Homework Set Three
ECE 271B - Winter 2019
Department of Computer and Electrical Engineering
University of California, San Diego

In this homework set, we study the boosting algorithm. This is an iterative procedure that, given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i \in \{-1, 1\}$, and a definition of empirical risk

$$R_{emp}[g] = \frac{1}{n} \sum_i \phi(y_i g(\mathbf{x}_i)), \tag{1}$$

implements the following steps at iteration t:

- **dataset reweighting:** each training example $\mathbf{x}_i$ receives a non-negative weight ω_i;
- **weak learner selection:** a weak learner $\alpha_t(\mathbf{x})$ is selected from a set of functions U, so as to optimize a function that depends on all examples $\mathbf{x}_i$ and their weights ω_i;
- **step size:** a step size w_t is computed according to

$$w_t = \arg\min_w R_{emp}[g_t + w\alpha_t]; \tag{2}$$

- **update:** the ensemble learner is updated to

$$g_{t+1}(\mathbf{x}) = g_t(\mathbf{x}) + w_t \alpha_t(\mathbf{x}). \tag{3}$$

Problem 1. In this problem, we consider a monotonically decreasing loss function $\phi(\cdot)$.

a) Derive the equations for the reweighting and weak learner selection steps of the boosting algorithm. Note that, to receive full credit, you must derive the most detailed equation possible. For example, what is an expression for the weights?

b) In class, we introduced the Perceptron loss

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

What is the expression of the boosting weights ω_i for this loss?

c) Assume that the Perceptron loss is used and the set U is the set of functions

$$u(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

parametrized by a unit vector $\mathbf{w}$ (i.e. $||\mathbf{w}|| = 1$) with the dimension of $\mathbf{x}$. Derive the boosting algorithm for this scenario.

d) Compare the algorithm of **c)** to the Perceptron learning algorithm. Give a detailed discussion of the similarities and differences.

e) Consider the generic boosting algorithm of **a)**. Could this algorithm work with the 0/1-loss?

Problem 2. We have seen that neural networks are frequently trained with the entropy loss

$$L(\mathbf{x}, z) = -z \log \sigma[g(\mathbf{x})] - (1 - z) \log(1 - \sigma[g(\mathbf{x})]),$$

where $z \in \{0, 1\}$ is the class label of example $\mathbf{x}$ and

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

is the sigmoid non-linearity.

a) Note that labels $z \in \{0, 1\}$ can be converted to labels $y \in \{-1, 1\}$ with the simple transformation $y = 2z - 1$. Show that, when this relationship holds, the entropy loss can be written as

$$L(\mathbf{x}, y) = \log(1 + e^{-yg(\mathbf{x})}). \tag{4}$$

Using this equivalence, what can you say about the ability of neural networks to enforce margins? Are they likely to produce classifiers of large margin? Why?

b) Derive the boosting algorithm of **1a)** for the loss function of (4). Note that, to receive full credit, you must derive the most detailed equation possible.

c) In class, we have derived a similar algorithm for the exponential loss, i.e. for $\phi(v) = e^{-v}$. Compare the two algorithms and discuss how they differ. For which type of problems would it make a difference to use one algorithm vs. the other? Why?

Problem 3. In this problem, we consider the AdaBoost algorithm with classification weak learners, i.e. $\alpha_t(\mathbf{x}) \in \{-1, 1\}$.

a) Consider the implementation of the algorithm using decision stumps as weak learners, i.e.

$$u(\mathbf{x}, j, t) = \begin{cases} 1, & x_j \geq t \\ -1, & x_j < t. \end{cases}$$

Let U be the set of all weak learners such that t belongs to a pre-defined set of T thresholds. Assume that, for each weak learner, there is also a weak learner of opposite polarity, i.e. $u'(\mathbf{x}, j, t) = -u(\mathbf{x}, j, t)$. For $\mathbf{x} \in R^d$, what is the complexity of the boosting algorithm?

b) Rather than decision stumps, it was decided to implement weak learners as

$$u(\mathbf{x}; t) = \text{sgn}[\mathbf{w}^T \mathbf{x} - t],$$

where $\mathbf{w}$ is computed by linear discriminant analysis. Derive all equations needed to implement the weak learner selection step.

c) Repeat **b)** for weak learners implemented with a Gaussian classifier, i.e. that implement the BDR under the assumption of two Gaussian classes of mean μ_i, covariance $\mathbf{\Sigma}_i$, and probability $\pi_i, i \in \{-1, 1\}$. The parameters of the Gaussians are learned by maximum likelihood.

Problem 4. A classical problem of signal processing is the problem of decomposing a signal $f(t)$ into a combination of functions $\alpha_n(t)$ so that

$$f(t) = \sum_n w_n \alpha_n(t).$$

When the functions $\alpha_n(t)$ form an orthonormal basis of the space of functions $f(t)$, this is fairly straightforward. It suffices to compute the projections

$$w_n = \int f(t)\alpha_n(t)dt.$$

For example, this is the case of the Fourier series expansion, where the basis functions $\alpha_n(t)$ are cosines of multiple frequencies. The problem is, however, more complex when the functions $\alpha_k(t)$ form an overcomplete dictionary D, i.e. they are not linearly independent.

A commonly used algorithm to solve this problem is the *matching pursuit* algorithm. A residual function $r(t)$ is first initialized with $r_0(t) = f(t)$. The algorithm then iterates between the following steps:

- find
$$\alpha_n(t) = \arg\max_{\alpha(t)\in D} \left| \int r_n(t)\alpha(t)dt \right|;$$
- compute
$$w_n = -\left| \frac{\int r_n(t)\alpha_n(t)dt}{\int \alpha_n^2(t)dt} \right|;$$
- update the residual
$$r_{n+1}(t) = r_n(t) - w_n\alpha_n(t).$$

For a discrete-time signal, sampled at times t_i, the dot-products become summations, e.g.

$$\alpha_n(t) = \arg\max_{\alpha(t)\in D} \left| \sum_i r_n(t_i)\alpha(t_i) \right|.$$

Show that the matching pursuit algorithm is really just an implementation of boosting on a training set $\{(t_i, f(t_i))\}$ with weak learners $\alpha_k(t)$ and loss function $L(y, g(x)) = (y - g(x))^2$.

Problem 5. In this problem, we will use boosting to, once again, classify digits. In all questions, you will use the training set contained in the directory `MNISTtrain` (60,000 examples) and the test set in the directory `MNISTtest` (10,000). To reduce training time, **we will only use 20,000 training examples**. To read the data, you should use the script `readMNIST.m` (use readDigits=20,000 or readDigits=10,000 respectively, and offset=0). This returns two matrices. The first (`imgs`) is a matrix with $n \in \{10000, 20000\}$ rows, where each row is a 28×28 image of a digit, vectorized into a 784-dimensional row vector. The second (`labels`) is a matrix with $n \in \{10000, 20000\}$ rows, where each row contains the class label for the corresponding image in `imgs`. Since there are 10 digit classes, we will learn 10 binary classifiers. Each classifier classifies one class against all others. For example, classifier 1 assigns label $Y = 1$ to the images of class 1 and label $Y = -1$ to images of all other classes. Boosting is then used to learn an ensemble rule

$$h_1(\mathbf{x}) = \text{sgn}[g_1(\mathbf{x})] \qquad g_1(\mathbf{x}) = \sum_t w_t \alpha_t(\mathbf{x}).$$

After we learn the 10 rules $g_i(\mathbf{x}), i \in \{1, \ldots, 10\}$, we assign the image to the class of largest score

$$c(\mathbf{x}) = \arg\max_i g_i(\mathbf{x}).$$

This implementation is called the "one-vs-all" architecture. We will use decision stumps

$$u(\mathbf{x}; j, t) = \begin{cases} 1 & x_j \geq t \\ -1 & x_j < t \end{cases}$$

as weak learners. For each weak learner, also consider the "twin" weak learner of opposite polarity

$$u'(\mathbf{x}; j, t) = -u(\mathbf{x}; j, t).$$

This simply chooses the "opposite label" from that selected by $u(\mathbf{x}; j, t)$. Note that $j \in \{1, \ldots, 748\}$. For thresholds, use

$$t = \frac{i}{N}, i \in \{0, \ldots, N\} \text{ and } N = 50.$$

a) Run AdaBoost for $T = 250$ iterations. For each binary classifier, plot train and test errors vs. iteration. Report the test error of the final classifier. For each iteration t, store the index of the example of largest weight, i.e. $i^* = \arg\max_i w_i^{(t)}$. At iterations $\{5, 10, 50, 100, 250\}$ store the margin $\gamma_i = \gamma(\mathbf{x}_i)$ of each example.

b) For each binary classifier, make a plot of cumulative distribution function (cdf) of the margins γ_i of all training examples after $\{5, 10, 50, 100, 250\}$ iterations (the cdf is the function $F(a) = P(\gamma \leq a)$, and you can do this by computing an histogram and then a cumulative sum of histogram bins.) Comment on what these plots tell you about what boosting is doing to the margins.

c) We now visualize the weighting mechanism. For each of the 10 binary classifiers, do the following.

- Make a plot of the index of the example of largest weight for each boosting iteration.
- Plot the three "heaviest" examples. These are the 3 examples that were most frequently selected, across boosting iterations, as the example of largest weight.

Comment on what these examples tell you about the weighting mechanism.

d) We now visualize what the weak learners do. Consider the weak learners $\alpha_t, t \in \{1, \ldots, T\}$, chosen by each iteration of AdaBoost. Let $i(t)$ be the index of the feature $\mathbf{x}_{i(t)}$ selected at time t. Note that $i(t)$ corresponds to an image (pixel) location. Create a 28×28 array $\mathbf{a}$ filled with the value 128. Then, for $\alpha_t, t \in \{1, \ldots, T\}$, do the following.

- If the weak learner selected at iteration t is a regular weak learner (outputs 1 for $x_{i(t)}$ greater than its threshold), store the value 255 on location $i(t)$ of array $\mathbf{a}$.
- If the weak learner selected at iteration t is a twin weak learner (outputs -1 for $x_{i(t)}$ greater than its threshold), store the value 0 on location $i(t)$ of array $\mathbf{a}$.

Create the array $\mathbf{a}$ for each of the 10 binary classifiers and make a plot of the 10 arrays. Comment on what the classifiers are doing to reach their classification decision.