Take-Home Quiz Three Solutions
ECE 271B - Winter 2022
Department of Electrical and Computer Engineering
University of California, San Diego

Problem 1.

a) The derivative of the empirical risk at point g_t and along direction u is

$$\begin{aligned} D_u R_{emp}[g_t] &= \left[\frac{d}{d\epsilon} R_{emp}[g_t + \epsilon u]\right]_{\epsilon=0} \\ &= \frac{1}{n}\sum_i \left[\frac{d}{d\epsilon}\phi(y_i g_t(\mathbf{x}_i) + \epsilon y_i u(\mathbf{x}_i))\right]_{\epsilon=0} \\ &= \frac{1}{n}\sum_i \phi'(y_i g_t(\mathbf{x}_i)) y_i u(\mathbf{x}_i). \end{aligned}$$

Assuming that $\phi(\cdot)$ is a decreasing function, the boosting weights are

$$\omega_i = -\phi'(y_i g_t(\mathbf{x}_i))$$

and the weak learner selection step is

$$\alpha_t = \arg\max_{u\in U} \sum_i \omega_i y_i u(\mathbf{x}_i).$$

b) The Perceptron loss

$$\phi(v) = \max(-v, 0)$$

has derivative

$$\phi'(v) = \begin{cases} -1, & v < 0 \\ 0, & v \geq 0. \end{cases}$$

Hence, the boosting weights are

$$\omega_i(v) = \begin{cases} 1, & y_i g_t(\mathbf{x}_i) < 0 \\ 0, & y_i g_t(\mathbf{x}_i) \geq 0 \end{cases}$$

and the weak learner selection step is

$$\alpha_t = \arg\max_{u\in U} \sum_{i|y_i g_t(\mathbf{x}_i)<0} y_i u(\mathbf{x}_i).$$

c) In this case,

$$\alpha_t = \arg\max_{\mathbf{w}} \sum_{i|y_i g_t(\mathbf{x}_i)<0} y_i \mathbf{w}^T \mathbf{x}_i \quad s.t. \quad ||\mathbf{w}|| = 1.$$

To perform the optimization, we use the Lagrangian

$$L(\mathbf{w}, \lambda) = \sum_{i|y_i g_t(\mathbf{x}_i)<0} y_i \mathbf{w}^T \mathbf{x}_i + \lambda(||\mathbf{w}||^2 - 1).$$

Setting the gradient to zero, we get

$$\nabla_{\mathbf{w}} L(\mathbf{w}, \lambda) = \sum_{i|y_i g_t(\mathbf{x}_i)<0} y_i \mathbf{x}_i + 2\lambda \mathbf{w} = 0,$$

from which

$$\mathbf{w}^* = -\frac{1}{2\lambda} \sum_{i|y_i g_t(\mathbf{x}_i)<0} y_i \mathbf{x}_i = -\frac{1}{2\lambda} \mathbf{v}_t,$$

where

$$\mathbf{v}_t = \sum_{i|y_i g_t(\mathbf{x}_i)<0} y_i \mathbf{x}_i.$$

Using the constraint that $\mathbf{w}$ has unit norm, it follows that

$$\lambda = \frac{1}{2}||\mathbf{v}_t||,$$

$$\mathbf{w}^* = -\frac{1}{||\mathbf{v}_t||} \mathbf{v}_t$$

and

$$\alpha_t(\mathbf{x}) = -\frac{1}{||\mathbf{v}_t||} \mathbf{v}_t^T \mathbf{x}.$$

The step-size is then

$$\begin{aligned} w_t &= \arg\min_w R_{emp}[g_t + w\alpha_t] \\ &= \arg\min_w \sum_i \phi\left(y_i g_t(\mathbf{x}_i) - \frac{w}{||\mathbf{v}_t||} y_i \mathbf{v}_t^T \mathbf{x}_i \right). \end{aligned}$$

Finally, the update is

$$g_{t+1}(\mathbf{x}) = g_t(\mathbf{x}) + w_t \alpha_t(\mathbf{x}).$$

d) Note that, since

$$g_{t+1}(\mathbf{x}) = \sum_{i=0}^{t} w_i \alpha_i(\mathbf{x}) = -\sum_{i=0}^{t} \frac{w_i}{||\mathbf{v}_i||} \mathbf{v}_i^T \mathbf{x},$$

it follows that

$$g_{t+1}(\mathbf{x}) = \mathbf{z}_{t+1}^T \mathbf{x},$$

with

$$\mathbf{z}_{t+1} = -\sum_{i=0}^{t} w_i \frac{\mathbf{v}_i}{||\mathbf{v}_i||} = \mathbf{z}_t - w_t \frac{\mathbf{v}_t}{||\mathbf{v}_t||}. \quad (1)$$

In summary, each iteration of boosting computes $\mathbf{v}_t$ by summing $y_i\mathbf{x}_i$ for all points in the set

$$E_t = \{\mathbf{x}_i | y_i g_t(\mathbf{x}_i) < 0\}$$

and then updates $\mathbf{z}$ using (1). On the other hand, each iteration of the Perceptron computes $y_i\mathbf{x}_i$ for each point in E_t and updates $\mathbf{w}$ with

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_i \mathbf{x}_i.$$

Hence, the algorithms are the same, up to the following differences.

1. While boosting sums over all points in E_t and updates, the Perceptron updates after each point. This is the difference between gradient descent (boosting) and stochastic gradient descent (Perceptron). In this sense, the Perceptron is a stochastic gradient descent version of boosting.
2. The boosting update uses a different step-size w_t per iteration, while the Perceptron uses the same step of η in all iterations.

Note that both stochastic descent and variable step-sizes improve the convergence of the algorithm. However, in general, you cannot combine both. So, when all is said and done, the Perceptron is a variant of boosting for a particular loss function (Perceptron loss) and a particular class of weak learners (of the form $\mathbf{w}^T\mathbf{x}$).

e) No. In this case, the loss is

$$\phi(v) = \begin{cases} 1, & v < 0 \\ 0, & v \geq 0 \end{cases}$$

and has the negative Dirac delta as derivative

$$\phi'(v) = -\delta(v) = \begin{cases} -\infty, & v = 0 \\ 0, & v \neq 0. \end{cases}$$

Hence, the boosting weights are

$$\omega_i(v) = \begin{cases} \infty, & y_i g_t(\mathbf{x}_i) = 0 \\ 0, & y_i g_t(\mathbf{x}_i) \neq 0 \end{cases}$$

and obviously ill-conditioned. Points such that $g(\mathbf{x}_i) = 0$ are the only ones to receive non-zero weight. Besides the numerical difficulties of infinite weights (which one could maybe replace by a finite constant), it would be very easy for the algorithm to "run out of examples" after just one or two boosting iterations. Hence, the algorithm would not work well in general.

Problem 2.

a) The entropy loss is

$$\begin{aligned} L(\mathbf{x}, z) &= -z \log \sigma[g(\mathbf{x})] - (1-z) \log(1 - \sigma[g(\mathbf{x})]) \\ &= z \log(1 + e^{-g(\mathbf{x})}) + (1-z) \log(1 + e^{g(\mathbf{x})}) \\ &= \begin{cases} \log(1 + e^{-g(\mathbf{x})}), & z = 1 \\ \log(1 + e^{g(\mathbf{x})}), & z = 0. \end{cases} \end{aligned}$$

It follows, from $y = 2z - 1$, that

$$\begin{aligned} L(\mathbf{x}, y) &= \begin{cases} \log(1 + e^{-g(\mathbf{x})}), & y = 1 \\ \log(1 + e^{g(\mathbf{x})}), & y = -1 \end{cases} \\ &= \log(1 + e^{-yg(\mathbf{x})}). \end{aligned}$$

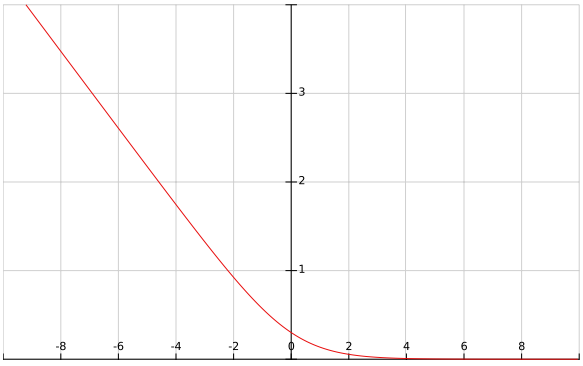
Note that this is a margin loss, since

$$L(\mathbf{x}, y) = \phi(yg(\mathbf{x}))$$

with

$$\phi(v) = \log(1 + e^{-v})$$

is the margin loss shown below.



Furthermore, because ϕ is decreasing, $\phi(v) \to 0$ as $v \to \infty$, and $\phi(0) = \log(2) > 0$, ϕ is a margin enforcing loss. This implies that neural networks trained with this loss are likely to create large margins.

b) For the loss of **a)**,

$$\phi'(v) = -\frac{e^{-v}}{1 + e^{-v}} = -(1 - \sigma(v))$$

where

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

is the sigmoid non-linearity. Hence, the boosting weights are

$$\omega_i = 1 - \sigma(y_i g_t(\mathbf{x}_i))$$

and the weak learner selection step is

$$\alpha_t = \arg\max_{u \in U} \sum_i \omega_i y_i u(\mathbf{x}_i).$$

The optimal step-size is

$$w_t = \arg\min_w \sum_i \log(1 + e^{-y_i(g_t(\mathbf{x}_i + w\alpha_t(\mathbf{x}_i)}])$$

and the ensemble learner is updated to

$$g_{t+1}(\mathbf{x}) = g_t(\mathbf{x}) + w_t\alpha_t(\mathbf{x}).$$

c) The algorithms are identical, up to the weighting function. In both cases, we have

$$\omega_i = -\phi'(y_i g_t(\mathbf{x}_i))$$

with

$$-\phi'(v) = 1 - \sigma(v) \text{ for the entropy loss}$$

and

$$-\phi'(v) = e^{-v} \quad \text{for the exponential loss.}$$

As shown above, the two weighting functions differ significantly only for large negative v. While the weighting function of the entropy loss is constant, that of the exponential loss grows exponentially. This implies that, for the exponential loss, examples $\mathbf{x}_i$ such that $y_i g_t(\mathbf{x}_i)$ is very negative will become dominant on the weak learner selection stage. A very negative margin $y_i g_t(\mathbf{x}_i)$ typically signals outliers, i.e. points that cannot be correctly classified anyway. Hence, if the data has outliers, these can become dominant and the learning algorithm will work very hard to accommodate them. This usually results in poor performance for points that are not outliers, which are the ones we care the most about. In result, the algorithm will not be robust. For this reason, the entropy loss (used in neural nets) is frequently preferred to the exponential loss of AdaBoost. In fact, the boosting algorithm that we have just derived is commonly used and known as LogitBoost.

Problem 3.

a) There are $T \times d \times 2$ weak learners in U. The algorithm has to search through all of these at each boosting iteration. Hence, for B iterations, the algorithms has complexity $O(BTd)$.

b) We have seen that the LDA projection vector $\mathbf{w}$ is the largest eigenvector of $[\mathbf{S}_W + \gamma\mathbf{I}]\mathbf{S}_B$, where

$$\begin{aligned} \mathbf{S}_W &= \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_{-1}, \\ \mathbf{S}_B &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T, \end{aligned}$$

and γ a regularization constant. Hence, we just have to estimate the means and covariances, for which we use sample means and covariances. Note, however, that we must account for the sample weights ω_i. We start by normalizing the weights, i.e. compute

$$\beta_k = \frac{\omega_k}{\sum_j \omega_j},$$

so that they add up to 1. Next, for $i \in \{-1, 1\}$, we use the estimates

$$\begin{aligned} \boldsymbol{\mu}_i &= \sum_{k|y_i=i} \beta_k \mathbf{x}_k \\ \boldsymbol{\Sigma}_i &= \sum_{k|y_i=i} \beta_k (\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T. \end{aligned}$$

c) The weak learner computes the Gaussian BDR

$$u(\mathbf{x}) = \arg \max_{i \in \{-1,1\}} \mathcal{G}(\mathbf{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\pi_i$$

as in **b)**. The probabilities π_i are estimated with

$$\pi_i = \sum_{k|y_i=i} \beta_k.$$

Problem 4.

This problem has empirical risk

$$R_{emp}[g_n] = \frac{1}{n}\sum_i (f(t_i) - g_n(t_i))^2.$$

The derivative of the empirical risk at point g_n along direction u is

$$\begin{aligned} D_u R_{emp}[g_n] &= \left[\frac{d}{d\epsilon} R_{emp}[g_n + \epsilon u]\right]_{\epsilon=0} \\ &= \frac{1}{n}\sum_i \left[\frac{d}{d\epsilon}(f(t_i) - g_n(t_i) - \epsilon u(t_i))^2\right]_{\epsilon=0} \\ &= -\frac{1}{n}\sum_i 2(f(t_i) - g_n(t_i))u(t_i). \end{aligned}$$

Hence, the derivative has maximum magnitude if we choose the weak learner

$$\alpha_n(t) = \arg\max_{\alpha(t)\in D} |\sum_i (f(t_i) - g_n(t_i))\alpha(t_i)|.$$

The step-size is then

$$\begin{aligned} w_n &= \arg\min_w R_{emp}[g_n + w\alpha_n] \\ &= \arg\min_w \frac{1}{n}\sum_i (f(t_i) - g_n(t_i) - w\alpha_n(t_i))^2. \end{aligned}$$

Seting the derivative with respect to w to zero, we obtain

$$-2\sum_i (f(t_i) - g_n(t_i) - w\alpha_n(t_i))\alpha_n(t_i) = 0$$

or

$$w_n = \frac{\sum_i (f(t_i) - g_n(t_i))\alpha_n(t_i)}{\sum_i \alpha_n^2(t_i)},$$

and, since the second-derivative is $\sum_i \alpha_n(t_i)^2 \geq 0$, the risk is minimized by w_n.

Finally, the update is

$$g_{n+1}(t) = g_n(t) + w_n\alpha_n(t).$$

Noting that, at each iteration n, the residual is

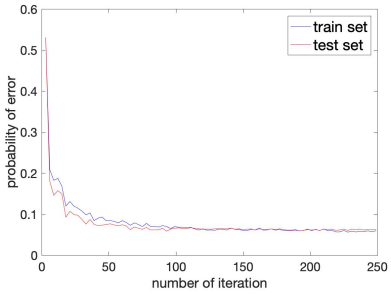
$$r_n(t) = f(t) - g_n(t),$$

the steps of the algorithm can be written as

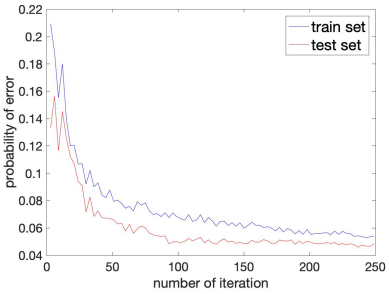
- choose weak learner $\alpha_n(t) = \arg\max_{\alpha(t)\in D} |\sum_i r_n(t_i)\alpha(t_i)|$
- choose step $w_n = \frac{\sum_i r_n(t_i)\alpha_n(t_i)}{\sum_i \alpha_n^2(t_i)}$
- update residual $r_{n+1}(t) = f(t) - g_{n+1}(t) = r_n(t) - w_n\alpha_n(t)$.

Problem 5.

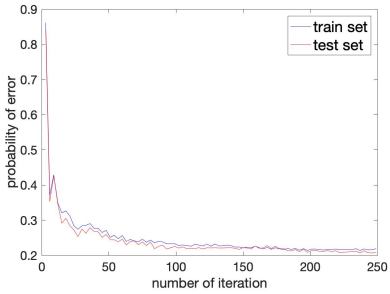
a) The plots of training and test error are shown below. The test error of the final classifier is 9.13%.



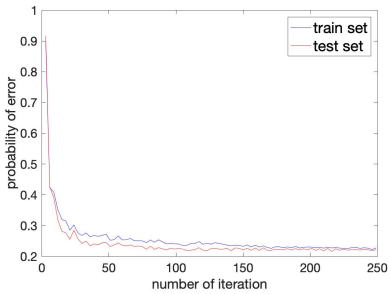
digit 0



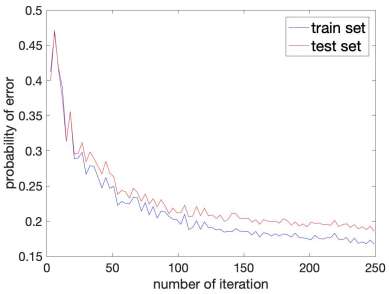
digit 1



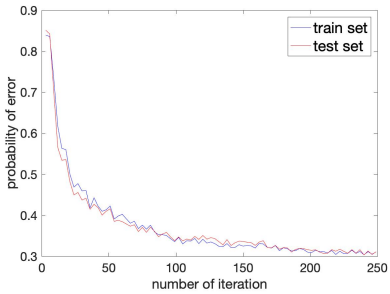
digit 2



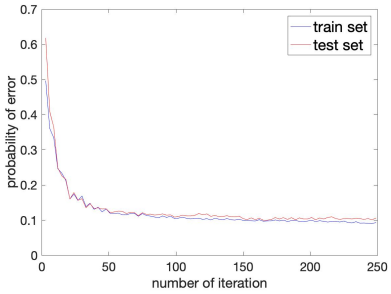
digit 3



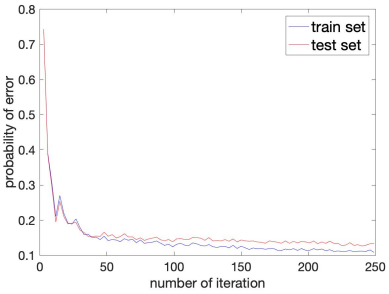
digit 4



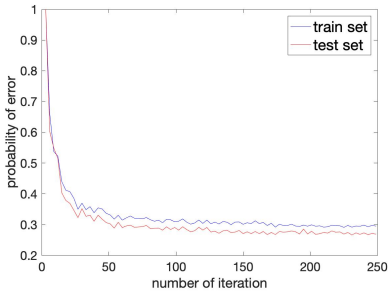
digit 5



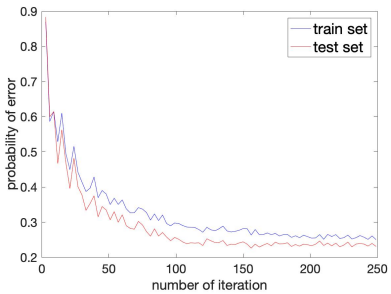
digit 6



digit 7

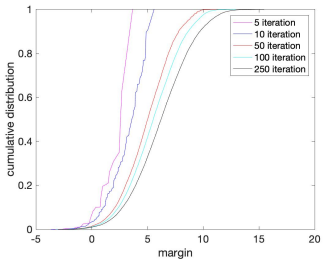


digit 8

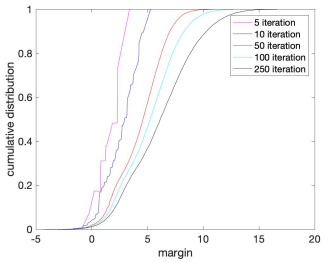


digit 9

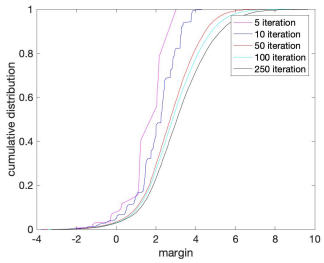
b) The plots of the cdf of the margin are shown below. Note that the behavior is the same in all cases. The curve is always sigmoidal shaped (as one would expect for a cdf), has a sharp transition for few iterations, and a much slower transition for large iterations. Note that, after 5 iterations, almost all examples have margins in $[-2, 2]$. However, as boosting progresses, the margins increase, especially on the positive side. The curves tilt more to the side of positive margins. At 250 iterations, the positive range of margins increases substantially. Note that the number of points correctly classified is increasing (error rate decreases, as we saw in **a)**) and these points have larger margin. This implies that the classifier is much more confident that they are well classified.



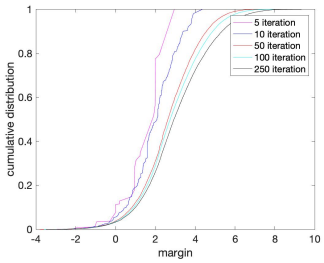
digit 0



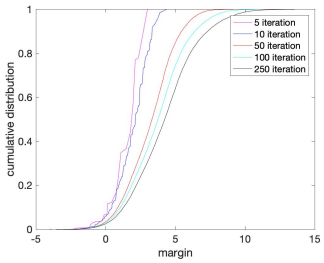
digit 1



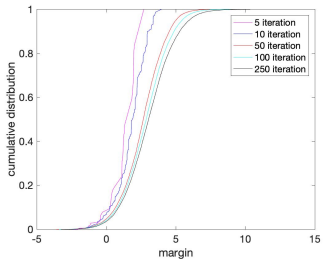
digit 2



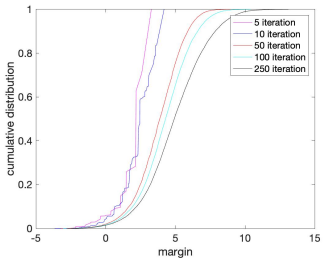
digit 3



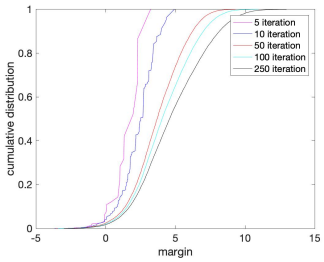
digit 4



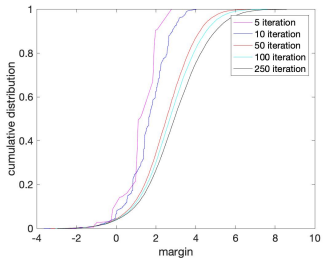
digit 5



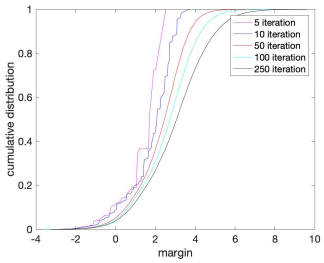
digit 6



digit 7

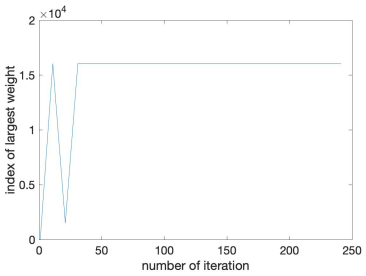


digit 8

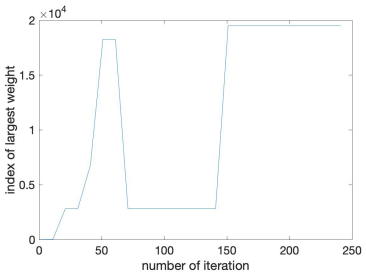


digit 9

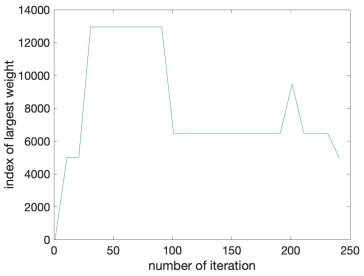
c) The figure below shows the index of the example of largest weight for each boosting iteration, per digit. The three digits selected more frequently as having the largest weight are shown below the plot. Note that these tend to either be difficult examples of the digit or other digits that have a lot of common pixels with the digit. These examples are the digits that pose most difficulty to the learning algorithm as it forms the decision boundary for the classification of the digit.



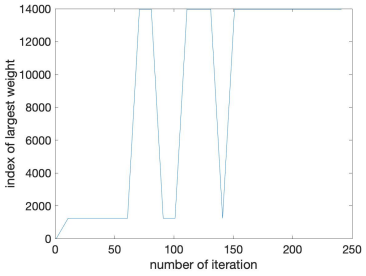
digit 0



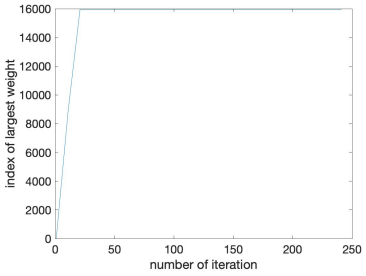
digit 1



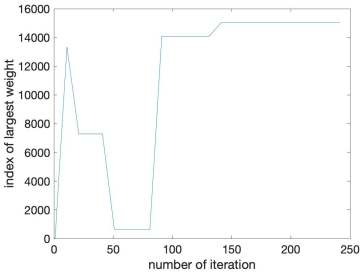
digit 2



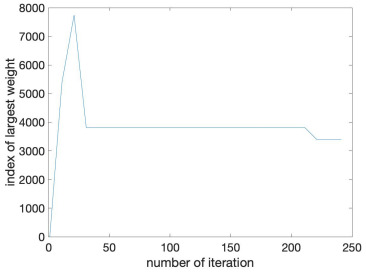
digit 3



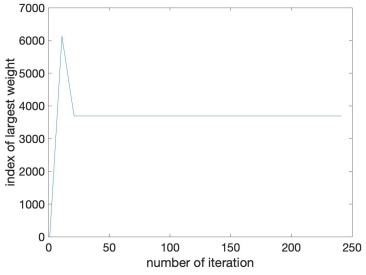
digit 4



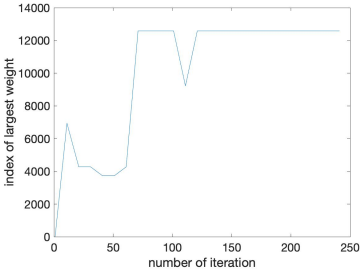
digit 5



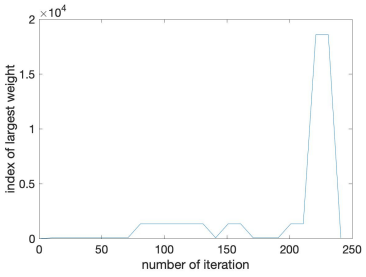
digit 6



digit 7



digit 8



digit 9

d) The arrays recovered for digits 0 to 9 are the ones shown below. Note that most of the weak learners threshold pixels on the digit boundary. If you look closely, you can see some information about the shape of the digits. This effect would likely be more pronounced for more sophisticated weak learners.