HOMEWORK 2 - V1

STA 414/2104 Winter 2021

University of Toronto

Version history: V0 → V1: add ϵI to covariance in Q2.b

- **Deadline:** Mon, Feb. 22, at 13:59.
- **Submission:** You need to submit your solutions through Crowdmark, including all your derivations, plots, and your code. You can produce the file however you like (e.g. LaTeX, Microsoft Word, etc), as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code. For this assignment, you should submit the filled out starter code right after your answers.

MNIST dataset. In this assignment, you will fit both generative and discriminative models using the MNIST dataset of handwritten numbers.

Each datapoint in the MNIST `http://yann.lecun.com/exdb/mnist/` dataset is a 28x28 black-and-white image of a handwritten digit in $\{0 \dots 9\}$, and a label indicating which digit.

MNIST is the 'fruit fly' of machine learning - a simple standard problem useful for comparing the properties of different algorithms. A starter Python code that loads and plots the MNIST dataset is attached. For this assignment, we will *binarize* the data, converting grey-scale pixels to either black or white (0 or 1) with > 0.5 being the cutoff (already done in the starter code).

The starter code `hw2-train.py` is to be used in both questions below. You will need to write the missing parts of the functions and return it back for evaluation. Note that each missing part should be typically a few lines of code, so make sure your code is compact. When comparing models, you will need a training and test set. Build a dataset of only 2000 training samples (controlled by `N_data`) to use when coding or debugging, to make loading and training faster. Inspect the starter code carefully, before you start coding.

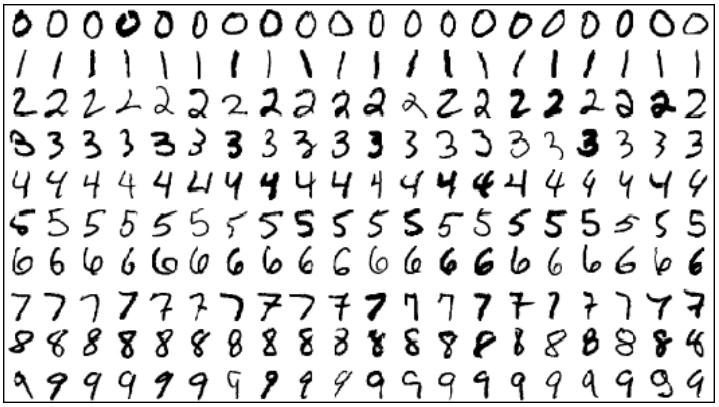

Fig 1: Samples from MNIST data set.

1. Multi-class Logistic Regression Classifier - 50 pts. In this question, you will fit a discriminative model using gradient descent. Our model will be multi-class logistic regression:

$$p(t_k = 1|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{i=0}^{9} \exp(\mathbf{w}_i^T \mathbf{x})}$$

Omit bias (intercept) parameters for this question.

(a) (5pts) How many parameters does this model have?

(b) (10pts) Write down the log-likelihood and convert it into a minimization problem over the cross-entropy loss E. Derive the gradient of E with respect to each $\mathbf{w}_k$, i.e., $\nabla_{\mathbf{w}_k} E(\mathbf{w})$.

(c) (30pts) Code up a gradient descent optimizer using the starter code provided to you, and minimize the cross-entropy loss. Report the final training and the test accuracy achieved. The training must be done over the full training dataset, unless there are computational issues, in which case you can reduce the number of training samples depending on the memory available. Report the number of samples used to obtain the final result. Hint: For `log_softmax` function, use `scipy.special.logsumexp` (already imported in the starter code) or its equivalent to make your code more numerically stable. Avoid nested `for` loops, and instead use matrix operations to keep your code fast. Each missing chunk should be a few lines of code!

(d) (5pts) Plot the final weights obtained as 10 images.

What to submit?

a) Number of parameters.
b) Log-likelihood, resulting cross-entropy minimization, and the gradient.
c) Final training and test errors as well as the number of samples used in training.
d) Figure containing each weight $\mathbf{w}_k$ as an image.
e) Your entire code should be attached to the end of your answers.

2. Gaussian Discriminant Analysis - 50 pts. In this part, we train a generative model using the MNIST dataset. Assuming that the data generating distribution is Gaussian, i.e.

$$p(\mathbf{x}|\mathcal{C}_k) = \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}). \tag{2.1}$$

We know that the posterior $p(\mathcal{C}_k|\mathbf{x})$ can be written in terms of the softmax function

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\exp\{a_k\}}{\sum_j \exp\{a_j\}} \quad \text{where} \quad a_k = \mathbf{w}_k^T \mathbf{x} + w_{k0}. \tag{2.2}$$

Here, we also know that

$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k \quad \text{and} \quad w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log(p(\mathcal{C}_k)). \tag{2.3}$$

(a) (5pts) Write down the log-likelihood implied by this model and find the maximum likelihood estimator (MLE) for the priors $p(\mathcal{C}_k) = \pi_k$ and the class means $\boldsymbol{\mu}_k$, for $k = 1, ..., K$. Note that you do not need to derive the MLE for the covariance matrix.

(b) (20pts) Compute the MLEs obtained in the previous part together with the following estimator for the covariance matrix

$$\widehat{\boldsymbol{\Sigma}} = \sum_{k=1}^{K} \frac{N_k}{N} \widehat{\boldsymbol{\Sigma}}_k \quad \text{where} \quad \widehat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T, \tag{2.4}$$

where N_k is the number of images that belong to class k, and N is the total number of images. In order to make $\widehat{\boldsymbol{\Sigma}}$ invertible, add ϵI for ϵ small e.g. $\epsilon = 1/N$. Plot the means of each class as an image. Hint: In this part, if you use the entire training dataset to train your model, your computer's memory will likely run out. Start with a small number `N_data` = 2000 and slowly increase it. In your final model, use as many samples as permitted by the computer memory. Report this number below. Try to avoid for loops as much as possible. Many of these operations can be written as matrix-matrix products. Take advantage of 1-of-K encoding.

(c) (15pts) Using the MLE estimators obtained in previous part as well as the posterior (2.2), make predictions on both training and test sets and report the obtained accuracy in each dataset. Also, report the number of training images used to compute the MLE estimators.

(d) (5pts) Briefly compare the performance of this model to that of logistic regression.

(e) (5pts) Using the generative model you trained, generate 10 images from digit 0 and 10 images from digit 3.

What to submit?

a) Log-likelihood, MLE for class means and the priors, your derivations.
b) Figure containing each class mean $\boldsymbol{\mu}_k$ as an image.
c) Final training and test errors as well as the number of samples used in training.
d) Brief comparison of final accuracies.
e) 20 images you generated.
f) Your entire code should be attached to the end of your answers.

1. Multi-class Logistic Regression Classifier - 50 pts. In this question, you will fit a discriminative model using gradient descent. Our model will be multi-class logistic regression:

$$p(t_k = 1|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{i=0}^{9} \exp(\mathbf{w}_i^T \mathbf{x})}$$

Omit bias (intercept) parameters for this question.

(a) (5pts) How many parameters does this model have?

(b) (10pts) Write down the log-likelihood and convert it into a minimization problem over the cross-entropy loss E. Derive the gradient of E with respect to each $\mathbf{w}_k$, i.e., $\nabla_{\mathbf{w}_k} E(\mathbf{w})$.

(c) (30pts) Code up a gradient descent optimizer using the starter code provided to you, and minimize the cross-entropy loss. Report the final training and the test accuracy achieved. The training must be done over the full training dataset, unless there are computational issues, in which case you can reduce the number of training samples depending on the memory available. Report the number of samples used to obtain the final result. Hint: For `log_softmax` function, use `scipy.special.logsumexp` (already imported in the starter code) or its equivalent to make your code more numerically stable. Avoid nested `for` loops, and instead use matrix operations to keep your code fast. Each missing chunk should be a few lines of code!

(d) (5pts) Plot the final weights obtained as 10 images.

What to submit?

a) Number of parameters.
b) Log-likelihood, resulting cross-entropy minimization, and the gradient.
c) Final training and test errors as well as the number of samples used in training.
d) Figure containing each weight $\mathbf{w}_k$ as an image.
e) Your entire code should be attached to the end of your answers.

(a).

From the question, the dataset we use is MINIST.

∴ Each datapoint in this dataset is a 28×28 black-and-white image of a handwritten digit in $\{0, \ldots, 9\}$

∴ we have a total of 10 classes $\{0, \ldots, 9\}$

For each class, we have $28 \times 28 = 784$ data points.

$$W = \begin{bmatrix} w_0 \rightarrow & 1\times 784 \\ \vdots & \vdots \\ w_9 \rightarrow & 1\times 784 \end{bmatrix}_{10\times 784}$$

$784 \times 10 = 7840$

$\therefore$ this model has **7840** parameters.

(b).

likelihood fcn: $p(T \mid X, w_0, \dots, w_9) = \prod_{n=1}^{N} \left(\prod_{k=0}^{9} p(C_k \mid x_n)^{t_{nk}} \right)$

$= \prod_{n=1}^{N} \left(\prod_{k=0}^{9} y_{nk}^{t_{nk}} \right)$

log-likelihood: $\ln p(T \mid X, w_0, \dots, w_9) = \ln \prod_{n=1}^{N} \left(\prod_{k=0}^{9} y_{nk}^{t_{nk}} \right)$

$= \sum_{n=1}^{N} \left(\sum_{k=0}^{9} t_{nk} \ln y_{nk} \right)$

cross-entropy loss E: $E(w) = -\ln p(T \mid X, w_0, \dots, w_9)$

$= -\sum_{n=1}^{N} \left(\sum_{k=0}^{9} t_{nk} \ln y_{nk} \right)$

$$y_{nk} = p(C_k \mid x_n)$$

$$= \frac{e^{w_k^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}}$$

$$\Rightarrow \ln y_{nk} = \ln e^{w_k^T x_n} - \ln \sum_{i=0}^{q} e^{w_i^T x_n}$$

$$= w_k^T x_n - \ln \left(\sum_{i=0}^{q} e^{w_i^T x_n} \right)$$

Gradient :

$$\frac{\partial(\ln y_{nk})}{\partial w_j} \quad :$$

① $k = j$

$$\frac{\partial(\ln y_{nk})}{\partial w_j} = x_n - \frac{1}{\sum_{i=0}^{q} e^{w_i^T x_n}} \cdot e^{w_j^T x_n} \cdot x_n$$

$$= x_n \left(1 - \frac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}} \right)$$

② $k \neq j$

$$\frac{\partial(\ln y_{nk})}{\partial w_j} = 0 - \frac{1}{\sum_{i=0}^{q} e^{w_i^T x_n}} \cdot e^{w_j^T x_n} \cdot x_n$$

$$= - \frac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}} \cdot x_n$$

$$\therefore \quad \frac{\partial(\ln y_{nk})}{\partial w_j} = \begin{cases} x_n \left(1 - \dfrac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}} \right), & \text{if } k = j \\ - \dfrac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}} \cdot x_n, & \text{if } k \neq j \end{cases}$$

$$\frac{\partial\left(\sum_{i=0}^{q} t_{nk} \ln y_{nk}\right)}{\partial w_j} :$$

① $k = j$

$$\frac{\partial(\ln y_{nk})}{\partial w_j} = x_n \left(1 - \frac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}}\right)$$

$t_{nj} = 1$

$\frac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}} = y_{nj}$

② $k \neq j$

only difference.

which would be t_{nj}, $t_{nj} \in \{0, 1\}$

$$\frac{\partial(\ln y_{nk})}{\partial w_j} = \left(0 - \frac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}}\right) \cdot x_n$$

$t_{nj} = 0$

$\frac{e^{w_j^T x_n}}{\sum_{i=0}^{q} e^{w_i^T x_n}} = y_{nj}$

$$\Rightarrow \frac{\partial\left(\sum_{i=0}^{q} t_{nk} \ln y_{nk}\right)}{\partial w_j} = x_n (t_{nj} - y_{nj})$$

$$\Rightarrow \frac{\partial\left(\sum_{n=1}^{N}\left(\sum_{i=0}^{q} t_{nk} \ln y_{nk}\right)\right)}{\partial w_j} = \sum_{n=1}^{N} x_n \left(t_{nj} - y_{nj}\right)$$

$$\Rightarrow \frac{\partial\left(-\sum_{n=1}^{N}\left(\sum_{i=0}^{q} t_{nk} \ln y_{nk}\right)\right)}{\partial w_j} = -\sum_{n=1}^{N} x_n \left(t_{nj} - y_{nj}\right)$$

$$= \sum_{n=1}^{N} \left(y_{nj} - t_{nj}\right) x_n$$

$$\therefore \frac{\partial E(w)}{\partial w_j} = \sum_{n=1}^{N} \left(y_{nj} - t_{nj}\right) x_n .$$

$$\nabla_{w_k} E(w) = \begin{bmatrix} \frac{\partial(E(w))}{\partial w_0} \\ \vdots \\ \frac{\partial(E(w))}{\partial w_q} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{n=1}^{N} \left(y_{n0} - t_{n0}\right) x_n \\ \vdots \\ \sum_{n=1}^{N} \left(y_{nq} - t_{nq}\right) x_n \end{bmatrix}$$

$$= \sum_{n=1}^{N} \begin{bmatrix} y_{n0} - t_{n0} \\ \vdots \\ y_{nq} - t_{nq} \end{bmatrix} x_n$$

$\therefore$ log-likelihood: $\sum_{n=1}^{N} \left(\sum_{k=0}^{9} t_{nk} \ln y_{nk} \right)$

cross-entropy loss E: $- \sum_{n=1}^{N} \left(\sum_{k=0}^{9} t_{nk} \ln y_{nk} \right)$

Gradient:

$$\begin{bmatrix} \sum_{n=1}^{N} (y_{n0} - t_{n0}) x_n \\ \vdots \\ \sum_{n=1}^{N} (y_{nq} - t_{nq}) x_n \end{bmatrix}$$

$$= \sum_{n=1}^{N} \begin{bmatrix} y_{n0} - t_{n0} \\ \vdots \\ y_{nq} - t_{nq} \end{bmatrix} x_n$$

$$w^{t+1} \leftarrow w^t - \eta \nabla_{w_K} E[w]$$

2. Gaussian Discriminant Analysis - 50 pts. In this part, we train a generative model using the MNIST dataset. Assuming that the data generating distribution is Gaussian, i.e.

$$p(\mathbf{x}|\mathcal{C}_k) = \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}). \tag{2.1}$$

We know that the posterior $p(\mathcal{C}_k|\mathbf{x})$ can be written in terms of the softmax function

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\exp\{a_k\}}{\sum_j \exp\{a_j\}} \quad \text{where} \quad a_k = \mathbf{w}_k^T \mathbf{x} + w_{k0}. \tag{2.2}$$

Here, we also know that

$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \quad \text{and} \quad w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log(p(\mathcal{C}_k)). \tag{2.3}$$

(a) (5pts) Write down the log-likelihood implied by this model and find the maximum likelihood estimator (MLE) for the priors $p(\mathcal{C}_k) = \pi_k$ and the class means $\boldsymbol{\mu}_k$, for $k = 1, ..., K$. Note that you do not need to derive the MLE for the covariance matrix.

(b) (20pts) Compute the MLEs obtained in the previous part together with the following estimator for the covariance matrix

$$\widehat{\boldsymbol{\Sigma}} = \sum_{k=1}^{K} \frac{N_k}{N} \widehat{\boldsymbol{\Sigma}}_k \quad \text{where} \quad \widehat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T, \tag{2.4}$$

where N_k is the number of images that belong to class k, and N is the total number of images. In order to make $\widehat{\boldsymbol{\Sigma}}$ invertible, add ϵI for ϵ small e.g. $\epsilon = 1/N$. Plot the means of each class as an image. Hint: In this part, if you use the entire training dataset to train your model, your computer's memory will likely run out. Start with a small number `N_data` = 2000 and slowly increase it. In your final model, use as many samples as permitted by the computer memory. Report this number below. Try to avoid for loops as much as possible. Many of these operations can be written as matrix-matrix products. Take advantage of 1-of-K encoding.

(c) (15pts) Using the MLE estimators obtained in previous part as well as the posterior (2.2), make predictions on both training and test sets and report the obtained accuracy in each dataset. Also, report the number of training images used to compute the MLE estimators.

(d) (5pts) Briefly compare the performance of this model to that of logistic regression.

(e) (5pts) Using the generative model you trained, generate 10 images from digit 0 and 10 images from digit 3.

What to submit?

a) Log-likelihood, MLE for class means and the priors, your derivations.
b) Figure containing each class mean $\boldsymbol{\mu}_k$ as an image.
c) Final training and test errors as well as the number of samples used in training.
d) Brief comparison of final accuracies.
e) 20 images you generated.
f) Your entire code should be attached to the end of your answers.

(a).

$$p(t, X \mid \pi, \{\mu_0, \dots, \mu_q\}, \Sigma) = \prod_{n=1}^{N} p(t_n, x_n \mid \pi, \{\mu_0, \dots, \mu_q\}, \Sigma)$$

$$= \prod_{n=1}^{N} \left(\prod_{k=0}^{q} \left(\pi_k N(x_n \mid \mu_k, \Sigma) \right)^{t_{nk}} \right)$$

log-likelihood : $\ln p(t, X \mid \pi, \{\mu_0, \dots, \mu_q\}, \Sigma)$

$$= \ln \prod_{n=1}^{N} \left(\prod_{k=0}^{q} \left(\pi_k N(x_n \mid \mu_k, \Sigma) \right)^{t_{nk}} \right)$$

$$= \sum_{n=1}^{N} \sum_{k=0}^{q} \left(t_{nk} \ln \pi_k + t_{nk} \ln N(x_n \mid \mu_k, \Sigma) \right)$$

- MLE for π_k :

maximize $\ln p(t, X \mid \pi, \{\mu_0, \dots, \mu_q\}, \Sigma)$

subject to $\sum_{k=0}^{q} \pi_k = 1$.

$$f = \sum_{n=1}^{N} \sum_{k=0}^{q} \left(t_{nk} \ln \pi_k + t_{nk} \ln N(x_n \mid \mu_k, \Sigma) \right) + \lambda \left(\sum_{k=0}^{q} \pi_k - 1 \right)$$

$$\frac{\partial f}{\partial \pi_k} = \frac{\partial}{\partial \pi_k} \left(\sum_{n=1}^{N} \sum_{k=0}^{K} \left(t_{nk} \ln \pi_k + t_{nk} \ln N(x_n \mid \mu_k, \Sigma) \right) + \lambda \left(\sum_{k=0}^{K} \pi_k - 1 \right) \right)$$

$$= \frac{\partial}{\partial \pi_k} \left(\sum_{n=1}^{N} \sum_{k=0}^{K} t_{nk} \ln \pi_k \right) + 0 + \frac{\partial}{\partial \pi_k} \left(\lambda \sum_{k=0}^{K} \pi_k \right)$$

$$= \sum_{n=1}^{N} t_{nk} \cdot \frac{1}{\pi_k} + \lambda$$

$$\sum_{n=1}^{N} t_{nk} \cdot \frac{1}{\pi_k} + \lambda = 0$$

$$-\lambda = \frac{\sum_{n=1}^{N} t_{nk}}{\pi_k}$$

$$\hat{\pi}_k = \frac{\sum_{n=1}^{N} t_{nk}}{-\lambda}$$

$$\because \sum_{k=0}^{K} \hat{\pi}_k = 1.$$

$$\therefore -\lambda = N$$

$$\lambda = -N$$

$$\therefore \hat{\pi}_k = \frac{\sum_{n=1}^{N} t_{nk}}{N}$$

· MLE for μ_k :

$$\frac{\partial}{\partial \mu_k}\left(\sum_{n=1}^{N}\sum_{k=0}^{K}\left(t_{nk}\ln \pi_k + t_{nk}\ln N(x_n|\mu_k,\Sigma)\right)\right)$$

$$= \frac{\partial}{\partial \mu_k}\left(\sum_{n=1}^{N}\sum_{k=0}^{K} t_{nk}\ln N(x_n|\mu_k,\Sigma)\right)$$

$$= \frac{\partial}{\partial \mu_k}\left(\sum_{n=1}^{N}\sum_{k=0}^{K} t_{nk}\cdot\left(-\frac{1}{2}(x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k)\right)\right)$$

$\frac{\partial}{\partial \mu_k}$: $(\Sigma^{-1}+(\Sigma^{-1})^T)(x_n-\mu_k)$

$\because \Sigma^{-1}$ is symmetric

$\therefore \Rightarrow 2\Sigma^{-1}(x_n-\mu_k)$

$$= \sum_{n=1}^{N} t_{nk}\left(-\frac{1}{2}\right)\left(2\Sigma^{-1}(x_n-\mu_k)\right)$$

$$= -\sum_{n=1}^{N} t_{nk}\Sigma^{-1}(x_n-\mu_k)$$

$$-\sum_{n=1}^{N} t_{nk}\Sigma^{-1}(x_n-\mu_k) = 0$$

$$\Sigma^{-1}\sum_{n=1}^{N} t_{nk}x_n - \Sigma^{-1}\sum_{n=1}^{N} t_{nk}\mu_k = 0$$

$$\sum_{n=1}^{N} t_{nk}x_n = \mu_k\sum_{n=1}^{N} t_{nk}$$

$$\hat{\mu}_k = \frac{\sum_{n=1}^{N} t_{nk}x_n}{\sum_{n=1}^{N} t_{nk}}$$

$$= \frac{\sum_{n\in C_k} x_n}{N_k}$$

$\therefore$ log-likelihood :

$$\sum_{n=1}^{N} \sum_{k=0}^{9} \left(t_{nk} \ln \pi_k + t_{nk} \ln N(x_n | \mu_k, \Sigma) \right)$$

MLE estimators : $\hat{\pi}_k = \dfrac{\sum_{n=1}^{N} t_{nk}}{N}$

$$\hat{\mu}_k = \frac{\sum_{n=1}^{N} t_{nk} x_n}{\sum_{n=1}^{N} t_{nk}}$$

$$= \frac{\sum_{n \in C_k} x_n}{N_k}$$