CS 391L: ML

Homework 2 - Theory

(1) Let's assume S is linearly seperable, which means there exists w such that $w \cdot x_i \cdot y_i > 0$ for all (x_i, y_i) in S.

For the Perceptron algorithm, the weight update rule is $w \leftarrow w + \eta \cdot x_i \cdot y_i$ for a misclassified example (x_i, y_i).

Now, consider a different order of the training examples, denoted as S':

$$S' = (x_{\pi(1)}, y_{\pi(1)}), (x_{\pi(2)}, y_{\pi(2)}), \ldots, (x_{\pi(n)}, y_{\pi(n)})$$

Where π is a permutation function that shuffles the indices of the training examples.

Let w' be the weights obtained by running the Perceptron algorithm on S' with the same initial weights as w.

Now, for each misclassified example (x_i, y_i) in S', the weight update rule can be written as:

$$w' \leftarrow w' + \eta \cdot x_i \cdot y_i$$

Since w' and w are updated using the same examples and the same learning rate, the weight updates will be the same regardless of the order of the examples.

Therefore, both w and w' will converge to the same set of weights that correctly classify all examples in S and S', respectively.

Hence, the Perceptron algorithm makes the same number of mistakes and ends up with the same final weights, regardless of the order of the training examples, if the training set is linearly seperable.

(2)

$\phi(z) = \max(0, -z)$ is known as hinge loss function which penalizes misclassifications and encourages correct classification with a margin.

Let's consider running stochastic gradient descent (SGD) to minimize following function.

$$\frac{1}{m} \sum_{i=1}^{m} \phi(y^i \cdot w^T x^i)$$

where $S = \{(x^1, y^1), \ldots, (x^m, y^m)$ is the training set, $x^i \in \mathbb{R}^n$ is the feature vector and $y^i \in -1, 1$ is the corresponding label.

To relate this to the Perceptron algorithm, let's consider the gradient of the hinge loss function:

$$\nabla \phi(z) = \begin{cases} -y & \text{if } z < 0 \\ 0 & \text{if } z \geq 0 \end{cases}$$

Since we are ignoring the discontinuity at $\phi(0)$ as stated in the note of problem, which simplifies to

$$\nabla \phi(z) = -y \text{ if } z < 0$$

The SGD update rule for the i-th example is given by:

$w_{new} = w_{old} - \eta \nabla \phi(y^i \cdot w^T x^i)$

Substituting the gradient, we have:

$$w_{new} = \begin{cases} w_{old} + \eta y^i x^i & \text{if } y^i \cdot w^T x^i < 0 \\ w_{old} \end{cases}$$

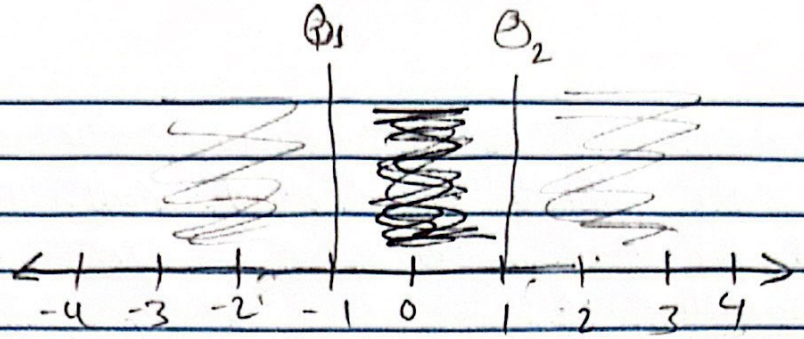
Now, let's compare this with the update rule of the Perceptron algorithm.

$$w_{new} = \begin{cases} w_{old} + \eta y^i x^i & \text{if } y^i \cdot w^T x^i < 0 \\ w_{old} & \text{if } y^i \cdot w^T x^i \geq 0 \end{cases}$$

We can observe that the update rules of SGD with the hinge loss and the Perceptron algorithm are the same. The only difference lies in the interpretation of the algorithms. In the Perceptron algorithm, the weights are updated as long as there is a misclassification, while in SGD with the hinge loss, the weights are updated only when the predicted label and the true label have opposite signs and the margin is violated.

Therefore, we can conclude that running SGD to minimize the hinge loss function is equivalent to running the perceptron algorithm. Both algorithms aim to find a weight vector that correctly classifies the training examples and minimizes the misclassification errors.

3
(a)



To show that for any distribution on $\mathbb{R}$ and any unknown labeling function $c \in C$ (a 3-piece classifier), there exists a decision stump $h \in H$ with error at most $\frac{1}{3}$, we can use a probabilistic argument.

Let's consider a distribution on $\mathbb{R}$ with finite support bounded within $[-B, B]$ for some large B. We want to find a decision stump $h_{\theta,b}$ such that $\mathbb{P}[h_{\theta,b}(x) \neq c(x)] \leq \frac{1}{3}$.

Since c is a 3 piece classifier, it partitions the real line into three intervals: I_1, I_2 and I_3.
Let b_1, b_2 and b_3 denote the labels assigned by c to the intervals I_1, I_2 and I_3 respectively.

Now, consider the decision stump $h_{\theta,b}$, where θ is chosen uniformly at random from the interval $[-B, B]$ and $b \in -1, 1$ is chosen randomly with equal probability. The decision stump $h_{\theta,b}$ assign the label b to all points less than or equal to θ and the label $-b$ to points greater than θ.
Let's calculate the probability of misclassification by $h_{\theta,b}$:
$\mathbb{P}[h_{\theta,b}(x) \neq c(x)] = \mathbb{P}[h_{\theta,b}(x) \neq b_i \text{ for any } i] =$
$\mathbb{P}[x \in I_1 \text{ and } h_{\theta,b}(x) \neq b_1] + \mathbb{P}[x \in I_2 \text{ and } h_{\theta,b}(x) \neq b_2]$
$+ \mathbb{P}[x \in I_3 \text{ and } h_{\theta,b}(x) \neq b_3] \leq \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = \frac{1}{3}$

The inequality holds because for each interval I_i, the probability that θ falls within I_i is $\frac{1}{3}$ and for that interval, $h_{\theta,b}$ agrees with c with probability 1.

Therefore, we have shown that for any distribution on $\mathbb{R}$ and any unknown labeling function c that is a 3 piece classifier, there exists a decision stump $h_{\theta,b}$ with error at most $\frac{1}{3}$ i.e., $P[h(x) \neq c(x)] \leq \frac{1}{3}$.

(b)
- Initialize the best error to infinity : best_error := ∞
- Iterate over all possible thresholds θ in the training sets and all possible labels $b \in -1, 1$
  - Compute error on train set for decision stump $h_{\theta,b}$.
  - If error is smaller than best error, update best error and store corresponding decision stump as the best stump so far.
  - Return best decision stump found.

For each possible threshold θ in the training set and each possible label $b \in -1, 1$, we construct the decision stump $h_{\theta,b}$.

To compute the error on the training set for a decision stump $h_{\theta,b}$, we compare the predicted labels $h_{\theta,b}(x)$ with true labels y for each training example (x, y). The error is the fraction of training examples for which the predicted label does not match the true label.

we keep track of the best error found so far and upade it whenever a decision stump with a lower error is discovered.
Finally, we return the decision stump with the lowest error as the ERM decision stump.

(c) The concept class C consists of 3 piece classifiers, which are relatively complex functions defined on the real line. On the other hand, the class H of decision stumps is a simpler subclass of C that only partitions the real line based on single threshold.
When we move from requiring a learner for C to requiring a learner for H, we simplify the hypothesis space. Decision stumps have a more restricted representation power compared to 3-piece classifiers. By limiting the complexity of the hypothesis class, we introduce a bias that can aid in generalization.

Intuitively, we can expect that by increasing the size of the traning set m, we can better approximate the true error of the concept class C.
This is because increasing the number of training examples allows the learner to observe more instances from the domain, providing more information to learn the underlying patterns and structure of the data.

By using decision stumps from H as weak learners, we are effectively approximating the complex 3 piece classifiers in C using simpler functions.
As the traning set size m increases, the learner has a better chance of finding a decision stump that can

Capture the important discriminative features of the data.

In other words, the larger m becomes the more likely we are to find a decision stump in H that aligns well with the true underlying concept in C. As a result, the training error of the decision stump becomes a good approximation of the true error, allowing for reasonable generalization to unseen data.

By weakly learning C using H, we leverage the simplicity and efficiency of decision stumps to approximate the more complex 3 piece classifiers. The ability to approximate C using H and achieve good generalization with a sufficiently large training set demonstrates the usefulness of weak learners in learning complex concept classes.

(4) In the AdaBoost algorithm, the distribution D_t is updated at each iteration t based on the accuracy of the current weak classifier h_t.

Let's assume we have obtained classifier h_t at iteration t. We want to show that with respect to the distribution D_{t+1} generated for the next iteration, the accuracy h_t is exactly $\frac{1}{2}$.

Recall that the distribution D_{t+1} is updated using the following formula:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t y_i h_t(x_i)\right),$$

where Z_t is a normalization factor and α_t is the weight associated with classifier h_t.

To determine the accuracy of h_t with respect to D_{t+1}, we need to compute the weighted sum of misclassifications. Let $I(h_t(x_i) \neq y_i)$ be the indicator function that evaluates to 1 if h_t misclassifies examples i and 0 otherwise. Then the weighted sum of misclassifications is given by:

$$\sum_{i=1}^{N} D_{t+1}(i)\, I\left(h_t(x_i) \neq y_i\right)$$

Substituting the expression for D_{t+1} we have:

$$\sum_{i=1}^{N} \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t y_i h_t(x_i)\right) I\left(h_t(x_i) \neq y_i\right)$$

Now, notice that if $h_{(t)}(x_i) = y_i$, then $I(h_t(x_i) \neq y_i) = 0$
Therefore, the sum reduces to:

$$\sum_{h_t(x_i) \neq y_i} \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t y_i h_t(x_i)\right)$$

Since $h_t(x_i)$ and y_i have opposite signs when h_t misclassifies example i, we can write $h_t(x_i) y_i = -1$
Substituting this into the sum, we get:

$$\sum_{h_t(x_i) \neq y_i} \frac{D_t(i)}{Z_t} \cdot \exp(\alpha_t)$$

Recall that the update rule for α_t is $\alpha_t = \frac{1}{2} \ln\left(\frac{1-err_t}{err_t}\right)$, where err_t is the weighted error of classifier h_t.
Since the weighted error is equal to the sum above divided by Z_t, we can rewrite it as:

$$err_t = \frac{\sum_i h_t(x_i) \neq y_i D_t(i)}{Z_t}$$

Substituting this into the expression, we have:

$$\sum_{h_t(x_i) \neq y_i} \frac{D_t(i)}{Z_t} \cdot \exp(\alpha_t) = err_t \cdot \exp(\alpha_t)$$

Using the definition of α_t we can rewrite it as:

$$err_t \cdot \exp(\alpha_t) = err_t \cdot \frac{1-err_t}{err_t} = 1 - err_t$$

Therefore, the weighted sum of misclassification is equal to $1 - err_t$.

Since the distribution D_{t+1} is normalized such that $\sum_{i=1}^{N} D_{t+1}(i) = 1$, the accuracy of classifier h_t with respect to D_{t+1} is given by:

$$1 - (1 - err_t) = err_t$$

Since err_t is the weighted error of classifier h_t, the accuracy of h_t with respect to D_{t+1} is exactly $\frac{1}{2}$.