CS480/680: Introduction to Machine Learning

Homework 3

Due: 11:59 pm, July 08, 2024, submit on LEARN.

NAME

student number

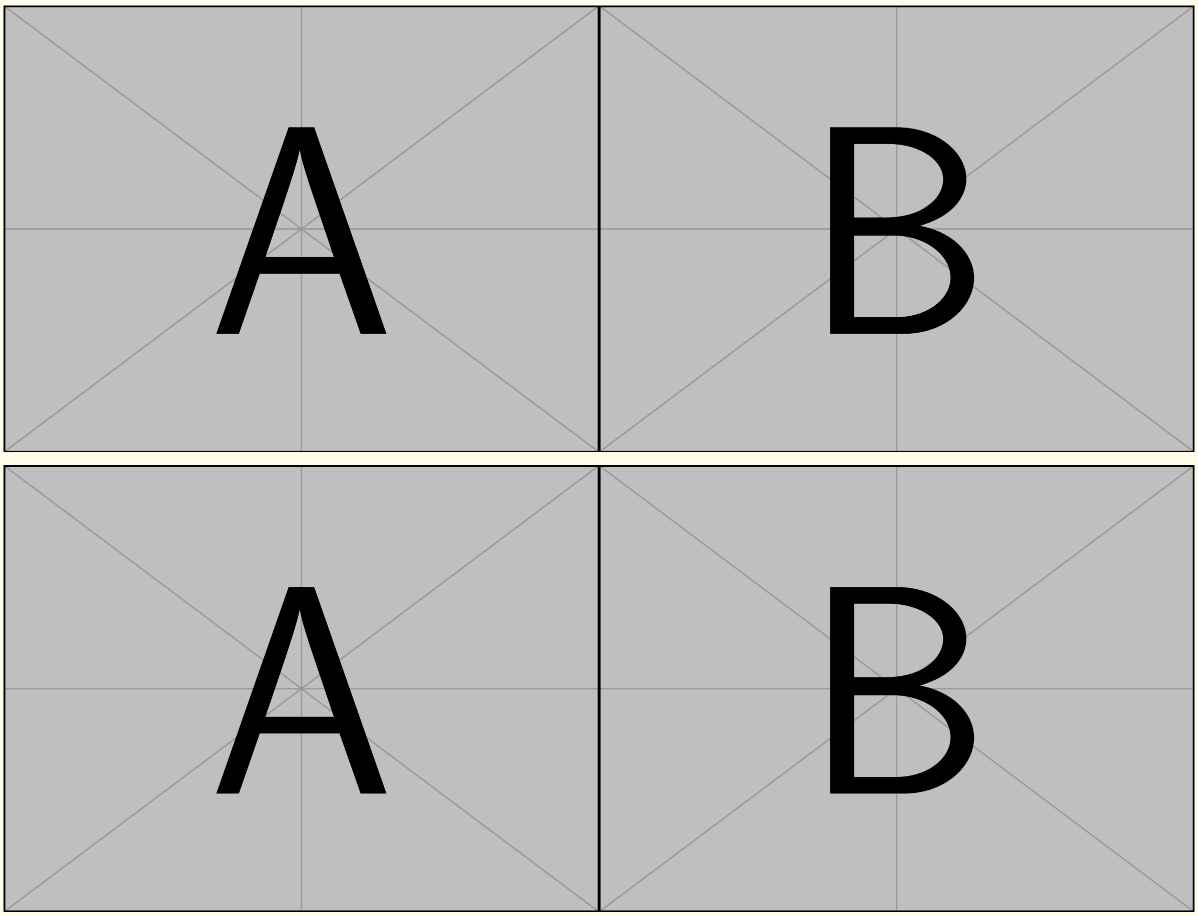
Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TA can easily run and verify your results. Make sure your code runs!

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

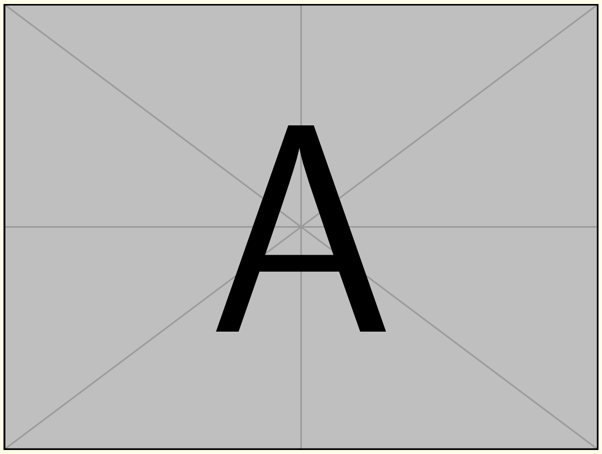
Exercise 1: Vision Transformers (10 pts)

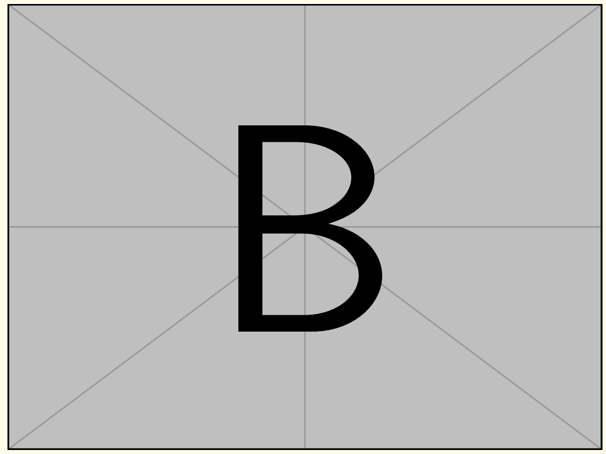
Please follow the instructions of this `ipynb` file.

1. (1+3+2 = 6 pts) Complete the missing coding parts in the provided `ipynb` file.
2. (1 pt) Visualization of patches:



3. (1 pt) The test accuracy I obtained on MNIST is: xxx%
4. (2 pts) Training / Validation accuracy vs. epoch:





Exercise 2: Adaboost (8 pts)

In this exercise we will implement Adaboost. Recall that Adaboost aims at minimizing the exponential loss:

$$\min_{\mathbf{w}} \sum_i \exp\left(-y_i \sum_j w_j h_j(\mathbf{x}_i)\right), \tag{1}$$

where h_j are the so-called weak learners, and the combined classifier

$$h_{\mathbf{w}}(\mathbf{x}) := \sum_j w_j h_j(\mathbf{x}). \tag{2}$$

Note that we assume $y_i \in \{\pm 1\}$ in this exercise, and we simply take $h_j(\mathbf{x}) = \mathrm{sign}(\pm x_j + b_j)$ for some $b_j \in \mathbb{R}$. Upon defining $M_{ij} = y_i h_j(\mathbf{x}_i)$, we may simplify our problem further as:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathbf{1}^\top \exp(-M\mathbf{w}), \tag{3}$$

where exp is applied component-wise and $\mathbf{1}$ is the vector of all 1s.

Recall that $(s)_+ = \max\{s, 0\}$ is the positive part while $(s)_- = \max\{-s, 0\} = |s| - s_+$.

Algorithm 1: Adaboost.

Input: $M \in \mathbb{R}^{n\times d}$, $\mathbf{w}_0 = \mathbf{0}_d$, $\mathbf{p}_0 = \mathbf{1}_n$, $\mathsf{max\_pass} = 300$

Output: $\mathbf{w}$

for $t = 0, 1, 2, \ldots, \mathsf{max\_pass}$ **do**

- $\mathbf{p}_t \leftarrow \mathbf{p}_t / (\mathbf{1}^\top \mathbf{p}_t)$ `// normalize`
- $\boldsymbol{\epsilon}_t \leftarrow (M)_-^\top \mathbf{p}_t$ `//` $(\cdot)_-$ `applied component-wise`
- $\boldsymbol{\gamma}_t \leftarrow (M)_+^\top \mathbf{p}_t$ `//` $(\cdot)_+$ `applied component-wise`
- $\boldsymbol{\beta}_t \leftarrow \frac{1}{2}(\ln \boldsymbol{\gamma}_t - \ln \boldsymbol{\epsilon}_t)$ `//` ln `applied component-wise`
- choose $\boldsymbol{\alpha}_t \in \mathbb{R}^d$ `// decided later`
- $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \boldsymbol{\alpha}_t \odot \boldsymbol{\beta}_t$ `//` $\odot$ `component-wise multiplication`
- $\mathbf{p}_{t+1} \leftarrow \mathbf{p}_t \odot \exp(-M(\boldsymbol{\alpha}_t \odot \boldsymbol{\beta}_t))$ `// exp applied component-wise`

1. (2 pts) We claim that Algorithm 1 is indeed the celebrated Adaboost algorithm if the following holds:

   - $\boldsymbol{\alpha}_t$ is one-hot (i.e., 1 at some entry and 0 everywhere else), namely, it indicates which weak classifier is chosen at iteration t.
   - $M \in \{\pm 1\}^{n\times d}$, i.e., if all weak classifiers are $\{\pm 1\}$-valued.

   With the above conditions, prove that (a) $\boldsymbol{\gamma}_t = \mathbf{1} - \boldsymbol{\epsilon}_t$, and (b) the equivalence between Algorithm 1 and the Adaboost algorithm in class. [Note that our labels here are $\{\pm 1\}$ and our $\mathbf{w}$ may have nothing to do with the one in class.]

Ans: Clearly, $\boldsymbol{\epsilon}_t + \boldsymbol{\gamma}_t = |M|^\top \mathbf{p}_t = \mathbf{1}$ since $\mathbf{p}_t$ is normalized and $|M|_{ij} \equiv 1$.

Thus, $\boldsymbol{\beta}_t = \frac{1}{2}\ln\frac{1-\boldsymbol{\epsilon}_t}{\boldsymbol{\epsilon}_t}$. Let $\boldsymbol{\alpha} = \mathbf{e}_j$ (i.e., choosing the j-th classifier), then

$$[\exp(-M(\boldsymbol{\alpha}_t \odot \boldsymbol{\beta}_t))]_i = \exp(-y_i h_j(\mathbf{x}_i)\beta_{jt}) = \exp(-y_i h_j(\mathbf{x}_i)\tfrac{1}{2}\ln\tfrac{1-\epsilon_{j,t}}{\epsilon_{j,t}}) \tag{4}$$

$$= \left[\frac{\epsilon_{j,t}}{1-\epsilon_{j,t}}\right]^{\frac{1}{2}y_i h_j(\mathbf{x}_i)} \tag{5}$$

$$= \left[\frac{\epsilon_{j,t}}{1-\epsilon_{j,t}}\right]^{-\frac{1}{2}} \cdot \begin{cases} \left[\frac{\epsilon_{j,t}}{1-\epsilon_{j,t}}\right]^1, & \text{if } h_j(\mathbf{x}_i) = y_i \\ \left[\frac{\epsilon_{j,t}}{1-\epsilon_{j,t}}\right]^0, & \text{if } h_j(\mathbf{x}_i) \neq y_i \end{cases}. \tag{6}$$

Note that the first factor does not depend on i, and hence will be canceled out in the normalization step. So the update on $\mathbf{p}_t$ is exactly the same as the one in class.

2. (2 pts) Let us derive each week learner h_j. Consider each feature in turn, we train d linear classifiers that each aims to minimize the weighted training error:

$$\min_{b_j\in\mathbb{R}, s_j\in\{\pm1\}} \sum_{i=1}^n p_i [\![y_i(s_j x_{ij} + b_j) \leq 0]\!], \tag{7}$$

where the weights $p_i \geq 0$ and $\sum_i p_i = 1$. Find (with justification) an optimal value for each b_j and s_j. [If multiple solutions exist, you can use the middle value.] If it helps, you may assume p_i is uniform, i.e., $p_i \equiv \frac{1}{n}$.

Ans: W.l.o.g. we may let $s_j = 1$, and we split the sum into two parts:

$$\min_{b_j} \sum_{i:y_i=1} p_i [\![b_j \leq -s_j x_{ij}]\!] + \sum_{i:y_i=-1} p_i [\![b_j \geq -s_j x_{ij}]\!]. \tag{8}$$

We sort $\{-s_j x_{ij}\}$ increasingly and arrange $\{p_i\}$ accordingly. Let $\ell_1 = \sum_{i:y_i=1} p_i$. We traverse the list of x_{ij} and perform the update

$$\ell_{i+1} \leftarrow \ell_i - y_i p_i. \tag{9}$$

Finally, we find

$$I_j := \operatorname*{argmin}_{i=1,\ldots,n,n+1} \ \ell_i \tag{10}$$

and set (with $\epsilon > 0$ being any number)

$$b_j = \begin{cases} -s_j x_{1,j} - \epsilon, & \text{if } I_j = 1 \\ -s_j x_{n,j} + \epsilon, & \text{if } I_j = n+1 \\ \frac{-s_j x_{I_j-1,j} - s_j x_{I_j,j}}{2}, & \text{otherwise} \end{cases}. \tag{11}$$

3. (2 pts) [Parallel Adaboost.] Implement Algorithm 1 with the following choices:
   - $\boldsymbol{\alpha}_t \equiv \mathbf{1}$
   - pre-process M by dividing a constant so that for all i (row), $\sum_j |M_{ij}| \leq 1$.

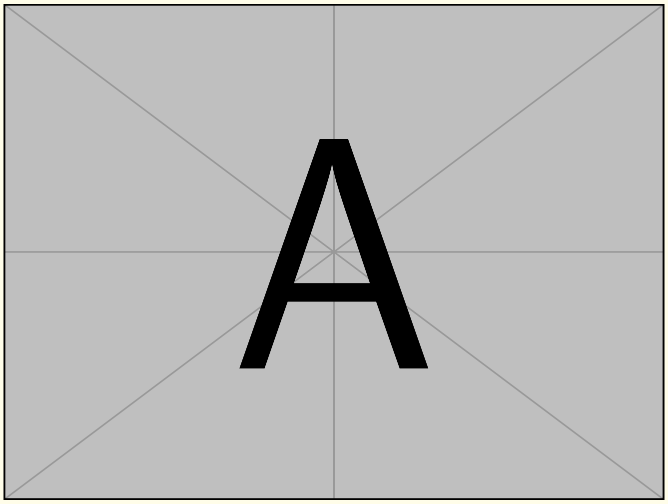
   Run your implementation on the default dataset (available on course website), and report the training loss in (3), training error, and test error w.r.t. the iteration t, where

$$\text{error}(\mathbf{w}; \mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} [\![y h_{\mathbf{w}}(\mathbf{x}) \leq 0]\!]. \tag{12}$$

   [Recall that $h_{\mathbf{w}}(\mathbf{x})$ is defined in (2) while each h_j is decided in Ex 2.2 with uniform weight $p_i \equiv \frac{1}{n}$. In case you fail to determine h_j, in Ex 2.3 and Ex 2.4 you may simply use $h_j(\mathbf{x}) = \text{sign}(x_j - m_j)$ where m_j is the median value of the j-th feature in the training set.]

   [Note that $\mathbf{w}_t$ is dense (i.e., using all weak classifiers) even after a single iteration.]

   Ans: We report all 3 curves in one figure, with clear coloring and legend to indicate which curve is which.



4. (2 pts) [Sequential Adaboost.] Implement Algorithm 1 with the following choice:

   - $j_t = \mathrm{argmax}_j \, |\sqrt{\epsilon_{t,j}} - \sqrt{\gamma_{t,j}}|$ and $\boldsymbol{\alpha}_t$ has 1 on the j_t-th entry and 0 everywhere else.

   Run your implementation on the default dataset (available on course website), and report the training loss in (3), training error, and test error in (12) w.r.t. the iteration t.

   [Note that $\mathbf{w}_t$ has at most t nonzeros (i.e., weak classifiers) after t iterations.]

   Ans: We report all 3 curves in one figure, with clear coloring and legend to indicate which curve is which.