6.867 Machine Learning

Problem set 4

Due Thursday, October 31, in class

What and how to turn in?

Turn in short written answers to the questions explicitly stated, and when requested to explain or prove. Do **not** turn in answers when requested to "think", "consider", "try" or "experiment" (except when specifically instructed). You may turn in answers to questions marked "optional"— they will be read and corrected, but a grade will not be recorded for them.

Turn in all MATLAB code explicitly requested, or that you used to calculate requested values. It should be clear exactly what command was used to get the answer to each question.

To help the graders (including yourself...), please be neat, answer the questions briefly, and in the order they are stated. Staple each "Problem" separately, and be sure to write your name on the top of every page.

Problem 1: Boosting

In this problem, we slightly modify the AdaBoost algorithm seen in class to better explore some properties of the algorithm. Specifically, we no longer normalize the weights on the training examples after each iteration. The modified algorithm, which is set to run for T iterations, is shown in Algorithm 1.

Note that in the modified version, the weights associated with the training examples are no longer guaranteed to sum to one after each iteration (and therefore can not be viewed as a "distribution"), but the algorithm is still valid. Let us denote the sum of weights at the start of iteration t by $Z_t = \sum_{i=1}^{n} w_i^{(t)}$. At the start of the first iteration of boosting, $Z_1 = n$.

Let us now investigate the behavior of Z_t, as a function of t.

1. [**10pt**] At the t^{th} iteration we found a weak classifier that achieves a weighted training error ϵ_t. Show that the choice of "votes" given by $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ is optimal in the

Algorithm 1 AdaBoost: slightly modified version from the lecture (the weights are not normalized to sum to one).

Input: Set of n labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $y_i = \pm 1$.

Set of associated weights $w_1^{(1)}, \ldots, w_n^{(1)}$, initially all $w_i^{(1)} = 1$.

Required number of iterations, T.

for $t = 1, \ldots, T$ **do**

Find a weak classifier h_t, which outputs binary predictions $h_t(\mathbf{x}) = \pm 1$, such that its weighted training error

$$\epsilon_t = \frac{1}{Z_t} \sum_{i=1}^{n} w_i^{(t)} \left(1 - \delta\left(h_t(\mathbf{x_i}), y_i\right)\right)$$

satisfies $\epsilon_t < 1/2$.

Set the vote $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.

Update the weights : for each $i = 1, \ldots, n$ set

$$w_i^{(t+1)} = w_i^{(t)} e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$$

end for

Output: the combined classifier defined by

$$\hat{h}_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

sense that it minimizes Z_{t+1}.

Advice: Look at Z_{t+1} as a function of α, and find the value for which the function achieves its minimum. You may also find the following notational shorthand useful:

$$W_I = \sum_{i=1}^{n} w_i^{(t)}\big(1 - \delta(y_i, h_t(\mathbf{x}_i))\big), \quad W_C = \sum_{i=1}^{n} w_i^{(t)}\delta(y_i, h_t(\mathbf{x}_i)),$$

where W_C is the total weight of points classified by h_t correctly, and W_I the total weight of misclassified points. $\delta(y, h_t(\mathbf{x})) = 1$ whenever the label predicted by h_t is correct and zero otherwise. The weights here are those available at the start of iteration t.

2. [**5pt**] Show that the sum of weights Z_t is monotonically decreasing as a function of t.

   We know now that Z_t decreases, and that our choice of α_t is optimal for minimizing Z_{t+1}; but is this quantity useful in any way? It is useful in bounding how successive boosting iterations reduce the training error.

3. [**10pt**] Show that the training error (the average number of misclassified training examples) of the combined classifier after m iterations of boosting,

   $$\hat{h}_m(\mathbf{x}) = \sum_{t=1}^{m} \alpha_t h_t(\mathbf{x}),$$

   is bounded from above by Z_{m+1}/n.

   Advice: In the definition of the algorithm, the weights (and therefore Z_t) are defined recursively. You will find it helpful to "unroll" the definition, and write out the value of Z_{t+1} in terms of the initial weights (which are all 1) and the subsequent updates, using $\alpha_1, \ldots, \alpha_m$. You may also find the following (simple) fact helpful: for any $x \geq 0$, $e^x \geq 1$.

   We have shown that AdaBoost tries in each iteration to minimize an upper bound on the training error of the combined classifier, and guarantees that this bound will monotonically decrease as the algorithm proceeds. How much the bound decreases at each iteration depends on the weighted training error of each weak classifier.

   In the AdaBoost algorithm presented above, the vote α_t assigned to a weak classifier h_t does not differentiate between the "directions" of mistakes it makes (misclassifying a positive example as negative or vice versa). We already know that the magnitude of the vote is related to the performance of the classifier - that is, a classifier which achieves lower weighted training error will get a higher α. But what if h_t makes almost all of its mistakes on, say, negative examples? That is, for a training example $\mathbf{x}$ if $h_t(\mathbf{x}) = -1$, with very high probability $\mathbf{x}$ is indeed negative, but if $h_t(\mathbf{x}) = +1$, it is quite likely that in fact $\mathbf{x}$ is negative.

   Recall that the vote α_t measures the influence of h_t in the final (combined) classifier. Thus, it would make sense to assign a high vote to its decisions in the "direction" in which h_t makes fewer mistakes, and a lower vote to the opposite decisions. Algorithm 2 describes the modification in AdaBoost that allows for such a refinement.

Algorithm 2 Modified boosting algorithm, allowing different votes for positive and negative decisions

Input: Set of n labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $y_i = \pm 1$.

Set of associated weights $w_1^{(1)}, \ldots, w_n^{(1)}$, initially all $w_i^{(1)} = 1$.

Required number of iterations, T.

for $t = 1, \ldots, T$ **do**

Find a weak classifier h_t, which outputs binary predictions $h_t(\mathbf{x}) = \pm 1$, such that its weighted training error

$$\epsilon_t = \frac{1}{Z_t} \sum_{i=1}^{n} w_i^{(t)} \left(1 - \delta\left(h_t(\mathbf{x_i}), y_i\right)\right)$$

satisfies $\epsilon_t < 1/2$

Set the votes α_t, β_t (see Question 5).

Update the weights : for each $i = 1, \ldots, n$ set

$$w_i^{(t+1)} = w_i^{(t)} e^{-y_i v_t(\mathbf{x}_i)}$$

where

$$v_t(\mathbf{x}) = \begin{cases} \alpha_t & \text{if } h_t(\mathbf{x}) = 1, \\ -\beta_t & \text{if } h_t(\mathbf{x}) = -1. \end{cases}$$

end for

Output: the combined classifier defined by

$$\hat{h}_T(\mathbf{x}) = \sum_{t=1}^{T} v_t(\mathbf{x})$$

---

4. [**3pt**] Express Z_{t+1} as a function of α_t, β_t and the weights at the start of t-th iteration.

5. [**12pt**] Fill in the missing part in the algorithm above. That is, find the rules for computing α_t and β_t that would minimize Z_{t+1}. You may use the following notational shorthands: $W_{++}^{(t)}$ is the total weight (under the weights active at the beginning of the t-th iteration) of the positive examples that are classified as positive by h_t; $W_{+-}^{(t)}$ is the total weight of the positive examples misclassified by h_t; and similarly defined $W_{--}^{(t)}$ and $W_{-+}^{(t)}$.

Problem 2: Complexity

Here we try to understand a bit better some implications of the fact that the feature vectors corresponding to the radial basis kernel are functions (living in an infinite dimensional space). It turns out that the VC-dimension of the classifier that uses a radial basis kernel is infinite. We can actually make a bit stronger statement: with a radial basis kernel we can perfectly separate *any* finite set of *distinct* training points. Why is this a stronger statement? To claim that the VC-dimension of a classifier is n we only need to find *some* set of n training points that we can shatter; this need not hold for all sets of n points.

To get started we'll assume that we have a set of n points in $\mathcal{R}^d$. The dimension d is assumed to be finite and, perhaps surprisingly, plays no role in the argument. We also make the critical assumption that all the training points are distinct. The radial basis kernel we use has the following form:

$$K(\mathbf{x}, \mathbf{x}') = \exp\{ -\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2 \}$$

where σ^2 is a width parameter specifying how quickly the kernel vanishes as the points move further away from each other. All of our results hold for any positive finite value of σ^2. In other words, whether or not we'll be able to perfectly separate the training points has nothing to do with the kernel width. The kernel width does affect generalization performance, however.

1. [**15pt**] Let's proceed in stages. To make things easier we are going to prove a bit stronger result than necessary. In particular, we'll show that

   $$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y_i \mathbf{w}^T \phi(\mathbf{x}_i) = 1,\ i = 1, \ldots, n$$

   has a solution regardless of how we set the ± 1 training labels y_i. You should convince yourself first that this is consistent with our goal. Here $\phi(\mathbf{x}_i)$ is the feature vector (function actually) corresponding to the radial basis kernel. Our formulation here is a bit non-standard for two reasons. We try to find a solution where *all* the points are support vectors. This is not possible for all valid kernels but makes our life easier here. We also omit the bias term since it is not needed for the result.

   Introduce Lagrange multipliers for the constraints similarly to finding the SVM solution (see also the tutorial on Lagrange multipliers distributed along with the lecture slides) and show the form that the solution $\hat{\mathbf{w}}$ has to take. You can assume that $\mathbf{w}$ and $\phi(\mathbf{x}_i)$ are finite vectors for the purposes of these calculations. Note that the Lagrange multipliers here are no longer constrained to be positive. Since you are trying to satisfy equality constraints, the Lagrange multipliers can take any real value.

   We are after $\hat{\mathbf{w}}$ as a function of the Lagrange multipliers. (this should not involve lengthy calculations).

2. [**5pt**] Put the resulting solution back into the classification (margin) constraints and express the result in terms of a linear combination of the radial basis kernels.

3. [**5pt**] Indicate briefly how we can use the following Michelli theorem to show that any n by n kernel matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j = 1, \ldots, n$ is invertible.

   Theorem: *If $\rho(t)$ is monotonic function in $t \in [0, \infty)$, then the matrix $\rho_{ij} = \rho(\|\mathbf{x}_i - \mathbf{x}_j\|)$ is invertible for any distinct set of points $\mathbf{x}_i$, $i = 1, \ldots, n$.*

4. [**10pt**] Based on the above results put together the argument to show that we can indeed find a solution where all the points are support vectors.

   Of course the fact that we can in principle separate any set of training examples does not mean that our classifier does well (on the contrary). So, why do we use the radial basis kernel? The answer is given by the more refined notions of VC-dimension such as the $V(\gamma)$ dimension. This refined notion of complexity takes into account the margin that we achieve for the linear classifier in the feature space. In other words, $V(\gamma)$ is defined as the number of points that we can shatter with *margin* γ. γ here is the distance of the closest training point to the linear boundary in the feature space (the distance is measured in the feature space).

   This is useful in our case here since we know that if the feature vectors lie within a sphere of radius R then $V(\gamma) \leq R^2/\gamma^2$. The larger the margin requirement, the fewer points we can shatter. This is particularly convenient for radial basis kernels since

   $$\phi(\mathbf{x})^T \phi(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}) = 1$$

   In other words, the feature vectors have norm exactly one. If we now use a radial basis kernel and look for a classifier that separates the training points with margin γ what is the $V(\gamma)$ dimension of our classifier as a function of γ?

5. [**10pt**] Show a one dimensional example where adjusting the kernel width makes a difference in terms of the margin that we can achieve for the points.