Assignment 2

CSE 353 : Machine Learning

I: Logistic Regression and Naïve Bayes **15 points**

Suppose in a binary classification problem, the input variable $\mathbf{x}$ is n-dimensional and the ouput is a binary class label $y \in \mathcal{Y} = \{0, 1\}$. In this situation, there is an interesting connection between two learners: *logistic regression* and *naïve Bayes classifier*.

(a) Write down the expressions for the class conditional probability for each class, i.e., $P(y = 1|\mathbf{x})$ and $P(y = 0|\mathbf{x})$, for logistic regression.

$$P(y = 1|\mathbf{x}) = \frac{1}{1+e^{-<w,x>}}$$
$$P(y = 0|\mathbf{x}) = \frac{e^{-<w,x>}}{1+e^{-<w,x>}}$$

(b) Using Bayes' rule, derive the posterior probabilities for each class, i.e., $P(y = 1|\mathbf{x})$ and $P(y = 0|\mathbf{x})$, for naïve Bayes.

$$P(y = 1|x) = \frac{P(x|y=1)P(y=1)}{P(x)} \; P(y = 0|x) = \frac{P(x|y=0)P(y=0)}{P(x)}$$

(c) Assuming a Gaussian likelihood function in each of the n dimensions, write down the full likelihood function $f(\mathbf{x}|d)$ for naïve Bayes.

$$f(x|d) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

(d) Assuming a uniform prior on the two classes and using the results from parts (b) and (c) above, derive a full expression for $P(y = 1|\mathbf{x})$ for naïve Bayes.

$$P(y = 1|x) = \frac{P(x|y=1)P(y=1)}{P(x)} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

(e) Show that with appropriate manipulation and parameterization, $P(y = 1|\mathbf{x})$ in naïve Bayes from part (d) is equivalent to $P(y = 1|\mathbf{x})$ for logistic regression in part (a).

II: Boosting Confidence **10 points**

Let A be an algorithm for which there exists a constant $\delta_0 \in (0, 1)$ and a function $m_{\mathcal{H}} : (0, 1) \to \mathbb{N}$ such that for every $\epsilon \in (0, 1)$, if $m > m_{\mathcal{H}}(\epsilon)$, then for every distribution $\mathcal{D}$ it holds that with probability at least $1 - \delta_0$, $L_{\mathcal{D}}(A(S)) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$.

Now, consider the following steps:

(i) Divide the data into $K+1$ chunks where each of the first k chunks consists of $m_{\mathcal{H}}(\epsilon)$ examples.
(ii) Traing the first k chunks using A.
(iii) Use the last chunk to choose from the k hypotheses that A generated from the k chunks.

Show that this 3-step procedure of using A for agnostic PAC-learning of $\mathcal{H}$ has a sample complexity:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq k \cdot m_{\mathcal{H}}(\epsilon) + \left\lceil \frac{2\log(4k/\delta)}{\epsilon^2} \right\rceil \qquad \text{where } k = \lceil \log(\delta)/\log(\delta_0) \rceil$$

Hint: *Use Corollary 4.6 from the textbook.*

If we divide the data into $K+1$ chunks of size $m_{\mathcal{H}}(\epsilon)$ where each is trained using our learning algorithm, we require $k \cdot m_{\mathcal{H}}(\epsilon)$ samples to ensure we obtain k independent hypotheses that are not based on the the same sample set.
With $|\mathcal{H}|$ hypotheses, we can apply Corollary 4.6 from the textbook, which states that for a finite hypothesis class $\mathcal{H}$, domain Z, and a loss function $l : \mathcal{H} \times Z \to [0, 1]$, the class is PAC learnable with sample complexity $m_{\mathcal{H}}(\epsilon, \delta) \leq \lceil \frac{2log(2|\mathcal{H}|/\delta)}{\epsilon^2} \rceil$.
If we apply this to the size-k set of hypotheses generated by our k chunks, we determine: $m_{\mathcal{H}}(\epsilon, \delta) \leq \lceil \frac{2log(2k/\delta)}{\epsilon^2} \rceil$
In order to obtain a fair hypothesis independent of the training samples for any of the chunks, we can upper bound the sample comlexity of this system:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq k \cdot m(\epsilon) + \lceil \frac{2log(2k/\delta)}{\epsilon^2} \rceil$$

III: AdaBoost Weights **10 points**

In the lectures, we discussed (informally) how the weighting mechanism of AdaBoost helps the learner to focus on the problematic examples in the next iteration. Here, you have to formally argue for it:

Show that the error h_t w.r.t. $\mathbf{D}^{(t+1)}$ is exactly 0.5. That is, show that

$$\forall\, t \in [T] \quad \sum_{i \in [m]} D_i^{(t+1)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]} = 0.5$$

The update step for adaptive boosting applies the following:

$$D_i^{t+1} = \frac{D^t e^{-w_t y_i h(x_i)}}{\sum_{j=1}^m D_j^t e^{-w_t y_j h(x_j)}} \text{ where } w_t = \frac{1}{2} log(\frac{1}{\epsilon_t} - 1), \epsilon = \sum D_i^t \mathbb{1}_{[y_i \neq h(x_i)]}$$

so we must show:

$$\sum_{i \in [m]} \frac{D_i^t e^{-w_t y_i h(x_i)}}{\sum_{j=1}^m D_j^t e^{-w_t y_j h(x_j)}} = 0.5$$

IV: Failure of Cross-Validation **10 points**

Cross-validation works well in practice, but there are some pathological cases where it might fail. Suppose that the label is chosen at random according to $P[y = 1] = P[y = 0] = 1/2$. Consider a learning algorithm that outputs the constant predictor $h(x) = 1$ if the number of 1s in the training set labels is odd, and $h(x) = 0$ otherwise. Prove that the difference between the leave-one-out estimate and the true error in such a case is always $1/2$.

Honestly I am unable to understand how the difference between the leave-one-out error and the true error are $\frac{1}{2}$. Cross-validation doesn't work because simply adding or removing a training sample from the traning sample set will result in a different hypothesis. Furthermore, only a constant hypothesis of $h(x) = 1$ or $h(x) = 0$ is produced. SInce the labelling function is $P(1) = P(0) = \frac{1}{2}$, the true error of the hypothesis selected by validation will always be $\frac{1}{2}$. Assuming the sample is drawn iid, the empirical loss of the sample will also be $\frac{1}{2}$.

V: Local Minima of 0-1 Loss **10 points**

Here, you are being asked to construct an example where 0-1 loss suffers from local minima.

Construct a training sample $S \in (\mathbb{R}^d \times \{\pm 1\})^m$ for which there exists a vector $\mathbf{w}$ and some positive

scalar ϵ such that:

(a) $\forall\ \mathbf{w}'$ such that $||\mathbf{w} - \mathbf{w}'|| \leq \epsilon$ we have $L_S(\mathbf{w}) \leq L_S(\mathbf{w}')$ (with 0-1 loss). This means that $\mathbf{w}$ is a local minimum of L_S.

(b) $\exists\ \mathbf{w}^*$ such that $L_S(\mathbf{w}) < L_S(\mathbf{w})$. This means that $\mathbf{w}$ is not a global minimum of L_S.

A function will have non-global local minima if it is non convex - e.g. if we consider the a hypothesis class of 1-dimensional halfspaces trying to classify points in a distribution that follows the form:

$$f(x) = 1 \text{ if } a < x < b$$

$$f(x) = 0 \text{ otherwise}$$

.

So a more specific example would be a uniform distribution of points from 0 to 1 with labelling function $f(x) = 1$ if $\frac{1}{6} < x < \frac{3}{6}$, else 0. Given a sufficiently sized representative sample drawn from distribution, there exists no way of dividing the domain into two halfspaces such that all minima are global.

VI: Bounded Logistic Regression is Lipschitz and Smooth **20 points**

Show that logistic regression with the bounded hypothesis class $\mathcal{H} = \mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x}|| \leq B\}$ (where B is a positive scalar), the label set $\mathcal{Y} = \{\pm 1\}$, and the loss function $l(\mathbf{w}, (\mathbf{x}, y)) = \log(1 + \mathrm{e}^{-y\langle \mathbf{w}, \mathbf{x}\rangle})$, is convex, Lipschitz, and smooth. For Lipschitzness and smoothness, specify the parameters as well.

By claim 12.4 in the textbook, for a function $f : \mathbb{R}^d \to \mathbb{R}$ that can be written as $f(w) = g(< w, x > +y)$ for $x \in \mathbb{R}^d, y \in \mathbb{R}, g : \mathbb{R} \to \mathbb{R}$ - if g is convex, f is convex.
In this case, $g(a) = log(1 + e^a)$, $f = g(-y < w, x >)$, so the function is convex.

In order to show that the function is Lipschitz, we need to show

$$\forall w_1, w_2, \exists \rho > 0, ||f(w_1) - f(w_2)|| \leq \rho ||w_1 + w_2||$$

$$||log(1 + e^{-y<w_1,x>}) - log(1 + e^{-y<w_2,x>})|| \leq \rho ||w_1 + w_2||$$

$$log(\frac{1 + e^{-y<w_1,x>}}{1 + e^{-y<w_2,x>}}) \leq \rho ||w_1 + w_2||$$

$$log(\frac{1}{1 + e^{-y<w_2,x>}} + \frac{e^{-y<w_1,x>}}{1 + e^{-y<w_2,x>}}) \leq \rho ||w_1 + w_2||$$

Each term inside the log of the right side is bounded by 1, so:

$$log(\frac{1}{1 + e^{-y<w_2,x>}} + \frac{e^{-y<w_1,x>}}{1 + e^{-y<w_2,x>}}) \leq log(2)$$

$$\frac{log(\frac{1}{1+e^{-y<w_2,x>}} + \frac{e^{-y<w_1,x>}}{1+e^{-y<w_2,x>}})}{||w_1 + w_2||} \leq \frac{log(2)}{||w_1 + w_2||}$$

Since $||w_1 + w_2|| \leq < u, u > \cdot < v, v > \leq B * B$, we claim the function is $\frac{log(2)}{B^2}$-Lipschitz. Since a function that is σ-Lipschitz is σ-smooth, we claim it os also $\frac{log(2)}{B^2}$-smooth.

VII: Unsupervised Learning without Uniform Convergence **25 points**

Suppose our hypothesis class $\mathcal{H}$ is $\mathcal{B}$, the unit ball in $\mathbb{R}^d$, the domain is $Z = \mathcal{B} \times \{0, 1\}^d$, and

the loss function $l : Z \times \mathcal{H} \to \mathbb{R}$ is defined as $l(\mathbf{w}, (\mathbf{x}, \mathbf{a})) = \sum_{i \in [d]} a_i (x_i - w_i)^2$. This problem corresponds to an **unsupervised learning** task, meaning that we do not try to predict the label of $\mathbf{x}$. Instead, we try to find the *center of mass* of the distribution over $\mathcal{B}$. The vector $\mathbf{a}$ plays an interesting role: in each pair $(\mathbf{x}, \mathbf{a})$, it indicates which features of $\mathbf{x}$ are turned on (value is 1) and which ones are turned off (value is 0). A hypothesis is a vector $\mathbf{w}$ representing the center of mass of the distribution, and the loss function is the squared Euclidean distance between $\mathbf{x}$ and $\mathbf{w}$, but only with respect to the elements of $\mathbf{x}$ that are turned on.

(i) Show that this problem is learnable using the RLM rule with a sample complexity that does not depend on d.

Applying the RLM rule means $argmin_w(R(w)+L_S(w))$ or in this case $argmin_w(R(w)+\sum_{i \in [d]} a_i(x_i - w_i)^2)$. Since we know that the hypothesis class refers to the unit ball in d dimensions, regardless of the number of dimensions, we control the total distance using the L-2 norm.

(ii) Consider a distribution $\mathcal{D} \sim Z$ as follows: $\mathbf{x}$ is fixed to be some $\mathbf{x}_0$, and each element of $\mathbf{a}$ is sampled to be either 1 or 0 with equal probability.

(a) Show that if $d \gg 2m$, then there is a high probability of sampling a set of examples such that there exists some $j \in [d]$ for which $a_j = 0$ for all the examples in the training set.

The probability across m d-dimensional vectors that a feature will be 0, given that the probability any feature will be 0 is $P(0) = (0.5)^m * d$. As d increases, the greater the chances that a feature will be set to 0. Meanwhile, as m increases, the chances a feature for all samples will be set to 0 goes to 0.

(b) Show that such a sample is not ϵ-representative.

A training set is ϵ representative with respect to a domain, hypothesis class, loss function, and distribution if: $\forall h \in \mathcal{H}, |L_S(h) - L_D(h)| \leq \epsilon$. This sampling is not ϵ representative because it represents all unit balls, not all of which may contain any of the points in the sample set. In which case, the accuracy of an infinite number of hypotheses in $\mathcal{H}$ is 0.

(c) Conclude that the sample complexity of uniform convergence must grow with d.

As d increases, the loss function increases since we add distances between feature values across all dimensions. The more dimensions there are, the greater the individual errors are for each data point and more. As a result, greater sample complexity is required.

(iii) If d is taken to infinity, show that we obtain a problem that is learnable but for which the uniform convergence property does not hold.

If we take d to infinity, we cannot calculate the loss using the Euclidean distance from the center of the ball. However, it is learnable with some degree of confidence as long as one uses an appropriate loss function, perhaps by using a vector's euclidean distance from B in a sufficiently large number of dimensions to determine the loss.