

CSCI-GA 3033: Honors Machine Learning**Assignment 2**

Due: November 9, 2026.

Submit on Gradescope.

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!

[Text in square brackets are hints that can be ignored.]

Data are in `data/`. Use NumPy and plotting tools for your implementations; implement the requested learning algorithms yourself.

Exercise 1: Decision Trees (8 pts)

In this exercise we will implement decision trees for binary classification. Use the provided stub file `ex1.py`.

Recall: decision trees are constructed by repeatedly splitting of nodes. We split a node by measuring the loss of splitting with respect to each possible feature and threshold, and split based on the feature and threshold that minimizes this loss. Mathematically:

$$X_L = \{x : x \in X \wedge x[i] \leq j\},$$

$$X_R = \{x : x \in X \wedge x[i] > j\},$$

where $x[i]$ is the i th coordinate of point x . The vector of labels y is split into vectors y_L and y_R using the same indices.

The loss of splitting a training dataset into a left and right half is computed as

$$\ell(X, y, i, j) = \frac{|y_L|}{|y|} \ell(y_L) + \frac{|y_R|}{|y|} \ell(y_R)$$

We will consider the following loss functions ℓ , specialized for binary classification. (These binary-classification criteria omit constant scaling factors.) Define $\hat{p}$ for a vector of labels y to be $\frac{|\{y_j=1: y_j \in y\}|}{|y|}$, that is, the fraction of labels which are 1. We have the following three loss functions.

Misclassification error:

$$\min\{\hat{p}, 1 - \hat{p}\}$$

Gini coefficient:

$$\hat{p}(1 - \hat{p})$$

Entropy:

$$-\hat{p} \log_2(\hat{p}) - (1 - \hat{p}) \log_2(1 - \hat{p})$$

We do not split a node if it is pure (i.e., consists entirely of either 0's or 1's), or if a split would exceed a maximum depth hyperparameter provided to the decision tree (recall that the depth of a single-node tree is 0).

- a) (6 pts) Implement and train decision trees on the provided dataset D. Create a different plot for each of the three loss functions (misclassification error, Gini index, and entropy). The x-axis of each plot should show the maximum depth of the tree (starting from 0), and the y-axis should indicate the accuracy. Include two trend lines, one for the training accuracy and test accuracy. Observe and comment on how the different loss functions perform, and how train and test accuracy change as a function of the maximum depth.
- b) (2 pts) Implement and use Bagging on decision trees. Use the entropy loss, and create an ensemble of 101 decision trees, capping the maximum depth at 3. Repeat 11 times, report the median, minimum, and maximum test classification accuracy. Repeat, but using the random forests method. In particular, at each split, consider only a random $\sqrt{d}$ features when deciding which dimension to split on. Since $\sqrt{d} \approx 3.6$ for the given dataset, choose a random 4 features for each time. Report the classification accuracies as before. Comment on performance between the two, as well as how they perform in comparison to the non-ensemble methods from the previous part.

Exercise 2: Adaboost (5 pts)

Recall the update rules of Adaboost:

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^n w_j^t}, \quad i = 1, \dots, n \quad (1)$$

$$\epsilon_t = \epsilon_t(h_t) = \sum_{i=1}^n p_i^t \cdot \mathbb{I}[h_t(\mathbf{x}_i) \neq y_i] \quad (2)$$

$$\beta_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \quad (3)$$

$$w_i^{t+1} = w_i^t \exp(-y_i \beta_t h_t(\mathbf{x}_i)), \quad i = 1, \dots, n. \quad (4)$$

Here we use i and t to index the training examples and iterations, respectively. $\mathbb{I}[A] = 1$ if the event A holds and 0 otherwise. Here $y_i \in \{\pm 1\}$ and **we also assume** $h_t(\mathbf{x}_i) \in \{\pm 1\}$. In this exercise we offer additional insights about Adaboost.

- a) (1 pt) Prove by induction that the updates defined above are equivalent to the following updates where the y 's and h 's are $\{0, 1\}$ rather than $\{\pm 1\}$:

$$\tilde{\beta}_t = \frac{\epsilon_t}{1 - \epsilon_t} \quad (5)$$

$$\tilde{w}_i^{t+1} = \tilde{w}_i^t \tilde{\beta}_t^{1 - |\tilde{h}_t(\mathbf{x}_i) - \tilde{y}_i|}, \quad (6)$$

where $\tilde{y}_i = \frac{y_i + 1}{2}$, $\tilde{h}(\mathbf{x}_i) = \frac{h(\mathbf{x}_i) + 1}{2} \in \{0, 1\}$.

[Hint: Show that p_i^t remains the same under the two seemingly different updates.]

Ans:

- b) (1 pt) Recall in the logistic regression lecture we made the linear assumption on the log odds ratio:

$$\log \frac{p(y = 1 | \mathbf{X} = \mathbf{x})}{p(y = -1 | \mathbf{X} = \mathbf{x})} \approx \mathbf{w}^\top \mathbf{x} + b. \quad (7)$$

Adaboost in effect tries to approximate the log odds ratio using *additive* functions:

$$\log \frac{p(y = 1 | \mathbf{X} = \mathbf{x})}{p(y = -1 | \mathbf{X} = \mathbf{x})} \approx 2 \sum_{t=1}^T \beta_t h_t(\mathbf{x}), \quad (8)$$

where each $h_t(\mathbf{x})$ is a weak classifier. Indeed, fixing $\mathbf{X} = \mathbf{x}$, prove that the minimizer of the following exponential loss

$$\min_{H \in \mathbb{R}} \mathbb{E}[\exp(-yH) | \mathbf{X} = \mathbf{x}] \quad (9)$$

is (proportional to) the log odds ratio. Here the expectation is wrt the conditional distribution $p(y | \mathbf{X} = \mathbf{x})$.

[Any function can be approximated arbitrarily well by additive functions but clearly not by linear functions, thus the power of Adaboost.]

Ans:

- c) (1 pt) Suppose h_t is a weak classifier whose error $\epsilon_t > 1/2$, i.e. worse than random guessing! In this case it makes sense to flip h_t to $\bar{h}_t(\mathbf{x}) = -h_t(\mathbf{x})$. Compute the error $\epsilon_t(\bar{h}_t)$ and the resulting β_t . Do we get the same update for w in (4)? Explain.

Ans:

- d) (1 pt) Adaboost is a greedy algorithm where we find the weak classifiers sequentially. At iteration t , the classifiers $h_s, s < t$ are already found along with their coefficients β_s . Suppose h_t is given by some oracle, to find the optimal coefficient β_t , we solve an empirical approximation of the exponential loss (9):

$$\min_{\beta \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^n \exp(-y_i [H_{t-1}(\mathbf{x}_i) + \beta h_t(\mathbf{x}_i)]), \quad (10)$$

where needless to say, $H_{t-1}(\mathbf{x}) := \sum_{s=1}^{t-1} \beta_s h_s(\mathbf{x})$. While it is possible to solve (10) directly, we gain more insights by defining a distribution over the training examples $(\mathbf{x}_i, y_i), i = 1, \dots, n$:

$$p_i^t = \frac{\exp(-y_i H_{t-1}(\mathbf{x}_i))}{\sum_{i=1}^n \exp(-y_i H_{t-1}(\mathbf{x}_i))}, \quad (11)$$

so that we can rewrite (10) equivalently as:

$$\min_{\beta \in \mathbb{R}} \hat{\mathbb{E}}_t \exp[-y \beta h_t(\mathbf{x})], \quad (\mathbf{x}, y) \sim \mathbf{p}^t. \quad (12)$$

(Here the hat notation is to remind you that this is an empirical expectation specified by $\mathbf{p}^t$ over our training data.) Let ϵ_t be defined as in (2). Prove that the optimal β in (12) is given in (3).

[Hint: We remind again that both y and $h_t(\mathbf{x})$ are $\{\pm 1\}$ -valued. Split training examples according to $h_t(\mathbf{x}_i) = y_i$ or not.]

Ans:

- e) (1 pt) What is the training error

$$\epsilon_{t+1}(h_t) = \sum_{i=1}^n p_i^{t+1} \cdot \mathbb{I}[h_t(\mathbf{x}_i) \neq y_i] \quad (13)$$

of the weak classifier h_t on the next round $t + 1$? Justify your answer. You may assume $0 < \epsilon_t < 1$ so that all quantities are well-defined.

[Hint: This exercise should be simple, given what you have done in d). Split training examples according to $h_t(\mathbf{x}_i) = y_i$ or not.]

Ans:

Exercise 3: k -Means and Expectation-Maximization (3 pts)

This exercise connects EM for a Gaussian mixture to k -means as the shared variance tends to zero. Part (a) shows that the EM updates approach the updates in Lloyd's k -means algorithm. Part (b) shows that the scaled negative log-likelihood approaches the k -means objective. Thus, in this limit, both the updates and the objectives agree.

Consider a Gaussian mixture with K components, equal fixed mixture weights $1/K$, and shared fixed covariance $\sigma^2 I_d$, where $\sigma > 0$. Only the means $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ are learned. Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ be the observations. The responsibility r_{ik} of component k for observation i is

$$r_{ik} = \frac{\exp(-\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 / (2\sigma^2))}{\sum_{j=1}^K \exp(-\|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2 / (2\sigma^2))}.$$

- a) (1 pt) Hold the current means fixed and assume that every observation has a unique nearest mean. Determine the limit of r_{ik} as $\sigma \rightarrow 0$. Derive the EM mean update and show that it approaches the centroid update in Lloyd's k -means algorithm, provided every limiting cluster is nonempty. **Ans:**
- b) (2 pts) The negative log-likelihood, after omitting the Gaussian normalization term that is independent of the means, is

$$L_\sigma(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K) = - \sum_{i=1}^n \log \left[\frac{1}{K} \sum_{k=1}^K \exp \left(-\frac{\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2}{2\sigma^2} \right) \right].$$

For fixed means, prove that

$$\lim_{\sigma \rightarrow 0} 2\sigma^2 L_\sigma = \sum_{i=1}^n \min_{1 \leq k \leq K} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2.$$

[For each observation, factor out the exponential corresponding to its nearest mean.] **Ans:**

Exercise 4: Gaussian Mixture Model (GMM) (10 pts)

Notation: For a matrix A , $|A|$ denotes its determinant. For a diagonal matrix $\text{diag}(\mathbf{s})$, $|\text{diag}(\mathbf{s})| = \prod_i s_i$.

Algorithm 1: EM for GMM.

Input: $X \in \mathbb{R}^{n \times d}$, $K \in \mathbb{N}$, initialization for model
 // model includes $\pi \in \mathbb{R}_+^K$ and for each $1 \leq k \leq K$, $\mu_k \in \mathbb{R}^d$ and $S_k \in \mathbb{S}_+^d$
 // $\pi_k \geq 0$, $\sum_{k=1}^K \pi_k = 1$, S_k symmetric and positive definite.
 // random initialization suffices for full credit.
 // alternatively, can initialize r by randomly assigning each data to one of the K components
Output: model, ℓ

```

1 for iter = 1 : MAXITER do
  // step 2, for each  $i = 1, \dots, n$ 
2   for  $k = 1, \dots, K$  do
3      $r_{ik} \leftarrow \pi_k |S_k|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_i - \mu_k)^\top S_k^{-1}(\mathbf{x}_i - \mu_k)]$  // compute responsibility
  // for each  $i = 1, \dots, n$ 
4    $r_i \leftarrow \sum_{k=1}^K r_{ik}$ 
  // for each  $k = 1, \dots, K$  and  $i = 1, \dots, n$ 
5    $r_{ik} \leftarrow r_{ik} / r_i$  // normalize
  // compute negative log-likelihood
6    $\ell(\text{iter}) = -\sum_{i=1}^n \log(r_i)$ 
7   if  $\text{iter} > 1$  &&  $|\ell(\text{iter}) - \ell(\text{iter} - 1)| \leq \text{TOL} * |\ell(\text{iter})|$  then
8     break
  // step 1, for each  $k = 1, \dots, K$ 
9    $r_{.k} \leftarrow \sum_{i=1}^n r_{ik}$ 
10   $\pi_k \leftarrow r_{.k} / n$ 
11   $\mu_k = \sum_{i=1}^n r_{ik} \mathbf{x}_i / r_{.k}$ 
12   $S_k \leftarrow (\sum_{i=1}^n r_{ik} \mathbf{x}_i \mathbf{x}_i^\top / r_{.k}) - \mu_k \mu_k^\top$ 

```

- a) (5 pts) Derive and implement the EM algorithm for the diagonal Gaussian mixture model, where all covariance matrices are constrained to be diagonal. Algorithm 1 recaps all the essential steps and serves as a hint rather than a verbatim instruction. In particular, you must change the highlighted steps accordingly (with each S_k being a diagonal matrix), along with formal explanations. Analyze the space and time complexity of your implementation.

[You might want to review the steps we took in class for a simpler case, and ensure you can derive the updates in Algorithm 1. Then adapt the steps to the simpler diagonal case. The solution should look like $s_j = \frac{\sum_{i=1}^n r_{ik} (x_{ij} - \mu_j)^2}{\sum_{i=1}^n r_{ik}} = \frac{\sum_{i=1}^n r_{ik} x_{ij}^2}{\sum_{i=1}^n r_{ik}} - \mu_j^2$ for the j -th diagonal. Multiplying an $n \times p$ matrix with a $p \times m$ matrix costs $O(mnp)$. Do not maintain a diagonal matrix explicitly; using a vector for its diagonal suffices.]

[Warning: You may run into issues involving NaNs. You will have to diagnose these issues and fix them.]

To stop the algorithm, set a maximum number of iterations (say $\text{MAXITER} = 500$) and also monitor the change of the negative log-likelihood ℓ :

$$\ell = -\sum_{i=1}^n \log \left[\sum_{k=1}^K \pi_k |2\pi S_k|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_i - \mu_k)^\top S_k^{-1}(\mathbf{x}_i - \mu_k)] \right], \quad (14)$$

where $\mathbf{x}_i$ is the i -th column of $X^\top$. As a debug tool, note that ℓ should decrease from step to step, and we can stop the algorithm if the decrease is smaller than a predefined threshold, say $\text{TOL} = 10^{-5}$.

Run your algorithm on `data/gmm_dataset.csv`, for $k = 1$ to 10. Generate a plot with k on the x-axis and the negative log-likelihood of the data under the final trained model on the y-axis. What do you think the most appropriate choice of k is? Explain and justify how and why you chose this value. [You may want to focus on more than just maximizing the log-likelihood.] For your chosen value of k , report the parameters (mixing weights, mean vectors, and vectors corresponding to the diagonals of the covariance matrices) of your trained model. When reporting them, sort the components in increasing order of mixing weights.

Ans:

- b) (5 pts) Next, we apply (the adapted) Algorithm 1 in part a) to the MNIST dataset. For each of the 10 classes (digits), we can use its (only its) training images to estimate its (class-conditional) distribution by fitting a GMM (with say $K = 5$, roughly corresponding to 5 styles of writing this digit). This gives us the density estimate $p(\mathbf{x}|y)$ where $\mathbf{x}$ is an image (of some digit) and y is the class (digit). We can now classify the test set using the Bayes classifier:

$$\hat{y}(\mathbf{x}) = \arg \max_{c=0,\dots,9} \underbrace{\Pr(Y = c) \cdot p(X = \mathbf{x}|Y = c)}_{\propto \Pr(Y=c|X=\mathbf{x})}, \quad (15)$$

where the probabilities $\Pr(Y = c)$ can be estimated using the training set, e.g., the proportion of the c -th class in the training set, and the density $p(X = \mathbf{x}|Y = c)$ is estimated using GMM for each class c separately. Report your error rate on the test set as a function of K (if time is a concern, using $K = 5$ will receive full credit).

Reduce the dimension of the MNIST images using PCA before fitting the GMMs. Fit a single PCA transformation on the training set and apply the same transformation to the training and test images. Report the number of principal components retained.

[Your running time should be on the order of minutes (for one K), if you do not introduce extra for-loops in Algorithm 1.]

[In case you are wondering, our classification procedure above belongs to the so-called plug-in estimators (plug the estimated densities to the known optimal Bayes classifier). However, note that estimating the density $p(X = \mathbf{x}|Y = c)$ is actually harder than classification. Solving a problem (e.g. classification) through some intermediate harder problem (e.g. density estimation) is almost always a bad idea.]

Ans: