e-01	8.03159811 e-01	2.68114546 e+00

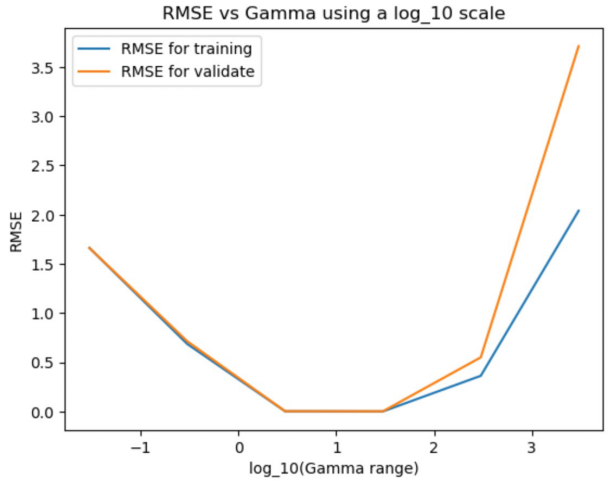
100 \ 0.5	4.08728352 e+00	1.45649433 e+00	5.65270843 e-01	1.70310 210e-01	4.75244767 e-01	2.71260080 e+00
300 \ 1.5	2.49456692 e+00	1.26578002 e+00	4.32094314 e-02	5.35701 314e-03	4.55467526 e-01	2.86491319 e+00
600 \ 3	1.66039080 e+00	7.12829752 e-01	3.37635740 e-03	2.38184 368e-03	5.50008578 e-01	3.70961585 e+00

The best mean is 2.38184368e-03, and gamma is 30.0.

Std values:

K\Gamma _d	0.01gamma_ d	0.1gamma _d	1gamma_d	10gamma_ d	100gamma _d	1000gamm a_d
30\ 0.15	2.13486901e -01	2.6259982 8e-01	2.3894231 2e-02	2.7645244 8e-02	1.4554940 3e-02	4.35274540 e-02
60 \ 0.3	1.05690485e -01	1.3130619 6e-01	1.7460003 2e-02	3.6577422 2e-02	1.0515434 3e-01	6.27359836 e-02
100 \ 0.5	1.35627427e +00	2.9883960 5e-02	1.0492157 5e-02	5.0314615 0e-03	3.5875475 0e-02	4.33283289 e-02
300 \ 1.5	7.10305241e -01	9.9809536 5e-02	1.5199918 7e-03	1.7999326 9e-03	1.2471030 9e-02	1.98708682 e-01
600 \ 3	6.37824637e -02	1.0102537 7e-02	4.0615684 0e-04	2.9546746 2e-04	3.1142862 5e-02	4.36848908 e-01

(iv)



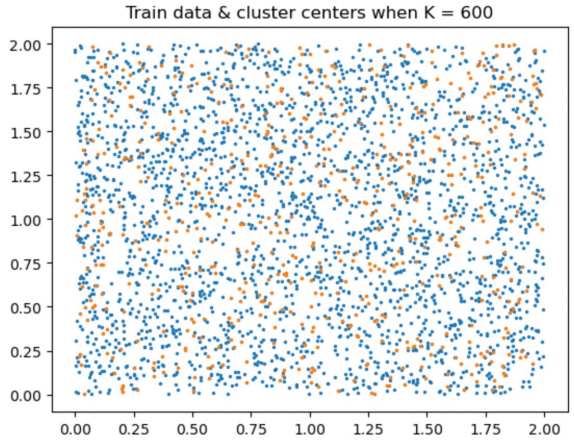
(v) What is the smallest value of M or K (and its associated γ) that would give RMSE at least a factor of 10 lower than the trivial system of (b)?

The smallest K is 100, the gamma = 5.0

What factor reduction in number of hidden units (dimensionality of the expanded feature space) from the original M=3000 in part (c) does this RCC model represent?

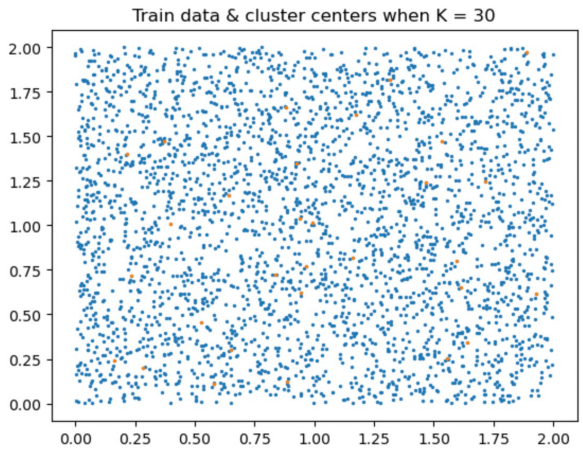
3000 / 100 = 30

(vi) Plot in original 2D feature space, the training data points x& and the cluster centers μ' for yo ur best values of hyperparameters.

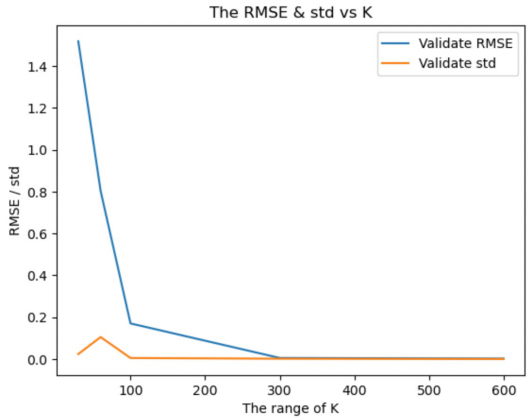


Then repeat the plots for your RCC model

And again for your lowest-complexity-model (M=30 or K=30).



(vii) Plot the validation error and its standard deviation vs. the second hyperparameter (M for (d), K for (e))



(f) Give the d.o.f. and number of constraints for the second layer (linear regressor) for each of (c), (d), and (e), for your best model of each; and again for your RCC model for each of (d), (e).

d.o.f. :
best model of (c) = M + 1 = 3000 + 1 = 3001， M = 3000
best model of (d) = M + 1 = 600 + 1 = 601, M = 600
best model of (e) = 601, K = 600
RCC model of (d) = 101, M = 100
RCC model of (e) = 101, K = 100

# of constraints:
Best model of (c) = 3000
Best model of (d) = 3000
Best model of (e) = 3000
RCC model of (d) = 3000
RCC model of (e) = 3000

(g) Run the best model from each of (c), (d), and (e); and run the RCC model of (d), (e), on your test set. Report the RMSE of each (5 models total).

The RMSE of best model from c is 2.5778315496757243e-07
The RMSE of best model from d is 0.0021838235900409615
The RMSE of best model from e is 0.0018462935462773137
The RMSE of RCC model from d is 0.17901963364551002
The RMSE of RCC model from e is 0.1661766513476608

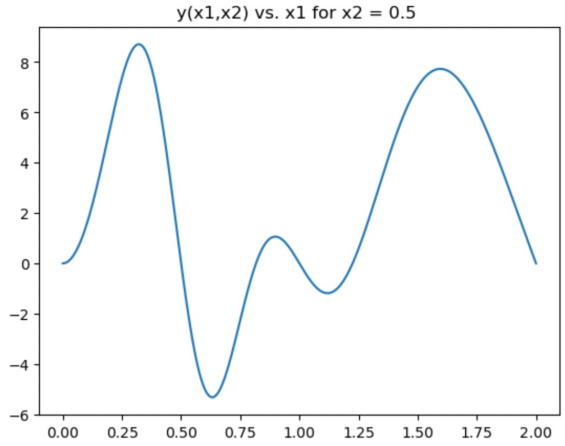
(h) Compare and comment on your results from (b)-(g). Specifically, observe and try to explain d ifferences in performance for different values of M (or K) and γ during model selection.

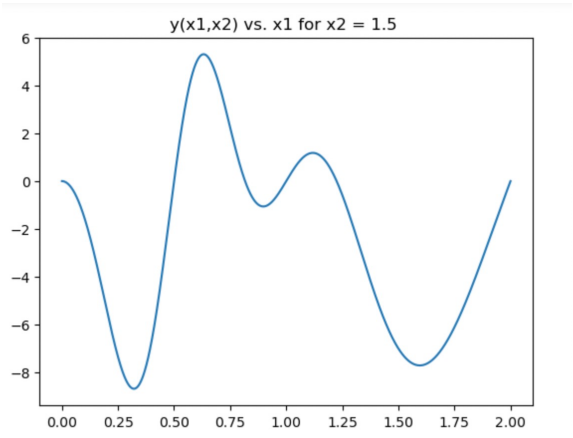
As we using the whole training set as the center points, and observing the RMSE for both train an d validation set we can find that, the RMSE remains low for all the training set but there exists ov erfitting according to the raise of RMSE for validation set with the change of γ.

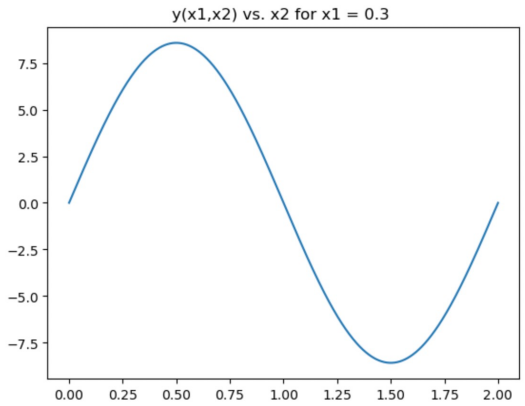
The best model from (d) has good performance for both of training set and validation set and has raise of RMSE for both train set and validation set when the γ changes. But comparing to the RC C model, we can find that when the centers has only the minimum number, this may cause the hi gh RMSE on test set.

The best model from (e) also has good performance for both of training set and validation set but has a slightly raise of RMSE for training set when the γ changes. But comparing to the RCC mod el, we can still find that there are only 100 centers which cause the high RMSE on the test set.

(i) To visuialize the target function, plot $y(x1, x2)$ vs. $x1$ for $x2 = 0.5$ and for $x2 = 1.5$. Also plot $y(x1, x2)$ vs. $x2$ for $x1 = 0.3$. (3 plots total)

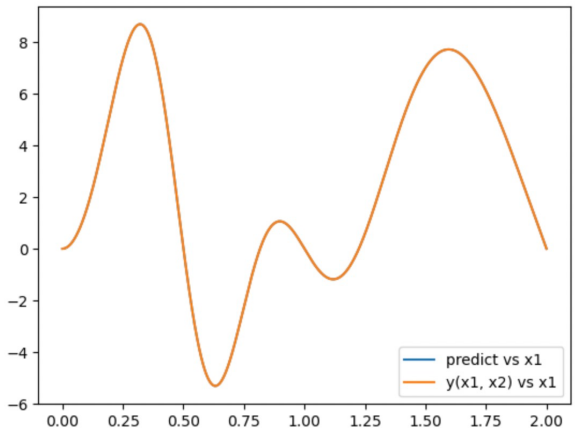


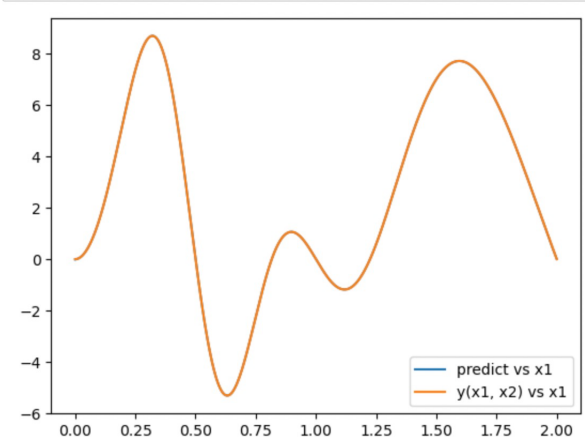




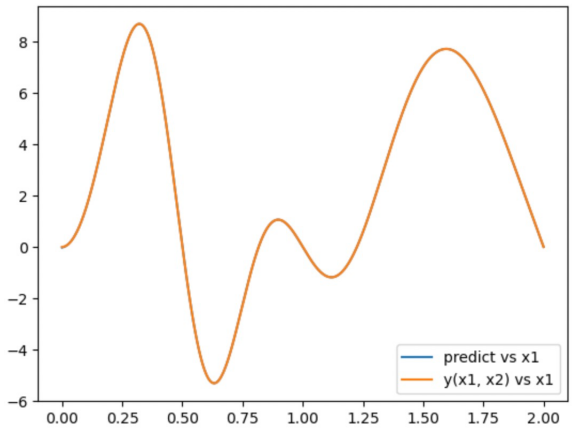
To compare the prediction with the target, plot $f[(x1, x2)$ and $y(x1, x2)$ vs. $x1$ for $x2 = 0.5$, for t he following cases: your best result from each of (c), (d), (e); and also your RCC model from eac h of (d), (e). (5 plots total for comparing prediction with target)

best result from each of (c), (d):

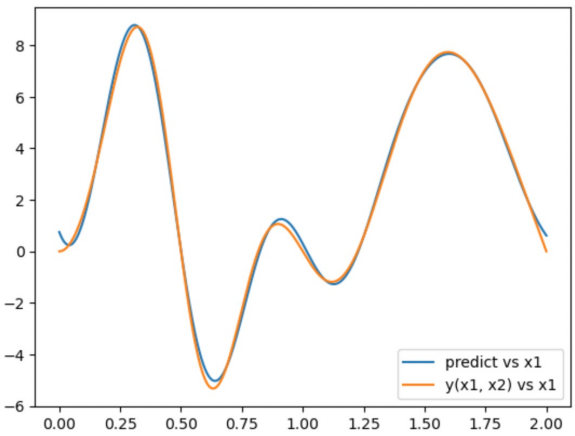


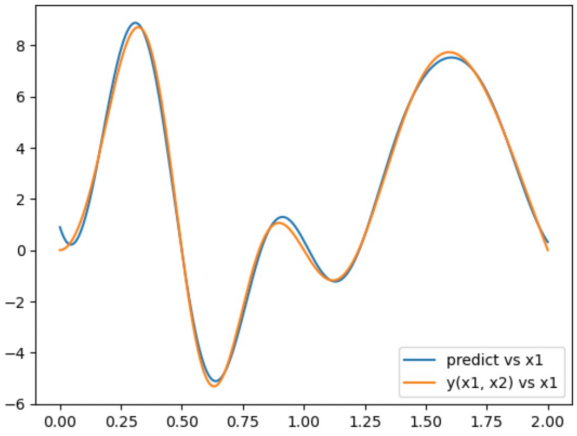


best result from each of (e):



RCC model from each of (d), (e):


predict vs x1
y(x1, x2) vs x1


predict vs x1
y(x1, x2) vs x1