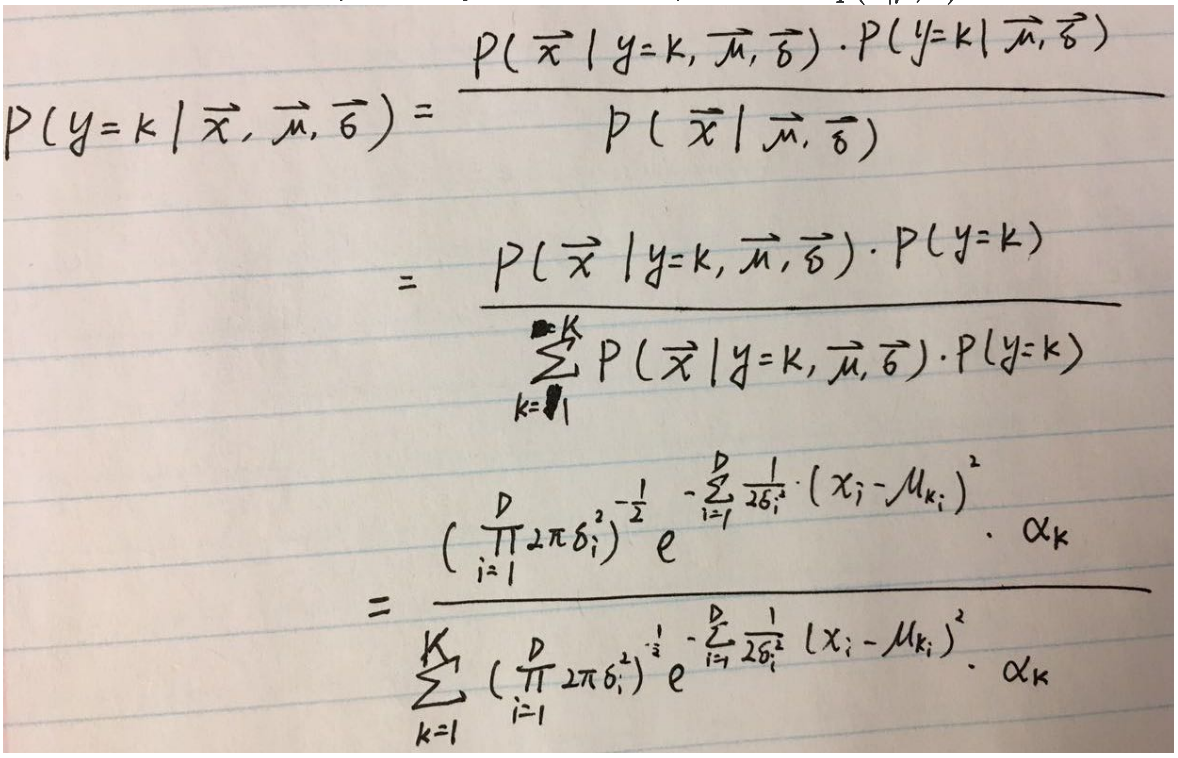
Report on HW2

ZHANG Qianhao / 1004654377

1 Class-Conditional Gaussians (4%)

> In this question, you will derive the maximum likelihood estimates for class-conditional Gaussians with independent features (diagonal covariance matrices), i.e. Gaussian Naive Bayes, with shared variances. Start with the following generative model for a discrete class label $y \in (1, 2, ..., K)$ and a real valued vector of d features $\mathbf{x} = (x_1, x_2, \ldots, x_d)$

1. Use Bayes' rule to derive an expression for p(y = k | $\mathbf{x}$, μ, σ).
   [Hint: Use the law of total probability to derive an expression for $p(\mathbf{x}|\mu,\sigma)$.]

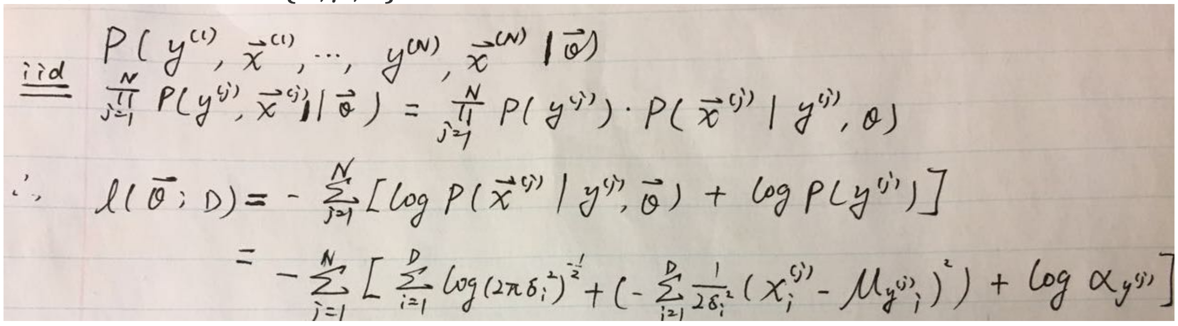


$$P(y=k|\vec{x},\vec{\mu},\vec{\sigma}) = \frac{P(\vec{x}|y=k,\vec{\mu},\vec{\sigma})\cdot P(y=k|\vec{\mu},\vec{\sigma})}{P(\vec{x}|\vec{\mu},\vec{\sigma})}$$

$$= \frac{P(\vec{x}|y=k,\vec{\mu},\vec{\sigma})\cdot P(y=k)}{\sum_{k=1}^{K} P(\vec{x}|y=k,\vec{\mu},\vec{\sigma})\cdot P(y=k)}$$

$$= \frac{\left(\prod_{i=1}^{D} 2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{-\sum_{i=1}^{D}\frac{1}{2\sigma_i^2}\cdot(x_i-\mu_{k_i})^2}\cdot\alpha_k}{\sum_{k=1}^{K}\left(\prod_{i=1}^{D} 2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{-\sum_{i=1}^{D}\frac{1}{2\sigma_i^2}(x_i-\mu_{k_i})^2}\cdot\alpha_k}$$

2. Write down an expression for the negative likelihood function (NLL)
   $l(\theta; D) = -log(p(y^{(1)}, \mathbf{x}^{(1)}, \ldots, y^{(N)}, \mathbf{x}^{(N)}|\theta)) \ \ldots \ (3)$
   of a particular dataset $D = \{(y^{(1)}, \mathbf{x}^{(1)}), \ldots, (y^{(N)}, \mathbf{x}^{(N)})\}$
   with parameters $\theta = \{\alpha, \mu, \sigma\}$ (Assume that the data are iid.)

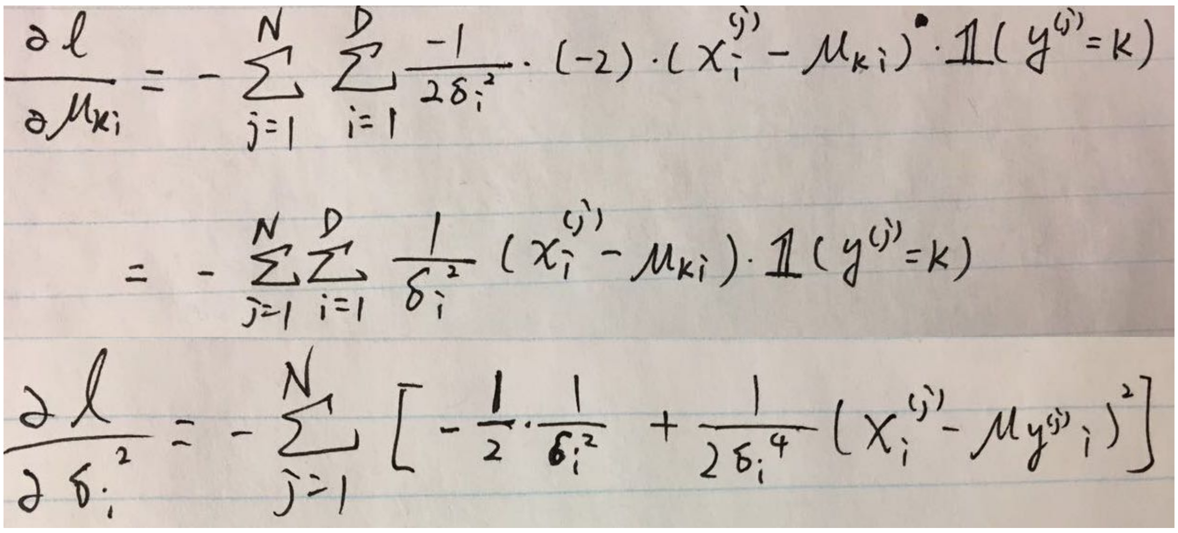


$$P(y^{(1)},\vec{x}^{(1)},\ldots,y^{(N)},\vec{x}^{(N)}|\vec{\theta}) \overset{iid}{=} \prod_{j=1}^{N} P(y^{(j)},\vec{x}^{(j)}|\vec{\theta}) = \prod_{j=1}^{N} P(y^{(j)})\cdot P(\vec{x}^{(j)}|y^{(j)},\theta)$$

$$\therefore\ l(\vec{\theta};D) = -\sum_{j=1}^{N}\left[\log P(\vec{x}^{(j)}|y^{(j)},\vec{\theta}) + \log P(y^{(j)})\right]$$

$$= -\sum_{j=1}^{N}\left[\sum_{i=1}^{D}\log(2\pi\sigma_i^2)^{-\frac{1}{2}} + \left(-\sum_{i=1}^{D}\frac{1}{2\sigma_i^2}(x_i^{(j)}-\mu_{y^{(j)}i})^2\right) + \log\alpha_{y^{(j)}}\right]$$

3. Take partial derivatives of the likelihood with respect to each of the parameters μ_{ki} and with respect to the shared variances σ_i^2.

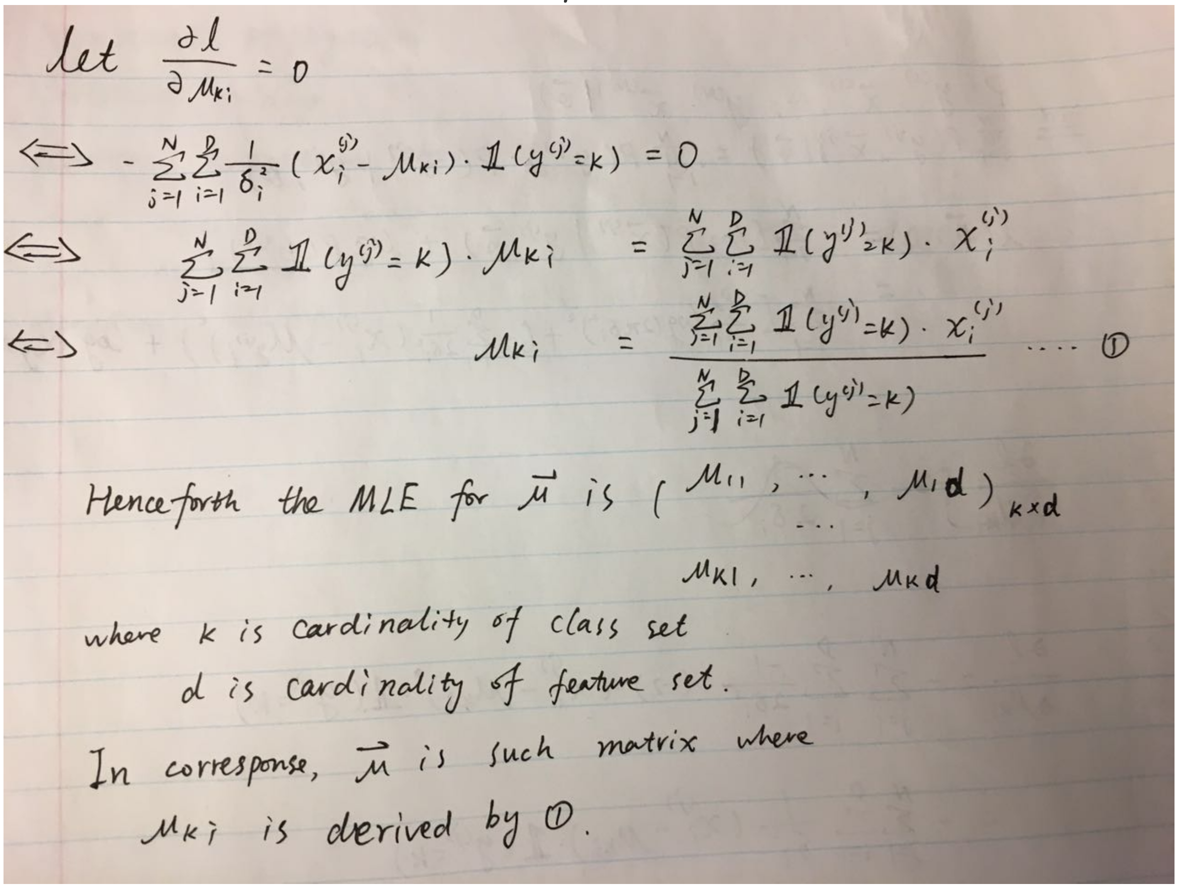


$$\frac{\partial l}{\partial \mu_{ki}} = -\sum_{j=1}^{N}\sum_{i=1}^{D} \frac{-1}{2\delta_i^2}\cdot(-2)\cdot(x_i^{(j)}-\mu_{ki})\cdot \mathbb{1}(y^{(j)}=k)$$

$$= -\sum_{j=1}^{N}\sum_{i=1}^{D} \frac{1}{\delta_i^2}(x_i^{(j)}-\mu_{ki})\cdot\mathbb{1}(y^{(j)}=k)$$

$$\frac{\partial l}{\partial \delta_i^2} = -\sum_{j=1}^{N}\left[-\frac{1}{2}\cdot\frac{1}{\delta_i^2}+\frac{1}{2\delta_i^4}(x_i^{(j)}-\mu_{y^{(j)}i})^2\right]$$

4. Find the maximum likelihood estimates for μ and σ.



let $\frac{\partial l}{\partial \mu_{ki}} = 0$

$$\Leftrightarrow -\sum_{j=1}^{N}\sum_{i=1}^{D}\frac{1}{\delta_i^2}(x_i^{(j)}-\mu_{ki})\cdot\mathbb{1}(y^{(j)}=k) = 0$$

$$\Leftrightarrow \sum_{j=1}^{N}\sum_{i=1}^{D}\mathbb{1}(y^{(j)}=k)\cdot\mu_{ki} = \sum_{j=1}^{N}\sum_{i=1}^{D}\mathbb{1}(y^{(j)}=k)\cdot x_i^{(j)}$$

$$\Leftrightarrow \mu_{ki} = \frac{\sum_{j=1}^{N}\sum_{i=1}^{D}\mathbb{1}(y^{(j)}=k)\cdot x_i^{(j)}}{\sum_{j=1}^{N}\sum_{i=1}^{D}\mathbb{1}(y^{(j)}=k)} \quad \cdots\cdots \text{①}$$

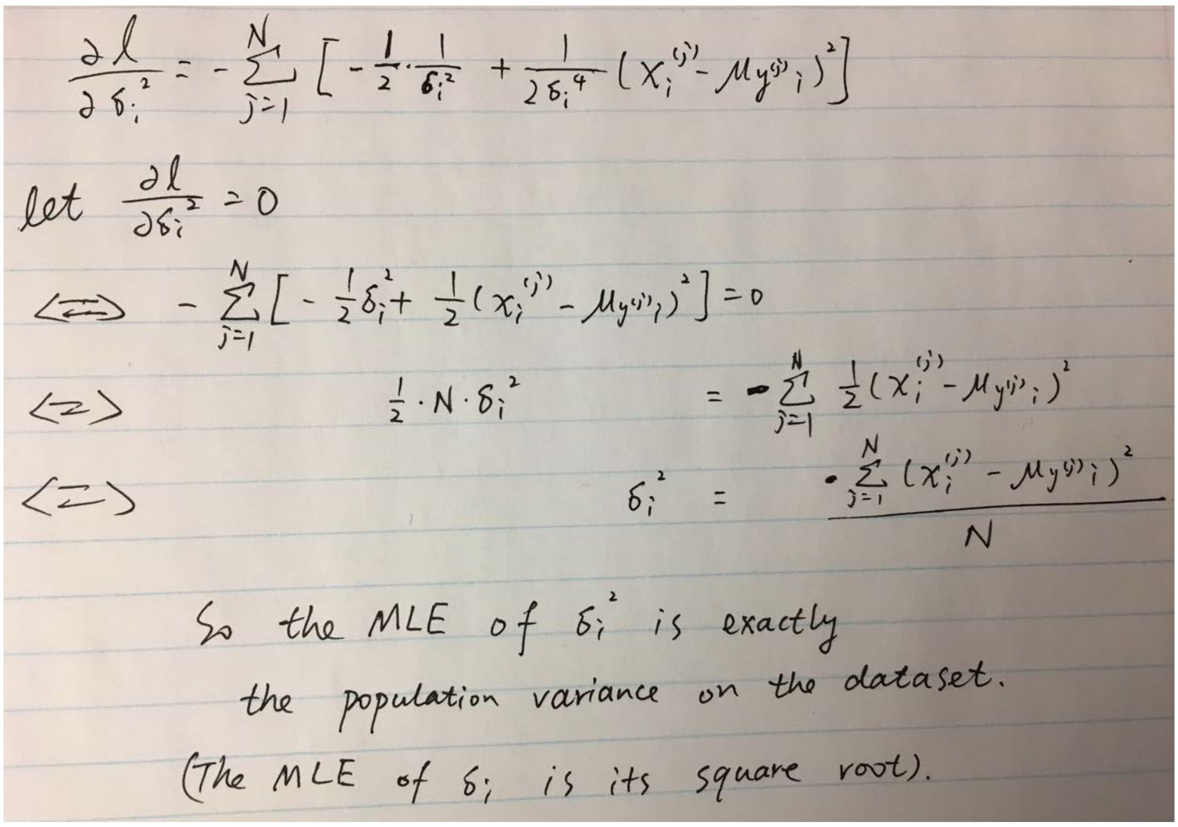
Henceforth the MLE for $\vec{\mu}$ is $\begin{pmatrix}\mu_{11}, & \cdots, & \mu_{1d}\\ & \cdots & \\ \mu_{K1}, & \cdots, & \mu_{Kd}\end{pmatrix}_{k\times d}$

where k is cardinality of class set

d is cardinality of feature set.

In corresponse, $\vec{\mu}$ is such matrix where

μ_{ki} is derived by ①.



$$\frac{\partial l}{\partial \delta_i^2} = -\sum_{j=1}^{N}\left[-\frac{1}{2}\cdot\frac{1}{\delta_i^2} + \frac{1}{2\delta_i^4}\left(x_i^{(j)} - \mu_{y^{(j)},i}\right)^2\right]$$

let $\frac{\partial l}{\partial \delta_i^2} = 0$

$$\iff -\sum_{j=1}^{N}\left[-\frac{1}{2}\delta_i^2 + \frac{1}{2}\left(x_i^{(j)} - \mu_{y^{(j)},i}\right)^2\right] = 0$$

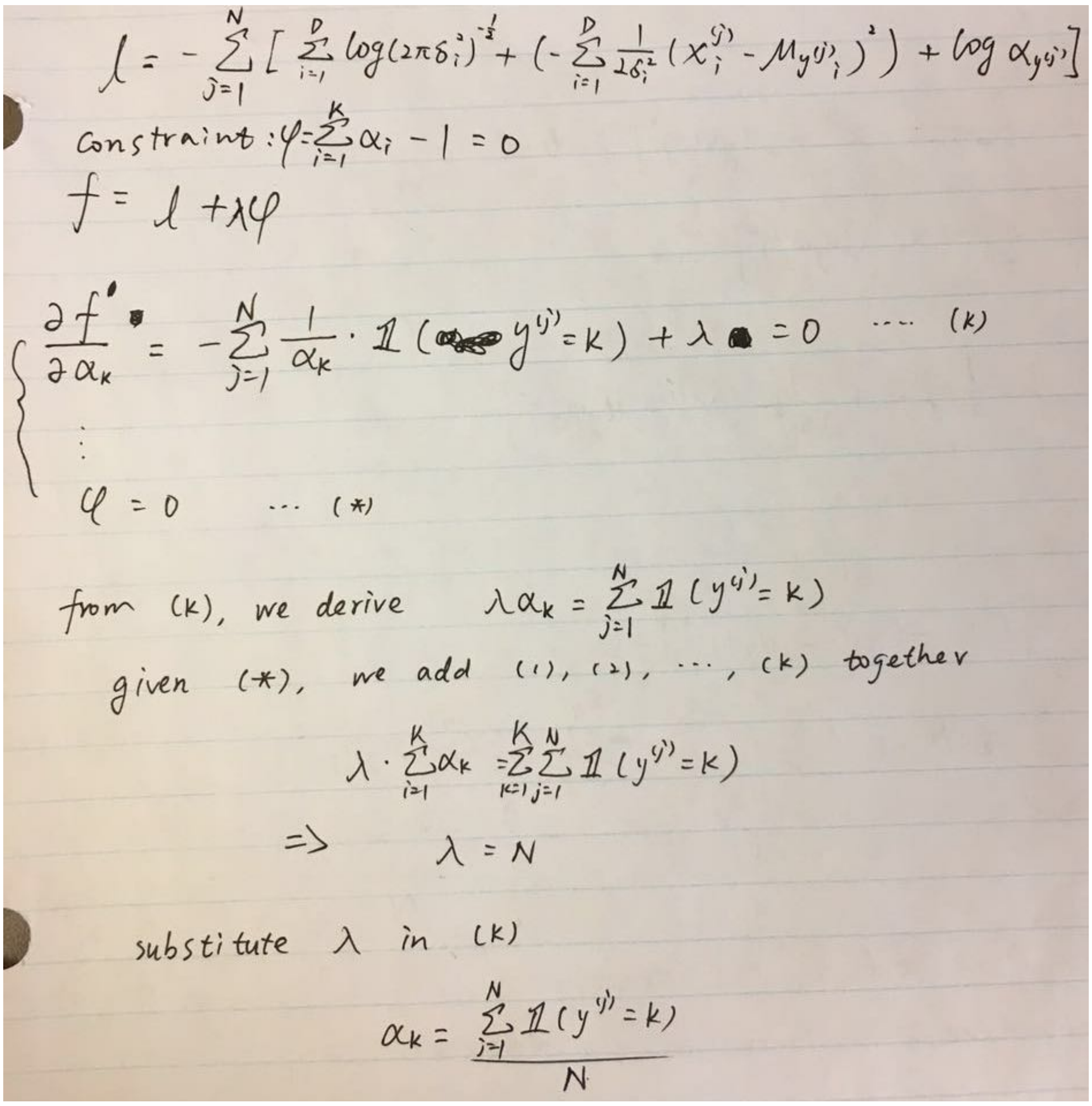
$$\iff \frac{1}{2}\cdot N\cdot \delta_i^2 = \sum_{j=1}^{N}\frac{1}{2}\left(x_i^{(j)} - \mu_{y^{(j)},i}\right)^2$$

$$\iff \delta_i^2 = \frac{\sum_{j=1}^{N}\left(x_i^{(j)} - \mu_{y^{(j)},i}\right)^2}{N}$$

So the MLE of δ_i^2 is exactly the population variance on the dataset.

(The MLE of δ_i is its square root).

5. Extra work for interested students: show that the MLE for α_k is indeed given by its frequency.



$$l = -\sum_{j=1}^{N}\left[\sum_{i=1}^{D}\log(2\pi\sigma_i^2)^{-\frac{1}{2}} + \left(-\sum_{i=1}^{D}\frac{1}{2\sigma_i^2}(x_i^{(j)} - \mu_{y^{(j)}_i})^2\right) + \log\alpha_{y^{(j)}}\right]$$

Constraint: $\varphi = \sum_{i=1}^{K}\alpha_i - 1 = 0$

$$f = l + \lambda\varphi$$

$$\left\{\begin{array}{l} \frac{\partial f}{\partial\alpha_k} = -\sum_{j=1}^{N}\frac{1}{\alpha_k}\cdot\mathbb{1}(y^{(j)}=k) + \lambda = 0 \quad \cdots\ (k) \\ \vdots \\ \varphi = 0 \quad \cdots\ (*) \end{array}\right.$$

from (k), we derive $\lambda\alpha_k = \sum_{j=1}^{N}\mathbb{1}(y^{(j)}=k)$

given (*), we add (1), (2), ⋯, (k) together

$$\lambda\cdot\sum_{i=1}^{K}\alpha_k = \sum_{k=1}^{K}\sum_{j=1}^{N}\mathbb{1}(y^{(j)}=k)$$

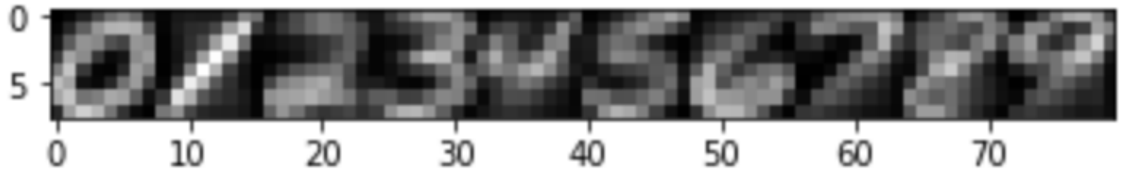
$$\Rightarrow \quad \lambda = N$$

substitute λ in (k)

$$\alpha_k = \frac{\sum_{j=1}^{N}\mathbb{1}(y^{(j)}=k)}{N}$$

2 Handwritten Digit Classification (11%)

0. Load the data and plot the means for each of the digit classes in the training data (include these in your report). Given that each image is a vector of size 64, the mean will be a vector of size 64 which needs to be reshaped as an 8 × 8 2D array to be rendered as an image. Plot all 10 means side by side using the same scale.



2.1 K-NN Classifier

1. Build a simple K nearest neighbor classifier using Euclidean distance on the raw pixel data.

- For K = 1 report the train and test classification accuracy.
  KNN-1 train accuracy: 1.0
  KNN-1 test accuracy: 0.96875
- For K = 15 report the train and test classification accuracy.

KNN-15 train accuracy: 0.9594285714285714
KNN-15 test accuracy: 0.9585

2. For K > 1 K-NN might encounter ties that need to be broken in order to make a decision.

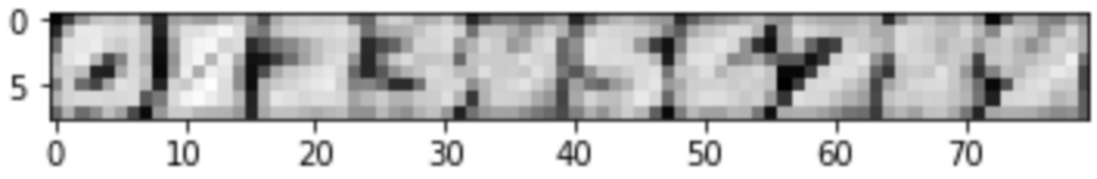
- Choose any (reasonable) method you prefer and explain it briefly in your report.
When there is a tie in model prediction, there is no posterior knowledge from the model we can rely on.
Therefore the tie can be broken by the prior probability: which prediction appears more frequently in the training set.
In the contruction of knn, the count of each label will be recorded to break the tie in such case. If the prior probability also hits a tie, we break it randomly.

3. Use 10 fold cross validation to find the optimal K in the 1-15 range. You may use the KFold implementation in sklearn or your existing code from Assignment 1.

- Report this value of K along with the train classification accuracy, the average accuracy across folds and the test accuracy.
K: 3, trainset accuracy: 0.9834285714285714,
average accuracy across folds: 0.9637142857142857, test accuracy: 0.96975

2.2 Conditional Gaussian Classifier Training

1. Plot an 8 by 8 image of the log of the diagonal elements of each covariance matrix Σ_k. Plot all ten classes side by side using the same grayscale.



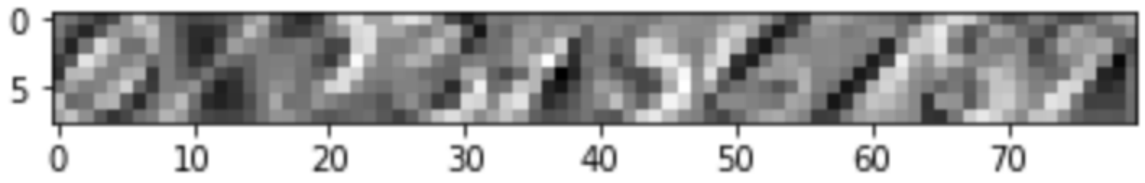
2. Using the parameters you fit on the training set and Bayes rule, compute the average conditional log-likelihood on both the train and test set and report it.

- average conditional log-likelihood for
train set: -0.12462443666862971
test set: -0.19667320325525603

3. Select the most likely posterior class for each training and test data point as your prediction, and report your accuracy on the train and test set.

- accuracy on trainset: 0.9814285714285714
accuracy on testset: 0.97275

4. Extra work for interested students: Compute the leading eigenvectors (largest eigenvalue) for each class covariance matrix (can use np.linalg.eig) and plot them side by side as 8 by 8 images.



2.3 Naive Bayes Classifier Training

1. Convert the real-valued features x into binary features b using 0.5 as a threshold.

- As seen from *binarize_data(pixel_values)*

```
return np.where(pixel_values > 0.5, 1.0, 0.0)
```

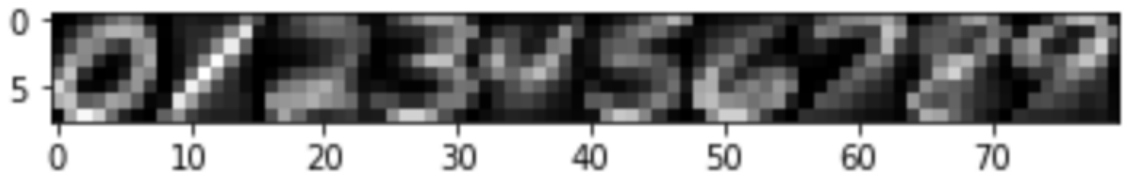
2. Using these new binary features b and the class labels, train a Bernoulli Naive Bayes classifier using MAP estimation with prior $Beta(\alpha, \beta)$ with $\alpha = \beta = 2$.

- As seen from *compute_parameters(train_data, train_labels)*

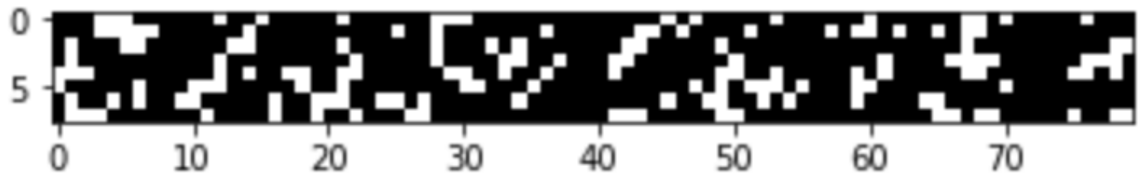
```
eta_k = (np.sum(train_data[ind], axis=0) + a - 1) / (ind.shape[0] + a + b -
2)
```

- The regularization technique is equivalent because with a blank image and a black image the occurence of every feature will be incremented by 1, whereas the sample size will be incremented by 2.
  This is exactly the case in the above code: when $\alpha = \beta = 2$, the numerator is increased by (a - 1) = 1, and the denominator is increased by (a + b - 2) = 2.

3. Plot each of your η_k vectors as an 8 by 8 grayscale image. These should be presented side by side and with the same scale.



4. Given your parameters, sample one new data point for each of the 10 digit classes. Plot these new data points as 8 by 8 grayscale images side by side.



5. Using the parameters you fit on the training set and Bayes rule, compute the average conditional log-likelihood on both the train and test set and report it.

- average conditional log-likelihood for
  train set: -0.9437538618002514
  test set: -0.9872704337253585

6. Select the most likely posterior class for each training and test data point, and report your accuracy on the train and test set.

- accuracy on trainset: 0.7741428571428571
  accuracy on testset: 0.76425

2.4 Model Comparison

Briefly (in a few sentences) summarize the performance of each model. Which performed best? Which performed worst? Did this match your expectations?

- Both KNN-3 model and conditional Gaussian model achieved training set accuracy over 98%, but the test set accuracy of Gaussian model (97%) is better than KNN (96%), indicating that Gaussian model generalizes better. Naive Bayes model has a mediocre performance on both training set (77%) and test set (76%).
  Based on the test accuracy, the best model is conditional Gaussian classifier, the worst is naive Bernoulli Bayes classifier.
  This is indeed what I would expect. The assumption of Naive Bayes model is that features [x1, .., xd] are independent given the class y. This obviously does not hold for a hand-written digit, where adjacent pixels are more likely to appear together in the pattern of the number it represents. The assumption of nearest-neighbor is that train cases in the same class should be close to each other, but this is not entirely true given there are many fonts of the same number, and the distance of them may be even bigger than cross-class cases (such as italic '1' and sans-serif '7'). Hence it may generalize worse than Gaussian model, where the comparison goes with the statistical average and covariance and thus relies less on the specificity of one single case.