CIS 520, Machine Learning, Fall 2020
Homework 6
Due: Monday, November 9th, 11:59pm
Submit to Gradescope

Chun Chang

1 Programming: Missing Data Imputation

For this question, refer to the Jupyter Notebook. You will be implementing different imputation techniques – the notebook will have detailed instructions.

1.1 Zero Imputation

Look for the code marked `#TODO` in the notebook and complete the code between the segments according to the instructions.

- Add the accuracy and the Frobenius norm in this report.

Frobenius Norm	51.66
Accuracy	60%

Table 1: Accuracy and Frobenius norm for Zero Imputation

- Add the code of the completed zeroImpute method.

```
# TODO
```

Listing 1: zeroImpute method

1.2 Mean Imputation

Look for the code marked `#TODO` in the notebook and complete the code between the segments according to the instructions.

- Add the accuracy and the Frobenius norm in this report.

Frobenius Norm	13.76
Accuracy	84%

Table 2: Accuracy and Frobenius norm for Mean Imputation

- Add the code of the completed meanImpute method.

```
# TODO
```

Listing 2: meanImpute method

1.3 Regression Imputation

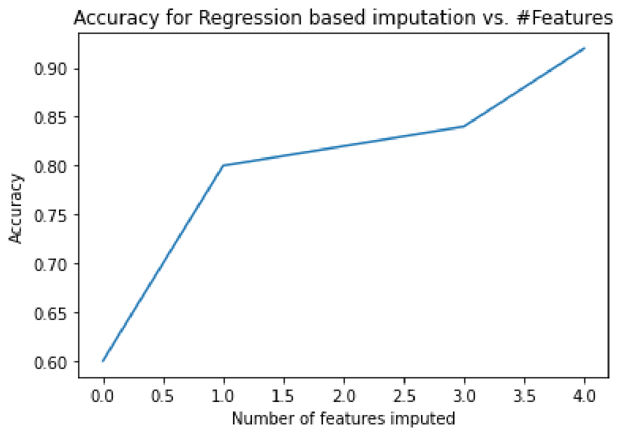
Look for the code marked `#TODO` in the notebook and complete the code between the segments according to the instructions.

- Complete the following table.

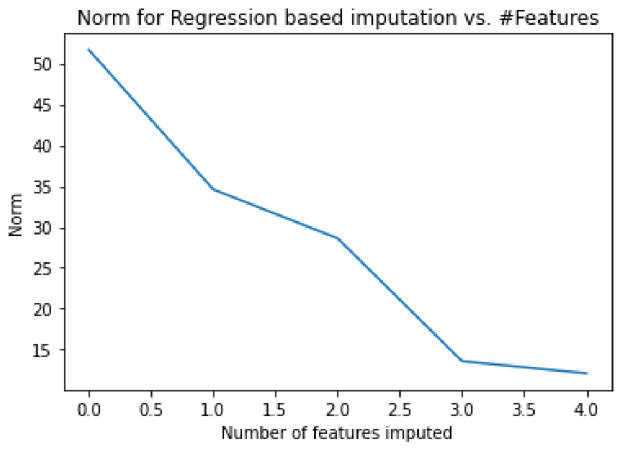
Epoch	Frobenius Norm	Accuracy
After Base Imputation	51.66	60 %
1	12.1	92%
2	7.98	92%
3	8.13	96%
4	7.52	96%
5	7.83	98%

Table 3: Accuracy and Frobenius norm for Zero Imputation

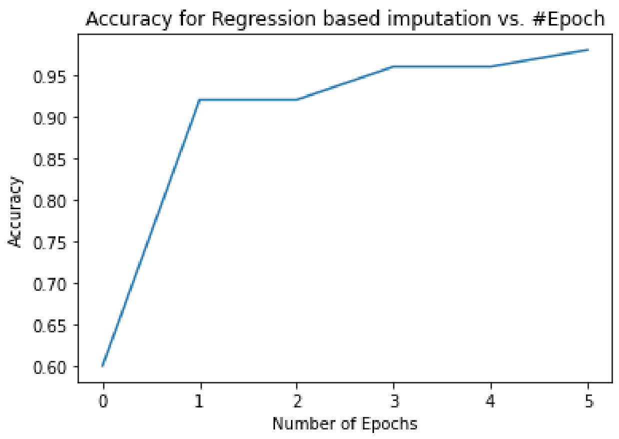
- Plot for Accuracy for Regression based imputation vs. Number of Features imputed



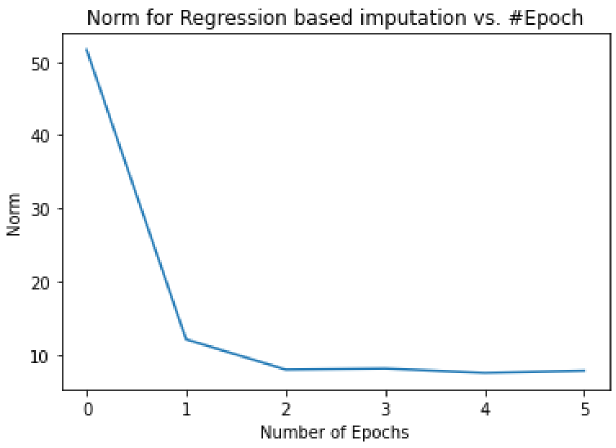
- Plot for Norm for Regression based imputation vs. Number of Features imputed



- Plot for Accuracy for Regression based imputation vs Number of Epochs



- Plot for Norm for Regression based imputation vs. Number of Epochs



- Add the code of the completed regressedImpute method.

```
# TODO
```

Listing 3: regressedImpute method

1.4 Follow Up Questions

1. Which is the best of the three imputation methods? Why (briefly)?

   The linear regression imputed data works best. It might be the features are correlated, and closed to linear distribution.

2. Could you potentially do a better job of imputation if you also used y vales as well as x's? How might that help? If using the y's for imputation helps, why do people mostly not use them?

   Hint: think through the whole process of model estimation and use.

   We want to use the imputation model to impute the missing data in the test data set as well, which we don't know what the labels are. So we could only use the X for imputation.

3. Describe the trend for the accuracy and the norm as impute more features for regression imputation. Give a plausible explanation behind the trend.

   The norm goes down and the accuracy goes up as more missing data gets imputed by linear regression.

   It might be the features of data are correlated, and the label are also correlated with the features. So with the linear regression imputed data, the difference between imputed data and the original data get small and we get better accuracy.

4. Describe the trend for the accuracy and the norm as re-impute again for several epochs for regression imputation. Give a plausible explanation behind the trend.

   As we have more linear regression imputation, the data tends to be overfitted and the accuracy goes to 100 % but the norm also increased because we force the missing data as linear as much as possible.

2 Programming: K-Means Analysis

In this assignment, we will implement the K-means algorithm and perform it on a breast cancer data set. Please follow the attached Python notebook to write and plot the functions.

2.1 Part 1: K-means iteration function

Write the K-means iteration function as is described in the notebook. Each iteration takes the data and current centroid values for each class as parameters, and returns both updated centroid values along with a list of labels telling which class each datapoint belongs to.

1. Fill in the table below by reporting the resulting cluster labels and resulting centroids from running the iteration function on the given parameters.

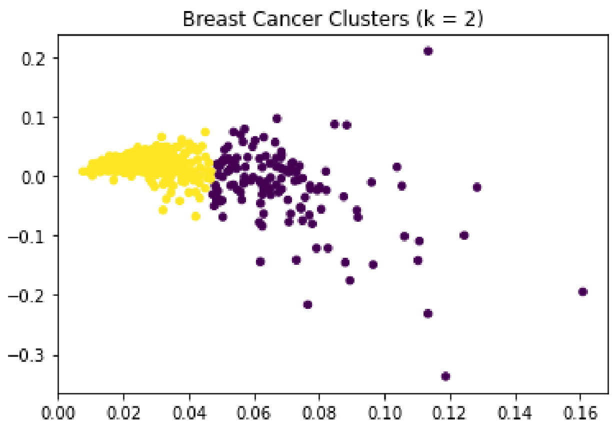
X	**Initial Centroids**	**Resulting Cluster Labels**	**Resulting Centroids**
[[1], [2], [10], [12]]	[1, 2]	[0,1,1,1]	[[1],[8]]
[[1], [2], [10], [12]]	[1, 8]	[0,0,1,1]	[[1.5],[11]]
[[1], [2], [10], [12]]	[2, 2]	[0,0,0,0]	[[6],[2]]
[[0,5,0],[0,5,0],[0,4,3],[0,3,4]]	[[2.5,0,0],[-2.5,0,0]]	[0,0,0,0]	[[0.0, 4.25, 1.75],[-2.5, 0.0, 0.0]]

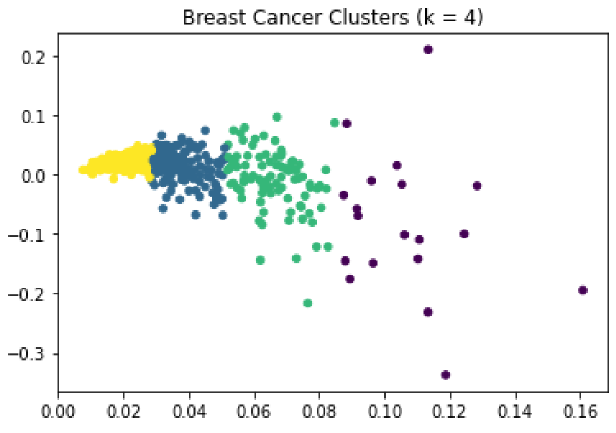
2. If an iteration of the k-means algorithm returns less than K classes, what might that indicate about the data?

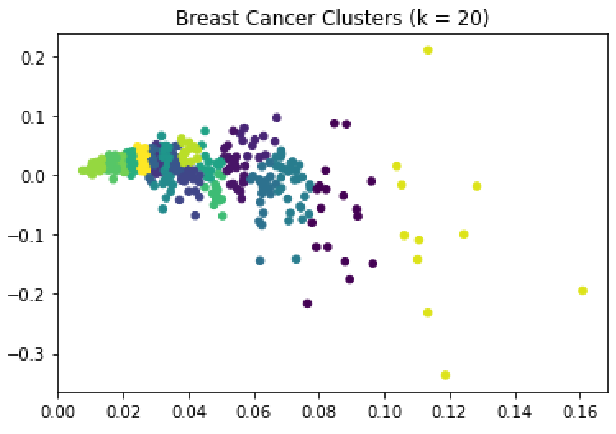
2.2 Part 2: Putting the algorithm together

Now write the whole K-means function using your iteration function, as is described in the notebook. The function takes in the data set with the number of classes and returns the final centroid values, the final list of labels telling which class each data point belongs to, and the number of K-means iterations.

1. Put your 3 sanity-check graphs here.







2. Write down how many iterations it took for your k-means to converge for each of the sanity-check graphs. How does this number compare with what you expected it to be? How does the number of iterations seem to vary proportional to k?

   k = 2, iter = 9,
   k = 4, iter = 23,
   k = 20, iter = 20

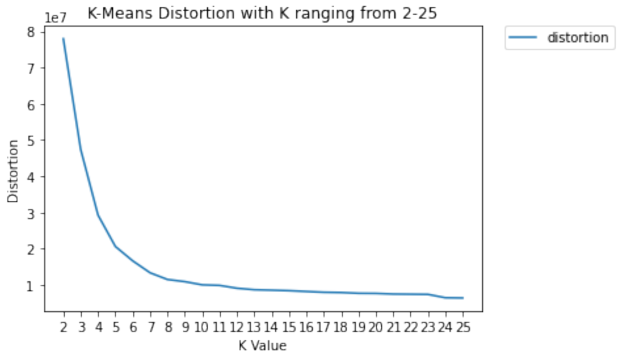
   There is no strong relationship between number of iteration and the number of the clusters.

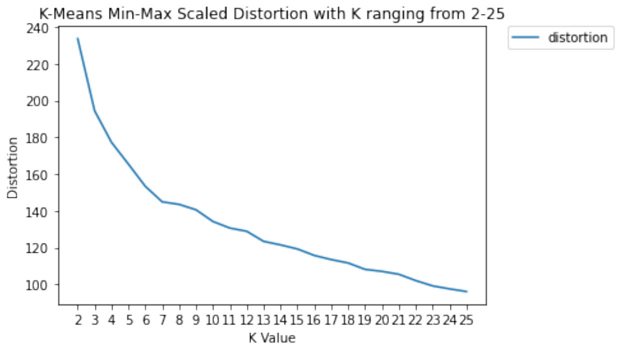
The number of iteration might depend of the nature of the data

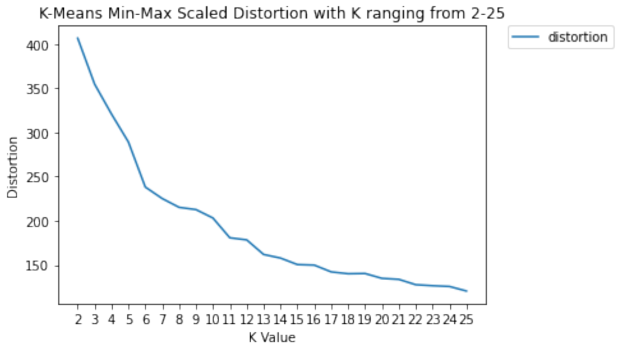
2.3 Part 3: Choosing the value of K

Write the test_cluster_size function using your k-means function, as is described in the notebook. The function takes in the data set with the number of classes and returns a list of scores where each score is the distortion for that value of k.

1. Using the breast cancer data set, record your labeled plots for distortion, min-max scaled distortion, and log scaled distortion here







2. Why can't we hold out some of the data in a validation set, and choose the value of k which minimizes a cross-validation error like we have in the past for algorithms like linear regression?

   The holdout dataset will cause the movements of the centroids so we could not use cross-validation to pick the number of clusters.

3 Performance Measures for Face Detection in Images

Consider a face detection problem, where the goal is to build a system that can automatically detect faces in images. Two research groups develop systems for this problem using slightly different approaches. In both cases, the central component of the system is a binary classifier which, when applied to a 24×24 image, decides whether or not it is a face. The two groups train their classifiers using different learning algorithms. Moreover, when given a new image, they also apply their classifiers in slightly different ways: group A tests 24×24 regions of the image taking strides of size 2 (so, for example, for a 100×100 image, $(39)^2$ regions would be tested); group B tests 24×24 regions of the image taking strides of size 5 (so here, for a 100×100 image, only $(16)^2$ regions would be tested).[1] On a standard benchmark suite of test images that contains 300 faces altogether, the two groups have the following performances (assume the regions tested by both systems include all the 300 true face regions):

w **Research group**	**Number of regions tested**	**Number of faces detected correctly**	**Number of non-face regions detected as faces**
A	20,000	280	100
B	12,500	270	60

1. Based on the above numbers, calculate the TPR (recall), TNR, and precision of each group's system as tested. Also calculate the resulting geometric mean (GM) and F_1 measures for each system. If you were to select a system based on the GM measure, which system would you choose? Would your choice change if you were to select a system based on the F_1 measure? (Note, the geometric mean (GM) is defined as $\sqrt{TPR \times TNR}$.)

A	**Predicted Positive**	**Predicted Negative**
Positive	280	20
Negative	100	19600

B	**Predicted Positive**	**Predicted Negative**
Positive	270	30
Negative	60	12140

$$\begin{aligned}
TPR_A &= \frac{280}{300} \\
TNR_A &= \frac{19600}{19700} \\
GM_A &= \sqrt{\frac{280}{300} * \frac{19600}{19700}} \\
&= 0.963 \\
Precision &= \frac{280}{380} \\
Recall &= \frac{280}{300} \\
F_1^A &= \frac{2(Pr * Re)}{(Pr + Re)} \\
&= 0.823
\end{aligned}$$

[1]In practice, the 24×24 classifier would also be applied to multiple scaled versions of the input image; we ignore this issue here for simplicity.

$$\begin{aligned}
TPR_B &= \frac{270}{300} \\
TNR_B &= \frac{12140}{12200} \\
GM_B &= \sqrt{\frac{270}{300} * \frac{12140}{12200}} \\
&= 0.946 \\
Precision &= \frac{270}{330} \\
Recall &= \frac{270}{300} \\
F_1^B &= \frac{2(Pr * Re)}{(Pr + Re)} \\
&= 0.857
\end{aligned}$$

For GM, I chose method A.
For F_1 I chose B.

2. Which performance measure would be more suitable for this problem – the GM measure or the F_1 measure? Why?

   F_1 only takes the predicted true into consideration.
   And GM takes the predicted true and predicted false into consideration.

   So for the system we have, we want the system to identify the true face region and none-face region. Therefore the GM is better choice for the evaluation

3. Another way to determine which method to choose would be to look at the ROC curve. Because you are given instances of different algorithms, not the algorithm itself, each method corresponds to a *point* on the TPR vs. FPR graph, not a curve.

   What is the Euclidean distance from each instance to the 0-error (perfect classification) point $(0, 1)$? Based on this metric, which method would you choose?

$$\begin{aligned}
FPR_A &= \frac{100}{19700} \\
FPR_B &= \frac{60}{12200} \\
Eu_A((FPR_A, TPR_A), (0,1)) &= Eu_A((\frac{100}{19700}, \frac{280}{300}), (0,1)) = 0.0668 \\
Eu_B((FPR_B, TPR_B), (0,1)) &= Eu_B((\frac{60}{12200}, \frac{270}{300}), (0,1)) = 0.1
\end{aligned}$$

4. Now assume that there is a third group C which trains their classifier using a newly discovered learning algorithm. Some of the resulting statistical measures are described below:

Research group	**TPR**	**TNR**	**FPR**	**FNR**
C	0.95	0.990	0.01	0.05

(a) Suppose you worked for a social media platform, where you want to identify as many faces as possible for your photo tagging systems (prioritize recall). Would you prefer the algorithm created by group C over the algorithms created by groups A and B?

The goal is to identify as many as faces as possible so the chosen method shall have the highest rate of identification of the faces over the given true positive data.

C is chosen.

(b) Now suppose you worked for law enforcement, where every face detected will be checked against a criminal database, which is an expensive operation. You'd like maximize specificity (1 - FPR) in this case to avoid unnecessary costs. Would you prefer the algorithm created by group C over the algorithms created by groups A and B?

$$SPE_A = 1 - \frac{100}{19700} = 0.9949$$
$$SPE_B = 1 - \frac{60}{12200} = 0.995$$
$$SPE_C = 1 - 0.01 = 0.99$$

To minimize the cost of incorrect identification of the faces, the method shall have the lowest FNR to avoid wrong identifications

B is chosen

4 EM Algorithm with Red and Blue Coins

Your friend has two coins: a red coin and a blue coin, with biases p_r and p_b, respectively (i.e. the red coin comes up heads with probability p_r, and the blue coin does so with probability p_b). She also has an inherent preference π for the red coin. She conducts a sequence of m coin tosses: for each toss, she first picks either the red coin with probability π or the blue coin with probability $1 - \pi$, and then tosses the corresponding coin; the process for each toss is carried out independently of all other tosses. You don't know which coin was used on each toss; all you are told are the outcomes of the m tosses (heads or tails). In particular, for each toss i, define a random variable X_i as

$$X_i = \begin{cases} 1 & \text{if the } i\text{-th toss results in heads} \\ 0 & \text{otherwise.} \end{cases}$$

Then the data you see are the values $x_1, \ldots, x_m$ taken by these m random variables. Based on this data, you want to estimate the parameters $\theta = (\pi, p_r, p_b)$. To help with this, for each toss i, define a latent (unobserved) random variable Z_i as follows:

$$Z_i = \begin{cases} 1 & \text{if the } i\text{-th toss used the red coin} \\ 0 & \text{otherwise.} \end{cases}$$

1. Let X be a random variable denoting the outcome of a coin toss according to the process described above, and let Z be the corresponding latent random variable indicating which coin was used, also as described above (both X and Z take values in $\{0, 1\}$ as above). Write an expression for the joint distribution of X and Z. Give your answer in the form

$$p(x, z; \theta) = \sum p(z)p(x|z) = ((1-\pi)P_b^x(1-P_b)^{1-x})^{1-z} * (\pi P_r^x(1-P_r)^{1-x})^z$$

2. Write an expression for the complete-data log-likelihood, $\ln Ł_c(\theta) = \sum_{i=1}^m \ln p(x_i, z_i;\, \theta)$.

$$\begin{aligned}
L_c(\theta) &= \sum *ln(\left[(1-\pi)*(P_b)^x(1-P_b)^{1-x}\right]^{1-z} * \left[\pi * (P_r)^x(1-P_r)^{1-x}\right]^z) \\
&= \sum(1-z)ln((1-\pi)*(P_b)^x(1-P_b)^{1-x}) + \sum(z)ln(\pi*(P_r)^x(1-P_r)^{1-x}) \\
&= \sum(1-z)ln(1-\pi) + \sum(1-z)xln(P_b) + \sum(1-z)(1-x)ln(1-P_b) \\
&+ \sum(z)ln(\pi) + \sum(z)xln(P_r) + \sum(z)(1-x)ln(1-P_r)
\end{aligned}$$

3. Suppose you knew the values z_i taken by the latent variables Z_i. What would be the maximum-likelihood parameter estimates $\hat{\theta}$? Give expressions for $\hat{\pi}$, $\hat{p}_r$, and $\hat{p}_b$ (in terms of x_i and z_i).

$$\begin{aligned}
\hat{\theta} &= argmax(Ł_c(\theta)) \\
\hat{\theta} &:= \partial\frac{Ł_c(\theta)}{\partial\theta} = 0 \\
\because \partial\frac{Ł_c(\theta)}{\partial\pi} &= -\sum\frac{(1-z)}{(1-\pi)} + \sum\frac{z}{\pi} \\
\therefore \hat{\pi} &= \frac{\sum z}{m} \\
\because \partial\frac{Ł_c(\theta)}{\partial P_r} &= -\sum\frac{(1-x)z}{(1-P_r)} + \sum\frac{zx}{P_r} \\
\therefore \hat{P_r} &= \frac{\sum zx}{\sum z} \\
\because \partial\frac{Ł_c(\theta)}{\partial P_b} &= -\sum\frac{(1-z)(1-x)}{(1-P_b)} + \sum\frac{(1-z)x}{P_b} \\
\therefore \hat{P_b} &= \frac{\sum(1-z)x}{\sum(1-z)}
\end{aligned}$$

4. In the absence of knowledge of z_i, one possibility for estimating θ is to use the EM algorithm. Recall that the algorithm starts with some initial parameter estimates θ^0, and then on each iteration t, performs an E-step followed by an M-step. Let θ^t denote the parameter estimates at the start of iteration t. In the E-step, for each toss i, the algorithm requires computing the posterior distribution of the latent variable Z_i under the current parameters θ^t. Calculate the posterior probability ¶$(Z_i = 1 \,|\, X_i = x_i;\, \theta^t)$. *(Hint: Use Bayes' rule.)*

$$\begin{aligned}
P(Z_i = 1 \,|\, X_i = x_i;\, \theta^t) &= \frac{P(X_i, Z_i = 1; \theta^t)}{P(X_i)} \\
&= \frac{P(X_i, Z_i = 1; \theta^t)}{\sum_z P(X_i, Z_i)} \\
&= \frac{\pi_t P_{r,t}^x(1-P_{r,t})^{1-x}}{\pi_t P_{r,t}^x(1-P_{r,t})^{1-x} + (1-\pi_t)P_{b,t}^x(1-P_{b,t})^{1-x}}
\end{aligned}$$

5. For each toss i, denote the posterior probability computed in part (d) above by γ_i^t (so that $\gamma_i^t = P(Z_i = 1 \,|\, X_i = x_i;\, \theta^t)$). Then the expected complete-data log-likelihood with respect to these posterior distributions is

$$\sum_{i=1}^m \left(\gamma_i^t \cdot \ln p(x_i, 1;\, \theta) + (1-\gamma_i^t) \cdot \ln p(x_i, 0;\, \theta)\right).$$

The M-step of the EM algorithm requires finding parameters θ^{t+1} that maximize this expected complete-data log-likelihood.

Determine the updated parameters θ^{t+1}. Give expressions for π^{t+1}, p_r^{t+1}, and p_b^{t+1} (in terms of x_i and γ_i^t).

$$\sum_{i=1}^{m}\Big(\gamma_i^t \cdot \ln p(x_i, 1;\, \theta) + (1-\gamma_i^t) \cdot \ln p(x_i, 0;\, \theta)\Big)$$

the above expression is equivalent to the expression in part3,

$$= \sum (1-z) ln((1-\pi) * (P_b)^x (1-P_b)^{1-x}) + \sum (z) ln(\pi * (P_r)^x (1-P_r)^{1-x})$$

where

$$\begin{aligned} z &= \gamma_i^t \\ 1-z &= 1-\gamma_i^t \\ &\therefore \\ \partial\frac{Ł_c(\theta)}{\partial\pi} &= \hat{\pi}^{t+1} = \frac{\sum \gamma_i^T}{m} \\ \partial\frac{Ł_c(\theta)}{\partial P_r} &= \hat{P_r}^{t+1} = \frac{\sum \gamma_i^t x}{\sum \gamma_i^t} \\ \partial\frac{Ł_c(\theta)}{\partial P_b} &= \hat{P_b}^{t+1} = \frac{\sum (1-\gamma_i^t) x}{\sum (1-\gamma_i^t)} \end{aligned}$$